UNIVERSIDADE FEDERAL FLUMINENSE

MARCELO RODRIGUES DE HOLANDA MAIA

NOVEL DATA MINING METHODS AND APPLICATIONS IN COMBINATORIAL OPTIMIZATION AND BIOINFORMATICS

NITERÓI

2022

MARCELO RODRIGUES DE HOLANDA MAIA

NOVEL DATA MINING METHODS AND APPLICATIONS IN COMBINATORIAL OPTIMIZATION AND BIOINFORMATICS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação.

Orientador:
ALEXANDRE PLASTINO DE CARVALHO

Coorientador:
UÉVERTON DOS SANTOS SOUZA

NITERÓI

2022

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

M217n Maia, Marcelo Rodrigues de Holanda
Novel Data Mining Methods and Applications in Combinatorial Optimization and Bioinformatics / Marcelo Rodrigues de Holanda Maia ; Alexandre Plastino de Carvalho, orientador ; Uéverton dos Santos Souza, coorientador. Niterói, 2022.
181 f.

Tese (doutorado)-Universidade Federal Fluminense, Niterói, 2022.

DOI: http://dx.doi.org/10.22409/PGC.2022.d.10647433745

1. Mineração de dados. 2. Otimização combinatória. 3. Biologia computacional. 4. Produção intelectual. I. Carvalho, Alexandre Plastino de, orientador. II. Souza, Uéverton dos Santos, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

MARCELO RODRIGUES DE HOLANDA MAIA

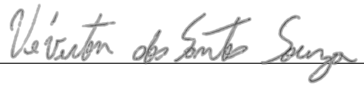
NOVEL DATA MINING METHODS AND APPLICATIONS IN COMBINATORIAL OPTIMIZATION AND BIOINFORMATICS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Doutor em Computação. Área de concentração: Ciência da Computação.

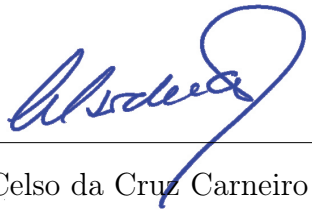
Aprovada em abril de 2022.

BANCA EXAMINADORA

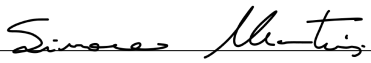
Prof. Alexandre Plastino de Carvalho
Orientador – UFF



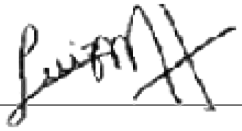
Prof. Uéverton dos Santos Souza
Coorientador – UFF



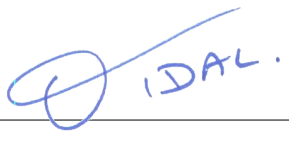
Prof. Celso da Cruz Carneiro Ribeiro
UFF



Profª. Simone de Lima Martins
UFF



Prof. Luiz Henrique de Campos Merschmann
UFLA



Prof. Thibaut Victor Gaston Vidal
Polytechnique Montréal

Niterói
2022

Acknowledgements

Throughout the development of this work, I have received support and assistance from several parties.

This research was financed in part by: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) [finance code 001]; and Instituto Brasileiro de Geografia e Estatística (IBGE). It was enabled in part by computational resources and support provided by Calcul Québec (www.calculquebec.ca), Compute Canada (www.computecanada.ca) and UFF's Laboratório de Inteligência Computacional (LabIC). This support is gratefully acknowledged.

I would like to thank Professors Alexandre Plastino and Uéverton Souza, my doctoral supervisors at UFF, and Professor Alex Freitas, who supervised me during my research visit to the University of Kent. Their expertise was crucial in formulating research questions and methodology, and their insightful feedback helped me bring my work to a much higher level.

I would also like to acknowledge other researchers who contributed to or supported this work. Professor Puca Huachi Vaz Penna from UFOP was a collaborator in the proposal of the MineReduce approach and its application to the HFVRP. Professor Igor Machado Coelho from UFF was very helpful in providing access to LabIC's computational resources and support for their use. Professor Thibaut Vidal from Polytechnique Montréal granted access to computational resources of Calcul Québec/Compute Canada and provided helpful tips on how to use them. Professor Thibaut Vidal also contributed valuable insights to build upon his HGS-CVRP metaheuristic. Professor João Pedro de Magalhães from the University of Liverpool provided a domain-expert analysis of results on the genetics of ageing.

Abstract

Data mining is a process that has been applied in several fields, aiming at discovering interesting and potentially useful knowledge from data in the form of models and patterns. This thesis compiles a series of research contributions established by the author's work on the proposal and application of data mining methods in the combinatorial optimization and bioinformatics fields.

Data mining techniques have been incorporated into metaheuristics in the combinatorial optimization literature, leading to solution quality improvement and convergence speedup. This thesis explores a novel approach, named MineReduce, for incorporating data mining into metaheuristics, which was applied in this work to solve variants of the vehicle routing, vertex cover, facility location and travelling salesperson problems. The heuristics developed based on the MineReduce approach achieved relevant results, overcoming previous state-of-the-art algorithms. Additionally, this thesis presents the application of a previous data mining approach – the Multi Data Mining (MDM) approach – to solve a vehicle routing problem variant, which also achieved state-of-the-art results.

Classification is a data mining task based on building, from a dataset of labelled records, a model capable of classifying unlabelled data records. This thesis reports methodological contributions related to classification and their applications in bioinformatics. It introduces novel approaches to build classifier ensembles for uncertain data. These proposals were applied for classifying ageing-related genes and predicting drug side effects using real data. The experimental results produced evidence that the proposed approaches can improve the predictive performance of ensembles on uncertain data. Additionally, this thesis introduces novel approaches for interpreting Naive Bayes ensembles, which were applied to identify relevant protein interactions to classify ageing-related genes, producing results consistent with current biological knowledge and new insights in this field.

Keywords: Data mining. Metaheuristics. Problem size reduction. Bioinformatics. Classification. Uncertain data.

Resumo

Mineração de dados é um processo que tem sido aplicado em diversas áreas, com o objetivo de descobrir conhecimento novo e potencialmente útil a partir de dados na forma de modelos e padrões. Esta tese compila uma série de contribuições estabelecidas pelo autor na proposta e aplicação de métodos de mineração de dados nas áreas de otimização combinatória e bioinformática.

Técnicas de mineração de dados têm sido incorporadas em metaheurísticas na literatura de otimização combinatória, levando à melhoria da qualidade das soluções e à aceleração da convergência. Esta tese explora uma nova abordagem, chamada MineReduce, para incorporar mineração de dados em metaheurísticas, que foi aplicada para resolver variantes dos problemas de roteamento de veículos, cobertura de vértices, localização de instalações e caixeiro viajante. As heurísticas desenvolvidas com base na abordagem MineReduce alcançaram importantes resultados, superando algoritmos do estado-da-arte. Adicionalmente, esta tese apresenta a aplicação de uma abordagem anterior de mineração de dados – a estratégia *Multi Data Mining* (MDM) – para resolver uma variante do problema de roteamento de veículos, que também alcançou resultados em nível de estado-da-arte.

Classificação é uma tarefa de mineração de dados baseada na construção, a partir de um conjunto de registros rotulados, de um modelo capaz de classificar registros não rotulados. Esta tese relata contribuições metodológicas relacionadas à classificação e suas aplicações em bioinformática. Ela apresenta novas abordagens para construir comitês de classificadores para dados incertos, que foram aplicadas para classificar genes relacionados ao envelhecimento e prever efeitos colaterais de drogas, usando dados reais. Os resultados evidenciam que as abordagens propostas melhoram o desempenho preditivo de comitês ao lidar com dados incertos. Além disso, esta tese apresenta novas abordagens para interpretar comitês de Naive Bayes, que foram aplicadas para identificar interações entre proteínas relevantes para classificar genes relacionados ao envelhecimento, produzindo resultados consistentes com o conhecimento biológico atual e novos *insights* neste campo.

Palavras-chave: Mineração de dados. Metaheurísticas. Redução de tamanho de problema. Bioinformática. Classificação. Dados incertos.

Contents

1 Introduction

Data mining is a process that has been applied in several fields, aiming at discovering new and potentially useful knowledge from data in the form of models and patterns (WITTEN; FRANK; HALL, 2011; HAN; KAMBER; PEI, 2012; ZAKI; MEIRA JR, 2014). It can be categorized into predictive and descriptive data mining. Predictive models provide inferences and predictions on the data domain, whereas descriptive patterns reveal implicit relations and hidden rules in the data. Classification and regression are examples of predictive data mining tasks. Frequent itemsets, association rules and data clusters are examples of descriptive patterns. This thesis compiles a series of research contributions established by the author's work on the proposal and application of predictive and descriptive data mining methods in the combinatorial optimization and bioinformatics fields.

Research on the synergies between machine learning (ML) – as a broader category that includes data mining – and metaheuristics has become increasingly popular in the combinatorial optimization literature (JOURDAN; DHAENENS; TALBI, 2006; ZHANG et al., 2011; CORNE; DHAENENS; JOURDAN, 2012; CALVET et al., 2017; MARTINS; ROSSETI; PLASTINO, 2018; TALBI, 2021). Metaheuristics can help solve ML problems that can be modelled as optimization problems, and ML-based approaches have been applied to improve the performance of metaheuristics. One particularly successful approach consists in incorporating data mining techniques into metaheuristics, leading to solution quality improvement and convergence speedup (RIBEIRO; PLASTINO; MARTINS, 2006; SANTOS; MARTINS; PLASTINO, 2008; GUERINE; ROSSETI; PLASTINO, 2014; PLASTINO et al., 2014; MARTINS; ROSSETI; PLASTINO, 2018; MAIA; PLASTINO; PENNA, 2018).

This thesis introduces and explores a novel approach, named MineReduce, for incorporating data mining into metaheuristics (MAIA; PLASTINO; PENNA, 2020), presenting its well-succeeded application to solve variants of the vehicle routing, vertex cover, facility location and travelling salesperson problems. The heuristics developed based on the MineReduce approach achieved relevant results, overcoming previous state-of-the-art algorithms. Additionally, this thesis presents the application of a previous hybrid data

mining approach – the Multi Data Mining (MDM) approach (PLASTINO et al., 2014) – to solve a vehicle routing problem variant, which also achieved state-of-the-art results. The MineReduce and MDM approaches are based on frequent itemset mining, a descriptive data mining task.

Classification is a predictive data mining task based on building, from a dataset of labelled records, a model capable of classifying unlabelled data records. This thesis reports methodological contributions related to classification and their applications in bioinformatics. Specifically, it introduces novel approaches for coping with uncertain data and for model interpretation.

This work addresses feature-value uncertainty, which occurs when some feature values in an instance are not precisely known. It has been shown that incorporating information on uncertainty into classification algorithms can improve predictive performance (GE; XIA; NADUNGODAGE, 2010; TSANG et al., 2011; ANGIULLI; FASSETTI, 2013; XIE; XU; HU, 2018), but this is still an under-explored research topic, particularly for categorical features, since most previous methods focus on uncertain numerical features. This thesis introduces novel approaches for building classifier ensembles that cope with uncertain categorical features. These proposals have been applied for classifying ageing-related genes and predicting drug side effects using real data (KUHN et al., 2015; SZKLARCZYK; SANTOS, et al., 2015; TACUTU et al., 2017; SZKLARCZYK; GABLE, et al., 2018). The experimental results showed that the proposed approaches improved the predictive performance of ensembles on uncertain data.

Interpretable modelling is an emerging topic of high importance in the data mining field since understanding how and why computational models make predictions may be critical for some applications (GUIDOTTI et al., 2018). Besides, it can help domain experts learn more about the considered problem. In particular, interpretable models have been applied in genetics and genomics to derive novel biological insights (AZODI; TANG; SHIU, 2020). This thesis introduces novel approaches for interpreting Naive Bayes ensembles, which were applied to identify relevant protein interactions to classify ageing-related genes, producing results consistent with current biological knowledge and new insights in this field.

This thesis consists of seven research papers and this complementary text. This text provides a general overview of this thesis' objectives, results and contributions, whereas the papers, included as appendices, provide detailed descriptions of the proposed methods, applications, experimental evaluations, and specific conclusions. The following papers

compose this thesis:

1. **MAIA, M. R. H.**; PLASTINO, A.; PENNA, P. H. V. "MineReduce: An approach based on data mining for problem size reduction". *Computers & Operations Research*, v. 122, 104995, 2020. (Appendix A).

2. **MAIA, M. R. H.**; PLASTINO, A.; SOUZA, U. S. "MineReduce-based iterated tabu search for the minimum weighted vertex cover problem". Submitted to *Applied Soft Computing*. (Appendix B).

3. **MAIA, M. R. H.**; REULA, M.; PARREÑO-TORRES, C.; VUPPULURI, P. P.; PLASTINO, A.; SOUZA, U. S.; CESCHIA, S.; PAVONE, M.; SCHAERF, A. "Metaheuristic Techniques for the Capacitated Facility Location Problem with Customer Incompatibilities". Submitted to *Soft Computing*. (Appendix C).

4. **MAIA, M. R. H.**; SANTANA, Í.; ROSSETI, I.; SOUZA, U. S.; PLASTINO, A. "MineReduce-based Metaheuristic for the Minimum Latency Problem". Submitted to the 14th Metaheuristics International Conference (MIC'2022). (Appendix D).

5. **MAIA, M. R. H.**; PLASTINO, A.; SOUZA, U. S. "An improved hybrid genetic search with data mining for the CVRP". In: 12th DIMACS Implementation Challenge: Vehicle Routing Problems. (Appendix E).

6. **MAIA, M. R. H.**; PLASTINO, A.; FREITAS, A. A. "An Ensemble of Naive Bayes Classifiers for Uncertain Categorical Data". In: 2021 IEEE International Conference on Data Mining (ICDM). 2021. p. 1222–1227. (Appendix F).

7. **MAIA, M. R. H.**; PLASTINO, A.; FREITAS, A. A.; MAGALHÃES, J. P. "Interpretable Ensembles of Classifiers for Uncertain Data with Bioinformatics Applications". Submitted to *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. (Appendix G).

Paper 1 presents the proposal of MineReduce, an approach based on data mining for problem size reduction in the combinatorial optimization context. The idea of using data mining for this purpose was first introduced in an algorithm specifically designed for the Heterogeneous Fleet Vehicle Routing Problem (HFVRP), proposed in the author's master's thesis (MAIA, 2015). Afterwards, during this research work, that idea originated the general approach named MineReduce formalized in Paper 1, which also reported novel

results for the application of the MineReduce approach to the HFVRP. This paper was published in *Computers & Operations Research*.

Paper 2, which presents an application of the MineReduce approach to the Minimum Weighted Vertex Cover Problem (MWVCP), is currently under review in *Applied Soft Computing*. It is an extended version of a paper published in the proceedings of the 2020 edition of the *International Conference on Optimization and Learning* (MAIA; PLASTINO; SOUZA, 2020).

Paper 3 is a work originated from the Metaheuristics Competition of the *Metaheuristics Summer School* (MESS 2020+1)[1]. The task was to propose metaheuristic methods to solve a new variant of the facility location problem. This thesis' author was the competition gold medal winner with a method based on the MineReduce approach (MR-MS-ILS). The paper, co-authored by the competition's organizers and competing teams, introduces the Multi-Source Capacitated Facility Location Problem with Customer Incompatibilities (MS-CFLP-CI), presenting and comparing the competing methods. It is currently under review in *Soft Computing*.

Paper 4 presents an application of the MineReduce approach for the Minimum Latency Problem (MLP). This paper is currently under review in the 14th edition of the *Metaheuristics International Conference*.

Paper 5 describes a heuristic method proposed for the Capacitated Vehicle Routing Problem (CVRP) track of the 12th DIMACS Implementation Challenge[2]. The proposed method (MDM-HGS) applies an approach referred to as MDM (which stands for *multi data mining*) that uses frequent patterns extracted from good solutions by a data mining process (PLASTINO et al., 2014). The paper was part of the author's entry, which ranked second, and was presented at the challenge's workshop.

Paper 6 presents the proposal of an approach named Biased Random Subspaces (BRS) for building classifier ensembles for uncertain categorical data. It also presents the application of the BRS approach to build ensembles of Naive Bayes classifiers on datasets of ageing-related genes. The paper was published in the proceedings of the 2021 edition of the *IEEE International Conference on Data Mining*.

Paper 7 builds upon the work reported in Paper 6, proposing two new approaches for building classifier ensembles for uncertain categorical data: Biased Bootstrap (BB) and Biased Splitting (BS). It presents the application of the BRS, BB and BS approaches

[1] https://www.ants-lab.it/mess2020/#competition
[2] http://dimacs.rutgers.edu/programs/challenge/vrp/cvrp

to build Random Forests and ensembles of Naive Bayes classifiers on datasets from two bioinformatics domains: ageing-related genes and drug side effects. Furthermore, this paper introduces two approaches for interpreting Naive Bayes classifier ensembles and presents their application to the ageing-related genes datasets. It is currently under review in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

The remainder of this complementary text is organised as follows. Each of Chapters 2–9 discusses one research contribution: the proposal of the MineReduce Approach (Chapter 2) and its application to the HFVRP (Chapter 3), the MWVCP (Chapter 4), the MS-CFLP-CI (Chapter 5), and the MLP (Chapter 6); the application of the MDM approach to the CVRP (Chapter 7); the proposal of classifier ensemble approaches for uncertain categorical data and their application in bioinformatics domains (Chapter 8); and the proposal of interpretability approaches for Naive Bayes ensembles and their application in a bioinformatics domain (Chapter 9). Lastly, Chapter 10 presents general conclusions and future work directions.

2 The MineReduce approach

This chapter discusses MineReduce, an approach based on data mining for problem size reduction in combinatorial optimization. The MineReduce approach has been proposed in (MAIA; PLASTINO; PENNA, 2020), a paper published in *Computers & Operations Research*, presented in Appendix A. Its application to various combinatorial optimization problems, detailed in Chapters 3–6, demonstrated its effectiveness in providing metaheuristic methods with enhanced performance regarding both solution quality and convergence speed. Thus, it constitutes a promising contribution to the combinatorial optimization field. Section 2.1 provides a background on problem size reduction, whereas Section 2.2 describes the MineReduce approach.

2.1 Problem size reduction in combinatorial optimization

Any approach for solving a combinatorial optimization problem (COP) relies on some form of search on its solution space, which is the domain of the function to be optimized. Solving a COP is usually a challenging task because its solution space grows exponentially with its size. Therefore, problem size reduction (PSR) is beneficial in this context, especially in the case of large-scale COPs (GAVISH; PIRKUL, 1985a; GAVISH; PIRKUL, 1985b; GAVISH; SRIKANTH, 1986).

If a COP is analyzed with respect to an integer programming formulation, its size is defined considering the number of decision variables, the cardinality of the decision variables' domains, and the number of constraints. PSR techniques aim to reduce the number of decision variables of the problem or the range of values in their domains.

The general process of PSR techniques consists of:

1. Transforming a problem P into a modified problem P' such that the solution space of P' is smaller than that of P.
2. Solving P'.

3. Transforming the solution to P' into a solution to P.

The application of this procedural framework is exemplified by the Reduce-Optimize-Expand (ROE) method (MONTIEL; DIAZ-DELGADILLO; SEPÚLVEDA, 2013). In the first step (reduce), the ROE method reduces the size of the problem instance just before applying a solving algorithm. Once the reduction is achieved, the next step (optimize) is the application of a combinatorial optimization method, exact or heuristic, to obtain an optimal or suboptimal solution. Then it executes the last step (expand) to obtain the final result.

The "reduce" step is naturally the most crucial procedure in the PSR process since better decisions in this step will produce solutions of better quality in the subsequent stages (MONTIEL; DIAZ-DELGADILLO; SEPÚLVEDA, 2013; MONTIEL; DELGADILLO, 2015; DELGADILLO; MONTIEL; SEPÚLVEDA, 2016). Strategies to make these decisions are highly dependent on the structure and features of the problem, so they vary significantly among distinct classes of COPs.

One general form of accomplishing size reduction is fixing values of decision variables, which can be regarded as fixing elements either in or out of the solution. This kind of reduction is common for many classes of COPs. For example, in binary formulations of classical routing problems – such as the travelling salesperson problem (TSP) or the vehicle routing problem (VRP) – the decision variables refer to edges in the problem instance graph. The value set to a variable indicates whether the corresponding edge is in (one) or out of (zero) the solution.

Work on the multilevel paradigm, which involves the recursive application of the PSR steps, has provided evidence that it can aid metaheuristics to find better solutions faster for some COPs (WALSHAW, 2008). In a multilevel optimization method, the original problem instance is recursively coarsened (reduced), creating a multilevel hierarchy of reductions. An initial solution is found (at the coarsest level) and then, at each level in reverse order, iteratively expanded to a solution to the parent level's instance and refined, usually with a local search algorithm.

Chen (2015) proposed a fix-and-optimize approach for mixed integer programming problems, which has been integrated into a variable neighbourhood search framework. It decomposes the original problem by fixing values of binary variables based on their interrelatedness.

C. Blum et al. (2016) introduced a hybrid metaheuristic called Construct, Merge, Solve

& Adapt (CMSA). It is based on the idea of generating solutions, reducing the original instance by merging the components in the generated solutions (in a way such that a solution to the original instance can be derived from a solution to a reduced instance), solving the reduced instances to optimality, and adapting these reduced instances based on an ageing mechanism.

Kenny et al. (2019) presented another hybrid metaheuristic, based on a strategy known as "merge search", which has some similarities with CMSA. The main difference between them is that, while CMSA generates solutions from scratch, the merge search algorithm starts with a single initial seed solution and uses local search to generate a population of neighbouring solutions to the initial seed solution. Then it goes through an iterative process of generating a population, merging, and solving the reduced instance using an exact method. The merging procedure is based on the intersections between all of the solutions in the population. The solution to a reduced instance becomes the initial seed solution for the next generation. As each iteration produces a new set of solutions, there is no need for an ageing mechanism to regulate the population.

2.2 Approach description

Usually, multi-start metaheuristics perform a sequence of independent iterations composed of a generation phase and a local search phase. The generation phase builds an initial solution and generally consists of applying more straightforward methods, usually based on a combination of greedy and random strategies. Most of the computational effort is employed in the local search phase, which improves the initial solution.

Previous approaches that incorporate data mining into multi-start metaheuristics eliminate the independence of their iterations by introducing a memory mechanism into them (MARTINS; ROSSETI; PLASTINO, 2018). At some point, iterations begin to benefit from knowledge accumulated in previous iterations. This knowledge – patterns mined from an elite set of solutions – is used to generate better initial solutions.

The data mining procedure in these approaches relies on the FPmax* algorithm proposed by Grahne and Zhu (2003), which mines maximal frequent itemsets. An itemset is considered frequent if it achieves a given minimum support, i.e., if it is present in at least a given minimum number of the elite set solutions. Hence, mined patterns are composed of items that frequently appear together in the sub-optimal solutions of the elite set. Intuitively, it is assumed that these items should likely be part of the best solutions to the

problem. Thus, they are included in initial solutions.

The MineReduce approach builds upon these ideas. Since the mined patterns are assumed to likely be part of the best solutions to a problem instance, they could be well-suited for reducing the size of that instance. The items in a pattern could, for example, be fixed in the solution, which in turn would be equivalent to fixing (or at least constraining) values of decision variables related to those items. Another possibility is that the items in a pattern could be merged in a condensed representation, also producing a reduced-size instance.

Fig. 1 presents a generalized conceptual framework of the MineReduce approach, showing the steps that compose its PSR process.

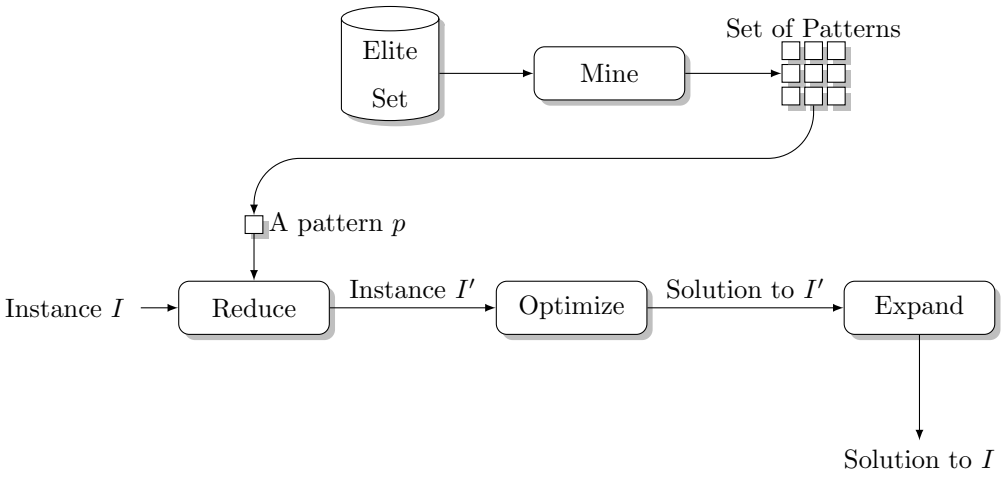


Figure 1: Conceptual framework of the MineReduce approach. Adapted from (MAIA; PLASTINO; PENNA, 2020, p. 3).

The proposed approach requires as input an *elite set* of solutions for the problem instance in consideration. A set of patterns is extracted from the elite set by applying a data mining method. Then, a pattern p is selected for driving the PSR process. The *Reduce* step uses pattern p to transform the problem instance I into a reduced-size instance I'. In the *Optimize* step, a solution to I' is obtained. Finally, in the *Expand* step, the solution to I' is transformed into a solution to I.

The extraction of patterns from an elite set by using a data mining method works exactly like in the *Multi Data Mining* (MDM) approach (PLASTINO et al., 2014), i.e., the best solutions found during the execution of the applied heuristic are stored in the elite set until it becomes stable (unaltered for a given period), which triggers the data mining method.

The three subsequent steps compose a full PSR process conceived to work as a method for generating new solutions. For example, in a multi-start metaheuristic, at each iteration occurring after a set of patterns is mined, one of the patterns would be selected for carrying out the PSR process, producing a new initial solution.

The *Optimize* step may be accomplished through the application of any method that produces a feasible (and presumably good) solution for the reduced instance. For example, the original metaheuristic's optimization procedures (e.g., solution initialization and local search) may be used, or an exact method may be applied to solve the reduced instance optimally if it is small enough.

This modified solution generation method makes MineReduce different from previous approaches incorporating data mining techniques into metaheuristics. Methods based on the MDM approach (or its predecessor, the DM approach) use the mined patterns as a starting point for constructing initial solutions (MARTINS; ROSSETI; PLASTINO, 2018). MineReduce, on the other hand, performs a procedure that reduces the original instance by deleting or merging elements that are in a pattern, finds a solution to the reduced instance and expands the solution found, which will be used as the starting point for a search on the original solution space.

In this sense, the MineReduce approach shares some aspects with the multilevel paradigm (WALSHAW, 2008). MineReduce-based methods reduce the problem instance to a "coarser" level, find a solution to the reduced instance, expand the solution (back to the original instance's level), and refine it. In this case, the number of levels is constant (two), but the main difference resides in the reduction strategy. In the multilevel paradigm, this is often done by adapting construction heuristics, whereas the MineReduce approach relies on patterns extracted from an elite set of solutions through data mining techniques.

Regarding the reduction strategy, the MineReduce approach is closer to merge search algorithms (KENNY et al., 2019), which reduce instances by merging variables based on the intersections observed in a set of solutions. MineReduce's reduction strategy also relies on similarities observed in a set of solutions. However, they differ significantly in how they build such a set of solutions and how similarities between solutions are characterized and identified.

Talbi (2021) has proposed a taxonomy for metaheuristics that incorporate machine learning (ML) – as a broader category that includes data mining – in their design. According to that taxonomy, MineReduce-based methods are primarily classified as *problem-*

level ML-supported metaheuristics since this approach uses data mining for hierarchical problem decomposition (defining and solving smaller subproblems). They can also be classified as *low-level ML-supported metaheuristics* since data mining is used in a process that drives the initialization of solutions. Finally, regarding the *learning time* criteria, they are classified as *online* ML-supported metaheuristics since they gather knowledge during the search while solving the problem.

3 MineReduce application to the Heterogeneous Fleet Vehicle Routing Problem

This chapter summarizes the results obtained by applying the MineReduce approach to the Heterogeneous Fleet Vehicle Routing Problem (HFVRP). A MineReduced-based multi-start iterated local search method was proposed for this problem in (MAIA; PLASTINO; PENNA, 2020), the paper that also proposed the MineReduce approach, published in *Computers & Operations Research* and presented in Appendix A. Section 3.1 describes the HFVRP, Section 3.2 describes how the MineReduce elements have been implemented in the proposed method, and Section 3.3 reports a summary of the obtained results.

3.1 Problem definition

The HFVRP is described as follows. Let $G = (V,A)$ be a directed graph, where $V = \{0,1,...,n\}$ is a set composed of $n+1$ vertices and $A = \{(i,j) : i,j \in V, i \neq j\}$ is the set of arcs. Vertex 0 is the depot, where the vehicle fleet is located, whereas the set $V' = V \backslash \{0\}$ consists of the remaining vertices representing the n customers. Each customer $i \in V'$ is associated with a non-negative demand q_i. The fleet consists of m distinct vehicle types, which compose a set $M = \{1,...,m\}$. For each vehicle type $u \in M$, there are m_u vehicles available, each with a capacity Q_u and a fixed cost f_u. Finally, for each combination of an arc $(i,j) \in A$ and a vehicle type $u \in M$, there is an associated cost $c_{ij}^u = d_{ij} r_u$, where d_{ij} is the distance between vertices i and j, and r_u is the dependent (variable) cost per unit distance associated with the vehicle type u.

A route is defined by a pair (R,u), where $R = (i_1,i_2,...,i_{|R|})$, $i_1 = i_{|R|} = 0$, and $\{i_2,...,i_{|R|-1}\} \subseteq V'$; that is, each route is a circuit in G that starts and ends at the depot and is assigned to a vehicle of type $u \in M$. A route (R,u) is feasible if the sum of all customers' demands on R does not exceed the capacity Q_u of the vehicle assigned to it. A route's cost is the sum of the assigned vehicle's fixed cost and the dependent costs

associated with each traversed arc and the vehicle type. Thus, the aim is to discover feasible routes such that each customer is visited precisely once, the total quantity of routes assigned to vehicles of each type $u \in M$ does not exceed m_u, and the sum of all route costs is minimized.

HFVRP instances are typically categorized with respect to certain criteria. Two primary classes are related to the limitations on the fleet. The problem that characterizes the first class, known as the Fleet Size and Mix (FSM) problem (GOLDEN et al., 1984), can be considered as a particular case of the above definition in which $m_u = \infty, \forall u \in M$. Therefore, this problem consists of identifying the best composition of the fleet and its best routing scheme. For the second category of instances, in which the fleet is limited, the corresponding problem is known as the Heterogeneous Fixed Fleet VRP (HFFVRP) (TAILLARD, E. D., 1999) and consists of optimizing the routing for an available fixed fleet.

The FSM and HFFVRP classes may be further subdivided with respect to the types of vehicle costs considered. The possibilities are as follows: both fixed and dependent costs (the general case), fixed costs only (a particular case in which $r_u = 1, \forall u \in M$), or dependent costs only (a particular case in which $f_u = 0, \forall u \in M$). Five subclasses of FSM and HFFVRP instances concerning this criterion are differentiated in the literature: (1) the FSM problem with both fixed and dependent costs, denoted by FSM-FD (FERLAND; MICHELON, 1988); (2) the FSM problem with fixed costs only, denoted by FSM-F (GOLDEN et al., 1984); (3) the FSM problem with dependent costs only, denoted by FSM-D (TAILLARD, E. D., 1999); (4) the HFFVRP with both fixed and dependent costs, denoted by HFFVRP-FD (LI; GOLDEN; WASIL, 2007); and (5) the HFFVRP with dependent costs only, denoted by HFFVRP-D (TAILLARD, E. D., 1999).

3.2 Proposed method

The proposed method was built through the application of the MineReduce approach in the multi-start iterated local search (MS-ILS) proposed by Penna, Subramanian, and Ochi (2013) for the HFVRP. This section describes how the MineReduce elements have been implemented in the proposed method, referred to as MR-MS-ILS.

In this proposal, solutions in the elite set are represented as sets of arcs. This representation allows the application of the data mining procedure to extract maximal frequent itemsets. As described in Section 3.1, each route in the HFVRP is represented

by a pair (R,u), where $R = (i_1,i_2,...,i_{|R|})$ is a list of vertices, ordered according to the defined visiting sequence, and u is the type of vehicle assigned to the route. In the adopted alternative representation, for each route (R,u), the list R is decomposed into a set of arcs $\{(i_1,i_2),(i_2,i_3),...,(i_{|R|-1},i_{|R|})\}$. Then, the vehicle type u is assigned to each arc in the set, resulting in a set in which each element is a pair composed of an arc (i_r,i_{r+1}), $r = 1,2,...,|R|-1$, in R and the vehicle type u, with the form $\{((i_1,i_2),u),((i_2,i_3),u),...,((i_{|R|-1},i_{|R|}),u)\}$. An arc must be present and associated with the same vehicle type in a minimum number of solutions in the elite set to belong to a pattern. Consequently, the patterns mined are formed of route segments, each one with a vehicle type assigned.

In MR-MS-ILS, the *Reduce* step of MineReduce relies on the use of patterns mined from the elite set to perform a vertex-based PSR procedure by merging customer vertices that appear consecutively (in the same route segment) in a pattern into one customer cluster vertex.

In a binary formulation of the HFVRP, decision variables are defined as: $x_{ij}^k = 1$ if a vehicle of type k travels from customer i to customer j; and $x_{ij}^k = 0$ otherwise. Therefore, differently from the classic VRP, in this case, a vertex-based PSR procedure is not equivalent to fixing values of decision variables. Instead, the original set of decision variables is replaced with a smaller one (since the set of customers is reduced).

For using this strategy, it is necessary to extend the HFVRP model presented in Section 3.1. Since it must be possible to represent a route segment as a customer vertex, each customer $i \in V$ must have an associated l_i value corresponding to the length of the underlying route segment. This value is used to calculate the cost $c_i^u = l_i r_u$, which represents the variable cost for a vehicle of type u to traverse the route segment represented by i, for each combination of a customer $i \in V$ and a vehicle type $u \in M$. Customer vertices of regular (non-reduced) instances and customer vertices of reduced instances that are not customer cluster vertices have a length of 0. The cost of a route becomes, in this extended model, the sum of the fixed cost of the vehicle associated with the route and the variable costs associated with (i) the combination of the vehicle type and each traversed arc and (ii) the combination of the vehicle type and each visited customer vertex.

Let $G = (V, A)$ be a directed graph associated with an HFVRP instance and p a pattern consisting of a set of route segments in that instance, each segment defined by a pair (R', u), where $R' = (i_1, i_2, ..., i_{|R'|})$. Let $G^* = (V^*, A^*)$ be a directed graph associated with a reduced version of the instance associated with G based on p. Such a reduced

version can be obtained as follows. Initially, G^* is defined as a copy of G. For each route segment $(R', u) \in p$, each of the customers in R' is removed from G^* – that is, the vertex corresponding to the customer is removed from V^* and the arcs that connect that vertex to the others are removed from A^*. Also, a customer cluster vertex $i_{R'}$ corresponding to the route segment is added to V^* and arcs connecting $i_{R'}$ to the other vertices in V^* are added to A^*. The demand for $i_{R'}$ is given by $q_{i_{R'}} = \sum_{r'=1}^{|R'|} q_{i_{r'}}$, that is, the sum of customer demands in R'. The distance from each vertex $i^* \in V^*$ to $i_{R'}$ is given by $d_{i^* i_{R'}} = d_{i^* i_1}$, that is, the distance from i^* to i_1 (the first customer in R'). The distance from $i_{R'}$ to each vertex $i^* \in V^*$ is given by $d_{i_{R'} i^*} = d_{i_{|R'|} i^*}$, that is, the distance from $i_{|R'|}$ (the last customer in R') to i^*. Finally, the length of $i_{R'}$ is given by $l_{i_{R'}} = \sum_{r'=1}^{|R'|-1} d_{i_{r'} i_{r'+1}}$, that is, the sum of the distances between consecutive customers in R'.

3.3 Results summary

The 96 HFFVRP-FD benchmark instances introduced by Duhamel, Lacomme, and Prodhon (2010) have been used in the experimental analysis (four instances for tuning parameters and 92 for testing the algorithms). These instances are divided into four sets: Set 1, with 15 instances, each with fewer than 100 customers; Set 2, with 38 instances, each with 100 to 150 customers; Set 3, with 31 instances, each with 151 to 200 customers; and Set 4, with 12 instances, each with more than 200 customers.

Extensive computational experiments were conducted to compare the proposed MR-MS-ILS, the original MS-ILS and its previous hybrid data mining version MDM-MS-ILS (MAIA; PLASTINO; PENNA, 2018), a state-of-the-art method for this problem. Table 1 summarizes the results from this comparison, presenting the numbers of wins/ties (the number of instances for which a method achieved the best results) and the average percentage difference (APD) to the original MS-ILS for each instance set separately and for all 92 instances (global).

The results show a clear superiority of MR-MS-ILS to the other two methods. It achieved better solution quality for the vast majority of the instances and much faster convergence for all of them. MR-MS-ILS attained better average costs for 83% of the instances (65% with statistical significance) compared to the MS-ILS and better average costs for 76% of the instances (64% with statistical significance) in comparison to the MDM-MS-ILS. Suppose small instances (Set 1) – for which the three heuristics have presented similar performance – are excluded from the comparison. In that case, the

Table 1: Summarized results for the HFVRP (MR-MS-ILS vs MS-ILS and MDM-MS-ILS)

		MS-ILS			MDM-MS-ILS			MR-MS-ILS		
		Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
Set 1	Wins/Ties	10	**7**	-	**13**	**7**	-	**13**	6	**14**
	APD				**−0.01%**	**0.00%**	−14.28%	**−0.01%**	0.05%	**−63.72%**
Set 2	Wins/Ties	12	3	-	11	8	-	**29**	**26**	**37**
	APD				−0.01%	−0.04%	−13.71%	**−0.15%**	**−0.19%**	**−65.09%**
Set 3	Wins/Ties	1	-	-	1	-	-	**29**	**30**	**30**
	APD				−0.06%	−0.01%	−13.83%	**−0.39%**	**−0.46%**	**−57.01%**
Set 4	Wins/Ties	-	-	-	-	-	-	**11**	**11**	**11**
	APD				−0.10%	−0.03%	−14.34%	**−0.56%**	**−0.52%**	**−53.24%**
Global	Wins/Ties	23	10	-	25	15	-	**82**	**73**	**92**
	APD				−0.04%	−0.02%	−13.91%	**−0.26%**	**−0.28%**	**−60.83%**

superiority of MR-MS-ILS becomes even more evident: better average costs for 91% of the instances (76% with statistical significance) in comparison to the MS-ILS and better average costs for 86% of the instances (74% with statistical significance) in comparison to the MDM-MS-ILS. Moreover, MR-MS-ILS found new best solutions for 22 instances.

Additionally, the results obtained by MR-MS-ILS were compared to the results reported in the literature for two other state-of-the-art algorithms: HLS (KOCHETOV; KHMELEV, 2015) and HILS-RVRP (PENNA; SUBRAMANIAN; OCHI, et al., 2019). Since the results reported for these other methods were obtained on different machines, the computational times have been adjusted according to the respective CPU performance ratings retrieved from the PassMark CPU benchmarks[1].

Table 2: Summarized results for the HFVRP (MR-MS-ILS vs HLS and HILS-RVRP)

		HLS			HILS-RVRP			MR-MS-ILS		
		Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
Set 1	Wins/Ties	5	**6**	1	**13**	**6**	**10**	10	**6**	3
	APD				**−0.09%**	**−0.09%**	**−119.22%**	−0.08%	−0.03%	−89.85%
Set 2	Wins/Ties	7	5	-	**26**	**21**	**37**	19	11	-
	APD				**−0.22%**	**−0.21%**	**−171.14%**	−0.19%	−0.18%	−140.31%
Set 3	Wins/Ties	2	3	-	**16**	7	**29**	13	**20**	1
	APD				**−0.54%**	−0.32%	**−167.21%**	−0.51%	**−0.50%**	−147.24%
Set 4	Wins/Ties	1	1	-	4	2	**10**	**6**	**8**	1
	APD				−0.36%	−0.22%	**−176.96%**	**−0.53%**	**−0.43%**	−156.42%
Global	Wins/Ties	15	15	1	**59**	36	**86**	48	**45**	5
	APD				**−0.32%**	−0.23%	**−162.46%**	**−0.32%**	**−0.30%**	−136.74%

HILS-RVRP and MR-MS-ILS have presented the best results regarding both solution quality and computational time. The APD values presented for these algorithms are relative to HLS. Regarding solution quality, a balance is observed for the smallest instances (Set 1), whereas HILS-RVRP has the best results for medium-size instances (Set 2), and MR-MS-ILS presents the best results for the largest instances (Sets 3 and 4). Computational times reported for HILS-RVRP and MR-MS-ILS are much shorter

[1] As listed at https://www.cpubenchmark.net on March 27, 2020

than those reported for HLS. Between HILS-RVRP and MR-MS-ILS, the former presents shorter computational times. This comparison showed that MR-MS-ILS is competitive, achieving superior performance for large instances.

To further inspect the behaviour of the new MineReduce-based method, additional analyses were performed using instance 02 from Set 3 of Duhamel, Lacomme, and Prodhon (2010).

Fig. 2 provides an evaluation of the solution costs obtained throughout the execution of each method. The charts in the first row show, for each method, the solution costs obtained per iteration in the generation and local search phases, whereas the second row provides enlarged views focusing on the local search phase curve. In Fig. 3, the charts exhibit the computational time spent in the generation and local search phases per iteration. The dashed vertical lines indicate the iterations preceding a data mining method run.

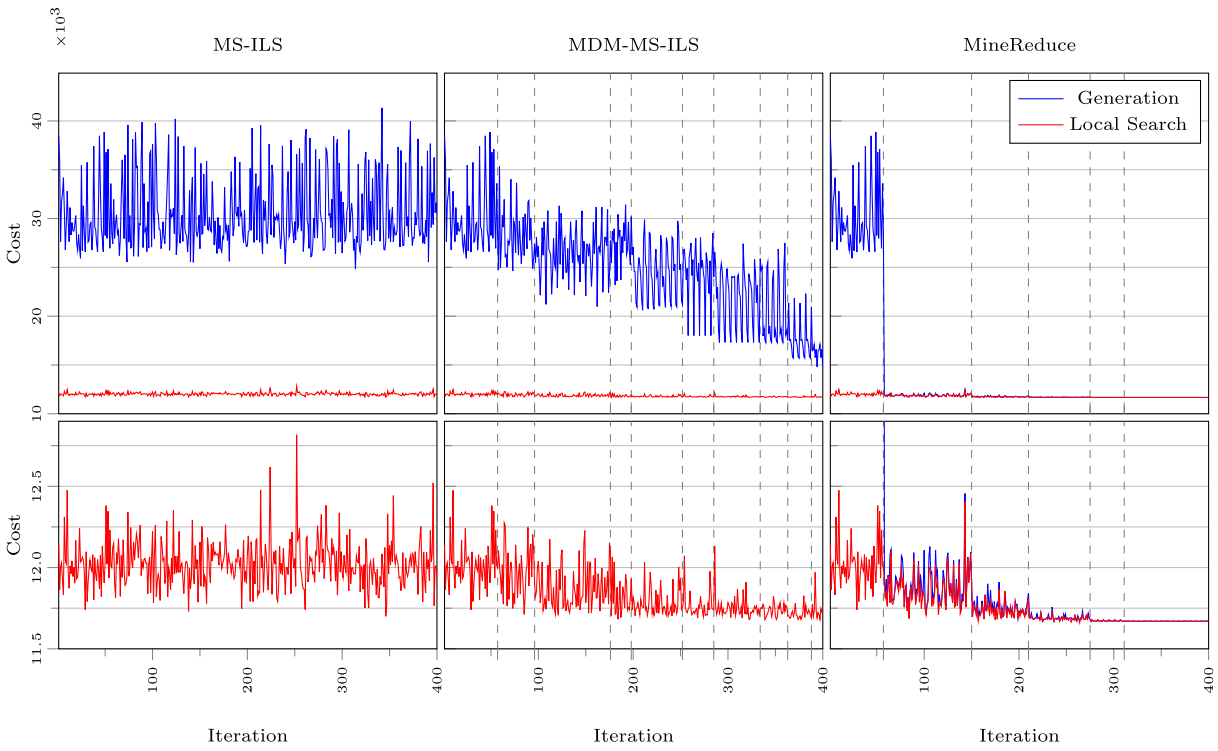


Figure 2: Cost vs. iteration charts illustrating the behavior of MS-ILS, MDM-MS-ILS and MR-MS-ILS (MineReduce) for instance 02 from Set 3 of Duhamel, Lacomme, and Prodhon (2010). Source: (MAIA; PLASTINO; PENNA, 2020, p. 11).

In Fig. 2, the charts in the first row show that the reduction in the costs of the solutions generated by MR-MS-ILS after data mining is much more expressive than that observed for MDM-MS-ILS. The enlarged views in the second row show that the costs of generated solutions fall to a level even lower than that of the costs of solutions discovered

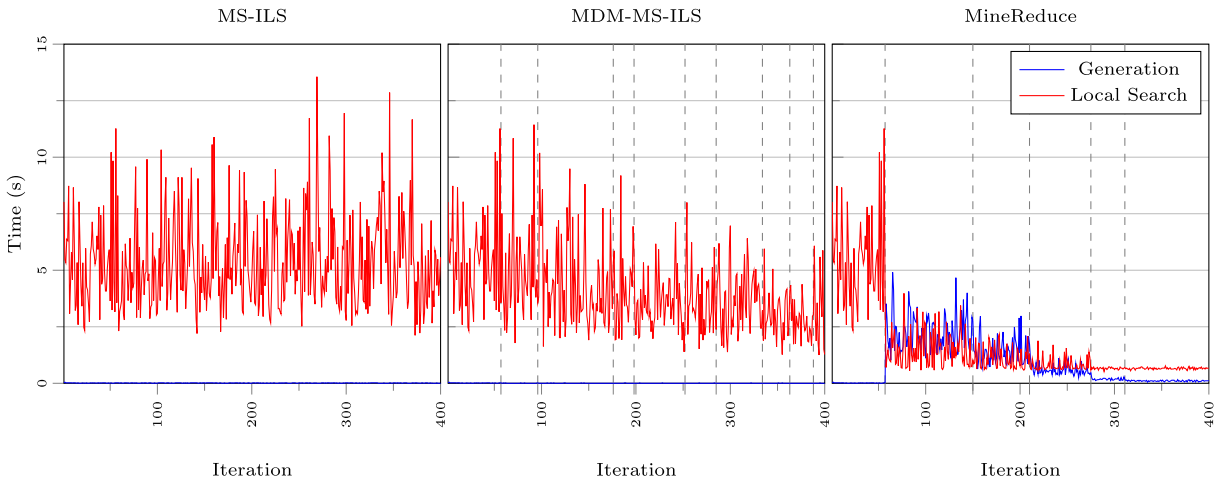


Figure 3: Time vs. iteration charts illustrating the behavior of MS-ILS, MDM-MS-ILS and MR-MS-ILS (MineReduce) for instance 02 from Set 3 of Duhamel, Lacomme, and Prodhon (2010). Source: (MAIA; PLASTINO; PENNA, 2020, p. 11).

through the local search in previous iterations, and the local search finds even better solutions after each run of the data mining procedure.

On the other hand, as shown in the last chart of Fig. 3, the computational time spent in the generation phase – which is close to zero in the first iterations – increases after the first run of the data mining procedure. This increase is due to the execution of the problem size reduction process, which includes a local search on a reduced version of the problem instance. However, the increase in time spent in the generation phase is compensated by an expressive reduction in time spent in the local search phase. After data mining, the combined time spent in the generation and local search phases per iteration is significantly reduced.

The overall reduction in computational time can be explained by the fact that MineReduce shrinks the problem instance and, consequently, the search space. Therefore, the first local search (embedded in MineReduce's generation phase, over the reduced instance) performs a much smaller number of movement evaluations. Afterwards, the second local search (actual local search phase, over the original instance) starts from a higher-quality solution, so it converges faster as well.

In MR-MS-ILS, as the charts show, the reduction of solution costs and computational time is intensified after each execution of the data mining procedure, reaching very low levels in the last iterations, which explains the superiority demonstrated by the results of the experiments presented in Tables 1 and 2.

One last experiment was conducted to assess convergence speed based on time-to-target (TTT) plots (AIEX; RESENDE; RIBEIRO, 2007). A TTT plot displays, on the ordinate axis, the probability that an algorithm will discover a solution at least as good as a given target within a given running time, which is shown on the abscissa axis. In this experiment, each method was run 100 times, with 100 different random seeds, targeting a solution cost ≤ 11780. The obtained chart is shown in Fig. 4.

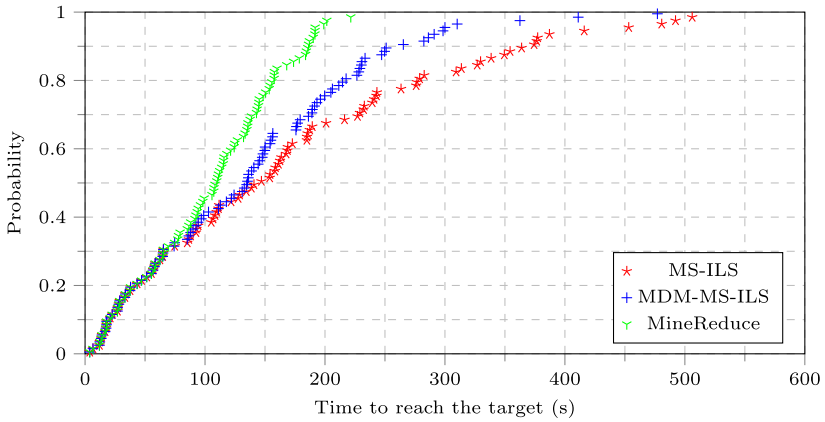


Figure 4: TTT plots comparing all methods for instance 02 from Set 3 of Duhamel, Lacomme, and Prodhon (2010). Source: (MAIA; PLASTINO; PENNA, 2020, p. 12).

The chart shows that MR-MS-ILS converges faster than the other methods. It is possible to observe that the probability that the target will be reached within 200 seconds, for example, is approximately 97% for MR-MS-ILS, 76% for MDM-MS-ILS, and 67% for MS-ILS.

The analyses of the charts presented in Figs. 2, 3 and 4 clearly illustrate the influence of the MineReduce approach in the behavior of metaheuristics. As expected, this behaviour leads to solution quality improvement and convergence speedup.

4 MineReduce application to the Minimum Weighted Vertex Cover Problem

This chapter summarizes the results obtained by applying the MineReduce approach to the Minimum Weighted Vertex Cover Problem (MWVCP). A MineReduced-based multi-start iterated tabu search method was proposed for this problem in (MAIA; PLASTINO; SOUZA, 2020), a paper published in the proceedings of the 2020 edition of the *International Conference on Optimization and Learning*. Additionally, an extended version of that paper, presented in Appendix B, is currently under review in *Applied Soft Computing*. Section 4.1 describes the MWVCP, Section 4.2 describes how the MineReduce elements have been implemented in the proposed method, and Section 4.3 reports a summary of the obtained results.

4.1 Problem definition

The Minimum Weighted Vertex Cover Problem (MWVCP) is a well-known combinatorial optimization problem. Given an undirected graph where each vertex has a positive weight, MWVCP aims at finding a subset of the vertices covering all edges of the graph (a vertex cover) with the minimum total weight.

It is a generalization of the classical vertex cover problem (VCP), which lies in the roots of the NP-completeness theory as one of Karp's 21 NP-complete problems (KARP, 1972). The VCP is also central in the parameterized complexity theory (FELLOWS et al., 2018), and the integer variant of the MWVCP (where the weights are positive integers) is known to be fixed-parameter tractable as well (NIEDERMEIER; ROSSMANITH, 2003).

Beyond its theoretical interest, the MWVCP has many practical applications. For instance, graphs are naturally well-suited for modelling transportation networks. Vertices can represent locations of interest (such as cities or road intersections, for instance),

each edge can represent a link between two locations (such as a road or a rail line, for instance), and the weights of the vertices can represent costs (or other valuation attributes) associated to the respective locations. In such a scenario, minimum weighted vertex covers can be helpful in many real-world applications, which include the identification of critical nodes (BAZGAN; TOUBALINE; TUZA, 2011), the placement of charging stations for electric vehicles (FUNKE; NUSSER; STORANDT, 2015), and the placement of monitoring devices (GUSEV, 2020).

4.2 Proposed method

The proposed method was built through the application of the MineReduce approach in the multi-start iterated tabu search (MS-ITS) proposed by Zhou et al. (2016), a state-of-the-art method for the MWVCP. This section describes how the MineReduce elements have been implemented in the proposed method, referred to as MR-MS-ITS.

Each mined pattern is a set of vertices considered likely to be part of an optimal solution. The PSR is accomplished by deleting the vertices that are part of a mined pattern (assuming they are part of the solution) and then deleting the remaining isolated vertices (which, in turn, cannot be part of the solution).

The representation of solutions and patterns for this problem is straightforward, and so is the reduction process, which produces a graph that perfectly matches the original problem formulation. Therefore, no additional adaptations to the model are necessary.

4.3 Results summary

The standard MWVCP benchmark instances introduced by Shyu, Yin, and Lin (2004) were used in the experimental analysis. Each instance consists of an undirected and vertex-weighted graph with n vertices and m edges. These instances are divided into three sets: SPI, with 400 small-scale instances (n between 10 and 25, m between 10 and 200); MPI, with 710 middle-scale instances (n between 50 and 300, m between 50 and 5000); and LPI, with 15 large-scale instances (n between 500 and 1000, m between 500 and 20000).

Furthermore, sets SPI and MPI are subdivided into two types of instances. In instances of Type I, weights and degrees of vertices are not interrelated, whereas, in instances of Type II, they are (the weight $w(i)$ on vertex i is randomly distributed over the

interval $[1, d(i)^2]$, where $d(i)$ is the degree of vertex $i, 1 \leq i \leq n$). All instances in the LPI set are of Type I.

There are ten instances of each type in the SPI and MPI sets for each combination of n and m, whereas there is only one instance for each combination in the LPI set. Each heuristic was run 30 times for each instance. Like in previous state-of-the-art studies (SHYU; YIN; LIN, 2004; JOVANOVIC; TUBA, 2011; BOUAMAMA; BLUM; BOUKERRAM, 2012; ZHOU et al., 2016), results were averaged over all runs for each combination of n and m.

Extensive computational experiments were conducted to compare MR-MS-ITS and the original MS-ITS. Table 3 summarizes the results from this comparison, presenting the numbers of wins/ties (the number of cases for which a method achieved the best results) and the average percentage difference (APD) to the original MS-ITS for MPI and LPI sets separately and for all of them together (global). Results for the SPI sets are not presented since both methods obtained optimal solutions for all of their instances in less than 0.2s on average.

Table 3: Summarized results for the MWVCP

		MS-ITS			MR-MS-ITS		
		Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
Set MPI (Type I)	Wins/Ties	n/a	13	5	n/a	**39**	**34**
	APD				n/a	**−0.020%**	**−8.475%**
Set MPI (Type II)	Wins/Ties	n/a	24	10	n/a	**32**	**22**
	APD				n/a	**−0.003%**	**−7.808%**
Set LPI	Wins/Ties	2	-	2	**14**	**15**	**13**
	APD				**−0.304%**	**−0.588%**	**−5.723%**
Global	Wins/Ties	2	37	17	**14**	**86**	**69**
	APD				**−0.304%**	**−0.113%**	**−7.747%**

The results show a clear superiority of MR-MS-ITS to MS-ITS. It achieved equal or better average solution costs in all cases and faster convergence in most cases. The average cost improvements obtained by MR-MS-ITS were statistically significant for 94 out of the 390 MPI instances of Type I, 18 out of the 320 MPI instances of Type II, and all 15 LPI instances.

To further inspect the behaviour of the new MineReduce-based method, additional analyses were performed using instance vc_800_2000 from the LPI set of Shyu, Yin, and Lin (2004).

Fig. 5 provides an evaluation of the solution costs obtained throughout the execution of each method. The charts in the first row show, for each method, the solution costs

obtained per iteration in the generation and local search phases, whereas the second row provides enlarged views focusing on the local search phase curve.

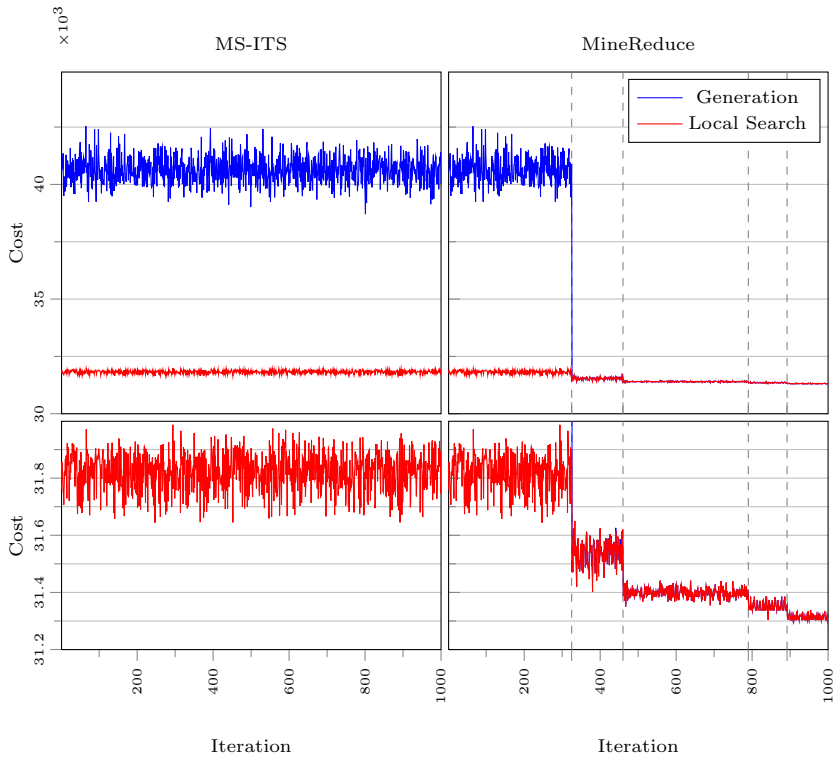


Figure 5: Cost vs. iteration charts illustrating the behavior of MS-ITS and MR-MS-ITS (MineReduce) for instance vc_800_2000 from the LPI set of Shyu, Yin, and Lin (2004).

The decrease in the costs of solutions generated by MR-MS-ITS after data mining is expressive. The charts in the second row show that the costs obtained in the generation phase are even lower than those of solutions found in the local search phase of previous iterations, and even better solutions are found after each data mining run. Furthermore, it can be noticed that the solutions generated by MR-MS-ITS achieved a level of quality that the local search could not improve in most iterations, which is evidenced by the overlapping curves. The explanation is that these initial solutions, which were local optima for the reduced instances, were also local optima for the original instance.

Another experiment was conducted to assess convergence speed based on time-to-target (TTT) plots (AIEX; RESENDE; RIBEIRO, 2007). The target cost used was 31660. The obtained chart is shown in Fig. 6. The chart shows that MR-MS-ITS converges much faster than MS-ITS. The probability that the target will be reached within 40 seconds, for example, is nearly 100% for MR-MS-ITS and about 46% for MS-ITS.

Additionally, MineReduce has been compared to a kernelization algorithm to assess its problem size reduction (PSR) effectiveness. Kernelization is an exact method based on the

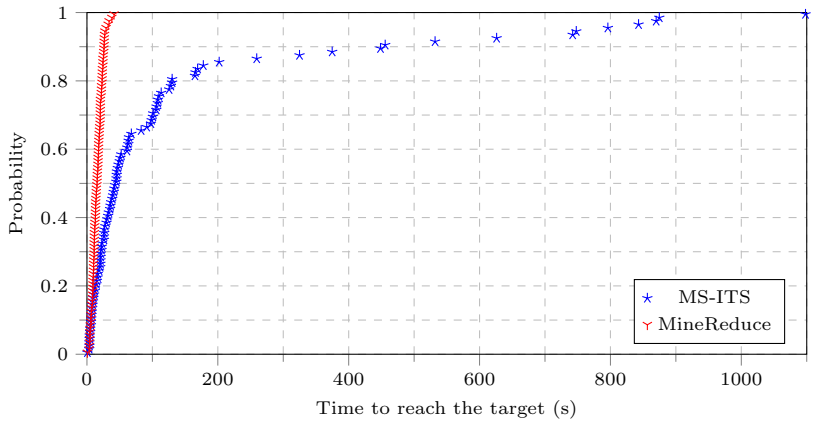


Figure 6: TTT plots comparing MS-ITS and MR-MS-ITS (MineReduce) for instance vc_800_2000 from the LPI set of Shyu, Yin, and Lin (2004).

parameterized complexity theory (DOWNEY; FELLOWS, 1999) for reducing in polynomial time a problem instance to a kernel based on the application of a set of reduction rules. A kernel can be regarded as an equivalent version of the original instance with a size bounded by a function of a parameter (which represents an aspect of the problem), and an optimal solution for the original instance can be straightforwardly derived from an optimal solution for the kernel.

The kernelization algorithm used in this comparison was obtained through an adaptation of a kernelization algorithm for the vertex cover problem described in (CYGAN et al., 2015). Let (G, k) be a parameterized instance of MWVCP, where the parameter k is an upper bound on the optimal solution value for the weighted input graph G. In the following reduction rules, $N(v)$ denotes the neighbourhood of vertex v, $w(v)$ denotes the weight of vertex v, $w(S)$ denotes the total weight of all vertices in S, and $\mu(G)$ denotes the minimum vertex weight of G.

Rule MWVCP-1: If G contains an isolated vertex v, then delete v from G. The new instance is $(G - v, k)$.

Rule MWVCP-2: If there is a vertex v such that $w(N(v)) > k$, then delete v (and its incident edges) from G, and decrement k by $w(v)$. The new instance is $(G-v, k-w(v))$.

The algorithm repeatedly applies rules MWVCP-1 and MWVCP-2 until neither of their conditions is satisfied. This process completely removes the vertices of degree 0 (which are certainly out of an optimal solution) and those with a neighbourhood of total weight at least $k + 1$ (part of an optimal solution). This algorithm produces a kernel of the original instance with at most $(k/\mu(G))^2 + k/\mu(G)$ vertices and $(k/\mu(G))^2$ edges.

A methodology to experimentally evaluate and compare PSR techniques was formulated. An alternative interpretation for PSR (kernelization being regarded as a special kind of PSR) has been proposed to define suitable metrics for this assessment. PSR can be regarded as a binary classification problem in which the objective is to analyze each substructure of a problem instance and determine whether it must be part of an optimal solution. The substructures classified by a PSR method are removed from the instance, whereas the remaining substructures (those the PSR method cannot classify) compose the reduced instance. Given this interpretation, assessment metrics for classification problems (THARWAT, 2021) can be used to evaluate PSR methods experimentally.

We can refer to "in solution" as the positive class and to "out of solution" as the negative class. Then, for a given optimal solution, there are five possible outputs for each choice made: true positive (TP) is an element that is part of the solution and is correctly classified; false positive (FP) is an element that is not part of the solution but is misclassified; true negative (TN) is an element that is not part of the solution and is correctly classified; false negative (FN) is an element that is part of the solution but is misclassified; and non-classified (NC) is an element that cannot be classified by the PSR method (and, hence, is part of the reduced instance).

A false positive in this context causes any solution obtained from the reduced instance to be non-optimal. Therefore, the *precision* of a PSR outcome (reduced instance) is related to its capacity to output an optimal solution. On the other hand, the *recall* of a PSR outcome is related to the degree of size reduction it represents regarding the original instance. Finally, the F-measure combines precision and recall in one score that indicates the overall quality of a PSR outcome.

MineReduce and the kernelization algorithm have been evaluated regarding *precision*, *recall* and *F-measure*. The values for these metrics vary between 0 and 1, where 1 is the optimal value.

For this experiment, the instances of the SPI set (the only ones with known optimal solutions) were used. Since MineReduce can produce more than one reduced instance for each original instance (one for each mined pattern), two values of each metric were computed for it. One (labelled "best") takes only the best-score reduction into account, and the other (labelled "avg.") is the average value of all reductions.

The charts in Fig. 7 and Fig. 8 present the average measures for instances of Type I and II, respectively. They show MineReduce achieves optimal precision in most cases, and its overall average precision is over 0.990, almost the optimum value.

