Computers & Operations Research 161 (2024) 106450

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Multi-constructor CMSA for the maximum disjoint dominating sets problem

Roberto Maria Rosati [a,b,*], Salim Bouamama [c], Christian Blum [b]

[a] DPIA, University of Udine, via delle Scienze 206, Udine, 33100, Italy
[b] Artificial Intelligence Research Institute (IIIA-CSIC), Campus of the UAB, Bellaterra, 08193, Spain
[c] Department of Computer Science, Mechatronics Laboratory (LMETR), Ferhat Abbas University Sétif 1, Sétif, 19000, Algeria

ARTICLE INFO

Keywords:
CMSA
Reinforcement learning
Instance reduction
Maximum disjoint dominating sets problem
Domatic partition problem
Wireless sensor network

ABSTRACT

We propose the Multi-Constructor CMSA, a Construct, Merge, Solve and Adapt (CMSA) algorithm that employs multiple heuristic procedures, respectively solution constructors, for the Maximum Disjoint Dominating Sets Problem (MDDSP). At every iteration of the search procedure, the solution components built by the constructors are merged into a sub-instance, which is subsequently solved by an exact solver and then adapted to keep only beneficial solution components. In our CMSA the solution constructors are chosen at random according to their relative probabilities, which are adapted during the search, through a mechanism based on reinforcement learning. We test two variants of the new Multi-Constructor CMSA that employ, respectively, two and six solution constructors, on a new set of 3600 problem instances, encompassing random graphs, Watts–Strogatz networks and Barabási-Albert networks, generated through a Hammersley sampling procedure on the instance space. We compare our algorithm against six heuristics from the literature, as well as with the standard version of CMSA. Furthermore, we employ an integer linear programming (ILP) model that is able to achieve a good performance for small, sparse graphs. Overall, the experimental results show that all versions of CMSA outperform by a large margin the previous state of the art and that, among the variants of CMSA, the novel version that combines two constructors provides slightly better results than the other ones, more prominently on larger graphs.

1. Introduction

Construct, Merge, Solve & Adapt (CMSA) is a general hybrid algorithm for combinatorial optimisation proposed by Blum et al. (2016), based on the idea of (1) constructing a reduced instance of the full problem by means of merging solution components obtained, for example, by the repeated execution of a randomised construction heuristic, and (2) the solution of the reduced instance by means of an exact solver. The reduced instance is also referred to as sub-instance. The algorithm is also equipped with an ageing mechanism to ensure that unpromising solution components are discarded after a certain number of iterations. Existing applications of CMSA to hard combinatorial optimisation problems include the ones to the multi-dimensional knapsack problem (Blum and Ochoa, 2021), to prioritised test data generation (Ferrer et al., 2021), and to refuelling and maintenance planning of nuclear power plants (Dupin and Talbi, 2021), just to name a few.

In this work, we introduce the Multi-Constructor CMSA that makes use of multiple independent construction heuristics, that we, hereby, name *constructors*. For every solution construction, CMSA chooses one of the available heuristics which is then used in a probabilistic way. In order to bias the choice toward the most promising constructors for the tackled problem instance, choice probabilities[1] are associated with the constructors. We investigate the use of an adaptive mechanism based on reinforcement learning for the online adaptation of these weights. After the completion of every CMSA iteration, the algorithm assigns rewards to the constructors according to their relative contributions to the solution found by the exact solver at the given iteration. Moreover, the weights are updated according to these rewards. The learning process is guided by a parameter α, called *learning rate*, that determines the pace of learning from new samples. We also employ a minimum threshold τ to avoid the weight of a given constructor becoming too small, which would basically impede any possibility for it to be chosen again. Note that weight-adaptive metaheuristics such as the one proposed in this work are quite common in the literature. Examples range from methods based on local search – such as Alicastro et al. (2021), Canca et al. (2017), Queiroz dos Santos et al. (2014) – to population-based techniques, such as Nagra et al. (2019), Wang et al. (2015).

* Corresponding author at: DPIA, University of Udine, via delle Scienze 206, Udine, 33100, Italy.
E-mail addresses: robertomaria.rosati@uniud.it (R.M. Rosati), salim.bouamama@univ-setif.dz (S. Bouamama), christian.blum@iiia.csic.es (C. Blum).

[1] In this context, the terms *weight* and *probability*, in reference to constructors, are synonymous and interchangeable.

[2] Note that in the MDDSP it is not required that each vertex of the input graph forms part of a dominating set from a solution.

https://doi.org/10.1016/j.cor.2023.106450
Received 30 January 2023; Received in revised form 24 August 2023; Accepted 5 October 2023
Available online 7 October 2023

The proposed technique is applied to the Maximum Disjoint Dominating Sets Problem (MDDSP), a notoriously difficult optimisation problem that deals with finding the largest number of disjoint dominating sets in a given graph. Its applications are found in wireless sensor networks (WSN), studied for their applications in environmental monitoring, security surveillance, healthcare, and emergency operations (Akyildiz et al., 2002), as well as in heterogeneous multi-agent systems (Mesbahi and Egerstedt, 2010). Very much related to the MDDSP is the so-called Domatic Partition Problem (DPP), which requires – in addition to what is necessary in the MDDSP – that the disjoint dominating sets form a partition of the graph, that is, all vertices are actually assigned to exactly one set.[2] Nevertheless, note that the transformation of a solution of the MDDSP into one of the DPP is trivial because it suffices to add to any dominating set in the solution those vertices that were not chosen to be part of the dominating sets in the solution. Vice versa, a solution of the DPP is always a solution to the MDDSP. Another related problem is the Domatic Number Problem (DNP), a search problem aimed at discovering the *domatic number* of a graph, the maximum number of disjoint dominating sets that can be found in the graph, without the need to actually determine these sets.

In a previous work (Rosati et al., 2023a), we proposed a standard CMSA algorithm for MDDSP that outperformed the approaches from the literature. In the present work, we compare the Multi-Constructor CMSA with the standard CMSA and the other methods from the literature. A natural choice concerning the constructors is to select a set of randomised greedy heuristics that can be used to quickly generate solutions. In the case of the MDDSP, we produced stochastic variants of six greedy heuristics that are found in the literature. Even though some of them perform much better than others as stand-alone deterministic algorithms, beforehand it is not possible to judge their usefulness for CMSA. For this reason, we consider all of them as valid candidates for the Multi-Constructor CMSA. For the experiments, we generate a new set of graphs, belonging to three different models: random graphs, Watts–Strogatz networks, and Barabási–Albert networks. The instances are generated from an extended instance space, including graphs with up to 1000 vertices on a wider range of densities. The previously available data sets only contained random graphs with up to 250 vertices, and very sparse or very dense graphs were not considered. For the generation of the new instances, we propose a procedure based on a Hammersley point set (Hammersley and Handscomb, 1964), aimed at guaranteeing that all regions of the instance space are covered evenly.

In particular, we compare two versions of the Multi-Constructor CMSA against the standard CMSA and the heuristic approaches from the literature. Hereby, the first variant of the Multi-Constructor CMSA makes use of all six greedy heuristics from the literature, while the second variant only uses the two best greedy heuristics. Experimental results confirm that all versions of CMSA significantly improve over the previous state of the art. Finally, we remark that we outline also an Integer Linear Programming (ILP) model for the MDDSP and we implement it in CPLEX. We present three variants of the model, two of which incorporate symmetry breaking constraints. They produce good results on small and sparse graphs, where they can find many optimal solutions within the time limit of 1 h, but they do not yield comparable performance with the other methods on larger or denser graphs.

The remainder of this paper is organised as follows. Section 2 resumes the current advances in research on the MDDSP. In Section 3, we describe the problem and outline the ILP model. Section 4 presents the currently existing heuristics for the MDDSP, which are used in our work as constructors. In Section 5, we go into the algorithmic details of the Multi-Constructor CMSA and the strategy for the adaptive online learning of constructor weights. Section 6 contains the analysis and the discussion of the experimental results. Finally, the conclusions and a discussion on future work are given in Section 7.

2. Related work

Interest in the MDDSP and related problems dates back to the 70 s. The first mention of the Domatic Number Problem, which deals with deriving the domatic number of a given graph without the need for deriving the corresponding disjoint dominating sets, is found in the work of Cockayne and Hedetniemi (1975). Many other studies are found in the literature on the DNP. Most of them are primarily interested in specific families of graphs. Although not many exact approaches can be found in the literature for MDDSP-like problems, mainly due to their complexity, some exact approaches have been designed for finding optimal solutions to the DNP. In particular, the first exact deterministic exponential-time algorithm for the 3-DNP was designed by Riege and Rothe (2005). The algorithm has a running time of $\mathcal{O}(2.9416^n)$ and uses polynomial space, which is in contrast to a naive approach that runs in $\mathcal{O}(3^n)$ of time. This time complexity was later improved to $\mathcal{O}(2.695^n)$ by Riege et al. (2007). Combining the two main techniques typically used for the design of exact exponential-time algorithms – inclusion & exclusion, respectively measure & conquer – Van-Rooij (2010) provided a fast polynomial-space algorithm that computes the domatic number in $\mathcal{O}(2.7139^n)$ time.

The Domatic Partition Problem was first introduced by Cockayne and Hedetniemi (1977), two years after the DNP. In the same paper, the authors showed that the problem has an upper bound of $\delta(G)+1$, where $\delta(G)$ is the minimum degree of all vertices in G. In other words, an optimal solution to the DPP can never have more disjoint dominating sets than $\delta(G)+1$. Furthermore, the problem was shown to belong to the class of NP-hard problems on general graphs (Garey and Johnson, 1979). In fact, it remains NP-hard for co-bipartite graphs (Poon et al., 2012). Moreover, unless P=NP, the MDDSP has no polynomial-time α-approximation algorithm for any constant α smaller than 1.5 (Cardei et al., 2002). Existing polynomial-time approximation schemes can be found in Feige et al. (2002). In addition to the above, the NP-completeness of the 3-domatic partition problem was proofed for general graphs by Cardei et al. (2002) and still holds when restricted to planar bipartite graphs (Poon et al., 2012) and planar unit disk graphs (Nguyen and Huynh, 2007).

A natural way to produce a feasible solution to the MDDSP is to greedily construct dominating sets of preferably small cardinality with the ultimate goal to maximise the number of disjoint dominating sets that can be generated.[3] In this context, many greedy heuristic strategies have been proposed in the literature and experimentally tested, such as the ones given in Balbal et al. (2021), Cardei et al. (2002), Islam et al. (2009), Nguyen and Huynh (2007). They are discussed in detail in Section 4. Additionally, Landete and Sainz-Pardo (2022) presented an exact decomposition algorithm for finding a domatic partition on separable graphs, that is, graphs with at least one node (cut-vertex) whose removal produces two or more connected components. Finally, the only metaheuristic proposed so far in the literature is the CMSA presented in Rosati et al. (2023a). This approach was able to outperform the approaches for the MDDSP existing to date.

3. Problem description

Let $G = (V, E)$ be an undirected graph where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. A *dominating set* in G is a set of vertices $\mathcal{D} \subseteq V$ such that every vertex $v \in V \setminus \mathcal{D}$ is adjacent to at least one vertex $v' \in \mathcal{D}$. The decision problem of determining whether a dominating set of size $|\mathcal{D}| \leq K$ exists is NP-complete (Garey and Johnson, 1979), whereas the related *minimum dominating set* (MDS) problem, which requires finding the smallest dominating set in a given

[3] Remember that any solution to the MDDSP can be trivially transformed to a solution to the DPP by adding those vertices that do not belong to any of the disjoint dominating sets to one of the dominating sets of the MDDSP-solution.

graph, is NP-hard (Irving, 1991). In this work, we are interested in the *maximum disjoint dominating sets problem* (MDDSP), in which a valid solution $S = \{D_1, \dots, D_k\}$ consists of a collection of disjoint dominating sets D_i $(i = 1, \dots, k)$ of G, where disjoint means that $D_i \cap D_j = \emptyset$ for all $i \neq j \in \{1, \dots, k\}$. The objective function value $f(S)$ of a valid solution S is the number of disjoint dominating sets in S, that is, $f(S) := |S|$. The goal is to find a valid solution S^* that maximises f. This description applies to the DPP as well, with the supplementary condition that $\bigcup_{D \in S} D = V$.

In addition to its research significance from a theoretical and computational point of view, solving the MDDSP is relevant from a real-world application perspective. Its utility is found, for example, in heterogeneous multi-agent systems (Mesbahi and Egerstedt, 2010) and in wireless sensor networks (WSN). In particular, in WSNs disjoint dominating sets find applications in mechanisms for sleep-wake cycling (Cardei and Du, 2005; Cardei et al., 2002). The aim of such a mechanism is to prolong the lifetime of a battery-powered sensor network. In fact, the expected lifetime of a sensor network equipped with a disjoint-sets-based sleep-wake cycling mechanism is directly proportional to the number of disjoint dominating sets that can be found in the graph.

Note that finding at least one dominating set in a graph is always possible and trivial, given that the node set V of the graph is a dominating set of $G = (V, E)$. However, note that a solution $S = \{V\}$, which has an objective function value of $f(S) = |S| = 1$, is the worst solution that can be found, even though it exists in each input graph. Furthermore, every graph without isolated vertices contains at least two disjoint dominating sets (Ore, 1962). In general, the number of dominating sets on graphs is a number between 1 and $\delta(G) + 1$, where $\delta(G)$ is the minimum degree of all vertices in graph G. Indeed, $\delta(G)+1$ is a proven upper bound for the number of disjoint dominating sets in G (Cockayne and Hedetniemi, 1977). It is actually rather easy to verify this upper bound. Given a solution $S = \{D_1, \dots, D_k\}$ for the input graph G, any vertex $v \in V$ must be dominated by a different vertex in all $D_i \in S$. However, a vertex $v \in V$ with degree $deg(v)$ can only be dominated by itself or by any of its neighbours, that is, by at most $deg(v) + 1$ different vertices. Therefore, there will be no more than $\delta(G) + 1$ disjoint dominating sets in G. Nevertheless, this value, which is very easy to calculate and definitely useful when solving the problem in practice, does not imply that an actual solution with $\delta(G)+1$ disjoint dominating sets exists. That is to say, a solution with $\delta(G) + 1$ dominating sets is surely optimal, but there is no guarantee that the given input graph contains a solution of value $\delta(G) + 1$.