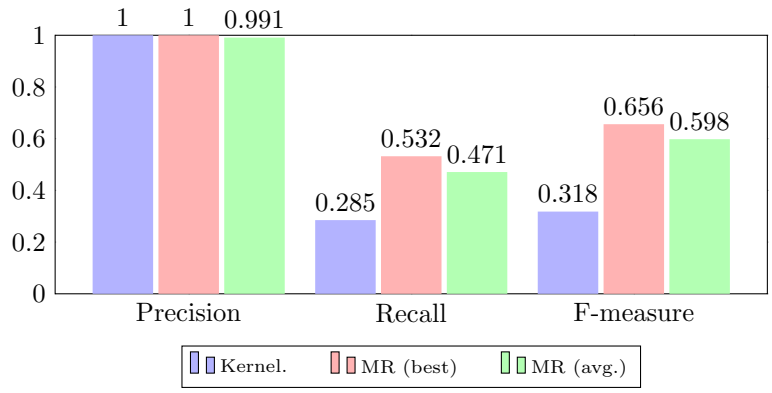


Figure 7: Average PSR assessment measures for Set SPI (Type I)

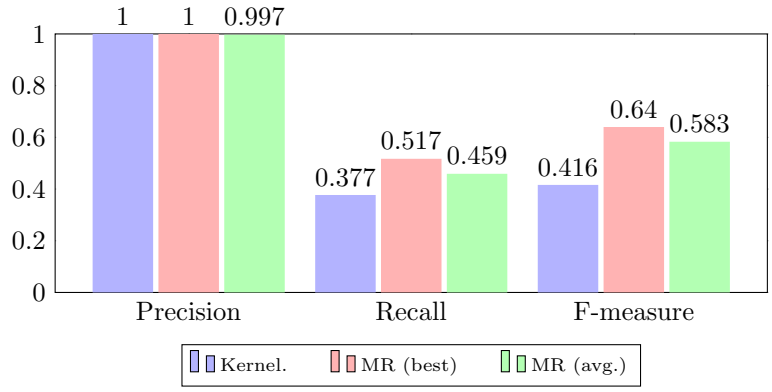


Figure 8: Average PSR assessment measures for Set SPI (Type II)

Also, regarding the recall, MineReduce performed generally better than the kernelization algorithm. For Type I instances, the kernelization algorithm achieved an overall average of 0.285, whereas MineReduce achieved 0.471 (avg.) and 0.532 (best). For Type II instances, the kernelization algorithm achieved an average of 0.377, whereas MineReduce achieved 0.459 (avg.) and 0.517 (best).

Therefore, MineReduce also obtained significantly higher F-scores than the kernelization algorithm. For Type I instances, the kernelization algorithm obtained an overall average of 0.318, whereas MineReduce achieved 0.598 (avg.) and 0.656 (best). For Type II instances, the kernelization algorithm achieved an average of 0.416, whereas MineReduce achieved 0.583 (avg.) and 0.640 (best).

Although MineReduce is a heuristic method, these results show that it has achieved an average precision almost equal to that of the kernelization algorithm while fixing significantly more elements in the solution (higher recall).

5 MineReduce application to the Multi-Source Capacitated Facility Location Problem with Customer Incompatibilities

This chapter summarizes the results obtained by applying the MineReduce approach to the Multi-Source Capacitated Facility Location Problem with Customer Incompatibilities (MS-CFLP-CI). The MS-CFLP-CI is a new variant of the Capacitated Facility Location Problem proposed in the MESS 2020+1 Metaheuristics Competition[1]. A MineReduced-based multi-start iterated local search method was proposed for this problem and ranked first in the competition. This method is described in a paper co-authored by the competition's organizers and the members of its top-3 finalist teams, which is currently under review in *Soft Computing*. The paper, presented in Appendix C, introduces the problem, describes and compares the competing methods. Section 5.1 describes the MS-CFLP-CI, Section 5.2 describes how the MineReduce elements have been implemented in the proposed method, and Section 5.3 reports a summary of the obtained results.

5.1 Problem definition

Let $\mathcal{J} = \{1,\ldots,J\}$ be a set of facilities such that each facility $j \in \mathcal{J}$ has a capacity s_j and an opening cost f_j, and let $\mathcal{I} = \{1,\ldots,I\}$ be a set of customers such that each customer i has a demand of quantity of goods d_i to be completely satisfied by one or more facilities. The shipping cost c_{ij} is the cost per unit of transporting goods from facility j to customer i. Finally, a set Γ of pairs of incompatible customers is defined.

The problem consists in defining the number of goods moved from each facility $j \in \mathcal{J}$ to each customer $i \in \mathcal{I}$ to minimize the sum of the cost of opening facilities and the cost to ship the goods from the facilities to the customers. The constraints are the following:

[1] https://www.ants-lab.it/mess2020/#competition

- The total quantity of goods taken from an open facility cannot exceed its capacity;
- The total quantity of goods supplied to a customer must be exactly equal to its demand;
- The same facility cannot supply two incompatible customers.

5.2 Proposed method

This section describes how the MineReduce elements have been implemented in the proposed method, a MineReduce-based Multi-Start ILS (MR-MS-ILS).

The mined patterns are frequent sets of pairs $\langle i,j \rangle$, i.e., sets of customer-facility assignments that are frequent in solutions of the elite set E. After these frequent itemsets are mined, their assignments are filled with the corresponding minimum quantities in E, i.e., for each pair $\langle i,j \rangle$ in a frequent itemset, a supplied quantity q is set, which corresponds to the minimum positive supplied quantity for that pair among all solutions in E. Therefore, each final pattern is a set of assignments $\langle i,j,q \rangle$.

Patterns are used to reduce the size of the problem instance based on the assumption that their elements shall be part of the solution. In the particular case of reducing this problem, an extension to the problem formulation is needed. We introduce incompatibilities between customers and facilities, represented by a set Γ', such that for each $\langle i,j \rangle \in \Gamma'$, i cannot be supplied by j. Hence, an additional set of constraints is introduced:

- A customer cannot be supplied by an incompatible facility.

The reduction process works as follows. Given a problem instance $\mathcal{P}$ and a pattern p, the reduced instance $\mathcal{P}'$ is initialized as a copy of $\mathcal{P}$. Then, for each assignment $\langle i,j,q \rangle$ in p, the following changes are applied to $\mathcal{P}'$: (i) decrease both s_j and d_i by q; (ii) set $f_j = 0$; and (iii) for each customer i' incompatible with i (from Γ), add a pair $\langle i',j \rangle$ to Γ'.

As a consequence, the domain of each x_{ij} variable will be reduced, and the additional constraints will fix the values of many other decision variables, effectively reducing the problem instance size.

5.3 Results summary

Four methods have been compared: the MineReduce-based Multi-Start ILS (MR-MS-ILS), proposed by this thesis' author; a Greedy Randomized Adaptive Search Procedure (GRASP) and a Permutation Coded Evolutionary Algorithm (PcEA), proposed by other competing teams; and a Multistart Greedy (MG) algorithm, proposed by the competition organizers.

The methods have been tested on artificial instances created for the competition. Instances have been partitioned into two sets: a *training* set with 20 instances, which was publicly available since the competition started so the competitors could use it for tuning their algorithms; and a *validation* set with ten instances, which was kept hidden during the competition as it was used for defining the final ranking.

The comparisons are based on the best solution found by each method in a given timeout (t). Two different timeouts are used based on the number of facilities J in the problem instance. The first is the one proposed for the competition, which grants $10\sqrt{J}$ seconds for solving each instance, whereas the second one grants J seconds per instance.

Tables 4 and 5 summarize the results for the first and the second timeouts, respectively, presenting the numbers of wins (the number of instances for which a method achieved the best results) and the average percentage difference (APD) to PcEA for the training and validation sets separately and for all of them together (global).

Table 4: Summarized results for the MS-CFLP-CI ($t = 10\sqrt{J}$)

		PcEA		GRASP		MG		MR-MS-ILS	
		Best Cost	Avg. Cost	Best Cost	Avg. Cost	Best Cost	Avg. Cost	Best Cost	Avg. Cost
Train. Set	Wins	-	-	3	3	2	2	**15**	**15**
	APD			−76.79%	−78.09%	−76.78%	−78.24%	**−80.21%**	**−81.58%**
Valid. Set	Wins	-	-	1	1	3	3	**6**	**6**
	APD			−76.77%	−77.64%	−77.14%	−78.08%	**−80.34%**	**−81.24%**
Global	Wins	-	-	4	4	5	5	**21**	**21**
	APD			−76.79%	−77.94%	−76.90%	−78.19%	**−80.25%**	**−81.46%**

Additionally, Table 6 shows the average ranks throughout all runs (10×4) upon all instances for each combination of dataset and timeout, which was the criteria adopted in the competition for ranking the methods. Note that if a method had been better than all others in all runs, it would have obtained ranks 1, 2, ..., 10 for its ten runs, resulting in an average rank of 5.5 (the average of numbers 1, 2, ..., 10).

The results show that MR-MS-ILS is indisputably superior to the other methods.

Table 5: Summarized results for the MS-CFLP-CI ($t = J$)

		PcEA		GRASP		MG		MR-MS-ILS	
		Best Cost	Avg. Cost	Best Cost	Avg. Cost	Best Cost	Avg. Cost	Best Cost	Avg. Cost
Train. Set	Wins	-	-	3	3	1	2	**16**	**15**
	APD			−76.71%	−77.82%	−76.56%	−77.82%	**−80.18%**	**−81.35%**
Valid. Set	Wins	-	-	1	1	2	2	**7**	**7**
	APD			−76.63%	−77.69%	−76.73%	−77.92%	**−80.36%**	**−81.37%**
Global	Wins	-	-	4	4	3	4	**23**	**22**
	APD			−76.68%	−77.78%	−76.62%	−77.86%	**−80.24%**	**−81.35%**

Table 6: Average ranks obtained by the methods

	Train. Set		Valid. Set	
	$t = 10\sqrt{J}$	$t = J$	$t = 10\sqrt{J}$	$t = J$
MR-MS-ILS	**8.3**	**7.7**	**9.2**	**8.7**
MG	17.7	18.4	16.4	16.9
GRASP	20.5	20.4	20.9	20.8
PcEA	35.5	35.5	35.5	35.5

It achieved better solution quality for the vast majority of the instances and far better average ranks, establishing best known solutions for 23 out of the 30 instances in the proposed benchmark set.

During the development of the proposed method, experiments comparing MR-MS-ILS to a version of it with the MineReduce-based components disabled (MS-ILS) were conducted to measure the effect of the MineReduce approach – using the training set instances with $t = 10\sqrt{J}$. Table 7 summarizes the results obtained in this comparison, showing that the MineReduce approach significantly improves the baseline method's performance.

Table 7: Summarized results for the MS-CFLP-CI (MR-MS-ILS vs MS-ILS)

	MS-ILS		MR-MS-ILS	
	Best Cost	Avg. Cost	Best Cost	Avg. Cost
Wins	-	-	**19**	**20**
APD			**−2.42%**	**−2.46%**

6 MineReduce application to the Minimum Latency Problem

This chapter summarizes the results obtained by applying the MineReduce approach to the Minimum Latency Problem (MLP). A MineReduced-based version of a hybrid method that combines components of GRASP, ILS and RVND was proposed for this problem in a paper submitted to the 14th edition of the *Metaheuristics International Conference* and presented in Appendix D. Section 6.1 describes the MLP, Section 6.2 describes how the MineReduce elements have been implemented in the proposed method, and Section 6.3 reports a summary of the obtained results.

6.1 Problem definition

The MLP is a variant of the well-known travelling salesperson problem (TSP) where the objective is to minimize the sum of arrival times at vertices in a Hamiltonian cycle. It can model several real-world applications like: distribution logistics, machine scheduling, and disaster relief (FISCHETTI; LAPORTE; MARTELLO, 1993; BLUM, A. et al., 1994; CAMPBELL; VANDENBUSSCHE; HERMANN, 2008).

Let $G = (V,A)$ be a directed graph, where $V = \{0,1,...,n\}$ is a set composed of $n+1$ vertices and $A = \{(i,j) : i,j \in V, i \neq j\}$ is the set of arcs. Vertex 0 is the depot from where the salesperson departs, whereas the set $V' = V \backslash \{0\}$ consists of the remaining vertices representing the n customers. For each arc $(i,j) \in A$, there is an associated travel time t_{ij}. The aim is to find a Hamiltonian cycle $(i_0,i_1,...,i_{n+1})$ in G, where $i_0 = i_{n+1} = 0$ (i.e., the cycle starts and ends at the depot), that minimizes the sum of arrival times, given by $\sum_{k=1}^{n+1} l(i_k)$, where $l(i_k) = \sum_{m=0}^{k-1} t_{i_m i_{m+1}}$ represents the latency of vertex i_k (i.e., the total travel time to reach i_k).

6.2 Proposed method

The proposed method was built through the application of the MineReduce approach in the MDM-GILS-RVND algorithm, a state-of-the-art method for the MLP proposed by Santana, Plastino, and Rosseti (2022). MDM-GILS-RVND is the result of the incorporation of data mining into a hybrid method that combines components of GRASP, ILS and RVND (GILS-RVND), proposed by M. M. Silva et al. (2012). This section describes how the MineReduce elements have been implemented in the proposed method, referred to as MR-GILS-RVND.

One key aspect of MDM-GILS-RVND is a move evaluation procedure inherited from GILS-RVND. It consists of a framework that uses preprocessed data structures to compute costs of neighbor solutions in constant amortized time operations (VIDAL et al., 2011; KINDERVATER; SAVELSBERGH, 2018). In practice, three data structures are used to store the partial costs of each subsequence of vertices of a local minimum solution, where the cost of every neighbor solution is reached by computing their partial costs on a "by concatenation" fashion. These data structures and the concatenation process are described as follows:

- The *duration* $T(\sigma)$ of a sequence σ, which is the total travel time to perform the visits in the sequence.
- The *cost* $C(\sigma)$ to perform a sequence σ, when starting at time 0.
- The *delay* $W(\sigma)$ associated with a sequence σ, which is the number of customers visited in the sequence.

Let $T(i)$, $C(i)$ and $W(i)$ denote the values of the re-optimization data structures corresponding to a subsequence with only a single vertex i. In this case: $T(i) = 0$ and $C(i) = 0$ since there is no travel time; and $W(i) = 1$ if i is a customer, otherwise $W(i) = 0$. These values can be computed on larger subsequences by induction on the concatenation operator $\oplus$ as follows. Let $\sigma = (\sigma_u, \ldots, \sigma_v)$ and $\sigma' = (\sigma'_w, \ldots, \sigma'_x)$ be two subsequences. The subsequence $\sigma \oplus \sigma' = (\sigma_u, \ldots, \sigma_v, \sigma'_w, \ldots, \sigma'_x)$ is characterized by the following values:

- $T(\sigma \oplus \sigma') = T(\sigma) + t_{\sigma_v \sigma'_w} + T(\sigma')$
- $C(\sigma \oplus \sigma') = C(\sigma) + max(W(\sigma'), 1)(T(\sigma) + t_{\sigma_v \sigma'_w}) + C(\sigma')$
- $W(\sigma \oplus \sigma') = W(\sigma) + W(\sigma')$

The reduction process adopted for this problem is similar to the one adopted for the HFVRP (see Section 3.2). In this case, a mined pattern is a set of subsequences of customers. These subsequences can be contracted by replacing all vertices in a subsequence with a single vertex.

Let $G = (V, A)$ be a directed graph associated with an MLP instance and p a pattern consisting of a set of subsequences of customer vertices in that instance. Let $G^* = (V^*, A^*)$ be a directed graph associated with the corresponding reduced instance based on p. Such a reduced version can be obtained as follows. Initially, G^* is defined as a copy of G. For each subsequence $\sigma = (i_1, i_2, ..., i_{|\sigma|}) \in p$ selected to be contracted, each of the customers in σ is removed from G^* – that is, the vertex corresponding to the customer is removed from V^* and the arcs that connect that vertex to the others are removed from A^*. Then, a customer vertex i_σ corresponding to the subsequence is added to V^* and arcs connecting i_σ to the other vertices in V^* are added to A^*. The travel time from each vertex $i^* \in V^*$ to i_σ is given by $t_{i^* i_\sigma} = t_{i^* i_1}$, that is, the travel time from i^* to i_1 (the first customer in σ). The travel time from i_σ to each vertex $i^* \in V^*$ is given by $t_{i_\sigma i^*} = t_{i_{|\sigma|} i^*}$, that is, the travel time from $i_{|\sigma|}$ (the last customer in σ) to i^*.

The values in the "by concatenation" framework structures for a subsequence with only the single vertex i_σ are defined as $T(i_\sigma) = T(\sigma)$, $C(i_\sigma) = C(\sigma)$ and $W(i_\sigma) = W(\sigma)$.

In this implementation, all subsequences in a pattern are sorted in decreasing length (number of traversed arcs) order. Then, they are contracted from the longest to the shortest until a portion γ (a parameter) of all pattern's arcs has been used. The adoption of this strategy was motivated by preliminary tests showing that the mined patterns contained too many arcs, producing reduced instances that were too small. Hence, after expanding solutions found for the reduced instances, a considerable effort was still necessary for the local search on the original instance. Using only a portion of the arcs in a pattern adds control to the reduction factor. Finally, longer subsequences have been favoured because they are less likely to occur than short subsequences given the same minimum support. Therefore, they represent more robust and relevant portions of the patterns.

6.3 Results summary

In the experimental analysis, 56 TSP benchmark instances with 120 to 1379 customers from TSPLIB (REINELT, 1991) were used. They have been split into a *training set*, with 12 instances used for parameter tuning, and a *validation set*, with the remaining 44

instances used for evaluation only. Extensive computational experiments were conducted to compare MR-GILS-RVND and the original MDM-GILS-RVND.

Table 8 summarizes the results obtained in the experiments using the instances in the validation set. MDM-GILS-RVND and MR-GILS-RVND are compared regarding the numbers of wins in best cost, average cost and average CPU running time, the number of new best solutions found, and the summed number of best known solutions (BKS) and new best solutions found.

Table 8: Results summary (all validation instances)

	MDM-GILS-RVND	MR-GILS-RVND
Wins Best Cost	4	6
Wins Avg. Cost	12	18
Wins Avg. Time	25	19
New best	-	5
Nb BKS + new best	37	39

The comparison on all 44 benchmark instances shows that MR-GILS-RVND overcomes MDM-GILS-RVND regarding solution quality, obtaining better solutions for most instances. Furthermore, MR-GILS-RVND found new best solutions for five instances.

On the other hand, MDM-GILS-RVND obtained more wins in average time. This can be explained by the fact that all 15 instances with $n \leq 195$ are in the validation set. These small instances are easier than the others, and all of them have been solved optimally by exact methods. The original MDM-GILS-RVND finds their optimal solutions in a few seconds. Thus, the slight computational overhead introduced by applying the size reduction process in MR-GILS-RVND is not compensated by a convergence speedup as usual since the solutions cannot be further improved. Therefore, a separate comparison is presented in Table 9 considering only the instances with $n > 195$ (the 29 largest instances).

Table 9: Results summary ($n > 195$)

	MDM-GILS-RVND	MR-GILS-RVND
Wins Best Cost	4	6
Wins Avg. Cost	9	17
Wins Avg. Time	15	14
New best	-	5
Nb BKS + new best	22	25

For these larger instances, both methods are technically tied regarding average com-

putational time, whereas the superiority of MR-GILS-RVND regarding solution quality is further evidenced, with about twice the number of wins of MDM-GILS-RVND in average cost. Hence, these results show that the MineReduce approach, applied in MR-GILS-RVND, improved solution quality without increasing computational time over MDM-GILS-RVND. Additionally, MR-GILS-RVND found new best solutions for 6 out of the 12 instances in the training set.

7 Data mining application to the Capacitated Vehicle Routing Problem

Multi Data Mining (MDM) is an approach for incorporating data mining into metaheuristics that was initially proposed for a hybrid version of GRASP named MDM-GRASP (PLASTINO et al., 2014), which was, in turn, an adaptive version of the DM-GRASP metaheuristic (RIBEIRO; PLASTINO; MARTINS, 2006; SANTOS; MARTINS; PLASTINO, 2008). In DM-GRASP, an *elite set* E keeps the best solutions found in the first half of the multi-start iterations. Then, a data mining method is used to extract patterns from E, which are used to build initial solutions in the second half. In MDM, E keeps the best solutions found throughout all multi-start iterations, the data mining method is invoked every time E is considered *stable* (based on a number of consecutive iterations without any changes), and the obtained patterns are used in the subsequent iterations.

This chapter summarizes the results obtained by applying MDM to the Capacitated Vehicle Routing Problem (CVRP). An MDM-based version of the Hybrid Genetic Search (HGS) metaheuristic of Vidal (2022) was proposed for the CVRP track of the 12th DIMACS Implementation Challenge[1]. It is described in (MAIA; PLASTINO; SOUZA, 2022), a paper that was submitted with the author's entry, which ranked second and was presented at the challenge's workshop. The paper is included in Appendix E. Section 7.1 describes the CVRP, Section 7.2 describes how the MDM elements have been implemented in the proposed method, and Section 7.3 reports a summary of the obtained results.

7.1 Problem definition

The CVRP is described as follows. Let $G = (V,A)$ be a directed graph, where $V = \{0,1,...,n-1\}$ is a set composed of n vertices and $A = \{(i,j) : i,j \in V, i \neq j\}$ is the set of arcs. Vertex 0 is the depot from where the vehicles depart, whereas the set $V' = V \backslash \{0\}$ consists of the remaining vertices representing $n-1$ customers. Each customer $i \in V'$ is

[1] http://dimacs.rutgers.edu/programs/challenge/vrp/cvrp

associated with a non-negative quantity q_i that specifies the demand for some resource, and a vehicle can carry a maximum quantity Q of the resource. For each arc $(i,j) \in A$, there is an associated cost d_{ij} representing the distance between vertices i and j. A route is defined by a sequence of vertices $R = (i_1,i_2,...,i_{|R|})$, $i_1 = i_{|R|} = 0$, and $\{i_2,...,i_{|R|-1}\} \subseteq V'$; that is, each route is a circuit in G that starts and ends at the depot. A route R is feasible if the sum of all customers' demands on R does not exceed the vehicle capacity Q. A route's cost is the sum of the costs (distances) associated with each traversed arc. The aim is to discover feasible routes such that each customer is visited precisely once, and the sum of all route costs is minimized. There are no restrictions on the number of routes in a solution.

7.2 Proposed method

The proposed method was built through the application of the MDM approach in the state-of-the-art Hybrid Genetic Search proposed by Vidal (2022) for the CVRP. This section describes how the MDM elements have been implemented in the proposed method, referred to as MDM-HGS.

The application of the MDM approach elements in our MDM–HGS implementation is similar to that adopted in the multi-start ILS for the HFVRP from (MAIA; PLASTINO; PENNA, 2018). Each pattern is a frequent set of paths connecting customers in the solutions from the elite set E. The difference to the heterogeneous fleet variant is the assignment of a vehicle type to each path, which does not apply to the canonical CVRP.

The Hybrid Genetic Search is an iterative population-based metaheuristic that restarts the population whenever it completes a given number of consecutive iterations without improving the overall best solution.

In MDM-HGS, E is updated whenever a new feasible solution is inserted in the population if E is not full or the new solution is better than the worst solution in E. It is considered *stable* when one of the following criteria is met: (i) E has not been mined yet, and its contents have not changed for a number of consecutive restarts greater than or equal to δR_{MAX}, where δ is a parameter and R_{MAX} is the maximum number of restarts, dynamically estimated based on the time limit, the elapsed time and the number of completed restarts; or (ii) the contents of E have changed after it was last mined, but they have not changed for a number of consecutive restarts greater than or equal to δR_{MAX}.

A new solution generation method is introduced in MDM-HGS. It is based on a ran-

domized version of the Clarke and Wright heuristic (CLARKE; WRIGHT, 1964; SÖRENSEN; ARNOLD; CUERVO, 2019) and uses the paths from a mined pattern to initialize routes.

7.3 Results summary

The methods have been tested on the 141 benchmark instances from CVRPLIB[2] used in the first phase of the challenge. Extensive computational experiments were conducted to compare MDM-HGS and the original method (HGS-CVRP) with respect to the best solution found in a given timeout and the primal integral (PI) – the performance measure used in the challenge, which rewards a balance of convergence speed and solution quality (BERTHOLD, 2013). Each algorithm was run ten times on each instance. Table 10 summarizes the results, showing for each algorithm:

- The global average PI;
- The average percentage gap of the average cost from the BKS;
- The number of wins in average PI, average cost, and best Cost;
- The numbers of best known and new best solutions found.

Three comparisons are presented: one on all 141 instances; one on the 100 instances from the X set (UCHOA et al., 2017), used for tuning parameters; and another on all instances not in the X set. As these comparisons show, MDM-HGS obtains significantly better PI values than HGS-CVRP and solutions of higher quality.

Table 10: Summarized results for the CVRP

	All instances		X instances		All except X instances	
	HGS-CVRP	MDM-HGS	HGS-CVRP	MDM-HGS	HGS-CVRP	MDM-HGS
Global Avg PI	0.5243853	0.4250732	0.1192855	0.1147131	1.5124335	1.1820491
Avg Gap	0.43%	0.33%	0.08%	0.07%	1.30%	0.97%
Wins Avg PI	42	99	32	68	10	31
Wins Avg Cost	37	58	26	38	11	20
Wins Best Cost	18	52	12	33	6	19
No. BKSs	69	71	52	51	17	20
New Best	2	2	0	0	2	2

In the challenge, all competing methods are ranked according to their PI value for each instance. The best method gets 10 points, the second 8, then 6, 5, 4, 3, 2, and 1. The final ranking is based on the total point score obtained by each method on all instances. As CVRP is the most central among vehicle routing problem variants, the CVRP track was

[2] http://vrp.atd-lab.inf.puc-rio.br

the one that attracted more attention in the challenge, with 16 competing teams. Table 11 presents the rankings from Phase 1 for all 141 instances and five subsets separately.

Table 11: Rankings from Phase 1 – 12th DIMACS Implementation Challenge – CVRP track. Source: http://dimacs.rutgers.edu/programs/challenge/vrp/results

All (141)		X (100)		Very Large (10)		Classic (7)		Golden (12)		DIMACS (12)	
Solver	Score	Solver	Score	Solver	Score	Solver	Score	Solver	Score	Solver	Score
FHCSolver	931	**MDM-HGS**	**663**	FSP4D	100	**MDM-HGS**	**56**	**MDM-HGS**	**90**	FHCSolver	83
MDM-HGS	**896**	FHCSolver	642	FHCSolver	80	POP-HGS	45	FHCSolver	90	FSP4D	75
POP-HGS	814	POP-HGS	605	AILS-II	56	FHCSolver	36	POP-HGS	89	POP-HGS	74
FSP4D	737	HGSRR	480	HGSRR	44	FSP4D	31	FSP4D	84	HGSRR	73
HGSRR	678	FSP4D	447	LKHSP	36	HGSRR	23	HGSRR	58	**MDM-HGS**	**62**
AILS-II	558	AILS-II	416	**MDM-HGS**	**25**	ILS-SP	12	MAESN	43	AILS-II	52
MAESN	406	MAESN	352	optaplanner	19	MAESN	11	AILS-II	27	GA+TBD	29
GA+TBD	291	GA+TBD	222	ALNS++	17	AILS-II	7	GA+TBD	26	LKHSP	8
ILS-SP	70	ILS-SP	49	GA+TBD	7	GA+TBD	7	LKHSP	0	ALNS++	8
LKHSP	57	ALNS++	17	ILS-SP	5	LKHSP	6	ALNS++	0	ILS-SP	4
ALNS++	42	LKHSP	7	POP-HGS	1	optaplanner	0	ILS-SP	0	optaplanner	0
optaplanner	19	optaplanner	0	MAESN	0	ALNS++	0	optaplanner	0	Clowder	0
Clowder	0	Clowder	0	Clowder	0	Clowder	0	Clowder	0	MAESN	0
Resilience	0	Resilience	0	Resilience	0	OR-Tools	0	OR-Tools	0	Resilience	0
DDVRP	0	DDVRP	0	OR-Tools	0	Resilience	0	Resilience	0	OR-Tools	0
OR-Tools	0	OR-Tools	0	DDVRP	0	DDVRP	0	DDVRP	0	DDVRP	0

The top-8 competitors from Phase 1 qualified for the second and final phase, in which 100 new instances (kept hidden) were used. Table 12 presents the rankings from Phase 2.

Table 12: Rankings from Phase 2 – 12th DIMACS Implementation Challenge – CVRP track. Source: http://dimacs.rutgers.edu/programs/challenge/vrp/results

Solver	Score	Average PI	Median PI
FHCSolver	643	0.053	0.021
MDM-HGS	**622**	0.067	0.017
AILSII	536	0.034	0.015
POP-HGS	525	0.064	0.024
HGSRR	477	0.067	0.026
FSP4D	458	0.088	0.033
MAESN	389	0.098	0.032
GA+TBD	250	0.171	0.078

MDM-HGS was a close second to FHCSolver in the overall ranking. Some other noticeable results can be observed. Six out of the eight finalist methods are based on the HGS metaheuristic: FHCSolver (JIANG et al., 2022), POP-HGS (QUEIROGA; SADYKOV, 2022), HGSRR (SIMENSEN; HASLE; STÅLHANE, 2022), MAESN (ZHENG et al., 2022) and GA+TBD (VOIGT et al., 2022), apart from MDM-HGS. MDM-HGS overcame all of these other methods in the overall ranking except for FHCSolver. Furthermore, besides being close to FHCSolver in the overall ranking, MDM-HGS ranked first in two out of the five instance subsets in Phase 1.

These results show that MDM-HGS stands as a competitive method in the CVRP state-of-the-art, overcoming the original HGS-CVRP and many other novel algorithms.

8 Ensembles of classifiers for uncertain categorical data and their applications in bioinformatics

This chapter discusses three approaches proposed for building classifier ensembles for uncertain categorical data. The first approach, named Biased Random Subspaces, has been proposed in (MAIA; PLASTINO; FREITAS, 2021), a paper published in the proceedings of the 2021 edition of the *IEEE International Conference on Data Mining*, presented in Appendix F. The other two approaches, named Biased Bootstrap and Biased Splitting, have been proposed in a paper submitted to the *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, presented in Appendix G. Their application in two bioinformatics domains – ageing-related genes and drug side effects – produced evidence that they can effectively provide classifier ensembles with enhanced predictive performance on uncertain categorical data. Thus, the associated contributions are two-fold: the approaches constitute a methodological contribution to the data mining field, whereas the results obtained constitute contributions to the bioinformatics field. Section 8.1 describes the proposed approaches, whereas Section 8.2 reports a summary of the obtained results.

8.1 Proposed approaches

Data uncertainty can be categorised into existential uncertainty, which occurs when the existence of some instance is uncertain, and value uncertainty, which can be further categorised into class-label uncertainty or feature-value uncertainty. This work addresses feature-value uncertainty, which occurs when some feature values in an instance are not precisely known. This uncertainty can naturally arise due to the limited precision of data collection technology, particularly in bioinformatics or biomedical domains. An uncertain feature value is usually represented by a probability distribution on the corresponding feature's domain.

It has been shown that incorporating information on uncertainty into classification

algorithms can improve predictive performance (GE; XIA; NADUNGODAGE, 2010; TSANG et al., 2011; ANGIULLI; FASSETTI, 2013; XIE; XU; HU, 2018), but this is still an under-explored research topic, particularly for categorical features, since most previous methods focus on uncertain numerical features. Hence, this work proposes new ensemble methods for coping with uncertain categorical features.

This work focuses on ensemble methods that learn many base classifiers independently on random subsets of the original training set and then aggregate the base classifiers' predictions. Such ensemble methods usually have better predictive performance and are more robust to slight data variations than any single base classifier. In particular, Bagging methods randomly sample subsets of the instances in the dataset with replacement, which is called bootstrap sampling (BREIMAN, 1996), and Random Subspaces methods randomly sample subsets of the features in the dataset (HO, 1998).

Let $F = \{f_1, f_2, \ldots, f_m\}$ be the set of predictive features, where $m \geq 1$, and $C = \{c_1, c_2, \ldots, c_q\}$ be the set of classes, where $q \geq 2$. The domain of a feature f_j is $dom(f_j)$. A dataset $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$ consists of n labelled instances. Each instance in D, identified by an index i, is associated with a feature vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and a class label $y_i \in C$. In the classification problem, the objective is to construct a model from D capable of predicting the class of an unlabelled instance given its corresponding feature vector.

Let $U \subseteq F$ be the set of uncertain features, all of which are assumed to be categorical in this work. When f_j is a categorical feature, its domain is a finite set of values $dom(f_j) = \{v_{j1}, v_{j2}, \ldots, v_{j|dom(f_j)|}\}$, $|dom(f_j)| \geq 2$. If a feature f_j is not uncertain, its corresponding value x_{ij} for an instance i is represented by a single value. Otherwise it is a discrete probability distribution represented by a probability vector P_{ij}. That is:

$$x_{ij} = \begin{cases} x_{ij} \in dom(f_j), & \text{if } f_j \in F \setminus U \\ P_{ij} = (p_{ij1}, p_{ij2}, \ldots, p_{ij|dom(f_j)|}), & \text{otherwise} \end{cases}$$

where p_{ijk} represents the probability that x_{ij} takes the value v_{jk} and $\sum_{k=1}^{|dom(f_j)|} p_{ijk} = 1$.

This work proposes new approaches for building ensembles of classifiers that incorporate uncertainty about the value of categorical features into the model. The intuition motivating these proposals is that the higher the degree of uncertainty for a given feature (or instance), the less it might contribute to the predictive performance as it provides less reliable information. Furthermore, missing values are also accounted for since they represent another factor that may undermine the contribution of a feature (or instance)

to the model.

A bias value is computed for each feature f_j based on its degree of uncertainty and on its fraction of missing values in the dataset, given by:

$$b_{*j} = \left(1 - \frac{1}{|I \setminus M_{*j}|} \sum_{i \in I \setminus M_{*j}} E_{ij}\right) \times \frac{|I \setminus M_{*j}|}{|I|}$$

where $I = \{1,2,\ldots,n\}$ is the set of indices of all instances in D, M_{*j} is the set of indices of instances in D with a missing value for feature f_j, and E_{ij} is the normalized entropy of the probability distribution represented by P_{ij} if f_{ij} is an uncertain feature (or zero, otherwise), that is:

$$E_{ij} = \begin{cases} \frac{\sum_{k=1}^{|dom(f_j)|} p_{ijk} log(p_{ijk})}{log(1/|dom(f_j)|)}, & \text{if } f_j \in U \\ 0, & \text{otherwise} \end{cases}$$

In the feature bias definition, the first factor (between parentheses) is the complement of the mean entropy over all probability distributions associated with feature f_j, whereas the second factor is the fraction of non-missing values for the feature. Therefore, the computed bias is a value in the range [0,1] with higher values indicating lower uncertainty degrees, i.e., more reliable features.

The feature bias values are normalized over all features, defining a probability distribution $B = (\beta_1,\beta_2,\ldots,\beta_m)$, where a probability β_j associated with a feature f_j is given by $\beta_j = b_{*j}/(\sum_{l=1}^{m} b_{*l})$.

Recall that in the general Random Subspaces strategy, each base classifier in the ensemble is trained with a different set of features, sampled from F. The first proposed approach, named *Biased Random Subspaces* (BRS), uses the probability distribution B to sample the features to be considered by each base classifier in the ensemble instead of the default uniform distribution.

Random Forests usually do not sample features before generating each tree. Nonetheless, they sample a subset of features to be considered as candidate features when splitting each node of a tree. Hence, another approach, named *Biased Splitting* (BS), is proposed for building Random Forests. Like the BRS approach, it uses the probability distribution B to sample the candidate features.

Finally, a third approach, named *Biased Bootstrap* (BB), is proposed for instance sampling analogously to the proposal of the BRS and BS aproaches for feature sampling.

Hence, the bias value for an instance identified by index i is defined as:

$$b_{i*} = \left(1 - \frac{1}{|F \setminus M_{i*}|} \sum_{f_j \in F \setminus M_{i*}} E_{ij}\right) \times \frac{|F \setminus M_{i*}|}{|F|}$$

where M_{i*} is the set of features in D with a missing value for instance i.

The instance bias values are normalized over all instances, defining a probability distribution $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$, where a probability γ_i associated with an instance identified by index i is given by $\gamma_i = b_{i*}/(\sum_{l=1}^{n} b_{l*})$.

The BB approach uses the probability distribution Γ to sample the instances to be used for training each base classifier.

Note that no assumption is made about if or how the base classifiers in the ensemble handle uncertain data, as the focus is on the BRS, BS, and BB approaches to cope with uncertainty at the ensemble level. Even if the base classifiers do not cope with uncertainty, this approach can still be straightforwardly applied. As an example, it can be done by replacing each probability distribution P_{ij} corresponding to an uncertain feature in the dataset with its expected value, i.e., the value v_{jk} that maximises p_{ijk}.

8.2 Results summary

The proposed approaches have been evaluated on real data from two application domains. The first domain is the classification of ageing-related genes regarding their effect on the lifespan of an organism, which may be positive (pro-longevity) or negative (anti-longevity). For this domain, four new datasets have been generated by integrating data from the GenAge database (TACUTU et al., 2017) and the STRING database (SZKLARCZYK; GABLE, et al., 2018). Each dataset contains data regarding ageing-related genes of one of the four major model organisms from the GenAge database: *C. elegans* (roundworm), *D. melanogaster* (fruit fly), *M. musculus* (mouse), and *S. cerevisiae* (baker's yeast). Each instance in these datasets refers to an ageing-related gene of the corresponding model organism and consists of uncertain features referring to protein-protein interactions and a binary class variable indicating if the instance is positive (pro-longevity gene) or negative (anti-longevity gene) according to the GenAge database. Each protein-protein interaction (PPI) feature refers to one protein and has a binary domain, indicating whether or not an interaction between the protein encoded by the corresponding gene (the current instance) and the protein referred by the feature has been observed.

The second domain involves the prediction of drugs' side effects. For this domain, six new datasets have been generated by integrating data from the SIDER database (KUHN et al., 2015) and the STITCH database (SZKLARCZYK; SANTOS, et al., 2015). Each side-effect dataset refers to one of the six most frequent side effects in the SIDER database: nausea, headache, dermatitis, rash, vomiting, and dizziness. Each instance in these datasets refers to a drug and consists of uncertain features referring to protein-chemical interactions and a binary class variable indicating whether the corresponding drug has the side effect represented in the dataset (positive) or not (negative). Each protein-chemical interaction (PCI) feature refers to one protein and has a binary domain, indicating whether or not an interaction between the corresponding chemical (drug) and the protein referred by the feature has been observed.

A value x_{ij} of an uncertain binary feature f_j for an instance i in a dataset is represented by a probability distribution $P_{ij} = (p_{ij1},p_{ij2})$, where p_{ij1} and p_{ij2} are the complementary probabilities of x_{ij} taking each of the two values in $dom(f_j)$. Therefore, each probability distribution representing a PPI or PCI feature value is encoded by a single value p_{ij}, and $P_{ij} = (p_{ij},1-p_{ij})$. In the generated datasets, this value p_{ij} is the confidence score (interaction probability) obtained from the STRING or STITCH databases for the corresponding PPI or PCI features, respectively.

These datasets are particularly challenging for having many features, a small number of instances, and a very high percentage of missing values (when there is no information regarding a specific interaction in the STRING or STITCH databases). PPI and PCI features with low support (annotating less than ten instances) have been discarded to avoid overfitting. As usual in the literature using PPIs as predictive features for classifying genes, missing values are represented as zeros.

In the experiments, three baseline ('uncertainty-unaware') ensemble methods were considered: two kinds of ensembles of Naive Bayes (NB) classifiers and one Random Forest (RF). NB has obtained good results in many real-world domains, including the classification of ageing-related genes (WAN; FREITAS, 2013; SILVA; PLASTINO; FREITAS, 2018; SILVA, P. N. et al., 2021), and RF is currently one of the most powerful classification algorithms.

In ENB-NV (Ensemble of NB with Numeric Values), the NB classifiers treat each uncertain value (an interaction probability) as a numeric value and assume that the feature values' probability distributions are Gaussian. In ENB-EV (Ensemble of NB with Expected Values), the NB classifiers binarise each uncertain value using the threshold 0.5

and consider multivariate Bernoulli distributions for the data.

In the baseline Random Forest, RF-DFE (RF Distributing Fractions of Examples), the decision trees distribute fractions of examples over the child nodes when splitting a node that is assigned an uncertain feature, a common approach for handling uncertain data in decision trees (DUDAS; BOSTRÖM, 2009; QIN; XIA; LI, 2009; TSANG et al., 2011).

The baseline ensembles use conventional strategies for sampling instances and features. Three versions have been built for each by incorporating different combinations of the proposed approaches: BB, BRS and BB+BRS for NB ensembles; BB, BS and BB+BS for Random Forests.

The predictive performance of the algorithms has been assessed using two metrics: the Area Under the Receiver Operating Characteristic curve (AUROC) and the geometric mean of sensitivity and specificity (G-mean) (JAPKOWICZ; SHAH, 2011). Each algorithm was evaluated using the well-known 10-fold cross-validation.

A series of five experiments have been conducted. Experiment 1 compared ENB-NV and methods based on it applying the proposed approaches. The obtained rankings are presented in Table 13, which shows that ENB-NV+BRS was the best regarding both metrics.

Table 13: Average ranks obtained in Experiment 1

AUROC		G-mean	
Method	Avg. Rank	Method	Avg. Rank
ENB-NV+BRS	1.6	ENB-NV+BRS	1.6
ENB-NV	2.5	ENB-NV+BB	2.4
ENB-NV+BB	2.7	ENB-NV+BB+BRS	2.8
ENB-NV+BB+BRS	3.2	ENB-NV	3.2

Experiment 2 compared ENB-EV and methods based on it applying the proposed approaches. The obtained rankings are presented in Table 14, which shows that ENB-EV+BRS was the best regarding both metrics.

Table 14: Average ranks obtained in Experiment 2

AUROC		G-mean	
Method	Avg. Rank	Method	Avg. Rank
ENB-EV+BRS	1.4	ENB-EV+BRS	1.6
ENB-EV	2.4	ENB-EV	2.2
ENB-EV+BB+BRS	3.1	ENB-EV+BB+BRS	2.6
ENB-EV+BB	3.1	ENB-EV+BB	3.6

Experiment 3 compared the best NB ensembles regarding each of the AUROC and

G-mean metrics. The obtained rankings are presented in Table 15, which shows that ENB-EV+BRS was the best regarding AUROC, whereas ENB-NV+BRS was the best regarding G-mean.

Table 15: Average ranks obtained in Experiment 3

AUROC		G-mean	
Method	Avg. Rank	Method	Avg. Rank
ENB-EV+BRS	1.4	ENB-NV+BRS	1.0
ENB-NV+BRS	1.6	ENB-EV+BRS	2.0

Experiment 4 compared RF-DFE and methods based on it applying the proposed approaches. The obtained rankings are presented in Table 16, which shows that RF-DFE+BB was the best regarding AUROC, whereas RF-DFE+BS was the best regarding G-mean.

Table 16: Average ranks obtained in Experiment 4

AUROC		G-mean	
Method	Avg. Rank	Method	Avg. Rank
RF-DFE+BB	2.0	RF-DFE+BS	2.0
RF-DFE+BB+BS	2.1	RF-DFE+BB+BS	2.3
RF-DFE	2.7	RF-DFE	2.4
RF-DFE+BS	3.2	RF-DFE+BB	3.1

Finally, Experiment 5 compared the best NB ensemble and the best Random Forest regarding each of the AUROC and G-mean metrics. The obtained rankings are presented in Table 17, which shows that the RF-based methods overcame the NB-based ones, with RF-DFE+BB being the best regarding AUROC and RF-DFE+BS being the best regarding G-mean.

Table 17: Average ranks obtained in Experiment 5

AUROC		G-mean	
Method	Avg. Rank	Method	Avg. Rank
RF-DFE+BB	1.1	RF-DFE+BS	1.3
ENB-EV+BRS	1.9	ENB-NV+BRS	1.7

As a general conclusion from all these experiments, the results support the hypothesis that the approaches BB, BRS and BS improve the predictive performance of ensembles on uncertain data. The BB and BS approaches proposed in this work obtained the best overall results since RF-DFE+BB and RF-DFE+BS were the best overall ensembles for the AUROC and G-mean metrics, respectively.

9 Interpretability approaches for Naive Bayes ensembles and their applications in bioinformatics

This chapter discusses two approaches proposed for adding interpretability to ensembles of NB classifiers, which have been proposed in a paper submitted to *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, presented in Appendix G. They have been applied to identify relevant protein interactions to classify ageing-related genes, producing results consistent with current biological knowledge and new insights in this context. Thus, the associated contributions are two-fold: the approaches constitute a methodological contribution to the data mining field, whereas the results obtained constitute contributions to the bioinformatics field. Section 9.1 describes the proposed approaches, whereas Section 9.2 reports a summary of the obtained results.

9.1 Proposed approaches

The first proposed approach for interpreting an ensemble of NB classifiers relies on the influence that a feature value $x_{ij} \in dom(f_j)$ has for determining the most likely class to be predicted for an instance by a single NB classifier, which is computed as an importance score. Then, the importance scores obtained from all the classifiers are combined into the ensemble's importance scores.

Given a feature vector $X_i = (x_{i1},x_{i2},\ldots,x_{im})$ associated with an unlabelled instance identified by index i, an NB classifier predicts the class $y \in C$ that maximizes the value given by $P(y|X_i) \propto P(y)\prod_{j=1}^{m} P(x_{ij}|y)$.

The definition of importance is firstly presented for binary classifiers, where $C = \{c_1,c_2\}$. The importance of a feature value x_{ij} in a given NB classifier in the ensemble is estimated by the following difference of conditional probabilities:

$$Diff(x_{ij},c_1,c_2,e) = |P(x_{ij}|c_1) - P(x_{ij}|c_2)|$$

where e is the classifier for which the difference is computed.

The higher the difference in the class-conditioned probability of a feature value between the two class labels, the more importance (influence) that feature value will have for determining the most likely class to be assigned to the testing instance. For datasets with more than two class labels, this idea can be generalised by summing the differences between all pairs of class labels in C:

$$Importance(x_{ij},C,e) = \sum_{r=1}^{q-1} \sum_{s=r+1}^{q} Diff(x_{ij},c_r,c_s,e)$$

The importance of a feature value x_{ij} for an ensemble of NB classifiers is computed by averaging its importance across all classifiers in the ensemble. However, different NB classifiers will generally use different feature subsets (due to the random subspaces approach). Intuitively, other things being equal, the larger the number of classifiers in the ensemble that use a feature, the larger the importance of a value of that feature. Therefore, we assume that, if a classifier e_u does not use a feature f_j, then $Importance(x_{ij},C,e_u) = 0$. Hence, the ensemble-wide importance is defined as:

$$Importance(x_{ij},C) = \frac{\sum_{u=1}^{t} Importance(x_{ij},C,e_u)}{t}$$

where e_u is the u-th classifier and t is the total number of classifiers in the ensemble.

This equation is appropriate when the predicted class returned by the ensemble is computed by a simple majority vote of all classifiers, i.e., all classifiers have the same weight in the voting. If weighted voting is used instead (where the weight of a vote is proportional to the classifier's confidence in its prediction), then the importance equation could be easily modified to compute a correspondingly weighted average over the t classifiers.

Finally, once the importance value has been computed for all feature values x_{ij}, all feature values are ranked in decreasing order of importance. Then a user (domain expert) can focus on interpreting the top-ranked feature values, i.e., the most important ones for predicting the class variable in the ensemble.

Note that in the case of binary domain features, where $dom(f_j) = \{v_{j1},v_{j2}\}$, the importance computed for both values will be the same due to the complementarity of probabilities in use. Given two class labels c_r and c_s such that $c_r \in C$, $c_s \in C$ and $c_r \neq c_s$,

the following relations apply:

$$P(v_{j1}|c_r) = 1 - P(v_{j2}|c_r) \tag{9.1}$$

$$P(v_{j1}|c_s) = 1 - P(v_{j2}|c_s) \tag{9.2}$$

The difference of the class-conditioned probabilities of the value v_{j1} between c_r and c_s for a classifier e would be:

$$Diff(v_{j1},c_r,c_s,e) = |P(v_{j1}|c_r) - P(v_{j1}|c_s)| \tag{9.3}$$

By replacing (9.1) and (9.2) in (9.3), we obtain:

$$\begin{aligned} Diff(v_{j1},c_r,c_s,e) &= |(1 - P(v_{j2}|c_r)) - (1 - P(v_{j2}|c_s))| \\ &= |1 - P(v_{j2}|c_r) - 1 + P(v_{j2}|c_s)| \\ &= |P(v_{j2}|c_s) - P(v_{j2}|c_r)| \\ &= |P(v_{j2}|c_r) - P(v_{j2}|c_s)| \\ &= Diff(v_{j2},c_r,c_s,e) \end{aligned}$$

Therefore, in the case of binary domain features (like the PPI and PCI features in the datasets generated in this work), the importance computed for both values in the domain would be the same. Then, the importance computed for any of the values in a feature's domain can be interpreted as that feature's importance.

The approach based on conditional probability differences measures the importance of each feature value separately, ignoring its importance in the context of all other feature values. This is consistent with NB assuming that each feature is independent of all others conditioned on the class variable, but it does not directly measure the influence of a feature value on class prediction. Naive Bayes makes class predictions using the formula: $P(y|X_i) \propto P(y) \prod_{j=1}^{m} P(x_{ij}|y)$. Therefore, whether or not a conditional probability will make a difference in the choice of the predicted class depends on the entire set of conditional probabilities and the prior class probability.

Hence, a second approach that considers sets of feature values is proposed. The basic principle is the same adopted in the definitions of *anchors* by Ribeiro, Singh, and Guestrin (2018) and *minimal sufficient factors* by Watson et al. (2021). The aim is to find, for each instance, a minimal set of features that is sufficient to preserve the class prediction, such that changes to the other feature values of the instance would not change the class

predicted by the model.

Note that the larger the value of $Importance(x_{ij}, C, e)$, the higher the influence of x_{ij} for a class prediction in model e, but even the feature with the highest $Importance$ value may still not be sufficient for a given prediction.

However, this notion can be used to find a minimal sufficient set of features. The basic idea is to sort all features in increasing order of their $Importance$ values and then identify the minimal set of top features in that sorted list which, together, are sufficient for preserving the class prediction made by the classifier.

Let c_i be the class predicted by the classifier for instance i. A feature value x_{ij} is said to "support" the prediction of c_i if and only if $P(x_{ij}|c_i) > P(x_{ij}|y), \forall y \in C \setminus \{c_i\}$. That is, the feature value x_{ij} becomes more likely if instance i has class c_i than if that instance has another class. Naturally, when searching for a minimal sufficient set of features, only feature values that support the class predicted by the classifier need to be considered.

A method for identifying a minimal sufficient set of features for the class prediction for a given instance is presented in Algorithm 1.

Algorithm 1 MinimalSufficientSet(X_i, e)

$c_i \leftarrow$ class predicted by e for X_i
$SuppSet \leftarrow \{f_j | P(x_{ij}|c_i) > P(x_{ij}|y), \forall y \in C \setminus \{c_i\}\}$
$S \leftarrow SuppSet$
Calculate $Importance(x_{ij}, C, e)$ for all $f_j \in S$
$SortedFeats \leftarrow$ SortByImportance($SuppSet$)
$e' \leftarrow e$
for each feature f_j in $SortedFeats$ **do**
    Remove f_j from e'
    $c'_i \leftarrow$ class predicted by e' for X_i
    if $c'_i = c_i$ **then**
        $S \leftarrow S \setminus \{f_j\}$
    else
        Exit loop
    end if
end for
return S

Based on the sufficiency criterion, a measure of the importance of a feature f_j for a classifier e given the set of instances X, denoted $SImportance(f_j, e, X)$, is defined as the proportion of instances in X for which f_j is in the minimal sufficient set returned by Algorithm 1.

The importance of a feature for the entire ensemble is computed by simply averaging

its importance across all classifiers in the ensemble:

$$SImportance(f_j, X) = \frac{\sum_{u=1}^{t} SImportance(f_j, e_u, X)}{t}$$

where e_u is the u-th classifier and t is the total number of classifiers in the ensemble.

9.2 Results summary

The proposed approaches for interpreting NB ensembles have been applied to the four ageing-related datasets mentioned in Chapter 8 to identify the top-ranked PPI features for classification.

Among all NB ensembles evaluated, ENB-EV+BRS and ENB-NV+BRS achieved the best overall AUROC and G-mean values, respectively. ENB-EV+BRS has been selected for model interpretation since it uses binarised features, facilitating interpretation.

A model has been trained for each dataset by applying ENB-EV+BRS to all instances. Then, these models were used to produce rankings of features in decreasing order of importance.

Table 18 presents the top-10 features for each organism (dataset) regarding the importance measure based on conditional probabilities, whereas Table 19 presents the top-10 features regarding the measure based on minimal sufficient features.

For each feature, columns 2–5 present its importance-based rank, the corresponding protein ID from the STRING database, and the corresponding gene's symbol and name.

Columns 6 and 7 present, each one, the absolute and relative frequencies of the feature value 1 (considering the threshold 0.5 used) for the corresponding feature in the instances of the Pro-longevity and Anti-longevity classes, respectively. These values can reveal whether a PPI is related to a pro- or anti-longevity effect.

The last column shows whether or not the feature is involved in occurrences of Simpson's paradox (PEARL, 2009, 2014). An occurrence of Simpson's paradox for a feature means that the direction of association between the feature and the class variable (i.e., whether the occurrence of a feature value increases or decreases the probability of a class label) is reversed when we consider a combination of the feature and a confounder variable.

The top-ranked features have been analysed regarding their relevance for the biology of ageing by Professor João Pedro de Magalhães from the University of Liverpool, a

Table 18: Top-10 PPI features in the ENB-EV+BRS model according to the importance measure based on conditional probabilities

Organ.	Feat. Rank	STRING ID	Gene Symbol	Protein Name	Freq. in Pro-long.	Freq. in Anti-long.	SP
Worm	1	R13H8.1h	daf-16	Forkhead box protein O	40 (15.6%)	43 (8.5%)	no
	2	F42G8.12	isp-1	Cytochrome b-c1 complex subunit Rieske, mitochondrial	9 (3.5%)	58 (11.5%)	no
	3	C34E10.6.1	atp-2	ATP synthase subunit beta, mitochondrial	6 (2.3%)	58 (11.5%)	no
	4	T05E11.1	rps-5	40S ribosomal protein S5	5 (1.9%)	60 (11.9%)	no
	5	F40F11.1.2	rps-11	Ribosomal protein, small subunit	4 (1.6%)	53 (10.5%)	no
	6	C26F1.4.2	rps-30	40S ribosomal protein S30	5 (1.9%)	43 (8.5%)	no
	7	B0250.1	rpl-2	60S ribosomal protein L8	3 (1.2%)	44 (8.7%)	no
	8	B0393.1.1	rps-0	40S ribosomal protein SA	6 (2.3%)	51 (10.1%)	no
	9	H28O16.1a	H28O16.1	ATP synthase subunit alpha, mitochondrial	6 (2.3%)	56 (11.1%)	no
	10	Y56A3A.19	Y56A3A.19	Acyl carrier protein	2 (0.8%)	51 (10.1%)	no
Fly	1	FBpp0082516	Hsc70-4	Heat shock 70 kDa protein cognate 4	34 (29.3%)	9 (13.0%)	no
	2	FBpp0305736	Sod	Superoxide dismutase [Cu-Zn]	30 (25.9%)	4 (5.8%)	no
	3	FBpp0081956	Hsp70Ab	Heat shock protein 70Ab	22 (19.0%)	4 (5.8%)	no
	4	FBpp0293589	foxo	Forkhead box protein O	36 (31.0%)	14 (20.3%)	no
	5	FBpp0081986	Hsp70Aa	Major heat shock 70 kDa protein Aa	23 (19.8%)	4 (5.8%)	no
	6	FBpp0070899	schlank	Schlank, isoform A	31 (26.7%)	12 (17.4%)	no
	7	FBpp0086226	Sod2	Superoxide dismutase [Mn], mitochondrial	31 (26.7%)	8 (11.6%)	no
	8	FBpp0088134	CaMKI	Calmodulin-dependent protein kinase activity	29 (25.0%)	10 (14.5%)	no
	9	FBpp0077974	park	E3 ubiquitin-protein ligase parkin	20 (17.2%)	1 (1.4%)	no
	10	FBpp0305095	Hsp83	Heat shock protein 83	26 (22.4%)	10 (14.5%)	yes
Mouse	1	ENSMUSP00000056668	Igf-1	Insulin-like growth factor 1	10 (19.6%)	16 (51.6%)	no
	2	ENSMUSP00000029175	Src	Neuronal proto-oncogene tyrosine-protein kinase Src	3 (5.9%)	12 (38.7%)	no
	3	ENSMUSP00000050683	Foxo3	Forkhead box protein O3	8 (15.7%)	13 (41.9%)	no
	4	ENSMUSP00000055308	Foxo1	Forkhead box protein O1	5 (9.8%)	11 (35.5%)	no
	5	ENSMUSP00000000369	Rem1	GTP-binding protein REM 1	7 (13.7%)	10 (32.3%)	yes
	6	ENSMUSP00000101315	Pik3cd	Phosphatidylinositol 4,5-bisphosphate 3-kinase catalytic subunit delta isoform	1 (2.0%)	7 (22.6%)	no
	7	ENSMUSP00000102538	Ngf	Beta-nerve growth factor	5 (9.8%)	8 (25.8%)	yes
	8	ENSMUSP00000031697	Cul1	Cullin-1	6 (11.8%)	7 (22.6%)	no
	9	ENSMUSP00000120152	Stat3	Signal transducer and activator of transcription 3	11 (21.6%)	14 (45.2%)	no
	10	ENSMUSP00000115578	Ubc	Polyubiquitin-C	15 (29.4%)	4 (12.9%)	no
Yeast	1	YLR167W	RPS31	Fusion-protein cleaved to yield ribosomal protein S31 and ubiquitin	0 (0.0%)	58 (17.3%)	no
	2	YIL133C	RPL16A	Ribosomal 60S subunit protein L16A	0 (0.0%)	51 (15.2%)	no
	3	YBR048W	RPS11B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	52 (15.5%)	no
	4	YPL090C	RPS6A	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	51 (15.2%)	no
	5	YGL103W	RPL28	Ribosomal 60S subunit protein L28	0 (0.0%)	61 (18.2%)	no
	6	YNL096C	RPS7B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	50 (14.9%)	no
	7	YJR145C	RPS4A	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	51 (15.2%)	no
	8	YNL069C	RPL16B	Ribosomal 60S subunit protein L16B	0 (0.0%)	48 (14.3%)	no
	9	YBR031W	RPL4A	Ribosomal 60S subunit protein L4A	1 (2.2%)	53 (15.8%)	no
	10	YNL302C	RPS19B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	47 (14.0%)	no

microbiologist who leads the Integrative Genomics of Ageing Group[1], whose research broadly focuses on understanding the genetic, cellular, and molecular mechanisms of ageing. According to his analysis, the top-ranked features fit well with current knowledge

[1] http://rejuvenomicslab.com

Table 19: Top-10 PPI features in the ENB-EV+BRS model according to the importance measure based on sufficiency

Organ.	Feat. Rank	STRING ID	Gene Symbol	Protein Name	Freq. in Pro-long.	Freq. in Anti-long.	SP
Worm	1	F42G8.12	isp-1	Cytochrome b-c1 complex subunit Rieske, mitochondrial	9 (3.5%)	58 (11.5%)	no
	2	C26F1.4.2	rps-30	40S ribosomal protein S30	5 (1.9%)	43 (8.5%)	no
	3	C34E10.6.1	atp-2	ATP synthase subunit beta, mitochondrial	6 (2.3%)	58 (11.5%)	no
	4	B0250.1	rpl-2	60S ribosomal protein L8	3 (1.2%)	44 (8.7%)	no
	5	F40F11.1.2	rps-11	Ribosomal protein, small subunit	4 (1.6%)	53 (10.5%)	no
	6	B0393.1.1	rps-0	40S ribosomal protein SA	6 (2.3%)	51 (10.1%)	no
	7	T05E11.1	rps-5	40S ribosomal protein S5	5 (1.9%)	60 (11.9%)	no
	8	F28D1.7.1	rps-23	40S ribosomal protein S23	4 (1.6%)	48 (9.5%)	no
	9	C49H3.11.1	rps-2	40S ribosomal protein S2	7 (2.7%)	57 (11.3%)	no
	10	Y56A3A.19	Y56A3A.19	Acyl carrier protein	2 (0.8%)	51 (10.1%)	no
Fly	1	FBpp0305736	Sod	Superoxide dismutase [Cu-Zn]	30 (25.9%)	4 (5.8%)	no
	2	FBpp0081956	Hsp70Ab	Heat shock protein 70Ab	22 (19.0%)	4 (5.8%)	no
	3	FBpp0082516	Hsc70-4	Heat shock 70 kDa protein cognate 4	34 (29.3%)	9 (13.0%)	no
	4	FBpp0081986	Hsp70Aa	Major heat shock 70 kDa protein Aa	23 (19.8%)	4 (5.8%)	no
	5	FBpp0086226	Sod2	Superoxide dismutase [Mn], mitochondrial	31 (26.7%)	8 (11.6%)	no
	6	FBpp0293589	foxo	Forkhead box protein O	36 (31.0%)	14 (20.3%)	no
	7	FBpp0077974	park	E3 ubiquitin-protein ligase parkin	20 (17.2%)	1 (1.4%)	no
	8	FBpp0070899	schlank	Schlank, isoform A	31 (26.7%)	12 (17.4%)	no
	9	FBpp0088134	CaMKI	Calmodulin-dependent protein kinase activity	29 (25.0%)	10 (14.5%)	no
	10	FBpp0078604	Aux	Auxilin, isoform A	17 (14.7%)	3 (4.3%)	no
Mouse	1	ENSMUSP00000056668	Igf-1	Insulin-like growth factor 1	10 (19.6%)	16 (51.6%)	no
	2	ENSMUSP00000029175	Src	Neuronal proto-oncogene tyrosine-protein kinase Src	3 (5.9%)	12 (38.7%)	no
	3	ENSMUSP00000050683	Foxo3	Forkhead box protein O3	8 (15.7%)	13 (41.9%)	no
	4	ENSMUSP00000055308	Foxo1	Forkhead box protein O1	5 (9.8%)	11 (35.5%)	no
	5	ENSMUSP00000101553	Ins2	Insulin-2	6 (11.8%)	13 (41.9%)	no
	6	ENSMUSP00000099878	Rps6	40S ribosomal protein S6	3 (5.9%)	8 (25.8%)	no
	7	ENSMUSP00000021090	Grb2	Growth factor receptor-bound protein 2	3 (5.9%)	8 (25.8%)	no
	8	ENSMUSP00000099621	Rpa2	Replication protein A 32 kDa subunit	12 (23.5%)	1 (3.2%)	no
	9	ENSMUSP00000120152	Stat3	Signal transducer and activator of transcription 3	11 (21.6%)	14 (45.2%)	no
	10	ENSMUSP00000102538	Ngf	Beta-nerve growth factor	5 (9.8%)	8 (25.8%)	yes
Yeast	1	YLR167W	RPS31	Fusion-protein cleaved to yield ribosomal protein S31 and ubiquitin	0 (0.0%)	58 (17.3%)	no
	2	YNL096C	RPS7B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	50 (14.9%)	no
	3	YJR145C	RPS4A	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	51 (15.2%)	no
	4	YGL103W	RPL28	Ribosomal 60S subunit protein L28	0 (0.0%)	61 (18.2%)	no
	5	YBR048W	RPS11B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	52 (15.5%)	no
	6	YKR094C	RPL40B	Ubiquitin-ribosomal 60S subunit protein L40B fusion protein	0 (0.0%)	59 (17.6%)	no
	7	YIL133C	RPL16A	Ribosomal 60S subunit protein L16A	0 (0.0%)	51 (15.2%)	no
	8	YGL030W	RPL30	Ribosomal 60S subunit protein L30	0 (0.0%)	50 (14.9%)	no
	9	YGL123W	RPS2	Protein component of the small (40S) subunit	1 (2.2%)	58 (17.3%)	no
	10	YKL009W	MRT4	Protein involved in mRNA turnover and ribosome assembly	0 (0.0%)	55 (16.4%)	no

on longevity/ageing and previous similar analyses, and new insights into the genetics of ageing can be drawn from the rankings, as follows.

Overall, the top-ranked features fit nicely into pro- and anti-longevity pathways enriched in GenAge (FERNANDES et al., 2016). Of note, in worms, ribosomal proteins and

mitochondrial proteins involved in oxidative phosphorylation have been previously found enriched in anti-longevity processes (FERNANDES et al., 2016), while in flies, responses to oxidative stress (like antioxidant enzymes) are among enriched pro-longevity processes. In mice, the insulin signalling pathway is a top enriched anti-longevity pathway (FERNANDES et al., 2016), in line with these results.

Out of the 40 top-ranked PPI features in Table 18, interestingly, 15 represent ribosomal proteins, namely 5 of the top-10 PPI features for the worm dataset and all the top-10 PPI features for the yeast dataset. It is also worth observing in Table 18 the relative frequency of genes (dataset instances) of each class (Pro- vs Anti-longevity) that interact with the gene associated with each of these 15 ribosomal proteins (PPI features). In all those 15 table rows, the relative frequency of Anti-longevity genes interacting with the corresponding ribosomal protein is substantially higher than that of Pro-longevity genes interacting with that ribosomal protein. The relative frequency differences are particularly striking for 9 of the 10 yeast PPI features in the table, which have a relative frequency of 0% for Pro-longevity genes and relative frequencies varying from 14.0% to 18.2% for Anti-longevity genes.

Interestingly, despite this strong pattern, none of these 9 yeast ribosomal proteins is included in GenAge (TACUTU et al., 2017) – the most comprehensive database of ageing-related genes. By itself, this strong pattern is not sufficient to conclude that those 9 ribosomal proteins have an anti-longevity effect on yeast, which in principle could be confirmed only via appropriate biological experiments. However, the pattern seems strong enough to justify further investigation about some of those 9 ribosomal proteins in future work.