Finally, we introduce some general concepts on undirected graphs that are required later in this paper. The open neighbourhood of vertex v is $N(v) := \{u \in V \mid (u, v) \in E\}$, which represents the set of vertices adjacent to v in G. The closed neighbourhood of vertex v is $N[v] := N(v) \cup \{v\}$, that is, $N[v]$ includes all vertices adjacent to v and v itself. The number of neighbours of v corresponds to its degree, which is denoted by $deg(v)$. In other words, $deg(v) = |N(v)|$.

3.1. ILP formulation

A natural ILP model for the MDDSP on graph $G = (V, E)$ is based on binary variables $x_{vs} \in \{0, 1\}$ for every vertex $v \in V$ and every theoretically possible dominating set $s = 1, \dots, \delta(G)+1$. If v is in the sth dominating set, then $x_{vs} = 1$, zero otherwise. A second binary variable $y_s \in \{0, 1\}$ tells whether the sth dominating set is chosen to be part of the solution. Given these two sets of variables, the ILP model for MDDSP reads as follows.

$$\max \sum_{s=1}^{\delta(G)+1} y_s \tag{1a}$$

$$\sum_{s=1}^{\delta(G)+1} x_{vs} \leq 1 \qquad \forall v \in V \tag{1b}$$

$$\sum_{u \in N(v)} x_{us} \geq y_s - x_{vs} \qquad \forall v \in V, \forall s \in \{1, \dots, \delta(G)+1\} \tag{1c}$$

$$y_s \geq x_{vs} \qquad \forall v \in V, \forall s \in \{1, \dots, \delta(G)+1\} \tag{1d}$$

$$x_{vs} \in \{0, 1\}, y_s \in \{0, 1\} \qquad \forall v \in V, \forall s \in \{1, \dots, \delta(G)+1\} \tag{1e}$$

The objective (1a) maximises the number of dominating sets in the solution. Constraints (1b) ensure that the dominating sets are disjoint, requiring each vertex v to be assigned to at most one dominating set. Constraints (1c) ensure that the sets that are chosen in the solution are dominating sets. To evaluate this constraint, we make use of the concept of open neighbourhood, introduced in Section 3. Constraints (1d) ensure that nodes are only assigned to active dominating sets. Finally, Constraints (1e) define the binary nature of the variables.

This model, however, does not include any symmetry breaking constraints. Note that symmetric – and, therefore, redundant – solutions are obtained simply by permuting the sets of a solution. In that way, a solution is obtained with exactly the same sets, just that the set indices/names are different. In particular, given a solution S with $|S|$ sets, symmetric solutions are obtained by all possible $|S|$-permutations of the $\delta(G) + 1$ indices, that is, every solution S can be expressed in $\frac{(\delta(G)+1)!}{(\delta(G)+1-|S|)!}$ ways. Adding symmetry breaking constraints to the ILP model reduces significantly the size of the search space, at the expense of a higher number of constraints that lead to an increased complexity.

In the following, we present two additional ILP models that implement two different symmetry breaking strategies. Both are obtained by adaptations of the constraints proposed by Méndez-Díaz and Zabala (2008) for graph colouring. For sake of clarity, we name the model described above as ILP, and the two models with symmetry breaking constraints as ILP-SM$_1$ and ILP-SM$_2$.

3.1.1. Model 1 with symmetry breaking constraints

Model ILP-SM$_1$ imposes that the number of vertices assigned to the sth set must be greater or equal than the number of vertices in the $s+1$-th set. This is done by adding to Model ILP the following inequalities:

$$\sum_{v \in V} x_{v,s} \geq \sum_{v \in V} x_{v,s+1} \quad \forall s = 1, \dots, \delta(G) \tag{2}$$

Constraints (2) lead to an ordering of the sets by decreasing size, that is, set s must represent a set with a higher or equal cardinality than the sth set.

3.1.2. Model 2 with symmetry breaking constraints

Méndez-Díaz and Zabala point out in their work that Model ILP-SM$_1$ does not prevent symmetries that originate from permutations of the indices within sets of the same size, a scenario that is likely to arise in the MDDSP. This is solved in Model ILP-SM$_2$ by adding to Model ILP the following constraints:

$$x_{v_i,s} = 0 \qquad \forall s \geq i + 1 \tag{3a}$$

$$x_{v_i,s} \leq \sum_{k=s-1}^{i-1} x_{v_k,s-1} \qquad \forall i \in V \setminus \{1\}, \forall 2 \leq s \leq i - 1 \tag{3b}$$

Model ILP-SM$_2$ assigns to sets of vertices a set index that corresponds to the smallest index of all vertices in the set. This model, therefore, also breaks the symmetries that originate from permutations of the indices within sets of the same size.

3.2. Graphical problem illustration

Consider the undirected graph $G = (V, E)$, with 11 vertices ($|V| = 11$) and 17 edges ($|E| = 17$), displayed in Fig. 1. Vertices are labelled $v_1, \dots, v_{11}$, while edges are unlabelled. An optimal solution is $S = \{D_1, D_2, D_3\}$, where $D_1 = \{v_1, v_4, v_6, v_8\}$, $D_2 = \{v_2, v_5, v_{10}\}$, and $D_3 = \{v_3, v_7, v_9\}$. The three sets are represented in the figure with three different background colours. It is easy to verify that all these sets are dominating sets. We show it for D_1:

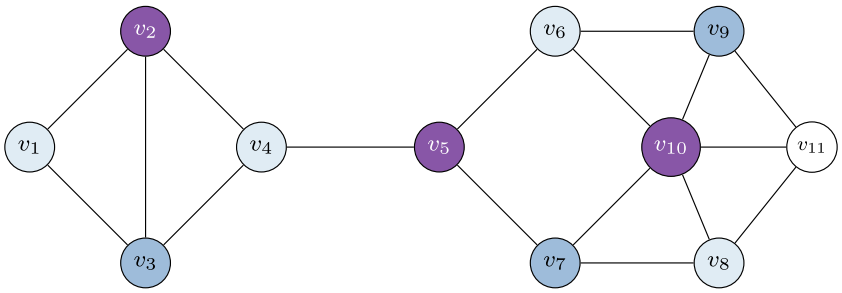

Fig. 1. A graph with 11 vertices and 17 edges. The upper bound for the domatic number is $\delta(G)+1=3$. Moreover, there exists an optimal solution with 3 disjoint dominating sets (as indicated by the vertices with a background colour different to white).

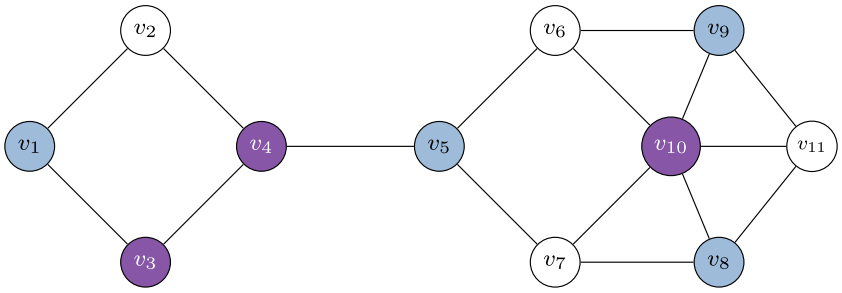

Fig. 2. A graph with 11 vertices and 16 edges. The upper bound for the domatic number is $\delta(G)+1=3$, but an optimal solution has only 2 disjoint dominating sets. One of the possible optimal solutions is indicated by blue and purple vertices. Uncoloured vertices do not belong to any dominating set of the displayed solution.

1. v_1 dominates the adjacent vertices v_2 and v_3
2. v_4 dominates v_2, v_3 and v_5
3. v_6 dominates v_5, v_{10} and v_9
4. v_8 dominates v_7, v_{10} and v_{11}

The same holds for $\mathcal{D}_2$ and $\mathcal{D}_3$. Furthermore the three sets are disjoint because $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$, $\mathcal{D}_1 \cap \mathcal{D}_3 = \emptyset$, and $\mathcal{D}_2 \cap \mathcal{D}_3 = \emptyset$. Hence, $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ is a solution to the MDDSP in graph G, with objective function value $f(\mathcal{S}) = 3$. If we add v_{11} to any of the dominating sets, $\mathcal{S}$ becomes a partition of V and, so, a valid and optimal solution also for the DPP.

Finally, we show through an example that an optimal solution with objective value $\delta(G)+1$, which we discussed above, does not exist in all graphs. Fig. 2 represents the same graph of Fig. 1, without the edge that connects vertices v_2 and v_3. In both examples, $\delta(G)+1 = 2+1 = 3$. We have seen above that a solution with value $f(\mathcal{S}) = 3$ exists for the graph in Fig. 1. However, a solution $\mathcal{S}' = \{\mathcal{D}_4, \mathcal{D}_5\}$, where $\mathcal{D}_4 = \{v_1, v_5, v_8, v_9\}$, and $\mathcal{D}_5 = \{v_3, v_4, v_{10}\}$, with a value of $f(\mathcal{S}') = 2$, is already an optimal solution for the graph in Fig. 2, and there are no solutions with three dominating sets. Solution $\mathcal{S}'$ is represented in Fig. 2 by means of different background colours for the vertices.

4. Greedy heuristics for the MDDSP

As mentioned in Section 2, a number of greedy heuristics have been presented in the related literature to solve the MDDSP. A greedy paradigm is a constructive approach that builds a solution from scratch. It generally starts with an empty solution and repeatedly adds one (or more) solution components to the current partial solution until a complete, valid solution is obtained. At each construction step, the next solution component(s) to be inserted in the incumbent partial solution are chosen according to a problem-dependent greedy function. To our best knowledge, the first greedy heuristic for the MDDSP was provided by Cardei et al. (2002). It is a vertex-colouring heuristic (henceforth called Colour-Dds) working in two phases and with a time complexity of $O(|V|^3)$. In the first phase, all vertices are coloured using the sequential Welsh-Powell colouring algorithm (Welsh and Powell, 1967) to generate all possible independent dominating sets. Each independent set is formed by vertices with the same colour, where colours are indicated by numbers. Remember that an independent set in a graph is a subset of vertices such that no two vertices in it are adjacent. In the second phase, for each independent set in ascending order of the colour identifiers, the algorithm checks whether it represents a dominating set or not. More specifically, if (1) the current independent set is not a dominating set and (2) there is no possibility to achieve that by adding some vertices from other independent sets with colour greater than the current one, then the termination condition of the algorithm is met. A pseudo-code for the Colour-Dds heuristic is shown in Algorithm 1.

Nguyen and Huynh (2007) also proposed three other deterministic greedy heuristics, namely Progressive Maximum Degree Disjoint Dominating Sets (P-Max), Progressive Minimum Degree Disjoint Dominating Sets (P-Min) and Random Lowest ID Disjoint Dominating Sets (R-Lid). These heuristics are two-step processes and adopt a similar heuristic mechanism in which a collection of disjoint dominating sets $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{|\mathcal{S}|}\}$ is formed by successively constructing $\mathcal{D}_k$ starting from an empty set for all $k = 1, \ldots, |\mathcal{S}|$. The construction of a dominating set $\mathcal{D}_k$ at step k for each of these greedy heuristics is done as follows. First, $\mathcal{D}_k$ is initialised to the empty set. Then, at each construction step, vertices of the input graph $G(V, E)$ are classified into three distinct sets with respect to $\bigcup_{i=1}^{k} \mathcal{D}_i$:

(i) BLACK vertices: vertices contained in $\bigcup_{i=1}^{k} \mathcal{D}_i$, that is, vertices inserted in one of the already generated dominating sets (including the current partial dominating set).
(ii) GREY vertices: vertices that are not BLACK but adjacent to some BLACK vertex.
(iii) WHITE vertices: all vertices from V that are neither BLACK nor GREY.

Given this classification, only WHITE vertices can be added to the current partial dominating set. In order to be able to make a choice, each WHITE vertex v is first evaluated by the following greedy function:

$$score^{P}(v) := \left| \left\{ u \in N(v) \mid u \text{ is a WHITE vertex with respect to } \bigcup_{i=1}^{k} \mathcal{D}_i \right\} \right| \quad (4)$$

Algorithm 1 Colour-Dds greedy heuristic for the MDDSP

input: a simple undirected graph $G = (V, E)$
output: a family of disjoint dominating sets $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k\}$

1: Let $\{v_1, v_2, \cdots, v_n\}$ be the vertices of V arranged in descending order of their degrees.
2: colour$[v_1] \leftarrow 1$
3: **for** $i = 2$ **to** n **do**
4: Colour v_i with the smallest possible colour (greater or equal to 1) not appearing in any neighbour v_j of v_i with $j < i$.
5: **end for**
6: $n_{\text{colours}} \leftarrow \max\{\text{colour}[v] \mid v \in V\}$
7: $\mathcal{D}_1 \leftarrow \{v \in V \mid \text{colour}[v] = 1\}$
8: $k \leftarrow 1$
9: **while** $(k < \min(\delta(G) + 1, n_{\text{colours}}))$ **do**
10: $\mathcal{D}_{k+1} \leftarrow \{v \in V \mid \text{colour}[v] = k + 1\}$
11: **for** each vertex $v \in V$ s.t. colour$[v] < k + 1$ **do**
12: **if** v is not dominated by a vertex with colour $k + 1$ **then**
13: **if** v has a neighbour u with largest colour greater than $k + 1$ **then**
14: colour$[u] \leftarrow k + 1$
15: $\mathcal{D}_{k+1} \leftarrow \mathcal{D}_{k+1} \cup \{u\}$
16: **else**
17: **go to** line 23 {Stop the algorithm}
18: **end if**
19: **end if**
20: **end for**
21: $k \leftarrow k + 1$
22: **end while**
23: **return** $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k\}$.