In mice, some brain and neuronal factors (e.g., Src) are also among the top features, which could fit GH/IGF1's neuroendocrine regulation (MAGALHÃES; MATSUDA, 2012). Alternatively, they could be related to ageing changes in the brain. As Src is not in GenAge, it could be an interesting target for future biological studies.

10 Conclusions and future work

This thesis reported a series of research contributions related to the proposal and application of data mining methods in the combinatorial optimization and bioinformatics fields.

A novel approach for incorporating data mining into metaheuristics named MineReduce was introduced and explored. It uses patterns mined from good solutions to help metaheuristics guide the search more effectively and efficiently.

The MineReduce approach was applied to state-of-the-art metaheuristics for different problems, achieving competitive results with solution quality improvement and convergence speedup:

- A MineReduce-based version of a multi-start iterated local search (PENNA; SUBRAMANIAN; OCHI, 2013) for the Heterogeneous Fleet Vehicle Routing Problem;
- A MineReduce-based version of a multi-start iterated tabu search (ZHOU et al., 2016) for the Minimum Weighted Vertex Cover Problem;
- A MineReduce-based version of a hybrid metaheuristic that combines the MDM approach with components of GRASP, ILS and RVND (SILVA, M. M. et al., 2012; SANTANA; PLASTINO; ROSSETI, 2022) for the Minimum Latency Problem.

Furthermore, a MineReduce-based multi-start iterated local search was designed from scratch for the Multi-Source Capacitated Facility Location Problem with Customer Incompatibilities, a new variant of the CFLP, introduced in the MESS 2020+1 Metaheuristics Competition[1]. The proposed MineReduce-based method won the competition by an overwhelming margin, hence taking the lead in the state-of-the-art for this new problem.

Additionally, the MDM approach was applied to the state-of-the-art hybrid genetic search metaheuristic for the Capacitated Vehicle Routing Problem (VIDAL, 2022). The

[1] https://www.ants-lab.it/mess2020/#competition

proposed MDM-based version overcame the original method, achieving higher solution quality and faster convergence. It was registered in the CVRP track of the 12th DIMACS Implementation Challenge[2], the latest edition of one of the most traditional and important series of challenges in the field (BURIOL et al., 2020). The proposed method achieved outstanding results, being a close second in the overall ranking and overcoming other 14 competitors (including four other methods based on the HGS metaheuristic).

Methodological contributions related to the classification task were also established by proposing novel approaches for building classifier ensembles for uncertain categorical data and interpreting ensembles of Naive Bayes classifiers.

Three novel ensemble approaches proposed for uncertain data have been applied in two domains from the bioinformatics field: the classification of ageing-related genes regarding their effect on longevity and the prediction of drug side effects. The obtained results demonstrated that these proposed approaches improve the predictive performance of ensembles on uncertain data.

Two novel approaches for interpreting NB ensembles were applied to identify relevant protein interactions to classify ageing-related genes, producing results consistent with current biological knowledge, evidencing that the proposed approaches provide a solid ground for model interpretation. Furthermore, new insights into the genetics of ageing have been drawn from the obtained results.

The effectiveness of the methods in this thesis has been evidenced not only by the results of extensive experimental analyses reported in this thesis but also by the corresponding published papers, the outstanding results achieved in competitions, and the derived biological insights.

Multiple research paths have been explored in this work. Hence, naturally, multiple future research directions can be drawn from them. The MineReduce approach can be further explored through its application to other combinatorial optimization problems and its hybridization with other metaheuristics. In particular, the success obtained with the hybridization of the MDM approach with HGS suggests the experimentation of a hybrid version of this metaheuristic with the MineReduce approach, which could not be accomplished in this work due to time constraints. The Biased Bootstrap, Biased Random Subspaces and Biased Splitting approaches can be further explored through their application in ensembles with different base models and other uncertain datasets. The approaches proposed for interpreting Naive Bayes ensembles can be used in other

[2] http://dimacs.rutgers.edu/programs/challenge/vrp/cvrp

classification problems to provide users with valuable insights. Finally, crossing an interdisciplinary boundary, new insights raised in this work regarding the potential roles of some proteins in the ageing process deserve investigation through adequate biological experimentation.

References

AIEX, R. M.; RESENDE, M. G. C.; RIBEIRO, C. C. TTT plots: a perl program to create time-to-target plots. *Optimization Letters*, v. 1, n. 4, p. 355–366, 2007. ISSN 1862-4480. DOI: 10.1007/s11590-006-0031-4.

ANGIULLI, F.; FASSETTI, F. Nearest Neighbor-Based Classification of Uncertain Data. *ACM Transactions on Knowledge Discovery from Data*, Association for Computing Machinery, New York, NY, USA, v. 7, n. 1, 2013. ISSN 1556-4681. DOI: 10.1145/2435209.2435210.

AZODI, C. B.; TANG, J.; SHIU, S.-H. Opening the Black Box: Interpretable Machine Learning for Geneticists. *Trends in Genetics*, v. 36, n. 6, p. 442–455, 2020. ISSN 0168-9525. DOI: 10.1016/j.tig.2020.03.005.

BAZGAN, C.; TOUBALINE, S.; TUZA, Z. The most vital nodes with respect to independent set and vertex cover. *Discrete Applied Mathematics*, v. 159, n. 17, p. 1933–1946, 2011. ISSN 0166-218X. DOI: 10.1016/j.dam.2011.06.023.

BERTHOLD, T. Measuring the impact of primal heuristics. *Operations Research Letters*, v. 41, n. 6, p. 611–614, 2013. ISSN 0167-6377. DOI: 10.1016/j.orl.2013.08.007.

BLUM, A. et al. The Minimum Latency Problem. In: PROCEEDINGS OF THE TWENTY-SIXTH ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING. Montreal, Quebec, Canada: Association for Computing Machinery, 1994. (STOC '94), p. 163–171. ISBN 0897916638. DOI: 10.1145/195058.195125.

BLUM, C. et al. Construct, Merge, Solve & Adapt A new general algorithm for combinatorial optimization. *Computers & Operations Research*, v. 68, p. 75–88, 2016. ISSN 0305-0548. DOI: 10.1016/j.cor.2015.10.014.

BOUAMAMA, S.; BLUM, C.; BOUKERRAM, A. A population-based iterated greedy algorithm for the minimum weight vertex cover problem. *Applied Soft Computing*, v. 12, n. 6, p. 1632–1639, 2012. ISSN 1568-4946. DOI: 10.1016/j.asoc.2012.02.013.

BREIMAN, L. Bagging predictors. *Machine Learning*, v. 24, n. 2, p. 123–140, 1996. ISSN 1573-0565. DOI: 10.1007/BF00058655.

BURIOL, L. S. et al. THE GUIDE TO NP-COMPLETENESS IS 40 YEARS OLD: AN HOMAGE TO DAVID S. JOHNSON. *Pesquisa Operacional*, v. 40, e236329, 2020. ISSN 1678-5142. DOI: 10.1590/0101-7438.2020.040.00236329.

CALVET, L. et al. Learnheuristics: hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Open Mathematics*, v. 15, n. 1, p. 261–280, 2017. DOI: 10.1515/math-2017-0029.

CAMPBELL, A. M.; VANDENBUSSCHE, D.; HERMANN, W. Routing for Relief Efforts. *Transportation Science*, v. 42, n. 2, p. 127–145, 2008. DOI: 10.1287/trsc.1070.0209.

CHEN, H. Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega*, v. 56, p. 25–36, 2015. ISSN 0305-0483. DOI: 10.1016/j.omega.2015.03.002.

CLARKE, G.; WRIGHT, J. W. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, v. 12, n. 4, p. 568–581, 1964. DOI: 10.1287/opre.12.4.568.

CORNE, D.; DHAENENS, C.; JOURDAN, L. Synergies between operations research and data mining: The emerging use of multi-objective approaches. *European Journal of Operational Research*, v. 221, n. 3, p. 469–479, 2012. ISSN 0377-2217. DOI: 10.1016/j.ejor.2012.03.039.

CYGAN, M. et al. Kernelization. In: PARAMETERIZED ALGORITHMS. Cham: Springer International Publishing, 2015. p. 17–49. ISBN 978-3-319-21275-3. DOI: 10.1007/978-3-319-21275-3_2.

DELGADILLO, F. J. D.; MONTIEL, O.; SEPÚLVEDA, R. Reducing the size of traveling salesman problems using vaccination by fuzzy selector. *Expert Systems with Applications*, v. 49, p. 20–30, 2016. ISSN 0957-4174. DOI: 10.1016/j.eswa.2015.11.026.

DOWNEY, R. G.; FELLOWS, M. R. Some Ad Hoc Methods: The Methods of Bounded Search Tree and Problem Kernel. In: PARAMETERIZED COMPLEXITY. New York, NY: Springer New York, 1999. p. 29–48. ISBN 978-1-4612-0515-9. DOI: 10.1007/978-1-4612-0515-9_3.

DUDAS, C.; BOSTRÖM, H. Using Uncertain Chemical and Thermal Data to Predict Product Quality in a Casting Process. In: PROCEEDINGS OF THE 1ST ACM SIGKDD WORKSHOP ON KNOWLEDGE DISCOVERY FROM UNCERTAIN DATA. Paris, France: Association for Computing Machinery, 2009. (U '09), p. 57–61. ISBN 9781605586755. DOI: 10.1145/1610555.1610563.

DUHAMEL, C.; LACOMME, P.; PRODHON, C. *A GRASPxELS with Depth First Search Split Procedure for the HVRP*. 2010. Available from: <https://hal.archives-ouvertes.fr/hal-00704498>.

FELLOWS, M. R. et al. What Is Known About Vertex Cover Kernelization? In: *Adventures Between Lower Bounds and Higher Altitudes*. Ed. by H.-J. Böckenhauer, D. Komm and W. Unger. Cham: Springer International Publishing, 2018. p. 330–356. ISBN 978-3-319-98355-4. DOI: 10.1007/978-3-319-98355-4_19.

FERLAND, J. A.; MICHELON, P. The Vehicle Scheduling Problem with Multiple Vehicle Types. *Journal of the Operational Research Society*, Taylor & Francis, v. 39, n. 6, p. 577–583, 1988. DOI: 10.1057/jors.1988.97.

FERNANDES, M. et al. Systematic analysis of the gerontome reveals links between aging and age-related diseases. *Human Molecular Genetics*, v. 25, n. 21, p. 4804–4818, 2016. ISSN 0964-6906. DOI: 10.1093/hmg/ddw307.

FISCHETTI, M.; LAPORTE, G.; MARTELLO, S. The Delivery Man Problem and Cumulative Matroids. *Operations Research*, v. 41, n. 6, p. 1055–1064, 1993. DOI: 10.1287/opre.41.6.1055.

FUNKE, S.; NUSSER, A.; STORANDT, S. Placement of Loading Stations for Electric Vehicles: No Detours Necessary! *Journal of Artificial Intelligence Research*, v. 53, p. 633–658, 2015. ISSN 1076-9757. DOI: 10.1613/jair.4688.

GAVISH, B.; PIRKUL, H. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. *Mathematical Programming*, v. 31, p. 78–105, 1985a. ISSN 1436-4646. DOI: 10.1007/BF02591863.

GAVISH, B.; PIRKUL, H. Zero-one integer programs with few constraints —Lower bounding theory. *European Journal of Operational Research*, v. 21, n. 2, p. 213–224, 1985b. ISSN 0377-2217. DOI: 10.1016/0377-2217(85)90033-5.

GAVISH, B.; SRIKANTH, K. An Optimal Solution Method for Large-Scale Multiple Traveling Salesmen Problems. *Operations Research*, v. 34, n. 5, p. 698–717, 1986. DOI: 10.1287/opre.34.5.698.

GE, J.; XIA, Y.; NADUNGODAGE, C. UNN: A Neural Network for Uncertain Data Classification. In: ZAKI, M. J. et al. (Eds.). *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 449–460. ISBN 978-3-642-13657-3. DOI: 10.1007/978-3-642-13657-3_48.

GOLDEN, B. et al. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, v. 11, n. 1, p. 49–66, 1984. ISSN 0305-0548. DOI: 10.1016/0305-0548(84)90007-8.

GRAHNE, G.; ZHU, J. Efficiently Using Prefix-trees in Mining Frequent Itemsets. In: GOETHALS, B.; ZAKI, M. J. (Eds.). *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*. 2003. Available from: <http://ceur-ws.org/Vol-90>.

GUERINE, M.; ROSSETI, I.; PLASTINO, A. Extending the Hybridization of Metaheuristics with Data Mining to a Broader Domain. In: INSTICC. PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS - VOLUME 3: ICEIS. SciTePress, 2014. p. 395–406. ISBN 978-989-758-027-7. DOI: 10.5220/0004891303950406.

GUIDOTTI, R. et al. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 5, 2018. ISSN 0360-0300. DOI: 10.1145/3236009.

GUSEV, V. V. The vertex cover game: Application to transport networks. *Omega*, v. 97, p. 102102, 2020. ISSN 0305-0483. DOI: 10.1016/j.omega.2019.08.009.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. Third Edition. Boston: Morgan Kaufmann, 2012. (The Morgan Kaufmann Series in Data Management Systems). ISBN 978-0-12-381479-1. DOI: 10.1016/C2009-0-61819-5.

HO, T. K. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 8, p. 832–844, 1998. DOI: 10.1109/34.709601.

JAPKOWICZ, N.; SHAH, M. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011. DOI: 10.1017/CBO9780511921803.

JIANG, S. et al. FHCSolver: Fast Hybrid CVRP Solver. In: 12TH DIMACS IMPLEMENTATION CHALLENGE: VEHICLE ROUTING PROBLEMS. 2022. Available from: <http://dimacs.rutgers.edu/events/details?eID=2086>.

JOURDAN, L.; DHAENENS, C.; TALBI, E.-G. Using Datamining Techniques to Help Metaheuristics: A Short Survey. In: ALMEIDA, F. et al. (Eds.). *Hybrid Metaheuristics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 57–69. ISBN 978-3-540-46385-6. DOI: 10.1007/11890584_5.

JOVANOVIC, R.; TUBA, M. An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. *Applied Soft Computing*, v. 11, n. 8, p. 5360–5366, 2011. ISSN 1568-4946. DOI: 10.1016/j.asoc.2011.05.023.

KARP, R. M. Reducibility among Combinatorial Problems. In: *Complexity of Computer Computations*. Ed. by R. E. Miller, J. W. Thatcher and J. D. Bohlinger. Boston, MA: Springer US, 1972. p. 85–103. ISBN 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

KENNY, A. et al. An Improved Merge Search Algorithm for the Constrained Pit Problem in Open-Pit Mining. In: PROCEEDINGS OF THE GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE. Prague, Czech Republic: Association for Computing Machinery, 2019. (GECCO '19), p. 294–302. ISBN 9781450361118. DOI: 10.1145/3321707.3321812.

KINDERVATER, G. A. P.; SAVELSBERGH, M. W. P. Vehicle routing: handling edge exchanges. In: LOCAL SEARCH IN COMBINATORIAL OPTIMIZATION. Princeton University Press, 2018. p. 337–360. DOI: 10.1515/9780691187563-013.

KOCHETOV, Y. A.; KHMELEV, A. V. A hybrid algorithm of local search for the heterogeneous fixed fleet vehicle routing problem. *Journal of Applied and Industrial Mathematics*, v. 9, n. 4, p. 503–518, 2015. ISSN 1990-4797. DOI: 10.1134/S1990478915040079.

KUHN, M. et al. The SIDER database of drugs and side effects. *Nucleic Acids Research*, v. 44, n. D1, p. d1075–d1079, 2015. ISSN 0305-1048. DOI: 10.1093/nar/gkv1075.

LI, F.; GOLDEN, B.; WASIL, E. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, v. 34, n. 9, p. 2734–2742, 2007. ISSN 0305-0548. DOI: 10.1016/j.cor.2005.10.015.

MAGALHÃES, J. P. DE; MATSUDA, A. Genome-Wide Patterns of Genetic Distances Reveal Candidate Loci Contributing to Human Population-Specific Traits. *Annals of Human Genetics*, v. 76, n. 2, p. 142–158, 2012. DOI: 10.1111/j.1469-1809.2011.00695.x.

MAIA, M. R. H.; PLASTINO, A.; FREITAS, A. A. An Ensemble of Naive Bayes Classifiers for Uncertain Categorical Data. In: 2021 IEEE INTERNATIONAL CONFERENCE ON DATA MINING (ICDM). 2021. p. 1222–1227. DOI: 10.1109/ICDM51629.2021.00148.

MAIA, M. R. H.; PLASTINO, A.; PENNA, P. H. V. MineReduce: An approach based on data mining for problem size reduction. *Computers & Operations Research*, v. 122, p. 104995, 2020. ISSN 0305-0548. DOI: 10.1016/j.cor.2020.104995.

MAIA, M. R. H.; PLASTINO, A.; SOUZA, U. S. An improved hybrid genetic search with data mining for the CVRP. In: 12TH DIMACS IMPLEMENTATION CHALLENGE: VEHICLE ROUTING PROBLEMS. 2022. Available from: <http://dimacs.rutgers.edu/events/details?eID=2073>.

MAIA, M. R. H.; PLASTINO, A.; SOUZA, U. S. MineReduce for the minimum weight vertex cover problem. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON OPTIMIZATION AND LEARNING (OLA'2020). Cádiz, Spain, 2020. p. 11–22.

MAIA, M. R. H. *Heurísticas Híbridas com Mineração de Dados para o Problema de Roteamento de Veículos com Frota Heterogênea*. 2015. Master's thesis – Universidade Federal Fluminense, Niterói, RJ, Brazil. Available from: <http://www.ic.uff.br/index.php/pos-graduacao/teses-e-dissertacoes>.

MAIA, M. R. H.; PLASTINO, A.; PENNA, P. H. V. Hybrid data mining heuristics for the heterogeneous fleet vehicle routing problem. *RAIRO - Operations Research*, v. 52, n. 3, p. 661–690, 2018. DOI: 10.1051/ro/2017072.

MARTINS, S. L.; ROSSETI, I.; PLASTINO, A. Data Mining in Stochastic Local Search. In: *Handbook of Heuristics*. Ed. by R. Martí, P. M. Pardalos and M. G. C. Resende. Cham: Springer International Publishing, 2018. p. 39–87. ISBN 978-3-319-07124-4. DOI: 10.1007/978-3-319-07124-4_11.

MONTIEL, O.; DELGADILLO, F. J. D. Reducing the Size of Combinatorial Optimization Problems Using the Operator Vaccine by Fuzzy Selector with Adaptive Heuristics. *Mathematical Problems in Engineering*, Hindawi Publishing Corporation, v. 2015, p. 713043, 2015. ISSN 1024-123X. DOI: 10.1155/2015/713043.

MONTIEL, O.; DIAZ-DELGADILLO, F. J.; SEPÚLVEDA, R. Combinatorial complexity problem reduction by the use of artificial vaccines. *Expert Systems with Applications*, v. 40, n. 5, p. 1871–1879, 2013. ISSN 0957-4174. DOI: 10.1016/j.eswa.2012.10.011.

NIEDERMEIER, R.; ROSSMANITH, P. On efficient fixed-parameter algorithms for weighted vertex cover. *Journal of Algorithms*, v. 47, n. 2, p. 63–77, 2003. ISSN 0196-6774. DOI: 10.1016/S0196-6774(03)00005-1.

PEARL, J. Comment: Understanding Simpson's Paradox. *The American Statistician*, Taylor & Francis, v. 68, n. 1, p. 8–13, 2014. DOI: 10.1080/00031305.2014.876829.

PEARL, J. Simpson's Paradox, Confounding, and Collapsibility. In: CAUSALITY. 2. ed.: Cambridge University Press, 2009. p. 173–200. DOI: 10.1017/CBO9780511803161.008.

PENNA, P. H. V.; SUBRAMANIAN, A.; OCHI, L. S. An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem. *Journal of Heuristics*, v. 19, n. 2, p. 201–232, 2013. ISSN 1572-9397. DOI: 10.1007/s10732-011-9186-y.

PENNA, P. H. V.; SUBRAMANIAN, A.; OCHI, L. S., et al. A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research*, v. 273, p. 5–74, 2019. ISSN 1572-9338. DOI: 10.1007/s10479-017-2642-9.

PLASTINO, A. et al. Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic. *Journal of Heuristics*, v. 20, p. 39–74, 2014. ISSN 1572-9397. DOI: 10.1007/s10732-013-9231-0.

QIN, B.; XIA, Y.; LI, F. DTU: A Decision Tree for Uncertain Data. In: THEERAMUNKONG, T. et al. (Eds.). *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 4–15. ISBN 978-3-642-01307-2. DOI: 10.1007/978-3-642-01307-2_4.

QUEIROGA, E.; SADYKOV, R. POP-HGS: Hybrid Genetic Search as a subsolver in a POPMUSIC algorithm. In: 12TH DIMACS IMPLEMENTATION CHALLENGE: VEHICLE ROUTING PROBLEMS. 2022. Available from: <http://dimacs.rutgers.edu/events/details?eID=2072>.

REINELT, G. TSPLIB–A Traveling Salesman Problem Library. *ORSA Journal on Computing*, v. 3, n. 4, p. 376–384, 1991. DOI: 10.1287/ijoc.3.4.376.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. Anchors: High-Precision Model-Agnostic Explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, p. 1527–1535, 2018. Available from: <https://dl.acm.org/doi/abs/10.5555/3504035.3504222>.

RIBEIRO, M. H.; PLASTINO, A.; MARTINS, S. L. Hybridization of GRASP Metaheuristic with Data Mining Techniques. *Journal of Mathematical Modelling and Algorithms*, v. 5, p. 23–41, 2006. ISSN 1572-9214. DOI: 10.1007/s10852-005-9030-1.

SANTANA, Í.; PLASTINO, A.; ROSSETI, I. Improving a state-of-the-art heuristic for the minimum latency problem with data mining. *International Transactions in Operational Research*, v. 29, n. 2, p. 959–986, 2022. DOI: 10.1111/itor.12774.

SANTOS, L. F.; MARTINS, S. L.; PLASTINO, A. Applications of the DM-GRASP heuristic: a survey. *International Transactions in Operational Research*, v. 15, n. 4, p. 387–416, 2008. DOI: 10.1111/j.1475-3995.2008.00644.x.

SHYU, S. J.; YIN, P.-Y.; LIN, B. M. T. An Ant Colony Optimization Algorithm for the Minimum Weight Vertex Cover Problem. *Annals of Operations Research*, v. 131, p. 283–304, 2004. ISSN 1572-9338. DOI: 10.1023/B:ANOR.0000039523.95673.33.

SILVA, M. M. et al. A simple and effective metaheuristic for the Minimum Latency Problem. *European Journal of Operational Research*, v. 221, n. 3, p. 513–520, 2012. ISSN 0377-2217. DOI: 10.1016/j.ejor.2012.03.044.

SILVA, P. N.; PLASTINO, A.; FREITAS, A. A. A Novel Genetic Algorithm for Feature Selection in Hierarchical Feature Spaces. In: PROCEEDINGS OF THE 2018 SIAM INTERNATIONAL CONFERENCE ON DATA MINING (SDM). 2018. p. 738–746. DOI: 10.1137/1.9781611975321.83.

SILVA, P. N. et al. A Novel Feature Selection Method for Uncertain Features: An Application to the Prediction of Pro-/Anti-Longevity Genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 18, n. 6, p. 2230–2238, 2021. DOI: 10.1109/TCBB.2020.2988450.

SIMENSEN, M.; HASLE, G.; STÅLHANE, M. Hybrid Genetic Search With Ruin-and-Recreate. In: 12TH DIMACS IMPLEMENTATION CHALLENGE: VEHICLE ROUTING PROBLEMS. 2022. Available from: <http://dimacs.rutgers.edu/events/details?eID=2071>.

SÖRENSEN, K.; ARNOLD, F.; CUERVO, D. P. A critical analysis of the "improved Clarke and Wright savings algorithm". *International Transactions in Operational Research*, v. 26, n. 1, p. 54–63, 2019. DOI: 10.1111/itor.12443.

SZKLARCZYK, D.; GABLE, A. L., et al. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research*, v. 47, n. D1, p. d607–d613, 2018. ISSN 0305-1048. DOI: 10.1093/nar/gky1131.

SZKLARCZYK, D.; SANTOS, A., et al. STITCH 5: augmenting protein–chemical interaction networks with tissue and affinity data. *Nucleic Acids Research*, v. 44, n. D1, p. d380–d384, 2015. ISSN 0305-1048. DOI: 10.1093/nar/gkv1277.

TACUTU, R. et al. Human Ageing Genomic Resources: new and updated databases. *Nucleic Acids Research*, v. 46, n. D1, p. d1083–d1090, 2017. ISSN 0305-1048. DOI: 10.1093/nar/gkx1042.

TAILLARD, E. D. A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO - Operations Research*, v. 33, n. 1, p. 1–14, 1999. DOI: 10.1051/ro:1999101.

TALBI, E.-G. Machine Learning into Metaheuristics: A Survey and Taxonomy. *ACM Computing Surveys*, Association for Computing Machinery, New York, NY, USA, v. 54, n. 6, 2021. ISSN 0360-0300. DOI: 10.1145/3459664.

THARWAT, A. Classification assessment methods. *Applied Computing and Informatics*, Emerald Publishing Limited, v. 17, n. 1, p. 168–192, 2021. ISSN 2634-1964. DOI: 10.1016/j.aci.2018.08.003.

TSANG, S. et al. Decision Trees for Uncertain Data. *IEEE Transactions on Knowledge and Data Engineering*, v. 23, n. 1, p. 64–78, 2011. DOI: 10.1109/TKDE.2009.175.

UCHOA, E. et al. New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*, v. 257, n. 3, p. 845–858, 2017. ISSN 0377-2217. DOI: 10.1016/j.ejor.2016.08.012.

VIDAL, T. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, v. 140, p. 105643, 2022. ISSN 0305-0548. DOI: 10.1016/j.cor.2021.105643.

VIDAL, T. et al. *A Unifying View on Timing Problems and Algorithms*. 2011. Available from: <https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2011-43.pdf>.

VOIGT, S. et al. Hybrid Large Neighborhood Search for the Capacitated Vehicle Routing Problem and the Vehicle Routing Problem with Time Windows. In: 12TH DIMACS IMPLEMENTATION CHALLENGE: VEHICLE ROUTING PROBLEMS. 2022. Available from: <http://dimacs.rutgers.edu/events/details?eID=2088>.

WALSHAW, C. Multilevel Refinement for Combinatorial Optimisation: Boosting Metaheuristic Performance. In: *Hybrid Metaheuristics: An Emerging Approach to Optimization*. Ed. by C. Blum. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 261–289. ISBN 978-3-540-78295-7. DOI: 10.1007/978-3-540-78295-7_9.

WAN, C.; FREITAS, A. Prediction of the pro-longevity or anti-longevity effect of Caenorhabditis Elegans genes based on Bayesian classification methods. In: 2013 IEEE International Conference on Bioinformatics and Biomedicine. 2013. p. 373–380. DOI: 10.1109/BIBM.2013.6732521.

WATSON, D. S. et al. Local explanations via necessity and sufficiency: unifying theory and practice. In: CAMPOS, C.; MAATHUIS, M. H. (Eds.). *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. PMLR, 2021. v. 161. (Proceedings of Machine Learning Research), p. 1382–1392. Available from: <https://proceedings.mlr.press/v161/watson21a.html>.

WITTEN, I. H.; FRANK, E.; HALL, M. A. *Data Mining: Practical Machine Learning Tools and Techniques*. Third Edition. Boston: Morgan Kaufmann, 2011. (The Morgan Kaufmann Series in Data Management Systems). ISBN 978-0-12-374856-0. DOI: 10.1016/B978-0-12-374856-0.00023-7.

XIE, Z.; XU, Y.; HU, Q. Uncertain data classification with additive kernel support vector machine. *Data & Knowledge Engineering*, v. 117, p. 87–97, 2018. ISSN 0169-023X. DOI: 10.1016/j.datak.2018.07.004.

ZAKI, M. J.; MEIRA JR, W. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9780511810114.

ZHANG, J. et al. Evolutionary Computation Meets Machine Learning: A Survey. *IEEE Computational Intelligence Magazine*, v. 6, n. 4, p. 68–75, 2011. DOI: 10.1109/MCI.2011.942584.

ZHENG, J. et al. MAESN: Solver Description. In: 12TH DIMACS IMPLEMENTATION CHALLENGE: VEHICLE ROUTING PROBLEMS. 2022. Available from: <http://dimacs.rutgers.edu/events/details?eID=2075>.

ZHOU, T. et al. Multi-start iterated tabu search for the minimum weight vertex cover problem. *Journal of Combinatorial Optimization*, v. 32, n. 2, p. 368–384, 2016. ISSN 1573-2886. DOI: 10.1007/s10878-015-9909-3.

APPENDIX A – MAIA, M. R. H.; PLASTINO, A.; PENNA, P. H. V. "MineReduce: An approach based on data mining for problem size reduction". Computers & Operations Research, v. 122, 104995, 2020[1]

[1] Available from: https://doi.org/10.1016/j.cor.2020.104995

Computers and Operations Research 122 (2020) 104995

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: http://ees.elsevier.com

MineReduce: An approach based on data mining for problem size reduction

Marcelo Rodrigues de Holanda Maia [a,*], Alexandre Plastino [a], Puca Huachi Vaz Penna [b]

[a] Instituto de Computação, Universidade Federal Fluminense, Avenida General Milton Tavares de Souza s/n, Niterói, RJ, 24210-346, Brazil
[b] Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto, Campus Universitário Morro do Cruzeiro, Ouro Preto, MG, 35400-000, Brazil

ARTICLE INFO

Article history:
Received 9 April 2019
Revised 6 April 2020
Accepted 5 May 2020
Available online 21 May 2020

Keywords:
Problem size reduction
Hybrid metaheuristics
Data mining
Combinatorial optimization
Vehicle routing

ABSTRACT

Hybrid variations of metaheuristics that include data mining strategies have been utilized to solve a variety of combinatorial optimization problems, with superior and encouraging results. Previous hybrid strategies applied mined patterns to guide the construction of initial solutions, leading to more effective exploration of the solution space. Solving a combinatorial optimization problem is usually a hard task because its solution space grows exponentially with its size. Therefore, problem size reduction is also a useful strategy in this context, especially in the case of large-scale problems. In this paper, we build upon these ideas by presenting an approach named MineReduce, which uses mined patterns to perform problem size reduction. We present an application of MineReduce to improve a heuristic for the heterogeneous fleet vehicle routing problem. The results obtained in computational experiments show that this proposed heuristic demonstrates superior performance compared to the original heuristic and other state-of-the-art heuristics, achieving better solution costs with shorter run times.

1. Introduction

Hybrid metaheuristics have been proposed and successfully applied to combinatorial optimization problems (COPs) in several areas, allowing near-optimal solutions to be found within an acceptable computational time frame. One successful example is the hybridization of the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic (Resende and Ribeiro, 2016) with data mining techniques (Martins et al., 2018).

Previous work that explored data science in combinatorial optimization produced highly significant results for several problems (Ribeiro et al., 2006; Santos et al., 2008; Plastino et al., 2011; Plastino et al., 2014; Barbalho et al., 2013; Guerine et al., 2016; Maia et al., 2018; Martins et al., 2018). Data-mining-hybridized heuristics were able to find solutions of higher quality while spending less computational time when compared to their non–hybridized counterparts and other state-of-the-art heuristics. They applied patterns (found by data mining procedures) to guide the construction of initial solutions.

The difficulty in solving a COP is due to the fact that its solution space grows exponentially with its size. Therefore, problem size reduction (PSR) – which consists of reducing the size of a problem instance, then solving the reduced instance and expanding it back – has been found to be a very useful strategy in this context, especially in the case of large-scale problems, since it can significantly reduce the search space of a COP (Gavish and Srikanth, 1986; Fischer and Merz, 2007; Delgadillo et al., 2016).

Building upon these ideas, in this paper, we present an approach named MineReduce, which uses mined patterns to perform problem size reduction. Previous methods that also incorporate data mining techniques into metaheuristics, in general, use the mined patterns as a starting point for the construction of initial solutions (Martins et al., 2018). MineReduce, on the other hand, performs a PSR procedure by removing or merging elements that are in a pattern, finding a solution to the reduced instance, and expanding the solution found, which will be used as the starting point for a local search on the original solution space.

In order to validate the MineReduce approach, we have applied it to extend a previous and state-of-the-art heuristic for the heterogeneous fleet vehicle routing problem (HFVRP), obtaining significantly better results in terms of both solution quality and computational time.

So, the main contribution of this work is twofold: an approach based on the idea of using mined patterns to perform PSR, and a state-of-the-art heuristic for solving the HFVRP based on this approach. The outcomes of computational experiments carried out on a large quantity of HFVRP instances from extensively used

* Corresponding author.
E-mail addresses: mmaia@ic.uff.br (M. Rodrigues de Holanda Maia), plastino@ic.uff.br (A. Plastino), puca@ufop.edu.br (P.H.V. Penna).

https://doi.org/10.1016/j.cor.2020.104995
0305-0548/

collections affirm the effectiveness of the proposed heuristic. The results show that the proposed heuristic based on MineReduce demonstrates superior performance when compared with a state-of-the-art heuristic proposed by Penna et al. (2013a), which served as a baseline for the development of our proposal, and also with a recent hybrid data mining approach proposed for the problem (Maia et al., 2018), achieving better solution costs with shorter run times. Furthermore, new best solutions to 22 instances were found.

Additionally, we have compared the results obtained by our proposed heuristic to those reported by Kochetov and Khmelev (2015) and Penna et al. (2019) for their state-of-the-art algorithms. MineReduce proved to be very competitive in this comparison, especially for large instances.

The remainder of this article is organized as follows. Section 2 presents a brief review of related work. Section 3 introduces the MineReduce approach. Section 4 describes the application of the MineReduce approach to the HFVRP. In Section 5, the outcomes from experiments using the MineReduce-based heuristic are analyzed and compared with those obtained using previous heuristics. Finally, Section 6 provides conclusions and directions for future work.

2. Related work

This section presents a brief review of related work regarding hybrid data mining heuristics (Section 2.1) and PSR in the context of combinatorial optimization (Section 2.2).

2.1. Hybrid data mining heuristics

One successful example of hybrid data mining heuristic is the proposal of a hybrid version of the GRASP metaheuristic which incorporates a data mining module (Ribeiro et al., 2006), called Data Mining GRASP (DM-GRASP). The basic idea of this hybrid metaheuristic is that patterns found in good solutions can be used to guide the search, leading to more effective exploration of the solution space. In this hybrid version, after the execution of half of the GRASP iterations, a data mining procedure is applied to extract patterns from an elite set composed of the best solutions found up to that point. These patterns represent features found in the elite set of solutions and are used to guide the search in the second half of the iterations. It has been successfully applied to the set packing (Ribeiro et al., 2006), maximum diversity (Santos et al., 2005), server replication for reliable multicasting (Santos et al., 2006), p-median (Plastino et al., 2011; Martins et al., 2018), 2-path network design (Barbalho et al., 2013) and one-commodity pickup-and-delivery traveling salesman (Guerine et al., 2016) problems. A survey of this approach has been presented by Santos et al. (2008).

Subsequently, a new version of the DM-GRASP metaheuristic, called Multi-Data Mining GRASP (MDM-GRASP), was proposed (Plastino et al., 2014). The main idea underlying this version is to run the data mining procedure multiple times in an adaptive mode. MDM-GRASP has also been applied to the p-median (Plastino et al., 2011), 2-path network design (Barbalho et al., 2013), server replication for reliable multicasting (Plastino et al., 2014) and one-commodity pickup-and-delivery traveling salesman (Guerine et al., 2016) problems, and it has achieved better results than those obtained by DM-GRASP, not only in terms of solution quality but also concerning computational time.

Recently, the data mining techniques used in DM-GRASP and MDM-GRASP have been incorporated into a multi-start ILS (MS-ILS) heuristic for the HFVRP, resulting in the state-of-the-art hybrid heuristics DM-MS-ILS and MDM-MS-ILS (Maia et al., 2018).

A survey of heuristics that incorporate data mining procedures was presented by Martins et al. (2018).

2.2. Problem size reduction

Any approach for solving a COP relies on some form of search on its solution space, which is the domain of the function to be optimized. Solving a COP is usually a hard task because its solution space grows exponentially with its size. Therefore, PSR is beneficial in this context, especially in the case of large-scale COPs (Gavish and Pirkul, 1985; Gavish and Pirkul, 1985; Gavish and Srikanth, 1986).

The size of a COP is defined by its decision variables, i.e., the dimensions of its solution space. Hence, PSR techniques have the aim of reducing the number of decision variables of the problem.

The general process of PSR techniques consists of: (i) transforming a problem P into a modified problem P' such that the number of decision variables of P' is smaller than the number of decision variables of P, (ii) solving P', and (iii) transforming the solution to P' into a solution to P.

The application of this procedural framework is exemplified by the Reduce-Optimize-Expand (ROE) method (Montiel et al., 2013). In the first step (reduce), the ROE method reduces the size of problem instances just before the application of a solving algorithm. Once the reduction is achieved, the next step (optimize) is the application of a combinatorial optimization method, exact or heuristic, to obtain an optimal or suboptimal solution. Then it executes the last step (expand) to obtain the final result.

Size reduction is naturally the most crucial procedure in the PSR process since better decisions in this step will produce solutions of better quality in the subsequent stages (Montiel et al., 2013; Montiel and Delgadillo, 2015; Delgadillo et al., 2016). Strategies to make these decisions are highly dependent on the structure and features of the problem, so they vary significantly among distinct classes of COPs.

One general form of accomplishing size reduction is fixing values of decision variables, which can be regarded as fixing elements either in or out of the solution. This kind of reduction is common for many classes of COPs, like routing problems, which include the HFVRP. For example, in binary formulations of classical routing problems – such as the traveling salesman problem (TSP) or the vehicle routing problem (VRP) – the decision variables refer to edges in the problem instance graph. The value set to a variable indicates whether the corresponding edge is in (one) or out of (zero) the solution.

Analyzing routing problems as graph-based models, which is typical for this class of problems, approaches to reduce the size of an instance can be either vertex-based or edge-based, as stated by Fischer and Merz (2007) for the TSP. In a vertex-based approach, subsets of vertices are merged into one single vertex or cluster-vertex. In an edge-based approach, a sequence of edges is merged into one single edge or fixed-path-edge. In both cases, what is implicitly done is also fixing the values of decision variables.

Studies found in the literature in which PSR has been successfully applied to routing problems include those by Gavish and Srikanth (1986); Min (1991), Barnhart et al. (2002); Fischer and Merz (2007), Ramos and Oliveira (2011); Lin (2011), Montiel et al. (2013); Montiel and Delgadillo (2015) and Delgadillo et al. (2016).

Work on the multilevel paradigm, which involves the recursive application of the PSR steps, has provided evidence that it can aid metaheuristics to find better solutions faster for some COPs (Walshaw, 2008). In a multilevel optimization method, the original problem instance is recursively coarsened (reduced), creating a multilevel hierarchy of reductions. An initial solution is found (at

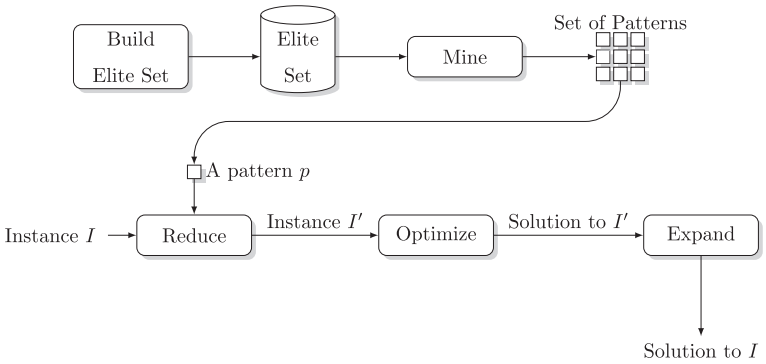


Fig. 1. Framework of the MineReduce approach.

the coarsest level) and then, at each level in reverse order, iteratively expanded to a solution to the parent level's instance and refined, usually with a local search algorithm.

Chen (2015) proposed a fix-and-optimize approach for mixed integer programming problems. It decomposes the original problem by fixing values of binary variables based on their interrelatedness. That approach has also been integrated into a variable neighborhood search framework, achieving excellent results for lot sizing (Chen, 2015; Li et al., 2017) and transportation (Toschi et al., 2018) problems.

Blum et al. (2016) introduced a hybrid metaheuristic called Construct, Merge, Solve & Adapt (CMSA). It is based on the idea of generating solutions, reducing the original instance by merging the components in the generated solutions (in a way such that a solution to the original instance can be derived from a solution to a reduced instance), solving the reduced instances to optimality, and adapting these reduced instances based on an ageing mechanism. The CMSA metaheuristic has achieved good results for some COPs, like the minimum common string partition (Blum et al., 2016; Blum and Raidl, 2016), the minimum covering arborescence (Blum et al., 2016), and the repetition-free longest common subsequence (Blum et al., 2016).

Kenny et al. (2019) presented another hybrid metaheuristic, based on a strategy known as "merge search", which has some similarities with CMSA. The main difference between them is that, while CMSA generates solutions from scratch, the merge search algorithm starts with a single initial seed solution and uses local search to generate a population of neighbouring solutions to the initial seed solution. Then it goes through an iterative process of generating a population, merging, and solving the reduced instance using an exact method. The merging procedure is based on the intersections between all of the solutions in the population. The solution to a reduced instance becomes the initial seed solution to the next generation. As each iteration produces a new set of solutions, there is no need for an ageing mechanism to regulate the population. This metaheuristic has achieved excellent results for the constrained pit problem.

3. The MineReduce approach

Several multi-start heuristics, such as the GRASP-based ones, perform a sequence of independent iterations, composed of a generation phase and a local search phase. The generation phase builds an initial solution and, in general, consists of the application of more straightforward methods, usually based on a combination of greedy and random strategies. Most of the computational effort is employed in the local search phase, which improves the initial solution.

Previous approaches that incorporate data mining into these heuristics eliminate the independence of their iterations by introducing a memory mechanism into them. At some point, iterations begin to benefit from knowledge accumulated in previous iterations. This knowledge – patterns mined from an elite set of solutions – is used to generate better initial solutions.

The data mining procedure in these approaches relies on the FPmax* algorithm (Grahne and Zhu, 2003), which mines maximal frequent itemsets. An itemset is considered to be frequent if it achieves a given minimum support, i.e., if it is present in at least a given minimum number of the elite set solutions. Hence, mined patterns are composed of items that frequently appear together in the sub-optimal solutions of the elite set. Intuitively, it is assumed these items should likely be part of the best solutions to the problem. Thus, they are included in initial solutions.

With the MineReduce approach, we build upon these ideas. Since we assume the mined patterns should likely be part of the best solutions to a problem instance, then they could be well-suited for reducing the size of that instance. The items in a pattern could, for example, be fixed in the solution, which in turn would be equivalent to fixing values of decision variables related to those items. Another possibility is that the items in a pattern could be merged in a condensed representation, also producing a reduced-size instance.

Fig. 1 presents the general framework of the MineReduce approach. Its first steps are to *build an elite set* of solutions and to *mine* patterns from this set. These steps are supposed to be done like in the previous approaches (Martins et al., 2018), i.e., the best solutions found are stored in the elite set until it becomes stable (unaltered for a given period), which triggers the data mining procedure.

The subsequent steps compose a full PSR process, which is intended to replace the heuristic's initial solution generation procedure. The *Reduce* step uses a pattern p to transform a problem instance I into a reduced-size instance I'. The *Optimize* step is accomplished through the application of the heuristic's original optimization procedures to I'. In the *Expand* step, the solution to I' is transformed into a solution to I, which concludes the MineReduce-based generation of an initial solution.

This modified constructive method is what makes MineReduce different from the previous approaches that also incorporate data mining techniques into metaheuristics. Previous methods derived

from the DM-GRASP and MDM-GRASP metaheuristics – e.g., the hybrid heuristics for the HFVRP proposed in (Maia et al., 2018) – use the mined patterns only as a starting point for the construction of initial solutions. MineReduce, on the other hand, performs a procedure that reduces the original instance by deleting or merging elements that are in a pattern, finds a solution to the reduced instance and expands the solution found, which will be used as the starting point for a local search on the original solution space.

In this sense, the MineReduce approach shares some aspects with the multilevel paradigm (Walshaw, 2008). Each iteration after the first call to the data mining procedure reduces the problem instance to a "coarser" level, finds a solution to the reduced instance, expands the solution (back to the original instance's level) and refines it with a local search. In this case, the number of levels is constant (two), but the main difference resides in the reduction strategy. In the multilevel paradigm, this is often done through adaptation of construction heuristics, whereas the MineReduce approach relies on the use of patterns extracted from an elite set of solutions through data mining techniques.

Regarding the reduction strategy, the MineReduce approach is closer to merge search algorithms (Kenny et al., 2019), which reduce instances by merging variables based on the intersections observed in a set of solutions. MineReduce's reduction strategy also relies on similarities observed in a set of solutions. However, they differ significantly in how they build such a set of solutions and in how similarities between its solutions are characterized and identified.

4. Application of the MineReduce approach to the HFVRP

This section presents the application of the MineReduce approach to extend the MS-ILS heuristic for the HFVRP proposed by Penna et al. (2013a).

4.1. The HFVRP

The HFVRP is described as follows. Let $G = (V, A)$ be a directed graph, where $V = \{0, 1, \ldots, n\}$ is a set composed of $n + 1$ vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs. Vertex 0 is the depot, where the vehicle fleet is located, whereas the set $V' = V \setminus \{0\}$ consists of the remaining vertices representing the n customers. Each customer $i \in V'$ is associated with a non-negative demand q_i. The fleet consists of m distinct vehicle types, which compose a set $M = \{1, \ldots, m\}$. For each vehicle type $u \in M$, there are m_u vehicles available, each with a capacity Q_u and a fixed cost f_u. Finally, for each combination of an arc $(i, j) \in A$ and a vehicle type $u \in M$, there is an associated cost $c_{ij}^u = d_{ij} r_u$, where d_{ij} is the distance between vertices i and j and r_u is the dependent (variable) cost per unit distance associated with the vehicle type u.

A route is defined by a pair (R, u), where $R = (i_1, i_2, \ldots, i_{|R|})$, $i_1 = i_{|R|} = 0$, and $\{i_2, \ldots, i_{|R|-1}\} \subseteq V'$; that is, each route is a circuit in G that starts and ends at the depot and is assigned to a vehicle of type $u \in M$. A route (R, u) is feasible if the sum of the demands of all customers on R does not exceed the capacity Q_u of the vehicle assigned to it. The cost of a route is the sum of the fixed cost of the assigned vehicle and the dependent costs associated with each of the traversed arcs in combination with the vehicle type. Thus, the aim is to discover feasible routes such that each customer is visited precisely once, the total quantity of routes assigned to vehicles of each type $u \in M$ does not exceed m_u, and the sum of all route costs is minimized.

HFVRP instances are typically categorized with respect to certain criteria. Two primary classes are related to the limitations on the fleet. The problem that characterizes the first class, known as the Fleet Size and Mix (FSM) problem (Golden et al., 1984), can be considered as a particular case of the above definition in which $m_u = \infty, \forall u \in M$. Therefore, this problem consists of identifying the best composition of the fleet as well as its best routing scheme. For the second category of instances, in which the fleet is limited, the corresponding problem is known as the Heterogeneous Fixed Fleet VRP (HFFVRP) (Taillard, 1999) and consists of optimizing the routing for an available fixed fleet.

The FSM and HFFVRP classes may be further subdivided with respect to the types of vehicle costs considered. The possibilities are as follows: both fixed and dependent costs (the general case), fixed costs only (a particular case in which $r_u = 1, \forall u \in M$), or dependent costs only (a particular case in which $f_u = 0, \forall u \in M$). Five subclasses of FSM and HFFVRP instances with respect to this criterion are differentiated in the literature: (1) the FSM problem with both fixed and dependent costs, denoted by FSM-FD (Ferland and Michelon, 1988); (2) the FSM problem with fixed costs only, denoted by FSM-F (Golden et al., 1984); (3) the FSM problem with dependent costs only, denoted by FSM-D (Taillard, 1999); (4) the HFFVRP with both fixed and dependent costs, denoted by HFFVRP-FD (Li et al., 2007); and (5) the HFFVRP with dependent costs only, denoted by HFFVRP-D (Taillard, 1999). To the best of our knowledge, the HFFVRP with fixed costs only has never been addressed.

An overview of the strategies in the literature for solving the HFVRP can be found in a survey by Baldacci et al. (2008). A literature survey of this problem that focuses on industrial aspects of the FSM problem has been presented by Hoff et al. (2010). A more recent literature survey on the HFVRP has been presented by Koç et al. (2016). The latter provides a computational comparison among state-of-the-art algorithms. It indicates that the best performances for the FSM problem had been achieved by the algorithms proposed by Choi and Tcha (2007); Penna et al. (2013a) and Vidal et al. (2014), whereas the best performances for the HFFVRP had been achieved by the algorithms proposed by Li et al. (2007); Liu (2013), Penna et al. (2013a) and Subramanian et al. (2012). After publication of the survey by Koç et al. (2016), other heuristic strategies presented competitive results (Maia et al., 2018; Penna et al., 2019) and an exact method found new optimal solutions for some instances (Pessoa et al., 2018).

Multi-start heuristics based on the Iterated Local Search (ILS) metaheuristic, in particular, have presented competitive results when applied to vehicle routing problems (Subramanian et al., 2012; Penna et al., 2013a; Penna et al., 2013b; Penna et al., 2019). Recently, hybrid heuristics for the HFVRP based on the incorporation of data mining strategies into the state-of-the-art MS-ILS heuristic of Penna et al. (2013a) were proposed and demonstrated superior performance compared with the original heuristic, achieving better solution costs with shorter run times (Maia et al., 2018). Therefore, the MS-ILS heuristic, described in the next section, was chosen as the basis for the application of the MineReduce approach. We compared the heuristic proposed in this work to the MS-ILS heuristic of Penna et al. (2013a) and to its hybrid version called MDM-MS-ILS, presented in (Maia et al., 2018).

4.2. MS-ILS heuristic

In this section, we describe – in a high level of abstraction (we suppress details that are irrelevant for the context of this work) – the MS-ILS (Multi-Start Iterated Local Search) heuristic proposed by Penna et al. (2013a) for solving the HFVRP, which served as a basis for the MineReduce heuristic proposed in this work. The steps of the MS-ILS are presented in Algorithm 1.

Algorithm 1 MS-ILS(*MaxIter*)

```
1: f(s*) ← ∞
2:for i ← 1 to MaxIter do
3:    s ← GenerateInitialSolution()
4:    s' ← ILS(s)
5:    if f(s') < f(s*) then
6:       s* ← s'
7:    end if
8: end for
9: return s*
```

The multi-start heuristic is run for *MaxIter* iterations (lines 2–8). In each iteration, a constructive procedure (GenerateInitialSolution()) applies a mix of simple greedy and randomized strategies for producing an initial solution (line 3). In the local search phase, the ILS heuristic is used to enhance the generated initial solution (line 4). If the solution s' returned by the ILS heuristic represents an improvement in cost, as given by the function f, then the best overall solution s^* is updated (lines 5–7). After the execution of the *MaxIter* multi-start iterations, the best solution found is returned (line 9).

4.3. MineReduce-MS-ILS

The high-level structure of a general MineReduce-based multi-start metaheuristic (MineReduce-MS) is presented in Algorithm 2. It uses the same strategies from the MDM-GRASP metaheuristic (Plastino et al., 2014) for building the elite set and triggering the data mining procedure. Its more distinctive aspect is the use of a MineReduce-based initial solution generation procedure (line 12).

Algorithm 2 MineReduce-MS(*MaxIter*, d, *MaxP*, *MinSup*, δ)

```
1: f(s*) ← ∞
2:E ← Ø
3: P ← Ø
4: for i ← 1 to MaxIter do
5:    if Stable(E, δ) then
6:       P ← Mine(E, MaxP, MinSup)
7:    end if
8:    if P = Ø then
9:       s ← GenerateInitialSolution()
10:   else
11:      p ← NextPattern(P)
12:      s ← MineReduceGeneration(p)    #MineReduce-based
   generation phase
13:   end if
14:   s' ← LocalSearch(s)
15:   UpdateEliteSet(E, s', d)
16:   if f(s') < f(s*) then
17:      s* ← s'
18:   end if
19: end for
20: return s*
```

Like in several typical multi-start metaheuristics, each iteration in this algorithm is composed of a generation phase (lines 8–13) and a local search phase (line 14). However, some differences can be observed. Whenever the elite set E becomes stable the data mining procedure is triggered – if the data mining procedure has not been called yet, E is stable if it has not been modified in the last δ iterations; otherwise, E is stable if it has been modified after the last call to the data mining procedure but has not been modified in the last δ iterations. The data mining procedure returns a set P composed of the *MaxP* largest patterns found with the minimum support *MinSup* (lines 5–7), which are arranged in decreasing order by size, forming a circular list. In the early iterations, when data mining has not yet been carried out and the pattern set P is therefore still empty, the initial solutions are generated through a traditional constructive procedure, based on a combination of greedy and random strategies (line 9). Once data mining has been performed, each initial solution is then generated through the MineReduce-based procedure (MineReduceGeneration(p)) using a pattern p picked from the current pattern set P following the sequence of the circular list (lines 11–12). After the local search phase (line 14), the elite set is updated (line 15), as is the best solution in case of improvement (lines 16–18).

The pseudo-code of the MineReduce-based constructive method, which implements the PSR steps defined in Fig. 1, is presented in Algorithm 3. It is worth observing that the processed instance I is regarded as an implicit input.

Algorithm 3 MineReduceGeneration(p)

```
1: μ ← ReduceInstance(p)      #Reduce step – I is reduced into
   I'
2: s' ← GenerateInitialSolution()      #Optimize step (lines 2–
   3) – over I'
3: s' ← LocalSearch(s')
4: s ← ExpandSolution(s', μ)      #Expand step – generates a
   solution to I
5: return s
```

In the *Reduce* step, the instance is reduced based on the provided pattern p, returning a map μ that associates elements of the reduced instance to their corresponding elements in the original instance (line 1). The *Optimize* step consists of the application of one complete iteration of the multi-start metaheuristic (generation and local search) to the reduced instance. An initial solution to the reduced instance is generated (line 2), and then a local search starting from it is performed (line 3). Finally, in the *Expand* step, the solution found in the local search is expanded based on the map μ (line 4) and returned (line 5).

The proposed MineReduce-MS-ILS heuristic is the application of the MineReduce approach on the MS-ILS proposed by Penna et al. (2013a). It is, therefore, an implementation of Algorithm 2 where the local search (line 14) is performed by the ILS procedure from Algorithm 1.

In this proposal, solutions in the elite set are represented as sets of arcs instead of sequences of vertices. This representation allows the application of the data mining procedure to extract maximal frequent itemsets. As described in Section 4.1, each route in the HFVRP is represented by a pair (R, u), where $R = (i_1, i_2, \ldots, i_{|R|})$ is a list of vertices, ordered according to the defined visiting sequence, and u is the type of vehicle assigned to the route. In the adopted alternative representation, for each route (R, u), the list R is decomposed into a set of arcs $\{(i_1, i_2), (i_2, i_3), \ldots, (i_{|R|-1}, i_{|R|})\}$. Then, the vehicle type u is assigned to each arc in the set, resulting in a set in which each element is a pair composed of an arc $(i_r, i_{r+1}), r = 1, 2, \ldots, |R| - 1$, in R and the vehicle type u, with the form $\{((i_1, i_2), u), ((i_2, i_3), u), \ldots, ((i_{|R|-1}, i_{|R|}), u)\}$. An arc must be present and associated with the same vehicle type in a minimum number of solutions in the elite set to belong to a pattern. Consequently, the patterns mined are formed of route segments, each one with a vehicle type assigned.

This pattern mining scheme for the HFVRP has been adopted in hybrid versions of the MS-ILS heuristic proposed in (Maia et al., 2018). However, these previous MS-ILS extensions used mined pat-

terns only as a starting point for the construction of initial solutions. Specifically, the constructive procedure used the route segments of a mined pattern, with their respective vehicle types assigned, as the initial routes of the solution, and afterwards completed the solution by inserting the remaining customers.

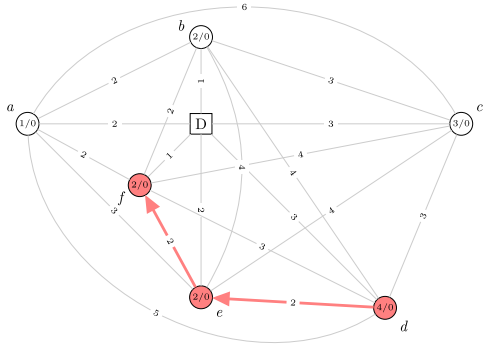
In MineReduce-MS-ILS, the *Reduce* step of MineReduce relies on the use of patterns mined from the elite set to perform a vertex-based PSR procedure by merging customer vertices that appear consecutively (in the same route segment) in a pattern into one customer cluster vertex.

In a binary formulation of the HFVRP, decision variables are defined as: $x_{ij}^k = 1$ if a vehicle of type k travels from customer i to customer j; and $x_{ij}^k = 0$ otherwise. Therefore, differently from the classic VRP, in this case, a vertex-based PSR procedure is not equivalent to fixing values of decision variables. Instead, the original set of decision variables is replaced by a smaller one (since the set of customers is reduced), and the values of the variables in the original set are defined by the values of the corresponding variables in the reduced set.

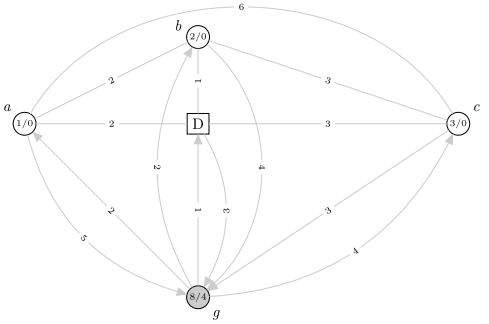
For using this strategy, it is necessary to extend the HFVRP model presented in Section 4.1. Since it must be possible to represent a route segment as a customer vertex, each customer $i \in V$ must have an associated l_i value corresponding to the length of the underlying route segment. This value is used to calculate the cost $c_i^u = l_i r_u$, which represents the variable cost for a vehicle of type u to traverse the route segment represented by i, for each combination of a customer $i \in V$ and a vehicle type $u \in M$. Customer vertices of regular (non-reduced) instances, as well as customer vertices of reduced instances that are not customer cluster vertices, have length 0. The cost of a route becomes, in this extended model, the sum of the fixed cost of the vehicle associated with the route and the variable costs associated (i) with the combination between the vehicle type and each of the traversed arcs, and (ii) with the combination of the type of vehicle and each of the visited customer vertices.

Let $G = (V, A)$ be a directed graph associated with an HFVRP instance and p a pattern consisting of a set of route segments in that instance, each segment defined by a pair (R', u), where $R' = (i_1, i_2, \ldots, i_{|R'|})$. Let $G^* = (V^*, A^*)$ be a directed graph associated with a reduced version of the instance associated with G based on p. Such a reduced version can be obtained as follows. Initially, G^* is defined as a copy of G. For each route segment $(R', u) \in p$, each of the customers in R' is removed from G^* – that is, the vertex corresponding to the customer is removed from V^* and the arcs that connect that vertex to the others are removed from A^*. Also, a customer cluster vertex $i_{R'}$ corresponding to the route segment is added to V^* and arcs connecting $i_{R'}$ to the other vertices in V^* are added to A^*. The demand for $i_{R'}$ is given by $q_{i_{R'}} = \sum_{r'=1}^{|R'|} q_{i_{r'}}$, that is, the sum of customer demands in R'. The distance from each vertex $i^* \in V^*$ to $i_{R'}$ is given by $d_{i^* i_{R'}} = d_{i^* i_1}$, that is, the distance from i^* to i_1 (the first customer in R'). The distance from $i_{R'}$ to each vertex $i^* \in V^*$ is given by $d_{i_{R'} i^*} = d_{i_{|R'|} i^*}$, that is, the distance from $i_{|R'|}$ (the last customer in R') to i^*. Finally, the length of $i_{R'}$ is given by $l_{i_{R'}} = \sum_{r'=1}^{|R'|-1} d_{i_{r'} i_{r'+1}}$, that is, the sum of the distances between consecutive customers in R'.

From the definition of a reduced instance described above, it is noted that, even if all distances d_{ij} associated with the arcs $(i, j) \in A$ are symmetric for an instance of the HFVRP (i.e., $d_{ij} = d_{ji}, \forall i, j \in V$), a reduced version will present asymmetric distances associated with the arcs in A^* that connect customer cluster vertices to the other vertices in V^*. This detail is relevant because many of the approaches proposed for the HFVRP handle only instances with symmetric distances or treat symmetric and asymmetric instances differently. Fig. 2 shows the reduction of an HFVRP instance based on a mined pattern.



(a) Non-reduced instance (with a mined pattern highlighted)



(b) Reduced instance

Fig. 2. An example of reduction of an HFVRP instance based on a mined pattern.

Fig. 2(a) presents a representation of the graph $G = (V, A)$ in the extended model described above. In this representation, the circular vertices (customers) present, in addition to the value of the demand, the value of the length (to the right) of the corresponding customer. In this non-reduced instance, all customer vertices have length 0. The pattern used in the presented reduction is highlighted in the graph of Fig. 2(a). This pattern consists of a single route segment: (R'_1, u_1), where $R'_1 = (d, e, f)$. The graph $G^* = (V^*, A^*)$ associated with the reduced version of the instance is presented in Fig. 2(b). The customer vertices in R'_1 are replaced in the graph by a customer cluster represented by the vertex g. This customer cluster's demand q_g is given by the sum of the demands of the customers it replaces: $q_g = q_d + q_e + q_f = 4 + 2 + 2 = 8$. The distance from each of the other vertices to g is given by the distances from these vertices to d (first customer in R'_1): $d_{Dg} = d_{Dd} = 3, d_{ag} = d_{ad} = 5, d_{bg} = d_{bd} = 4$ and $d_{cg} = d_{cd} = 3$. The distance from g to each of the other vertices is given by the distances from f (last customer in R'_1) to these vertices: $d_{gD} = d_{fD} = 1, d_{ga} = d_{fa} = 2, d_{gb} = d_{fb} = 2$ and $d_{gc} = d_{fc} = 4$. Finally, the length l_g of g is given by the sum of the distances between consecutive customers in R'_1 : $l_g = d_{de} + d_{ef} = 2 + 2 = 4$.

Table 1
Parameters tuning for MDM-MS-ILS and MineReduce with the irace package.

Parameter	Description	Tested values	MDM-MS-ILS	MineReduce
d	Maximum size of the elite set	{10, 15, 20}	10	10
MaxP	Maximum size of the patterns set	{6..10}	9	6
MinSup	Minimum support	{0.2, 0.3, . . . , 1.0}	0.7	0.2
δ	Number of iterations without modification in the elite set for it to be considered stable	{3..7}	3	3

5. Computational results

This section reports the results of the experiments carried out. The MineReduce-based heuristic, called MineReduce-MS-ILS (MineReduce for short), is evaluated and compared to the MS-ILS heuristic proposed by Penna et al. (2013a) and to its previous hybrid data mining version, MDM-MS-ILS (Maia et al., 2018).

In these experiments, we used the 96 HFFVRP-FD instances described by Duhamel et al. (2010), which are divided into four sets: Set 1, with 15 instances, each with fewer than 100 customers; Set 2, with 38 instances, each with 100 to 150 customers; Set 3, with 31 instances, each with 151 to 200 customers; and Set 4, with 12 instances, each with more than 200 customers.

The MineReduce-MS-ILS heuristic was implemented based on the original source code for the MS-ILS heuristic. All of the three heuristics were compiled using the GCC C++ compiler (g++) version 4.8.2. The experiments were run on an Intel® Core™ i7-5500U 2.40 GHz CPU with 8 GB of RAM running Windows 10 (x64). Each heuristic was executed 20 times, using distinct random number seeds, for each instance. In each case, we present the following values obtained over the 20 runs: best solution cost, average solution cost and average computational time (in seconds).

Additionally, we have compared the results obtained by MineReduce to those obtained by the state-of-the-art algorithms of Kochetov and Khmelev (2015) and Penna et al. (2019). In this comparison, we have considered their reported results since we did not have the chance to run experiments with their algorithms.

We report the parameters setting in Section 5.1. Section 5.2 presents the main results, with comprehensive comparisons of the performance of the heuristics tested, as well as comparisons between the results obtained by MineReduce and the results reported in the literature for other state-of-the-art algorithms. Lastly, we present the results of additional experiments carried out to further inspect the behavior of the tested heuristics in Section 5.3.

5.1. Parameters setting

The MS-ILS heuristic has only two parameters, both related to stopping criteria: *MaxIter*, which is the number of multi-start iterations, and β, which is used in the calculation of the maximum number of consecutive perturbations allowed without improvements in the ILS ($MaxIterILS = n + \beta v$). It has been previously observed that the quality of the solutions and the computational time tend to increase as the values of *MaxIter* and β are increased (Penna et al., 2013a), which was expected since they define the number of trials given to the algorithm. Therefore, the question of choosing values for these parameters is a trade-off between solution quality and computational time. In these experiments, we have adopted the following values for these parameters (for all of the heuristics): $MaxIter = 100$, which was adopted by Maia et al. (2018), and $\beta = 5$, which was recommended by Penna et al. (2013a).

For tuning the other parameters of the hybrid heuristics (MDM-MS-ILS and MineReduce), related to the data mining procedure, we have used the irace package, which is a software package that implements iterated racing procedures that have been successfully used to configure various state-of-the-art algorithms (López-Ibáñez et al., 2016). Specifically, we have used irace to derive a set of appropriate parameter values for each heuristic. The list of training instances (which have been taken out of the comparisons in Section 5.2) was composed of the first instance of each set – i.e., instances 01, 03, 02 and 18 from Sets 1, 2, 3 and 4 of Duhamel et al. (2010), respectively. We have defined as the stopping criterion for irace a maximum total execution time (irace's maxTime parameter) of 45 h. Besides the stopping criterion, we have used irace's default configuration.

Table 1 presents the results of the tuning process. The first three columns identify the parameters, their descriptions, and the sets of candidate values provided to irace, respectively. The remaining columns present the best configurations found by irace for each heuristic, which we have used in the experiments reported in this work.

5.2. Main results

This section presents the main results obtained in our computational experiments. For each set of instances, one table presents the results as follows. There is one row for each instance (I), containing the best known solution (BKS) cost reported in the literature – with the indication of the references reporting it (letter marks) – and the results achieved by the compared heuristics. Two extra rows are included at the bottom. The first indicates the number of times each heuristic attained the best result (# of wins), whereas the second indicates the average percentage difference (APD) obtained by the hybrid heuristics – i.e., the mean of the percentage differences in average cost or time for each of the hybrid heuristics compared to the MS-ILS heuristic. In the comparisons, the best values among all heuristics are in boldface. Costs equal to the BKS are shown in italic and the new best solutions found (i.e., solutions that are better than the previously best known solutions) are underlined.

Moreover, we present an assessment of the statistical significance of the variations in the average costs achieved by the compared heuristics. For this evaluation, we have used a one-tailed paired Student's t-test per instance for each pair of heuristics (MS-ILS vs. MineReduce and MDM-MS-ILS vs. MineReduce), with a significance level of 5%. The statistically significant differences are indicated, for each instance, in the average cost column of the heuristic that has the advantage.

Table 2 presents the comparison of the results for the instances from Set 1 of Duhamel et al. (2010). For this set, composed of small instances, the overall results were balanced regarding solution quality, with small differences between the heuristics. Regarding the average times, on the other hand, MineReduce obtained the best values for all instances and the best, and expressive, APD (−63.72%). A new best solution to instance 39 was found by all heuristics.

Table 3 presents the comparison of the results for the instances from Set 2 of Duhamel et al. (2010). MineReduce outperformed the other heuristics, achieving the best average costs for 26 of the 37

Table 2
Results – Set 1 of Duhamel et al. (2010).

		MS-ILS			MDM-MS-ILS			MineReduce		
I	BKS	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
08	4591.75[c–f]	4594.07	4596.13	52.1	***4591.75***	4595.89	44.6	***4591.75***	**4594.20**[†‡]	**30.2**
10	2107.55[a–f]	***2107.55***	2107.58	68.1	***2107.55***	***2107.55***	56.7	***2107.55***	***2107.55***	**29.5**
11	3367.41[d–g]	3368.50	3375.86	101.7	3368.50	**3374.01**	84.6	***3367.41***	3376.12	**49.0**
36	5684.61[de]	**5684.62**	5704.48	177.8	**5684.62**	5703.50	153.7	**5684.62**	**5702.82**	**70.5**
39	2921.36[f]	**2920.93**	**2931.17**[†]	116.5	**2920.93**	2933.49	104.6	**2920.93**	2933.75	**47.1**
43	8737.02[c–e]	8744.50	8754.30	119.9	***8737.02***	**8751.34**[‡]	105.7	8739.36	8756.97	**55.7**
52	4027.27[d–f]	**4029.42**	**4029.42**[†]	35.7	**4029.42**	**4029.42**[‡]	31.0	**4029.42**	4030.35	**15.6**
55	10244.34[a–f]	***10244.34***	**10255.81**[†]	24.5	***10244.34***	10258.72[‡]	21.2	***10244.34***	10313.29	**14.0**
70	6684.56[ef]	6688.69	6698.32	57.3	**6685.24**	**6693.99**	49.9	**6685.24**	6695.22	**29.8**
75	452.85[a–g]	***452.85***	***452.85***	4.2	***452.85***	***452.85***	3.4	***452.85***	***452.85***	**3.2**
82	4766.74[c–f]	***4766.74***	4771.10	49.6	***4766.74***	4771.26	46.7	***4766.74***	**4770.34**	**30.0**
92	564.39[a–g]	***564.39***	***564.39***	14.4	***564.39***	***564.39***	12.4	***564.39***	***564.39***	**9.1**
93	1036.99[b–g]	***1036.99***	**1037.18**[†]	15.5	***1036.99***	1037.28[‡]	13.4	***1036.99***	1038.38	**8.6**
94	1378.25[a–g]	***1378.25***	***1378.25***	35.6	***1378.25***	1378.27	31.6	***1378.25***	1378.34	**17.1**
# of wins		10	**7**	–	**13**	**7**	–	**13**	6	**14**
APD					**−0.01%**	**0.00%**	−14.28%	**−0.01%**	0.05%	**−63.72%**

[†] Statistically significant (MS-ILS vs. MineReduce).
[‡] Statistically significant (MDM-MS-ILS vs. MineReduce).
[a] Duhamel et al. (2010).
[b] Duhamel et al. (2011).
[c] Maia et al. (2018).
[d] Duhamel et al. (2013).
[e] Penna et al. (2019).
[f] Penna et al. (2013b).
[g] Kochetov and Khmelev (2015).

instances, with an APD of −0.19%. Its advantage was statistically significant for 22 instances in comparison to MS-ILS and for 19 instances in comparison to MDM-MS-ILS. MineReduce found new best solutions to five instances and the best known solutions to 12 other instances. Furthermore, MineReduce had the best average times for all instances, with the expressive APD of −65.09%.

Table 4 presents the comparison of the results for the instances from Set 3 of Duhamel et al. (2010). MineReduce again outperformed the other heuristics, achieving the best average costs for all 30 instances, with an APD of −0.46%. The differences were statistically significant for 27 instances in comparison to MS-ILS, and for 29 instances in comparison to MDM-MS-ILS. MineReduce found new best solutions to nine instances and the best known solution to another one. MineReduce also obtained the best average times for all instances, with an APD of −57.01%.

Table 5 presents the comparison of the results for the instances from Set 4 of Duhamel et al. (2010). MineReduce once more outperformed the other heuristics, achieving the best average costs for all 11 instances, with an APD of −0.52%. The differences were statistically significant in comparison to both MS-ILS and MDM-MS-ILS for all instances but one. Furthermore, MineReduce found new best solutions to six instances and obtained the best average times for all instances, with an APD of −53.24%.

Finally, we have compared the results obtained by MineReduce in our experiments to the results reported in the literature for two other state-of-the-art algorithms: HLS, presented by Kochetov and Khmelev (2015); and HILS-RVRP, presented by Penna et al. (2019).

The experiments with HLS were run on an Intel® Core™ i7 2.20 GHz CPU (the specific model was not reported), whereas those with HILS-RVRP were run on an Intel® Core™ i7-940 2.93 GHz CPU. It must be noticed that in both cases only ten runs were performed per instance, whereas we have performed 20 runs of MineReduce per instance in our experiments.

Since our experiments with MineReduce and those reported with HLS and HILS-RVRP have been performed on different CPU models, a fully precise comparison regarding computational time is not possible. Hence, in order to make this comparison as fair as possible, we consider an approximate scale ratio to compensate for the performance differences between the CPUs. The scale ratios we have adopted were based on the performance ratings from the PassMark CPU benchmarks (PassMark, 2020). Specifically, as all three algorithms were tested on single threads, we have used the single-thread performance ratings from the PassMark benchmarks.

As the specific CPU model used in the experiments with HLS could not be retrieved, we have assumed it to be the lowest-performance one among all i7 2.20 GHz models (in order to derive approximate lower bounds for the computational time reported). That was the i7-2675QM model. The single-thread ratings for the i7-2675QM 2.20 GHz (assumed for HLS), the i7-940 2.93 GHz (used for HILS-RVRP) and the i7-5500U 2.40 GHz (used for MineReduce) were 1101, 1334 and 1551, respectively. Therefore, the computational times reported for HLS have been multiplied by the ratio 1101/1551, whereas the ratio 1334/1551 has been used for the HILS-RVRP computational times.

Table 6 presents a summary of this comparison (the detailed comparison is presented in Appendix A). HILS-RVRP and MineReduce have presented the best results regarding both solution quality and computational time. The APD values presented for these algorithms are relative to HLS. Regarding solution quality, a balance is observed for the smallest instances (Set 1), whereas HILS-RVRP has the best results for medium-size instances (Set 2), and MineReduce presents the best results for the largest instances (Sets 3 and 4). Computational times reported for HILS-RVRP and MineReduce are much shorter than those reported for HLS. Between HILS-RVRP and MineReduce, the former presents shorter computational times.

5.3. Behavior analysis

To further inspect the behavior of the new MineReduce-based hybrid heuristic, additional experiments were performed using instance 02 from Set 3 of Duhamel et al. These experiments were motivated by those performed in the behavior analysis presented by Maia et al. (2018).

Table 3
Results – Set 2 of Duhamel et al. (2010).

		MS-ILS			MDM-MS-ILS			MineReduce		
I	BKS	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
05	10876.48[d]	10904.08	10973.10	129.7	10909.01	10950.87	110.8	***10876.48***	**10925.52**[†‡]	**76.4**
06	11688.64[d]	11771.85	11823.31	186.3	11723.12	11806.50	160.8	**11696.83**	**11761.32**[†‡]	**125.0**
07	8074.64[f]	8089.72	8149.27	122.0	8088.02	8141.89	105.0	**8076.53**	**8129.35**	**68.6**
12	3543.99[a–f]	***3543.99***	3546.89	175.9	***3543.99***	**3546.18**	151.0	***3543.99***	3547.65	**82.9**
13	6696.43[b–d]	6708.49	6714.25	256.3	6701.58	6714.60	219.7	**6697.58**	**6706.94**[†‡]	**102.8**
16	4156.97[b–f]	***4156.97***	4163.69	253.4	***4156.97***	**4162.15**	220.7	***4156.97***	4163.04	**126.1**
17	5362.83[bd]	**5365.52**	5380.09	120.9	5367.49	**5376.80**	103.0	5365.94	5381.80	**61.3**
2A	7793.16[b–e]	***7793.16***	7816.48	241.1	***7793.16***	**7807.30**	210.2	***7793.16***	7809.81	**113.1**
2B	8462.56[d]	8473.84	8497.01	304.6	8476.92	8502.85	260.1	**8453.35**	**8477.33**[†‡]	**137.0**
21	5139.84[cd]	5148.03	**5160.73**	245.1	***5139.84***	5162.26	214.0	***5139.84***	5162.74	**110.3**
25	7206.64[b]	**7209.29**	7250.69	537.7	7236.75	7252.06	482.2	7209.50	**7231.23**[†‡]	**281.3**
26	6393.47[f]	6448.79	6458.54	752.0	**6418.40**	**6456.43**	668.2	6438.69	6458.32	**275.7**
28	5530.55[d]	5531.30	5540.40	378.7	5535.15	5541.76	322.5	**5529.05**	**5537.72**[†‡]	**171.4**
30	6313.39[b]	**6320.14**	**6343.28**[†]	338.5	6333.82	6346.37	300.5	6331.40	6350.05	**117.9**
31	4091.52[bd]	4107.54	4125.63	519.0	4097.97	4122.48	441.9	***4091.52***	**4113.60**[†]	**218.8**
34	5747.25[d]	5778.46	5815.53	350.6	5775.58	5815.46	300.6	**5765.08**	**5792.18**[†‡]	**185.6**
40	11118.57[d]	11129.37	11163.23	300.9	11136.31	11160.28	264.6	**11111.89**	**11140.70**[†‡]	**162.0**
41	7571.44[e]	7619.76	7715.74	312.6	7619.76	7714.90	279.9	**7573.24**	**7637.37**[†‡]	**204.9**
47	16156.12[d]	16243.28	16314.30	151.4	16232.57	16299.40	134.4	***16156.12***	**16263.38**[†‡]	**83.9**
48	21287.90[e]	***21287.90***	21451.91	194.7	21383.96	21463.82	167.4	21329.71	**21413.41**[†‡]	**99.0**
51	7721.47[bd]	**7769.42**	**7794.11**	283.8	7780.04	7794.51[‡]	245.8	7780.04	7804.28	**117.0**
53	6434.83[b–e]	***6434.83***	6455.63	158.8	***6434.83***	6458.59	138.6	***6434.83***	**6448.50**[†‡]	**80.4**
60	17036.59[d]	17054.68	17101.45	247.5	17051.56	17090.47	212.9	**17045.33**	**17082.39**[†]	**125.2**
61	7292.03[cd]	7294.03	7304.63	200.6	***7292.03***	**7302.24**[‡]	181.5	***7292.03***	7305.89	**96.7**
66	12783.94[d]	12809.36	12882.42	646.0	12834.58	12889.40	465.2	**12772.07**	**12839.71**[†‡]	**295.9**
68	8935.89[c]	8997.40	9079.65	268.8	8991.80	9066.08	203.0	**8919.16**	**8992.62**[†‡]	**153.9**
73	10195.13[f]	10204.77	10212.39	281.2	10204.77	10213.31	235.0	**10203.84**	**10209.61**[†‡]	**136.7**
74	11586.58[d]	11595.36	11608.81	299.2	**11586.87**	11609.91	258.0	**11586.87**	**11599.16**[†‡]	**150.6**
79	7259.54[bd]	7275.74	7307.53	580.0	7262.91	7292.31	493.6	**7262.02**	**7290.96**[†]	**234.8**
81	10675.92[f]	**10689.90**	10702.57	181.7	10693.70	10706.12	157.3	10693.70	**10699.79**[‡]	**95.1**
83	10019.15[b]	10041.08	10052.44	219.2	10029.79	10047.39	198.3	***10019.15***	**10046.97**	**124.7**
84	7227.88[bd]	7237.13	7262.42	125.3	***7227.88***	**7258.76**[‡]	118.6	***7227.88***	7267.13	**72.4**
85	8773.08[d]	**8825.54**	8863.32	257.5	8827.98	8862.83	250.9	8827.98	**8857.80**	**171.6**
87	3753.87[a–f]	***3753.87***	3757.77	127.9	***3753.87***	3757.13	121.9	***3753.87***	**3755.06**[†]	**66.2**
88	12388.23[d]	12429.11	12512.64	156.9	12429.11	12494.25	153.4	**12402.85**	**12447.06**[†‡]	**118.6**
89	7086.36[d]	7105.15	7128.66	245.1	7100.90	7126.66	233.5	**7095.33**	**7110.97**[†‡]	**140.0**
90	2346.13[b]	2347.81	2358.88	105.3	**2346.43**	**2357.25**	87.0	**2346.43**	2357.59	**54.8**
# of wins		12	3	–	11	8	–	**29**	**26**	**37**
APD					−0.01%	−0.04%	−13.71%	**−0.15%**	**−0.19%**	**−65.09%**

[†] Statistically significant (MS-ILS vs. MineReduce)
[‡] Statistically significant (MDM-MS-ILS vs. MineReduce)
[a] Duhamel et al. (2010)
[b] Duhamel et al. (2013)
[c] Penna et al. (2013b)
[d] Penna et al. (2019)
[e] Maia et al. (2018)
[f] Kochetov and Khmelev (2015)

In the first of these experiments, the heuristics have been run with 400 iterations. Fig. 3 provides an evaluation of the solution costs obtained throughout the experiment. The charts in the first row show, for each heuristic, the solution costs obtained per iteration in the generation and local search phases, whereas the second row provides enlarged views focusing on the bottom part. In Fig. 4, the charts exhibit the computational time spent in the generation and local search phases per iteration. The dashed vertical lines indicate the iterations preceding a data mining method run.

In Fig. 3, the charts in the first row show that the reduction in the costs of the solutions generated by MineReduce after data mining is much more expressive than that observed for MDM-MS-ILS. The enlarged views in the second row show that the costs of generated solutions fall to a level even lower than that of the costs of solutions discovered through the local search in previous iterations, and the local search finds even better solutions after each run of the data mining procedure.

On the other hand, as shown in the last chart of Fig. 4, the computational time spent in the generation phase – which is close to zero in the first iterations – increases after the first run of the data mining procedure. This increase is due to the execution of the PSR process, which includes a local search on a reduced version of the problem instance. However, the increase in time spent in the generation phase is compensated by a reduction, quite expressive, in time spent in the local search phase. The amount of time spent in the generation and local search phases together per iteration is significantly reduced after data mining.

The overall reduction in computational time can be explained by the fact that MineReduce shrinks the problem instance and, consequently, the search space. Therefore, the first local search (embedded in the hybrid generation phase, over the reduced instance) performs a much smaller number of movement evaluations, thus it is much faster. Afterwards, the second local search (actual local search phase, over the original instance) starts from a high-quality solution, so it converges faster as well.

In MineReduce, as the charts show, the benefits – reduction of solution costs and computational time – are intensified after each execution of the data mining procedure, reaching very low levels in

Table 4
Results – Set 3 of Duhamel et al. (2010).

		MS-ILS			MDM-MS-ILS			MineReduce		
I	BKS	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
04	10748.17[e]	10792.86	10827.34	550.9	10772.57	10820.86	470.5	**10725.96**	**10772.83**[†‡]	**329.2**
09	7603.38[e]	7651.75	7667.25	369.0	7648.92	7663.31	313.8	**7618.34**	**7652.09**[†‡]	**225.6**
14	5644.98[a]	5663.21	5705.01	672.8	5666.74	5703.68	602.6	**5654.49**	**5671.42**[†‡]	**449.5**
15	8220.64[e]	8283.50	8313.50	505.0	8281.52	8315.08	443.2	**8229.06**	**8266.88**[†‡]	**412.0**
24	9101.47[a]	9170.10	9221.76	639.4	9163.03	9219.39	519.8	**9128.64**	**9169.28**[†‡]	**373.5**
29	9142.86[be]	9140.34	9154.66	850.1	9139.50	9158.06	728.7	**9136.41**	**9151.44**[‡]	**275.4**
33	9410.99[e]	9461.90	9494.98	824.2	9472.89	9501.33	731.5	**9411.12**	**9440.74**[†‡]	**505.4**
35	9555.92[e]	9580.51	9640.73	394.3	9594.32	9637.08	348.2	**9565.36**	**9617.15**[†‡]	**241.2**
37	6850.77[e]	6869.39	6887.86	605.9	6866.77	6890.96	525.1	**6854.99**	**6869.01**[†‡]	**345.9**
42	10817.90[e]	10931.85	11057.02	1069.6	10960.76	11045.49	956.7	**10842.11**	**10918.29**[†‡]	**767.3**
44	12191.48[e]	12241.46	12353.84	743.4	12243.66	12371.33	632.4	**12214.60**	**12308.14**[‡]	**331.0**
45	10476.25[e]	10512.17	10627.05	581.9	10512.17	10624.44	501.6	**10497.86**	**10540.84**[†‡]	**343.1**
50	12370.94[e]	12414.86	12455.66	1911.1	12405.04	12454.76	1703.6	**12333.66**	**12394.94**[†‡]	**836.6**
54	10351.97[e]	10354.94	10458.09	772.8	10354.94	10467.68	687.2	**10342.62**	**10408.42**[†‡]	**377.3**
56	31030.19[e]	31104.99	31262.19	400.9	31138.93	31267.46	358.4	**31020.27**	**31122.92**[†‡]	**280.9**
57	43378.37[c]	44856.76	45012.38	475.2	44856.76	45051.96	413.3	**44771.71**	**44854.99**[†‡]	**288.1**
59	14299.28[c]	14322.92	14388.71	1156.0	14325.27	14386.27	1019.9	**14310.58**	**14352.36**[†‡]	**511.9**
63	19951.76[a]	20149.87	20297.18	461.6	20161.32	20299.49	450.9	**19946.64**	**20047.97**[†‡]	**347.2**
64	17135.16[be]	***17135.16***	17169.00	501.2	17154.96	17170.87	454.9	***17135.16***	**17162.28**[‡]	**222.0**
67	10850.16[d]	10928.97	10982.48	898.8	10906.31	10982.45	601.3	**10861.53**	**10935.95**[†‡]	**361.8**
69	9147.54[e]	9177.52	9234.53	329.2	9177.52	9230.02	282.2	**9143.84**	**9190.45**[†‡]	**204.8**
71	9834.40[e]	9928.42	9962.99	484.6	9890.50	9960.87	418.8	**9847.58**	**9894.85**[†‡]	**253.6**
72	5883.33[b]	5929.17	5956.65	856.0	5929.17	5955.71	751.0	**5895.89**	**5929.19**[†‡]	**469.2**
76	11994.40[e]	12031.65	12088.63	464.6	12055.86	12089.98	395.1	**12009.05**	**12040.47**[†‡]	**283.4**
77	6916.01[e]	6947.64	6992.01	1529.3	6932.97	6982.48	1304.8	**6920.92**	**6959.31**[†‡]	**639.0**
78	7035.01[a]	7134.80	7146.32	1205.1	**7056.50**	7133.94	1122.3	7062.22	**7123.49**[†]	**522.9**
80	6816.89[a]	6834.00	6853.94	611.0	6829.59	6848.51	552.6	**6826.59**	**6833.72**[†‡]	**296.6**
86	9027.84[e]	9043.61	9066.53	401.9	9033.46	9064.96	390.3	**9024.02**	**9050.11**[†‡]	**236.1**
91	6374.01[c]	6405.51	6423.51	629.3	6388.10	6421.60	527.3	**6363.21**	**6384.12**[†‡]	**487.1**
95	6175.62[be]	6233.26	6239.71	458.3	6229.31	6239.54	387.0	**6227.17**	**6233.96**[†‡]	**241.9**
# of wins		1	–	–	1	–	–	**29**	**30**	**30**
APD					−0.06%	−0.01%	−13.83%	**−0.39%**	**−0.46%**	**−57.01%**

[†] Statistically significant (MS-ILS vs. MineReduce).
[‡] Statistically significant (MDM-MS-ILS vs. MineReduce).
[a] Duhamel et al. (2013).
[b] Penna et al. (2013b).
[c] Kochetov and Khmelev (2015).
[d] Maia et al. (2018).
[e] Penna et al. (2019).

Table 5
Results – Set 4 of Duhamel et al. (2010).

		MS-ILS			MDM-MS-ILS			MineReduce		
I	BKS	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
19	11686.39[d]	11742.78	11782.36	1117.8	11696.58	11772.45	998.8	**11687.12**	**11725.38**[†‡]	**676.4**
22	13068.03[a]	13130.75	13150.24	1126.1	13123.16	13153.59	994.0	**13094.74**	**13145.70**	**587.6**
23	7741.01[d]	7794.58	7825.57	743.1	7792.16	7827.61	641.0	**7752.05**	**7783.74**[†‡]	**487.5**
27	8417.62[c]	8429.44	8459.19	1396.9	8433.13	8458.55	1232.4	**8422.36**	**8441.02**[†‡]	**598.1**
32	9378.30[c]	9445.69	9505.94	1661.1	9447.41	9489.79	1434.6	**9348.55**	**9404.36**[†‡]	**1045.4**
38	11194.68[d]	11212.19	11272.86	1048.4	11245.64	11278.79	930.4	**11192.74**	**11227.01**[†‡]	**683.0**
46	24428.54[b]	24607.77	24704.02	2128.1	24537.33	24694.08	1845.3	**24404.42**	**24580.68**[†‡]	**1345.2**
49	16181.17[d]	16268.83	16365.17	2404.3	16197.43	16334.44	2091.5	**16164.00**	**16257.90**[†‡]	**1779.7**
58	23370.42[d]	23628.32	23781.84	1013.9	23549.77	23776.21	851.5	**23396.28**	**23545.34**[†‡]	**659.3**
62	22952.06[b]	23064.45	23195.80	1312.8	23065.38	23219.89	1204.5	**22903.99**	**23043.85**[†‡]	**690.7**
65	13013.89[b]	13021.94	13074.23	1442.5	13027.21	13075.46	1110.4	**12975.38**	**13049.25**[†‡]	**583.6**
# of wins		–	–	–	–	–	–	**11**	**11**	**11**
APD					−0.10%	−0.03%	−14.34%	**−0.56%**	**−0.52%**	**−53.24%**

[†] Statistically significant (MS-ILS vs. MineReduce).
[‡] Statistically significant (MDM-MS-ILS vs. MineReduce).
[a] Duhamel et al. (2013).
[b] Penna et al. (2013b).
[c] Kochetov and Khmelev (2015).
[d] Penna et al. (2019).

Table 6
Summary of the comparison of MineReduce to other state-of-the-art algorithms

		HLS			HILS-RVRP			MineReduce		
		Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
Set 1	# of wins	5	**6**	1	**13**	**6**	**10**	10	**6**	3
	APD				**−0.09%**	**−0.09%**	**−119.22%**	−0.08%	−0.03%	−89.85%
Set 2	# of wins	7	5	–	**26**	**21**	**37**	19	11	–
	APD				**−0.22%**	**−0.21%**	**−171.14%**	−0.19%	−0.18%	−140.31%
Set 3	# of wins	2	3	–	**16**	7	**29**	13	**20**	1
	APD				**−0.54%**	−0.32%	**−167.21%**	−0.51%	**−0.50%**	−147.24%
Set 4	# of wins	1	1	–	4	2	**10**	**6**	**8**	1
	APD				−0.36%	−0.22%	**−176.96%**	**−0.53%**	**−0.43%**	−156.42%
Global	# of wins	15	15	1	**59**	36	**86**	48	**45**	5
	APD				**−0.32%**	−0.23%	**−162.46%**	**−0.32%**	**−0.30%**	−136.74%

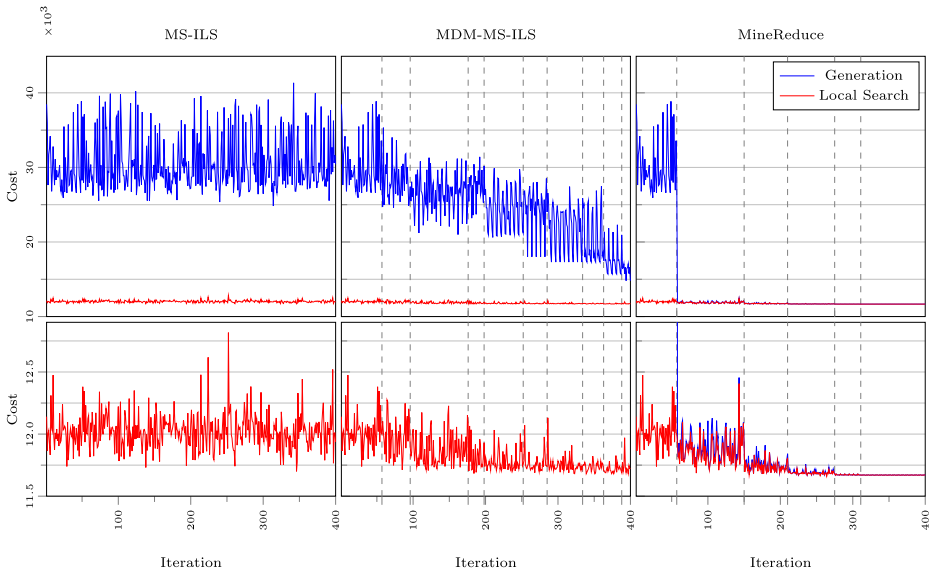


Fig. 3. Cost vs. iteration charts illustrating the behavior of MS-ILS, MDM-MS-ILS and MineReduce for instance 02 from Set 3 of Duhamel et al. over 400 iterations.

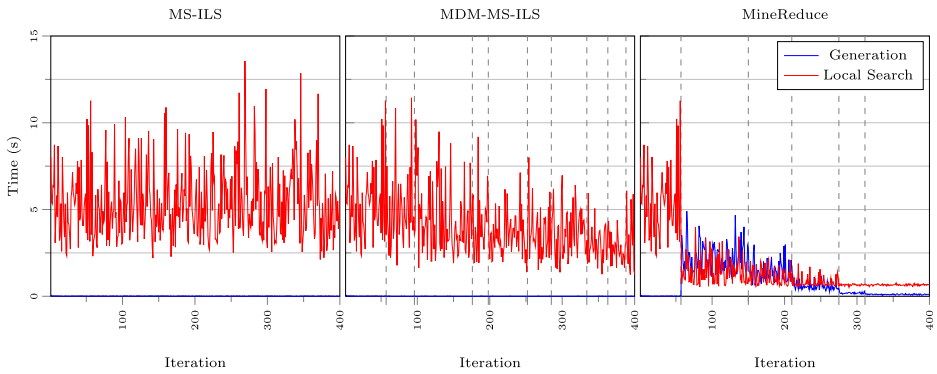


Fig. 4. Time vs. iteration charts illustrating the behavior of MS-ILS, MDM-MS-ILS and MineReduce for instance 02 from Set 3 of Duhamel et al. over 400 iterations.

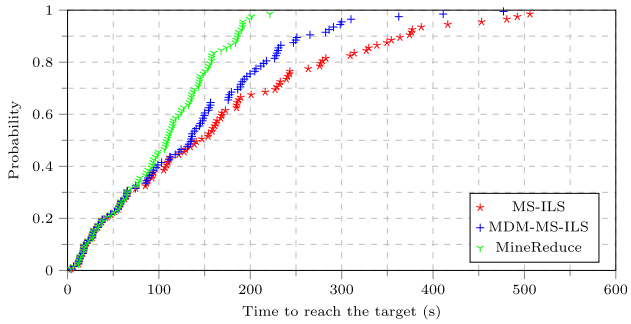


Fig. 5. TTT plots comparing all heuristics for instance 02 from Set 3 of Duhamel et al. (2010).

the last iterations, which explains the superiority demonstrated by the results of the experiments presented in Section 5.2. In this experiment, a new best solution to instance 02 of Duhamel et al., with a cost of 11660.12, was discovered by MineReduce at the 264th iteration.

The second experiment focused on the generation and evaluation of time-to-target (TTT) plots (Aiex et al., 2007). A TTT plot displays, on the ordinate axis, the probability that an algorithm will discover a solution at least as good as a given target cost within a given running time, which is shown on the abscissa axis. In this experiment, each heuristic was run 100 times, with 100 distinct random seeds, targeting a solution with a cost lower than or equal to 11780. The chart obtained is shown in Fig. 5.

The chart shows that MineReduce outperforms the other heuristics. It is possible to observe that the probability that the target will be reached within 200 s, for example, is approximately 97% for MineReduce, 76% for MDM-MS-ILS, and 67% for MS-ILS.

The analysis of the charts presented in Figs. 3–5 clearly illustrates the influence of the MineReduce approach in the behavior of the heuristics. As expected, this behavior causes a performance improvement, regarding both the quality of solutions found and the computational time spent.

6. Conclusion

Previous work that explored data science in combinatorial optimization produced highly significant results by applying patterns (found by data mining procedures) to guide the construction of initial solutions.

This work presents an approach that uses mined patterns to perform problem size reduction. The MineReduce approach was applied to extend a previous and state-of-the-art heuristic for the HFVRP (Penna et al., 2013a). The new hybrid heuristic obtained, named MineReduce-MS-ILS, produced significantly better results in terms of both solution quality and computational time when compared to the original heuristic and a previous hybrid version with data mining.

We carried out experiments on the 96 HFVRP benchmark instances from the sets of Duhamel et al. (2010) (four reserved for parameter tuning). The results attained show that the proposed approach is very promising, as the MineReduce-based heuristic reached the best average solution costs and computational times for most instances. Moreover, it obtained new upper bounds for 22 instances.

The proposed heuristic presented better performances than the MS-ILS heuristic and a previous hybrid version with data mining (MDM-MS-ILS). MineReduce attained better average costs for 83% of the instances (65% with statistical significance) in comparison to the MS-ILS and better average costs for 76% of the instances (64% with statistical significance) in comparison to the MDM-MS-ILS. If small instances (Set 1) – for which the three heuristics have presented similar performance – are excluded from the comparison, then the superiority of MineReduce is clearer revealed: better average costs for 91% of the instances (76% with statistical significance) in comparison to the MS-ILS and better average costs for 86% of the instances (74% with statistical significance) in comparison to the MDM-MS-ILS.

Furthermore, we have compared the results obtained by MineReduce in our experiments to those reported in the literature for two other state-of-the-art algorithms: HLS, presented by Kochetov and Khmelev (2015); and HILS-RVRP, presented by Penna et al. (2019). This comparison showed that MineReduce is very competitive, especially for large instances.

The reported results are evidence that heuristics based on MineReduce can generate initial solutions of better quality. Therefore, a significant improvement in the quality of solutions obtained throughout the local search phase is observed as well. Additionally, a consistent reduction of the convergence time of the local search phase is also noticed. Therefore, the proposed MineReduce approach shall be further studied and explored in future work, including its application to other heuristics and other optimization problems.

CRediT authorship contribution statement

Marcelo Rodrigues de Holanda Maia: Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Alexandre Plastino:** Conceptualization, Methodology, Writing - review & editing, Supervision. **Puca Huachi Vaz Penna:** Conceptualization, Methodology, Writing - review & editing.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was partially supported by grants 310444/2018-7 and 438473/2018-3 from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil).

Appendix A. Detailed comparison of MineReduce to other state-of-the-art algorithms

This appendix presents a detailed comparison of the results obtained by MineReduce in our experiments to the results reported

in the literature for other state-of-the-art algorithms: HLS (Kochetov and Khmelev, 2015) and HILS-RVRP (Penna et al., 2019). Tables A.7, A.8, A.9 and A.10 show the comparisons on instances of Sets 1, 2, 3 and 4 of Duhamel et al. (2010), respectively.

Table A.7
Detailed comparison of MineReduce to other state-of-the-art algorithms – Set 1 of Duhamel et al. (2010).

	HLS			HILS-RVRP			MineReduce		
I	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
08	4596.52	4597.65	180.3	**4591.75**	4595.65	**10.3**	**4591.75**	**4594.20**	30.2
10	2108.10	2108.32	131.3	**2107.55**	2107.83	**10.4**	**2107.55**	**2107.55**	29.5
11	**3367.41**	**3372.57**	305.2	**3367.41**	3373.77	**19.0**	**3367.41**	3376.12	49.0
36	5709.31	5738.34	254.1	**5684.61**	**5700.14**	**25.6**	5684.62	5702.82	70.5
39	2926.59	**2928.99**	143.4	2921.40	2932.75	**16.8**	**2920.93**	2933.75	47.1
43	8737.13	8749.47	352.1	**8737.02**	**8746.38**	64.8	8739.36	8756.97	**55.7**
52	4035.59	4035.59	43.3	**4027.27**	4030.44	**13.8**	4029.42	**4030.35**	15.6
55	10256.16	10264.37	32.7	**10244.34**	**10250.98**	**11.4**	**10244.34**	10313.29	14.0
70	6689.61	6729.87	142.7	**6684.56**	**6694.43**	**13.5**	6685.24	6695.22	29.8
75	**452.85**	**452.85**	**0.7**	**452.85**	**452.85**	1.0	**452.85**	**452.85**	3.2
82	4769.35	4772.58	127.8	**4766.74**	4771.33	31.2	**4766.74**	**4770.34**	**30.0**
92	**564.39**	**564.39**	20.6	**564.39**	564.65	**3.9**	**564.39**	**564.39**	9.1
93	**1036.99**	**1036.99**	13.5	**1036.99**	1038.34	**6.0**	**1036.99**	1038.38	8.6
94	**1378.25**	**1378.25**	31.9	**1378.25**	**1378.25**	27.0	**1378.25**	1378.34	**17.1**
# of wins	5	**6**	1	**13**	**6**	**10**	10	**6**	3
APD				**−0.09%**	**−0.09%**	**−119.22%**	−0.08%	−0.03%	−89.85%

Table A.8
Detailed comparison of MineReduce to other state-of-the-art algorithms – Set 2 of Duhamel et al. (2010).

	HLS			HILS-RVRP			MineReduce		
I	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
05	10896.35	10937.73	637.5	**10876.48**	**10897.93**	**22.9**	**10876.48**	10925.52	76.4
06	11760.08	11783.45	1031.4	**11688.64**	**11734.52**	**37.5**	11696.83	11761.32	125.0
07	**8074.64**	8135.00	398.9	8089.46	8144.80	**23.6**	8076.53	**8129.35**	68.6
12	**3543.99**	**3545.82**	448.6	**3543.99**	3547.92	**49.5**	**3543.99**	3547.65	82.9
13	6709.28	6728.61	606.2	**6696.43**	**6703.23**	**43.8**	6697.58	6706.94	102.8
16	**4156.97**	**4160.43**	767.4	**4156.97**	4164.03	**59.1**	**4156.97**	4163.04	126.1
17	5381.19	5403.10	311.6	**5362.83**	**5367.76**	**36.2**	5365.94	5381.80	61.3
2A	7820.37	7862.82	851.8	**7793.16**	**7796.54**	**37.3**	**7793.16**	7809.81	113.1
2B	8577.50	8601.68	609.8	8462.56	8499.95	**46.8**	**8453.35**	**8477.33**	137.0
21	5160.03	5175.71	592.7	**5139.84**	5166.11	**40.6**	**5139.84**	**5162.74**	110.3
25	7209.61	**7218.87**	1730.6	**7209.29**	7230.50	**106.5**	7209.50	7231.23	281.3
26	**6393.47**	**6433.33**	760.3	6433.21	6461.05	**128.2**	6438.69	6458.32	275.7
28	5538.45	5550.86	1107.4	5530.55	5542.80	**87.6**	**5529.05**	**5537.72**	171.4
30	6329.09	6361.97	390.4	**6315.70**	**6342.42**	**71.3**	6331.40	6350.05	117.9
31	4105.67	4130.97	1456.6	**4091.52**	**4112.64**	**87.9**	**4091.52**	4113.60	218.8
34	5784.25	5799.40	1110.9	**5747.25**	**5785.59**	**56.4**	5765.08	5792.18	185.6
40	11156.86	11184.18	994.5	11118.57	11171.17	**77.9**	**11111.89**	**11140.70**	162.0
41	7606.16	7643.93	1359.4	7597.27	7672.27	**58.6**	**7573.24**	**7637.37**	204.9
47	16291.49	16332.46	424.5	**16156.12**	**16247.77**	**35.4**	**16156.12**	16263.38	83.9
48	21316.55	21444.07	470.6	**21309.94**	**21391.58**	**39.3**	21329.71	21413.41	99.0
51	–	–	–	**7721.47**	**7787.85**	**50.3**	7780.04	7804.28	117.0
53	6483.51	6497.73	470.6	**6434.83**	6454.77	**31.0**	**6434.83**	**6448.50**	80.4
60	17073.80	17106.54	895.8	**17036.59**	**17055.35**	**63.1**	17045.33	17082.39	125.2
61	7308.84	7320.97	396.8	**7292.03**	**7302.40**	**32.2**	**7292.03**	7305.89	96.7
66	12790.56	12862.79	1404.1	12783.94	12922.52	**97.8**	**12772.07**	**12839.71**	295.9
68	–	–	–	8970.63	9123.03	**58.4**	**8919.16**	**8992.62**	153.9
73	**10195.13**	10215.26	1161.3	10195.33	**10195.36**	**63.3**	10203.84	10209.61	136.7
74	11598.92	11634.25	885.2	**11586.58**	**11591.23**	**70.9**	11586.87	11599.16	150.6
79	7266.75	7310.23	1197.5	**7259.54**	**7289.26**	**105.2**	7262.02	7290.96	234.8
81	**10675.92**	**10690.71**	387.6	10686.31	10700.27	**49.9**	10693.70	10699.79	95.1
83	10041.06	10050.45	950.5	10020.07	10048.17	**62.3**	**10019.15**	**10046.97**	124.7
84	7228.38	7244.86	480.6	**7227.88**	**7237.93**	**46.8**	**7227.88**	7267.13	72.4
85	8812.03	8842.44	1602.2	**8773.08**	**8818.55**	**78.4**	8827.98	8857.80	171.6
87	**3753.87**	3757.12	362.7	**3753.87**	3756.97	**27.0**	**3753.87**	**3755.06**	66.2
88	12406.93	12452.74	877.4	**12388.23**	**12405.80**	**40.0**	12402.85	12447.06	118.6
89	7099.68	7120.97	944.8	**7086.36**	**7102.98**	**65.8**	7095.33	7110.97	140.0
90	2350.68	2358.22	322.3	**2346.43**	**2356.31**	**41.1**	**2346.43**	2357.59	54.8
# of wins	7	5	–	**26**	**21**	**37**	19	11	–
APD				**−0.22%**	**−0.21%**	**−171.14%**	−0.19%	−0.18%	−140.31%

Table A.9
Detailed comparison of MineReduce to other state-of-the-art algorithms – Set 3 of Duhamel et al. (2010).

	HLS			HILS-RVRP			MineReduce		
I	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
04	10861.29	10892.04	3007.7	10748.17	10775.93	**147.6**	**10725.96**	**10772.83**	329.2
09	7671.00	7700.14	2219.0	**7603.38**	**7630.55**	**200.0**	7618.34	7652.09	225.6
14	5680.64	5705.25	2227.6	5657.62	5697.17	**307.3**	**5654.49**	**5671.42**	449.5
15	8255.95	8273.64	4050.5	**8220.64**	8285.79	**136.2**	8229.06	**8266.88**	412.0
24	9145.18	9178.51	2383.0	**9119.92**	9189.22	**140.9**	9128.64	**9169.28**	373.5
29	9151.41	9182.21	1566.7	9142.86	**9149.12**	**199.7**	**9136.41**	9151.44	275.4
33	9419.00	9452.46	4322.4	**9410.99**	9471.26	**296.7**	9411.12	**9440.74**	505.4
35	9605.62	9640.89	2324.1	**9555.92**	**9585.91**	**123.9**	9565.36	9617.15	241.2
37	6894.98	6915.59	1751.9	**6850.77**	6875.28	**211.0**	6854.99	**6869.01**	345.9
42	10940.03	11012.21	3913.5	**10817.90**	10995.75	**211.9**	10842.11	**10918.29**	767.3
44	12549.34	12604.56	2032.3	**12191.48**	12314.24	**137.0**	12214.60	**12308.14**	331.0
45	10664.81	10719.12	1787.4	**10476.25**	10614.48	**126.5**	10497.86	**10540.84**	343.1
50	12466.27	12482.40	3895.7	12370.94	12430.18	**322.0**	**12333.66**	**12394.94**	836.6
54	10433.38	10477.30	2399.3	10351.97	10435.58	**174.8**	**10342.62**	**10408.42**	377.3
56	31236.61	31286.03	1600.0	31030.19	31144.98	**223.7**	**31020.27**	**31122.92**	280.9
57	**43378.37**	**43581.83**	2046.5	44781.64	44899.36	**215.6**	44771.71	44854.99	288.1
59	**14299.28**	**14324.68**	3836.8	14304.46	14357.81	**268.5**	14310.58	14352.36	511.9
63	20154.56	20262.46	2616.6	20022.94	20281.49	**183.3**	**19946.64**	**20047.97**	347.2
64	17157.37	17180.65	1408.4	**17135.16**	**17157.79**	**91.2**	**17135.16**	17162.28	222.0
67	10971.29	11024.31	2517.9	10884.91	10945.00	**236.8**	**10861.53**	**10935.95**	361.8
69	9222.25	9270.15	1512.0	9147.54	9190.46	**101.4**	**9143.84**	**9190.45**	204.8
71	9976.24	9988.03	2425.6	**9834.40**	9915.73	**93.1**	9847.58	**9894.85**	253.6
72	5950.67	5977.67	2588.9	5903.81	5949.29	**193.7**	**5895.89**	**5929.19**	469.2
76	12007.57	**12036.84**	3027.6	**11994.40**	12040.78	**119.0**	12009.05	12040.47	283.4
77	6943.61	7036.46	2734.4	**6916.01**	6974.86	**233.8**	6920.92	**6959.31**	639.0
78	7101.20	7162.22	3014.8	**7053.62**	**7122.27**	**450.7**	7062.22	7123.49	522.9
80	6833.02	6844.95	2033.1	**6819.71**	6843.88	**182.1**	6826.59	**6833.72**	296.6
86	9056.31	9085.76	1783.2	9027.84	**9048.94**	**216.8**	**9024.02**	9050.11	236.1
91	6374.01	6398.88	3491.1	6374.27	6403.29	**363.9**	**6363.21**	**6384.12**	487.1
95	6223.54	6255.38	1332.4	**6175.62**	**6232.75**	476.9	6227.17	6233.96	**241.9**
# of wins	2	3	–	**16**	7	**29**	13	**20**	1
APD				**−0.54%**	−0.32%	**−167.21%**	−0.51%	**−0.50%**	−147.24%

Table A.10
Detailed comparison of MineReduce to other state-of-the-art algorithms – Set 4 of Duhamel et al. (2010).

	HLS			HILS-RVRP			MineReduce		
I	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time	Best Cost	Avg. Cost	Avg. Time
19	11760.02	11819.28	4849.1	**11686.39**	11745.69	**236.4**	11687.12	**11725.38**	676.4
22	13240.93	13270.10	5921.0	**13091.16**	**13134.19**	658.7	13094.74	13145.70	**587.6**
23	7769.33	7793.84	3658.6	**7741.01**	**7782.68**	**329.8**	7752.05	7783.74	487.5
27	**8417.62**	8444.78	5349.5	8422.92	8442.97	**320.1**	8422.36	**8441.02**	598.1
32	9378.30	9418.40	12004.5	9382.60	9436.70	**440.3**	**9348.55**	**9404.36**	1045.4
38	11224.72	11271.90	4384.8	11194.68	11254.27	**456.8**	**11192.74**	**11227.01**	683.0
46	24697.34	24832.07	8173.4	24566.23	24698.60	**426.2**	**24404.42**	**24580.68**	1345.2
49	16282.44	16368.02	12796.0	16181.17	16322.51	**559.7**	**16164.00**	**16257.90**	1779.7
58	23480.52	**23543.71**	6192.9	**23370.42**	23641.18	**253.7**	23396.28	23545.34	659.3
62	23035.91	23094.21	6182.9	23010.35	23097.54	**295.0**	**22903.99**	**23043.85**	690.7
65	13036.33	13102.95	5709.4	13043.54	13063.89	**248.0**	**12975.38**	**13049.25**	583.6
# of wins	1	1	–	4	2	**10**	**6**	**8**	1
APD				−0.36%	−0.22%	**−176.96%**	**−0.53%**	**−0.43%**	−156.42%

References

Aiex, R.M., Resende, M.G.C., Ribeiro, C.C., 2007. TTT plots: a perl program to create time-to-target plots. Optim. Lett. 1, 355–366. https://doi.org/10.1007/s11590-006-0031-4.

Baldacci, R., Battarra, M., Vigo, D., 2008. Routing a heterogeneous fleet of vehicles. The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, US., Boston, MA, pp. 3–27. https://doi.org/10.1007/978-0-387-77778-8_1.

Barbalho, H., Rosseti, I., Martins, S.L., Plastino, A., 2013. A hybrid data mining GRASP with path-relinking. Comput. Oper. Res. 40, 3159–3173. https://doi.org/10.1016/j.cor.2012.02.022.

Barnhart, C., Krishnan, N., Kim, D., Ware, K., 2002. Network design for express shipment delivery. Comput. Optimi. Appl. 21, 239–262. https://doi.org/10.1023/A:1013721018618.

Blum, C., Blesa, M.J., 2016. Construct, merge, solve and adapt: application to the repetition-free longest common subsequence problem. In: Chicano, F., Hu, B., García-Sánchez, P. (Eds.), Evolutionary Computation in Combinatorial Optimization. Springer International Publishing, Cham, pp. 46–57. https://doi.org/10.1007/978-3-319-30698-8_4.

Blum, C., Pinacho, P., López-Ibáñez, M., Lozano, J.A., 2016. Construct, merge, solve & adapt a new general algorithm for combinatorial optimization. Comput. Oper. Res. 68, 75–88. https://doi.org/10.1016/j.cor.2015.10.014.

Blum, C., Raidl, G.R., 2016. Hybridization based on problem instance reduction. Hybrid Metaheuristics: Powerful Tools for Optimization. Springer International Publishing, Cham, pp. 45–62. https://doi.org/10.1007/978-3-319-30883-8_3.

Chen, H., 2015. Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. Omega 56, 25–36. https://doi.org/10.1016/j.omega.2015.03.002.

Choi, E., Tcha, D.-W., 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. Comput. Oper. Res. 34, 2080–2095. https://doi.org/10.1016/j.cor.2005.08.002.

Delgadillo, F.J.D., Montiel, O., Sepúlveda, R., 2016. Reducing the size of traveling salesman problems using vaccination by fuzzy selector. Expert Syst. Appl. 49, 20–30. https://doi.org/10.1016/j.eswa.2015.11.026.

Duhamel, C., Gouinaud, C., Lacomme, P., Prodhon, C., 2013. A multi-thread GRASPxELS for the heterogeneous capacitated vehicle routing problem. Hybrid Metaheuristics. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 237–269. https://doi.org/10.1007/978-3-642-30671-6_9.

Duhamel, C., Lacomme, P., Prodhon, C., 2010. A GRASPxELS with Depth First Search Split Procedure for the HVRP. Technical Report Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes..

Duhamel, C., Lacomme, P., Prodhon, C., 2011. Efficient frameworks for greedy split and new depth first search split procedures for routing problems. Comput. Oper. Res. 38, 723–739. https://doi.org/10.1016/j.cor.2010.09.010.

Ferland, J.A., Michelon, P., 1988. The vehicle scheduling problem with multiple vehicle types. J. Oper. Res. Soc. 39, 577–583. https://doi.org/10.1057/jors.1988.97.

Fischer, T., Merz, P., 2007. Reducing the size of traveling salesman problem instances by fixing edges. Evolutionary Computation in Combinatorial Optimization: 7th European Conference, EvoCOP 2007, Valencia, Spain, April 11–13, 2007. Proceedings. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 72–83. https://doi.org/10.1007/978-3-540-71615-0_7.

Gavish, B., Pirkul, H., 1985. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. Math. Program. 31, 78–105. https://doi.org/10.1007/BF02591863.

Gavish, B., Pirkul, H., 1985. Zero-one integer programs with few constraints – efficient branch and bound algorithms. Eur. J. Oper. Res. 22, 35–43. https://doi.org/10.1016/0377-2217(85)90113-4.

Gavish, B., Srikanth, K., 1986. An optimal solution method for large-scale multiple traveling salesmen problems. Oper. Res. 34, 698–717. https://doi.org/10.1287/opre.34.5.698.

Golden, B., Assad, A., Levy, L., Gheysens, F., 1984. The fleet size and mix vehicle routing problem. Comput. Oper. Res. 11, 49–66. https://doi.org/10.1016/0305-0548(84)90007-8.

Grahne, G., Zhu, J., 2003. Efficiently using prefix-trees in mining frequent itemsets. In: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations.

Guerine, M., Rosseti, I., Plastino, A., 2016. Extending the hybridization of metaheuristics with data mining: dealing with sequences. Intell. Data Anal. 20, 1133–1156. https://doi.org/10.3233/IDA-160860.

Hoff, A., Andersson, H., Christiansen, M., Hasle, G., Løkketangen, A., 2010. Industrial aspects and literature survey: fleet composition and routing. Comput. Oper. Res. 37, 2041–2061. https://doi.org/10.1016/j.cor.2010.03.015.

Kenny, A., Li, X., Ernst, A.T., Sun, Y., 2019. An improved merge search algorithm for the constrained pit problem in open-pit mining. Proceedings of the Genetic and Evolutionary Computation Conference GECCO '19. ACM, New York, NY, USA, pp. 294–302. https://doi.org/10.1145/3321707.3321812.

Koç, Ç., Bektaş, T., Jabali, O., Laporte, G., 2016. Thirty years of heterogeneous vehicle routing. Eur. J. Oper. Res. 249, 1–21. https://doi.org/10.1016/j.ejor.2015.07.020.

Kochetov, Y.A., Khmelev, A.V., 2015. A hybrid algorithm of local search for the heterogeneous fixed fleet vehicle routing problem. J. Appl. Ind. Math. 9, 503–518. https://doi.org/10.1134/S1990478915040079.

Li, F., Golden, B., Wasil, E., 2007. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Comput. Oper. Res. 34, 2734–2742. https://doi.org/10.1016/j.cor.2005.10.015.

Li, L., Song, S., Wu, C., Wang, R., 2017. Fix-and-optimize and variable neighborhood search approaches for stochastic multi-item capacitated lot-sizing problems. Probl. Eng. Math.. https://doi.org/10.1155/2017/7209303.

Lin, C.K.Y., 2011. A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources. Comput. Oper. Res. 38, 1596–1609. https://doi.org/10.1016/j.cor.2011.01.021.

Liu, S., 2013. A hybrid population heuristic for the heterogeneous vehicle routing problems. Transp. Res. Part E Logist. Transp. Rev. 54, 67–78. https://doi.org/10.1016/j.tre.2013.03.010.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58. https://doi.org/10.1016/j.orp.2016.09.002.

Maia, M.R.H., Plastino, A., Penna, P.H.V., 2018. Hybrid data mining heuristics for the heterogeneous fleet vehicle routing problem. RAIRO-Oper. Res. 52, 661–690. https://doi.org/10.1051/ro/2017072.

Martins, D., Vianna, G.M., Rosseti, I., Martins, S.L., Plastino, A., 2018. Making a state-of-the-art heuristic faster with data mining. Ann. Oper. Res. 263, 141–162. https://doi.org/10.1007/s10479-014-1693-4.

Martins, S.L., Rosseti, I., Plastino, A., 2018. Data mining in stochastic local search. Handbook of Heuristics. Springer International Publishing, Cham, pp. 39–87. https://doi.org/10.1007/978-3-319-07124-4_11.

Min, H., 1991. A multiobjective vehicle routing problem with soft time windows: the case of a public library distribution system. Socioecon. Plan. Sci. 25, 179–188. https://doi.org/10.1016/0038-0121(91)90016-K.

Montiel, O., Delgadillo, F.J.D., 2015. Reducing the size of combinatorial optimization problems using the operator vaccine by fuzzy selector with adaptive heuristics. Math. Probl. Eng. 2015. https://doi.org/10.1155/2015/713043.

Montiel, O., Diaz-Delgadillo, F.J., Sepúlveda, R., 2013. Combinatorial complexity problem reduction by the use of artificial vaccines. Expert Syst. Appl. 40, 1871–1879. https://doi.org/10.1016/j.eswa.2012.10.011.

PassMark, 2020. CPU benchmarks. https://www.cpubenchmark.net accessed on March 27, 2020..

Penna, P.H.V., Subramanian, A., Ochi, L.S., 2013a. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J. Heuristics 19, 201–232. https://doi.org/10.1007/s10732-011-9186-y.

Penna, P.H.V., Subramanian, A., Ochi, L.S., Vidal, T., Prins, C., 2019. A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. Ann. Oper. Res. 273, 5–74. https://doi.org/10.1007/s10479-017-2642-9.

Penna, P.H.V., Vidal, T., Ochi, L.S., Prins, C., 2013b. New compound neighborhoods structures for the heterogeneous fixed fleet vehicle routing problem. Proceedings of the XLV Brazilian Symposium on Oper. Res., pp. 3623–3633.

Pessoa, A., Sadykov, R., Uchoa, E., 2018. Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. Eur. J. Oper. Res. 270, 530–543. https://doi.org/10.1016/j.ejor.2018.04.009.

Plastino, A., Barbalho, B., Santos, L.F.M., Fuchshuber, R., Martins, S.L., 2014. Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic. J. Heuristics 20, 39–74. https://doi.org/10.1007/s10732-013-9231-0.

Plastino, A., Fuchshuber, R., Martins, S.L., Freitas, A.A., Salhi, A., 2011. A hybrid data mining metaheuristic for the p-median problem. Stat. Anal. Data Min. 4, 313–335. https://doi.org/10.1002/sam.10116.

Ramos, T.R.P., Oliveira, R.C., 2011. Delimitation of service areas in reverse logistics networks with multiple depots. J. Oper. Res. Soc. 62, 1198–1210. https://doi.org/10.1057/jors.2010.83.

Resende, M.G.C., Ribeiro, C.C., 2016. Optimization by GRASP: Greedy Randomized Adaptive Search Procedures. Springer, New York, New York, NY. https://doi.org/10.1007/978-1-4939-6530-4.

Ribeiro, M.H., Plastino, A., Martins, S.L., 2006. Hybridization of GRASP metaheuristic with data mining techniques. J. Math. Model. Algorithms 5, 23–41. https://doi.org/10.1007/s10852-005-9030-1.

Santos, L.F., Martins, S.L., Plastino, A., 2008. Applications of the DM-GRASP heuristic: a survey. Int. Trans. Oper. Res. 15, 387–416. https://doi.org/10.1111/j.1475-3995.2008.00644.x.

Santos, L.F., Milagres, R., Albuquerque, C.V., Martins, S., Plastino, A., 2006. A hybrid grasp with data mining for efficient server replication for reliable multicast. IEEE Globecom, pp. 1–6.

Santos, L.F., Ribeiro, M.H., Plastino, A., Martins, S.L., 2005. A hybrid GRASP with data mining for the maximum diversity problem. Hybrid Metaheuristics: Second International Workshop, HM 2005, Barcelona, Spain, August 29–30, 2005. Proceedings. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 116–127. https://doi.org/10.1007/11546245_11.

Subramanian, A., Penna, P.H.V., Uchoa, E., Ochi, L.S., 2012. A hybrid algorithm for the heterogeneous fleet vehicle routing problem. Eur. J. Oper. Res. 221, 285–295. https://doi.org/10.1016/j.ejor.2012.03.016.

Taillard, E.D., 1999. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO-Oper. Res. 33, 1–14. https://doi.org/10.1051/ro:1999101.

Toschi, M., Lanzarone, E., Anaya-Arenas, A.M., Bélanger, V., Nicoletta, V., Ruiz, A., 2018. A fix-and-optimize variable neighborhood search for the biomedical sample transportation problem. IFAC-PapersOnLine 51, 992–997. https://doi.org/10.1016/j.ifacol.2018.08.478. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. Eur. J. Oper. Res. 234, 658–673. https://doi.org/10.1016/j.ejor.2013.09.045.

Walshaw, C., 2008. Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance. In: Blum, C., Aguilera, M.J.B., Roli, A., Sampels, M. (Eds.), Hybrid Metaheuristics: An Emerging Approach to Optimization. Springer, Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-78295-7_9.

APPENDIX B – MAIA, M. R. H.; PLASTINO, A.; SOUZA, U. S. “MineReduce-based iterated tabu search for the minimum weighted vertex cover problem”. Submitted to Applied Soft Computing

MineReduce-based iterated tabu search for the minimum weighted vertex cover problem

Marcelo Rodrigues de Holanda Maia[a,b,*], Alexandre Plastino[a], Uéverton dos Santos Souza[a]

[a] *Instituto de Computação, Universidade Federal Fluminense, Avenida General Milton Tavares de Souza s/n, Niterói, RJ, Brazil, 24210-346*
[b] *Instituto Brasileiro de Geografia e Estatística, Rua General Canabarro 706, Rio de Janeiro, RJ, Brazil, 20271-020*

Abstract

The minimum weighted vertex cover problem is a generalization of the classical vertex cover problem where each vertex has a positive weight. It is a well-known combinatorial optimization problem related to real-world applications in a wide range of fields. Since it is NP-hard, most practical approaches for this problem rely on non-exact methods, such as metaheuristics. Previous work has shown that the performance of metaheuristics can benefit from the use of data mining techniques, which can improve the obtained solutions. In a strategy that has been successfully used for over a decade, data mining techniques are applied to extract patterns from good solutions found in the early stages of the heuristic process, and these patterns are introduced into the solutions generated afterwards. Recently, a novel approach that uses data mining for problem size reduction, called MineReduce, has been proposed and achieved even more impressive results on the improvement of metaheuristics. In this work, we apply the MineReduce approach to improve the performance of a state-of-the-art heuristic for the minimum weighted vertex cover problem based on a multi-start iterated tabu search. We compared the original heuristic to its MineReduce-based version. The results show that the latter obtains better solutions while spending less computational time. Additionally, we assessed the effectiveness of the problem size reduction performed by MineReduce, comparing it to a kernelization algorithm. Despite the lack of guarantees on optimality or size-bounding, in practice, the reduction carried out by MineReduce was effective.

Keywords: Metaheuristics, Data mining, Problem size reduction, Vertex cover

1. Introduction

The Minimum Weighted Vertex Cover Problem (MWVCP) is a well-known combinatorial optimization problem that generalizes the classical vertex cover problem and has a wide range of applications. Given an

*Corresponding author
Email addresses: mmaia@ic.uff.br, marcelo.h.maia@ibge.gov.br (Marcelo Rodrigues de Holanda Maia), plastino@ic.uff.br (Alexandre Plastino), ueverton@ic.uff.br (Uéverton dos Santos Souza)

undirected graph where each vertex has a positive weight, in the MWVCP we are asked to find a subset of the vertices which cover all edges of the graph (a vertex cover) and has the minimum total weight.

Since the MWVCP is NP-hard, most practical approaches for this problem rely on non-exact methods, such as metaheuristics.

Previous works show that hybrid metaheuristics that incorporate data mining techniques can find better solutions within a shorter time when compared to their non-hybridized versions and other state-of-the-art methods [1, 2, 3]. They apply patterns extracted from good solutions (found by data mining procedures) to guide the construction of new solutions.

Recently, a novel approach for the hybridization of data mining and metaheuristics named MineReduce was proposed [4]. It uses mined patterns to perform problem size reduction (PSR) – a process in which a problem instance is reduced to a smaller-size version, the reduced instance is solved, and the solution found is expanded, so it becomes a solution to the original instance. The MineReduce approach has already been applied to extend a previous and state-of-the-art heuristic for the heterogeneous fleet vehicle routing problem, producing significantly better results in terms of both solution quality and computational time.

In this work, we apply the MineReduce approach to improve the performance of a state-of-the-art heuristic for the MWVCP based on a multi-start iterated tabu search [5]. We compared the original heuristic to its MineReduce-based version. The results show that the MineReduce-based version obtains better solutions while expending less computational time.

Additionally, to assess how effective is the problem size reduction performed by MineReduce, we compare it to a kernelization algorithm. Kernelization is an exact method, based on the parameterized complexity theory [6], for reducing in polynomial time a problem instance to a kernel. A kernel can be regarded as a bounded-size version of the original instance such that an optimal solution for the original instance can be straightforwardly derived from an optimal solution for the kernel. Despite the lack of guarantees on optimality or size-bounding, the practical performance of the reduction carried out by MineReduce was effective.

The remainder of this article is organized as follows. Section 2 presents a review of related work on the MWVCP and also on the hybridization of metaheuristics with data mining. In Section 3, we present the MineReduce-based heuristic for the MWVCP. Section 4 presents and analyzes the outcomes of our experiments regarding the general performance of MineReduce. In Section 5, we assess its problem size reduction effectiveness. Finally, Section 6 provides conclusions.

2. Related work

2.1. The minimum weighted vertex cover problem

The Minimum Weighted Vertex Cover Problem (MWVCP) is a well-known combinatorial optimization problem. Given an undirected graph where each vertex has a positive weight, in MWVCP we are asked to find a subset of the vertices which cover all edges of the graph (a vertex cover) and has the minimum total weight.

It is a generalization of the classical vertex cover problem, which lies in the roots of the theory of NP-completeness as one of Karp's 21 NP-complete problems [7]. The vertex cover problem is also central in the parameterized complexity theory [8] and the integer variant of the MWVCP (where the weights are positive integers) is known to be fixed-parameter tractable as well [9].

Beyond its theoretical interest, the MWVCP has many practical applications. For instance, graphs are naturally well-suited for modelling transportation networks. Vertices can represent locations of interest (such as cities or road intersections, for instance), each edge can represent a link between two locations (such as a road or a rail line, for instance), and the weights of the vertices can represent costs (or other valuation attributes) associated to the respective locations. In such a scenario, minimum weighted vertex covers can be useful in many real-world applications, which include the identification of critical nodes [10], the placement of charging stations for electric vehicles [11], and the placement of monitoring devices [12].

Since the MWVCP is NP-hard, most practical approaches for this problem rely on heuristic methods. The following approaches stand among the most important ones proposed for this problem in the last decades. Shyu et al. [13] proposed an ant colony optimization algorithm (ACO). Jovanovic and Tuba [14] proposed a hybridization of the ACO with a strategy that uses information about the best-found solution to perform some corrections on the pheromone trail (ACO+SEE). Bouamama et al. [15] proposed an algorithm that, at each iteration, establishes a population of solutions and refines it using a randomized iterated greedy heuristic (PBIG). Li et al. [16] designed a local search framework based on a weighted configuration checking strategy, a dynamic scoring strategy and a vertex selection strategy (DLSWCC). Xie et al. [17] proposed a particle swarm optimization algorithm that relies on a translation of the MWVCP into the minimal test cost attribute reduction problem (IQPSO-R). Zhou et al. [5] proposed a multi-start heuristic based on an iterated tabu search (MS-ITS), which was the base heuristic for the implementation of our MineReduce version.

2.2. Metaheuristics hybridization with data mining

Multi-start local search heuristics are iterative methods in which each iteration has two phases: generation and local search. In the generation phase, an initial solution is constructed, whereas, in the local search phase, the solution is improved. Each iteration produces a solution (usually a local optimum), and the best overall solution is returned.