The three greedy heuristics from Nguyen and Huynh differ then in how a WHITE vertex is chosen at each construction step. In the case of P-Min and P-Max the next white vertex to be placed in $\mathcal{D}_k$ is the one with the maximum, respectively minimum, value of the greedy function $score^P()$. R-Lid, on the other side, does not make use of this greedy function. It simply prefers the WHITE vertex with the lowest index. When no further WHITE vertex can be added to $\mathcal{D}_k$, $\mathcal{D}_k$ may still not be a valid dominating set. Therefore, uncovered vertices are processed one after the other. In particular, such a vertex becomes covered if it has at least one GREY neighbour that can be added to $\mathcal{D}_k$. If it is not possible to cover all vertices, then the algorithm termination criterion is reached prior to the construction of the next dominating set $\mathcal{D}_{k+1}$. The pseudo-code of P-Max is presented in Algorithm 2. Note that P-Min is obtained from Algorithm 2 by replacing line 14 with the following instruction:

$$v^* \leftarrow \text{argmin}\{score^P(v) \mid v \in V\} \tag{5}$$

Note that we include the pseudo-codes of both Colour-Dds and P-Max in order to improve the overall quality and comprehensibility of our paper. We provide clarifications and additional details that were lacking in the original algorithm descriptions. This is important because the original textual descriptions of the algorithms were partially open to interpretation, particularly for readers without a strong background in graph colouring algorithms and domination. This causes some difficulty for such readers to correctly understand and implement the algorithms. Our intention is to enhance the clarity and accessibility of the algorithms by presenting them in a more easily understandable manner.

Two years later, an improved version of P-Max was introduced by Islam et al. (2009). This algorithm will henceforth be denoted as Iam, which is an acronym composed of the initials of the authors' surnames. In contrast to P-Max, Iam is a one-step construction process of time complexity $O(n^3)$ in which the set of feasible solution components

Algorithm 2 Greedy heuristic P-Max for the MDDSP

input: a simple, undirected graph $G = (V, E)$
output: a family of disjoint dominating sets $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_{k-1}\}$

1: colour$[v] \leftarrow$ WHITE for all $v \in V$
2: $dominating\_set\_flag \leftarrow$ **true**; $k \leftarrow 1$
3: **while** ($dominating\_set\_flag$ = **true**) **do**
4: **for** each vertex $v \in V$ **do**
5: covered$[v] \leftarrow$ **false**
6: **if** (colour$[v]$ = GREY) **then** colour$[v] \leftarrow$ WHITE **end if**
7: **end for**
8: $\mathcal{D}_k \leftarrow \emptyset$
9: $vertex\_cover\_flag \leftarrow$ **true**
10: **while** ($vertex\_cover\_flag$ = **true**) **do**
11: **if** $(\nexists\, v \in V$ **s.t.** colour$[v]$ = WHITE**) then**
12: $vertex\_cover\_flag \leftarrow$ **false**
13: **else**
14: $v^* \leftarrow \text{argmax}\{score^P(v) \mid v \in V\}$
15: $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{v^*\}$
16: **for** each neighbour u of v^* **do**
17: **if** (colour$[u]$ = WHITE) **then** colour$[u] \leftarrow$ GREY
18: covered$[u] \leftarrow$ **true**
19: **end for**
20: colour$[v^*] \leftarrow$ BLACK; covered$[v^*] \leftarrow$ **true**
21: **end if**
22: **end while**
23: **for** (each vertex $v \in \bigcup_{i=1}^{k-1} \mathcal{D}_i$ **s.t.** covered$[v]$ = **false**) **do**
24: **if** (v has a grey neighbour u **s.t.** $u \notin \bigcup_{i=1}^{k} \mathcal{D}_i$) **then**
25: $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{u\}$
26: **for** (each neighbour w of u) **do** covered$[w] \leftarrow$ **true end for**
27: colour$[u] \leftarrow$ BLACK
28: **else**
29: $dominating\_set\_flag \leftarrow$ **false**
30: **end if**
31: **end for**
32: **if** ($dominating\_set\_flag$) **then** $k \leftarrow k + 1$ **end if**
33: **end while**
34: **return** $\mathcal{S} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_{k-1}\}$.

includes all vertices of V that are not part of the set of BLACK vertices as defined previously for the case of P-Max. The employed greedy function is now defined as in Eq. (6) where ties are broken by choosing the vertex with the minimum number of BLACK neighbours. If the tie is still unresolved, the vertex with the lowest index is selected, similarly to what is done in R-Lid.[4]

$$score^P(v) := |\{u \in N[v] \mid u \text{ is a WHITE vertex with respect to } \mathcal{D}_k\}| \tag{6}$$

Finally, the currently best greedy algorithm (called Mdds-Gh) for the MDDSP was more recently presented by ourselves in Balbal et al. (2021). It was mainly developed for tackling a weighted variant of the MDDSP. Moreover, it was used as one of the principal components of a metaheuristic proposed for the same problem variant in Bouamama et al. (2022).

Mdds-Gh can be easily adapted to tackle the MDDSP by incorporating Iam's greedy score function depicted in Eq. (6). This adaptation also improves the handling of algorithm termination criteria and enhances overall performance. It is worth noting that Iam faces an issue with one of its stopping conditions that can be met even when there is a possibility of forming additional remaining dominating sets. This aspect has been addressed and solved in Mdds-Gh by using just one

[4] In reference to Eq. (6), remember that $N[v]$ represents the closed neighbourhood of v (that is, the set of neighbours of v including itself).

stopping condition. Unlike IAM, the algorithm checks, before beginning the current construction process, if there is at least one vertex such that all its neighbours from the closed neighbourhood are BLACK (part of previously constructed dominating sets). If this condition is met, no further dominating set can be found, and the algorithm terminates.

Note that the first five existing greedy heuristics were described here in detail. In contrast, MDDS-GH is comprehensively described in Balbal et al. (2021). Therefore, we decided against repeating its description in this section.

5. Multi-constructor CMSA for the MDDSP

In this work, we extend the preliminary CMSA approach for the MDDSP presented in Rosati et al. (2023a) by introducing the possibility to use multiple constructors. In the construction step of the Multi-Constructor CMSA, every solution is generated by a constructor chosen at random from a pre-defined portfolio. Hereby, each constructor has a weight value assigned, and the probability to be chosen depends on these weight values. In particular, we consider all six greedy heuristics from Section 4 as constructors for our CMSA approach. In order to produce (possibly) different solutions at each call to a constructor, they are used in a probabilistic way, which work as follows: at every greedy step, the classical deterministic greedy function is applied with a probability equal to the value of the parameter $0 \leq d_{\text{rate}} < 1$.[5] Otherwise, a candidate list containing the first c_{list} candidates, sorted by value of their greedy function from best to worst, is built, and a solution component is drawn uniformly at random from the list.

Our choice of CMSA for tackling the MDDSP is motivated by the fact that this metaheuristic has already shown a high potential for related optimisation problems in graphs. Examples include the minimum capacitated dominating set problem (Pinacho-Davidson et al., 2019) and the maximum happy vertices problem (Lewis et al., 2019). Furthermore, CMSA has been shown to perform well for real-world problems such as the prioritised pairwise test data generation problem (Ferrer et al., 2021) or the bus driver scheduling problem with complex break constraints (Rosati et al., 2023b).

5.1. Algorithmic details

The ILP model for the MDDSP from Section 3.1 uses certain constraints to ensure that the generated sets are both disjoint and dominating. These constraints are rather challenging, even for high-performance ILP solvers. Therefore, for our CMSA, we employ another paradigm based on the separation between (1) the generation of a large number of feasible dominating sets and (2) the subsequent selection of a collection of those sets such that the sets in the collection are vertex-disjoint. Theoretically, if we had the means to enumerate the full collection $\mathcal{C}$ of all possible dominating sets of input graph $G = (V, E)$, an optimal solution to the MDDSP could be obtained by solving the following set packing ILP formulation:

$$\max \sum_{D \in \mathcal{C}} x_D \tag{7a}$$

$$\text{s.t.} \sum_{\{D \in \mathcal{C} | v \in D\}} x_D \leq 1 \qquad \forall\, v \in V \tag{7b}$$

$$x_D \in \{0, 1\} \qquad \forall\, D \in \mathcal{C} \tag{7c}$$

This ILP model is based on a binary variable x_D for each dominating set $D \in \mathcal{C}$, whereby a value of $x_D = 1$ means that D is chosen to be part of the solution. Constraints (7b) ensure that each vertex of G is present in at most one of the chosen dominating sets. In this way, the chosen dominating sets are pairwise disjoint.

[5] d_{rate} stands for "determinism rate".

Algorithm 3 Multi-Constructor CMSA for the MDDSP

input: a graph $G(V, E)$, values for n_{sols}, d_{rate}, c_{list}, age_{limit}, t_{exc}
output: a family of disjoint dominating sets $\mathcal{S}_{\text{bsf}} = \{D_1, D_2, \cdots, D_k\}$

1: $t \leftarrow 1$
2: $(p_{1,t}, \ldots, p_{K,t}) \leftarrow \left(\frac{1}{K}, \ldots, \frac{1}{K}\right)$
3: $\mathcal{S}_{\text{bsf}} \leftarrow \emptyset$; $\mathcal{C}' \leftarrow \emptyset$
4: **while** $f(\mathcal{S}_{\text{bsf}}) < \delta(G) + 1$ **and** CPU time limit not reached **do**
5: $\quad\hat{\mathcal{C}} \leftarrow \emptyset$
6: $\quad$**for** $i \leftarrow 1, \ldots, \frac{n_{\text{sols}}}{|V|}$ **do**
7: $\quad\quad h \leftarrow$ ChooseConstructor($\mathcal{H}$,$(p_{1,t}, \ldots, p_{K,t})$)
8: $\quad\quad \mathcal{S}_{\text{cur}} \leftarrow$ Construct(G,h); $\hat{\mathcal{C}} \leftarrow \hat{\mathcal{C}} \cup \mathcal{S}_{\text{cur}}$
9: $\quad\quad$**if** $f^{\text{lex}}(\mathcal{S}_{\text{cur}}) > f^{\text{lex}}(\mathcal{S}_{\text{bsf}})$ **then** $\mathcal{S}_{\text{bsf}} \leftarrow \mathcal{S}_{\text{cur}}$ **end if**
10: $\quad$**end for**
11: $\quad$**for** all $D \in \hat{\mathcal{C}}, D \notin \mathcal{C}'$ **do**
12: $\quad\quad age[D] \leftarrow 0$
13: $\quad\quad \mathcal{C}' \leftarrow \mathcal{C}' \cup D$
14: $\quad$**end for**
15: $\quad\mathcal{S}_{\text{exc}} \leftarrow$ ApplyExactSolver($\mathcal{C}', t_{\text{exc}}$)
16: $\quad$**while** $r(\mathcal{S}_{\text{exc}}) = 1$ **do**
17: $\quad\quad \mathcal{S}_{\text{exc}} \leftarrow$ ApplyRepairProcedure($\mathcal{S}_{\text{exc}}$)
18: $\quad$**end while**
19: $\quad$**if** $f^{\text{lex}}(\mathcal{S}_{\text{exc}}) > f^{\text{lex}}(\mathcal{S}_{\text{bsf}})$ **then** $\mathcal{S}_{\text{bsf}} \leftarrow \mathcal{S}_{\text{exc}}$ **end if**
20: $\quad(p_{1,t+1}, \ldots, p_{K,t+1}) \leftarrow$ ReinforceProb($(p_{1,t}, \ldots, p_{K,t})$,$\mathcal{C}'$,$\mathcal{S}_{\text{exc}}$,$\alpha$,$\tau$)
21: $\quad$Adapt($\mathcal{C}'$, $\mathcal{S}_{\text{exc}}$, age_{limit})
22: $\quad t \leftarrow t + 1$
23: **end while**
24: **output:** $\mathcal{S}_{\text{bsf}}$

However, in practice, there is no efficient way to enumerate all dominating sets of a reasonably large graph. And even if there was one, the size of $\mathcal{C}$ would be too large for the above ILP model to be solvable by nowadays' ILP solvers. On the contrary, we can efficiently generate a subset (or sub-instance) $\mathcal{C}' \subset \mathcal{C}$, containing a subset of all the possible dominating sets of G. The resulting ILP model where $\mathcal{C}'$ replaces $\mathcal{C}$ is tractable for a Mixed Integer Programming (MIP) solver, provided that the size of $\mathcal{C}'$ is kept reasonably small.

This idea constitutes the core of our CMSA approach for the MDDSP. The full CMSA loop, represented also graphically in Fig. 3, works roughly as follows:

1. CONSTRUCT: build heuristically a certain number of dominating sets.
2. MERGE: merge the dominating sets from these solutions with the sub-instance $\mathcal{C}'$.
3. SOLVE: solve the set packing based ILP model on the sub-instance $\mathcal{C}'$.
4. ADAPT: delete from $\mathcal{C}'$ those dominating sets that do not appear in good solutions.

Algorithm 3 provides the full pseudo-code of our Multi-Constructor CMSA algorithm for the MDDSP. First of all, the weights $(p_{1,t}, \ldots, p_{K,t})$ of the K constructors are initialised to the same value $1/K$, to give them all the same probability of being chosen at the first iteration. Moreover, the sub-instance $\mathcal{C}'$ and the best solution found so far $\mathcal{S}_{\text{bsf}}$ are initialised to empty sets.