Hybrid multi-start heuristics that incorporate data mining techniques have been applied, achieving superior results, to several combinatorial optimization problems [1, 2, 3]. They build an elite set by storing the best solutions found until an interruption criterion is satisfied. Once this criterion is satisfied, a data mining procedure is triggered. It mines patterns from the solutions in the elite set, which are then used in the generation phase of subsequent iterations. The idea is that, by starting from more promising initial solutions, the local search can find even better solutions within a shorter convergence time. A survey on this subject was presented in [1].

The data mining procedure used in these hybridizations relies on the formulation of the frequent itemset mining problem, which is one step of the association rule mining process [18]. This problem can be defined as follows. Let $C = \{c_1, c_2, ..., c_\eta\}$ be the set of all items in the application domain. A transaction T is a subset of C, and a dataset D is a set of transactions. An itemset I with absolute support sup in D is a subset of C that occurs in sup of the transactions in D. The frequent itemset mining problem consists in extracting from D all itemsets with support greater than or equal to $minsup$, which is a parameter that specifies the minimum support for an itemset to be considered frequent. In the data mining hybridization context, the elite set is the dataset, and each solution is a transaction. More specifically, the strategy adopted in the hybridization is based on maximal frequent itemset mining. A frequent itemset is maximal if none of its supersets is frequent. The FPmax* algorithm [19] is employed in this approach to mine maximal frequent itemsets.

More recently, a novel approach, named MineReduce, has been proposed and successfully applied to the heterogeneous fleet vehicle routing problem, achieving better results than the previous data mining hybridization approaches [4]. In the MineReduce approach, mined patterns are used to perform PSR. It relies on the principle that mined patterns are composed of structures that are likely to be in an optimal solution, so these structures can just be contracted or removed from the instance (since we assume they should be part of the searched solution).

In the heuristics that are based on this approach, the generation phase of the multi-start framework is based on a PSR process. It reduces the size of the problem instance, searches for solutions to the reduced instance and, afterwards, maps the solution found to a solution to the original instance.

3. Applying the MineReduce approach to the MWVCP

3.1. State-of-the-art heuristic

A multi-start iterated tabu search (MS-ITS) heuristic for the MWVCP [5] is used as a basis for the incorporation of the MineReduce approach. It has been tested on widely used benchmark instances, obtaining highly competitive solution quality and computational efficiency with respect to the state-of-the-art algorithms in the literature.

The steps of the MS-ITS heuristic are presented – in a high level of abstraction (we suppress details that are irrelevant in the context of this work) – in Algorithm 1. Based on the multi-start framework, it iteratively alternates between an initial solution construction procedure (line 2) and an ITS procedure (line 3). After N_{start} iterations, the best solution found is returned (line 5).

Algorithm 1 MS-ITS(N_{start})

1: **for** $iter \leftarrow 1$ **to** N_{start} **do**
2: $V' \leftarrow$ GenerateInitialSolution()
3: ITS(V', V'_{gbest})
4: **end for**
5: **return** V'_{gbest}

In the initial solution construction, the vertex set V' is first set to be empty. Then, every edge is evaluated and, if it is uncovered, one of its incident vertices must be chosen to be inserted into V'. Two strategies for choosing one of the vertices are employed: random and greedy. More precisely, the greedy mechanism chooses the vertex with less weight, whereas the other randomly chooses one of the candidate vertices with equal probabilities. Additionally, the probabilities of selecting one mechanism or the other are also equal. The ITS procedure searches for local optima around the initial solution and updates the global best solution (V'_{gbest}) when there is an improvement.

3.2. MineReduce-based heuristic for the MWVCP

The MineReduce approach relies on a PSR process based on patterns mined from a set of good solutions, i.e., substructures that are likely to be part of an optimal solution.

For the MWVCP, each mined pattern is a set of vertices considered likely to be part of an optimal solution. We propose a MineReduce-based version of the MS-ITS heuristic, called MineReduce-MS-ITS, in which the PSR is accomplished by deleting the vertices that are part of a mined pattern (assuming they are part of the solution) and then deleting the remaining isolated vertices (which, in turn, cannot be part of the solution). Its pseudocode is presented in Algorithm 2.

Whenever the elite set E becomes stable – that is, when it remains unmodified over δ iterations – the data mining procedure is performed, which updates the pattern set P (line 3). In the initial iterations, when P is still empty since data mining has not been performed yet, the original construction procedure generates the initial solutions (line 6). Once data mining has been performed, the initial solutions start being generated by the MineReduce-based construction procedure (MineReduceGeneration(p)) using a pattern p selected from the current pattern set P (lines 8–9). Algorithm 3 presents the pseudocode of the MineReduce-based constructive method.

Algorithm 2 MineReduce-MS-ITS($N_{start}, d, MaxP, MinSup, \delta$)

1: **for** $i \leftarrow 1$ **to** N_{start} **do**
2: **if** Stable(E, δ) **then**
3: $P \leftarrow$ Mine($E, MaxP, MinSup$)
4: **end if**
5: **if** $P = \emptyset$ **then**
6: $V' \leftarrow$ GenerateInitialSolution()
7: **else**
8: $p \leftarrow$ NextPattern(P)
9: $V' \leftarrow$ MineReduceGeneration(p)
10: **end if**
11: ITS(V', V'_{gbest}, E, d)
12: **end for**
13: **return** V'_{gbest}

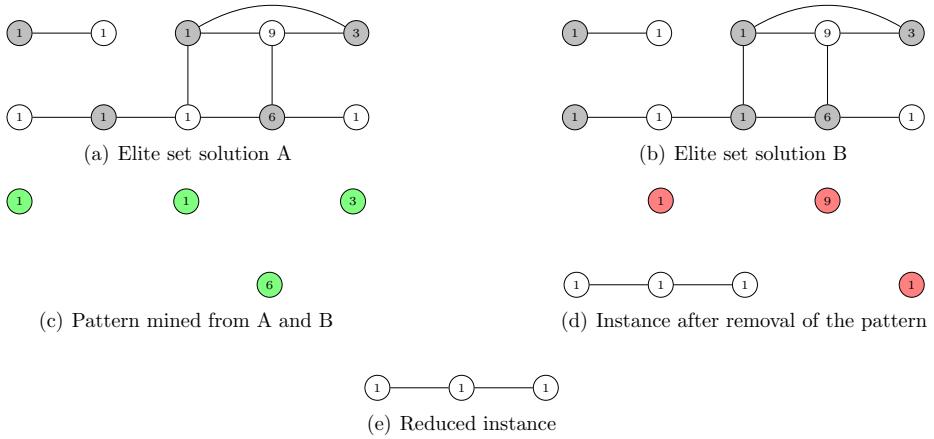
Algorithm 3 MineReduceGeneration(p)

1: ReduceInstance(p)
2: $V' \leftarrow$ GenerateInitialSolution()
3: ITS(V', V'_{rbest})
4: $V'' \leftarrow$ ExpandSolution(V'_{rbest}, p)
5: **return** V''

First, the problem size reduction method reduces the instance, i.e., it deletes the vertices in pattern p, as well as the remaining isolated vertices (line 1). The original heuristic's methods for the generation (line 2) and local search (line 3) are used to find a solution for the reduced instance. Then, the best solution found for the reduced instance (V'_{rbest}) is expanded, i.e., the vertices in p are added back, so it becomes a solution to the original instance (line 4). Finally, the resulting solution (V'') is returned. The solution obtained through this process is used as an initial solution for the local search on the original instance. Based on the observed trend, it is expected that, by starting from higher quality initial solutions, the local search will find even better solutions, within less computational time.

Figure 1 illustrates the application of MineReduce's PSR to an instance from [13] (vc_10_10_ds_05). Figures 1a and 1b present two solutions (gray vertices) that compose an elite set for the instance. The data mining procedure identifies a frequent pattern containing four vertices, shown in Figure 1c. The first step of MineReduce's reduction is to mark the vertices in the pattern as part of the solution and delete them. The removal of the pattern, which produces the graph presented in Figure 1d, leaves some isolated vertices

(coloured in the figure). Then, the second and final step is to mark these remaining isolated vertices as non-part of the solution and also delete them. Figure 1e shows the reduced instance.



(a) Elite set solution A

(b) Elite set solution B

(c) Pattern mined from A and B

(d) Instance after removal of the pattern

(e) Reduced instance

Figure 1: MineReduce's PSR applied to an MWVCP instance

4. Experimental results

In this section, we report experimental results to provide evidence of the improvements obtained from incorporating the MineReduce approach into a heuristic method. First, we have implemented the MS-ITS heuristic (Algorithm 1) as described in [5]. Then, we built its enhanced version, which applies the MineReduce approach, called MineReduce-MS-ITS (MineReduce for short).

We compare the performance of the MineReduce-based heuristic with the original MS-ITS on standard MWVCP benchmark instances. Additionally, we implemented the kernelization algorithm presented in Section 5.1 and compared the kernels produced by it with the reduced instances obtained by MineReduce.

4.1. Experimental setup

Our computational experiments were carried out on the standard MWVCP benchmark instances originally proposed in [13]. Each instance consists of an undirected and vertex-weighted graph with n vertices and m edges. These instances are divided into three sets: SPI, with 400 small-scale instances (n between 10 and 25, m between 10 and 200); MPI, with 710 middle-scale instances (n between 50 and 300, m between 50 and 5000); and LPI, with 15 large-scale instances (n between 500 and 1000, m between 500 and 20000).

Furthermore, sets SPI and MPI are subdivided into two types of instances. In instances of Type I, weights and degrees of vertices are not interrelated, whereas, in instances of Type II, they are (the weight $w(i)$ on vertex i is randomly distributed over the interval $[1, d(i)^2]$, where $d(i)$ is the degree of vertex $i, 1 \leq i \leq n$). All instances in the LPI set are of Type I.

For the SPI and MPI sets, there are ten instances of each type, for each combination of n and m, whereas for the LPI set, there is only one instance for each combination. Each heuristic was run 30 times for each instance. Like in previous state-of-the-art studies [15, 14, 13, 5], the results for sets SPI and MPI are presented as the average values of all runs for each combination of n and m.

Table 1 reports the parameters setting. The top group is composed of the MS-ITS related parameters, whereas the bottom group is composed of the data mining hybridization related ones.

Table 1: Parameters setting

Parameter	Description	Value
α	Consecutive iteration number of perturbation without improvement	$n/3 + 50$
β	Consecutive iteration number of tabu search without improvement	50
γ	Probability employed in tabu search strategy	1/3
τ	Percent of key-vertex set removed in the perturbation phase	$1/random(3, 6)$
λ	Tabu tenure	$20 + random(5)$
N_{start}	Maximum restart times	100
d	Maximum size of the elite set	10
$MaxP$	Maximum size of the patterns set	10
$MinSup$	Minimum support used in the data mining procedure	0.7
δ	Number of non-changing iterations to consider the elite set stable	5 (5% of N_{start})

The MS-ITS related parameters setting is the same adopted in [5], except for N_{start} (the number of multi-start iterations), which was 20 in that work, and we set to 100. This way, the effects of the MineReduce approach could be more easily observed and analyzed. The data mining hybridization related parameters setting is the same adopted in previous applications (e.g., [3]) except for $MinSup$. The mentioned applications used a minimum support value of 0.2. That means substructures present in at least 20% of the elite set instances could be taken as patterns. As the minimum support is increased, the criterion to take a solution substructure as a pattern gets more restrictive, which leads to a trade-off: lower values produce larger and more numerous patterns whereas patterns with higher support are more likely to be part of an optimal solution. Therefore, we have chosen to use a higher value for the minimum support: 0.7.

4.2. Performance comparison

This section summarizes results from experiments comparing the performance of the MineReduce heuristic to the original MS-ITS heuristic regarding solution quality and computational time. Tables 2, 3 and 4 report these results in detail. Each table contains two additional rows: one presenting the number of wins or ties (#wins‖ties) -– i.e., the number of cases for which the corresponding heuristic obtained the best result -– and another showing the average percentage difference (Avg. P.D.) achieved by MineReduce (with respect to the MS-ITS heuristic). The best values for each comparison are presented in bold.

Both heuristics obtained optimal solutions for all instances in the SPI set in less than 0.2s on average. It is worth mentioning that optimal solutions are still unknown for the problem instances of sets MPI and LPI.

The results from tests on set MPI, reported in Tables 2 and 3, show both heuristics present equivalent performances for the smallest instances, whereas MineReduce is dominantly superior in terms of solution quality for the largest instances. For MPI instances of Type I, in comparison with the original state-of-the-art heuristic (MS-ITS), MineReduce obtained better results in 67% of the cases (21 out of 39) and equal results in the other 33% of the cases (13 out of 39). For MPI instances of Type II, MineReduce obtained better results in 25% of the cases (8 out of 32) and equal results in 75% of the cases (24 out of 32). Furthermore, MineReduce has better times for most cases: 87% of the cases (34 out of 39) for Type I and 69% of the cases (22 out of 32) for Type II.

The results from tests on set LPI (Table 4), the one with the largest instances, show a clear dominance of MineReduce, which overcame the original MS-ITS heuristic regarding average objective values for all 15 instances.

Additionally, we have assessed the statistical significance of the variations in the average costs obtained. For this evaluation, we have used a paired Wilcoxon signed-rank test per instance, with a significance level of 5%. The tests showed the improvements obtained by MineReduce were statistically significant for 94 out of the 390 MPI instances of Type I, 18 out of the 320 MPI instances of Type II, and all 15 LPI instances.

These results indicate that MineReduce presented better performance than the original MS-ITS heuristic in terms of solution quality and running time.

4.3. Behaviour analysis

Additional experiments for a behaviour analysis like the one presented in [4] were performed, using instance vc_800_2000. The result of the first of these experiments is presented in Fig. 2. The charts in the first row present the solution costs obtained by the respective heuristics per iteration in each phase (generation and local search). The second row displays enlarged views with a focus on the lower region. The dashed vertical lines indicate the iterations that precede the triggering of data mining.

Table 2: Results for set MPI (Type I)

		MS-ITS		MineReduce	
n	m	Avg	Time	Avg	Time
50	50	**1280.0**	0.125	**1280.0**	**0.121**
	100	**1735.3**	**0.113**	**1735.3**	0.128
	250	**2272.3**	**0.183**	**2272.3**	0.189
	500	**2661.9**	**0.278**	**2661.9**	0.287
	750	**2951.0**	0.343	**2951.0**	**0.331**
	1000	**3193.7**	0.339	**3193.7**	**0.310**
100	100	**2534.2**	0.214	**2534.2**	**0.209**
	250	3601.7	0.370	**3601.6**	**0.350**
	500	**4600.6**	**0.556**	**4600.6**	0.567
	750	**5045.5**	0.694	**5045.5**	**0.631**
	1000	**5508.2**	0.863	**5508.2**	**0.754**
	2000	**6051.9**	1.297	**6051.9**	**1.106**
150	150	**3666.9**	0.419	**3666.9**	**0.384**
	250	4720.1	0.559	**4720.0**	**0.536**
	500	6166.8	0.870	**6165.4**	**0.808**
	750	6956.6	1.142	**6956.5**	**1.004**
	1000	7360.8	1.293	**7360.1**	**1.164**
	2000	8549.8	1.970	**8549.6**	**1.806**
	3000	8899.9	2.441	**8899.8**	**2.245**
200	250	**5551.6**	0.794	**5551.6**	**0.778**
	500	7194.4	1.244	**7193.8**	**1.168**
	750	8270.9	**1.523**	**8270.5**	1.542
	1000	9150.1	1.795	**9147.1**	**1.673**
	2000	10833.6	2.683	**10830.8**	**2.404**
	3000	11596.8	3.623	**11596.3**	**3.145**
250	250	6148.8	1.150	**6148.7**	**0.975**
	500	8443.7	1.777	**8439.7**	**1.556**
	750	9754.2	2.237	**9747.3**	**2.006**
	1000	10757.2	2.492	**10752.8**	**2.245**
	2000	12757.2	3.715	**12753.3**	**3.206**
	3000	13726.1	4.656	**13723.8**	**3.964**
	5000	14674.1	6.070	**14669.9**	**5.342**
300	300	7296.1	1.560	**7295.8**	**1.354**
	500	9413.7	2.235	**9404.1**	**1.960**
	750	11043.3	2.790	**11031.8**	**2.487**
	1000	12108.1	3.207	**12104.4**	**2.810**
	2000	14742.7	4.632	**14735.0**	**4.086**
	3000	15852.4	5.870	**15843.1**	**5.036**
	5000	17354.8	7.757	**17345.2**	**6.754**
#wins‖ties		13	5	**39**	**34**
Avg. P.D.				-0.020%	-8.475%

The decrease in the costs of solutions generated by MineReduce after data mining is expressive. The charts in the second row show that the costs obtained in the generation phase are even lower than those of solutions found in the local search phase of previous iterations, and even better solutions are found after each run of the data mining procedure. Furthermore, it can be noticed that the solutions generated by

Table 3: Results for set MPI (Type II)

		MS-ITS		MineReduce	
n	m	Avg	Time	Avg	Time
50	50	**83.7**	**0.120**	**83.7**	0.122
	100	**271.2**	**0.098**	**271.2**	0.122
	250	**1853.4**	**0.174**	**1853.4**	0.201
	500	**7825.1**	0.265	**7825.1**	**0.259**
	750	**20079.0**	0.346	**20079.0**	**0.303**
100	50	**67.2**	**0.112**	**67.2**	0.132
	100	**166.6**	**0.194**	**166.6**	0.218
	250	**886.5**	**0.374**	**886.5**	0.381
	500	**3693.6**	0.545	**3693.6**	**0.492**
	750	**8680.2**	0.710	**8680.2**	**0.615**
150	50	**65.8**	**0.172**	**65.8**	0.207
	100	**144.0**	**0.314**	**144.0**	0.347
	250	**615.8**	0.631	**615.8**	**0.586**
	500	**2331.5**	0.918	**2331.5**	**0.843**
	750	**5698.5**	1.137	**5698.5**	**1.023**
200	50	**59.6**	**0.214**	**59.6**	0.226
	100	**134.5**	**0.419**	**134.5**	0.424
	250	**483.1**	0.922	**483.1**	**0.820**
	500	**1803.9**	1.443	**1803.9**	**1.229**
	750	**4043.5**	1.765	**4043.5**	**1.407**
250	250	**419.0**	1.248	**419.0**	**1.150**
	500	1434.3	1.998	**1434.2**	**1.595**
	750	3256.2	2.393	**3256.1**	**1.925**
	1000	5986.2	2.796	**5986.1**	**2.278**
	2000	25640.3	3.961	**25637.9**	**3.165**
	5000	170349.9	6.640	**170297.4**	**5.157**
300	250	**399.4**	1.529	**399.4**	**1.378**
	500	**1216.4**	2.607	**1216.4**	**2.193**
	750	2639.4	3.245	**2639.3**	**2.715**
	1000	**4796.0**	3.790	**4796.0**	**3.028**
	2000	20890.6	5.106	**20884.4**	**4.015**
	5000	141243.6	8.739	**141231.0**	**6.903**
#wins‖ties		24	10	**32**	**22**
Avg. P.D.				-0.003%	-7.808%

MineReduce achieved a level of quality that could not be improved by the local search in most iterations, which is shown by the overlapping of the curves.

In the second experiment, we produce time-to-target (TTT) plots [20]. A TTT plot presents the probability (ordinate axis) that an algorithm will find a solution with a cost equal to or lower than a given target within a given computational time (abscissa axis). The target cost used was 31660. The chart obtained is presented in Fig. 3. It shows that MineReduce outperforms the MS-ITS heuristic. The results show the probability that the target will be reached within 40 seconds, for example, is nearly 100% for MineReduce, and approximately 46% for MS-ITS.

Table 4: Results for set LPI

			MS-ITS			MineReduce		
n	m	BKS	Best	Avg	Time (s)	Best	Avg	Time (s)
500	500	12616[a]	12623	12634.1	4.719	**12616**	**12621.0**	**4.581**
	1000	16465[a]	16478	16501.5	6.831	**16466**	**16468.3**	**6.310**
	2000	20863[a]	**20863**	20893.2	9.583	20869	**20875.3**	**8.846**
	5000	27241[b]	27250	27381.0	14.776	**27241**	**27252.3**	**13.988**
	10000	29573[c]	29588	29744.1	20.617	**29573**	**29596.1**	**19.871**
800	500	15025[a]	**15025**	15038.8	9.241	**15025**	**15025.0**	**7.843**
	1000	22747[a]	22756	22790.3	16.817	**22747**	**22753.0**	**14.337**
	2000	31283[d]	31560	31654.9	24.650	**31338**	**31375.1**	**23.451**
	5000	38553[b]	38795	38998.6	**34.228**	**38620**	**38714.2**	35.635
	10000	44351[b]	44552	44751.4	44.839	**44356**	**44436.0**	**42.975**
1000	1000	24723[e]	24776	24823.5	24.352	**24723**	**24750.6**	**20.654**
	5000	45203[e]	45424	45864.4	49.964	**45260**	**45310.7**	**48.327**
	10000	51378[a]	51841	52345.1	**62.913**	**51462**	**51633.8**	64.907
	15000	57994[e]	58484	58905.0	77.521	**58055**	**58247.5**	**76.612**
	20000	59651[e]	60133	60442.1	91.117	**59717**	**59910.8**	**88.889**
#wins‖ties			2	-	2	**14**	**15**	**13**
Avg. P.D.						-0.304%	-0.588%	-5.723%

[a]BKS first reported in [15], [b]BKS first reported in [5], [c]BKS first reported in [14], [d]BKS first reported in [17], [e]BKS first reported in [16]

5. Problem size reduction assessment

In this section, we assess the effectiveness of the PSR carried out by MineReduce and compare it to a kernelization algorithm. Kernelization is an exact method, based on the parameterized complexity theory [6], for reducing in polynomial time a problem instance to a kernel. A kernel can be regarded as a bounded-size version of the original instance such that an optimal solution for the original instance can be straightforwardly derived from an optimal solution for the kernel.

To the best of our knowledge, this is the first time that a methodology to evaluate and compare kernelization algorithms with other PSR techniques is formulated.

The kernelization algorithm used in our comparison is presented in Section 5.1. Our assessment methodology is defined in Section 5.2, and the results are shown in Section 5.3.

5.1. A simple kernelization algorithm for the MWVCP

A simple kernelization algorithm for the MWVCP can be obtained through a straightforward adaptation of a kernelization algorithm for the vertex cover problem from [21]. First, we define a parameterized version of MWVCP, where the parameter k is an upper bound on the optimal solution value of a weighted input graph G. In the following reduction rules, $N(v)$ denotes the neighbourhood of vertex v, $w(v)$ denotes the weight of vertex v, $w(S)$ denotes the total weight of all vertices in S, and $\mu(G)$ denotes the minimum vertex weight of G.

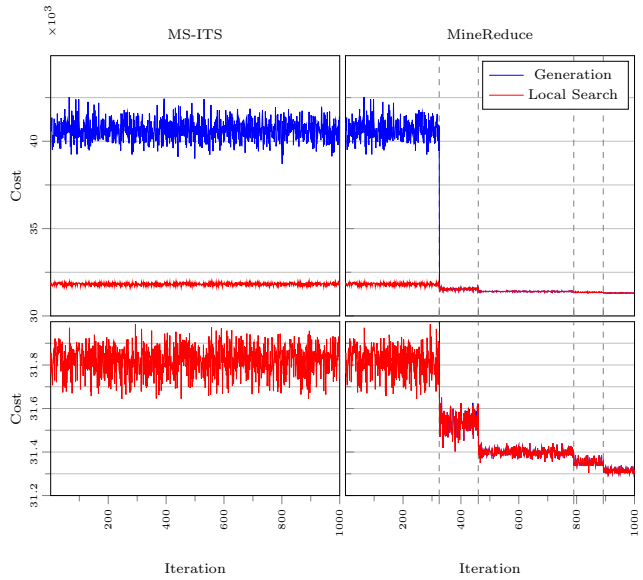


Figure 2: Cost vs. iteration charts illustrating the behavior of MS-ITS and MineReduce over 1000 iterations (on instance vc_800_2000)

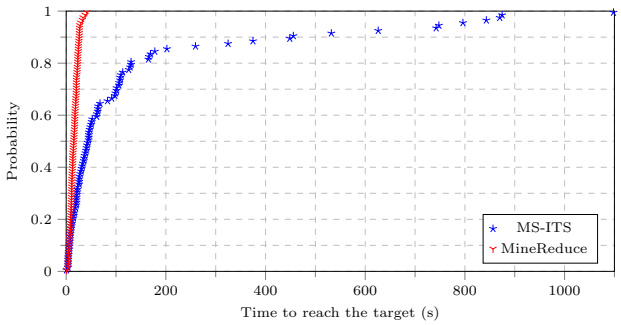


Figure 3: TTT plots comparing MS-ITS and MineReduce (on instance vc_800_2000, with target cost 31660)

Rule MWVCP-1: If G contains an isolated vertex v, then delete v from G. The new instance is $(G - v, k)$.

Rule MWVCP-2: If there is a vertex v such that $w(N(v)) > k$, then delete v (and its incident edges) from G, and decrement k by $w(v)$. The new instance is $(G - v, k - w(v))$.

The algorithm consists of repeatedly applying rules MWVCP-1 and MWVCP-2 until neither of their conditions is satisfied. This process completely removes the vertices of degree 0 (which are certainly out of an optimal solution) and those with a neighbourhood of total weight at least $k+1$ (which are certainly part of an optimal solution). This algorithm produces a kernel of the original instance with at most $(k/\mu(G))^2+k/\mu(G)$ vertices and $(k/\mu(G))^2$ edges.

5.2. Methodology

Kernelization algorithms are usually evaluated and compared based on the order of the upper bounds they impose on the size of kernels. Another relevant measure for a kernelization algorithm is its asymptotic complexity, even though all of them are polynomial by definition. Due to its heuristic nature, MineReduce does not guarantee the preservation of optimal solutions nor an upper bound on the size of the reduced instances. Therefore, they need to be evaluated experimentally regarding these properties.

To define suitable metrics for this assessment, we present an alternative interpretation for PSR (kernelization being regarded as a special kind of PSR). PSR can be regarded as a binary classification problem in which the objective is to analyze each substructure of a problem instance and determine whether it must be part of an optimal solution or not. The substructures classified by a PSR method are removed from the instance, whereas the remaining substructures (those the PSR method is not able to classify) compose the reduced instance.

Given the above interpretation, we can rely on assessment metrics for classification problems [22] to experimentally evaluate PSR methods. We can refer to "in solution" as the positive class and to "out of solution" as the negative class. Then, for a given optimal solution, there are five possible outputs for each choice made: true positive (TP) is an element that is part of the solution and is correctly classified; false positive (FP) is an element that is not part of the solution but is misclassified; true negative (TN) is an element that is not part of the solution and is correctly classified; false negative (FN) is an element that is part of the solution but is misclassified; and non-classified (NC) is an element that cannot be classified by the PSR method (and, hence, is part of the reduced instance).

Both the kernelization algorithm from Section 5.1 and MineReduce rely primarily on positive classification. Negative classification is a consequence of the positive classification decisions (the deletion of vertices identified as part of a solution leaves isolated vertices in the graph, which in turn are fixed out of the solution).

Therefore, the following classification assessment metrics were found to be the most suitable for this evaluation:

- Precision: the proportion of true positives to the total number of elements classified as positive, given by $P = TP/(TP + FP)$.

- Recall: the proportion of true positives to the total number of actual positive (AP) elements, given by $R = TP/AP$.
- F-measure: also known as F-score, represents the harmonic mean of precision and recall, given by $F = 2 \times (P \times R)/(P + R)$.

The values for these metrics vary in a range between 0 and 1, being 1 the optimal value.

A false positive in this context causes any solution obtained from the reduced instance to be non-optimal. Therefore, the precision of a PSR outcome (reduced instance), which is inversely proportional to FP, is related to its capacity of outputting an optimal solution. On the other hand, the recall of a PSR outcome is related to the degree of size reduction it represents regarding the original instance. Finally, the F-measure combines precision and recall in one score that indicates the overall quality of a PSR outcome.

For this experiment, we have used the instances of the SPI set, the only ones with known optimal solutions. The kernelization algorithm was run on all instances of the SPI set. Then, we have computed P, R and F for each of the resulting kernels and each of the reduced instances produced by MineReduce.

In the cases where no vertex has been fixed in the solution (i.e., $TP = 0$ and $FP = 0$), it was assumed $P = 1$. Since in these cases $R = 0$, it follows that $F = 0$. For instances that admit more than one optimal solution, the metrics were computed for each one of them. Then, the one that outputs the higher values was taken into account.

5.3. Results

Tables 5 and 6 show the results obtained for instances of Type I and II, respectively. They present an average value over all ten instances for each combination of n and m. Since MineReduce can produce more than one reduced instance for each original instance (one for each mined pattern), two values of each metric were computed for it. One (labelled "best") takes only the best-score reduction into account, and the other (labelled "avg.") presents the average value of all reductions.

The charts in Fig. 4 and Fig. 5 present the average measures for instances of Type I and II, respectively. They show MineReduce achieves optimal precision in most cases, and its overall average precision is over 0.990, almost the optimum value. Also, regarding the recall, MineReduce performed generally better than the kernelization algorithm. For Type I instances, the kernelization algorithm achieved an overall average of 0.285, whereas MineReduce achieved 0.471 (avg.) and 0.532 (best). For Type II instances, the kernelization algorithm achieved an overall average of 0.377, whereas MineReduce achieved 0.459 (avg.) and 0.517 (best).

Therefore, MineReduce also obtained significantly higher F-scores than the kernelization algorithm. For Type I instances, the kernelization algorithm obtained an overall average of 0.318, whereas MineReduce achieved 0.598 (avg.) and 0.656 (best). For Type II instances, the kernelization algorithm achieved an average of 0.416, whereas MineReduce achieved 0.583 (avg.) and 0.640 (best).

Table 5: PSR assessment – SPI (Type I)

		Kernel.		MR (best)			MR (avg.)		
n	m	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
10	10	0.420	0.480	1.000	0.360	0.459	1.000	0.360	0.459
	20	0.467	0.529	1.000	0.511	0.642	1.000	0.458	0.589
	30	0.721	0.815	1.000	0.559	0.682	1.000	0.559	0.682
	40	1.000	1.000	1.000	0.441	0.526	1.000	0.441	0.526
15	20	0.039	0.069	1.000	0.485	0.581	1.000	0.375	0.484
	40	0.110	0.146	1.000	0.428	0.571	1.000	0.375	0.516
	60	0.364	0.454	1.000	0.384	0.510	1.000	0.354	0.464
	80	0.975	0.986	1.000	0.658	0.778	1.000	0.652	0.772
	100	1.000	1.000	1.000	0.754	0.857	1.000	0.754	0.857
20	20	0.114	0.125	1.000	0.415	0.473	0.980	0.382	0.444
	40	0.020	0.033	1.000	0.502	0.624	1.000	0.469	0.593
	60	0.000	0.000	1.000	0.609	0.748	0.997	0.450	0.602
	80	0.021	0.040	1.000	0.527	0.662	0.994	0.396	0.540
	100	0.049	0.080	1.000	0.510	0.669	0.993	0.445	0.606
	120	0.165	0.250	1.000	0.560	0.700	1.000	0.526	0.672
25	40	0.000	0.000	1.000	0.692	0.807	0.993	0.634	0.752
	80	0.006	0.012	1.000	0.598	0.739	0.984	0.497	0.639
	100	0.000	0.000	1.000	0.575	0.721	0.968	0.399	0.557
	150	0.021	0.038	1.000	0.538	0.693	0.941	0.432	0.587
	200	0.202	0.312	1.000	0.527	0.686	0.974	0.458	0.621

Table 6: PSR assessment – SPI (Type II)

		Kernel.		MR (best)			MR (avg.)		
n	m	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
10	10	0.413	0.474	1.000	0.217	0.298	1.000	0.147	0.217
	20	0.860	0.889	1.000	0.460	0.603	1.000	0.436	0.584
	30	0.890	0.923	1.000	0.464	0.555	1.000	0.461	0.553
	40	0.975	0.986	1.000	0.550	0.613	1.000	0.550	0.613
15	20	0.290	0.310	1.000	0.310	0.417	1.000	0.286	0.391
	40	0.071	0.116	1.000	0.526	0.658	1.000	0.491	0.626
	60	0.680	0.731	1.000	0.673	0.800	1.000	0.601	0.740
	80	0.808	0.876	1.000	0.456	0.596	1.000	0.456	0.596
	100	1.000	1.000	1.000	0.746	0.811	1.000	0.746	0.811
20	20	0.127	0.147	1.000	0.494	0.607	1.000	0.436	0.548
	40	0.154	0.184	1.000	0.557	0.697	0.993	0.400	0.528
	60	0.043	0.074	1.000	0.447	0.577	1.000	0.388	0.516
	80	0.244	0.298	1.000	0.466	0.596	1.000	0.416	0.550
	100	0.263	0.331	1.000	0.587	0.731	1.000	0.495	0.653
	120	0.269	0.372	1.000	0.550	0.697	1.000	0.526	0.675
25	40	0.023	0.040	1.000	0.631	0.761	1.000	0.467	0.598
	80	0.016	0.029	1.000	0.619	0.759	1.000	0.477	0.626
	100	0.020	0.036	1.000	0.502	0.648	1.000	0.436	0.582
	150	0.180	0.225	1.000	0.541	0.690	0.967	0.467	0.616
	200	0.214	0.284	1.000	0.539	0.692	0.981	0.491	0.645

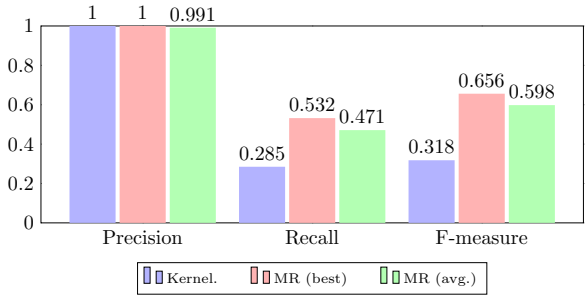


Figure 4: Average PSR assessment measures for Set SPI (Type I)

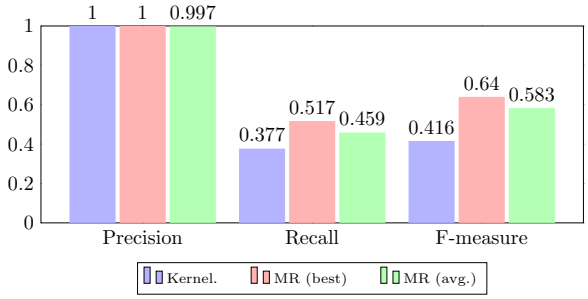


Figure 5: Average PSR assessment measures for Set SPI (Type II)

These results show that, although MineReduce is a heuristic method, it has achieved an average precision almost equal to that of the kernelization algorithm while fixing significantly more elements in the solution (higher recall).

6. Conclusion

Our experimental results, using the MWVCP, reinforce the effectiveness of the MineReduce approach, showing that it obtains better solutions within less computational time. While both MineReduce and the original MS-ITS heuristic presented equivalent performances for the smallest instances, MineReduce was dominantly superior for the largest ones.

For the medium-scale instances set (MPI), MineReduce obtained better results in 41% of the cases and the same results in the other 59%, with shorter computational times in 79% of the cases. For the large-scale instances set (LPI), MineReduce overcame the original MS-ITS heuristic regarding average objective values for all instances, with shorter computational times in 87% of the cases.

The behaviour analysis showed that MineReduce generates solutions of a significantly higher quality and does it significantly faster in comparison to the original MS-ITS heuristic.

Additionally, we have compared the reduced instances produced by MineReduce to the kernels produced by a kernelization algorithm. The results show that MineReduce's PSR is effective, being able to identify more items that should be part of an optimal solution, with minimal precision loss or even no precision loss at all.

In future work, the effectiveness of MineReduce's PSR shall be further assessed, with applications to other problems and comparisons to more sophisticated kernelization algorithms.

Acknowledgements

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil) [grant numbers 303726/2017-2, 310444/2018-7]; Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ, Brazil) [grant number E-26/203.272/2017]; and Instituto Brasileiro de Geografia e Estatística (IBGE, Brazil).

References

[1] S. L. Martins, I. Rosseti, A. Plastino, Data Mining in Stochastic Local Search, Springer International Publishing, Cham, 2018, Ch. 3, pp. 39–87. doi:10.1007/978-3-319-07124-4_11.

[2] A. Plastino, B. Barbalho, L. F. M. Santos, R. Fuchshuber, S. L. Martins, Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic, Journal of Heuristics 20 (1) (2014) 39–74. doi:10.1007/s10732-013-9231-0.

[3] L. F. Santos, S. L. Martins, A. Plastino, Applications of the DM-GRASP heuristic: a survey, International Transactions in Operational Research 15 (4) (2008) 387–416. doi:10.1111/j.1475-3995.2008.00644.x.

[4] M. R. H. Maia, A. Plastino, P. H. V. Penna, MineReduce: An approach based on data mining for problem size reduction, Computers & Operations Research 122 (2020) 104995. doi:10.1016/j.cor.2020.104995.

[5] T. Zhou, Z. Lü, Y. Wang, J. Ding, B. Peng, Multi-start iterated tabu search for the minimum weight vertex cover problem, Journal of Combinatorial Optimization 32 (2) (2016) 368–384. doi:10.1007/s10878-015-9909-3.

[6] R. G. Downey, M. R. Fellows, Some Ad Hoc Methods: The Methods of Bounded Search Tree and Problem Kernel, Springer New York, New York, NY, 1999, Ch. 3, pp. 29–48. doi:10.1007/978-1-4612-0515-9_3.

[7] R. M. Karp, Reducibility among Combinatorial Problems, Springer US, Boston, MA, 1972, pp. 85–103. doi:10.1007/978-1-4684-2001-2_9.

[8] M. R. Fellows, L. Jaffke, A. I. Király, F. A. Rosamond, M. Weller, What Is Known About Vertex Cover Kernelization?, Springer International Publishing, Cham, 2018, pp. 330–356. doi:10.1007/978-3-319-98355-4_19.

[9] R. Niedermeier, P. Rossmanith, On efficient fixed-parameter algorithms for weighted vertex cover, Journal of Algorithms 47 (2) (2003) 63–77. doi:10.1016/S0196-6774(03)00005-1.

[10] C. Bazgan, S. Toubaline, Z. Tuza, The most vital nodes with respect to independent set and vertex cover, Discrete Applied Mathematics 159 (17) (2011) 1933–1946. doi:10.1016/j.dam.2011.06.023.

[11] S. Funke, A. Nusser, S. Storandt, Placement of loading stations for electric vehicles: No detours necessary!, Journal of Artificial Intelligence Research 53 (2015) 633–658. doi:10.1613/jair.4688.

[12] V. V. Gusev, The vertex cover game: Application to transport networks, Omega (2019) 102102, In Press.doi:10.1016/j.omega.2019.08.009.

[13] S. J. Shyu, P.-Y. Yin, B. M. T. Lin, An ant colony optimization algorithm for the minimum weight vertex cover problem, Annals of Operations Research 131 (1) (2004) 283–304. doi:10.1023/B:ANOR.0000039523.95673.33.

[14] R. Jovanovic, M. Tuba, An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem, Applied Soft Computing 11 (8) (2011) 5360–5366. doi:10.1016/j.asoc.2011.05.023.

[15] S. Bouamama, C. Blum, A. Boukerram, A population-based iterated greedy algorithm for the minimum weight vertex cover problem, Applied Soft Computing 12 (6) (2012) 1632–1639. doi:10.1016/j.asoc.2012.02.013.

[16] R. Li, S. Hu, H. Zhang, M. Yin, An efficient local search framework for the minimum weighted vertex cover problem, Information Sciences 372 (2016) 428–445. doi:10.1016/j.ins.2016.08.053.

[17] X. Xie, X. Qin, C. Yu, X. Xu, Test-cost-sensitive rough set based approach for minimum weight vertex cover problem, Applied Soft Computing 64 (2018) 423–435. doi:10.1016/j.asoc.2017.12.023.

[18] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd Edition, Morgan Kaufmann, Boston, 2012.

[19] G. Grahne, J. Zhu, Efficiently using prefix-trees in mining frequent itemsets, in: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2003.

[20] R. M. Aiex, M. G. C. Resende, C. C. Ribeiro, TTT plots: a perl program to create time-to-target plots, Optimization Letters 1 (4) (2007) 355–366. doi:10.1007/s11590-006-0031-4.

[21] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Kernelization, Springer International Publishing, Cham, 2015, Ch. 2, pp. 17–49. doi:10.1007/978-3-319-21275-3_2.

[22] A. Tharwat, Classification assessment methods, Applied Computing and Informatics. (2018) In Press.doi:10.1016/j.aci.2018.08.003.

APPENDIX C – MAIA, M. R. H.; REULA, M.; PARREÑO-TORRES, C.; VUPPULURI, P. P.; PLASTINO, A.; SOUZA, U. S.; CESCHIA, S.; PAVONE, M.; SCHAERF, A. “Metaheuristic Techniques for the Capacitated Facility Location Problem with Customer Incompatibilities”. Submitted to Soft Computing

Springer Nature 2021 LaTeX template

Metaheuristic Techniques for the Capacitated Facility Location Problem with Customer Incompatibilities

Marcelo R. H. Maia[1,2*], Miguel Reula[7], Consuelo Parreño-Torres[3], Prem Prakash Vuppuluri[4], Alexandre Plastino[1], Uéverton S. Souza[1], Sara Ceschia[5], Mario Pavone[6] and Andrea Schaerf[5]

[1*]Institute of Computing, Fluminense Federal University, Av. Gen. M. Souza, Niterói, 24210–346, RJ, Brazil.
[2]ENCE, Brazilian Institute of Geography and Statistics, R. Gen. Canabarro 706, Rio de Janeiro, 20271–020, RJ, Brazil.
[3]Departamento de Estadística e Investigación Operativa, Universidad de Valencia, C/ Doctor Moliner, 50, Burjassot, 46100, Valencia, Spain.
[4]Department of Electrical Engineering, Faculty of Engineering, Dayalbagh Educational Institute (Deemed to be University), Dayalbagh, Agra, 282005, Uttar Pradesh, India.
[5]DPIA, University of Udine, Via delle Scienze 206, Udine, 33100, Italy.
[6]DMI, University of Catania, v.le A. Doria 6, Catania, 95125, Italy.
[7]Departamento de Estadística, Universidad Carlos III de Madrid, C/ Madrid, 126, Getafe, 28903, Madrid, Spain.

*Corresponding author(s). E-mail(s): mmaia@ic.uff.br;
Contributing authors: miguel.reula@uc3m.es; consuelo.parreno@uv.es; vpremprakash@dei.ac.in; plastino@ic.uff.br; ueverton@ic.uff.br; sara.ceschia@uniud.it; mario.pavone@unict.it; andrea.schaerf@uniud.it;

Springer Nature 2021 LaTeX template

Abstract

We study a novel version of the Capacitated Facility Location Problem, which includes incompatibilities among customers. For this problem, we propose and compare on a fair common ground a portfolio of metaheuristic techniques developed independently from each other. We tested our techniques on a new dataset composed of instances of increasing size, varying from medium to very large ones. The outcome is that the technique based on data mining has been able to outperform the others on most instances, except for the extreme cases (smallest and largest instances), for which it is often overcome by other simpler ones. In order to encourage future comparisons on this problem, we make instances and the solution validator available to the community.

Keywords: Facility Location, Metaheuristics, Multi-sourcing

1 Introduction

We study the classical *Capacitated Facility Location Problem* (CFLP) in which facilities must be selected for opening and customers must be supplied by open facilities. Constraints concern customers whose demand must be completely satisfied and the capacity of facilities that cannot be exceeded. We consider the *multi-source* version of the problem, called MS-CFLP, in which each customer can be supplied by more than one facility. The objective function to minimize is the sum of opening and shipping costs, the latter being determined using an input matrix of per-unit costs.

Following the classification of facility location models proposed by Klose and Drexl (2005), our problem has the following features: network location model, minsum objective, single-stage, capacity constraints, multi-sourcing, single-product, static and deterministic.

We add the novel constraint that some pairs of customers cannot be supplied by the same facility. This constraint models the situation in which customers do not accept that their suppliers also refurnish their competitors. We name the resulting problem as MS-CFLP-CI (for Customer Incompatibilities).

For this new problem, we propose a portfolio of metaheuristic techniques, each one developed independently by a different research group. The idea originates from the Metaheuristics Summer School (MESS 2020+1) as a competition of techniques among its participants.

In detail, we propose a MineReduce Multi-Start Iterated Local Search, a GRASP approach, a Permutation Coded Evolutionary Algorithm, and a Multistart Greedy technique.

The experimental analysis is performed on the same computing system in order to provide a fair ground for comparison. To this end, the same running time is granted to all techniques, and it is related to the size of the instance solved.

All techniques have been tuned on a set of training instances and run on the (unseen) validation instances. Instances have a size ranging from 50 to 3000 facilities and have been created by an ad hoc generator.

The outcome is that the hybrid approach based on data mining outperforms all the others for most instances. However, there is a different behavior for the instances on the two extremes of the size range. Specifically, for the smallest instances, the GRASP approach turns out to be the best, whereas, for very large ones, the simple greedy technique is able to obtain results that slightly overtake the mining approach.

In order to foster future comparisons, all the data is made available on the web at the MESS 2020+1 website https://www.ants-lab.it/mess2020/#competition. It also includes the solution validator, which provides against possible misunderstanding in the formulation[1].

The paper is organized as follows. Section 2 provides the definition of the problem. Section 3 illustrates the related work. The proposed solution techniques are discussed in Section 4. Experimental results are shown in Section 5. Finally, conclusions are drawn in Section 6.

2 Problem Definition

Let $\mathcal{J} = \{1, \ldots, J\}$ be a set of *facilities*, such that each facility $j \in \mathcal{J}$ has a *capacity* s_j and an *opening cost* f_j, and $\mathcal{I} = \{1, \ldots, I\}$ a set of *customers*, such that each customer i has a *demand* of quantity of goods d_i to be completely satisfied by one or more facilities. We also have a *shipping cost* c_{ij} that is the cost *per unit* of transporting goods from facility j to customer i. Finally, we are given a set Γ of pairs of *incompatible customers*, such that for each $\langle i_1, i_2 \rangle \in \Gamma$, i_1 and i_2 cannot be served by the same facility.

The problem consists in selecting values for the following decision variables: (i) non-negative integer variables x_{ij} representing the quantity of goods moved from facility $j \in \mathcal{J}$ to customer $i \in \mathcal{I}$; (ii) binary variables y_j, such that a facility j is open if $y_j = 1$, closed otherwise.

The constraints are the following:

- The total quantity of goods taken from an open facility cannot exceed its capacity:
$$\sum_{i \in \mathcal{I}} x_{ij} \leq s_j y_j \quad \forall j \in \mathcal{J} \tag{1}$$
- The total quantity of goods brought to a customer must be exactly equal to its demand:
$$\sum_{j \in \mathcal{J}} x_{ij} = d_i \quad \forall i \in \mathcal{I} \tag{2}$$
- Two incompatible customers cannot be supplied by the same facility:
$$x_{i_1 j} = 0 \vee x_{i_2 j} = 0 \quad \forall \langle i_1, i_2 \rangle \in \Gamma, \forall j \in \mathcal{J} \tag{3}$$

[1] The source code of the different solution techniques will be made available too.

- Domain of decision variables:

$$0 \leq x_{ij} \leq d_i \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \tag{4}$$

$$y_j \in \{0,1\} \quad \forall j \in \mathcal{J} \tag{5}$$

Note that Constraints (1) ensure that if facility j is not open, no demand for any customer can be filled from it. The model above is not linear as it contains disjunctions in Constraints (3), but it could be easily linearized using the classical big M technique.

The objective function is the sum of two components: the cost of opening the selected facilities and the cost to ship the goods from the facilities to the customers.

$$min\ z = \sum_{j \in \mathcal{J}, i \in \mathcal{I}} c_{ij} x_{ij} + \sum_{j \in \mathcal{J}} f_j y_j \tag{6}$$

We assume that all input values are non-negative integers and the total capacity of facilities is enough for the total demand, i.e. $\sum_{j \in \mathcal{J}} s_j \geq \sum_{i \in \mathcal{I}} d_i$.

Instances are written in the MiniZinc data format (Nethercote et al, 2007). An example of an input file is shown in Figure 1.

```
Facilities = 4;
Customers = 10;

Capacity = [100, 40, 60, 60];
FixedCost = [860, 350, 440, 580];
Demand = [12, 17, 5, 13, 20, 20, 17, 19, 11, 20];
ShippingCost = [|27, 66, 44, 55
                |53, 89, 68, 46
                |17, 40, 18, 61
                |20, 68, 44, 78
                |42, 89, 65, 78
                |57, 55, 49, 31
                |89, 101, 90, 16
                |37, 31, 23, 55
                |76, 60, 63, 44
                |82, 107, 91, 31|];

Incompatibilities = 3;
IncompatiblePairs = [| 1, 10 | 2, 7 | 8, 9 |];
```

Fig. 1 An example of an input file in MiniZinc format.

The uncapacitated FLP (without customer incompatibilities), where $s_j = +\infty, \forall j \in \mathcal{J}$, was proven to be $\mathcal{NP}$-hard by Cornuéjols et al (1983) through a reduction from the node cover problem; hence also the MS-CFLP-CI is $\mathcal{NP}$-hard.

3 Related Work

Location problems have been widely studied in the literature under the names of plant, warehouse, or facility location problems. The literature on this research area is vast. Thus, we decided to focus our review on contributions related to the specific variant of the problem MS-CFLP, i.e. the deterministic multi-source capacitated one, where demands are deterministic, each customer can be served by multiple facilities and each facility has a maximum capacity. A more comprehensive overview of models and algorithms for facility location problems can be found in the surveys by Klose and Drexl (2005), Melo et al (2009) and Fernández and Landete (2015).

Many solution approaches are ad hoc algorithms that exploit the mathematical structure of the models. In particular, several methods use some form of Lagrangean relaxation (Barahona and Chudak, 2005; Avella et al, 2009; Görtz and Klose, 2012; Caserta and Voß, 2020).

To the best of our knowledge, state-of-the-art algorithms for solving the MS-CFLP are the exact ones proposed by Fischetti et al (2016) based on Benders decomposition and the Lagrangian-based branch-and-bound by Görtz and Klose (2012). Among heuristic methods, the best performance is obtained by Guastaroba and Speranza (2012) who implemented the MIP-based heuristic called *Kernel Search* and Caserta and Voß (2020) who presented a matheuristic based on the *Corridor Method*. These methods are tested on well-known datasets available from the OR-Library and proposed by Avella and Boccia (2009) and Avella et al (2009).

Guastaroba and Speranza (2012) solved to optimality instances up to 1000 facilities and 1000 customers within a timeout of one hour, and they improved upon previous work most of the best known values for large instances (up to 2000 facilities and 4400 customers). For those instances (Avella et al, 2009), denoted as *Test Bed A*, *B* and *C*, Fischetti et al (2016) were able to prove optimality for 210 out of 445 cases in 50,000 seconds, while Caserta and Voß (2020) matched the optimal solutions for 79 of 295 instances (*Test Bed C* was not tested) with an average running time of less than 600 seconds. Among exact methods, the B&B of Görtz and Klose (2012) remains competitive in terms of running times with Fischetti et al (2016) for medium size instances.

Recently, Avella et al (2021) devised a new class of valid inequalities that, embedded into a cut-and-branch procedure, improved many upper and lower bounds for 2000×2000 instances of *Test Bed C* with a time limit of one hour.

4 Solution Techniques

This section presents the solution techniques autonomously developed by the different research groups in the context of the Metaheuristics Summer School (MESS 2020+1). In detail, MineReduce-based Multi-Start ILS was implemented by Marcelo Maia, under the supervision of Alexandre Plastino and Uéverton Souza; GRASP by Miguel Reula and Consuelo Parreño-Torres; Permutation Coded Evolutionary Algorithm by Prem Prakash Vuppuluri, and

finally Multistart Greedy Algorithm by Sara Ceschia, Mario Pavone and Andrea Schaerf (competition organizers).

4.1 MineReduce-based Multi-Start ILS

MineReduce is an approach based on data mining for problem size reduction that has obtained promising results (Maia et al, 2020). This section describes an application of the MineReduce approach for the MS-CFLP-CI that uses a multi-start iterated local search (Lourenço et al, 2003) as base metaheuristic.

4.1.1 MineReduce elements

The MineReduce approach (Maia et al, 2020) applies patterns extracted from good solutions (using a data mining technique) to perform problem size reduction – a process in which a problem instance is reduced to a smaller-size version, the reduced instance is solved, and the solution found is expanded, so that it becomes a solution for the original instance.

The elite set E keeps the η best solutions among those obtained by the end of the iterations in the multi-start structure. When E is considered *stable*, a data mining method is used to extract patterns from its solutions. This happens when one of the following criteria is met: (i) E has not been mined yet, and its contents have not changed for a number of consecutive multi-start iterations greater than $\alpha \times I_{max}$, where I_{max} is the maximum number of iterations, dynamically estimated based on the elapsed time and the number of completed iterations; (ii) E has not been mined yet, $|E| = \eta$ and the elapsed time is greater than half the time limit; or (iii) the contents of E have changed after it was last mined, but they have not changed for a number of consecutive iterations greater than $\alpha \times I_{max}$.

The data mining method returns the β largest frequent itemsets with minimum support γ found in E. They are frequent sets of pairs $\langle i, j\rangle$, i.e., sets of customer-facility assignments with a relative frequency greater than or equal to γ in the best solutions. After these frequent itemsets are mined, their assignments are filled with the corresponding minimum quantities in E, i.e., for each pair $\langle i, j\rangle$ in a frequent itemset, a supplied quantity q is set, which corresponds to the minimum positive supplied quantity for that pair among all solutions in E. Therefore, each final pattern is a set of assignments $\langle i, j, q\rangle$.

Patterns are used to reduce the size of the problem instance based on the assumption that their elements shall be part of the solution. In the particular case of reducing this problem, an extension to the problem formulation is needed. We introduce incompatibilities between customers and facilities, represented by a set Γ', such that for each $\langle i, j\rangle \in \Gamma'$, i cannot be supplied by j. Hence, an additional set of constraints is introduced:

- A customer cannot be supplied by an incompatible facility:

$$x_{ij} = 0 \quad \forall \langle i, j\rangle \in \Gamma' \tag{7}$$

The reduction process works as follows. Given a problem instance $\mathcal{P}$ and a pattern p, the reduced instance $\mathcal{P}'$ is initialized as a copy of $\mathcal{P}$. Then, for each assignment $\langle i, j, q \rangle$ in p, the following changes are applied to $\mathcal{P}'$: (i) Decrease both s_j and d_i by q; (ii) Set $f_j = 0$; and (iii) For each customer i' incompatible with i (from Γ), add a pair $\langle i', j \rangle$ to Γ'.

As a consequence, the domain of each x_{ij} variable will be reduced, and Constraints (7) will fix the values of many other decision variables, effectively reducing the problem instance size.

Finally, a solution for $\mathcal{P}'$ can be expanded into the equivalent solution for $\mathcal{P}$ through the inclusion of all assignments from p.

4.1.2 Algorithmic description

The structure of the MineReduce-based Multi-Start ILS (MR-MS-ILS) is depicted in Algorithm 1, where σ^* is the best solution found. At each iteration of the multi-start structure, an initial solution is generated (line 4) and improved by the ILS (line 5). E is updated by the end of each iteration (line 6), as well as σ^* in case of improvement (line 7). Every time one of the stabilization criteria is met, P is filled with patterns mined from E (line 3).

Algorithm 1 MR-MS-ILS

1: $z(\sigma^*) \leftarrow \infty$; $E \leftarrow \varnothing$; $P \leftarrow \varnothing$
2: **while** time limit not exceeded **do**
3: **if** Stable(E, α) **then** $P \leftarrow$ Mine(E, β, γ)
4: $\sigma_0 \leftarrow$ InitialSolution(P, ε)
5: $\sigma \leftarrow$ ILS(σ_0, δ, κ)
6: UpdateElite(E, σ, η)
7: **if** $z(\sigma) < z(\sigma^*)$ **then** $\sigma^* \leftarrow \sigma$
8: **end while**
9: **return** σ^*

4.1.3 Initial solution

The default initial solution generation (when $P = \varnothing$) works as follows. First, it opens facilities until the total capacity of all open facilities is greater than or equal to the total demand of all customers in $\mathcal{I}$. Then, for each open facility j, a customer i is drawn, and an assignment $\langle i, j, q \rangle$ is added to the solution.

Once all open facilities have been initialized with an assignment, the procedure iterates upon all customers and, for each customer i, adds to the solution the cheapest feasible assignments considering the open facilities only, until the demand of i is fully met or there is no feasible assignment possible with the currently open facilities. When the latter situation happens, the procedure opens another facility and resume the assignments.

The quantity of goods q chosen for an assignment $\langle i, j, q\rangle$ is always the maximum possible one, which corresponds to the minimum between the residual demand of i and the residual capacity of j.

Two alternative strategies for choosing the next facility to open are considered. The *greedy opening* (GO) strategy chooses the facility j with the lowest ratio f_j/s_j among all closed facilities. The *random opening* (RO) strategy draws a facility from the set of closed facilities using the roulette wheel method, with a selection probability for each facility j proportional to its ratio s_j/f_j. The strategy to be used is defined by the parameter ε.

When $P \neq \varnothing$, it means that patterns have already been mined from E. In this case, the MineReduce approach generates an initial solution through a reduce-optimize-expand process. Firstly, the problem instance is reduced based on one of the patterns in P. Then a solution for the reduced instance is obtained by applying the default initial solution generation and the ILS. Finally, the solution is expanded into its equivalent for the original instance.

4.1.4 Iterated Local Search

The strategies applied in this ILS were inspired by an iterated tabu search proposed for the single-source CFLP by Ho (2015). Solution $\hat{\sigma}$ is initialized with the result of a local search on the initial solution σ_0. At each iteration, $\hat{\sigma}$ is perturbed resulting in solution σ', which is then improved by a local search to obtain solution $\bar{\sigma}$. If solution $\bar{\sigma}$ satisfies the acceptance criterion $z(\bar{\sigma}) < (1 + \delta) \times z(\bar{\sigma}^*)$, the search continues from solution $\bar{\sigma}$ (i.e., $\hat{\sigma} \leftarrow \bar{\sigma}$).

Two neighbourhood structures for a solution σ are used. Neighbourhood $\mathcal{N}_1(\sigma)$ consists of all feasible solutions that can be obtained from σ by reassigning goods supplied to a customer i from a facility j_1 to another facility j_2. This type of move reassigns the maximum possible quantity of goods, which is the minimum between x_{ij_1} and the residual capacity of j_2. Neighbourhood $\mathcal{N}_2(\sigma)$ consists of all feasible solutions that can be obtained from σ by exchanging a customer i_1 from a facility j_1 with another customer i_2 from another facility j_2. This type of move makes a full reassignment of the supplied goods, i.e., all goods supplied to i_1 by j_1 are reassigned to j_2 and vice-versa.

A multi improvement (MI) strategy (Rios et al, 2016; Silva et al, 2022), which applies a sequence of multiple independent moves at each step, is used for neighbourhood exploration. Two moves are independent if the feasibility and cost improvement of any of these moves are not affected by the application of the other to the same solution (i.e., if they have no facilities in common).

Let σ be the seed solution from which the local search starts. First, all possible moves from σ to solutions in $\mathcal{N}_1(\sigma) \cup \mathcal{N}_2(\sigma)$ are evaluated, and a priority queue of improving moves M is built in decreasing order of cost improvement. Then, a series of steps are performed until M is empty or the time limit is exceeded. The following actions are taken at each step: (i) every move in M that is independent of all its predecessors is applied, and (ii) M is updated.

At each iteration of the ILS, one of the following perturbation operators is randomly selected and applied to solution $\hat{\sigma}$:

1. *Close one facility.* Randomly choose a facility j that supplies only one customer i and close it. Then reassign the goods supplied to i by j to the remaining open facilities by making the cheapest feasible assignments.
2. *Open one facility.* Randomly choose a closed facility and open it.
3. *Close one facility and open one facility.* Randomly choose an open facility j_1 and a closed facility j_2 such that $s_{j_2} \geq \sum_{i \in \mathcal{I}} x_{ij_1}$. Close j_1 and open j_2, then reassign all supplies from j_1 to j_2.
4. *Close one facility and open two facilities.* Select an open facility j_1 and two closed facilities j_2 and j_3 such that $s_{j_2} + s_{j_3} \geq \sum_{i \in \mathcal{I}} x_{ij_1}$ and the opening cost improvement is maximum. Close j_1 and open j_2 and j_3, then reassign all supplies from j_1 to j_2 and j_3 by making the cheapest feasible assignments.
5. *Open one facility and close two facilities.* Select a closed facility j_1 and two open facilities j_2 and j_3 such that: $s_{j_1} \geq \sum_{i \in \mathcal{I}} (x_{ij_2} + x_{ij_3})$, there are no incompatibilities between customers supplied by j_2 and customers supplied by j_3, and the opening cost improvement is maximum. Open j_1 and close j_2 and j_3, then reassign all supplies from j_2 and j_3 to j_1.

For the perturbation mechanism to be effective, the local search will not consider any move that would reverse a facility opening or closure applied by the perturbation operator at iteration v of the ILS.

4.1.5 Parameters tuning

The irace package (López-Ibáñez et al, 2016) was used to tune the parameters independently on five subsets of the training instances, defined by size ranges. Table 1 presents the best configurations found.

Table 1 Best parameter configurations for MR-MS-ILS

Range	α	β	γ	δ	ε	η	κ
$J \leq 150$	0.07	10	0.4	0.01	RO	5	100
$150 < J \leq 600$	0.03	6	0.9	0.01	GO	10	200
$600 < J \leq 1400$	0.04	6	0.8	0.05	GO	5	100
$1400 < J \leq 2000$	0.03	6	0.8	0.05	GO	5	100
$J > 2000$	0.04	1	1.0	0.02	GO	5	200

4.2 GRASP

In this section, we describe a greedy randomized adaptive search procedure (GRASP) to solve the MS-CFLP-CI. This method has been applied in a large number of problems and has behaved like a very robust metaheuristic procedure (Festa and Resende, 2018). Algorithm 2 shows the pseudocode of the algorithm proposed, which consists of a constructive phase, Section 4.2.1, and of an improvement phase, Section 4.2.2. The construction phase generates a set of β feasible solutions and, in line 4, returns the solution with the lowest objective function $z(\sigma_0)$. If the current solution σ_0 is a candidate for improvement,

the improvement phase is carried out in the framework of the Variable Neighborhood Descent (VND) algorithm in line 5. It is a candidate to be improved if it is the first iteration of GRASP or if by improving σ_0 by 5% more than the best improvement so far b, the overall GRASP solution σ^* could be upgraded, $z(\sigma_0) - 1.05b < z(\sigma^*)$. The 5%, as well as other parameters used throughout the algorithm, have been adjusted by performing an extensive computational analysis comparing different parameter values.

Algorithm 2 Greedy Randomized Adaptive Search Procedure

```
1:  function GRASP(β)
2:      z(σ*) ← ∞;
3:      while time limit not exceeded do
4:          σ0 ← ConstructionPhase (β)
5:          if Candidate (σ0 ) then σ ← VND(σ0)
6:          end if
7:          if z(σ) < z(σ*) then σ* ← σ
8:          end if
9:      end while
10:     return σ*
11: end function
```

4.2.1 Construction phase

The construction phase runs β times a randomized heuristic algorithm in order to generate β feasible solutions. Only the best of those β feasible solutions moves on to the improvement phase.

The randomized heuristic algorithm consists of a combination of greedy and random routines. The first and third stages are randomized, while the second one is completely deterministic. The following vectors will be used to describe it.

- For each facility $j \in \mathcal{J}$ it is calculated the fixed opening cost per unit, i.e., f_j/s_j and we generate the vector o^f, sorting it by increasing order of these quotients, and in the event of a draw, by increasing capacity of the facility j.
- For each customer $i \in \mathcal{I}$, we generate a vector $best^f[i]$ with the five facilities from which it is cheapest to serve it, sorting them by increasing shipping cost, and in the event of a draw, by increasing opening cost of the facility j.
- For each facility $j \in \mathcal{J}$ which is the most suitable for some customer, the vector $best^c[j]$ contains all the customers for which facility j is the most suitable. These facilities are ordered by non-increasing difference between the customer's shipping cost from its second-best facility and the customer's shipping cost from facility j; in case of a tie, by non-increasing difference between the customer's shipping cost from its third-best facility and the

customer's shipping cost from facility j; and, in case of another tie, by non-increasing difference between the customer's shipping cost from its fourth-best facility and the customer's shipping cost from facility j. If there is still a tie, it is ordered by non-increasing demand of the corresponding customer.

- We use the vector $rand^c$ to refer to a vector composed of all randomly ordered customers. It is randomized before each algorithm's stage.

The three stages of the randomized heuristic algorithm are listed below.

Stage 1. Initialization. We go through vector $rand^c$. Given a customer i, we try to satisfy its demand by using the most suitable facilities for it, those of $best^f[i]$. We select an unopened facility in $best^f[i]$ (if they are all open, we move on to the next customer) and allocate the unassigned demand of customer i. Let this facility be j, we go through vector $best^c[j]$, assigning if possible the demand of its customers to the facility j. Once the facility has covered $\delta\%$ of its capacity, the procedure is finished, and this allocation becomes part of the partial solution. Otherwise, if all vector $best^c[j]$ has been traversed and this amount has not been covered, facility j is closed and, therefore, the customers we would have allocated are de-allocated.

Stage 2. Ordered facilities. We go through vector o^f. Given a facility j, satisfying $j \in \bigcup_{i \in \mathcal{I}} best^f[i][1]$. We select the first unassigned customer in $best^c[j]$, i. Next, we assign the demand of customer i to the facilities in the vector $best^f[i]$ that are already open, have availability and are not assigned to incompatible customers. If not all of the demand of customer i has been allocated and facility j is unopened, we open it and allocate the demand of the customer. When facility j is no longer available, the stage moves on to another facility.

Stage 3. Remaining allocation. Finally, a third stage is applied in order to achieve feasible solutions, so we have to cover the unassigned demand of all the customers. We go through vector $rand^c$. Given a customer with unassigned demand, customer i, we go through already open facilities where it can be assigned (there is availability and there are no customers incompatible with i assigned) by non-increasing shipping cost, assigning the goods of the customer until the customer is completely served. If the process finishes and there is still unassigned demand of customer i, we go through vector $best^f[i]$, opening facilities not already opened and assigning the goods of the customer to the facility until the customer is completely served. If the process finishes and there is still unassigned demand of customer i, the unopened facilities are randomly cycled through, opening the facilities and assigning the goods of the customer to them until the customer is completely served.

4.2.2 Variable Neighborhood Descent

In this algorithm, we have implemented a sequential VND in which the search is performed by exploring neighborhood structures of a current solution one after another. The algorithm requires as input an initial solution σ_0 and an ordered list N of neighborhood structures that must be examined. Once an improving solution is detected in some neighborhood structure, it replaces the current solution and the search re-starts from it by exploring the first neighborhood structure in the list LS. Otherwise, if there are no improving solutions in the neighborhood LS_k, the search is continued by exploring N_{k+1} and so on up to the $LS_{k_{max}}$.

We have designed and implemented three local search algorithms describing three neighbourhood structures ($k_{max} = 3$). In all of them, the first-improvement search strategy is used, replacing the current solution and re-starting the search from it each time a better solution is obtained. Each local search algorithm is run iteratively until a total of $m_N = 20$ iterations without improvement are reached. They all are based on the well-known destruction-and-repair method and share the repair phase but differ in the destruction phase. All details are described below:

Destruction phase. This phase destroys a part of the current solution. The selection of the elements to be removed is completely randomized. In LS_1, we choose between 1% and 5% of the open facilities in the current solution. Then, the corresponding demand units of those customers that were served by these facilities are left unassigned. In LS_2, between 1% and 5% of the opened facilities are also randomly selected but, instead of closing them, we unassign up to 50% of the customers demand that were served from this facility. Finally, in LS_3, we select between 1% and 5% of the customers and completely unassign them.

Repair phase. It goes through the unassigned customers. Given a customer i, the vector of facilities to which it can be assigned (without incompatible customers and with sufficient availability) is split into two sets according to the following criteria:

Set 1 - Consists of already opened facilities and of closed facilities whose capacity is greater than or equal to the unassigned demand of the customer i.
Set 2 - Facilities that do not meet the conditions to be in set 1.

Within both sets, facilities are ordered by increasing unit cost. Calculating the unit cost as the shipping cost plus, if the facility is still closed, the quotient between its fixed opening cost and the maximum demand of customer i that could be allocated by opening the corresponding facility. The repair phase goes through both sets, first through set 1 and then through set 2, assigning the goods of the customer to the facilities until the customer is completely served.

4.3 Permutation Coded Evolutionary Algorithm (PcEA)

This section describes an evolutionary algorithm for the problem that encodes solutions as concatenated permutations of customers and facilities. Two different sets of variation operators are probabilistically applied separately to the customer and facility genes to evolve solutions in a 1-elitist, steady-state evolutionary framework. If there are no improvements for a certain number of (empirically determined) iterations, the population is reinitialized to a set of random solutions seeded with the globally best solution obtained thus far. The key elements of the proposed Permutation-coded Evolutionary Algorithm (PcEA) are discussed further as follows.

4.3.1 Solution Representation

Chromosomes are encoded as concatenated permutations of customers and facilities. A permutation of customers indicates the sequence in which customer demands are to be satisfied, and the associated facility permutation represents the order in which facilities are to be opened for servicing customer demands. A chromosome is thus comprised of $|customers|+|facilities|$ genes.

4.3.2 Objective Function Decoder

Each chromosome is decoded as follows: Facilities are opened in permutation order until the total capacity of the opened facilities matches the combined customer demand. Thereafter, in the sequence defined by the customers' permutation, each customer is allocated goods from the open facility having the lowest allocation cost possible while ensuring that there are no conflicts with customers that have already been allocated goods from that facility. If no open facility satisfies this no-conflict requirement, then the next facility in the facility permutation is opened, and the corresponding customer demand is satisfied. If all facilities have already been opened, then the facility with the lowest increase in cost is chosen for allocation.

4.3.3 Evolutionary Framework: Operators and Policies

A steady-state, 1-elitist evolutionary algorithm for the problem uses two different recombination operators, one for the customer permutations and the other for facility permutations. The population is initialized to a fixed set of chromosomes, each comprised of a random permutation of customers, concatenated with a random permutation of facilities. If there is no improvement in the objective function value for a preset number of iterations, the population is re-initialized to a set of randomly generated chromosomes and seeded with the global best solution obtained thus far. Parents are selected using tournaments of size three. If two solutions have the same objective function value, then the solution with a lesser number of violations is preferred. For evolving better customer permutations, Davis' order crossover (OX) operator (Davis, 1991) is applied. The OX crossover operates in linear time and has the added

benefit of preserving the relative order of the customers inherited from good parents. The OX crossover takes two parental customer permutations, i_1 and i_2, with corresponding facility permutations j_1 and j_2, and produces two offspring customer permutations. The offspring that results in the best total cost when combined with either j_1 or j_2 replaces the worst solution in the population if it is better (has lower cost). Facilities listed earlier in the facility permutation have a greater chance of being included in the set of opened facilities as compared to those that occur further down in the permutation. This implies that such facilities would appear early in parental facility permutations. Good offspring should include facilities that were likely used in their parents. A suitable crossover operator that enables offspring to inherit such "good" parental facilities is the Alternating-position (APX) crossover (Larrañaga et al, 1997), which builds the offspring by first selecting alternate genes from each parent and then removing duplicates by deleting each latter duplicate entry in the list. For example, consider parent permutations $p_1 = (1, 0, 3, 2, 4)$ and $p_2 = (3, 1, 4, 0, 2)$. Starting with the first location in p_1, we generate the string $(1, 3, 0, 1, 3, 4, 2, 0, 4, 2)$. Retaining only the first occurrence of each number, we get the offspring $(1, 3, 0, 4, 2)$. Mutation is performed using a simple, constant-time swap mutation operator.

4.3.4 Parameter Selection

Several calibration trials were performed using a fractional factorial design approach for setting the parameter values. In the course of the trials, the probabilities of crossover and mutation were set to 0.6 and 0.4, respectively. The OX and APX crossover probabilities were set to 0.5 each. The population was comprised of 40 chromosomes and was re-initialized in the absence of any improvement in the fitness of the best solution of the population over 10,000 iterations.

4.4 Multistart Greedy Algorithm

Our last technique is a multistart greedy (MG) algorithm. The greedy procedure selects at each step a triple $\langle i, j, q \rangle$, where i is a customer, j is a facility, and q is the quantity of goods to be moved from j to i. The procedure stops when all customers are fully supplied.

The procedure for selecting the triple iterates upon all pairs $\langle i, j \rangle$ searching for the "best" feasible one. The quantity q to be moved from j to i is the maximum possible one, which corresponds to the minimum between the residual amount of i and the residual capacity of j at that step. An assignment is considered feasible if $q > 0$ and it does not violate customer incompatibilities with previous assignments.

The best pair is the one that corresponds to the minimum supply cost (per unit), but taking into account also the opening cost in the case that the assignment is the first one to the facility j. The opening cost is not accounted for in full, as this choice would penalize too much the assignments to currently

closed facilities (given that a certain number of facilities has to be opened anyway). That is, we count the opening cost multiplied by the fraction of s_j moved to i, in turn, multiplied by a factor α, which is a parameter of the method. For example, if $f_j = 400$, $s_j = 50$, and $x_{ij} = 10$, than the share of the opening cost considered in the computation of the best pair is $400 \times (10/50) \cdot \alpha = 80\alpha$.

The algorithm also implements a *random tie-break* strategy, which selects in a uniform random way in case of equal cost. Given that an exactly equal cost is rather unlikely, we consider as equal all choices within a given threshold β above the current best pair $\langle i, j \rangle$. The threshold β is the second and last parameter of the method, which is tuned on the training instances.

Given that the whole procedure runs faster than the granted time, it is repeated several times as long as the timeout is not exceeded. The lower score obtained upon all runs is returned.

The tuning procedure for parameters α and β was performed using the tool JSON2RUN (Urli, 2013), which uses an F-Race procedure (Birattari et al, 2010) for selecting the best configuration. The winning configuration turned out to be $\alpha = 0.25$ and $\beta = 0.288$.

5 Experimental Results

We now describe the experimental results. We first introduce the datasets used and the other computational settings (Section 5.1), and then we present and discuss the results (Section 5.2).

5.1 Settings

The search methods have been tested on artificial instances created by a generator specifically designed for this purpose. Instances have been partitioned into two datasets: the training instances used for tuning the parameters and the validation instances used for the comparison. Instances are available at the MESS 2020+1 website https://www.ants-lab.it/mess2020/#competition, along with the solution validator used to check the correctness of the results.

The features of the instances are shown in Tables 2 and 3, where J and I are the numbers of facilities and customers, SI is the number of incompatibilities, AOC is the average opening cost, ASC is the average supply cost, and DR is the ratio between the total demand and the total capacity.

All the experiments were run on an AMD Ryzen Threadripper PRO 3975WX 32-Cores (3.50 GHz) with Ubuntu Linux 20.4. One single core was dedicated to each experiment.

We make comparisons based on two different timeouts, both based on the number of facilities J. The first is the one proposed for the MESS 2020+1 competition, which grants $10\sqrt{J}$ seconds for solving each instance, whereas the second one runs longer and grants exactly J seconds per instance.

Instance	J	I	SI	AOC	ASC	DR
wlp01	50	115	383	648.00	52.4447	0.451
wlp02	100	253	1718	678.50	53.4662	0.484
wlp03	150	345	3447	671.73	52.2599	0.440
wlp04	200	479	6292	674.80	53.5689	0.446
wlp05	250	601	9750	664.60	53.7353	0.472
wlp06	300	705	13701	640.50	52.6482	0.452
wlp07	400	1012	27635	649.85	53.0710	0.499
wlp08	500	1277	43632	629.38	52.7900	0.489
wlp09	600	1483	59477	646.70	53.0887	0.480
wlp10	700	1733	83291	647.23	52.7992	0.475
wlp11	800	2020	109668	640.52	52.8867	0.491
wlp12	900	2159	126847	653.07	52.9303	0.453
wlp13	1000	2305	142457	633.63	52.9190	0.446
wlp14	1200	2927	232119	650.87	52.7535	0.471
wlp15	1400	3445	320634	645.94	52.8993	0.474
wlp16	1600	4067	450067	640.66	52.7801	0.490
wlp17	1800	4373	520406	649.83	52.9662	0.464
wlp18	2000	4908	657679	647.71	52.3566	0.473
wlp19	2500	5882	927868	650.94	52.8090	0.453
wlp20	3000	7800	1653786	655.39	52.6730	0.496

Table 2 Features of the training instances

Instance	J	I	SI	AOC	ASC	DR
wlp21	75	172	879	618.27	51.1787	0.471
wlp22	175	428	4744	642.17	53.4794	0.461
wlp23	275	694	13197	625.78	52.8361	0.516
wlp24	450	1128	34546	667.69	52.4473	0.464
wlp25	650	1619	71398	637.80	52.3215	0.482
wlp26	850	2007	109325	656.33	52.5074	0.461
wlp27	1100	2847	224656	652.87	52.0253	0.495
wlp28	1500	3474	323388	647.10	52.7387	0.445
wlp29	1900	4522	556567	649.75	52.4321	0.461
wlp30	2750	6965	1335044	647.70	52.4357	0.487

Table 3 Features of the validation instances

5.2 Comparison results

The results are shown in Tables 4 and 5 for the timeouts $10\sqrt{J}$ and J, respectively. Each solution method was run for ten repetitions for each instance, collecting both the best (minimum) and average values of the objective function. Values in boldface are the best average ones among the different techniques. Best known values are highlighted in italics.

For completeness, the tables include results on both training (wlp01-wlp20) and validation instances (wlp21-wlp30).

We notice that we have only a marginal improvement by moving from the shorter to the longer timeout. More precisely, the improvement is 0.4% in general and 0.64% specifically for MR-MS-ILS.

The results shown in Tables 4 and 5 clearly set forth the fact that the best technique is MR-MS-ILS. In order to have a more quantitative measure of the respective positioning, in Table 6 we show the average ranks throughout all

Table 4 Comparative results on training and validation instances with the competition timeout equal to $10\sqrt{J}$ seconds.

	MR-MS-ILS		GRASP		PcEA		MG	
Inst.	min	avg	min	avg	min	avg	min	avg
wlp01	28930	29042.8	*28716*	**28716.0**	34794	38104.6	34377	34377.0
wlp02	54481	54857.5	*52959*	**52978.4**	93654	96409.8	59371	60123.3
wlp03	66664	67573.0	65297	**65982.7**	125439	129441.5	72790	73491.9
wlp04	89567	**89998.2**	92554	92917.8	185456	189776.1	97881	98537.8
wlp05	111657	**112316.8**	113129	114046.1	246786	248870.6	116591	117707.8
wlp06	119246	**119663.3**	121927	122381.9	274556	281500.9	127049	127444.3
wlp07	177144	**178075.4**	179177	181078.2	410276	423626.7	183931	184359.3
wlp08	206264	**207366.9**	212680	214065.8	513355	524780.7	211824	212844.1
wlp09	240643	**241209.6**	250598	252469.3	587296	598667.8	245177	246264.8
wlp10	264070	**265974.0**	282294	285293.5	685886	693202.8	275733	276638.1
wlp11	315432	**317478.4**	329515	333581.0	803541	823183.1	322667	323597.5
wlp12	325662	**326866.6**	343083	347253.7	836416	847748.5	332815	333641.0
wlp13	345187	**346972.7**	368851	370288.4	867534	882054.8	350269	351854.2
wlp14	431272	**433564.5**	469236	471515.2	1132800	1151057.0	437224	439194.0
wlp15	502208	**505621.2**	539085	543393.0	1341840	1362081.0	503872	505720.1
wlp16	583013	586513.0	614605	617661.7	1573080	1590964.0	582578	**584067.9**
wlp17	605499	**607916.4**	656490	662380.1	1675160	1686622.0	615850	618357.3
wlp18	680058	**682308.5**	735216	740149.2	1873040	1888869.0	685807	688308.4
wlp19	811283	816436.7	875530	879310.8	2193560	2216029.0	807495	**810275.4**
wlp20	1047390	**1050399.0**	1139250	1142439.0	3029960	3049038.0	1047750	1051128.0
wlp21	38378	38614.8	38067	**38067.0**	56295	58720.7	44175	44384.8
wlp22	*79097*	**79543.7**	79663	80042.3	162369	165933.8	86152	86498.7
wlp23	127974	**128520.1**	129973	130723.9	292163	296105.7	135018	135725.2
wlp24	181839	**182762.4**	191457	192219.2	442001	451811.3	190984	192143.6
wlp25	251526	**252157.1**	266185	268229.5	636457	643867.1	257740	258823.7
wlp26	321737	324410.2	334955	337259.4	792034	800491.4	321048	**323888.5**
wlp27	420488	**423275.1**	457089	460011.5	1130560	1135313.0	430418	431795.1
wlp28	496949	**498744.1**	537813	544822.5	1295500	1310530.0	503643	505813.4
wlp29	666113	668016.6	694562	698315.3	1715710	1730402.0	*648185*	**650312.2**
wlp30	945626	950802.3	1015340	1018550.0	2657790	2678259.0	*938118*	**941497.7**
Avg	351179.9	352900.0	373843.2	376204.7	922176.9	933115.4	355551.1	356960.5

runs (10×4) upon all instances. In detail, each column shows the average ranks obtained by the solution techniques in ten runs, distinct for each combination of dataset and timeout. To give an intuitive interpretation of these numbers, we mention that if a search method had been better than all the others in all runs, it would have obtained the ranks 1, 2, ..., 10 for its ten runs, resulting in an average rank of 5.5 (as the average of the numbers 1, 2, ..., 10).

Table 6 highlights that there is no remarkable difference between the columns, showing that the ranks are not significantly affected by the specific dataset and the timeout. It also confirms that MR-MS-ILS is indisputably superior, with MG and GRASP following with similar results among them, and PcEA coming clearly last.

Table 5 Comparative results on training and validation instances with a longer timeout equal to J seconds.

	MR-MS-ILS		GRASP		PcEA		MG	
Inst.	min	avg	min	avg	min	avg	min	avg
wlp01	28954	29187.7	*28716*	**28716.0**	37083	38707.3	34377	34377.0
wlp02	54525	54986.0	52964	**52974.0**	94319	97649.3	59802	60309.5
wlp03	67116	67515.7	*64835*	**65228.8**	125562	128428.2	73082	73626.3
wlp04	*89353*	**90097.3**	92230	92713.7	183182	187998.4	97688	98376.7
wlp05	*111198*	**111996.4**	112910	113694.5	242057	246960.4	116926	117589.9
wlp06	*119026*	**119517.6**	121091	121723.7	263728	270517.5	126512	127076.5
wlp07	*175351*	**176970.9**	179576	180507.4	410269	420639.1	182984	183762.0
wlp08	*204443*	**206330.3**	211764	213343.9	506223	518945.9	211290	212434.7
wlp09	*239168*	**240196.1**	249803	251643.2	589107	594147.7	244383	245963.3
wlp10	*263522*	**264732.8**	282039	283748.2	674063	685068.5	274749	275731.7
wlp11	*315004*	**315861.8**	327130	331905.4	794325	807641.3	321717	322793.6
wlp12	*322927*	**325025.8**	343007	344787.2	819777	832156.9	331836	333121.0
wlp13	*343661*	**345289.7**	363988	369420.1	859628	875872.6	350302	351110.8
wlp14	*429773*	**430846.7**	464422	466695.2	1135170	1149954.0	435204	438032.0
wlp15	*501288*	**502256.7**	538021	542999.7	1337170	1352481.0	502656	504619.0
wlp16	*580334*	581952.5	615030	616797.6	1571960	1594143.0	580420	**581677.1**
wlp17	*604664*	**605662.1**	657930	661021.7	1669090	1680890.0	615837	617687.9
wlp18	*676474*	**677861.6**	734691	740262.2	1860850	1885431.0	685085	686665.1
wlp19	806865	809119.0	875776	879397.0	2173710	2201856.0	*806410*	**807670.9**
wlp20	*1039930*	**1042677.0**	1136870	1139800.0	3034740	3051647.0	1046200	1048376.0
wlp21	38432	38664.5	*38067*	**38067.0**	58536	60212.2	44249	44404.8
wlp22	79166	**79535.4**	79802	80008.3	159961	163977.8	85902	86340.1
wlp23	*127456*	**127872.4**	128910	130145.9	285400	291050.9	134577	135443.8
wlp24	*181543*	**182421.1**	190445	191552.5	431239	441818.1	191585	192049.1
wlp25	*249385*	**250501.2**	265852	266933.0	632079	641208.5	258131	258859.5
wlp26	*319674*	**322523.9**	331647	333954.7	783655	797693.4	322686	323645.8
wlp27	*419489*	**421135.7**	452744	456838.3	1116220	1134452.0	428997	430503.6
wlp28	*493148*	**495849.8**	539455	542105.6	1289030	1305156.0	503144	504143.6
wlp29	654788	659738.7	694248	697222.9	1723080	1733075.0	648287	**649255.3**
wlp30	940025	942947.7	1014220	1015930.0	2665460	2682455.0	938323	**940252.1**
Avg	349222.7	350642.5	372939.4	375004.6	917555.8	929074.5	355111.4	356196.6

Table 6 Average ranks obtained by the solvers for 10 repetitions for each instance.

Dataset	Training		Validation	
Timeout	competition	long	competition	long
MR-MS-ILS	8.325	7.695	9.19	8.7
GRASP	20.455	20.395	20.88	20.85
PcEA	35.5	35.5	35.5	35.5
MG	17.72	18.41	16.43	16.95

6 Conclusions

We have proposed a portfolio of metaheuristic techniques for solving the Multi-Source Capacitated Facility Location Problem with Customer Incompatibilities, which consists in selecting the facilities to be open and the shipping plan, with an additional constraint about incompatibilities between pairs of (rival) customers that cannot be served by the same facility.

For the solution of this problem, we designed four metaheuristic methods developed autonomously by different research groups, participating in the competition of the Metaheuristics Summer School (MESS 2020+1).

We performed an extensive experimental analysis by using a training dataset for tuning the algorithms and a validation dataset to compare and rank the methods. The benchmark datasets and the solution checker are publicly available at the MESS 2020+1 website. Results are shown for two different timeouts and for both datasets. The outcome is that there is a clear ranking that is independent of the timeout and the dataset.

In the future, we plan to apply our search methods to other versions of the facility location problem, including different constraints and objectives. In addition, we could test our methods on other datasets with different sizes and structures and compare with state-of-the-art results.

Possible directions for improvements lie in the hybridization of the different methods. Indeed, some of the components coming from one method could be used profitably inside the others – for example, the greedy algorithm as initial solution selection for the other methods.

References

Avella P, Boccia M (2009) A cutting plane algorithm for the capacitated facility location problem. Computational Optimization and Applications 43(1):39–65

Avella P, Boccia M, Sforza A, et al (2009) An effective heuristic for large-scale capacitated facility location problems. Journal of Heuristics 15(6):597

Avella P, Boccia M, Mattia S, et al (2021) Weak flow cover inequalities for the capacitated facility location problem. European Journal of Operational Research 289(2):485–494

Barahona F, Chudak FA (2005) Near-optimal solutions to large-scale facility location problems. Discrete Optimization 2(1):35–50

Birattari M, Yuan Z, Balaprakash P, et al (2010) F-race and iterated F-race: An overview. In: Experimental methods for the analysis of optimization algorithms. Springer, Berlin, p 311–336

Caserta M, Voß S (2020) A general corridor method-based approach for capacitated facility location. International Journal of Production Research 58(13):3855–3880

Cornuéjols G, Nemhauser G, Wolsey L (1983) The uncapacitated facility location problem. Tech. rep., Cornell University Operations Research and Industrial Engineering

Davis LD (1991) Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York

Fernández E, Landete M (2015) Fixed-charge facility location problems. In: Location science. Springer, p 47–77

Festa P, Resende MG (2018) GRASP. In: Marti R, Pardalos P, Resende M (eds) Handbook of Heuristics. Springer, p 465–488

Fischetti M, Ljubić I, Sinnl M (2016) Benders decomposition without separability: A computational study for capacitated facility location problems. European Journal of Operational Research 253(3):557–569

Görtz S, Klose A (2012) A simple but usually fast branch-and-bound algorithm for the capacitated facility location problem. INFORMS Journal on Computing 24(4):597–610

Guastaroba G, Speranza MG (2012) Kernel search for the capacitated facility location problem. Journal of Heuristics 18(6):877–917

Ho SC (2015) An iterated tabu search heuristic for the single source capacitated facility location problem. Applied Soft Computing 27:169–178

Klose A, Drexl A (2005) Facility location models for distribution system design. European Journal of Operational Research 162(1):4–29

Larrañaga P, Kuilpers C, Poza M, et al (1997) Decomposing bayesian networks: Triangulation of the moral graph with genetic algorithms. Statistics and Computing 7:19–34

Lourenço HR, Martin OC, Stützle T (2003) Iterated Local Search, Springer US, Boston, MA, pp 320–353

López-Ibáñez M, Dubois-Lacoste J, Pérez Cáceres L, et al (2016) The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3:43–58

Maia MRH, Plastino A, Penna PHV (2020) MineReduce: An approach based on data mining for problem size reduction. Computers & Operations Research 122:104,995

Melo MT, Nickel S, Saldanha-Da-Gama F (2009) Facility location and supply chain management–a review. European Journal of Operational Research 196(2):401–412

Nethercote N, Stuckey PJ, Becket R, et al (2007) MiniZinc: Towards a standard CP modelling language. In: Bessière C (ed) CP 2007, LNCS, vol 4741. Springer, pp 529–543

Rios E, Coelho IM, Ochi LS, et al (2016) A benchmark on multi improvement neighborhood search strategies in cpu/gpu systems. In: 2016 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW), pp 49–54

Silva JCN, Coelho IM, Souza US, et al (2022) Finding the maximum multi improvement on neighborhood exploration. Optimization Letters 16:97–115

Urli T (2013) json2run: a tool for experiment design & analysis. CoRR abs/1305.1112

Statements and Declarations

Funding

Marcelo R. H. Maia has received research support from the Brazilian Institute of Geography and Statistics (IBGE, Brazil). Alexandre Plastino and Uéverton S. Souza have received research support from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil) [grant numbers 310444/2018-7 and 309832/2020-9]. Uéverton S. Souza has received research support from Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ, Brazil) [grant number E-26/201.344/2021].

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Author Contributions

Marcelo R. H. Maia designed the MR-MS-ILS algorithm under the supervision of Alexandre Plastino and Uéverton S. Souza. Miguel Reula and Consuelo Parreño-Torres designed the GRASP algorithm. Prem Prakash Vuppuluri designed the PcEA algorithm. Sara Ceschia, Mario Pavone and Andrea Schaerf formulated the problem definition, generated the benchmark datasets, designed the MG algorithm and conducted the experiments. The manuscript was jointly written by all authors, who read and approved its final version.

Data Availability

The datasets generated during the current study are available at the MESS 2020+1 website, https://www.ants-lab.it/mess2020/#competition.

APPENDIX D – MAIA, M. R. H.; SANTANA, Í.; ROSSETI, I.; SOUZA, U. S.; PLASTINO, A. "MineReduce-based Metaheuristic for the Minimum Latency Problem". Submitted to the 14th Metaheuristics International Conference (MIC'2022)

MineReduce-based Metaheuristic for the Minimum Latency Problem*

Marcelo Rodrigues de Holanda Maia[1,2][0000−0002−7207−1698],
Ítalo Santana[3][0000−0002−2424−5382],
Isabel Rosseti[1][0000−0001−8560−5313],
Uéverton dos Santos Souza[1][0000−0002−5320−9209], and
Alexandre Plastino[1][0000−0003−4039−0915]

[1] Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brazil
{mmaia,rosseti,ueverton,plastino}@ic.uff.br
[2] Instituto Brasileiro de Geografia e Estatística, Rio de Janeiro, RJ, Brazil
marcelo.h.maia@ibge.gov.br
[3] Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brazil
isantana@inf.puc-rio.br

Abstract. The minimum latency problem is a variant of the well-known travelling salesperson problem where the objective is to minimize the sum of arrival times at vertices. Recently, a proposal that incorporates a data mining process into a state-of-the-art metaheuristic by injecting patterns from high-quality solutions has consistently led to improved results in terms of solution quality and running time for this problem. This paper extends that proposal by leveraging data mining to contract portions of the problem frequently found in high-quality solutions. Our proposal aims at mitigating the burden of searching for improving solutions by periodically solving a reduced version of the original problem. Computational experiments conducted on a well-diversified set of instances demonstrate that our proposal improved solution quality without increasing computational time, introducing 11 new best solutions to the literature.

Keywords: Metaheuristics · Data mining · Size reduction · MLP.

1 Introduction

The minimum latency problem (MLP) is a variant of the well-known travelling salesperson problem where the objective is to minimize the sum of arrival times

* This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil) [grant numbers 310444/2018-7, 310624/2018-5, 309832/2020-9], Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ, Brazil) [grant number E-26/201.344/2021], Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES, Brazil) [grant number 88887.646206/2021-00], and Instituto Brasileiro de Geografia e Estatística (IBGE, Brazil).

at vertices in a Hamiltonian cycle. It can model several real-world applications like distribution logistics, machine scheduling and disaster relief [3, 5, 7].

Recently, a hybrid metaheuristic named MDM-GILS-RVND [19] has appeared as a high-performance algorithm for the MLP. This hybrid algorithm was a result of combining the *Multi Data Mining* (MDM) approach [14] and GILS-RVND [20], a state-of-the-art hybrid metaheuristic for the MLP that combines components of Greedy Randomized Adaptive Search Procedures (GRASP) [6], Iterated Local Search (ILS) [10], and Random Variable Neighborhood Descent (RVND) [21]. The combination with MDM, which relies on data mining to extract patterns from high-quality solutions followed by their insertion into initial solutions, made GILS-RVND significantly improve its state-of-the-art solution quality and computational time results.

In this paper, we propose another improvement through the application of the MineReduce approach, which has achieved promising results for variants of vehicle routing [12] and vertex cover [11] problems.

In the MineReduce approach, the patterns mined from an elite set of solutions are used to perform problem size reduction. A problem instance is reduced to a smaller-size version by contracting or deleting elements that appear in a mined pattern – as they are assumed to be part of a solution for the original instance. Then, the reduced instance is solved, and the solution found is expanded to become a solution for the original instance.

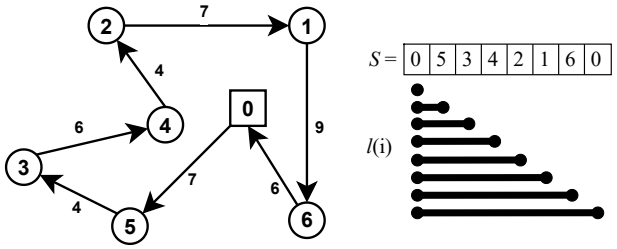
The results of our computational experiments, reported in this paper, show that the proposed MineReduce-based metaheuristic for the MLP, named MR-GILS-RVND, overcomes MDM-GILS-RVND, achieving higher solution quality without increasing CPU running time, particularly for larger instances. It found new best solutions for 11 out of 56 benchmark instances.

The remainder of this paper is organized as follows. Section 2 defines the problem and lists relevant methods from the literature to solve it. Section 3 describes the MDM-GILS-RVND metaheuristic for the MLP proposed in [19]. The MineReduce-based metaheuristic for the MLP proposed in this work is introduced in Section 4. Section 5 reports our experimental results. Finally, conclusions and directions for future work are presented in Section 6.

2 The Minimum Latency Problem

The MLP, described as follows, is a variant of the well-known travelling salesperson problem (TSP) and NP-hard as well [3]. Let $G = (V, A)$ be a directed graph, where $V = \{0, 1, ..., n\}$ is a set composed of $n + 1$ vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs. Vertex 0 is the depot from where the salesperson departs, whereas the set $V' = V \backslash \{0\}$ consists of the remaining vertices representing the n customers. For each arc $(i, j) \in A$, there is an associated travel time t_{ij}. The aim is to find a Hamiltonian cycle $(i_0, i_1, ..., i_{n+1})$ in G, where $i_0 = i_{n+1} = 0$ (i.e., the cycle starts and ends at the depot), that minimizes the sum of arrival times, given by $\sum_{k=1}^{n+1} l(i_k)$, where $l(i_k) = \sum_{m=0}^{k-1} t_{i_m i_{m+1}}$ represents the latency of vertex i_k (i.e., the total travel time to reach i_k).

We present a toy example in Figure 1 to illustrate an MLP solution and the computation of its cost. That is, we show a sequence of customer visits that forms a Hamiltonian cycle S (Fig. 1a) and its associated cost (Fig. 1b), or $f(S)$, which is 164. One can note that MLP is more challenging than TSP since minor changes in the ordering of customers in S can impact drastically $f(S)$ due to the sum of all cumulative costs of each customer, while a TSP solution cost is obtained by a simple sum of all traversed arcs.



(a) A Hamiltonian cycle for MLP (b) $f(S) = 164$

Fig. 1: Example of an MLP solution

In the literature, several exact algorithms were proposed for the MLP [1, 2, 4, 7, 17]. Thanks to these methods, existing instances with up to 200 customers can be solved optimally at the cost of significant computational time. In contrast, heuristics and metaheuristics are alternative methods that consistently find high-quality solutions in controllable running time. The best-performing ones are usually able to efficiently solve instances with up to 1000 customers [5, 7, 13, 16, 18–20]. In particular, GILS-RVND proposed in [20] has achieved state-of-the-art results for MLP and further improved with data mining as MDM-GILS-RVND [19].

3 The MDM-GILS-RVND Metaheuristic

The MDM-GILS-RVND metaheuristic [19] is an algorithm resulting from the incorporation of data mining into GILS-RVND [20], a state-of-the-art hybrid metaheuristic for the MLP that combines components of GRASP, ILS and RVND. This hybrid algorithm applies an approach known as Multi Data Mining (MDM), which uses frequent patterns extracted from good solutions by a data mining process to guide the search, initially proposed for a hybrid version of the GRASP metaheuristic (MDM-GRASP) [14].

In MDM, an *elite set* E keeps the d best solutions found during the execution of the metaheuristic, and a data mining method is periodically executed to extract a set of patterns P from E. These mined patterns are then used in the solution initialization process. The data mining method is based on the FPmax* algorithm [8], which mines maximal frequent itemsets. An itemset is considered frequent if it achieves a given minimum support value, i.e., if it is present in at least a given minimum number of the elite set solutions. Hence, mined patterns are composed of items that frequently appear together in the sub-optimal solutions of the elite set. Intuitively, it is assumed that these items should likely be part of the best solutions to the problem and, thus, they can favour the overall searching process when included in initial solutions. In MDM-GILS-RVND, the set of items representing each solution used for mining refers to the set of all arcs in the solution. Therefore, the mined patterns are sets of frequent paths between customers in the elite solutions.

Usually, in the MDM approach, the data mining method is invoked every time E is considered *stable* (when it does not change for a number of consecutive multi-start iterations). However, the stabilization criterion was not used in MDM-GILS-RVND because the number of multi-start iterations performed by the GILS-RVND metaheuristic is too small (only ten). Instead, it invokes the data mining method once half of the multi-start iterations are completed and, afterwards, whenever the elite set has been updated in the previous multi-start iteration.

The high-level structure of MDM-GILS-RVND is shown in Algorithm 1, where $f(s)$ denotes the cost of the solution s. In the first half of the multi-start iterations (lines 2–8), the algorithm's structure is the same as GILS-RVND, where each iteration builds an initial solution s using a GRASP-like constructive process, which depends on a parameter α that controls the balance between greediness and randomness (line 3). Then, it runs an ILS component to obtain a locally optimum solution s' (line 4) and updates the best solution s^* in case of improvement (line 6). In this first half of iterations, the elite set E is updated whenever a new eligible solution is found within the ILS component. A new solution is inserted into E if it is cheaper than the worst solution of E and different from any solution already present in E. In the second half (lines 9–18), the data mining method is invoked at the first iteration and whenever E has been updated in the previous iteration, returning a set P containing the largest $MaxP$ patterns with a relative minimum support value sup_{min} (line 11). An initial solution s is built by a hybrid constructive process based on one of the mined patterns selected from P (line 13), which starts by inserting all pattern elements in the partial solution and completes it using the original constructive method strategies. The remaining steps are the same as in the first half iterations, including inserting new solutions into E whenever all requirements are met. Once all multi-start iterations are finished, the algorithm returns the best solution found (line 19).

One key aspect of MDM-GILS-RVND is a move evaluation procedure inherited from GILS-RVND. It consists of a framework that uses preprocessed data

Algorithm 1 MDM-GILS-RVND

```
1: f(s*) ← ∞
2: for i ← 1, ..., I_Max/2 do
3:     s ← ConstructiveProc()
4:     s' ← ILS(s)
5:     if f(s') < f(s*) then
6:         s* ← s'
7:     end if
8: end for
9: for i ← 1, ..., I_Max/2 do
10:    if i = 1 or E was updated in iteration i − 1 then
11:        P ← Mine(E, sup_min, MaxP)
12:    end if
13:    s ← HybridConstructiveProc(p ∈ P)
14:    s' ← ILS(s)
15:    if f(s') < f(s*) then
16:        s* ← s'
17:    end if
18: end for
19: return s*
```

structures to compute costs of neighbor solutions in constant amortized time operations [9, 23]. In practice, three data structures are used to store the partial costs of each subsequence of vertices of a local minimum solution, where the cost of every neighbor solution is reached by computing their partial costs on a "by concatenation" fashion. We describe these data structures and how concatenation is performed as follows:

- The *duration* $T(\sigma)$ of a sequence σ, which is the total travel time to perform the visits in the sequence.
- The *cost* $C(\sigma)$ to perform a sequence σ, when starting at time 0.
- The *delay* $W(\sigma)$ associated with a sequence σ, which is the number of customers visited in the sequence.

Let $T(i)$, $C(i)$ and $W(i)$ denote the values of the re-optimization data structures corresponding to a subsequence with only a single vertex i. In this case: $T(i) = 0$ and $C(i) = 0$ since there is no travel time; and $W(i) = 1$ if i is a customer, otherwise $W(i) = 0$. These values can be computed on larger subsequences by induction on the concatenation operator $\oplus$ as follows. Let $\sigma = (\sigma_u, \ldots, \sigma_v)$ and $\sigma' = (\sigma'_w, \ldots, \sigma'_x)$ be two subsequences. The subsequence $\sigma \oplus \sigma' = (\sigma_u, \ldots, \sigma_v, \sigma'_w, \ldots, \sigma'_x)$ is characterized by the following values:

- $T(\sigma \oplus \sigma') = T(\sigma) + t_{\sigma_v \sigma'_w} + T(\sigma')$
- $C(\sigma \oplus \sigma') = C(\sigma) + max(W(\sigma'), 1)(T(\sigma) + t_{\sigma_v \sigma'_w}) + C(\sigma')$
- $W(\sigma \oplus \sigma') = W(\sigma) + W(\sigma')$

4 MineReduce-based Metaheuristic for the MLP

4.1 The MineReduce Approach

The MineReduce approach builds upon the ideas introduced by previous approaches for incorporating data mining into metaheuristics, like the MDM approach [14]. Since the mined patterns are assumed to likely be part of the best solutions to a problem instance, they are well-suited for reducing the size of that instance. For example, the items in a pattern (temporarily fixed in the solution) can be deleted from the instance or merged in a condensed representation.

MineReduce's first steps are to *build an elite set* of solutions and to *mine* patterns from this set. These steps are supposed to be carried out like in the MDM approach, i.e., the best solutions found are stored in the elite set until the data mining method is invoked. The subsequent steps compose a problem size reduction (PSR) process intended to replace a multi-start metaheuristic's initial solution generation method. The *Reduce* step uses a pattern p to transform a problem instance I into a reduced-size instance I'. The *Optimize* step is accomplished through the application of the metaheuristic's original optimization procedures to I'. The *Expand* step transforms the solution to I' into a solution to I, which concludes the MineReduce-based generation of an initial solution.

MineReduce has been successfully applied in metaheuristics for problems such as variants of vehicle routing and vertex cover, with considerable improvements in solution quality and computational time, especially compared with MDM-based metaheuristics [11, 12].

According to Talbi's taxonomy for metaheuristics that incorporate machine learning (ML) in their design [22], MineReduce-based methods are primarily classified as *problem-level ML-supported metaheuristics* since this approach uses data mining for hierarchical problem decomposition (defining and solving smaller subproblems). In addition, they can also be classified as *low-level ML-supported metaheuristics* given that data mining is used in a process that drives the initialization of solutions. Finally, regarding the *learning time* criteria, they are classified as *online* ML-supported metaheuristics since they gather knowledge during the search while solving the problem.

4.2 MineReduce-based GILS-RVND

The reduction process adopted for this problem is similar to the one adopted for a vehicle routing problem [12]. In this case, a mined pattern is a set of subsequences of customers. These subsequences can be contracted by replacing all vertices in a subsequence with a single vertex.

Let $G = (V, A)$ be a directed graph associated with an MLP instance and p a pattern consisting of a set of subsequences of customer vertices in that instance. Let $G^* = (V^*, A^*)$ be a directed graph associated with the corresponding reduced instance based on p. Such a reduced version can be obtained as follows. Initially, G^* is defined as a copy of G. For each subsequence $\sigma = (i_1, i_2, ..., i_{|\sigma|}) \in p$ selected to be contracted, each of the customers in σ is removed from G^* – that

is, the vertex corresponding to the customer is removed from V^* and the arcs that connect that vertex to the others are removed from A^*. Then, a customer vertex i_σ corresponding to the subsequence is added to V^* and arcs connecting i_σ to the other vertices in V^* are added to A^*. The travel time from each vertex $i^* \in V^*$ to i_σ is given by $t_{i^* i_\sigma} = t_{i^* i_1}$, that is, the travel time from i^* to i_1 (the first customer in σ). The travel time from i_σ to each vertex $i^* \in V^*$ is given by $t_{i_\sigma i^*} = t_{i_{|\sigma|} i^*}$, that is, the travel time from $i_{|\sigma|}$ (the last customer in σ) to i^*.

The values in the "by concatenation" framework structures for a subsequence with only the single vertex i_σ are defined as $T(i_\sigma) = T(\sigma)$, $C(i_\sigma) = C(\sigma)$ and $W(i_\sigma) = W(\sigma)$. Note that the need to adapt the "by concatenation" framework structures to work seamlessly with reduced instances made this application of MineReduce challenging even though the approach had previously been applied to another routing problem.

The structure of the MineReduce-based version of GILS-RVND, called MR-GILS-RVND, is depicted in Algorithm 2. The difference to Algorithm 1 is the use of a MineReduce-based constructive process instead of the hybrid constructive process from MDM-GILS-RVND in the last β iterations (line 16).

Algorithm 2 MR-GILS-RVND

```
1: f(s*) ← ∞
2: for i ← 1, ..., I_Max/2 do
3:     s ← ConstructiveProc()
4:     s' ← ILS(s)
5:     if f(s') < f(s*) then
6:         s* ← s'
7:     end if
8: end for
9: for i ← 1, ..., I_Max/2 do
10:    if i = 1 or E was updated in iteration i − 1 then
11:        P ← Mine(E, sup_min, MaxP)
12:    end if
13:    if i ≤ (I_Max/2) − β then
14:        s ← HybridConstructiveProc(p ∈ P)
15:    else
16:        s ← MR-ConstructiveProc(p ∈ P)
17:    end if
18:    s' ← ILS(s)
19:    if f(s') < f(s*) then
20:        s* ← s'
21:    end if
22: end for
23: return s*
```

The MineReduce-based constructive process, presented in Algorithm 3, is a PSR process based on a pattern, as defined by the MineReduce approach.

Algorithm 3 MR-ConstructiveProc(p)

```
1: ReduceInstance(p, γ)
2: s ← ConstructiveProc()
3: s' ← ILS(s)
4: s_0 ← ExpandSolution(s')
5: return s_0
```

In this implementation, we sort all subsequences in a pattern in decreasing length (number of traversed arcs) order. Then, we contract subsequences from the longest to the shortest until we have used a portion γ of all pattern's arcs. The adoption of this strategy was motivated by preliminary tests showing that the mined patterns contained too many arcs, producing reduced instances that were too small. Hence, after expanding solutions found for the reduced instances, a considerable effort was still necessary for the local search on the original instance. Using only a portion of the arcs in a pattern adds control to the reduction factor. Finally, we chose to favour longer subsequences because they are less likely to occur than short subsequences given the same minimum support. Therefore, they represent more robust and relevant portions of the patterns.

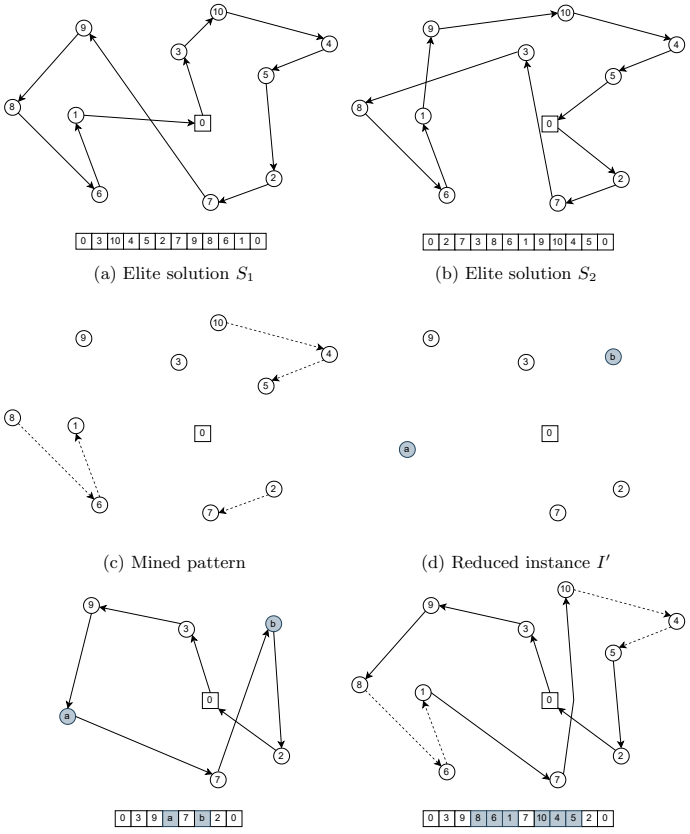
In Algorithm 3, the instance is reduced based on the provided pattern p (line 1). Then, a solution for the reduced instance is obtained by applying the original constructive and ILS methods from GILS-RVND (lines 2–3). Finally, the solution found is expanded, producing a solution for the original instance (line 4), which is returned (line 5).

Fig. 2 illustrates the application of MineReduce's PSR process to an MLP instance. Let S_1 and S_2 be two solutions composing an elite set (Fig. 2a and Fig. 2b, respectively) and $sup_{min} = 100\%$. The mined pattern is depicted in dashed lines in Fig. 2c. The longest subsequences in these patterns – $(8, 6, 1)$ and $(10, 4, 5)$ – are contracted into vertices a and b, respectively, resulting in the reduced instance I' shown in Fig. 2d. Then, a solution for I' (Fig. 2e is obtained using the original construction and search methods from GILS-RVND and expanded to become a solution for the original instance (Fig. 2f).

5 Computational Results

We have assessed the performance of our proposed method, MR-GILS-RVND, by running computational experiments comparing it to the original state-of-the-art MDM-GILS-RVND metaheuristic [19]. We have built MR-GILS-RVND upon the original MDM-GILS-RVND source code. Both were implemented in C++ and compiled with g++ 4.4.7. The experiments were run in a single thread on an Intel® Core™ i7-5500U 2.40 GHz CPU.

In these experiments, we used a set composed of 56 benchmark instances with 120 to 1379 customers from TSPLIB [15], which was also used to compare MDM-GILS-RVND and GILS-RVND in [19]. We ran both algorithms on ten tests using different random seeds for each instance.



(a) Elite solution S_1 (b) Elite solution S_2

(c) Mined pattern (d) Reduced instance I'

(e) Solution for I' (f) Expanded solution

Fig. 2: MineReduce's PSR applied to an MLP instance

The configuration of the MDM-GILS-RVND parameters adopted in [19] was used for both methods in our experiments: $I_{Max} = 10$ (the number of multi-start iterations); $I_{ILS} = min(100, n)$ (the number of ILS iterations); $R = \{0.00, 0.01, \ldots, 0.25\}$ (the possible values for α, a value that controls the greediness level of the constructive process, randomly chosen for each multi-start iteration); $sup_{min} = 70\%$ (the relative minimum support of the mined patterns); $d = 10$ (the capacity of the elite set); and $MaxP = 5$ (the number of patterns returned by the data mining process).

The parameters introduced in MR-GILS-RVND – β (the number of multi-start iterations applying the MineReduce-based constructive process) and γ, the portion of all arcs in a pattern that are used for contraction – had their values tuned based on the best trade-off between solution quality and computational time found in tests ran on a sample with 12 out of the 56 benchmark instances (with 262 to 1291 customers). We refer to these 12 instances as the *training set*. The following values were considered for tuning the parameters: $\{1, 2, \ldots, 5\}$ for β, and $\{60\%, 2/3, 70\%, 80\%\}$ for γ. The best values found were $\beta = 4$ and $\gamma = 2/3$. Hence, we used these values in the experiments ran on the remaining 44 instances – the *validation set*. MR-GILS-RVND, with this best parameter configuration, found new best solutions for 6 out of the 12 instances in the training set, which are reported in Table 1.

Table 1: New best solutions found by MR-GILS-RVND for training instances

Instance	Solution	Instance	Solution	Instance	Solution
gil262	285,043	rd400	2,762,336	gr431	21,154,740
si535	12,246,046	gr666	63,454,259	vm1084	94,608,098

Table 2 summarizes the results obtained in the experiments using the instances in the validation set. MDM-GILS-RVND and MR-GILS-RVND are compared regarding the numbers of wins in best cost, average cost and average CPU running time, the number of new best solutions found, and the summed number of best known solutions (BKS) and new best solutions found.

The comparison on all 44 benchmark instances shows that MR-GILS-RVND overcomes MDM-GILS-RVND regarding solution quality, obtaining better solutions for most instances. Furthermore, MR-GILS-RVND found new best solutions for five instances in this set.

On the other hand, MDM-GILS-RVND obtained more wins in average time. This can be explained by the fact that all 15 instances with $n \leq 195$ are in the validation set. These small instances are easier than the others, and all of them have been solved optimally by exact methods. The original MDM-GILS-RVND finds their optimal solutions in a few seconds. Thus, the slight computational overhead introduced by applying the size reduction process in MR-GILS-RVND

Table 2: Results summary (all validation instances)

	MDM-GILS-RVND	MR-GILS-RVND
Wins Best Cost	4	6
Wins Avg. Cost	12	18
Wins Avg. Time	25	19
New best	-	5
Nb BKS + new best	37	39

is not compensated by a convergence speedup as usual since the solutions cannot be further improved.

Therefore, a separate comparison is presented in Table 3 considering only the instances with $n > 195$ (the 29 largest instances). For these larger instances, both methods are technically tied regarding average computational time, whereas the superiority of MR-GILS-RVND regarding solution quality is further evidenced, with about twice the number of wins of MDM-GILS-RVND in average cost. Hence, these results show that the MineReduce approach, applied in MR-GILS-RVND, improved solution quality without increasing computational time over MDM-GILS-RVND.

Table 3: Results summary ($n > 195$)

	MDM-GILS-RVND	MR-GILS-RVND
Wins Best Cost	4	6
Wins Avg. Cost	9	17
Wins Avg. Time	15	14
New best	-	5
Nb BKS + new best	22	25

Table 4 presents the detailed solution cost comparison results for the validation set. For each instance, we report the BKS cost from the literature and the best and average costs obtained by each method over the ten runs. Winning values in the comparison are in bold, solution costs matching the BKS are presented in italics, and new best solution costs are underlined.

Table 5 presents the detailed running time comparison results for the validation set. For each instance, we report the average CPU running time in seconds obtained by each method over the ten runs. Again, winning values in the comparison are presented in bold.

6 Conclusion

This paper proposes a hybrid metaheuristic for the MLP based on the MineReduce approach, which uses patterns extracted from an elite set of solutions using data mining to reduce the size of problem instances.

The proposed method, named MR-GILS-RVND, was built through the application of the MineReduce approach on MDM-GILS-RVND [19], a state-of-the-art algorithm that applies another approach for incorporating data mining into metaheuristics, which consists of inserting mined patterns in initial solutions.

We conducted computational experiments with 56 benchmark instances from TSPLIB to compare MDM-GILS-RVND and MR-GILS-RVND. The reported results evidence that our proposed MR-GILS-RVND overcomes MDM-GILS-RVND, achieving better solutions for most instances without increasing computational time. Furthermore, the results show a more evident superiority of the MineReduce-based method on more challenging instances (with $n \geq 230$).

These results reinforce the potential of the MineReduce approach for improving the performance of metaheuristics within a conceptually simple framework already applied to other combinatorial optimization problems. In future work, further investigation on the current application can be made to other challenging problem variants (e.g., time windows) that may require specialized design in the "by concatenation" framework. Finally, we expect that the contributions made in this work lead to a better comprehension of challenges involving problem size reduction for hard combinatorial optimization problems.

References

1. Abeledo, H., Fukasawa, R., Pessoa, A., Uchoa, E.: The time dependent traveling salesman problem: polyhedra and algorithm. Mathematical Programming Computation **5**, 27–55 (2013). https://doi.org/10.1007/s12532-012-0047-y
2. Angel-Bello, F., Alvarez, A., García, I.: Two improved formulations for the minimum latency problem. Applied Mathematical Modelling **37**(4), 2257–2266 (2013). https://doi.org/10.1016/j.apm.2012.05.026
3. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M.: The minimum latency problem. In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing. p. 163–171. STOC '94, Association for Computing Machinery, New York, NY, USA (1994). https://doi.org/10.1145/195058.195125
4. Bulhões, T., Sadykov, R., Uchoa, E.: A branch-and-price algorithm for the minimum latency problem. Computers & Operations Research **93**, 66–78 (2018). https://doi.org/10.1016/j.cor.2018.01.016
5. Campbell, A.M., Vandenbussche, D., Hermann, W.: Routing for relief efforts. Transportation Science **42**(2), 127–145 (2008). https://doi.org/10.1287/trsc.1070.0209
6. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6**(2), 109–133 (1995). https://doi.org/10.1007/BF01096763

7. Fischetti, M., Laporte, G., Martello, S.: The delivery man problem and cumulative matroids. Operations Research **41**(6), 1055–1064 (1993). https://doi.org/10.1287/opre.41.6.1055
8. Grahne, G., Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets. In: Goethals, B., Zaki, M.J. (eds.) Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (2003)
9. Kindervater, G.A.P., Savelsbergh, M.W.P.: Vehicle routing: handling edge exchanges, pp. 337–360. Princeton University Press (2018). https://doi.org/10.1515/9780691187563-013
10. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated Local Search, pp. 320–353. Springer US, Boston, MA (2003). https://doi.org/10.1007/0-306-48056-5_11
11. Maia, M.R.H., Plastino, A., Souza, U.S.: MineReduce for the minimum weight vertex cover problem. In: Proceedings of the International Conference on Optimization and Learning (OLA'2020). pp. 11–22 (2020)
12. Maia, M.R.H., Plastino, A., Penna, P.H.V.: MineReduce: An approach based on data mining for problem size reduction. Computers & Operations Research **122**, 104995 (2020). https://doi.org/10.1016/j.cor.2020.104995
13. Mladenović, N., Urošević, D., Goos, P., Hanafi, S.: Variable neighborhood search for the travelling deliveryman problem. 4OR **11**, 57–73 (2013). https://doi.org/10.1007/s10288-012-0212-1
14. Plastino, A., Barbalho, H., Santos, L.F.M., Fuchshuber, R., Martins, S.L.: Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic. Journal of Heuristics **20**, 1899–1911 (2014). https://doi.org/10.1007/s10732-013-9231-0
15. Reinelt, G.: TSPLIB–a traveling salesman problem library. ORSA Journal on Computing **3**(4), 376–384 (1991). https://doi.org/10.1287/ijoc.3.4.376
16. Rios, E., Coelho, I.M., Ochi, L.S., Boeres, C., Farias, R.: A benchmark on multi improvement neighborhood search strategies in cpu/gpu systems. In: 2016 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW). pp. 49–54 (2016)
17. Roberti, R., Mingozzi, A.: Dynamic ng-path relaxation for the delivery man problem. Transportation Science **48**(3), 413–424 (2014). https://doi.org/10.1287/trsc.2013.0474
18. Salehipour, A., Sörensen, K., Goos, P., Bräysy, O.: Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem. 4OR **9**(2), 189–209 (2011). https://doi.org/10.1007/s10288-011-0153-0
19. Santana, I., Plastino, A., Rosseti, I.: Improving a state-of-the-art heuristic for the minimum latency problem with data mining. International Transactions in Operational Research **29**(2), 959–986 (2022). https://doi.org/10.1111/itor.12774
20. Silva, M.M., Subramanian, A., Vidal, T., Ochi, L.S.: A simple and effective metaheuristic for the minimum latency problem. European Journal of Operational Research **221**(3), 513–520 (2012). https://doi.org/10.1016/j.ejor.2012.03.044
21. Subramanian, A., Drummond, L., Bentes, C., Ochi, L., Farias, R.: A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research **37**(11), 1899–1911 (2010). https://doi.org/10.1016/j.cor.2009.10.011
22. Talbi, E.G.: Machine learning into metaheuristics: A survey and taxonomy. ACM Computing Surveys **54**(6) (2021). https://doi.org/10.1145/3459664
23. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A unifying view on timing problems and algorithms. Tech. rep., CIRRELT (2011)

Table 4: Solution cost comparison

Instance	BKS	MDM-GILS-RVND		MR-GILS-RVND	
		Best Cost	Avg. Cost	Best Cost	Avg. Cost
gr120	363,454[a,b]	*363,454*	***363,454.0***	*363,454*	363,700.3
pr124	3,154,346[a,b]	*3,154,346*	*3,154,346.0*	*3,154,346*	*3,154,346.0*
bier127	4,545,005[a,b]	*4,545,005*	4,545,691.9	*4,545,005*	***4,545,005.0***
ch130	349,874[a,b]	*349,874*	*349,903.5*	*349,874*	*349,903.5*
pr136	6,199,268[a,b]	*6,199,268*	**6,200,041.6**	*6,199,268*	6,200,360.9
gr137	4,061,498[a,b]	*4,061,498*	*4,061,498.0*	*4,061,498*	*4,061,498.0*
pr144	3,846,137[a,b]	*3,846,137*	*3,846,137.0*	*3,846,137*	*3,846,137.0*
ch150	444,424[a,b]	*444,424*	*444,424.0*	*444,424*	*444,424.0*
kroA150	1,825,769[a,c]	*1,825,769*	*1,825,769.0*	*1,825,769*	*1,825,769.0*
kroB150	1,786,546[a,c]	*1,786,546*	*1,786,546.0*	*1,786,546*	*1,786,546.0*
pr152	5,064,566[a,b]	*5,064,566*	*5,064,566.0*	*5,064,566*	*5,064,566.0*
u159	2,972,030[a,c]	*2,972,030*	2,972,204.2	*2,972,030*	2,972,204.2
si175	1,808,532[a,c]	*1,808,532*	*1,808,532.0*	*1,808,532*	*1,808,532.0*
brg180	174,750[a,c]	*174,750*	*174,750.0*	*174,750*	*174,750.0*
rat195	218,632[a,c]	*218,632*	**218,736.6**	*218,632*	218,771.3
d198	1,186,049[c]	*1,186,049*	1,186,273.3	*1,186,049*	***1,186,049.0***
kroA200	2,672,437[c]	*2,672,437*	2,672,444.2	*2,672,437*	***2,672,437.0***
kroB200	2,669,515[c]	*2,669,515*	**2,675,993.6**	*2,669,515*	2,676,444.9
gr202	2,909,247[d]	*2,909,247*	**2,913,368.4**	*2,909,247*	2,913,957.8
ts225	13,240,046[d]	*13,240,046*	***13,240,046.0***	*13,240,046*	13,240,533.0
tsp225	402,783[d]	*402,783*	402,933.3	*402,783*	**402,925.4**
pr226	7,196,869[d]	*7,196,869*	*7,196,869.0*	*7,196,869*	*7,196,869.0*
gr229	10,725,914[d]	*10,725,914*	10,731,249.9	*10,725,914*	**10,729,841.4**
pr264	5,471,615[d]	*5,471,615*	*5,471,615.0*	*5,471,615*	*5,471,615.0*
a280	346,989[d]	*346,989*	347,106.9	*346,989*	**347,098.9**
pr299	6,556,628[d]	*6,556,628*	**6,559,030.8**	*6,556,628*	6,559,653.7
lin318	5,619,810[d]	*5,619,810*	5,630,590.5	*5,619,810*	5,630,590.5
fl417	1,874,242[d]	*1,874,242*	***1,874,242.0***	*1,874,242*	1,874,246.0
pr439	17,829,541[d]	*17,829,541*	17,868,632.7	*17,829,541*	**17,866,993.1**
pcb442	10,301,705[d]	*10,301,705*	10,321,465.7	*10,301,705*	**10,321,299.8**
d493	6,677,458[d]	***6,677,458***	6,687,268.2	6,680,576	**6,685,445.4**
ali535	31,860,679[d]	*31,860,679*	31,910,477.9	*31,860,679*	**31,907,551.1**
pa561	658,870[d]	660,590	661,790.6	**660,127**	**661,757.9**
p654	7,827,273[d]	*7,827,273*	**7,827,867.8**	*7,827,273*	7,827,953.4
d657	14,112,540[d]	*14,112,540*	14,195,797.6	*14,112,540*	**14,194,627.6**
u724	13,504,408[d]	13,504,408	**13,537,514.7**	**13,491,599**	13,543,353.9
dsj1000	7,642,715,113[d]	7,642,715,113	7,664,531,851.2	**7,642,418,952**	**7,662,310,139.5**
dsj1000ceil	7,646,395,679[d]	7,646,395,679	7,676,973,751.4	**7,632,965,540**	**7,674,650,570.0**
si1032	46,896,355[d]	*46,896,355*	**46,896,783.6**	*46,896,355*	46,897,212.2
u1060	102,508,056[d]	102,539,819	102,759,493.6	**102,436,120**	**102,681,878.9**
pcb1173	30,890,385[d]	***30,890,385***	30,957,008.7	30,891,243	**30,945,301.5**
rl1304	144,592,447[d]	144,592,447	145,398,549.2	**144,585,587**	**145,320,131.5**
rl1323	155,697,857[d]	**155,719,283**	156,273,365.5	155,762,567	**156,229,029.1**
nrw1379	35,291,795[d]	***35,291,795***	**35,456,093.0**	35,329,106	35,461,487.4
Wins		4	12	**6**	**18**
Wins ($n > 195$)		4	9	**6**	**17**

[a]Optimality proven.
[b]From [17].
[c]From [4].
[d]From [19].

Table 5: Average running time comparison

Instance	MDM-GILS-RVND	MR-GILS-RVND
gr120	**10.73**	10.88
pr124	**6.83**	7.14
bier127	**10.31**	10.65
ch130	**11.99**	12.90
pr136	19.30	**17.29**
gr137	**9.25**	9.79
pr144	**11.31**	11.43
ch150	**14.19**	14.69
kroA150	20.77	**20.42**
kroB150	19.41	**18.68**
pr152	**13.13**	14.09
u159	**16.59**	16.70
si175	**19.04**	20.98
brg180	23.84	**23.29**
rat195	45.90	**45.39**
d198	42.79	**38.52**
kroA200	45.14	**43.15**
kroB200	49.30	**44.68**
gr202	39.28	**38.36**
ts225	**35.15**	47.88
tsp225	**56.59**	56.65
pr226	**37.91**	39.55
gr229	**53.37**	53.63
pr264	**51.31**	52.99
a280	117.64	**107.57**
pr299	**104.23**	106.88
lin318	**129.40**	133.96
fl417	**469.94**	499.70
pr439	405.55	**399.02**
pcb442	604.91	**601.88**
d493	1,034.42	**883.64**
ali535	**1,345.69**	1,361.16
pa561	**1,649.07**	1,728.66
p654	1,825.05	**1,742.79**
d657	2,571.72	**2,508.68**
u724	**4,209.10**	4,430.35
dsj1000	**17,630.28**	17,792.34
dsj1000ceil	17,185.99	**17,033.65**
si1032	2,794.02	**2,661.32**
u1060	**13,336.35**	14,128.16
pcb1173	19,192.65	**19,024.96**
rl1304	**17,636.57**	18,013.06
rl1323	22,081.21	**21,322.80**
nrw1379	**45,325.74**	48,402.52
Wins	**25**	19
Wins ($n > 195$)	**15**	14

APPENDIX E – MAIA, M. R. H.; PLASTINO, A.; SOUZA, U. S. "An improved hybrid genetic search with data mining for the CVRP". In: 12th DIMACS Implementation Challenge: Vehicle Routing Problems[1]

[1] Available from: http://dimacs.rutgers.edu/events/details?eID=2073

An improved hybrid genetic search with data mining for the CVRP

Marcelo Rodrigues de Holanda Maia[1,2] **Alexandre Plastino**[1]
Uéverton dos Santos Souza[1]

[1]Instituto de Computação
Universidade Federal Fluminense

[2]Escola Nacional de Ciências Estatísticas
Instituto Brasileiro de Geografia e Estatística

Email of Corresponding Author: mmaia@ic.uff.br

Abstract: The hybrid genetic search (HGS) metaheuristic has produced outstanding results for several variants of the vehicle routing problem. A recent implementation of HGS specialized to the capacitated vehicle routing problem (CVRP) stands as a state-of-the-art method for this variant. This paper proposes an improved HGS for the CVRP obtained by incorporating a new method for initializing the population to guide the search more efficiently and effectively. The initialization method introduced in this work combines an approach based on frequent patterns extracted from good solutions by a data mining process and a randomized version of the Clarke and Wright savings heuristic. As observed in our experimental comparison with the original algorithm, the proposed method provides significant improvements to the primal integral, a performance measure that rewards a balance of convergence speed and solution quality.

Team name: UFF–IC
Solver name: MDM–HGS
VRP tracks: CVRP

1 Introduction

In the last decade, the hybrid genetic search (HGS) metaheuristic has produced outstanding results for several variants of the vehicle routing problem (VRP), standing as a state-of-the-art method for this family of combinatorial optimization problems [10, 11, 12]. Its most recent implementation, specialized to the capacitated vehicle routing problem (CVRP), has outperformed all of the main algorithms available for this VRP variant in extensive experimental comparisons [10]. The results obtained by this specialized algorithm, referred to as HGS–CVRP, put it in a leading position among the existing CVRP methods. As an improvement upon the current top-performing method's results would represent a significant accomplishment, we aimed at producing an improved version of HGS for the CVRP in this work.