The main loop of CMSA starts at line 4, and it is terminated when either a total time limit is reached, or in case a best solution $\mathcal{S}_{\text{bsf}}$ such that $f(\mathcal{S}_{\text{bsf}}) = \delta(G) + 1$ is found. At each CMSA iteration, the CONSTRUCT and MERGE steps are repeated until $n_{\text{sols}}/|V|$ solutions are generated; see lines 6–14. In the construction procedure, the multi-constructor plays a crucial role. First, at line 7 one of the available constructors is chosen by means of a biased random selection. Hereby, constructors are weighted according to their corresponding probabilities. Then, a new

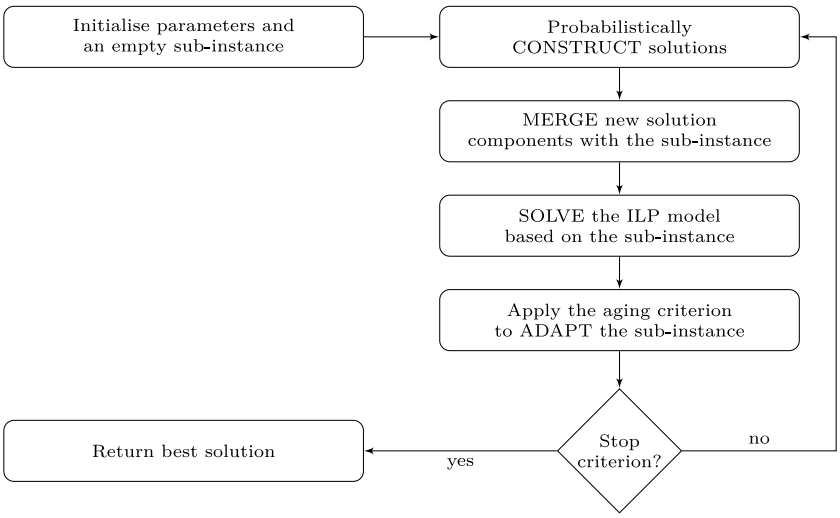

Fig. 3. General CMSA flow.

solution is generated with the chosen constructor (line 8). Naturally, every solution S_{cur} is compared with the incumbent S_{bsf}, using the lexicographic objective function presented in Section 5.3, as it sometimes may happen that a constructor is able to provide a new best solution. Then, the corresponding MERGE step is performed in lines 11–14. All the new dominating sets found in the $n_{sols}/|V|$ constructed solutions are added to C', and their $age[D]$ is initialised to zero.

Lines 15—20 contain the SOLVE phase. First of all, at line 15, CPLEX solves the set packing based ILP model corresponding to subinstance C'. The time limit is set to t_{exc} CPU seconds, or to the remaining time budget if this is less than t_{exc} CPU seconds. The output S_{exc} is the best solution returned by CPLEX within the given CPU time. In our experiments, we observed that the exact solver is often able to prove the optimality of S_{exc} for C'. In other words, often CPLEX does not spend all the allotted computation time. A residual function $r(S_{exc})$ is checked at lines 16–18 to verify whether at least one additional dominating can be generated using the vertices not included in any dominating set of S_{exc}; see Section 5.3 and Rosati et al. (2023a) for the definition of the function $r()$. If this is the case, a heuristic procedure, labelled ApplyRepairProcedure in Algorithm 3, is activated iteratively to build the additional dominating set(s), using the vertices in $V \setminus S_{exc}$. Then, at line 20, the probabilities of the constructors are reinforced according to the rules presented in Section 5.4.

Finally, the ADAPT phase takes place at line 21. First, the dominating sets $D \in S_{exc}$ generated by the repair procedure that are not already included in the sub-instance C', are added to C'. Second, the age values of all dominating sets from S_{exc} are reset to zero. Third, the age values of all remaining dominating sets from C' are incremented by one. Finally, all dominating sets $D \in C'$ with $age[D] > age_{limit}$ are removed from C'.

5.2. Parameters

Apart from graph G, our algorithm takes as input the values for seven parameters. Note that parameter values are determined through a statistically-principled tuning procedure, as discussed in Section 6.2. Five of these parameters, namely n_{sols}, d_{rate}, c_{list}, age_{limit}, and t_{exc}, are traditional CMSA parameters, while parameters α and τ, which are needed for the reinforcement learning mechanism applied to the constructor probabilities, are specific to our Multi-Constructor CMSA. The full list of parameters is contained in Table 1.

We aim to find one single parameter value setting that works reasonably well for the whole range of problem instances. However, in preliminary experiments, we observed that the number of solution constructions allowed per iteration was very sensitive to the size of the input graph. In case of too many solution constructions, the resulting sub-instances have too many dominating sets, that is, the corresponding ILP models have too many variables, which makes it nearly impossible for CPLEX to find a good – or even optimal – solution within the fixed CPU time limit of t_{exc} seconds. This is because the number of vertices of the input graph ($|V|$) determines the number of rows of the corresponding ILP model, while the number of dominating sets in the sub-instance determines the number of columns. For this reason, we also observed that for smaller graphs many more solution constructions could be afforded when compared to larger graphs. Therefore, we finally decided to make the number of solution constructions allowed per algorithm iteration inversely proportional to $|V|$ by setting it to $\frac{n_{sols}}{|V|}$.

Table 1
Parameters for CMSA.

Param.	Explanation
n_{sols}	parameter that serves for the calculation of the number of solutions to be generated at each algorithm iteration, as explained below.
d_{rate}	determinism rate for solution construction.
c_{list}	length of the candidate list for those solution construction steps in which a non-deterministic choice is performed.
age_{limit}	limits the number of iterations a dominating set can remain in the sub-instance C' without being chosen by the exact solver for the best solution to the sub-instance.
t_{exc}	time limit (in seconds) for the application of the MIP solver at each iteration of CMSA.
α	learning rate employed for the learning of constructor probabilities, only for the Multi-Constructor variant (Section 5.4).
τ	minimum probability that can be reached by a constructor, only for the Multi-Constructor variant (Section 5.4).

5.3. Lexicographic objective function

The MDDSP is characterised by the fact that many distinct solutions with identical objective function value coexist. This is because, generally, there are many different solutions with the same number of dominating sets. As a consequence, the search landscape is characterised by the presence of wide plateaus (Watson, 2010). The problem that arises in the presence of plateaus is that a metaheuristic, which is implicitly guided by gradients in the search landscape, may get lost. An effective way to deal with such a situation is to use a lexicographic

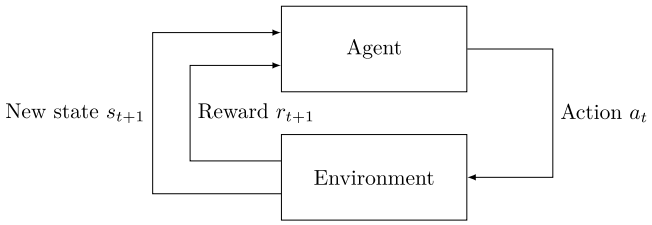

Fig. 4. Basic scheme of a reinforcement learning algorithm.

objective function $f^{\text{lex}}()$, using more criteria than just the objective function in order to differentiate between different solutions. This is done, for example, in Bruglieri and Cordone (2021), where the original objective function is used as a first criterion for comparing two solutions, and, only when the two solutions have the same original objective function value, a second criterion is evaluated to differentiate between them.

For our Multi-Constructor CMSA we maintain the same lexicographic objective function as proposed in Rosati et al. (2023a), based on the idea that the most promising solutions on a plateau are those that are nearest to building an additional dominating set with the unused vertices. More specifically, given two solutions S_1 and S_2, S_1 is said to be lexicographically better than S_2—that is, $f^{\text{lex}}(S_1) > f^{\text{lex}}(S_2)$—if and only if

1. $f(S_1) > f(S_2)$ **or**
2. $f(S_1) = f(S_2)$ **and** $r(S_1) > r(S_2)$

Hereby, the second criterion, $r(S)$, is a residual coverage function that calculates the fraction of the input graph G that can be covered with the vertices $V' \subset V$ that, in a given a solution S, are not assigned to any of the disjoint dominating sets. More specifically, $r(S)$ is defined as follows:

$$r(S) := \frac{\left|\bigcup_{v \in V'} N[v]\right|}{|V|} \tag{8}$$

In addition, the residual coverage function is used in CMSA (see lines 16–18 of Algorithm 3) to discover if there are hidden dominating sets contained in the set of unused vertices regarding CPLEX solution S_{exc}. For a more detailed discussion on the lexicographic objective function and on $r(S)$ we forward the reader to Rosati et al. (2023a).

5.4. Adaptive learning of constructor probabilities

As explained in Section 5.1, in our Multi-Constructor CMSA, the selection of the constructors is stochastic. For this purpose, a Multi-Constructor CMSA that utilises K constructors makes use of a vector of probabilities $(p_1, \ldots, p_K)$, with $\sum_{i=1}^{K} p_i = 1$, where p_i is the probability of the ith constructor. During the CONSTRUCT phase of CMSA, before the generation of a new solution, one of the constructors is selected by means of a biased random selection (*roulette wheel selection*), according to the given probabilities. Then, the next solution is built using the selected constructor. A trivial choice would be to assign the same probability $1/K$ to all constructors. However, it is reasonable to believe that not all constructors are equally useful and that higher probabilities should be assigned to the most promising ones. For this reason, we propose an online strategy for learning these probabilities, based on a *reinforcement learning* approach. Reinforcement learning is a learning paradigm where an intelligent agent learns the best policy for its actions by interacting with the environment. The agent takes actions in order to maximise its notion of *cumulative reward*. Fig. 4 shows a simple representation of the mechanism.

The application of reinforcement learning to metaheuristics is an active research stream. For instance, a similar mechanism is employed in Adaptive Large Neighbourhood Search (Ropke and Pisinger, 2006), where two vectors of probabilities are adapted during the search for the destroy operators, respectively the repair operators. Examples are found also in Evolutionary Algorithms, for example, with the application of a learning mechanism based on a Multi-Armed Bandit for adapting the weights of different operators (Fialho et al., 2010). More recently, Q-learning (Watkins, 1989) has been explored within various metaheuristics, with the aim of learning an ordering, or an execution sequence, of the operators, rather than their relative weights. We also recall similar examples regarding Iterated Local Search (Alicastro et al., 2021) and Hyper-Heuristics (Mischek and Musliu, 2022). In addition to the quality of the solution in terms of the objective function value, Hu and Raidl (2006) take into account also the time spent by each operator in a Variable Neighbourhood Search algorithm. Additional examples of weight-adaptive metaheuristics are found in Queiroz dos Santos et al. (2014) and Canca et al. (2017), for what concerns local search based metaheuristics, and in Wang et al. (2015) and Nagra et al. (2019) for population-based techniques. In general, the marriage of machine learning with combinatorial optimisation is considered a promising research field, as Bengio et al. (2021) point out in their methodological overview. A recent survey of the applications of machine learning specifically in metaheuristics is found in Karimi-Mamaghan et al. (2022).

In our Multi-Constructor CMSA, all probabilities are initialised to $1/K$ at the first iteration. Then, at every iteration, constructors receive a reward and the probabilities are updated according to the relative rewards obtained by the constructors. Learning is synchronised with CMSA iterations because the rewards can be computed only after the SOLVE phase, when a measure of the quality of the constructors is available. Eq. (9) shows the learning mechanism. Given a real parameter $\alpha \in [0, 1]$, called *learning rate*, we calculate the new probability $p_{i,t+1}$ of constructor i at iteration $t + 1$ as:

$$p_{i,t+1} = (1 - \alpha) \cdot p_{i,t} + \alpha \cdot r_{i,t} \tag{9}$$

where $r_{i,t}$ is the reward obtained by constructor i at iteration t and $p_{i,t}$ is the weight/probability of constructor i at the beginning of the iteration, inherited from the previous CMSA iteration (respectively, from the initial probabilities, at the first iteration). Given that $\sum_i^K p_{i,0} = \sum_i^K \frac{1}{K} = 1$, at the first iteration, and $\sum_i^K r_{i,t} = 1$, at every iteration t, this learning criterion also guarantees that the weights of the constructors sum exactly 1 along the whole process, without the need for normalisation or further adjustments. This is rather easy to proof. If, by hypothesis, at any generic iteration t, $\sum_i^K p_{i,t} = 1$ and $\sum_i^K r_{i,t} = 1$, then:

$$\sum_i^K p_{i,t+1} = \sum_i^K (1 - \alpha) \cdot p_{i,t} + \alpha \cdot r_{i,t} = \tag{10a}$$

$$(1 - \alpha) \cdot \sum_i^K p_{i,t} + \alpha \cdot \sum_i^K r_{i,t} = \tag{10b}$$

$$(1 - \alpha) \cdot 1 + \alpha \cdot 1 = 1 \tag{10c}$$

The parameter α sets the pace of the learning process. As shown in Eq. (11), the first component of Eq. (9) is the memory of past rewards, or, in other terms, the accumulated experience from the search, and it is weighted with $1 - \alpha$. The second component is the importance assigned to the reward obtained in the last iteration, which is weighted with α. If $\alpha = 1$, there is no memory in the process, and the next probabilities are influenced exclusively by the current rewards. On the opposite, if $\alpha = 0$, no learning is involved, and the probabilities of the constructors are never changed. Therefore, a proper tuning of α is quite critical for the effectiveness of the learning mechanism.

$$p_{i,t+1} = \underbrace{(1 - \alpha) \cdot p_{i,t}}_{\text{memory of past rewards}} + \underbrace{\alpha \cdot r_{i,t}}_{\text{learning from last rewards}} \tag{11}$$

Another concept to be taken into account is the *learning batch*, the number of iterations between subsequent learning steps. In Multi-Constructor CMSA, the learning batch coincides with exactly one CMSA

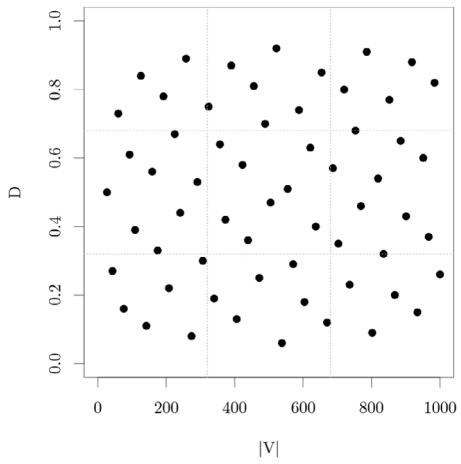

Fig. 5. The 60 Hammersley points sampled from the instance space and used for the generation of the problem instances. As the instance space has only two dimensions, it is possible to represent the generated points on a bi-dimensional plane.

iteration. This choice is quite natural because our metric, respectively reward function, for evaluating the quality of different constructors can only be calculated after the SOLVE phase of each iteration. In fact, the choice of the reward function $r_{i,t}$ is a critical design choice. The quality of the solutions generated by the constructors alone might not be a good predictor of their usefulness for CMSA. Therefore, a better metric is needed. Our idea for the MDDSP is to reward the constructors proportionally to the number of generated dominating sets that are found in the solution S_{exc} obtained by the exact solver. If the same dominating set was generated by multiple constructors, all of them are granted a reward. Eq. (12) shows the formula for assigning the rewards. It accounts also for the dominating sets generated heuristically by the repair procedure that are possibly found in S_{exc}, that are removed from the count. The rewards are calculated as follows:

$$r_{i,t} = \frac{|S_{\text{exc}} \bigcap C'_i|}{|S_{\text{exc}} \bigcap C' \setminus C'_r|} \tag{12}$$

where C'_i is the set of the dominating sets in solution S_{exc} that have been found by the ith constructor, and C'_r contains the dominating sets built by the repair procedure. This guarantees that $\sum_i^K r_{i,t} = 1$.

A downside of the proposed learning scheme is that a constructor that, even after an increasing number of consecutive iterations, has not received any reward may have its probability decreased to a very low value. This would make it nearly impossible that this constructor is selected, thus making it extremely improbable that it ever gets any future reward. Especially when α is set to high values, a constructor faces the risk to be kicked out extremely fast, even though it might be useful to have it later in the search. To guarantee a better trade-off between exploration and exploitation, it is advisable to put into place a mechanism that avoids such an extreme evolution of the probabilities. In order to deal with this issue, the learning scheme employs a minimum threshold τ for all constructor probabilities. To achieve this, Eq. (9) is rewritten as follows:

$$p_{i,t} = \max\{\tau, (1-\alpha) \cdot p_{i,t-1} + \alpha \cdot r_{i,t}\} \tag{13}$$

Naturally, when the minimum threshold τ is applied, the probabilities of the remaining constructors are also re-balanced such that the probabilities always sum to 1. Please note also that the minimum threshold τ implies that the probabilities are upper-bounded by a maximum value of $1 - (K-1)\tau$.

6. Experimental results

We denote the three versions of CMSA studied in this work as CMSA, CMSA-L_{all}, and CMSA-$L_{1,2}$. The first one is the standard CMSA, which only uses one constructor (MDDS-GH) and no reinforcement learning mechanism. The second one employs the six constructors described before in this paper, while the third one uses only the most promising constructors, namely, MDDS-GH and IAM. We perform a thorough comparison on a large instance space, that involves the two Multi-Constructor CMSA variants against the standard CMSA and the heuristics from the literature. We present and compare also the results obtained by CPLEX.

Our software was implemented in C++17 and compiled on Ubuntu 20.04.5 with g++ 9.4.0 in -O3 mode. We run all experiments on a machine equipped with an Intel Xeon Processor (Cascadelake), with 16 cores and a clock frequency of 2.4 GHz. We employed a maximum of one single core per experiment, including the call to exact solver within CMSA. For the comparison, we implemented also the six heuristics in C++17 and compiled and ran them under the same experimental setting. Finally, we implemented the ILP model for the MDDSP for CPLEX 20.1, through its C++ interface for Linux.

6.1. Instances

In our preliminary work (Rosati et al., 2023a) we considered random graphs with up to 250 vertices. In contrast, for the experimental evaluation of this paper, we propose a larger set of graphs with up to 1000 vertices and a wider range of densities. Furthermore, we make use of two additional categories of networks for benchmarking our algorithm: Watts–Strogatz, having small-world properties (Watts and Strogatz, 1998), and Barabási–Albert graphs, which are scale-free networks (Barabási and Albert, 1999). A generation procedure based on the Hammersley point set (Hammersley and Handscomb, 1964) ensures that all areas of the instance space are equally represented. The instance space is characterised by two dimensions: the number of vertices $|V|$ and the density D. Given an undirected graph $G = (V, E)$, the definition of D is as follows:

$$D = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)} \tag{14}$$

For the three categories of graphs, the problem instances were then generated according to the following procedure:

1. Selection of 60 pseudo-random points $(|V|, D)$ with the Hammersley sampling procedure, with $10 \leq |V| \leq 1000$, and $0.05 \leq D \leq 0.95$. This domain covers practically the whole instance space, avoiding at the same time the generation of graphs that are too small and with density values too extreme. The outcome of the generation procedure is shown graphically in Fig. 5. Additionally, information on the 60 points is provided in Tables 4, 5, and 6, in columns $|V|$ and D.
2. For every point, 20 graphs were randomly generated, resulting in a total of 1200 graphs, for each model. The graphs have $|V|$ vertices, and D is used as the probability for every possible edge to be present. Nonetheless, the distribution of the edges will vary in different kinds of graphs. For random graphs, the graph generator iterates over all pairs of nodes (u, v), and establishes stochastically if an edge between them exists, with probability D. Watts–Strogatz networks are similar, but in addition, vertices are clustered. In Barabási–Albert networks the degree follows a scale-free (or power law) distribution, that will result in few vertices being much more connected than most other vertices. Given that D is used as input parameter for probabilistic generation, the actual density of the generated graphs can be slightly different from the required one. However, the deviation is minimal: as a double-check, we verified that the actual average density

Table 2
Classification of graphs by size and density.

Size		Density	
Name	Feature	Name	Feature
Small	$\|V\| < 334$	Sparse	$D < \frac{1}{3}$
Medium	$334 \leq \|V\| < 667$	Medium	$\frac{1}{3} \leq D < \frac{2}{3}$
Large	$667 \leq \|V\|$	Dense	$\frac{2}{3} \leq D$

for each group of instances is identical to the required one, if rounded to two decimal digits.

We employed the same procedure also for the generation of an additional set of instances for parameter tuning. This additional set utilises a higher number of Hammersley points on the same instance space, with of a single instance per point, and is composed exclusively of random graphs. Table 2 introduces a classification of the instances depending on their size and density. The borders between regions are visible also in the background grid in Fig. 5.

The reasons for the inclusion of additional types of graphs compared to our preliminary study are twofold. On the one hand, random graphs are not realistic in many contexts. In contrast, Watts–Strogatz and Barabási-Albert networks are often used in network science, for modelling both natural and social phenomena. On the other hand, we aim to study the adaptability of CMSA to unseen graphs with different characteristics, where "unseen" refers to the fact that these special types of graphs are not included in the training set for parameter tuning. Finally, note that in Rosati et al. (2023a) we considered also random geometric graphs. However, we decided against using them in this work, because CMSA rather easily solved problem instances with up to 5000 vertices to optimality.

Finally, all instances – for tuning and for the final experimental evaluation – are available online for download, together with an instance and solution validator, at https://bitbucket.org/maximum-disjoint-dominating-sets-problem/maximumdisjointdominatingsets-instances.

6.2. Parameter tuning

We performed the parameter tuning with *irace*, which is a package for automatic algorithm configuration based on iterated racing (López-Ibáñez et al., 2016). The time limit given to the three CMSA variants for each application was $\frac{|V|}{2}$ CPU seconds, identical to the one which will later be used for the final experimental evaluation. This allows more time for larger graphs, in which the constructors take longer for the generation of solutions. Table 3 summarises the parameters involved in the tuning, the considered domains, and the outcome of the tuning procedure, for the three versions of CMSA. In particular, the tuning procedure considered four CMSA parameters (n_{sols}, d_{rate}, c_{list}, age_{limit}) and the hyperparameters α and τ, needed for the reinforcement learning procedure. Note that parameter t_{exc} was fixed to 15 CPU seconds according to the outcome of preliminary exploratory tuning tests as well as on domain-specific expertise. The choice of fixing one parameter in advance aimed at a simplification of the tuning process. Note also that the choice of setting the number of solution constructions per iteration to $n_{\text{sols}}/|V|$ was determined as being superior to a static setting of the number of solution constructions by the results of a dedicated tuning session with *irace*. Parameters d_{rate}, α and τ are real-valued, with an allowed precision of two digits behind the comma, while n_{sols}, c_{list}, and age_{limit} are natural numbers. We tuned separately the parameters of the three algorithms CMSA, CMSA-L$_{\text{all}}$, CMSA-L$_{1,2}$. Despite of the fact that values for the parameters of CMSA were already available from our preliminary work, we retuned them too, because of the new, larger set of instances. The previous values were tuned on graphs with up to 250 vertices, and with a tighter range of densities, which would be a clear overfitting on a particular region of the extended instance space. The parameter tuning was realised on the dedicated training set comprised only of random graphs, as described in Section 6.1, and the tuning budget was set to 5000 experiments for CMSA-L$_{\text{all}}$ and CMSA-L$_{1,2}$, and to 3000 experiments for CMSA, which has fewer parameters. This resulted in a total computation time of 367 h for CMSA-L$_{\text{all}}$ and CMSA-L$_{1,2}$, and 220 h for CMSA. Thanks to the availability of a cluster with multiple nodes, having 16 cores each, we could parallelise the experiments, so that the elapsed time needed for the tuning was approximately one day. By contrast, the total computational time needed for the validation was 2590 h. Less formal preliminary experiments, which are naturally conducted during the design and the implementation of the algorithm, are not accounted in the calculation.

Table 3
CMSA parameters, the considered domains for parameter tuning, and the finally determined parameter values.

Parameter	Domain	CMSA	CMSA-L$_{\text{all}}$	CMSA-L$_{1,2}$
n_{sols}	{2500, 100000}	84798	79560	86865
d_{rate}	[0.60, 1.00]	0.99	0.98	0.97
c_{list}	{2, 3, …,50}	9	3	2
age_{limit}	{2, 3, …,30}	4	4	11
α	[0.00, 1.00]	–	0.98	0.30
τ	[0.00, 0.10]	–	0.04	0.08

From the results in Table 3 we can observe that the two parameters related to learning, that only affect CMSA-L$_{\text{all}}$ and CMSA-L$_{1,2}$, are the ones that show the most important differences (with respect to the allowed value domains). In the case of CMSA-L$_{\text{all}}$ a very high value is obtained for α. This already indicates that some constructors are much more useful than others. Moreover, it can already be deduced from the first learning batch which constructors deserve higher weights. By the way, the final value of 0.04 for τ indicates that a certain grade of exploration is required anyway. On the other hand, CMSA-L$_{1,2}$ requires a setting of $\alpha = 0.30$ and $\tau = 0.08$, which implies a more moderate learning pace and a higher minimum threshold: both constructors are producing useful solution components. The values of the other parameters are rather similar for the three CMSA variants.

6.3. Results

Experimental results obtained by CMSA, CMSA-L$_{\text{all}}$, CMSA-L$_{1,2}$ and by the six deterministic greedy heuristics are displayed in Table 4 for random graphs, in Table 5 for Watts–Strogatz networks and in Table 6 for Barabási-Albert networks. Each table row corresponds to one of the 60 Hammersley points and shows averages over 10 independent runs (concerning the CMSA variants) for each of the 20 problem instances produced for the corresponding Hammersley point. Henceforth, we will call the 20 problem instances belonging to a specific Hammersley point an instance group. Best values within instance groups are marked in bold. No results in bold in a given row indicates that the best values are found by one of the ILP models. For all types of graphs, results allow the observation that all three CMSA variants outperform the heuristics on all instance groups. There are, however, some instance groups that appear to be easier to solve, given that all considered CMSA variants obtain exactly the same results. Interestingly, these ties are more frequent in instances with very low or very high density, regardless of their size. A summary of the comparative performances of the algorithms is provided in Table 8. CMSA performs very well on all graphs, including those types that were not included in the tuning. Among CMSA variants, the best performer is CMSA-L$_{1,2}$. Interestingly, the advantage of CMSA-L$_{1,2}$ appears to be greater on Watts–Strogatz and Barabási-Albert networks.

We do not report on computational times, because the stopping criterion is the same for all CMSA variants, and the times employed by the six greedy heuristics are negligible, as it is consistently below one second. The total computation times of the CMSA variants range from 13.5 s to 500 seconds. Sometimes, however, a CMSA run is

Table 4
Numerical results obtained for random graphs.