The combination of crossover- and neighborhood-based search in HGS, especially with the SWAP* neighborhood structure introduced in HGS–CVRP, provides an excellent balance of diversification and solution improvement. Hence, it seems there is little room for improvement in those components. There was, however, another research direction with a great potential for improvement: by introducing a new method for initializing the population, we could guide the search more efficiently and effectively.

Hence, this paper proposes an improved version of HGS for the CVRP, obtained by incorporating a new method for initializing the population. This new solution generation method relies on an approach we refer to as MDM (which stands for *multi data mining*) that uses frequent patterns extracted from good solutions by a data mining process [7] and a randomized version of the Clarke and Wright savings heuristic (CW) [5].

The MDM approach has been initially proposed for a hybrid version of the GRASP metaheuristic (MDM–GRASP), but it has later been used with other multi-start metaheuristics, including a multi-start ILS for the heterogeneous fleet VRP [6]. The CW heuristic is very effective in producing a relatively good solution in a short computing time and has been used for initializing solutions in some recent state-of-the-art methods [1, 2, 3].

We compare the original HGS–CVRP and our proposed algorithm, which we call MDM–HGS, in computational experiments evaluating both with respect to the primal integral (PI), a performance measure that rewards a balance of convergence speed and solution quality [4]. The results show that our version of HGS significantly outperforms the original one, converging faster and improving solution quality.

2 HGS with Data Mining for the CVRP

This section presents the improved version of HGS with data mining we propose for the CVRP. A general description of HGS–CVRP is given in Section 2.1. Section 2.2 describes MDM–HGS, our proposed version of HGS–CVRP.

2.1 HGS–CVRP: the Base Algorithm

HGS–CVRP is a population-based hybrid algorithm that combines neighborhood search strategies with a genetic crossover operator. At a high abstraction level, it works as follows. Once the population is initialized, the algorithm starts an iterative process consisting of four steps: (i) selecting two parent individuals from the population; (ii) recombining them by applying the crossover operator to produce an offspring; (iii) improving the offspring solution with a local search; (iv) inserting the resulting solution in the population and applying a population management mechanism. This iterative process is repeated until N_{IT} consecutive iterations without improvement are completed or a time limit T_{MAX} is exceeded. If T_{MAX} is defined, the algorithm restarts, re-initializing the population, every time it completes N_{IT} consecutive iterations without improvement.

In its population initialization phase, HGS–CVRP generates 4μ random solutions (where μ is a parameter representing the minimum population size), improves them through local search, and inserts them in the population.

2.2 MDM–HGS

MDM–HGS is a new version of HGS–CVRP based on the MDM approach, initially proposed for a hybrid version of the GRASP metaheuristic (MDM-GRASP) [7]. In the MDM approach, an *elite set* E keeps the α best solutions found during the execution of the metaheuristic. When E is considered *stable* (usually when it does not change for a number of consecutive multi-start iterations), a data mining method is used to extract a set of patterns P from its solutions. Then, the solution initialization process uses the patterns from P.

The realization of the MDM approach elements in our MDM–HGS implementation is similar to that adopted in the multi-start ILS for the heterogeneous fleet VRP from [6]. The data mining method returns the β largest frequent patterns with minimum support γ found in E. Each pattern is a set of paths connecting customers, which appear together in at least $\gamma|E|$ solutions from E. The difference to the heterogeneous fleet variant is the assignment of a vehicle type to each path, which does not apply to the canonical CVRP addressed in this work. These sets of paths represent elements that frequently appear together in good solutions. Hence, the mined patterns are used for solution initialization to guide the search through more promising regions in the solution space.

E is updated every time a new feasible solution is inserted in the population if E is not full or the new solution is better than the worst solution in E. It is considered *stable* when one of the following criteria is met: (i) E has not been mined yet, and its contents have not changed for a number of consecutive restarts greater than or equal to δR_{MAX}, where R_{MAX} is the maximum number of restarts, dynamically estimated based on the elapsed time and the number of completed restarts; or (ii) the contents of E have changed after it was last mined, but they have not changed for a number of consecutive restarts greater than or equal to δR_{MAX}.

Our new solution generation method is a randomized version of the well-known CW heuristic

that uses the paths from a mined pattern to initialize routes. Its randomization mechanism is a roulette wheel selection as in the CW implementation discussed in [8]. The *savings list* S is built when the problem instance data is loaded. It contains all edges that connect two customers in the input graph, sorted in decreasing order of their corresponding savings values. A saving value represents the saving in cost obtained by concatenating two routes, one where customer i is adjacent to the depot and another where customer j is adjacent to the depot, via the insertion of the edge $\langle i, j\rangle$ in the solution. It is given by $s_{ij} = d_{0i} + d_{0j} - d_{ij}$, where d_{ij} is the distance between vertices i and j, and vertex 0 is the depot.

The randomized CW solution generation method based on a pattern p is summarized in Algorithm 1. The solution is initialized with one route for each path in p (line 1) and one route for each customer that is not in p (line 2). Then the savings list is iteratively processed (lines 3–7). At each step, an edge $\langle i, j\rangle$ is selected among the t (a random number between 2 and 6, inclusive) first unvisited entries in S using the roulette wheel method (line 5). If customers i and j are both adjacent to the depot, in two different routes, and the vehicle capacity is sufficient for the total demand of both routes, then these routes are concatenated through the insertion of $\langle i, j\rangle$ in the solution (line 7). Finally, after S has been processed, to avoid an excessive number of routes, for each route that remains with only one customer i, i is moved to the cheapest position adjacent to the depot among all longer routes (line 9).

Algorithm 1 Pattern-based randomized CW solution generation

1: Initialize a route with each path in pattern p;
2: Initialize a route with each customer that is not in p;
3: **while** S contains unvisited positive-valued entries **do**
4: $t \leftarrow$ a random number from the interval $[2, 6]$;
5: $\langle i, j\rangle \leftarrow$ RouletteWheel(S, t);
6: **if** i and j are in two different routes, both are adjacent to the depot, and the vehicle capacity is sufficient **then**
7: Concatenate the two routes through $\langle i, j\rangle$;
8: **for all** routes with only one customer i **do**
9: Move i to the cheapest position adjacent to the depot in a longer route;
10: Return the generated solution;

In the population initialization, MDM–HGS generates $(1 - \eta)\varepsilon\mu$ solutions using the method described in Algorithm 1 and $\eta\varepsilon\mu$ random solutions using the original method from HGS–CVRP. Every generated solution is improved through local search and inserted in the population.

Two new parameters are introduced. $\eta \in (0, 1)$ controls the randomness level in the initial population to keep an appropriate level of diversification. ε replaces the constant 4 from HGS–CVRP as we considered that generating 4μ solutions for the initial population could excessively delay convergence, so we wish to try lower values for this factor.

Note that the patterns set P is empty when the population is first initialized. In that case, pattern p in Algorithm 1 is an empty set, so the method behaves like an ordinary CW heuristic. After the first activation of the data mining method, one of the mined patterns in P is selected for each solution generated by Algorithm 1.

3 Parameters Tuning

We have kept the values for the HGS–CVRP parameters as defined in [10]. The parameters introduced in MDM–HGS and the respective values considered on the tuning process are shown in Table 1. They have been tuned on the benchmark X instances [9] via an iterative process where, at each step, one parameter value varied while the remaining parameter values were fixed. The instances have been separated into three groups corresponding to size ranges based on the number of vertices n, and the tuning process was conducted for each range independently to minimize the average primal integral. Table 2 presents the best configurations found.

Table 1: MDM–HGS parameters and respective values considered on tuning

Parameter	Description	Considered values
α	Elite set capacity	$\{5, 10, 15\}$
β	Number of patterns mined from the elite set	$\{1, 2, \ldots, 10\}$
γ	Relative minimum suppport of mined patterns	$\{0.4, 0.5, \ldots, 1.0\}$
δ	Coefficient used in elite set stabilization criteria	$\{0.03, 0.04, \ldots, 0.07\}$
ε	Coefficient that controls initial population size	$\{1, 2, 3, 4\}$
η	Initial population randomness level	$\{0.1, 0.2, \ldots, 0.9\}$

Table 2: Best parameter configurations

Range	α	β	γ	δ	ε	η
$n \leq 200$	5	5	0.8	0.05	1	0.1
$200 < n \leq 400$	10	5	0.8	0.05	1	0.8
$n > 400$	5	5	0.8	0.05	1	0.2

4 Results

The experiments were run on AMD EPYC 7532 @ 2.4 GHz CPUs. We have performed ten independent runs of the algorithms with different seeds for each instance. A wall clock time limit of 1,800 seconds was set for instances with $n \leq 200$, 3,600 seconds for $200 < n \leq 400$, and 7,200 seconds for $n > 400$. Detailed tables of results are presented in Appendix A. For each of the 141

instances, we report the best known solution (BKS)[1] and, for each algorithm, the average primal integral (Avg PI), the average solution cost (Avg Cost), the percentage gap of Avg Cost from the BKS (Gap), and the best solution cost (Best Cost).

Table 3 summarizes the results, showing for each algorithm: the global average primal integral; the average Gap; the number of wins in Avg PI, Avg Cost, and Best Cost; and the numbers of best known and new best solutions found. We present three comparisons: one on all 141 instances, one on the X instances (used for tuning), and another on all instances not in the X set. As these comparisons show, MDM–HGS obtains significantly better PI values than HGS–CVRP and solutions of higher quality. HGS–CVRP found two new best solutions (for instances Loggi-n501-k24[2] and ORTEC-n701-k64) and MDM–HGS found other two (for instances Loggi-n601-k42 and Loggi-n901-k42).

Table 3: Results summary

	All instances		X instances		All except X instances	
	HGS–CVRP	MDM–HGS	HGS–CVRP	MDM–HGS	HGS–CVRP	MDM–HGS
Global Avg PI	0.5243853	0.4250732	0.1192855	0.1147131	1.5124335	1.1820491
Avg Gap	0.43%	0.33%	0.08%	0.07%	1.30%	0.97%
Wins Avg PI	42	99	32	68	10	31
Wins Avg Cost	37	58	26	38	11	20
Wins Best Cost	18	52	12	33	6	19
No. BKSs	69	71	52	51	17	20
New Best	2	2	0	0	2	2

5 Conclusion

This paper presented MDM–HGS, an improved version of the state-of-the-art HGS metaheuristic for the CVRP (HGS–CVRP) [10]. MDM–HGS was produced by introducing a new method for initializing the population, based on patterns extracted from a set of good solutions using data mining and on a randomized Clarke and Wright savings heuristic. We conducted experiments to compare the performance of the proposed MDM–HGS with that of HGS–CVRP. The results obtained demonstrate that MDM–HGS outperforms HGS–CVRP with respect to the primal integral, a performance measure that rewards a balance of convergence speed and solution quality. These results show that the proposed pattern-based initialization method significantly improves the search performance, allowing faster convergence and higher solution quality.

[1] As listed on the CVRPLIB website (http://vrp.atd-lab.inf.puc-rio.br) on December 16th, 2021.

[2] A solution for instance Loggi-n501-k24 with the same cost was listed in the results from the 12th DIMACS Implementation Challenge (http://dimacs.rutgers.edu/programs/challenge/vrp/cvrp/cvrp-competition) on January 23rd, 2022.

References

[1] L. Accorsi and D. Vigo, "A Fast and Scalable Heuristic for the Solution of Large-Scale Capacitated Vehicle Routing Problems", *Transportation Science* 55(4), 832-856, 2021.

[2] F. Arnold, M. Gendreau and K. Sörensen, "Efficiently solving very large-scale routing problems", *Computers & Operations Research* 107, 32-42, 2019.

[3] F. Arnold and K. Sörensen, "What makes a VRP solution good? The generation of problem-specific knowledge for heuristics", *Computers & Operations Research* 106, 280-288, 2019.

[4] T. Berthold, "Measuring the impact of primal heuristics", *Operations Research Letters* 41(6), 611-614, 2013.

[5] G. Clarke and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", *Operations Research* 12(4), 568-581, 1964.

[6] M. R. H. Maia, A. Plastino and P. H. V. Penna, "Hybrid data mining heuristics for the heterogeneous fleet vehicle routing problem", *RAIRO – Operations Research* 52(3), 661–690, 2018.

[7] A. Plastino, H. Barbalho, L. F. M. Santos, R. Fuchshuber and S. L. Martins, "Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic", *Journal of Heuristics* 20, 39–74, 2014.

[8] K. Sörensen, F. Arnold and D. P. Cuervo, "A critical analysis of the "improved Clarke and Wright savings algorithm"", *International Transactions in Operational Research* 26, 54–63, 2019.

[9] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal and A. Subramanian, "New benchmark instances for the Capacitated Vehicle Routing Problem", *European Journal of Operational Research* 257(3), 845-858, 2017.

[10] T. Vidal, "Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood", *Computers & Operations Research* 140, 105643, 2022.

[11] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi and W. Rei, "A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems", *Operations Research* 60(3), 611-624, 2012.

[12] T. Vidal, T. G. Crainic, M. Gendreau and C. Prins, "A unified solution framework for multi-attribute vehicle routing problems", *European Journal of Operational Research* 234(3), 658-673, 2014.

A Detailed Tables of Results

Winning values in the comparisons are highlighted in bold, best costs matching the corresponding BKS values are highlighted in italic, and new best solution costs are underlined.

Table 4: Comparison of HGS–CVRP and MDM–HGS over ten runs

Instance	BKS	HGS–CVRP				MDM–HGS			
		Avg PI	Avg Cost	Gap	Best Cost	Avg PI	Avg Cost	Gap	Best Cost
E-n101-k8	815	0.0007505	815.00	0.00%	*815*	**0.0004060**	815.00	0.00%	*815*
E-n101-k14	1067	0.0011573	1067.00	0.00%	*1067*	**0.0008523**	1067.00	0.00%	*1067*
CMT4	1028.42	0.0021390	1028.42	0.00%	*1028.42*	**0.0016326**	1028.42	0.00%	*1028.42*
CMT5	1291.29	0.0170969	1291.45	0.01%	1291.45	**0.0144392**	**1291.43**	0.01%	***1291.29***
F-n135-k7	1162	0.0012635	1162.00	0.00%	*1162*	**0.0003578**	1162.00	0.00%	*1162*
P-n101-k4	681	0.0007505	681.00	0.00%	*681*	**0.0002513**	681.00	0.00%	*681*
Tai385	24366.41	0.0190638	24368.14	0.01%	24367.91	**0.0152241**	**24367.84**	0.01%	***24366.41***
Golden_9	579.70	**0.0945582**	579.93	0.04%	*579.70*	0.1032073	**579.91**	0.04%	*579.70*
Golden_10	735.43	0.2880702	**736.77**	0.18%	***735.43***	**0.2842852**	737.04	0.22%	736.29
Golden_11	911.98	**0.2117253**	**913.12**	0.12%	912.69	0.2271350	913.25	0.14%	**912.61**
Golden_12	1101.24	0.3206600	1103.70	0.22%	1102.71	**0.2975944**	**1102.96**	0.16%	**1101.32**
Golden_13	857.19	**0.0187493**	857.19	0.00%	*857.19*	0.0275244	857.19	0.00%	*857.19*
Golden_14	1080.55	0.0091111	1080.55	0.00%	*1080.55*	**0.0062722**	1080.55	0.00%	*1080.55*
Golden_15	1337.27	0.1636654	1338.78	0.11%	**1337.73**	**0.1635862**	**1338.61**	0.10%	1338.19
Golden_16	1611.28	0.1592594	1612.52	0.08%	1611.79	**0.1478730**	**1612.31**	0.06%	***1611.28***
Golden_17	707.76	0.0011438	707.76	0.00%	*707.76*	**0.0007965**	707.76	0.00%	*707.76*
Golden_18	995.13	**0.0121521**	995.13	0.00%	*995.13*	0.0238004	995.13	0.00%	*995.13*
Golden_19	1365.60	0.0130377	**1365.60**	0.00%	*1365.60*	**0.0122402**	1365.61	0.00%	*1365.60*
Golden_20	1817.59	0.0367520	1817.96	0.02%	1817.64	**0.0312992**	**1817.82**	0.01%	***1817.59***
Antwerp1	477277	2.9081381	487770.30	2.20%	486959	**2.4651578**	**487178.40**	2.07%	**486497**
Antwerp2	291371	6.2112301	306085.20	5.05%	305190	**5.5196095**	**305692.30**	4.92%	**305039**
Brussels1	501734	6.8409837	528315.90	5.30%	527068	**4.4950502**	**521466.40**	3.93%	**520643**
Brussels2	345485	8.8157220	369502.50	6.95%	368166	**6.8585951**	**367347.20**	6.33%	**366817**
Flanders1	7240218	6.6227198	7576637.60	4.65%	7567917	**3.3106721**	**7412830.50**	2.38%	**7407580**
Flanders2	4373346	10.0000000	4979707.60	13.86%	4950737	**7.3120491**	**4616571.20**	5.56%	**4603333**
Ghent1	469532	4.5523259	486102.60	3.53%	485713	**3.6412099**	**482759.50**	2.82%	**482315**
Ghent2	257748	7.2426897	272874.40	5.87%	272165	**6.3879312**	**272738.60**	5.82%	**272117**
Leuven1	192848	1.9168784	**195232.00**	1.24%	**194813**	**1.8420827**	195391.80	1.32%	195102
Leuven2	111391	4.1023647	114801.30	3.06%	114458	**3.7980189**	**114687.00**	2.96%	**114412**
X-n101-k25	27591	0.0008475	27591.00	0.00%	*27591*	**0.0002881**	27591.00	0.00%	*27591*
X-n106-k14	26362	**0.0480004**	**26367.70**	0.02%	*26362*	0.0600124	26370.90	0.03%	*26362*
X-n110-k13	14971	0.0007182	14971.00	0.00%	*14971*	**0.0002805**	14971.00	0.00%	*14971*
X-n115-k10	12747	0.0010350	12747.00	0.00%	*12747*	**0.0002104**	12747.00	0.00%	*12747*

Table 5: Comparison of HGS–CVRP and MDM–HGS over ten runs (continued)

Instance	BKS	HGS–CVRP				MDM–HGS			
		Avg PI	Avg Cost	Gap	Best Cost	Avg PI	Avg Cost	Gap	Best Cost
X-n120-k6	13332	0.0017057	13332.00	0.00%	*13332*	**0.0005126**	13332.00	0.00%	*13332*
X-n125-k30	55539	0.0022937	55539.00	0.00%	*55539*	**0.0015046**	55539.00	0.00%	*55539*
X-n129-k18	28940	0.0021591	28940.00	0.00%	*28940*	**0.0012854**	28940.00	0.00%	*28940*
X-n134-k13	10916	0.0048763	10916.00	0.00%	*10916*	**0.0021764**	10916.00	0.00%	*10916*
X-n139-k10	13590	0.0011116	13590.00	0.00%	*13590*	**0.0006501**	13590.00	0.00%	*13590*
X-n143-k7	15700	0.0028865	15700.00	0.00%	*15700*	**0.0017933**	15700.00	0.00%	*15700*
X-n148-k46	43448	0.0022258	43448.00	0.00%	*43448*	**0.0012080**	43448.00	0.00%	*43448*
X-n153-k22	21220	**0.0266271**	**21224.70**	0.02%	21224	0.0286073	21224.90	0.02%	21224
X-n157-k13	16876	00016545	16876.00	0.00%	*16876*	**0.0006476**	16876.00	0.00%	*16876*
X-n162-k11	14138	0.0017371	14138.00	0.00%	*14138*	**0.0007897**	14138.00	0.00%	*14138*
X-n167-k10	20557	0.0046896	20557.00	0.00%	*20557*	**0.0032843**	20557.00	0.00%	*20557*
X-n172-k51	45607	0.0024970	45607.00	0.00%	*45607*	**0.0014888**	45607.00	0.00%	*45607*
X-n176-k26	47812	0.0050518	47812.00	0.00%	*47812*	**0.0040955**	47812.00	0.00%	*47812*
X-n181-k23	25569	0.0031308	25569.00	0.00%	*25569*	**0.0027489**	25569.00	0.00%	*25569*
X-n186-k15	24145	0.0037869	24145.00	0.00%	*24145*	**0.0024708**	24145.00	0.00%	*24145*
X-n190-k8	16980	0.0371058	16981.40	0.01%	*16980*	**0.0328346**	**16980.20**	0.00%	*16980*
X-n195-k51	44225	0.0052043	44225.00	0.00%	*44225*	**0.0035907**	44225.00	0.00%	*44225*
X-n200-k36	58578	0.0072826	58578.00	0.00%	*58578*	**0.0067785**	58578.00	0.00%	*58578*
X-n204-k19	19565	0.0020594	19565.00	0.00%	*19565*	**0.0014963**	19565.00	0.00%	*19565*
X-n209-k16	30656	0.0055620	30656.00	0.00%	*30656*	**0.0044601**	30656.00	0.00%	*30656*
X-n214-k11	10856	**0.0334471**	**10856.30**	0.00%	*10856*	0.0353976	10856.70	0.01%	*10856*
X-n219-k73	117595	0.0022935	117595.00	0.00%	*117595*	**0.0014856**	117595.00	0.00%	*117595*
X-n223-k34	40437	**0.0052572**	40437.00	0.00%	*40437*	0.0053860	40437.00	0.00%	*40437*
X-n228-k23	25742	0.0036312	25742.00	0.00%	*25742*	**0.0027064**	25742.00	0.00%	*25742*
X-n233-k16	19230	**0.0063406**	19230.00	0.00%	*19230*	0.0069371	19230.00	0.00%	*19230*
X-n237-k14	27042	0.0043749	27042.00	0.00%	*27042*	**0.0027764**	27042.00	0.00%	*27042*
X-n242-k48	82751	0.0446882	**82764.80**	0.02%	***82751***	**0.0444105**	82771.60	0.02%	82764
X-n247-k50	37274	0.0290335	37274.00	0.00%	*37274*	**0.0144882**	37274.00	0.00%	*37274*
X-n251-k28	38684	0.0205326	38684.00	0.00%	*38684*	**0.0155068**	38684.00	0.00%	*38684*
X-n256-k16	18839	0.0130645	18839.00	0.00%	*18839*	**0.0113000**	18839.00	0.00%	*18839*
X-n261-k13	26558	0.0233039	26558.10	0.00%	*26558*	**0.0167841**	**26558.00**	0.00%	*26558*
X-n266-k58	75478	0.1553469	**75553.80**	0.10%	***75478***	**0.1531489**	75557.80	0.11%	75517
X-n270-k35	35291	0.0378798	35303.00	0.03%	35303	**0.0373540**	35303.00	0.03%	35303
X-n275-k28	21245	0.0030218	21245.00	0.00%	*21245*	**0.0027920**	21245.00	0.00%	*21245*
X-n280-k17	33503	0.0781659	33510.00	0.02%	*33503*	**0.0452590**	**33503.70**	0.00%	*33503*
X-n284-k15	20215	0.1579628	20236.80	0.11%	20228	**0.1522369**	**20233.70**	0.09%	**20217**

Table 6: Comparison of HGS–CVRP and MDM–HGS over ten runs (continued)

Instance	BKS	HGS–CVRP				MDM–HGS			
		Avg PI	Avg Cost	Gap	Best Cost	Avg PI	Avg Cost	Gap	Best Cost
X-n289-k60	95151	0.1382099	95233.20	0.09%	95185	**0.1338773**	**95225.20**	0.08%	**95163**
X-n294-k50	47161	**0.0469679**	**47166.90**	0.01%	***47161***	0.0768137	47178.20	0.04%	47169
X-n298-k31	34231	0.0094907	34231.00	0.00%	*34231*	**0.0071716**	34231.00	0.00%	*34231*
X-n303-k21	21736	**0.0520567**	**21739.70**	0.02%	21738	0.0571379	21740.60	0.02%	21738
X-n308-k13	25859	**0.0469797**	25862.40	0.01%	25861	0.0533399	**25862.20**	0.01%	***25859***
X-n313-k71	94043	0.0616832	94073.90	0.03%	94044	**0.0538170**	**94052.70**	0.01%	94044
X-n317-k53	78355	0.0074638	78355.00	0.00%	*78355*	**0.0047830**	78355.00	0.00%	*78355*
X-n322-k28	29834	**0.0515693**	**29837.50**	0.01%	*29834*	0.0521901	29840.50	0.02%	*29834*
X-n327-k20	27532	0.0651764	27535.10	0.01%	*27532*	**0.0410109**	**27533.20**	0.00%	*27532*
X-n331-k15	31102	**0.0198349**	**31102.60**	0.00%	*31102*	0.0203486	31102.80	0.00%	*31102*
X-n336-k84	139111	0.1515611	139226.80	0.08%	139192	**0.1364896**	**139217.20**	0.08%	**139172**
X-n344-k43	42050	0.0626566	42060.80	0.03%	***42050***	**0.0506730**	**42059.20**	0.02%	42055
X-n351-k40	25896	0.2016053	25937.50	0.16%	**25921**	**0.1955979**	**25936.70**	0.16%	25930
X-n359-k29	51505	**0.2030894**	51568.00	0.12%	**51515**	0.2262600	**51553.80**	0.09%	51516
X-n367-k17	22814	0.0087575	22814.00	0.00%	*22814*	**0.0059225**	22814.00	0.00%	*22814*
X-n376-k94	147713	**0.0060703**	**147713.00**	0.00%	*147713*	0.0076381	147713.90	0.00%	*147713*
X-n384-k52	65928	0.1888654	66018.30	0.14%	65997	**0.1589981**	**65997.40**	0.11%	**65968**
X-n393-k38	38260	0.0139726	38260.00	0.00%	*38260*	**0.0111013**	38260.00	0.00%	*38260*
X-n401-k29	66154	0.1082049	66209.40	0.08%	66179	**0.0867611**	**66195.50**	0.06%	**66165**
X-n411-k19	19712	0.0389108	**19715.50**	0.02%	*19712*	**0.0376028**	19715.60	0.02%	*19712*
X-n420-k130	107798	**0.0295049**	**107815.50**	0.02%	107801	0.0321385	107818.50	0.02%	***107798***
X-n429-k61	65449	**0.0552796**	65472.10	0.04%	65457	0.0613447	**65469.40**	0.03%	**65455**
X-n439-k37	36391	**0.0168978**	**36394.60**	0.01%	*36391*	0.0206609	36396.00	0.01%	*36391*
X-n449-k29	55233	0.2160589	55308.80	0.14%	55288	**0.2138432**	**55308.40**	0.14%	**55251**
X-n459-k26	24139	**0.0639060**	**24141.70**	0.01%	*24139*	0.0770024	24146.70	0.03%	*24139*
X-n469-k138	221824	0.1502856	222043.60	0.10%	221956	**0.1296349**	**222006.80**	0.08%	**221841**
X-n480-k70	89449	0.0645461	89476.40	0.03%	89459	**0.0551488**	**89467.70**	0.02%	**89457**
X-n491-k59	66483	0.2180405	66584.70	0.15%	66523	**0.2063260**	**66568.50**	0.13%	**66512**
X-n502-k39	69226	0.0275632	69234.40	0.01%	69227	**0.0170000**	**69228.30**	0.00%	69227
X-n513-k21	24201	**0.0138413**	24201.00	0.00%	*24201*	0.0155519	24201.00	0.00%	*24201*
X-n524-k153	154593	**0.0595416**	**154632.90**	0.03%	154605	0.0776792	154654.40	0.04%	154605
X-n536-k96	94846	0.2390787	95032.90	0.20%	94977	**0.2137369**	**95002.80**	0.17%	**94930**
X-n548-k50	86700	0.0721160	86734.10	0.04%	86706	**0.0719223**	**86730.00**	0.03%	***86700***
X-n561-k42	42717	0.0480631	42723.80	0.02%	42719	**0.0347771**	**42723.00**	0.01%	42719
X-n573-k30	50673	**0.2297247**	**50761.30**	0.17%	50744	0.2376677	50764.20	0.18%	**50742**
X-n586-k159	190316	**0.1294136**	190470.00	0.08%	**190368**	0.1333633	**190449.30**	0.07%	190387

Table 7: Comparison of HGS–CVRP and MDM–HGS over ten runs (end)

Instance	BKS	HGS–CVRP				MDM–HGS			
		Avg PI	Avg Cost	Gap	Best Cost	Avg PI	Avg Cost	Gap	Best Cost
X-n599-k92	108451	0.1981210	108607.10	0.14%	108563	**0.1462474**	**108548.90**	0.09%	**108476**
X-n613-k62	59535	0.2549738	**59631.40**	0.16%	59576	**0.2136413**	59635.50	0.17%	**59561**
X-n627-k43	62164	**0.3246838**	62309.30	0.23%	62264	0.3252797	**62291.30**	0.20%	**62239**
X-n641-k35	63682	0.2993405	63815.20	0.21%	63732	**0.2251879**	**63759.40**	0.12%	**63709**
X-n655-k131	106780	**0.0299415**	**106794.80**	0.01%	*106780*	0.0305676	106798.90	0.02%	*106780*
X-n670-k130	146332	**0.3405979**	**146688.70**	0.24%	**146531**	0.3765850	146710.40	0.26%	146575
X-n685-k75	68205	**0.2494317**	68323.70	0.17%	68299	0.2672046	**68313.00**	0.16%	**68245**
X-n701-k44	81923	**0.3412277**	**82116.60**	0.24%	82038	0.3949137	82147.30	0.27%	**81995**
X-n716-k35	43373	**0.3249152**	43483.80	0.26%	43467	0.3281632	**43464.30**	0.21%	**43411**
X-n733-k159	136187	0.1932235	136372.90	0.14%	**136315**	**0.1735737**	**136345.10**	0.12%	136319
X-n749-k98	77269	**0.5295949**	**77594.10**	0.42%	**77513**	0.5572760	77597.50	0.43%	77516
X-n766-k71	114417	**0.3120103**	**114694.60**	0.24%	**114634**	0.3346094	114713.60	0.26%	114672
X-n783-k48	72386	0.6008088	72720.40	0.46%	72580	**0.4861709**	**72609.80**	0.31%	**72503**
X-n801-k40	73305	0.2799093	73430.50	0.17%	73374	**0.2636007**	**73390.00**	0.12%	**73311**
X-n819-k171	158121	0.2547253	158418.50	0.19%	158296	**0.2268259**	**158311.20**	0.12%	**158239**
X-n837-k142	193737	**0.3198199**	**194128.40**	0.20%	**193902**	0.3397571	194142.60	0.21%	194007
X-n856-k95	88965	**0.0903170**	**89013.40**	0.05%	88983	0.0942825	89026.40	0.07%	**88966**
X-n876-k59	99299	0.4500970	99597.10	0.30%	99484	**0.4368642**	**99589.30**	0.29%	**99441**
X-n895-k37	53860	0.5510356	54063.90	0.38%	53980	**0.4950917**	**54034.30**	0.32%	**53951**
X-n916-k207	329179	0.3122050	329782.40	0.18%	329653	**0.3054591**	**329735.40**	0.17%	**329547**
X-n936-k151	132715	**0.5774159**	**133342.00**	0.47%	133198	0.5897809	133360.00	0.49%	**133146**
X-n957-k87	85465	**0.1310368**	85524.60	0.07%	85502	0.1361873	**85523.10**	0.07%	**85479**
X-n979-k58	118976	0.3804908	**119186.30**	0.18%	119130	**0.3631164**	119188.20	0.18%	**119081**
X-n1001-k43	72355	0.6730433	72652.80	0.41%	72584	**0.6344107**	**72625.20**	0.37%	**72534**
Loggi-n401-k23	336946	0.0811768	337118.20	0.05%	337065	**0.0685315**	**337046.70**	0.03%	**336963**
Loggi-n501-k24	177466	0.0259036	177490.20	0.01%	**177428**[*]	**0.0248328**	**177484.40**	0.01%	177466
Loggi-n601-k19	113155	**0.1301022**	**113253.50**	0.09%	113181	0.1361894	113256.10	0.09%	**113174**
Loggi-n601-k42	347059	0.0216727	347061.50	0.00%	347052	**0.0103610**	**347058.90**	0.00%	**347046**
Loggi-n901-k42	246418	0.2081868	**246563.00**	0.06%	246441	**0.1877334**	246584.20	0.07%	**246360**
Loggi-n1001-k31	284356	0.6993173	285727.70	0.48%	**285362**	**0.6208397**	**285689.50**	0.47%	285521
ORTEC-n242-k12	123750	**0.0217995**	**123752.50**	0.00%	*123750*	0.0252933	123757.50	0.01%	*123750*
ORTEC-n323-k21	214071	**0.0334482**	**214088.40**	0.01%	*214071*	0.0359751	214096.40	0.01%	*214071*
ORTEC-n405-k18	200986	0.0048351	200986.00	0.00%	*200986*	**0.0034401**	200986.00	0.00%	*200986*
ORTEC-n455-k41	292516	**0.0315280**	**292544.80**	0.01%	*292516*	0.0526819	292571.00	0.02%	*292516*
ORTEC-n510-k23	184529	**0.1029263**	**184586.70**	0.03%	*184529*	0.2094845	184720.80	0.10%	*184529*
ORTEC-n701-k64	445592	**0.0647200**	**445676.00**	0.02%	**445541**	0.0994955	445824.10	0.05%	445601

[*]A solution for instance Loggi-n501-k24 with cost 177428 was listed in the results from the 12$^{\text{th}}$ DIMACS Implementation Challenge (http://dimacs.rutgers.edu/programs/challenge/vrp/cvrp/cvrp-competition) on January 23$^{\text{rd}}$, 2022.

APPENDIX F – MAIA, M. R. H.; PLASTINO, A.; FREITAS, A. A. “An Ensemble of Naive Bayes Classifiers for Uncertain Categorical Data”. In: 2021 IEEE International Conference on Data Mining (ICDM). 2021. p. 1222–1227[1]

[1] Available from: https://doi.org/10.1109/ICDM51629.2021.00148

2021 IEEE International Conference on Data Mining (ICDM)

An Ensemble of Naive Bayes Classifiers for Uncertain Categorical Data

Marcelo Rodrigues de Holanda Maia*†, Alexandre Plastino* and Alex A. Freitas‡
*Instituto de Computação
Universidade Federal Fluminense, Niterói, RJ, Brazil
E-mail: mmaia@ic.uff.br, plastino@ic.uff.br
†Instituto Brasileiro de Geografia e Estatística, Rio de Janeiro, RJ, Brazil
E-mail: marcelo.h.maia@ibge.gov.br
‡School of Computing
University of Kent, Canterbury, Kent, UK
E-mail: a.a.freitas@kent.ac.uk

2021 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-2398-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDM51629.2021.00148

***Abstract*—Coping with uncertainty is a very challenging issue in many real-world applications. However, conventional classification models usually assume there is no uncertainty in data at all. In order to fill this gap, there has been a growing number of studies addressing the problem of classification based on uncertain data. Although some methods resort to ignoring uncertainty or artificially removing it from data, it has been shown that predictive performance can be improved by actually incorporating information on uncertainty into classification models. This paper proposes an approach for building an ensemble of classifiers for uncertain categorical data based on biased random subspaces. Using Naive Bayes classifiers as base models, we have applied this approach to classify ageing-related genes based on real data, with uncertain features representing protein-protein interactions. Our experimental results show that models based on the proposed approach achieve better predictive performance than single Naive Bayes classifiers and conventional ensembles.**

***Keywords*-Classification, Ensemble, Uncertain data, Naive Bayes, Bioinformatics**

I. Introduction

Data uncertainty is a common issue in many real-world domains due to various reasons, including measurement errors, data staleness, repeated measurements, data generation and collection process. Coping with uncertainty is a challenging task in data mining applications since the reliability of the information used to build models significantly impacts their performance. However, conventional classification methods usually assume that data are precisely defined, effectively disregarding uncertainty.

In order to fill this gap, there has been a growing number of studies addressing the problem of classification based on uncertain data. Although some methods resort to ignoring uncertainty or artificially removing it from data, strategies for actually incorporating information on uncertainty into classification models have produced promising results, showing that this kind of approach can improve predictive performance [1]–[4].

In this paper, we propose a new approach for building an ensemble of classifiers tailored to cope with uncertainty in the values of categorical features. We rely on the hypothesis that the higher the degree of uncertainty for a given feature, the less it might contribute to the predictive performance as it provides less reliable information about the instances to be classified. We evaluate this proposed approach by applying it to build an ensemble of Naive Bayes classifiers.

Our experiments involve datasets of ageing-related genes containing uncertain features. Ageing can be defined as a progressive decline in the fitness of an organism that occurs with increasing age, ultimately ending in death. While it is unclear precisely what mechanisms drive ageing, genes certainly play an essential role in it [5]. Therefore, ageing genetics is an important subject in computational biology. In addition, ageing is a strategic research area because the proportion of elderly individuals among the population is increasing fast, and old age is the greatest risk factor for many diseases (including, e.g., most types of cancer).

The remainder of this paper is organized as follows. Section II reviews the background on the main topics covered in this work. In Section III, we introduce our proposed novel approach. Section IV describes our experimental methodology. Section V reports the results obtained. Finally, we present conclusions in Section VI.

II. Background

A. Ensemble Methods

In this work, we consider ensemble methods categorized as averaging methods. They build several base classifiers independently on random subsets of the original training set. Then, they aggregate the individual base classifiers' predictions to form a combined prediction.

These methods can be differentiated by how they draw random subsets of the original training set. In particular, Bagging methods are those that draw random subsets of the instances in the dataset with replacement [6], and if a method

This study was financed in part by: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil) [grant number 310444/2018-7]; Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES, Brazil) [finance code 001]; and Instituto Brasileiro de Geografia e Estatística (IBGE, Brazil).

978-1-6654-2398-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDM51629.2021.00148

draws random subsets of the features in the dataset, then it is known as Random Subspaces [7].

B. Naive Bayes Classifiers

Given a class variable y and a feature vector $X = (x_1, x_2, \ldots, x_m)$, based on the application of Bayes' theorem under the "naive" assumption that the features are conditionally independent given the value of the class variable, a Naive Bayes classifier predicts the class y that maximizes the approximation of $P(y|X)$ given by:

$$P(y|X) \propto P(y) \prod_{j=1}^{m} P(x_j|y) \quad (1)$$

We have chosen Naive Bayes as the base classifier to evaluate our proposed ensemble scheme since it has obtained good results in many real-world domains, including the classification of ageing-related genes [8]–[10] – the target application domain in this work. Furthermore, there are several reports on the use of Naive Bayes classifiers to build ensembles in the literature [11]–[13].

C. Classification with Uncertain Data

Data uncertainty is a common issue in many real-world applications due to various reasons, including measurement errors, data staleness, data generation and collection process. The forms of data uncertainty have been usually classified into existential uncertainty or value uncertainty.

Existential uncertainty refers to the case when it is uncertain whether an object exists. For example, an instance in a dataset could be associated with a probability representing the confidence of its occurrence.

In contrast, value uncertainty, which we address in this work, refers to the case when an instance is known to exist, but its feature values are not precisely known. An uncertain feature value is usually represented by a probability distribution on the domain of the feature.

Uncertainty in numerical feature values has been the focus of several studies, using multiple types of classification models such as neural networks [1], decision trees [2], k-nearest neighbors [3] and support vector machines [4]. Although some of those approaches could be adapted to cope with uncertain categorical features, this kind of value uncertainty has received relatively little attention in the literature. Another noticeable characteristic about past work in this field is that most experiments have been performed on data that were not originally uncertain. Instead, uncertainty was artificially introduced into the data through augmentation processes. In contrast, we evaluate our proposed approach using real-world data with uncertain categorical features.

III. Proposed Approach

The specification of the approach proposed in this work considers the following definitions.

Let $F = \{f_1, f_2, \ldots, f_m\}$ be the set of predictive features, where $m \geq 1$, and $C = \{c_1, c_2, \ldots, c_q\}$ be the set of classes, where $q \geq 2$. The domain of a feature f_j is $dom(f_j)$. A dataset $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$ consists of n labelled instances. Each instance in D, identified by an index i, is associated with a feature vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and a class label $y_i \in C$. The classification problem is to construct a model from D that is capable of predicting the class of an unlabelled instance given its corresponding feature vector.

Our uncertainty framework considers that some of the features are uncertain, i.e., there is a set of uncertain features $U \subseteq F$, all of which are assumed to be categorical. If f_j is a categorical feature, its domain is a finite set of values $dom(f_j) = \{v_{j1}, v_{j2}, \ldots, v_{j|dom(f_j)|}\}$, $|dom(f_j)| \geq 2$. If a feature f_j is not uncertain, its corresponding value x_{ij} for an instance i is represented by a single value v_{ij}. Otherwise it is a discrete probability distribution represented by a probability vector P_{ij}. That is:

$$x_{ij} = \begin{cases} v_{ij} \in dom(f_j), & \text{if } f_j \in F \setminus U \\ P_{ij} = (p_{ij1}, p_{ij2}, \ldots, p_{ij|dom(f_j)|}), & \text{otherwise} \end{cases}$$

where, if $f_j \in U$, $p_{ijk} \in [0, 1]$ represents the probability that x_{ij} assumes the value v_{jk} and $\sum_{k=1}^{|dom(f_j)|} p_{ijk} = 1$.

A. An Ensemble Approach for Coping with Uncertainty in Categorical Features

We propose a new approach for building an ensemble of classifiers that incorporates uncertainty about the value of categorical features into the model. The intuition motivating this proposal is that the higher the degree of uncertainty for a given feature, the less it might contribute to the predictive performance as it provides less reliable information about the instances to be classified. Furthermore, missing values, i.e., the absence of values for a feature in a dataset, are also considered since they represent another factor that may undermine the contribution of a feature to the model.

This approach relies on the use of a bias value computed for each feature f_j based on its degree of uncertainty and on its fraction of missing values in the dataset, given by:

$$b_j = \left(1 - \frac{1}{|I \setminus M_j|} \sum_{i \in I \setminus M_j} E_{ij}\right) \times \frac{|I \setminus M_j|}{|I|}$$

where $I = \{1, 2, \ldots, n\}$ is the set of indices of all instances in D, M_j is the set of indices of instances in D with a missing value for the feature f_j, and E_{ij} is the entropy of the probability distribution represented by P_{ij} if f_{ij} is an uncertain feature (or zero, otherwise), that is:

$$E_{ij} = \begin{cases} -\sum_{k=1}^{|dom(f_j)|} p_{ijk} log(p_{ijk}), & \text{if } f_j \in U \\ 0, & \text{otherwise} \end{cases}$$

In the feature bias definition, the first factor (between parentheses) is the complement of the mean entropy over

all probability distributions associated with the feature f_j, whereas the second factor is the fraction of non-missing values for the feature. Therefore, the computed bias is a value in the range $[0, 1]$ with higher values indicating lower uncertainty degrees, i.e., more reliable features.

The feature bias values are normalized over all features, defining a probability distribution $B = (\beta_1, \beta_2, \ldots, \beta_m)$, where a probability β_j associated with a feature f_j is given by $\beta_j = b_j/(\sum_{l=1}^{m} b_l)$.

Recall that in the general Random Subspaces strategy, each base classifier in the ensemble is trained with a different set of features, sampled from the full set F. In this approach, we use the probability distribution B to sample the features to be considered by each base classifier in the ensemble instead of the default uniform distribution. Hence, we call our proposed approach Biased Random Subspaces (BRS).

Note that no assumption is made about if or how the base classifiers in the ensemble handle uncertain data, as our focus is on the BRS approach to cope with uncertainty at the ensemble level. Even if the base classifiers do not cope with uncertainty, this approach can still be straightforwardly applied. As an example, it can be done by replacing each probability distribution P_{ij} corresponding to an uncertain feature in the dataset with its expected value, i.e., the value v_{jk} that maximizes p_{ijk}.

IV. Experimental Methodology

A. Dataset Creation

In this work, we have applied the proposed approach to the classification of ageing-related genes in different types of organisms, which several studies have addressed in recent years [8]–[10], [14]–[16]. In this problem, the objective is to identify the effect of genes on the longevity of an organism. More specifically, given an ageing-related gene, the problem is to predict whether its effect on the lifespan of an organism is positive (pro-longevity) or negative (anti-longevity).

The GenAge database, part of the Human Ageing Genomic Resources (HAGR) collection [17], is an essential resource in this context, comprising data about over two thousand genes, including their classification regarding longevity influence.

Genes encode proteins, and information about the interaction between two proteins, known as protein-protein interaction (PPI), has been used in past work addressing the classification of ageing-related genes [10], [14], [16], leading to improvements in predictive performance. STRING [18] is a database of PPIs that stem from computational predictions, from knowledge transfer between organisms, and from interactions aggregated from other databases.

Due to the technical difficulties of detecting PPIs via biological experiments, the available information on PPI is incomplete and exhibits varying levels of reliability. Furthermore, it is complemented with computational predictions (which are less reliable than biological experiments in general). Therefore, the STRING database provides a score for each PPI, computed by combining the probabilities from the different evidence channels, which means the available data on PPI are intrinsically uncertain.

We have generated four datasets[1] of ageing-related genes by integrating data from the GenAge database (Build 20) and the STRING database (Version 11.0). Each dataset contains data regarding ageing-related genes of one of the four major biomedical model organisms from the GenAge database: *C. elegans* (roundworm), *D. melanogaster* (fruit fly), *M. musculus* (mouse), and *S. cerevisiae* (baker's yeast).

Each instance in our datasets refers to an ageing-related gene of the corresponding model organism and consists of uncertain features referring to PPIs and a binary class variable indicating if the instance is positive (pro-longevity gene) or negative (anti-longevity gene) according to the GenAge database. Each PPI feature refers to one protein and has a binary domain, indicating whether or not an interaction between the protein encoded by the corresponding gene (the current instance) and the protein referred by the feature has been observed. Since these features are uncertain, they are represented by probability distributions according to our uncertainty framework.

A value x_{ij} for a PPI feature f_j corresponding to an instance i in the dataset is represented by a probability distribution $P_{ij} = (p_{ij1}, p_{ij2})$, where p_{ij1} and p_{ij2} are the complimentary probabilities of x_{ij} assuming each of the two values in $dom(f_j)$. Therefore, each probability distribution associated with a PPI feature value can actually be encoded by a single value p_{ij}, in which case $P_{ij} = (p_{ij}, 1 - p_{ij})$. In our datasets, this value is the confidence score obtained from the STRING database for the corresponding PPI, which indicates its probability of occurrence.

Some distinctive characteristics of these datasets, which make them quite challenging, are a very large number of features, a small number of instances, and a very high percentage of missing values (which occur when there is no information regarding a specific PPI in the STRING database). We have discarded PPI features with low support (annotating less than ten genes) to avoid overfitting.

Table I presents detailed information about the datasets. The first column indicates the corresponding model organism, whereas the remaining columns present, respectively, the number of instances, the number of features, the percentage of missing values, and the percentage of instances corresponding to each class (Anti- and Pro-longevity).

B. Algorithms Being Evaluated

In the experiments, we consider two baseline methods, both based on conventional Naive Bayes classifiers. Each of these methods uses a different interpretation of the uncertain feature values since they do not cope with uncertainty.

[1]The datasets used in the experiments are publicly available on the web at https://github.com/marcelorhmaia/ensembles-for-uncertain-data

Table I
INFORMATION ABOUT THE DATASETS USED IN THE EXPERIMENTS

Dataset	Instances	Features	Missing Values (%)	Class (%)	
				Anti	Pro
C. elegans	763	9692	93.8	66.3	33.7
D. melanogaster	185	3883	88.4	37.3	62.7
M. musculus	82	4216	78.4	37.8	62.2
S. cerevisiae	382	4274	90.3	88.0	12.0

The first baseline method (referred to as NB-NV) treats each uncertain value (the probability of occurrence of the corresponding PPI) as a numeric value. Therefore, it assumes that $dom(f_j) = [0, 1], \forall j \in \{1, 2, \ldots, m\}$ and that the feature value probability distributions are Gaussian.

The other baseline method (referred to as NB-EV) replaces each uncertain value with the expected value from the corresponding probability distribution, i.e., it binarizes the value representing the probability of occurrence of the corresponding PPI using the threshold value 0.5. It considers multivariate Bernoulli distributions for the data.

In both baseline methods, we replace missing values with zeros, i.e., if there is no information regarding a PPI in the dataset, we assume it does not occur, a case represented by a zero value, as usual in the literature using PPIs as predictive features for classifying genes.

For each baseline method, we build an ensemble that uses the conventional Bagging and Random Subspaces strategies (referred to as ENB-NV and ENB-EV, respectively) and another one that uses our proposed BRS approach in combination with conventional Bagging (ENB-NV+BRS and ENB-EV+BRS, respectively). Note that ENB-NV and ENB-EV do not cope with uncertainty, but only the proposed ENB-NV+BRS and ENB-EV+BRS ensembles do so.

We have used available implementations from the scikit-learn library [19] as the baseline methods, as well as for the conventional ensembles. The algorithms based on our proposed approaches have been implemented through the extension of scikit-learn's original methods[2]. We have set the ensembles to use 500 base classifiers, building each one on a different subset of the training set consisting of n instances drawn by the Bagging procedure and $\sqrt{m}$ features drawn by the Random Subspaces (or the BRS) procedure.

We first separate the experiments into two groups (consisting of methods based on the NV and EV approaches), and then compare three algorithms in each group:

- a single Naive Bayes classifier (one of the two baseline methods)
- a conventional ensemble of this base classifier
- another ensemble that uses our proposed BRS approach

Then we compare the best models from the two groups.

[2]The source-code used in the experiments is publicly available on the web at https://github.com/marcelorhmaia/ensembles-for-uncertain-data

C. Measuring Predictive Performance

We assess the predictive performance of the evaluated algorithms using two metrics: the Area Under the Receiver Operating Characteristic curve (AUROC) and the geometric mean of sensitivity and specificity.

The ROC curve is a method for evaluating the performance of a binary classifier by plotting its true-positive rate (sensitivity) versus its false-positive rate (one minus the specificity) at various threshold settings. The AUROC summarizes this information into one number.

The geometric mean of sensitivity and specificity (G-mean) measures the balance between predictive performances on both the majority and minority classes. Therefore, it is suitable for assessing predictive performance on imbalanced datasets, like the ones used in our experiments.

Each algorithm was evaluated by running a well-known 10-fold cross-validation procedure. In addition, we have assessed the statistical significance of the differences in the predictive performance measures between each pair of algorithms. We have used a paired Wilcoxon signed-rank test for each dataset for this evaluation, considering a significance level of 0.05.

V. COMPUTATIONAL RESULTS

In the first experiment, we have compared the three models based on NB-NV. Table II presents the specificity (Spec.) and sensitivity (Sens.) values obtained by each model, as these measures are used to compute predictive performance metrics. Specificity and sensitivity correspond to the recall values for the Anti-longevity and Pro-longevity classes, respectively.

Table II
SPECIFICITY AND SENSITIVITY VALUES (%) FOR MODELS BASED ON NB-NV

Dataset	NB-NV		ENB-NV		ENB-NV+BRS	
	Spec.	Sens.	Spec.	Sens.	Spec.	Sens.
C. elegans	70.54	56.61	35.54	94.47	42.30	87.47
D. melanogaster	36.19	85.12	76.65	42.85	80.23	44.56
M. musculus	46.83	86.45	64.33	50.67	64.33	49.24
S. cerevisiae	99.69	0.00	50.68	65.83	40.81	85.00

Tables III and IV present the results for this group of models regarding the AUROC and G-mean metrics, respectively. Besides the predictive performance values achieved by each model on each dataset, these tables present the average rank for each model in the last row. The best value in each comparison is presented in bold. Furthermore, the statistically significant differences in the comparisons between the model using our proposed BRS approach and each of the other models are indicated by superscript symbols next to the respective higher values.

Based on the results presented in Tables III and IV, the general conclusion from this first experiment is that ENB-

Table III
AUROC RESULTS (%) FOR MODELS BASED ON NB-NV

Dataset	NB-NV	ENB-NV	ENB-NV+BRS
C. elegans	63.52	71.46	**72.33**[a]
D. melanogaster	60.83	**65.62**	65.03
M. musculus	67.93	66.73	**68.51**
S. cerevisiae	49.84	**61.62**	61.22[a]
Avg. Rank	2.75	1.75	**1.50**

[a]Statistically significant (NB-NV vs. ENB-NV+BRS), p-value = 0.003 for *C. elegans*, p-value = 0.004 for *S. cerevisiae*.

Table IV
G-MEAN RESULTS (%) FOR MODELS BASED ON NB-NV

Dataset	NB-NV	ENB-NV	ENB-NV+BRS
C. elegans	**63.19**	57.94	60.83[b]
D. melanogaster	55.50	57.31	**59.79**
M. musculus	**63.63**	57.09	56.28
S. cerevisiae	0.00	57.76	**58.90**[a]
Avg. Rank	2.00	2.25	**1.75**

[a]Statistically significant (NB-NV vs. ENB-NV+BRS), p-value = 0.003.
[b]Statistically significant (ENB-NV vs. ENB-NV+BRS), p-value = 0.033.

NV+BRS is the best model regarding both AUROC and G-mean, with the average ranks of 1.50 and 1.75, respectively.

In the second experiment, we have compared the three models based on NB-EV. Table V presents the specificity and sensitivity values obtained by each model, whereas Tables VI and VII present the results for this group of models regarding the AUROC and G-mean metrics, respectively.

Table V
SPECIFICITY AND SENSITIVITY VALUES (%) FOR MODELS BASED ON NB-EV

Dataset	NB-EV		ENB-EV		ENB-EV+BRS	
	Spec.	Sens.	Spec.	Sens.	Spec.	Sens.
C. elegans	74.99	58.42	96.44	13.64	90.27	26.89
D. melanogaster	26.91	86.26	8.33	96.09	8.33	94.55
M. musculus	34.83	85.14	18.67	92.64	23.67	92.64
S. cerevisiae	85.56	36.67	98.83	5.00	94.45	24.17

Table VI
AUROC RESULTS (%) FOR MODELS BASED ON NB-EV

Dataset	NB-EV	ENB-EV	ENB-EV+BRS
C. elegans	76.89[a]	**76.91**[b]	74.81
D. melanogaster	66.65	64.05	**69.00**
M. musculus	66.87	69.26	**69.30**
S. cerevisiae	**77.13**	75.55	76.79
Avg. Rank	2.00	2.25	**1.75**

[a]Statistically significant (NB-EV vs. ENB-EV+BRS), p-value = 0.012.
[b]Statistically significant (ENB-EV vs. ENB-EV+BRS), p-value = 0.033.

Based on the results presented in Tables VI and VII, the general conclusion from the second experiment is that

Table VII
G-MEAN RESULTS (%) FOR MODELS BASED ON NB-EV

Dataset	NB-EV	ENB-EV	ENB-EV+BRS
C. elegans	**66.19**[a]	36.27	49.27[b]
D. melanogaster	**48.18**[a]	28.28	28.06
M. musculus	**54.46**	41.59	46.82
S. cerevisiae	**56.01**	22.23	47.78[b]
Avg. Rank	**1.00**	2.75	2.25

[a]Statistically significant (NB-EV vs. ENB-EV+BRS), p-value = 0.003 for *C. elegans*, p-value = 0.010 for *D. melanogaster*.
[b]Statistically significant (ENB-EV vs. ENB-EV+BRS), p-value = 0.005 for *C. elegans*, p-value = 0.017 for *S. cerevisiae*.

the proposed ENB-EV+BRS is the best model regarding AUROC and the second best (out of three) regarding G-mean, with the average ranks of 1.75 and 2.25, respectively. In this experiment, a single NB-EV classifier performed better than both ensembles regarding G-mean. Nonetheless, it is noticeable that our proposed BRS approach was still able to improve the predictive performance of an ensemble, as the ENB-EV+BRS model outperformed the ENB-EV.

Our third experiment aims at determining the best overall method regarding the AUROC metric. Table VIII presents the results for ENB-NV+BRS and ENB-EV+BRS, the best models from experiments 1 and 2, respectively, regarding this metric. The proposed ENB-EV+BRS was the best overall model in this comparison, outperforming the ENB-NV+BRS for all datasets.

Table VIII
AUROC RESULTS (%) FOR THE BEST MODEL FROM TABLE III AND THE BEST MODEL FROM TABLE VI

Dataset	ENB-NV+BRS[a]	ENB-EV+BRS[b]
C. elegans	72.33	**74.81**
D. melanogaster	65.03	**69.00**
M. musculus	68.51	**69.30**
S. cerevisiae	61.22	**76.79***
Avg. Rank	2.00	**1.00**

[a]Results from Table III. [b]Results from Table VI.
*Statistically significant, p-value = 0.012.

Finally, our fourth experiment aims at determining the best overall method regarding the G-mean metric. Table IX presents the results for ENB-NV+BRS and NB-EV, the best models from experiments 1 and 2, respectively, regarding this metric. ENB-NV+BRS was the best overall model in this comparison, with an average rank of 1.25.

As a general conclusion from the reported experiments, we can point out that the results support the hypothesis that our proposed BRS approach improves the predictive performance of ensembles on uncertain data, as the best overall models for the AUROC and G-mean metrics were the ENB-NV+BRS and ENB-EV+BRS, respectively, both based on the BRS approach.

Table IX
G-MEAN RESULTS (%) FOR THE BEST MODEL FROM TABLE IV AND THE BEST MODEL FROM TABLE VII

Dataset	ENB-NV+BRS[a]	NB-EV[b]
C. elegans	60.83	**66.19**
D. melanogaster	**59.79**[*]	48.18
M. musculus	**56.28**	54.46
S. cerevisiae	**58.90**	56.01
Avg. Rank	**1.25**	1.75

[a]Results from Table IV. [b]Results from Table VII.
[*]Statistically significant, p-value = 0.038.

VI. CONCLUSIONS

In this work, we have addressed the problem of classification in datasets containing categorical features with uncertain values, i.e., values represented by probability distributions in the respective features' domains. We have proposed an ensemble approach called Biased Random Subspaces (BRS) for coping with this kind of uncertainty, based on the hypothesis that features with lower uncertainty degrees have better class-discrimination power since there is higher confidence about their actual values across the dataset.

Our experiments have compared two types of single Naive Bayes classifiers, conventional ensembles of these classifiers and ensembles based on the BRS approach. We have applied them to classify ageing-related genes from four model organisms based on real data containing uncertain features referring to protein-protein interactions. The results show that the ensembles applying our BRS approach achieved the best overall predictive performance, supporting the hypothesis that applying BRS-based ensembles of classifiers is an effective approach to cope with uncertainty in categorical features, leading to higher predictive performance.

Some directions for future work include applying the proposed BRS approach to build ensembles of other base classifiers than Naive Bayes and perform experiments with other datasets containing uncertain data.

REFERENCES

[1] J. Ge, Y. Xia, and C. Nadungodage, "UNN: A neural network for uncertain data classification," in *Advances in Knowledge Discovery and Data Mining*, 2010, pp. 449–460.

[2] S. Tsang, B. Kao, K. Y. Yip, W.-S. Ho, and S. D. Lee, "Decision trees for uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 1, pp. 64–78, 2011.

[3] F. Angiulli and F. Fassetti, "Nearest neighbor-based classification of uncertain data," *ACM Trans. Knowl. Discov. Data*, vol. 7, no. 1, 2013.

[4] Z. Xie, Y. Xu, and Q. Hu, "Uncertain data classification with additive kernel support vector machine," *Data Knowl. Eng.*, vol. 117, pp. 87–97, 2018.

[5] D. Wieser, I. Papatheodorou, M. Ziehm, and J. M. Thornton, "Computational biology for ageing," *Philos. Trans. Royal Soc. B: Biol. Sci.*, vol. 366, no. 1561, pp. 51–63, 2011.

[6] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[7] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[8] C. Wan and A. Freitas, "Prediction of the pro-longevity or anti-longevity effect of caenorhabditis elegans genes based on bayesian classification methods," in *IEEE Int. Conf. on Bioinformatics and Biomedicine*, 2013, pp. 373–380.

[9] P. N. da Silva, A. Plastino, and A. A. Freitas, "A novel genetic algorithm for feature selection in hierarchical feature spaces," in *SIAM Int. Conf. on Data Mining*, 2018, pp. 738–746.

[10] P. N. da Silva, A. Plastino, F. Fabris, and A. A. Freitas, "A novel feature selection method for uncertain features: An application to the prediction of pro-/anti- longevity genes," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2020.

[11] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Mach. Learn.*, vol. 36, no. 1, pp. 105–139, 1999.

[12] A. Tsymbal, S. Puuronen, and D. W. Patterson, "Ensemble feature selection with the simple bayesian classification," *Inf. Fusion*, vol. 4, no. 2, pp. 87–100, 2003.

[13] A. Prinzie and D. Van den Poel, "Random multiclass classification: Generalizing random forests to random MNL and random NB," in *Database and Expert Systems Applications*, 2007, pp. 349–358.

[14] T. Huang et al., "Deciphering the effects of gene deletion on yeast longevity using network and machine learning approaches," *Biochim.*, vol. 94, no. 4, pp. 1017–1025, 2012.

[15] C. Wan, A. A. Freitas, and J. P. de Magalhães, "Predicting the pro-longevity or anti-longevity effect of model organism genes with new hierarchical feature selection methods," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 12, no. 2, pp. 262–275, 2015.

[16] I. Martire, P. N. da Silva, A. Plastino, F. Fabris, and A. A. Freitas, "A novel probabilistic Jaccard distance measure for classification of sparse and uncertain data," in *5th Symp. on Knowledge Discovery, Mining and Learning*, 2017, pp. 81–88.

[17] R. Tacutu et al., "Human Ageing Genomic Resources: new and updated databases," *Nucleic Acids Res.*, vol. 46, no. D1, pp. D1083–D1090, 2017.

[18] D. Szklarczyk et al., "STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Res.*, vol. 47, no. D1, pp. D607–D613, 2018.

[19] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

APPENDIX G – MAIA, M. R. H.; PLASTINO, A.; FREITAS, A. A.; MAGALHÃES, J. P. “Interpretable Ensembles of Classifiers for Uncertain Data with Bioinformatics Applications”. Submitted to IEEE/ACM Transactions on Computational Biology and Bioinformatics

Interpretable Ensembles of Classifiers for Uncertain Data with Bioinformatics Applications

Marcelo Rodrigues de Holanda Maia, Alexandre Plastino, Alex A. Freitas, and João Pedro de Magalhães

Abstract—Data uncertainty remains a challenging issue in many applications, but few classification algorithms can effectively cope with it. An ensemble approach for uncertain categorical features has recently been proposed, achieving promising results. It consists in biasing the sampling of features for each model in an ensemble so that less uncertain features are more likely to be sampled. Here we extend this idea of biased sampling and propose two new approaches: one for selecting training instances for each model in an ensemble and another for sampling features to be considered when splitting a node in a Random Forest training. We applied these approaches to classify ageing-related genes and predict drugs' side effects based on uncertain features representing protein-protein and protein-chemical interactions. We show that ensembles based on our proposed approaches achieve better predictive performance than conventional ensembles. Furthermore, we propose two approaches for interpreting an ensemble of Naive Bayes classifiers and analyse their results for our ageing-related datasets.

Index Terms—Classification, interpretability, uncertainty, bioinformatics, the biology of ageing.

1 Introduction

Data uncertainty can be categorised into existential uncertainty, which occurs when the existence of some data record is uncertain, and value uncertainty, which can be further categorised into class-label uncertainty or feature-value uncertainty. This work addresses feature-value uncertainty, which occurs when some feature values in a data record (instance) are not precisely known. This uncertainty can naturally arise due to the limited precision of data collection technology, particularly in bioinformatics or biomedical domains. An uncertain feature value is usually represented by a probability distribution on the corresponding feature's domain.

It has been shown that incorporating information on uncertainty into classification algorithms can improve predictive performance [1], [2], [3], [4], [5], but this is still an underexplored research topic, particularly for categorical features, since most previous methods focus on uncertain numerical features. Hence, this work proposes new ensemble methods for coping with uncertain categorical features.

We focus on ensemble methods that learn many base classifiers independently on random subsets of the original training set and then aggregate the base classifiers' predictions. In general, such ensemble methods usually have better predictive performance and are more robust to slight data variations than any single base classifier. In particular, Bagging methods randomly sample subsets of the instances in the dataset with replacement (which is called bootstrap sampling) [6], and Random Subspaces methods randomly sample subsets of the features in the dataset [7].

An ensemble approach for uncertain categorical features, named Biased Random Subspaces (BRS), has recently been proposed by Maia et al. [3]. It consists in biasing the random sampling of features for each model in an ensemble, based on the principle that features with lower uncertainty degrees should have better class-discrimination potential since the confidence about their actual values is higher.

Relying on the same hypothesis, this work extends the idea of biased random sampling by proposing two new approaches for building ensembles of classifiers that cope with uncertain categorical features. The first is a Biased Bootstrap (BB) approach for selecting training instances for each model in an ensemble. The second is a Biased Splitting (BS) approach for sampling features to be considered when splitting a node while building the trees of a Random Forest.

We evaluate our proposed approaches by using them to build Naive Bayes (NB) classifiers ensembles and Random Forests, and performing experiments on ten classification datasets with real uncertain information. This uncertainty consists of feature values' probability distributions extracted from real-world databases – unlike previous work, which typically used datasets with artificially generated uncertainty [1], [2], [4], [5].

Out of these 10 datasets, 4 were also used in [3]. In these datasets, each instance is an ageing-related gene, the class labels indicate whether a gene has a pro-longevity or anti-longevity effect on a particular organism's lifespan, as recorded in the GenAge database [8], and the features represent protein-protein interaction (PPI) information. The feature uncertainty is represented by probabilities of interactions between two proteins, obtained from the STRING database [9].