Instances		CMSA			Heuristics						Improvement	
$\|V\|$	D	CMSA	CMSA-L_{ALL}	CMSA-$L_{1,2}$	MDDS-GH	IAM	P-MAX	P-MIN	R-LID	COLOUR	(%)	(n)
27	0.50	**8.00**	7.96	7.96	6.20	6.10	5.75	4.90	5.30	5.55	29.03	1.80
43	0.27	6.32	6.26	6.26	5.20	4.40	4.65	3.95	4.25	4.40	21.54	1.12
60	0.73	**23.00**	**23.00**	**23.00**	21.50	20.95	18.10	15.30	16.40	16.50	6.98	1.50
76	0.16	5.70	5.70	5.70	4.80	4.00	4.15	3.75	3.95	3.95	18.75	0.90
93	0.61	**28.89**	28.72	28.80	25.20	24.20	19.90	16.65	18.05	18.45	14.64	3.69
109	0.39	**19.25**	19.13	19.22	17.40	16.95	13.70	11.30	12.60	12.90	10.63	1.85
126	0.84	**60.12**	60.02	60.04	54.35	53.75	43.20	36.40	38.75	39.80	10.62	5.77
142	0.11	6.78	6.80	6.82	6.10	5.15	5.05	4.15	4.55	4.60	11.80	0.72
159	0.56	**39.03**	38.83	**39.03**	36.05	35.60	28.00	23.10	25.50	25.75	8.27	2.98
175	0.33	23.76	23.77	**23.78**	22.30	21.55	17.00	13.95	15.65	15.75	6.64	1.48
192	0.78	**64.10**	**64.10**	**64.10**	62.65	61.95	52.30	43.65	48.15	48.65	2.31	1.45
208	0.22	18.79	**18.82**	18.81	17.65	16.95	13.60	10.90	12.10	12.35	6.63	1.17
225	0.67	**65.98**	65.54	65.77	59.85	59.30	46.90	38.65	42.60	43.65	10.24	6.13
241	0.44	41.16	**41.20**	41.18	39.35	38.90	29.60	24.65	27.25	27.50	4.70	1.85
258	0.89	128.28	**128.30**	128.26	118.65	118.85	92.00	75.45	81.95	82.80	7.95	9.45
274	0.08	9.06	**9.11**	**9.11**	8.25	7.35	6.65	5.20	6.00	6.05	10.42	0.86
291	0.53	57.78	57.76	**57.82**	55.95	55.55	42.80	35.15	39.20	39.35	3.34	1.87
307	0.30	34.53	34.54	**34.61**	33.25	32.35	24.60	20.05	22.35	22.85	4.09	1.36
324	0.75	107.99	107.86	**108.00**	101.30	100.90	76.70	62.90	69.60	70.05	6.61	6.70
340	0.19	24.94	24.94	**24.98**	23.75	22.75	17.60	14.10	15.70	16.15	5.18	1.23
357	0.64	**87.93**	87.84	87.82	85.30	84.65	64.90	53.05	58.70	59.50	3.08	2.63
373	0.42	**56.77**	56.76	56.75	54.90	54.20	40.85	33.20	37.00	37.40	3.41	1.87
390	0.87	148.51	147.96	**148.27**	142.90	142.65	120.05	100.00	110.75	111.85	3.93	5.61
406	0.13	**20.67**	20.63	20.66	19.45	18.60	14.40	11.45	12.95	13.15	6.27	1.22
423	0.58	**87.44**	87.42	87.42	85.30	85.10	65.15	53.45	59.40	60.00	2.51	2.14
439	0.36	55.41	**55.44**	**55.44**	53.55	52.95	39.65	31.85	35.80	36.35	3.53	1.89
456	0.81	**152.00**	**152.00**	**152.00**	149.35	149.05	118.25	97.05	107.20	107.85	1.77	2.65
472	0.25	41.93	**41.94**	**41.94**	40.35	39.65	29.55	23.75	26.75	26.90	3.94	1.59
489	0.70	**122.64**	122.59	122.59	121.15	120.35	96.10	79.35	88.35	88.65	1.23	1.49
505	0.47	**81.16**	81.11	**81.16**	79.40	78.80	58.80	47.90	53.50	54.25	2.22	1.76
522	0.92	**260.79**	260.74	260.72	246.45	248.60	188.15	152.70	167.25	168.80	4.90	12.19
538	0.06	12.93	**12.97**	12.96	11.80	10.75	9.05	7.30	8.05	8.30	9.92	1.17
555	0.51	95.84	95.84	**95.90**	93.65	93.50	70.40	57.10	63.65	64.50	2.40	2.25
571	0.29	56.06	56.10	**56.11**	54.55	54.10	39.80	31.90	36.00	36.40	2.86	1.56
588	0.74	**160.84**	157.96	160.07	154.55	154.25	122.45	101.35	112.80	113.50	4.07	6.29
604	0.18	38.03	38.03	**38.04**	36.95	36.00	26.75	21.40	23.95	24.40	2.95	1.09
621	0.63	137.00	137.06	**137.16**	133.95	133.85	100.95	82.75	92.10	93.05	2.40	3.21
637	0.40	84.15	84.16	**84.18**	82.40	82.05	60.05	48.50	54.95	55.10	2.16	1.78
654	0.85	**218.00**	217.99	**218.00**	216.50	216.00	178.40	147.00	163.30	164.60	0.69	1.50
670	0.12	**29.00**	**29.00**	**29.00**	27.90	27.15	20.25	15.95	18.15	18.65	3.94	1.10
687	0.57	132.83	**132.84**	132.82	130.35	130.30	96.50	78.50	87.80	89.05	1.91	2.49
703	0.35	79.97	79.94	**79.99**	78.30	77.60	57.25	45.80	51.60	51.95	2.16	1.69
720	0.80	239.26	236.78	**239.36**	228.25	229.70	171.20	139.10	154.95	156.35	4.21	9.66
736	0.23	55.96	55.98	**56.00**	54.65	53.65	39.50	31.30	35.60	35.80	2.47	1.35
753	0.68	**182.77**	182.72	182.74	179.35	179.90	132.80	109.30	121.55	122.40	1.60	2.87
769	0.46	113.27	**113.32**	113.28	110.95	110.80	81.95	66.20	74.75	75.20	2.14	2.37
786	0.91	324.62	322.62	**324.92**	308.60	310.50	250.95	208.75	233.25	234.40	4.64	14.42
802	0.09	26.02	**26.03**	26.02	25.30	24.45	18.20	14.70	16.55	16.85	2.89	0.73
819	0.54	141.38	**141.43**	**141.43**	138.80	138.65	103.85	84.85	95.05	96.05	1.89	2.63
835	0.32	84.58	**84.63**	84.62	83.00	82.20	59.95	48.15	54.50	54.85	1.96	1.63
852	0.77	233.58	232.53	**234.84**	228.65	229.00	181.30	149.05	166.55	167.60	2.55	5.84
868	0.20	56.54	56.56	**56.60**	55.30	54.55	39.85	31.80	35.55	36.20	2.35	1.30
885	0.65	189.34	189.38	**189.47**	186.55	185.80	142.35	116.80	130.80	131.45	1.57	2.92
901	0.43	121.50	121.50	**121.55**	119.45	119.00	87.10	70.65	79.20	79.80	1.76	2.40
918	0.88	**306.00**	**306.00**	**306.00**	305.05	304.20	259.40	215.60	238.60	241.05	0.31	0.95
934	0.15	**46.72**	**46.72**	46.70	45.35	44.80	32.50	25.90	29.30	29.85	3.02	1.37
951	0.60	185.90	185.92	**185.94**	183.35	183.30	135.95	110.65	123.70	125.20	1.41	2.59
967	0.37	110.25	110.22	**110.32**	108.60	108.00	78.50	63.45	71.80	72.00	1.58	1.72
984	0.82	**327.00**	323.32	326.99	315.35	317.60	235.55	192.65	213.80	216.10	2.96	9.40
1000	0.26	80.76	80.78	**80.80**	79.35	78.65	56.90	45.70	51.45	52.00	1.83	1.45

stopped before the computation time limit is reached, due to finding a provenly optimal solution. It can be said that the execution times of CMSA are about two orders of magnitude higher than the ones of the greedy heuristics. While this – at first sight – might result in an unfair comparison, we need to consider that greedy algorithms are simple procedures and that their speed advantage comes at the cost of much worse solutions. On the other hand, more sophisticated algorithms such as metaheuristics perform a much larger exploration of the solution space to find better solutions. Inevitably, they require a higher computational time. In our context, longer running times are fully justified by the obtained improvements over the greedy heuristics, which is indisputable on all instance groups and all graph categories. Indeed, the table column with the heading "Improvement (%)" in Tables 4, 5, and 6, shows the percentage gap between the best-performing CMSA and the best heuristic. The improvement is consistent for all instance groups. In particular, CMSA can find up to 30% more dominating sets

Table 5
Results obtained for Watts–Strogatz graphs.

Instances		CMSA			Heuristics						Improvement	
$\|V\|$	D	CMSA	CMSA-L_{ALL}	CMSA-$L_{1,2}$	MDDS-GH	IAM	P-MAX	P-MIN	R-LID	COLOUR	(%)	(n)
27	0.52	**9.00**	**9.00**	8.96	7.95	7.20	6.70	5.80	6.15	6.15	13.21	1.05
43	0.28	7.51	7.40	7.46	6.05	5.15	5.25	4.80	4.60	5.10	24.13	1.46
60	0.73	**24.20**	**24.20**	**24.20**	22.85	22.05	18.45	15.85	16.60	17.55	5.91	1.35
76	0.16	7.00	7.00	7.00	5.90	5.25	5.15	4.05	4.50	4.80	18.64	1.10
93	0.60	**29.67**	29.34	29.36	25.60	24.70	20.10	16.60	17.55	19.10	15.90	4.07
109	0.39	**20.60**	20.38	20.59	18.60	17.65	14.10	11.55	12.05	13.80	10.75	2.00
126	0.84	**61.26**	61.19	61.12	56.55	55.65	43.80	36.55	39.00	40.15	8.33	4.71
142	0.11	8.26	8.17	8.26	7.60	6.70	5.80	5.00	5.05	5.90	8.68	0.66
159	0.57	**39.98**	39.75	39.85	37.45	36.45	28.45	23.85	24.50	27.10	6.76	2.53
175	0.33	25.80	25.75	**25.84**	24.30	23.30	17.20	14.85	14.80	17.65	6.34	1.54
192	0.78	**64.45**	**64.45**	**64.45**	63.50	62.55	51.80	44.45	47.50	49.45	1.50	0.95
208	0.22	20.75	20.78	**20.84**	19.45	18.60	13.35	11.70	11.75	13.90	7.15	1.39
225	0.67	**66.88**	66.48	66.67	61.10	59.95	46.85	38.65	40.85	43.95	9.46	5.78
241	0.44	43.58	43.62	**43.65**	42.00	41.10	29.55	24.70	25.35	29.30	3.93	1.65
258	0.89	**127.61**	127.60	127.60	121.70	120.85	89.40	77.65	81.15	84.05	4.86	5.91
274	0.08	**11.00**	10.99	10.99	9.85	8.85	7.20	6.15	6.35	7.45	11.68	1.15
291	0.53	59.94	59.76	**60.08**	58.10	57.15	43.10	35.50	36.70	41.55	3.41	1.98
307	0.30	37.09	37.08	**37.11**	36.10	34.95	24.60	20.65	20.85	25.25	2.80	1.01
324	0.75	**108.00**	**108.00**	**108.00**	103.00	102.10	76.50	63.75	68.00	70.75	4.85	5.00
340	0.19	**27.00**	**27.00**	**27.00**	25.65	24.70	17.15	14.90	14.80	18.05	5.26	1.35
357	0.64	88.17	88.14	**88.24**	86.15	85.25	64.65	53.15	55.50	60.60	2.43	2.09
373	0.42	**59.08**	**59.08**	**59.08**	57.65	56.65	40.80	33.50	33.80	39.95	2.48	1.43
390	0.87	**157.70**	157.66	157.66	152.10	151.25	119.20	102.80	109.55	112.45	3.68	5.60
406	0.13	**22.38**	22.30	22.35	21.50	20.50	14.10	12.70	12.30	15.00	4.09	0.88
423	0.58	91.30	91.14	**91.36**	89.15	88.10	66.25	54.00	55.75	62.15	2.48	2.21
439	0.36	**59.02**	**59.02**	**59.02**	57.75	56.75	39.85	32.75	33.00	39.70	2.20	1.27
456	0.81	**152.00**	**152.00**	**152.00**	150.05	149.05	117.40	98.65	106.00	108.30	1.30	1.95
472	0.25	45.15	45.15	**45.19**	44.25	43.20	29.35	24.80	24.65	30.25	2.12	0.94
489	0.70	**123.08**	122.84	122.89	121.85	120.85	95.20	78.80	83.25	88.80	1.01	1.23
505	0.47	84.75	84.80	**84.84**	83.15	82.05	58.85	48.60	48.95	57.60	2.03	1.69
522	0.92	**258.32**	258.14	258.08	249.00	248.10	173.70	153.80	159.35	165.75	3.74	9.32
538	0.06	**14.90**	14.87	14.89	13.75	12.80	9.30	8.05	8.10	9.95	8.36	1.15
555	0.51	**102.17**	102.14	**102.17**	100.15	99.10	70.70	57.40	58.45	68.25	2.02	2.02
571	0.29	**60.88**	60.85	60.87	59.35	58.40	39.35	32.70	33.10	40.65	2.58	1.53
588	0.74	162.66	160.50	**162.88**	158.05	157.15	122.35	101.65	108.65	113.55	3.06	4.83
604	0.18	41.73	41.60	**41.78**	40.45	39.35	26.35	22.50	22.20	27.70	3.29	1.33
621	0.63	141.70	141.74	**141.82**	139.25	138.25	101.25	82.55	86.65	94.90	1.85	2.57
637	0.40	88.08	88.10	**88.13**	86.75	85.70	59.95	49.40	49.55	59.25	1.59	1.38
654	0.85	**218.05**	218.02	**218.05**	216.95	216.20	175.80	149.05	159.95	164.40	0.51	1.10
670	0.12	**32.00**	**32.00**	**32.00**	31.10	29.95	19.50	17.10	16.85	21.15	2.89	0.90
687	0.57	134.88	134.85	**134.92**	133.20	132.15	96.55	78.55	80.65	91.80	1.29	1.72
703	0.35	**85.08**	85.06	**85.08**	83.75	82.70	56.65	46.70	46.65	56.90	1.59	1.33
720	0.80	239.35	235.42	**239.61**	230.40	229.60	169.25	141.45	151.80	155.55	4.00	9.21
736	0.23	**61.26**	61.20	61.17	60.25	59.25	38.45	33.05	32.75	40.45	1.68	1.01
753	0.68	183.98	183.97	**184.00**	182.00	181.00	132.50	108.35	114.20	122.50	1.10	2.00
769	0.46	120.84	120.78	**120.88**	119.00	118.00	82.10	66.70	67.45	79.70	1.58	1.88
786	0.91	338.67	**338.70**	338.38	325.15	324.20	240.10	212.50	220.75	228.90	4.17	13.55
802	0.09	**29.00**	**29.00**	**29.00**	28.20	27.15	17.50	15.25	15.35	19.40	2.84	0.80
819	0.54	149.58	**149.60**	**149.60**	147.45	146.45	104.55	84.85	86.55	100.00	1.46	2.15
835	0.32	91.03	91.03	**91.04**	89.55	88.60	59.50	49.05	49.10	60.70	1.66	1.49
852	0.77	235.08	235.07	**235.25**	232.70	231.75	180.80	149.65	160.90	166.35	1.10	2.55
868	0.20	62.03	**62.04**	62.02	61.25	60.20	38.20	33.05	33.05	41.05	1.29	0.79
885	0.65	196.08	196.07	**196.18**	193.30	192.30	142.70	116.30	121.05	132.75	1.49	2.88
901	0.43	**126.81**	126.78	126.78	125.25	124.25	86.35	70.65	71.30	85.80	1.25	1.56
918	0.88	307.71	307.64	**307.74**	306.50	305.65	251.65	218.95	232.20	238.50	0.40	1.24
934	0.15	51.18	51.16	**51.24**	50.40	49.30	30.95	27.30	27.00	33.80	1.67	0.84
951	0.60	187.07	187.06	**187.10**	185.60	184.55	135.70	110.15	114.00	127.75	0.81	1.50
967	0.37	**117.39**	117.31	**117.39**	115.90	114.95	77.70	64.05	64.20	78.45	1.29	1.49
984	0.82	326.90	319.93	**327.14**	316.30	315.50	232.70	194.45	209.80	213.65	3.43	10.84
1000	0.26	**88.00**	**88.00**	**88.00**	86.80	85.75	55.05	46.95	46.45	58.30	1.38	1.20

in the smallest graphs. The relative improvement decreases as the graph size increases. The difference in terms of the absolute values, however, as shown in column "Improvement (n)", is remarkable, especially on dense graphs, with CMSA able to produce up to 14.42 more dominating sets on random graphs, and up to 13.55 more dominating sets on Watt-Strogatz networks. This is quite a notable result if we consider, for instance, the application to WSNs. Manufacturers of commercial sensor nodes declare operational lifetimes that range from days to months, or even years (Mak and Seah, 2009). Depending on the application area of a WSN, the potential lifetime of the network is obtained in relation to the number of disjoint dominating sets found on the graph. So, an extension of even a few percentage points in the lifetime of the WSN is translated into a gain of, at least, some days of autonomous functioning.

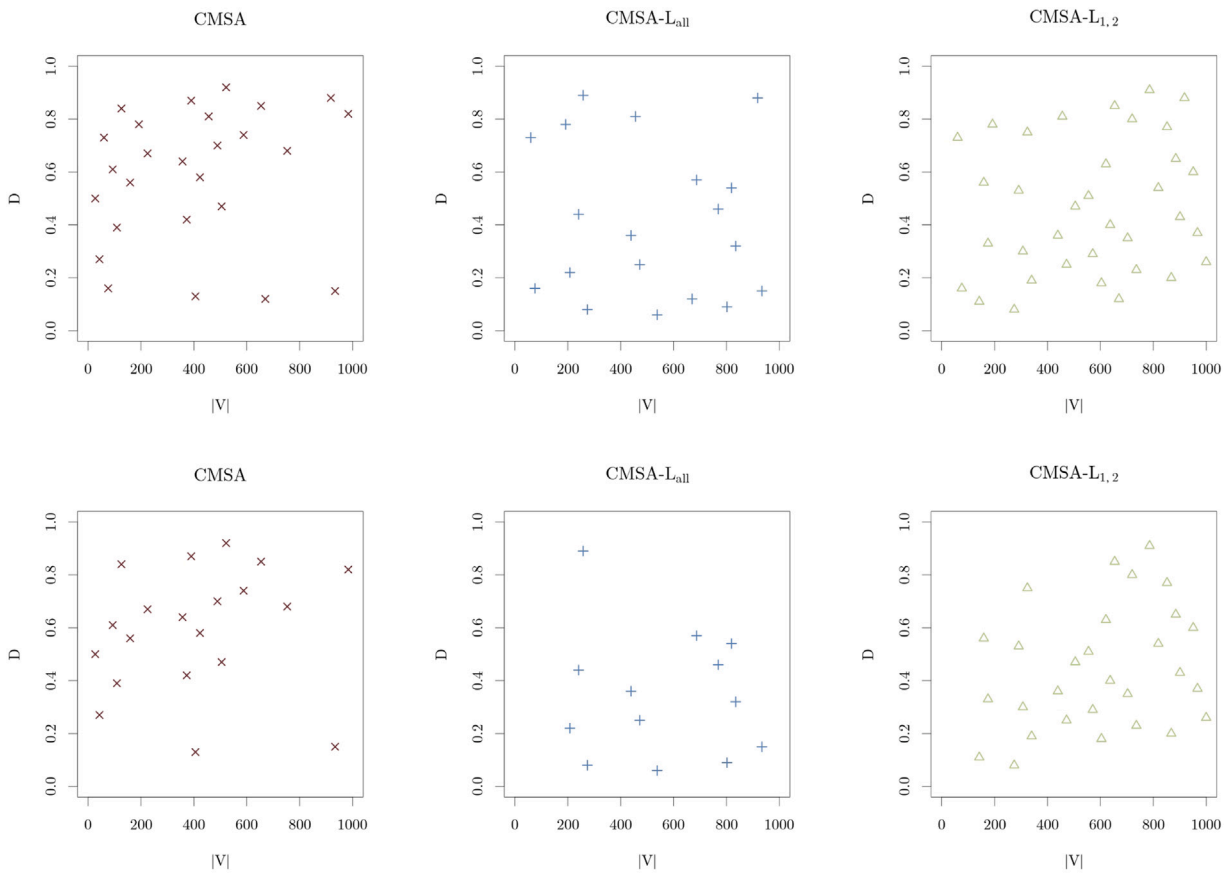

Fig. 6. Instance groups (Hammersley points) for which algorithms are the best-performing ones. The plots in the top row include all instances, while in the plots of the bottom row, the six points in which all CMSA variants perform equally well are removed.

Given that the running time granted to CMSA is much larger than the time needed by the greedy algorithms, the reader might wonder whether a heuristic algorithm for solving the MDDSP which is granted a comparable execution time would be able to compete with CMSA. However, all existing heuristics for the MDDSP are deterministic greedy algorithms, and non-deterministic metaheuristics have never been applied to the MDDSP. What can be done, however, is to run the best one of our randomised greedy heuristics (MDDS-GH) in a repeated way with the same computation time limit as used for CMSA, that is, with a time limit of $|V|/2$ seconds. This was done with the same values for parameters d_{rate} and c_{list} found in Table 3 in column CMSA. The output of this repeated, randomised greedy heuristic is the best solution found within $|V|/2$ seconds. Finally, we compare the obtained results with the ones of our best-performing CMSA variant (CMSA-L$_{1,2}$).

An analysis of the results shows that, in random graphs, CMSA-L$_{1,2}$ obtains on average 0.97 more dominating sets, which corresponds to an average percentage increment of 0.74%. In the context of Watt-Strogatz networks, CMSA-L$_{1,2}$ obtains on average 0.92 more dominating sets, corresponding to an average percentage increment of 0.72%. However, in the case of Barabási-Albert networks, the gap is significantly reduced, as CMSA-L$_{1,2}$ obtains on average just 0.05 more dominating sets, or, equivalently, an average percentage increment of 0.15%. This is not a surprising fact, considering the analogies between Barabási-Albert networks and random geometric graphs, that were studied in our previous work (Rosati et al., 2023a). Both are, indeed, characterised by locality and/or clustering of nodes. Moreover, we performed three paired Wilcoxon signed rank tests with continuity correction. The tests return that the difference in performance between CMSA-L$_{1,2}$ and the randomised, repeated greedy heuristic is statistically significant for random graphs and Watts–Strogatz networks, given the very low p-values ($< 2.2 \cdot 10^{-16}$). On the other hand, in the case of Barabási-Albert network, even though CMSA-L$_{1,2}$ generally produces better results, the gap is not statistically significant (the p-value is 0.079, above the significance threshold of 0.05), which means that we cannot reject the null hypothesis of equality of the true means of the two populations. These results suggest that the advantage of CMSA in the case of Barabási-Albert networks is limited to dense graphs (see Fig. 12), while its superiority is indisputable in the context of the other graph types.

We would also like to point out that the randomised, repeated version of greedy MDDS-GH that we used for comparison exists only because we developed it for its usage within CMSA, and that it can be seen as a particular use case of CMSA, with extreme parameter values such as a computation time of zero for solving the ILP model at each iteration. In the context of real-world applications, greedy algorithms are interesting because of their speed and usefulness in time-critical applications in which a user needs a response from the computer in fractions of seconds. In less time-critical applications, in which the decision maker has more time available for finding a good solution in advance, repeating in a loop the same randomised greedy algorithm for some time is generally not the best practice, especially if a metaheuristic (such as CMSA) that offers better exploration of the search space in the same running time exists.

In order to get more information on the behaviour of CMSA in different regions of the instance space, Fig. 6 shows six plots. All plots show the instance space, with the two axes that correspond to $|V|$ and D. The three plots from the top row indicate those instance groups (out of 60 Hammersley points) for which, from left to right, CMSA, CMSA-L$_{all}$, and CMSA-L$_{1,2}$ obtain the best results. The plots consider only random

Table 6
Results obtained for Barabási-Albert graphs.

Instances		CMSA			Heuristics						Improvement	
$\|V\|$	D	CMSA	CMSA-L_{ALL}	CMSA-$L_{1,2}$	MDDS-GH	IAM	P-MAX	P-MIN	R-LID	COLOUR	(%)	(n)
27	0.52	**7.00**	**7.00**	6.91	5.80	5.00	5.00	4.40	4.80	4.75	20.69	1.20
43	0.28	5.54	5.48	5.60	4.55	3.75	3.85	3.40	3.80	3.85	23.08	1.05
60	0.73	**17.58**	17.41	17.32	15.25	14.20	11.85	10.55	11.10	11.70	15.28	2.33
76	0.16	4.97	4.96	4.97	4.15	3.65	3.40	3.05	3.55	3.45	19.76	0.82
93	0.60	**19.77**	19.52	19.68	17.65	16.85	13.35	11.85	12.55	13.00	12.01	2.12
109	0.39	14.04	13.98	14.01	12.80	11.85	9.85	8.60	8.90	9.60	9.69	1.24
126	0.84	**36.21**	35.88	36.08	32.65	31.70	23.50	21.05	22.30	23.05	10.90	3.56
142	0.11	5.85	5.87	5.89	4.85	4.35	4.25	3.80	3.90	3.95	21.44	1.04
159	0.57	27.12	27.08	**27.15**	25.80	24.85	18.65	16.40	17.40	17.90	5.23	1.35
175	0.33	17.67	17.72	**17.74**	16.45	15.65	12.05	10.65	11.40	11.55	7.84	1.29
192	0.78	**45.19**	44.80	45.12	42.30	41.40	29.90	26.35	28.40	28.50	6.83	2.89
208	0.22	**14.00**	13.99	**14.00**	13.00	12.05	9.55	8.20	9.05	9.40	7.69	1.00
225	0.67	42.35	42.39	**42.46**	40.85	39.85	28.50	25.40	27.10	27.50	3.94	1.61
241	0.44	29.47	29.38	**29.50**	28.10	27.20	19.75	17.40	18.90	18.80	4.98	1.40
258	0.89	**66.42**	65.96	66.31	62.90	61.95	43.30	39.35	41.70	42.25	5.60	3.52
274	0.08	7.00	7.00	7.00	6.15	5.60	4.85	4.20	4.55	4.90	13.82	0.85
291	0.53	41.05	**41.06**	**41.06**	39.70	38.85	27.45	24.30	25.90	26.25	3.43	1.36
307	0.30	**25.00**	**25.00**	**25.00**	23.70	22.70	16.75	14.50	15.65	16.05	5.49	1.30
324	0.75	64.67	64.67	**64.78**	63.05	62.05	42.85	38.65	40.75	41.45	2.74	1.73
340	0.19	17.88	17.88	**17.89**	16.60	15.75	11.95	10.25	11.20	11.75	7.77	1.29
357	0.64	58.57	58.56	58.70	57.30	56.35	39.00	34.50	37.15	37.25	2.44	1.40
373	0.42	**39.98**	39.97	39.95	38.65	37.80	26.75	23.25	25.20	25.45	3.44	1.33
390	0.87	88.18	88.18	**88.25**	86.20	85.20	58.55	53.40	56.50	56.85	2.38	2.05
406	0.13	**14.40**	14.39	**14.40**	13.40	12.75	9.85	8.25	9.15	9.60	7.46	1.00
423	0.58	61.06	61.05	**61.09**	59.85	58.75	40.40	36.10	38.10	38.75	2.07	1.24
439	0.36	39.20	39.15	**39.24**	38.20	37.20	26.15	22.80	24.60	25.10	2.72	1.04
456	0.81	92.93	92.89	**92.94**	91.10	90.10	61.40	55.15	58.20	58.90	2.02	1.84
472	0.25	**30.00**	29.99	**30.00**	28.65	27.85	20.00	17.00	18.65	18.95	4.71	1.35
489	0.70	**83.08**	83.03	**83.08**	81.65	80.60	54.75	49.25	52.55	52.80	1.75	1.43
505	0.47	57.01	**57.02**	57.01	55.60	54.60	37.80	33.40	35.70	35.80	2.55	1.42
522	0.92	119.46	119.40	**119.52**	117.70	116.75	78.95	71.80	75.30	76.25	1.55	1.82
538	0.06	**9.01**	9.00	**9.01**	8.45	7.55	6.35	5.35	5.90	6.40	6.63	0.56
555	0.51	66.93	66.93	**66.94**	65.55	64.60	44.40	39.00	41.75	41.70	2.12	1.39
571	0.29	39.98	39.99	**40.01**	38.60	37.80	26.65	22.50	24.85	25.20	3.65	1.41
588	0.74	102.81	102.74	**102.88**	101.35	100.35	67.95	61.50	64.95	65.00	1.51	1.53
604	0.18	**27.00**	**27.00**	**27.00**	25.45	24.70	17.95	15.20	16.75	17.25	6.09	1.55
621	0.63	90.76	**90.78**	**90.78**	89.15	88.05	59.50	53.70	56.80	56.90	1.83	1.63
637	0.40	58.55	58.63	**58.68**	57.20	56.25	38.70	33.90	36.75	36.75	2.59	1.48
654	0.85	131.40	131.40	**131.46**	129.65	128.65	86.15	78.45	82.80	82.80	1.40	1.81
670	0.12	**20.32**	20.30	20.29	19.30	18.50	13.75	11.65	12.90	13.40	5.28	1.02
687	0.57	88.62	88.58	**88.72**	87.15	86.15	58.15	52.05	55.45	55.65	1.80	1.57
703	0.35	**56.00**	**56.00**	**56.00**	54.90	53.95	36.90	32.10	35.00	35.10	2.00	1.10
720	0.80	131.72	**131.81**	131.79	130.10	129.10	87.10	79.45	83.30	83.15	1.31	1.71
736	0.23	39.84	39.84	**39.91**	38.30	37.40	26.25	22.60	24.70	25.00	4.20	1.61
753	0.68	**115.03**	115.00	**115.03**	113.70	112.70	75.30	68.15	71.90	71.60	1.17	1.33
769	0.46	78.20	78.19	**78.21**	77.10	76.10	51.70	45.60	48.70	49.15	1.44	1.11
786	0.91	166.69	166.62	**166.70**	164.65	163.70	108.20	99.35	103.95	104.70	1.25	2.05
802	0.09	**18.18**	**18.18**	**18.18**	17.10	16.30	12.50	10.40	11.75	11.90	6.32	1.08
819	0.54	96.76	96.79	**96.80**	95.30	94.25	63.45	56.70	60.05	60.55	1.57	1.50
835	0.32	59.24	59.24	**59.29**	58.20	57.30	39.25	34.10	36.85	37.25	1.87	1.09
852	0.77	145.72	145.76	**145.78**	144.15	143.10	95.55	86.55	91.70	91.80	1.13	1.63
868	0.20	**40.00**	39.98	**40.00**	38.80	37.80	26.55	22.70	25.00	25.30	3.09	1.20
885	0.65	124.74	124.76	**124.81**	123.20	122.20	81.50	73.60	78.60	78.65	1.31	1.61
901	0.43	83.45	83.46	**83.53**	82.10	81.10	55.00	48.40	52.05	51.80	1.74	1.43
918	0.88	179.98	**180.01**	**180.01**	178.25	177.20	118.05	108.70	113.60	113.60	0.99	1.76
934	0.15	32.87	32.84	**32.90**	31.40	30.50	21.60	18.50	20.50	21.00	4.78	1.50
951	0.60	121.89	121.89	**121.92**	120.30	119.30	79.20	71.65	75.85	76.10	1.35	1.62
967	0.37	**76.61**	76.55	76.53	75.15	74.30	50.25	44.30	47.45	47.50	1.94	1.46
984	0.82	**175.10**	**175.10**	**175.10**	173.50	172.45	115.10	105.20	110.50	110.15	0.92	1.60
1000	0.26	56.58	**56.70**	56.61	55.60	54.60	37.35	32.30	35.20	35.80	1.98	1.10