The other 6 datasets are introduced in this work. In these datasets, each instance is a drug (chemical), the class labels indicate whether or not a drug has a particular side effect,

- *M.R.H. Maia is with Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil and Instituto Brasileiro de Geografia e Estatística, Rio de Janeiro, Brazil.*
  E-mail: mmaia@ic.uff.br; marcelo.h.maia@ibge.gov.br
- *A. Plastino is with Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil*
- *A.A. Freitas is with the School of Computing, University of Kent, Canterbury, UK*
- *J.P. de Magalhães is with the Integrative Genomics of Ageing Group, University of Liverpool, Liverpool, UK*

as recorded in the SIDER database [10], and the features represent protein-chemical interaction (PCI) information. The feature uncertainty is represented by probabilities of interactions between a chemical (drug) and a protein, obtained from the STITCH database [11].

We report two types of results. First, we compare the predictive performance of the ensemble methods using the aforementioned uncertainty-management approaches (BRS, BB and BS) against the accuracy of the corresponding standard ensemble methods (without the BRS, BB and BS approaches), using two well-known predictive performance measures: the Area Under the ROC curve (AUROC) and the geometric mean of Sensitivity and Specificity [12]. In general, the proposed uncertainty-aware ensemble methods outperformed the baseline uncertainty-unaware methods.

Second, we report the results of interpreting the best ensemble models learned from the ageing-related datasets. Model interpretation is an increasingly important topic in machine learning [13], and it has been used to derive novel biological insights in bioinformatics domains [14], [15]. Although a single NB classifier is naturally interpretable, interpreting an ensemble of NB classifiers is not trivial. Hence, we propose two approaches for interpreting an ensemble of NB classifiers (computing feature importance measures) as an additional contribution. The first measures the importance of a feature based on conditional probability differences, whereas the second is a more sophisticated approach based on finding the minimal set of features that is sufficient to preserve the class predicted for an instance (so that changes to the values of other features in that instance do not change the class predicted by the model). We use these two interpretation approaches for NB ensembles and a conventional feature importance measure for random forests to learn feature rankings for the ageing-related datasets, identifying the most important features for predicting such genes' effects on an organism's lifespan.

In summary, this paper extends the initial work from [3] by providing four new contributions. First, we propose two new approaches – Biased Bootstrap (BB) and Biased Splitting (BS) – for learning from uncertain categorical features, which complement the Biased Random Subspace (BRS) one introduced in [3]. Second, in [3] the BRS approach was used to create NB ensembles only; whilst in this paper, we create several types of ensembles, including NB ensembles with the BB approach and random forests with the BS and BB approaches. Third, the experiments in [3] used only 4 datasets of ageing-related genes, whilst in this paper, we use 10 datasets: those 4 datasets and 6 new datasets for predicting drugs' side effects. Fourthly, this paper introduces two approaches for interpreting an ensemble of Naive Bayes classifiers, whilst no such interpretation was attempted in [3]. We also use these interpretation approaches to analyse the best models learned from the ageing-related datasets, discussing the results from the perspective of the biology of ageing as an interdisciplinary contribution.

2 METHODS

2.1 Definitions

Let $F = \{f_1, f_2, \ldots, f_m\}$ be the set of predictive features, where $m \geq 1$, and $C = \{c_1, c_2, \ldots, c_q\}$ be the set of classes, where $q \geq 2$. The domain of a feature f_j is $dom(f_j)$. A dataset $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$ consists of n labelled instances. Each instance in D, identified by an index i, is associated with a feature vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and a class label $y_i \in C$. In the classification problem, the objective is to construct a model from D capable of predicting the class of an unlabelled instance given its corresponding feature vector.

Let $U \subseteq F$ be the set of uncertain features, all of which are assumed to be categorical in this work. If f_j is a categorical feature, its domain is a finite set of values $dom(f_j) = \{v_{j1}, v_{j2}, \ldots, v_{j|dom(f_j)|}\}$, $|dom(f_j)| \geq 2$. If a feature f_j is not uncertain, its corresponding value x_{ij} for an instance i is represented by a single value. Otherwise, it is a discrete probability distribution represented by a probability vector P_{ij}. That is:

$$x_{ij} = \begin{cases} x_{ij} \in dom(f_j), & \text{if } f_j \in F \setminus U \\ P_{ij} = (p_{ij1}, p_{ij2}, \ldots, p_{ij|dom(f_j)|}), & \text{otherwise} \end{cases}$$

where, if $f_j \in U$, $p_{ijk} \in [0, 1]$ represents the probability that x_{ij} takes the value v_{jk} and $\sum_{k=1}^{|dom(f_j)|} p_{ijk} = 1$.

2.2 Coping with uncertainty in ensemble models

Recently, an ensemble approach for coping with uncertainty in categorical features, named Biased Random Subspaces (BRS), has been proposed by Maia et al. [3]. It consists in biasing the random sampling of features for each model in an ensemble. Here we extend the idea of biased random sampling to two new approaches: a Biased Bootstrap (BB) approach for selecting training instances for each model in an ensemble (which can be used with any bagging-based ensemble algorithm) and a Biased Splitting (BS) approach for sampling features to be considered when splitting a node while building the trees of a Random Forest.

From [3], we have the definition of the bias value for a feature f_j, given by:

$$b_{*j} = \left(1 - \frac{1}{|I \setminus M_{*j}|} \sum_{i \in I \setminus M_{*j}} E_{ij}\right) \times \frac{|I \setminus M_{*j}|}{|I|}$$

where $I = \{1, 2, \ldots, n\}$ is the set of indices of all instances in D, M_{*j} is the set of indices of instances in D with a missing value for the feature f_j, and E_{ij} is the normalized entropy of the probability distribution represented by P_{ij} if f_{ij} is an uncertain feature (or zero, otherwise), that is:

$$E_{ij} = \begin{cases} \frac{\sum_{k=1}^{|dom(f_j)|} p_{ijk} log(p_{ijk})}{log(1/|dom(f_j)|)}, & \text{if } f_j \in U \\ 0, & \text{otherwise} \end{cases}$$

The feature bias values are normalized over all features, defining a probability distribution $B = (\beta_1, \beta_2, \ldots, \beta_m)$, where a probability β_j associated with a feature f_j is given by $\beta_j = b_{*j} / (\sum_{l=1}^{m} b_{*l})$.

Instead of the default uniform distribution from the general Random Subspaces strategy, the BRS approach uses the probability distribution B to sample the features to train each base classifier in the ensemble.

Random Forests usually do not sample features before generating each tree. Nonetheless, they sample a subset

of features to be considered as candidate features when splitting each node of a tree. Hence, we propose the Biased Splitting (BS) approach for this sampling, which uses the probability distribution B to sample the candidate features.

Finally, we propose the Biased Bootstrap (BB) approach for instance sampling, which is analogous to the BRS and BS aproaches for feature sampling. Hence, we define the bias value for an instance identified by index i, given by:

$$b_{i*} = \left(1 - \frac{1}{|F \setminus M_{i*}|} \sum_{f_j \in F \setminus M_{i*}} E_{ij}\right) \times \frac{|F \setminus M_{i*}|}{|F|}$$

where M_{i*} is the set of features in D with a missing value for instance i.

The instance bias values are normalized over all instances, defining a distribution $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$, where a probability γ_i associated with an instance identified by index i is given by $\gamma_i = b_{i*}/(\sum_{l=1}^{n} b_{l*})$.

The BB approach uses probability distribution Γ to sample the instances to be used for training each base classifier.

2.3 Interpreting NB ensembles via conditional probabilities

The first approach we propose for interpreting an ensemble of NB classifiers relies on the influence that a feature value $x_{ij} \in dom(f_j)$ has for determining the most likely class to be predicted for an instance by a single NB classifier, which we compute as an importance score. We then combine the importance scores obtained from all the classifiers into the ensemble's importance scores.

This approach does not address feature uncertainty. It assumes the base models of the ensemble are standard NB classifiers. Therefore, in this context, x_{ij} is always represented by a single value.

Given a feature vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ associated with an unlabelled instance identified by index i, a NB classifier predicts the class $y \in C$ that maximizes the value given by $P(y|X_i) \propto P(y) \prod_{j=1}^{m} P(x_{ij}|y)$.

We first present our definition of importance for binary classifiers, where $C = \{c_1, c_2\}$. The importance of a feature value x_{ij} in a given NB classifier is estimated by the following difference of conditional probabilities:

$$Diff(x_{ij}, c_1, c_2, e) = |P(x_{ij}|c_1) - P(x_{ij}|c_2)|$$

where e is the classifier for which the difference is computed. Clearly, the higher the difference in the class-conditioned probability of a feature value between the two class labels, the more importance (influence) that feature value will have for determining the most likely class to be assigned to the testing instance.

For datasets with more than two class labels, this idea can be generalised by summing the differences between all pairs of class labels in C:

$$Importance(x_{ij}, C, e) = \sum_{r=1}^{q-1} \sum_{s=r+1}^{q} Diff(x_{ij}, c_r, c_s, e)$$

The importance of a feature value x_{ij} for an ensemble of NB classifiers is computed by averaging its importance across all classifiers in the ensemble. However, different NB classifiers will generally use different feature subsets (due to the random subspaces approach). Intuitively, other things being equal, the larger the number of classifiers in the ensemble that use a feature, the larger the importance of a value of that feature. Therefore, we assume that, if a classifier e_u does not use a feature f_j, then $Importance(x_{ij}, C, e_u) = 0$. Hence, we define the ensemble-wide importance as:

$$Importance(x_{ij}, C) = \frac{\sum_{u=1}^{t} Importance(x_{ij}, C, e_u)}{t}$$

where e_u is the u-th classifier and t is the total number of classifiers in the ensemble.

This equation is appropriate when the predicted class returned by the ensemble is computed by a simple majority vote of all classifiers, i.e., all classifiers have the same weight in the voting. If weighted voting is used instead (where the weight of a vote is proportional to the classifier's confidence in its prediction), then the importance equation could be easily modified to compute a correspondingly weighted average over the t classifiers.

Finally, once the importance value has been computed for all feature values x_{ij}, we rank all feature values in decreasing order of importance. Then a user (domain expert) can focus on interpreting the top-ranked feature values, i.e., the most important ones for predicting the class variable in the ensemble.

Note that for binary domain features, where $dom(f_j) = \{v_{j1}, v_{j2}\}$, the importance computed for both values will be the same due to the complementarity of probabilities in use. Given two class labels c_r and c_s such that $c_r \in C$, $c_s \in C$ and $c_r \neq c_s$, the following relations apply:

$$P(v_{j1}|c_r) = 1 - P(v_{j2}|c_r) \quad (1)$$

$$P(v_{j1}|c_s) = 1 - P(v_{j2}|c_s) \quad (2)$$

The difference of the class-conditioned probabilities of the value v_{j1} between c_r and c_s for a classifier e would be:

$$Diff(v_{j1}, c_r, c_s, e) = |P(v_{j1}|c_r) - P(v_{j1}|c_s)| \quad (3)$$

By replacing (1) and (2) in (3), we obtain:

$$\begin{aligned} Diff(v_{j1}, c_r, c_s, e) &= |(1 - P(v_{j2}|c_r)) - (1 - P(v_{j2}|c_s))| \\ &= |1 - P(v_{j2}|c_r) - 1 + P(v_{j2}|c_s)| \\ &= |P(v_{j2}|c_s) - P(v_{j2}|c_r)| \\ &= |P(v_{j2}|c_r) - P(v_{j2}|c_s)| \\ &= Diff(v_{j2}, c_r, c_s, e) \end{aligned}$$

Therefore, for binary domain features (like the PPI and PCI features in our datasets), the importance computed for both values in the domain is the same. Then, the importance computed for any of the values in a feature's domain can be interpreted as that feature's importance.

2.4 Interpreting NB ensembles via minimal sufficient features

The approach based on conditional probability differences measures the importance of each feature value separately, ignoring its importance in the context of all other feature values. This is consistent with NB assuming that each

Algorithm 1. MinimalSufficientSet(X_i, e)
1: $c_i \leftarrow$ class predicted by e for X_i
2: $SuppSet \leftarrow \{f_j | P(x_{ij}|c_i) > P(x_{ij}|y), \forall y \in C \setminus \{c_i\}\}$
3: $S \leftarrow SuppSet$
4: Calculate $Importance(x_{ij}, C, e)$ for all $f_j \in S$
5: $SortedFeats \leftarrow$ SortByImportance($SuppSet$)
6: $e' \leftarrow e$
7: **for each** feature f_j in $SortedFeats$ **do**
8: Remove f_j from e'
9: $c_i' \leftarrow$ class predicted by e' for X_i
10: **if** $c_i' = c_i$ **then**
11: $S \leftarrow S \setminus \{f_j\}$
12: **else**
13: Exit loop
14: **end if**
15: **end for**
16: **return** S

feature is independent of all others conditioned on the class variable, but it does not directly measure the influence of a feature value on class prediction. Naive Bayes makes class predictions using the formula: $P(y|X_i) \propto P(y)\prod_{j=1}^{m} P(x_{ij}|y)$. Therefore, whether or not a conditional probability will make a difference in the choice of the predicted class depends on the entire set of conditional probabilities and the prior class probability.

Hence, we propose a second approach that considers sets of feature values. The principle we rely on is the same adopted in the definitions of *anchors* by Ribeiro et al. [16] and *minimal sufficient factors* by Watson et al. [17]. We seek to find, for each instance, a minimal set of features that is sufficient to preserve the class prediction, such that changes to the other feature values of the instance would not change the class predicted by the model.

Note that the larger the value of $Importance(x_{ij}, C, e)$, the higher the influence of x_{ij} for a class prediction in model e, but even the feature with the highest $Importance$ value may still not be sufficient for a given prediction.

However, we can use this notion to find a minimal sufficient set of features. The basic idea is to sort all features in increasing order of their $Importance$ values and then identify the minimal set of top features in that sorted list which, together, are sufficient for preserving the class prediction made by the classifier.

Let c_i be the class predicted by the classifier for instance i. A feature value x_{ij} is said to "support" the prediction of c_i if and only if $P(x_{ij}|c_i) > P(x_{ij}|y), \forall y \in C \setminus \{c_i\}$. That is, the feature value x_{ij} becomes more likely if instance i has class c_i than if that instance has another class. Naturally, when searching for a minimal sufficient set of features, we only need to consider feature values that support the class predicted by the classifier.

A method for identifying a minimal sufficient set of features for the class prediction for a given instance is presented in Algorithm 1.

Based on the sufficiency criterion, we define a measure of the importance of a feature f_j for a classifier e given the set of instances X, denoted $SImportance(f_j, e, X)$, as the proportion of instances in X for which f_j is in the minimal sufficient set returned by Algorithm 1.

The importance of a feature for the entire ensemble is computed by simply averaging its importance across all classifiers in the ensemble:

$$SImportance(f_j, X) = \frac{\sum_{u=1}^{t} SImportance(f_j, e_u, X)}{t}$$

where e_u is the u-th classifier and t is the total number of classifiers in the ensemble.

2.5 Datasets

We have evaluated the proposed approaches on real data from two application domains. The first domain is the classification of ageing-related genes regarding their effect on the lifespan of an organism, which may be positive (pro-longevity) or negative (anti-longevity). From this domain, we used the 4 datasets generated by Maia et al. [3], which integrate data from the GenAge database (Build 20) [8] and the STRING database (Version 11.0) [9]. Each dataset contains data regarding ageing-related genes of one of the 4 major model organisms from the GenAge database: *C. elegans* (roundworm), *D. melanogaster* (fruit fly), *M. musculus* (mouse), and *S. cerevisiae* (baker's yeast). Each feature in these datasets refers to a protein-protein interaction (PPI) extracted from the STRING database.

The second domain involves the prediction of drugs' side effects. For this domain, we have generated 6 new datasets by integrating data from the SIDER database (Version 4.1) [10] and the STITCH database (Version 5.0) [11]. The SIDER database contains information on marketed medicines and their recorded side effects (adverse drug reactions). STITCH is a database of protein-chemical interactions (PCI) that stem from computational predictions, knowledge transfer between organisms, and interactions aggregated from other databases.

Each side-effect dataset refers to one of the 6 most frequent side effects in the SIDER database: nausea, headache, dermatitis, rash, vomiting and dizziness. Each instance in these datasets refers to a drug and consists of uncertain features referring to PCIs and a binary class variable indicating whether the corresponding drug has the side effect represented in the dataset (positive) or not (negative). Each PCI feature refers to one protein and has a binary domain, indicating whether or not an interaction between the corresponding chemical (drug) and the protein referred by the feature has been observed. As uncertain features, they are represented by probability distributions.

A value x_{ij} of an uncertain binary feature f_j for an instance i in a dataset is represented by a probability distribution $P_{ij} = (p_{ij1}, p_{ij2})$, where p_{ij1} and p_{ij2} are the complementary probabilities of x_{ij} taking each of the two values in $dom(f_j)$. Therefore, each probability distribution representing a PPI or PCI feature value is encoded by a single value p_{ij}, and $P_{ij} = (p_{ij}, 1 - p_{ij})$. In our datasets, this value p_{ij} is the confidence score (interaction probability) obtained from the STRING or STITCH databases for the corresponding PPI or PCI features, respectively.

Table 1 presents detailed information about the datasets. They are particularly challenging for having a large number of features, a small number of instances, and a very high percentage of missing values (when there is no information

TABLE 1
Information about the Datasets

Dataset	Instances	Features	Missing Values (%)	Class (%)	
				Neg.	Pos.
AG-Worm	763	9692	93.8	66.3	33.7
AG-Fly	185	3883	88.4	37.3	62.7
AG-Mouse	82	4216	78.4	37.8	62.2
AG-Yeast	382	4274	90.3	88.0	12.0
SE-Nausea	1394	9096	97.5	15.4	84.6
SE-Headache	1394	9096	97.5	21.7	78.3
SE-Dermatitis	1394	9096	97.5	23.2	76.8
SE-Rash	1394	9096	97.5	23.8	76.2
SE-Vomiting	1394	9096	97.5	24.0	76.0
SE-Dizziness	1394	9096	97.5	27.3	72.7

regarding a specific interaction in the STRING or STITCH databases). To avoid overfitting, we have discarded PPI and PCI features with low support (annotating less than 10 instances). As usual in the literature using PPIs as features for classifying genes, we represent missing values as zeros.

2.6 Ensemble methods

We consider three baseline ('uncertainty-unaware') ensemble methods: two kinds of ensembles of NB classifiers and one Random Forest (RF).

The baseline NB ensembles, ENB-NV and ENB-EV, were also used by Maia et al. [3]. In ENB-NV (Ensemble of NB classifiers with Numeric Values), the NB classifiers treat each uncertain value (an interaction probability) as a numeric value and assume that the feature values' probability distributions are Gaussian. In ENB-EV (Ensemble of NB classifiers with Expected Values), the NB classifiers binarise each uncertain value into an expected value using the threshold 0.5 and consider multivariate Bernoulli distributions for the data.

In the baseline RF, RF-DFE (RF Distributing Fractions of Examples), the RF's decision trees distribute fractions of examples over the child nodes when splitting a node on an uncertain feature, a common approach for handling uncertain data in decision trees [4], [18], [19].

The baseline ensembles use conventional strategies for sampling instances and features. For each of them, we build three versions by incorporating different combinations of our proposed approaches: BB, BRS and BB+BRS for NB ensembles; BB, BS and BB+BS for Random Forests.

We have coded the algorithms by extending available implementations from the scikit-learn library [20]. We have set the number of base classifiers (NB or decision trees) in each ensemble to 500 and the number of instances used to build each one to n. The number of features to be sampled (to train each NB classifier or to split a tree node in a Random Forest) has been set to $\sqrt{m}$.

2.7 Predictive performance measures

We have assessed the predictive performance of the algorithms using two metrics: the Area Under the Receiver Operating Characteristic curve (AUROC) and the geometric mean of sensitivity and specificity (G-mean) [12].

We evaluated each algorithm using the well-known 10-fold cross-validation. Furthermore, we have assessed the statistical significance of the differences in the predictive performance measures between each pair of algorithms, using a paired Wilcoxon signed-rank test for each dataset, with a significance level of 0.05.

2.8 Simpson's paradox

Lex X be a binary feature, taking values x_1 or x_2, and Y be the class variable in a dataset. Let the dataset's instances be divided into two groups: those with $X = x_1$ and those with $X = x_2$. Consider a class label of interest, y_1 (e.g., the pro-longevity or anti-longevity label in our ageing-related datasets). Let $P(y_1|x_1)$ and $P(y_1|x_2)$ denote the conditional probabilities of y_1 for the corresponding groups of instances. Consider the scenario where each group of instances is further divided according to the values of another binary feature Z, called a confounder, taking values z_1 or z_2 (in our datasets the confounders are binary, but this condition could be relaxed). Simpson's paradox occurs when $P(y_1|x_1) > P(y_1|x_2)$ and $P(y_1|x_1, z_j) < P(y_1|x_2, z_j)$, for $j \in \{1, 2\}$ or vice-versa: $P(y_1|x_1) < P(y_1|x_2)$ and $P(y_1|x_1, z_j) > P(y_1|x_2, z_j)$, for $j \in \{1, 2\}$. That is, the paradox occurs if the conditional probability of the class label of interest y_1 'increases' ('decreases') from the group where $X = x_1$ to the group where $X = x_2$ but, surprisingly, the conditional probability of y_1 'decreases' ('increases') from the former to the latter group, both for instances with $Z = z_1$ and instances with $Z = z_2$ [21], [22], [23]. Hence, the paradox shows a reversal of the direction of the association between the values of a feature and the probability of a class label of interest, in the context of a confounder.

3 Computational Results

3.1 Assessing the predictive performance

3.1.1 Experiment 1

This experiment evaluated NB ensembles using NB-NV as base classifiers (i.e., Gaussian NB with Numeric Values of features). We have compared four ensembles: the baseline ensemble of NB-NVs, denoted ENB-NV; and three uncertainty-aware ensembles, combining ENB-NV with the biased sampling approaches, denoted ENB-NV+BB, ENB-NV+BRS and ENB-NV+BB+BRS.

Table 2 presents the results regarding the AUROC and G-mean metrics across the 10 datasets. The best values (for each metric and dataset) are in bold. The last row shows each ensemble's average rank (per metric). Superscript symbols indicate the statistically significant differences. Based on these results, ENB-NV+BRS is the best ensemble regarding AUROC and G-mean, with an average rank of 1.6 for both metrics.

3.1.2 Experiment 2

This experiment evaluated NB ensembles using NB-EV (which binarises uncertain values into Expected Values) as base classifiers. Again, we have compared four ensembles: the uncertainty-unaware baseline ensemble of NB-EVs, denoted ENB-NV; and three uncertainty-aware ensembles combining ENB-EV with the biased sampling approaches,

TABLE 2
Experiment 1 Results

Dataset	AUROC (%)				G-mean (%)			
	ENB-NV	ENB-NV +BB	ENB-NV +BRS	ENB-NV +BB+BRS	ENB-NV	ENB-NV +BB	ENB-NV +BRS	ENB-NV +BB+BRS
AG-Worm	71.46	72.32	**72.33**	72.26	57.94	57.34	**60.83***†	60.55†
AG-Fly	65.62	65.94	65.03	**66.37**	57.31	57.61	**59.79**	58.37
AG-Mouse	66.73†	63.66	**68.51**	66.96	57.09	59.94	56.28	**63.13**
AG-Yeast	61.62	**63.81***	61.22	62.74‡	57.76	**60.51**§	58.90§	54.66
SE-Nausea	58.89	56.53	**65.16***†§	51.10	20.47	24.97	**28.97***	25.60
SE-Headache	56.35§	54.64§	**56.92**§	51.75	**53.61**†§	23.20	43.22†§	24.45
SE-Dermatitis	58.06§	56.05§	**59.83**†§	51.86	20.81	36.11	**54.16***†§	21.48
SE-Rash	56.98§	55.24§	**59.78**†§	51.10	21.12	40.63§	**54.94***†§	20.40
SE-Vomiting	60.64†§	58.24§	**65.19***†§	53.69	23.18	**34.78**	33.55*§	28.70
SE-Dizziness	61.63†§	55.21	**65.01***†§	51.63	24.73	49.86	**57.00***†§	23.21
Avg. Rank	2.5	2.7	**1.6**	3.2	3.2	2.4	**1.6**	2.8

*Statistically significant advantage (vs ENB-NV)
†Statistically significant advantage (vs ENB-NV+BB)
‡Statistically significant advantage (vs ENB-NV+BRS)
§Statistically significant advantage (vs ENB-NV+BB+BRS)

denoted ENB-EV+BB, ENB-EV+BRS and ENB-EV+BB+BRS. Table 3 presents the results for this group of ensembles. In general, ENB-EV+BRS is the best ensemble for AUROC and G-mean, with the average ranks of 1.4 and 1.6, respectively.

3.1.3 Experiment 3

This experiment aimed at determining the best NB ensemble regarding each of the AUROC and G-mean metrics. Table 4 presents the results for ENB-NV+BRS and ENB-EV+BRS, the best ensembles from Experiments 1 and 2, respectively. ENB-EV+BRS obtained the best AUROC results, with the average rank of 1.4, whereas ENB-NV+BRS obtained the best G-mean results, with an average rank of 1.0.

3.1.4 Experiment 4

This experiment evaluated four RFs: RF-DFE (the baseline) and three uncertainty-aware RFs, combining RF-DFE with the biased sampling approaches: RF-DFE+BB, RF-DFE+BS, RF-DFE+BB+BS. Table 5 presents the results. RF-DFE+BB obtained the best AUROC results, with an average rank of 2.0; whereas RF-DFE+BS obtained the best G-mean results, also with an average rank of 2.0.

3.1.5 Experiment 5

This last experiment aimed at determining the best overall method regarding each of the AUROC and G-mean metrics. Table 6 presents the results. RF-DFE+BB obtained the best AUROC results (average rank: 1.1), whereas RF-DFE+BS obtained the best G-mean results (average rank: 1.3).

As a general conclusion from all these experiments, the results support the hypothesis that the approaches BB, BRS and BS improve the predictive performance of ensembles on uncertain data. The BB and BS approaches proposed in this work obtained the best overall results since RF-DFE+BB and RF-DFE+BS were the best overall ensembles for the AUROC and G-mean metrics, respectively.

3.2 Identifying the top-ranked PPI features

We have applied our two proposed approaches for interpreting NB ensembles to the four ageing-related datasets to identify the top-ranked PPI features for classification.

Among all NB ensembles evaluated in our experiments, ENB-EV+BRS and ENB-NV+BRS have achieved the best overall AUROC and G-mean values, respectively. We have selected ENB-EV+BRS for model interpretation since it uses binarised features, facilitating interpretation.

We have trained a model applying ENB-EV+BRS to each dataset's whole set of instances. Then, we have used this model to produce rankings of features in decreasing order of importance.

Table 7 presents the top-10 features for each organism (dataset) regarding the importance measure based on conditional probabilities. For each feature, columns 2–5 present, respectively, its importance-based rank, the corresponding protein ID from the STRING database, and the corresponding gene's symbol and name. Columns 6 and 7 present, each one, the absolute and relative frequencies of the feature value 1 (considering the threshold 0.5 used) for the corresponding feature in the instances of the Pro-longevity and Anti-longevity classes, respectively. The last column shows whether or not the feature is involved in occurrences of Simpson's paradox [21], [22] (discussed in Subsection 3.4).

Out of the 40 top-ranked PPI features in Table 7, interestingly, 15 represent ribosomal proteins, namely 5 of the top-10 PPI features for the worm dataset and all the top-10 PPI features for the yeast dataset. It is also worth observing in Table 7 the relative frequency of genes (dataset instances) of each class (Pro- vs Anti-longevity) that interact with the gene associated with each of these 15 ribosomal proteins (PPI features). In all those 15 table rows, the relative frequency of Anti-longevity genes interacting with the corresponding ribosomal protein is substantially higher than that of Pro-longevity genes interacting with that ribosomal protein. The relative frequency differences are particularly

TABLE 3
Experiment 2 Results

Dataset	AUROC (%)				G-mean (%)			
	ENB-EV	ENB-EV +BB	ENB-EV +BRS	ENB-EV +BB+BRS	ENB-EV	ENB-EV +BB	ENB-EV +BRS	ENB-EV +BB+BRS
AG-Worm	**76.91**†‡§	75.39§	74.81§	73.49	36.27	38.14	49.27*†	**54.68***†‡
AG-Fly	64.05	66.24	**69.00**	67.91	28.28	26.09	28.06	**30.10**
AG-Mouse	69.26	**70.03**	69.30	68.92	41.59	35.58	**46.82**	35.10
AG-Yeast	75.55	75.77	76.79	**78.30**	22.23	32.62	47.78*†	**67.77**†‡
SE-Nausea	50.45	46.81	**57.06**†	46.79	21.84	18.03	**25.68***†	19.20
SE-Headache	52.25†§	48.86	**54.86***†§	48.26	24.64	18.57	**25.76**†§	20.17
SE-Dermatitis	54.55†§	51.07	**58.57***†§	52.00	22.12	17.89	**23.51**†	21.64†
SE-Rash	53.65†§	49.19	**57.71***†§	49.76	21.72	17.59	**23.06**†	21.24†
SE-Vomiting	56.43†§	46.27	**65.91***†§	47.95†	**15.99**	14.21	15.00	14.18
SE-Dizziness	57.72†§	46.07	**65.20***†§	46.90	**23.78**†§	14.23	23.72†	21.58†
Avg. Rank	2.4	3.1	**1.4**	3.1	2.2	3.6	**1.6**	2.6

*Statistically significant advantage (vs ENB-EV)
†Statistically significant advantage (vs ENB-EV+BB)
‡Statistically significant advantage (vs ENB-EV+BRS)
§Statistically significant advantage (vs ENB-EV+BB+BRS)

TABLE 4
Experiment 3 Results

Dataset	AUROC (%)		G-mean (%)	
	ENB-NV +BRS	ENB-EV +BRS	ENB-NV +BRS	ENB-EV +BRS
AG-Worm	72.33	**74.81**	**60.83***	49.27
AG-Fly	65.03	**69.00**	**59.79***	28.06
AG-Mouse	68.51	**69.30**	**56.28***	46.82
AG-Yeast	61.22	**76.79***	**58.90**	47.78
SE-Nausea	**65.16**	57.06	**28.97**	25.68
SE-Headache	**56.92**	54.86	**43.22***	25.76
SE-Dermatitis	**59.83**	58.57	**54.16***	23.51
SE-Rash	**59.78***	57.71	**54.94***	23.06
SE-Vomiting	65.19	**65.91**	**33.55***	15.00
SE-Dizziness	65.01	**65.20**	**57.00***	23.72
Avg. Rank	1.6	**1.4**	**1.0**	2.0

*Statistically significant advantage

striking for 9 of the 10 yeast PPI features in the table, which have a relative frequency of 0% for Pro-longevity genes and relative frequencies varying from 14.0% to 18.2% for Anti-longevity genes.

Interestingly, despite this strong pattern, none of these 9 yeast ribosomal proteins is included in GenAge [8] – the most comprehensive database of ageing-related genes. By itself, this strong pattern does not allow us to conclude that those 9 ribosomal proteins have an anti-longevity effect on yeast, which in principle could be confirmed only via appropriate biological experiments. However, the pattern seems strong enough to justify further investigation about some of those 9 ribosomal proteins in future work.

Among the 5 ribosomal proteins in Table 7 for worm, 4 (rps-0, rps-5, rps-11, rpl-3) are included in the GenAge database. Actually, among the top-10 PPI features for worm in the table, 8 are associated with genes included in the GenAge database – the exceptions are atp-1 and rps-30.

Regarding the top-10 PPI features for mice, only the top-ranked one, igf1, is included in GenAge. Interestingly, igf1 is annotated as having an "unclear" effect on longevity in GenAge, whilst its closely related igf1r (igf1 receptor) is annotated as Anti-longevity. In Table 7, the relative frequency of Anti-longevity genes interacting with the igf1 gene is 51.6%, which is much larger than the relative frequency of such interaction in the Pro-longevity class: 19.6%.

Finally, out of the top-10 PPI features for fly in Table 7, 4 are associated with genes included in GenAge, namely Sod1, FOXO, Sod2, park. Among the 6 genes not included in GenAge, there are 3 heat shock proteins. Two of them, Hsp70Ab and Hsp70Aa, have a relative frequency of only 5.8% for the Anti-longevity class, with a much larger relative frequency of 19.0% and 19.8%, respectively, for the Pro-longevity class.

Table 8 presents the top-10 features for each organism regarding the importance measure based on minimal sufficient features. Most features from Table 7 are also in Table 8: 8 features for worm, 9 for fly, 6 for mouse and 6 for yeast.

As RF-DFE+BB+BS achieved the best predictive performance on the ageing-related datasets, we also produced feature rankings using it. We have used the default RF feature importance measure from the scikit-learn's implementation, based on the Gini index. Table 9 presents the top-10 features for each organism. Consistently with Tables 7 and 8, many of the top-10 PPI features for worm and yeast in Table 9 represent ribosomal proteins, which are mainly associated with the anti-longevity class.

3.3 Confirming the relevance of the top-ranked features for the biology of ageing

Overall, the top-ranked features fit well with current knowledge on longevity/ageing and previous similar analyses, as follows. The top-ranked features for worms include daf-16, a key regulator of longevity in worms. Mutations in daf-16 suppress the longevity effects caused by several

TABLE 5
Experiment 4 Results

Dataset	AUROC (%)				G-mean (%)			
	RF-DFE	RF-DFE +BB	RF-DFE +BS	RF-DFE +BB+BS	RF-DFE	RF-DFE +BB	RF-DFE +BS	RF-DFE +BB+BS
AG-Worm	**76.62**†‡§	75.41	74.86	74.73	52.95	56.55*	55.83*	**60.22***†‡
AG-Fly	67.87	**71.81**	67.27	69.57	**66.87**†	57.43	64.42†	65.91
AG-Mouse	68.38	67.68	70.66	**72.60**	**60.65**	56.63	59.14	56.63
AG-Yeast	77.44	78.75	78.51	**79.11**	51.99	51.03	52.46	**57.30**
SE-Nausea	69.33	**69.99**	69.08	69.52	55.91†§	16.40	**57.26**†§	16.40
SE-Headache	**68.55**§	66.66	67.59	66.88	**59.39**†§	20.18	57.96†§	22.62
SE-Dermatitis	65.11‡	66.44*‡	62.89	**66.86***‡	52.20†§	20.90	**54.75***†§	19.39
SE-Rash	64.83‡	**66.26**‡	63.45	66.11*‡	53.17†§	21.35	**55.05***†§	19.66
SE-Vomiting	68.02	**68.32**	68.15	67.83	51.05	58.69	56.20*	**59.29**
SE-Dizziness	68.55	69.02	67.74	**69.67**†‡	50.98	54.66	56.81*	**60.23**†
Avg. Rank	2.7	**2.0**	3.2	2.1	2.4	3.1	**2.0**	2.3

*Statistically significant advantage (vs RF-DFE)
†Statistically significant advantage (vs RF-DFE+BB)
‡Statistically significant advantage (vs RF-DFE+BS)
§Statistically significant advantage (vs RF-DFE+BB+BS)

TABLE 6
Experiment 5 Results

Dataset	AUROC (%)		G-mean (%)	
	ENB-EV +BRS	RF-DFE +BB	ENB-NV +BRS	RF-DFE +BS
AG-Worm	74.81	**75.41**	**60.83**	55.83
AG-Fly	69.00	**71.81**	59.79	**64.42***
AG-Mouse	**69.30**	67.68	56.28	**59.14**
AG-Yeast	76.79	**78.75**	**58.90**	52.46
SE-Nausea	57.06	**69.99***	28.97	**57.26***
SE-Headache	54.86	**66.66***	43.22	**57.96***
SE-Dermatitis	58.57	**66.44***	54.16	**54.75***
SE-Rash	57.71	**66.26***	54.94	**55.05***
SE-Vomiting	65.91	**68.32**	33.55	**56.20***
SE-Dizziness	65.20	**69.02***	**57.00***	56.81
Avg. Rank	1.9	**1.1**	1.7	**1.3**

*Statistically significant advantage

mutations [24]. Other top genes in worms include various ribosomal proteins, which is not surprising given that these control translation, which has been strongly associated with longevity regulation in worms and other organisms [25]. Mitochondrial genes are also among the top hits, which also fits current knowledge of the role of mitochondria in ageing and longevity regulation [26]. Lastly, one of the top genes is atg-7, an autophagy regulator that is a major longevity pathway in invertebrates, including in worms [27].

There are heat shock proteins and genes related to stress response among the top genes in flies. This fits well with the long-established observation that stress resistance is important for healthy ageing and longevity [28]. There are also antioxidant enzymes, like superoxide dismutase, thioredoxin and glutathione peroxidase; in invertebrates and flies, in particular, antioxidant protection has long been thought to be important for longevity [29]. Interestingly, in flies, we see repair pathways and mechanisms that protect against stress among top features, particularly with a high frequency of pro-longevity interactions. In yeast, most top features are ribosomal proteins, which, as mentioned earlier, have been related to longevity regulation in model organisms.

In mice, the top gene is igf-1, with a strong anti-longevity frequency. The growth hormone/insulin/IGF1 pathway is the major longevity pathway in mammals [30], [31], so this result fits well our knowledge of longevity. Also, other players in the pathway like forkhead box proteins, Pik3cd, Ins2 and Irs2 are among the top predictions. Some brain and neuronal factors (e.g., Src) are also among the top features, which could fit GH/IGF1's neuroendocrine regulation [30]. Alternatively, they could be related to ageing changes in the brain. As Src is not in GenAge, it could be an interesting target for future studies.

Overall, the top-ranked features fit nicely into pro- and anti-longevity pathways enriched in GenAge [27]. Of note, in worms, ribosomal proteins and mitochondrial proteins involved in oxidative phosphorylation have been previously found enriched in anti-longevity processes [27], while in flies, responses to oxidative stress (like antioxidant enzymes) are among enriched pro-longevity processes. In mice, the insulin signalling pathway is a top enriched anti-longevity pathway [27], in line with our results.

3.4 Detecting occurrences of Simpson's paradox

When interpreting an association between each top-ranked feature and a class label (pro- or anti-longevity), it is important to consider the possibility that the direction of that association might be misleading due to an occurrence of Simpson's paradox. By direction of association we mean whether the presence of a PPI is associated with an increased probability of the pro-longevity or conversely the anti-longevity class.

Simpson's paradox occurs when the direction of an association between two variables X and Y at the population (aggregated) level is reversed in all the sub-groups

TABLE 7
Top-10 PPI Features in the ENB-EV+BRS Model According to the Importance Measure Based on Conditional Probabilities

Organism	Feat. Rank	STRING ID	Gene Symbol	Protein Name	Freq. in Pro-long.	Freq. in Anti-long.	Parad.
Worm	1	R13H8.1h	daf-16	Forkhead box protein O	40 (15.6%)	43 (8.5%)	no
	2	F42G8.12	isp-1	Cytochrome b-c1 complex subunit Rieske, mitochondrial	9 (3.5%)	58 (11.5%)	no
	3	C34E10.6.1	atp-2	ATP synthase subunit beta, mitochondrial	6 (2.3%)	58 (11.5%)	no
	4	T05E11.1	rps-5	40S ribosomal protein S5	5 (1.9%)	60 (11.9%)	no
	5	F40F11.1.2	rps-11	Ribosomal protein, small subunit	4 (1.6%)	53 (10.5%)	no
	6	C26F1.4.2	rps-30	40S ribosomal protein S30	5 (1.9%)	43 (8.5%)	no
	7	B0250.1	rpl-2	60S ribosomal protein L8	3 (1.2%)	44 (8.7%)	no
	8	B0393.1.1	rps-0	40S ribosomal protein SA	6 (2.3%)	51 (10.1%)	no
	9	H28O16.1a	H28O16.1	ATP synthase subunit alpha, mitochondrial	6 (2.3%)	56 (11.1%)	no
	10	Y56A3A.19	Y56A3A.19	Acyl carrier protein	2 (0.8%)	51 (10.1%)	no
Fly	1	FBpp0082516	Hsc70-4	Heat shock 70 kDa protein cognate 4	34 (29.3%)	9 (13.0%)	no
	2	FBpp0305736	Sod	Superoxide dismutase [Cu-Zn]	30 (25.9%)	4 (5.8%)	no
	3	FBpp0081956	Hsp70Ab	Heat shock protein 70Ab	22 (19.0%)	4 (5.8%)	no
	4	FBpp0293589	foxo	Forkhead box protein O	36 (31.0%)	14 (20.3%)	no
	5	FBpp0081986	Hsp70Aa	Major heat shock 70 kDa protein Aa	23 (19.8%)	4 (5.8%)	no
	6	FBpp0070899	schlank	Schlank, isoform A	31 (26.7%)	12 (17.4%)	no
	7	FBpp0086226	Sod2	Superoxide dismutase [Mn], mitochondrial	31 (26.7%)	8 (11.6%)	no
	8	FBpp0088134	CaMKI	Calmodulin-dependent protein kinase activity	29 (25.0%)	10 (14.5%)	no
	9	FBpp0077974	park	E3 ubiquitin-protein ligase parkin	20 (17.2%)	1 (1.4%)	no
	10	FBpp0305095	Hsp83	Heat shock protein 83	26 (22.4%)	10 (14.5%)	yes
Mouse	1	ENSMUSP00000056668	Igf-1	Insulin-like growth factor 1	10 (19.6%)	16 (51.6%)	no
	2	ENSMUSP00000029175	Src	Neuronal proto-oncogene tyrosine-protein kinase Src	3 (5.9%)	12 (38.7%)	no
	3	ENSMUSP00000050683	Foxo3	Forkhead box protein O3	8 (15.7%)	13 (41.9%)	no
	4	ENSMUSP00000055308	Foxo1	Forkhead box protein O1	5 (9.8%)	11 (35.5%)	no
	5	ENSMUSP00000000369	Rem1	GTP-binding protein REM 1	7 (13.7%)	10 (32.3%)	yes
	6	ENSMUSP00000101315	Pik3cd	Phosphatidylinositol 4,5-bisphosphate 3-kinase catalytic subunit delta isoform	1 (2.0%)	7 (22.6%)	no
	7	ENSMUSP00000102538	Ngf	Beta-nerve growth factor	5 (9.8%)	8 (25.8%)	yes
	8	ENSMUSP00000031697	Cul1	Cullin-1	6 (11.8%)	7 (22.6%)	no
	9	ENSMUSP00000120152	Stat3	Signal transducer and activator of transcription 3	11 (21.6%)	14 (45.2%)	no
	10	ENSMUSP00000115578	Ubc	Polyubiquitin-C	15 (29.4%)	4 (12.9%)	no
Yeast	1	YLR167W	RPS31	Fusion-protein cleaved to yield ribosomal protein S31 and ubiquitin	0 (0.0%)	58 (17.3%)	no
	2	YIL133C	RPL16A	Ribosomal 60S subunit protein L16A	0 (0.0%)	51 (15.2%)	no
	3	YBR048W	RPS11B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	52 (15.5%)	no
	4	YPL090C	RPS6A	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	51 (15.2%)	no
	5	YGL103W	RPL28	Ribosomal 60S subunit protein L28	0 (0.0%)	61 (18.2%)	no
	6	YNL096C	RPS7B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	50 (14.9%)	no
	7	YJR145C	RPS4A	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	51 (15.2%)	no
	8	YNL069C	RPL16B	Ribosomal 60S subunit protein L16B	0 (0.0%)	48 (14.3%)	no
	9	YBR031W	RPL4A	Ribosomal 60S subunit protein L4A	1 (2.2%)	53 (15.8%)	no
	10	YNL302C	RPS19B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	47 (14.0%)	no

produced by partitioning that population according to the values of a third variable, Z, called a confounder [21], [22]. In other words, the direction of the association between variables X and Y is reversed when conditioning on each value of the confounder Z. In our classification task, X and Z are predictive features, whilst Y is the class variable.

Table 10 shows an example of this paradox in the fly dataset. Looking only at the aggregated data in the first row of the table, ignoring the interaction between the values of the Hsp83 and Hsc70-4 PPI features, we would conclude that the feature value "interaction with Hsp83 = yes" is more associated with the pro-longevity class than "interaction with Hsp83 = no". Actually, among the genes/proteins (instances) in our dataset that interact with Hsp83, 72.2% are pro-longevity genes, whilst among the genes/proteins that do not interact with Hsp83, 60.4% are pro-longevity genes. So, interacting with Hsp83 is associated with an increased probability of the pro-longevity class label.

However, when we look at the data partitioned by the values of the "interaction with Hsc70-4" feature in the second and third rows of the table, a different pattern emerges. Among genes/proteins that do not interact with Hsc70-4, the relative frequency of the pro-longevity class is higher among genes/proteins that do not interact with Hsp83 (58.4%) than among genes/proteins that interact with Hsp83 (50%). The same pattern is observed among genes/proteins that interact with Hsc70-4 (83.3% for Hsp83=no vs 76.7% for Hsp83=yes). Hence, in both subgroups of genes/proteins (interacting or not with Hsc70-4), interacting with Hsp83 is associated with decreased probability of the pro-longevity class label, the reverse of the direction of association observed for the aggregated data.

Tables 7 and 8 indicate Simpson's paradox occurrences for the feature Hsp83 in the fly dataset and features Rem1 and Ngf in the mouse dataset. Hence, when interpreting the association between those features and the class variable, one should be aware of those paradox occurrences to avoid drawing wrong conclusions about the data.

TABLE 8
Top-10 PPI Features in the ENB-EV+BRS Model According to the Importance Measure Based on Sufficiency

Organism	Feat. Rank	STRING ID	Gene Symbol	Protein Name	Freq. in Pro-long.	Freq. in Anti-long.	Parad.
Worm	1	F42G8.12	isp-1	Cytochrome b-c1 complex subunit Rieske, mitochondrial	9 (3.5%)	58 (11.5%)	no
	2	C26F1.4.2	rps-30	40S ribosomal protein S30	5 (1.9%)	43 (8.5%)	no
	3	C34E10.6.1	atp-2	ATP synthase subunit beta, mitochondrial	6 (2.3%)	58 (11.5%)	no
	4	B0250.1	rpl-2	60S ribosomal protein L8	3 (1.2%)	44 (8.7%)	no
	5	F40F11.1.2	rps-11	Ribosomal protein, small subunit	4 (1.6%)	53 (10.5%)	no
	6	B0393.1.1	rps-0	40S ribosomal protein SA	6 (2.3%)	51 (10.1%)	no
	7	T05E11.1	rps-5	40S ribosomal protein S5	5 (1.9%)	60 (11.9%)	no
	8	F28D1.7.1	rps-23	40S ribosomal protein S23	4 (1.6%)	48 (9.5%)	no
	9	C49H3.11.1	rps-2	40S ribosomal protein S2	7 (2.7%)	57 (11.3%)	no
	10	Y56A3A.19	Y56A3A.19	Acyl carrier protein	2 (0.8%)	51 (10.1%)	no
Fly	1	FBpp0305736	Sod	Superoxide dismutase [Cu-Zn]	30 (25.9%)	4 (5.8%)	no
	2	FBpp0081956	Hsp70Ab	Heat shock protein 70Ab	22 (19.0%)	4 (5.8%)	no
	3	FBpp0082516	Hsc70-4	Heat shock 70 kDa protein cognate 4	34 (29.3%)	9 (13.0%)	no
	4	FBpp0081986	Hsp70Aa	Major heat shock 70 kDa protein Aa	23 (19.8%)	4 (5.8%)	no
	5	FBpp0086226	Sod2	Superoxide dismutase [Mn], mitochondrial	31 (26.7%)	8 (11.6%)	no
	6	FBpp0293589	foxo	Forkhead box protein O	36 (31.0%)	14 (20.3%)	no
	7	FBpp0077974	park	E3 ubiquitin-protein ligase parkin	20 (17.2%)	1 (1.4%)	no
	8	FBpp0070899	schlank	Schlank, isoform A	31 (26.7%)	12 (17.4%)	no
	9	FBpp0088134	CaMKI	Calmodulin-dependent protein kinase activity	29 (25.0%)	10 (14.5%)	no
	10	FBpp0078604	Aux	Auxilin, isoform A	17 (14.7%)	3 (4.3%)	no
Mouse	1	ENSMUSP00000056668	Igf-1	Insulin-like growth factor 1	10 (19.6%)	16 (51.6%)	no
	2	ENSMUSP00000029175	Src	Neuronal proto-oncogene tyrosine-protein kinase Src	3 (5.9%)	12 (38.7%)	no
	3	ENSMUSP00000050683	Foxo3	Forkhead box protein O3	8 (15.7%)	13 (41.9%)	no
	4	ENSMUSP00000055308	Foxo1	Forkhead box protein O1	5 (9.8%)	11 (35.5%)	no
	5	ENSMUSP00000101553	Ins2	Insulin-2	6 (11.8%)	13 (41.9%)	no
	6	ENSMUSP00000099878	Rps6	40S ribosomal protein S6	3 (5.9%)	8 (25.8%)	no
	7	ENSMUSP00000021090	Grb2	Growth factor receptor-bound protein 2	3 (5.9%)	8 (25.8%)	no
	8	ENSMUSP00000099621	Rpa2	Replication protein A 32 kDa subunit	12 (23.5%)	1 (3.2%)	no
	9	ENSMUSP00000120152	Stat3	Signal transducer and activator of transcription 3	11 (21.6%)	14 (45.2%)	no
	10	ENSMUSP00000102538	Ngf	Beta-nerve growth factor	5 (9.8%)	8 (25.8%)	yes
Yeast	1	YLR167W	RPS31	Fusion-protein cleaved to yield ribosomal protein S31 and ubiquitin	0 (0.0%)	58 (17.3%)	no
	2	YNL096C	RPS7B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	50 (14.9%)	no
	3	YJR145C	RPS4A	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	51 (15.2%)	no
	4	YGL103W	RPL28	Ribosomal 60S subunit protein L28	0 (0.0%)	61 (18.2%)	no
	5	YBR048W	RPS11B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	52 (15.5%)	no
	6	YKR094C	RPL40B	Ubiquitin-ribosomal 60S subunit protein L40B fusion protein	0 (0.0%)	59 (17.6%)	no
	7	YIL133C	RPL16A	Ribosomal 60S subunit protein L16A	0 (0.0%)	51 (15.2%)	no
	8	YGL030W	RPL30	Ribosomal 60S subunit protein L30	0 (0.0%)	50 (14.9%)	no
	9	YGL123W	RPS2	Protein component of the small (40S) subunit	1 (2.2%)	58 (17.3%)	no
	10	YKL009W	MRT4	Protein involved in mRNA turnover and ribosome assembly	0 (0.0%)	55 (16.4%)	no

4 Conclusion

This work addresses the classification problem in datasets with uncertain categorical features, whose values are represented by probability distributions. We have proposed ensemble approaches called Biased Bootstrap (BB) and Biased Splitting (BS) for coping with this type of uncertainty, based on the principle that features with lower uncertainty degrees have better class-discrimination potential since there is higher confidence in their actual values across the dataset.

Our experiments have evaluated these two approaches on 10 datasets in the domains of ageing-related genes and drugs' side effects. For this evaluation, we have used real data with uncertain features referring to probabilities of protein-protein and protein-chemical interactions. Our results show the ensembles using the proposed BB and BS approaches achieved higher predictive performance than baseline ensembles that do not cope with feature uncertainty, supporting the hypothesis that the proposed approaches effectively cope with uncertainty in categorical features.

Furthermore, we have proposed two approaches for interpreting an ensemble of Naive Bayes classifiers based on feature importance measures used to rank features in decreasing order of their influence in the ensemble's predictions. The first approach is straightforwardly based on conditional probability differences, while the second, more sophisticated, approach is based on the concept of a minimal set of sufficient features for classifying each instance. We have applied these two feature-ranking approaches to the ageing-related datasets and compared them to the feature ranking produced with a conventional feature importance measure for random forests. An analysis of the top-ranked features showed that, overall, they fit well the current knowledge about the influence of genes/proteins on ageing. Besides, we have also pointed out some strong patterns involving longevity effects and genes that are not included in GenAge [8]. These findings suggest some targets for fu-

TABLE 9
Top-10 PPI Features in the RF-DFE+BB+BS Model

Organism	Feat. Rank	STRING ID	Gene Symbol	Protein Name	Freq. in Pro-long.	Freq. in Anti-long.	Parad.
Worm	1	M7.5	atg-7	AuTophaGy (Yeast Atg homolog)	33 (12.8%)	18 (3.6%)	no
	2	F40F11.1.2	rps-11	Ribosomal protein, small subunit	4 (1.6%)	53 (10.5%)	no
	3	Y37D8A.14	cco-2	Cytochrome c oxidase subunit 5A, mitochondrial	2 (0.8%)	48 (9.5%)	no
	4	B0412.4	rps-29	Ribosomal protein, small subunit	4 (1.6%)	46 (9.1%)	no
	5	Y45G12B.1a	nuo-5	NADH Ubiquinone Oxidoreductase	2 (0.8%)	41 (8.1%)	no
	6	F42C5.8	rps-8	40S ribosomal protein S8	4 (1.6%)	49 (9.7%)	no
	7	Y37E3.8a	Y37E3.8	Protein Y37E3.8, isoform a (Y37E3.8) mRNA, complete cds	3 (1.2%)	46 (9.1%)	no
	8	Y57G11C.34	mrps-7	28S ribosomal protein S7, mitochondrial	2 (0.8%)	54 (10.7%)	no
	9	Y105E8A.16.1	rps-20	Ribosomal protein, small subunit	3 (1.2%)	48 (9.5%)	no
	10	F58F12.1	F58F12.1	ATP synthase subunit delta, mitochondrial	4 (1.6%)	58 (11.5%)	no
Fly	1	FBpp0085780	CG15116	Glutathione peroxidase activity	21 (18.1%)	3 (4.3%)	no
	2	FBpp0070416	ph-p	Polyhomeotic-proximal chromatin protein	3 (2.6%)	4 (5.8%)	no
	3	FBpp0071973	Pi3K59F	Phosphotidylinositol 3 kinase 59F, a.k.a. Vacuolar protein sorting 34	21 (18.1%)	4 (5.8%)	no
	4	FBpp0070717	dhd	Thioredoxin-1	10 (8.6%)	1 (1.4%)	no
	5	FBpp0078138	CG7133	annotation not available	2 (1.7%)	0 (0.0%)	no
	6	FBpp0072932	PHGPx	Peroxidase activity	19 (16.4%)	2 (2.9%)	no
	7	FBpp0083975	Atg6	Beclin-1-like protein; Autophagy-related 6	23 (19.8%)	4 (5.8%)	no
	8	FBpp0082927	Prx3	Thioredoxin peroxidase 3	10 (8.6%)	0 (0.0%)	no
	9	FBpp0087354	Prx2540-2	Peroxiredoxin 2540-2	8 (6.9%)	0 (0.0%)	no
	10	FBpp0099922	Nos	Nitric oxide synthase	5 (4.3%)	0 (0.0%)	no
Mouse	1	ENSMUSP00000128260	Tfdp2	Transcription factor dp2	1 (2.0%)	1 (3.2%)	no
	2	ENSMUSP00000126874	Ccnt1	Cyclin-T1	1 (2.0%)	2 (6.5%)	no
	3	ENSMUSP00000101315	Pik3cd	Phosphatidylinositol 4,5-bisphosphate 3-kinase catalytic subunit delta isoform	1 (2.0%)	7 (22.6%)	no
	4	ENSMUSP00000030464	Pik3r3	Phosphatidylinositol 3-kinase regulatory subunit gamma	1 (2.0%)	7 (22.6%)	no
	5	ENSMUSP00000099991	Pdpk1	3-phosphoinositide-dependent protein kinase 1	1 (2.0%)	6 (19.4%)	no
	6	ENSMUSP00000021090	Grb2	Growth factor receptor-bound protein 2	3 (5.9%)	8 (25.8%)	no
	7	ENSMUSP00000025749	Rps6kb2	Ribosomal protein S6 kinase beta-2	1 (2.0%)	4 (12.9%)	no
	8	ENSMUSP00000038514	Irs2	Insulin receptor substrate 2	4 (7.8%)	9 (29.0%)	no
	9	ENSMUSP00000034296	Pik3r2	Phosphatidylinositol 3-kinase regulatory subunit beta	3 (5.9%)	8 (25.8%)	no
	10	ENSMUSP00000047839	Ppp1r13l	RelA-associated inhibitor	3 (5.9%)	0 (0.0%)	no
Yeast	1	YKR094C	RPL40B	Ubiquitin-ribosomal 60S subunit protein L40B fusion protein	0 (0.0%)	59 (17.6%)	no
	2	YKL148C	SDH1	Flavoprotein subunit of succinate dehydrogenase	11 (23.9%)	12 (3.6%)	no
	3	YKL180W	RPL17A	Ribosomal 60S subunit protein L17A	0 (0.0%)	56 (16.7%)	no
	4	YBR143C	SUP45	Polypeptide release factor (eRF1) in translation termination	0 (0.0%)	53 (15.8%)	no
	5	YER056C-A	RPL34A	Ribosomal 60S subunit protein L34A	0 (0.0%)	52 (15.5%)	no
	6	YBR048W	RPS11B	Protein component of the small (40S) ribosomal subunit	0 (0.0%)	52 (15.5%)	no
	7	YGR148C	RPL24B	Ribosomal 60S subunit protein L24B	0 (0.0%)	48 (14.3%)	no
	8	YBL092W	RPL32	Ribosomal 60S subunit protein L32	0 (0.0%)	50 (14.9%)	no
	9	YOR065W	CYT1	Cytochrome c1, heme protein, mitochondrial	9 (19.6%)	11 (3.3%)	no
	10	YLR009W	RLP24	Essential protein required for ribosomal large subunit biogenesis	0 (0.0%)	52 (15.5%)	no

TABLE 10
Simpson's Paradox Occurrence in the Fly Dataset

	Hsp83 = no		Hsp83 = yes	
	Total	Pro-long.	Total	Pro-long.
Aggregated data	149	90 (60.4%)	36	26 (72.2%)
Hsc70-4 = no	137	80 (58.4%)	6	3 (50.0%)
Hsc70-4 = yes	12	10 (83.3%)	30	23 (76.7%)

ture biological experiments that could confirm the longevity effects of some genes/proteins.

Acknowledgments

This study was financed in part by: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil) [grant number 310444/2018-7]; Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES, Brazil) [finance code 001]; and Instituto Brasileiro de Geografia e Estatística (IBGE, Brazil).

References

[1] F. Angiulli and F. Fassetti, "Nearest neighbor-based classification of uncertain data," *ACM Transactions on Knowledge Discovery from Data*, vol. 7, no. 1, 2013.

[2] J. Ge, Y. Xia, and C. Nadungodage, "UNN: A neural network for uncertain data classification," in *Advances in Knowledge Discovery and Data Mining*, M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, Eds. Berlin: Springer, 2010, pp. 449–460.

[3] M. R. H. Maia, A. Plastino, and A. A. Freitas, "An ensemble of naive bayes classifiers for uncertain categorical data," in *2021 IEEE International Conference on Data Mining (ICDM)*, 2021, pp. 1222–1227.

[4] S. Tsang, B. Kao, K. Y. Yip, W.-S. Ho, and S. D. Lee, "Decision trees for uncertain data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 1, pp. 64–78, 2011.

[5] Z. Xie, Y. Xu, and Q. Hu, "Uncertain data classification with additive kernel support vector machine," *Data & Knowledge Engineering*, vol. 117, pp. 87–97, 2018.

[6] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[7] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[8] R. Tacutu, D. Thornton, E. Johnson, A. Budovsky, D. Barardo, T. Craig, E. Diana, G. Lehmann, D. Toren, J. Wang, V. E. Fraifeld, and J. P. de Magalhães, "Human Ageing Genomic Resources: new and updated databases," *Nucleic Acids Research*, vol. 46, no. D1, pp. D1083–D1090, 2017.

[9] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, and C. Mering, "STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Research*, vol. 47, no. D1, pp. D607–D613, 2018.

[10] M. Kuhn, I. Letunic, L. J. Jensen, and P. Bork, "The SIDER database of drugs and side effects," *Nucleic Acids Research*, vol. 44, no. D1, pp. D1075–D1079, 2015.

[11] D. Szklarczyk, A. Santos, C. von Mering, L. J. Jensen, P. Bork, and M. Kuhn, "STITCH 5: augmenting protein–chemical interaction networks with tissue and affinity data," *Nucleic Acids Research*, vol. 44, no. D1, pp. D380–D384, 2015.

[12] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms*. Cambridge, UK: Cambridge University Press, 2011.

[13] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, 2018.

[14] C. B. Azodi, J. Tang, and S.-H. Shiu, "Opening the black box: Interpretable machine learning for geneticists," *Trends in Genetics*, vol. 36, no. 6, pp. 442–455, 2020.

[15] A. Freitas, D. Wieser, and R. Apweiler, "On the importance of comprehensible classification models for protein function prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 1, pp. 172–182, 2010.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[17] D. S. Watson, L. Gultchin, A. Taly, and L. Floridi, "Local explanations via necessity and sufficiency: Unifying theory and practice," in *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021)*, 2021.

[18] C. Dudas and H. Boström, "Using uncertain chemical and thermal data to predict product quality in a casting process," in *Proceedings of the 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data*, ser. U '09. New York, USA: ACM, 2009, p. 57–61.

[19] B. Qin, Y. Xia, and F. Li, "DTU: A decision tree for uncertain data," in *Advances in Knowledge Discovery and Data Mining*, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds. Berlin: Springer, 2009, pp. 4–15.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] J. Pearl, *Causality: Models, Reasoning, and Inference*. Cambridge, UK: Cambridge University Press, 2000.

[22] ——, "Comment: Understanding Simpson's paradox," *The American Statistician*, vol. 68, no. 1, pp. 8–13, 2014.

[23] G. Shmueli and I. Iahav, "The forest or the trees? Tackling Simpson's paradox with classification trees," *Production and Operations Management*, vol. 27, no. 4, pp. 696–716, 2018.

[24] X. Sun, W.-D. Chen, and Y.-D. Wang, "Daf-16/foxo transcription factor in aging and longevity," *Frontiers in Pharmacology*, vol. 8, p. 548, 2017.

[25] Y. Gonskikh and N. Polacek, "Alterations of the translation apparatus during aging and stress response," *Mechanisms of Ageing and Development*, vol. 168, pp. 30–36, 2017.

[26] N. Sun, R. J. Youle, and T. Finkel, "The mitochondrial basis of aging," *Molecular Cell*, vol. 61, no. 5, pp. 654–666, 2016.

[27] M. Fernandes, C. Wan, R. Tacutu, D. Barardo, A. Rajput, J. Wang, H. Thoppil, D. Thornton, C. Yang, A. Freitas, and J. P. de Magalhães, "Systematic analysis of the gerontome reveals links between aging and age-related diseases," *Human Molecular Genetics*, vol. 25, no. 21, pp. 4804–4818, 2016.

[28] P. Verbeke, J. Fonager, B. F. C. Clark, and S. I. S. Rattan, "Heat shock response and ageing: Mechanisms and applications," *Cell Biology International*, vol. 25, no. 9, pp. 845–857, 2001.

[29] F. L. Muller, M. S. Lustgarten, Y. Jang, A. Richardson, and H. Van Remmen, "Trends in oxidative aging theories," *Free Radical Biology and Medicine*, vol. 43, no. 4, pp. 477–503, 2007.

[30] J. P. de Magalhães and A. Matsuda, "Genome-wide patterns of genetic distances reveal candidate loci contributing to human population-specific traits," *Annals of Human Genetics*, vol. 76, no. 2, pp. 142–158, 2012.

[31] C. J. Kenyon, "The genetics of ageing," *Nature*, vol. 464, no. 7288, pp. 504–512, 2010.

Marcelo Rodrigues de Holanda Maia is a systems analyst at Instituto Brasileiro de Geografia e Estatística, Rio de Janeiro, Brazil. He received the BSc and MSc degrees in computer science in 2008 and 2015, respectively, from Universidade Federal Fluminense, Niterói, Brazil, where he now pursues a DSc degree. He undertook a research visit to the University of Kent, Canterbury, UK, from 2020 to 2021. His current research interests concentrate on data science and combinatorial optimization.

Alexandre Plastino is a full professor at Universidade Federal Fluminense, Niterói, Brazil. He obtained his BSc (1988) and MSc (1990) degrees in computer science from Universidade Federal do Rio de Janeiro and his DSc degree in computer science from Pontifícia Universidade Católica do Rio de Janeiro (2000). In 2008, he conducted a postdoctoral research at the University of Kent, Canterbury, UK. His current research interests concentrate on data science and combinatorial optimization.

Alex A. Freitas is a professor of computational intelligence at the University of Kent, Canterbury, UK. He has an interdisciplinary academic background, with a PhD degree in computer science (University of Essex, UK, 1997), in the area of machine learning (or data mining); and a research-oriented master's degree (an MPhil) in Biological Sciences (University of Liverpool, UK, 2011), in the area of the biology of ageing. His main research interests are machine learning, the biology of ageing, and applications of machine learning to the life sciences.

João Pedro de Magalhães graduated in microbiology from Escola Superior de Biotecnologia, Porto, Portugal, in 1999, and obtained the PhD degree from the University of Namur, Belgium, in 2004. Following a postdoc at the Harvard Medical School, in 2008, he joined the University of Liverpool, where he is currently a professor and leads the Integrative Genomics of Ageing Group (http://rejuvenomicslab.com). The group's research broadly focuses on understanding the genetic, cellular, and molecular mechanisms of ageing.