graphs, although similar considerations could be drawn also for the other kinds of instances. In the three plots of the bottom row, the easy instances where all methods are able to obtain the same average results have been eliminated. Such instances are mostly concentrated in the low end or in the high end of the density range. From the plots, we can observe that CMSA performs well for small and medium instances. On the other side, it seems to lose its effectiveness in the context of larger instances, for which CMSA-$L_{1,2}$ obtains the best solutions. Beyond the graphical impression, this is confirmed with statistical significance by a non-parametric test, which is discussed in Section 6.4. On the other hand, Algorithm CMSA-L_{all} appears to be particularly weak on very dense graphs. To provide the reader with the complete picture, Fig. 7 represent the best performers in a single plot, by using the same three symbols for the three algorithms as those already used in Fig. 6.

Figs. 8 and 9 display examples of the evolution of the constructor probabilities, obtained from a run of CMSA-L_{all}, and from two distinct runs of CMSA-$L_{1,2}$. In Fig. 8 we can observe the effect of having a high learning rate, set to 0.98. The probabilities converge very sharply toward their final values, starting already in the first iteration. The two constructors MDDS-GH and IAM receive almost all the reward and the weights of the other constructors are decreased to the minimum threshold $\tau = 0.04$. Having said that, it can be observed that the

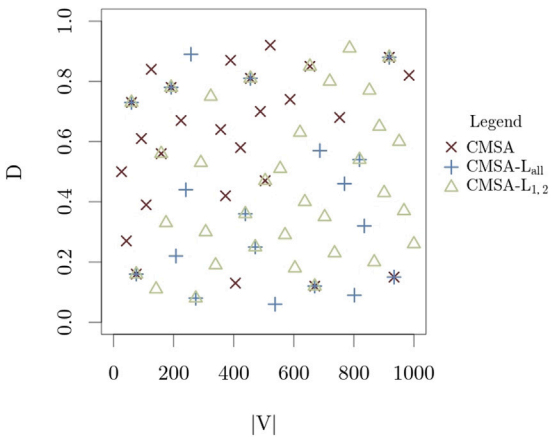

Fig. 7. Summary of the results shown in Fig. 6 in a single plot.

constructor P-Max receives some reward in the final phases of the search, which increases its relative weight. Fig. 9 contains two patterns of the evolution of the probabilities concerning Cmsa-L$_{1,2}$, from different runs on the same graph. Given that there are only two constructors and the probabilities keep summing to 1, the graphic is symmetric. In the two examples in Fig. 9, the probabilities of the two constructors converge toward similar values, although through a different evolution, especially at the beginning of the search process.

Finally, Table 7 displays the results obtained from the solution of the three models Ilp, Ilp-sm$_1$, and Ilp-sm$_2$ presented in Section 3.1, applied to all problem instances with $|V| < 300$, within a computation time limit of one hour. The reason why we only consider instances with $|V| < 300$ is twofold: on the one hand, the exact solver struggles in finding even any feasible solution for graphs of sizes slightly above 200 vertices. On the other hand, the application of CPLEX is quite memory intensive in the context of larger instances. Running such experiments would be, in any case, impractical. The table is organised as follows: for the three network types (RG: Random Graphs, WS: Watts–Strogatz Network and BA: Barabási-Albert networks), and for the three ILP models (Ilp: no symmetry breaking constraints, Ilp-sm$_1$ and Ilp-sm$_2$: the two models that implement symmetry breaking constraints), for every instance group the average objective function value, the execution time and the ratio of proven optimal solutions are provided. Bold values are used to mark those results which are of at least the same quality as the ones of CMSA. Indeed, there are some instances for which the ILP results are better than the ones of CMSA and the heuristics. This happens mainly on the smallest graphs ($|V| < 100$) and on very sparse graphs ($D < 0.1$). Usually, ILP solving requires much longer running time than CMSA, which is limited to $|V|/2$ CPU seconds, or the greedy heuristics which take less than a few seconds. There are a few cases, however, in which the ILP solver quickly finds a provenly optimal solution. This happens in the case of random graphs and for Barabási-Albert networks, but never for Watts–Strogatz ones. Interestingly, we do not observe any improvements due to the application of symmetry breaking constraints. The reasons for this might be twofold. First, modern commercial solvers like CPLEX already contain the implementation of techniques for detecting and handling symmetries, even when this is not explicitly specified in the model. Second, the reduction of the search space caused by symmetry breaking is not enough to outweigh the increased complexity of the model within the given computation time limit of one hour. The application of CPLEX to our Ilp model results in provenly optimal solutions for a total of 79 random graphs, 47 Watts–Strogatz networks, and 60 Barabási-Albert networks. We compare these solutions with the respective 790, 470 and 600 solutions found by Cmsa-L$_{1,2}$ in the 10 runs executed on each graph. On random graphs, Cmsa-L$_{1,2}$ is able to find the optimal solution, with identical cost as the one found by the ILP Model, in 648 out of 790 runs, with an average gap of 2.37%, computed on the total runs. On Watts–Strogatz networks, Cmsa-L$_{1,2}$ finds the optimal solution in 284 out of 470 runs with an average gap of 4.88%. Finally, on Barabási-Albert networks, Cmsa-L$_{1,2}$ finds the optimal solution in 302 out of 600 runs, with an average gap of 8.34%.

6.4. Statistical analysis

Additionally, R package `scmamp` (Calvo and Santafé Rodrigo, 2016) was used to facilitate and support the interpretation of the results from a statistical point of view. For this purpose, first, the results of the considered algorithms are compared simultaneously using the Friedman test for obtaining the rejection of the hypothesis that they all perform equally. Next, corresponding pairwise comparisons are performed using the Nemenyi post-hoc test (Garcia and Herrera, 2008) and, eventually, the output of this statistical analysis is presented by means of *critical difference* (CD) plots. The CD plot that compares all heuristic algorithms (CMSA variants and greedy heuristics) on all 3600 random, Watts–Strogatz and Barabási-Albert graphs together is shown in Fig. 10. The horizontal axis of a CD plot represents the range of algorithm ranks, while each of the vertical lines represents the average

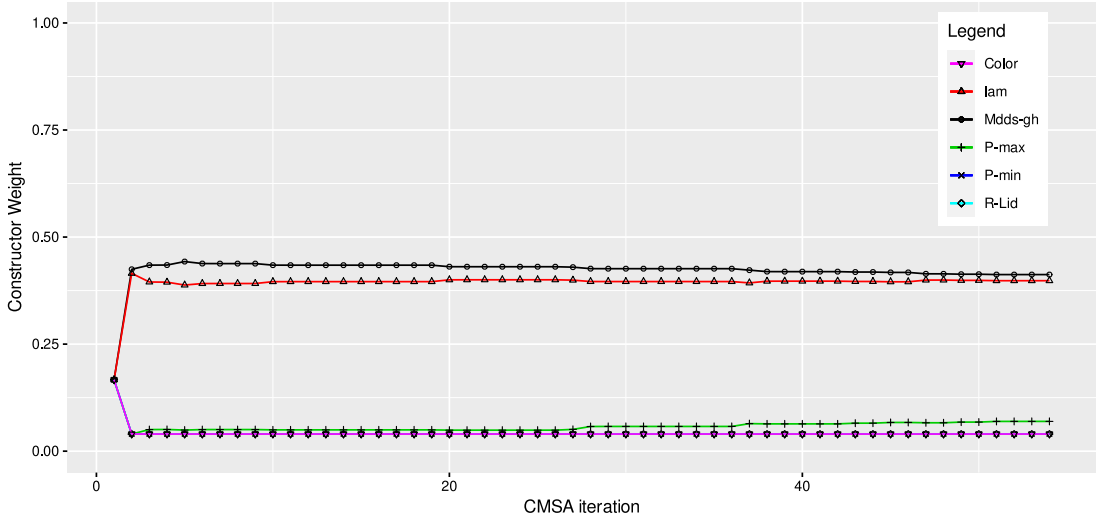

Fig. 8. Evolution of the probabilities of the six constructors during a run of Cmsa-L$_{ALL}$, for a random graph from the group ($|V| = 126$, $D = 0.84$).

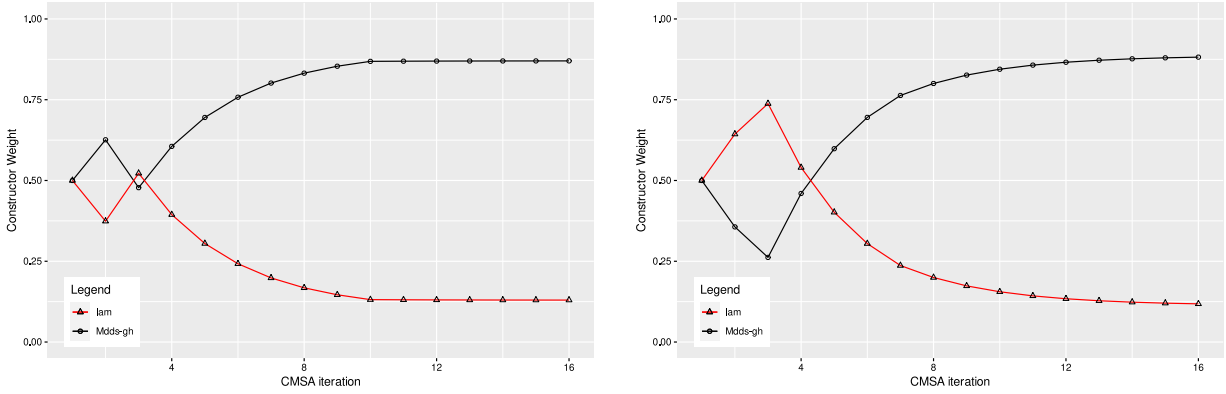

Fig. 9. Two examples of the evolution of the probabilities of the two constructors during runs of CMSA-L$_{12}$, for an instance from the group ($|V| = 934$, $D = 0.15$).

Table 7
Results obtained by the solution of three ILP models presented in Section 3.1 for random graphs as well as Watts–Strogatz and Barabási-Albert graphs.

	$\|V\|$	D	ILP			ILP-SM$_1$			ILP-SM$_2$		
			obj	time(s)	opt	obj	time(s)	opt	obj	time(s)	opt
	27	0.50	**8.00**	38	1.00	**8.00**	2205	0.40	**8.00**	347	0.95
	43	0.27	**6.45**	519	0.90	**6.45**	731	0.80	**6.50**	856	0.80
	60	0.73	**23.00**	3600	0.00	22.75	3600	0.00	22.85	3600	0.00
	76	0.16	**5.85**	1	1.00	**5.85**	1	1.00	**5.85**	1	1.00
	93	0.61	26.20	3600	0.00	24.15	3600	0.00	23.90	3600	0.00
	109	0.39	18.40	3600	0.00	16.75	3600	0.00	16.55	3600	0.00
	126	0.84	41.40	3600	0.00	13.95	3600	0.00	0.45	3600	0.00
	142	0.11	**7.20**	723	0.80	**7.10**	724	0.80	**7.20**	727	0.80
RG	159	0.56	1.00	3600	0.00	3.65	3600	0.00	0.00	3600	0.00
	175	0.33	16.80	3600	0.00	19.30	3600	0.00	8.40	3600	0.00
	192	0.78	31.80	3600	0.00	3.25	3600	0.00	0.00	3600	0.00
	208	0.22	11.40	3600	0.00	14.85	3600	0.00	10.80	3600	0.00
	225	0.67	1.00	3600	0.00	0.05	3600	0.00	0.00	3600	0.00
	241	0.44	1.00	3600	0.00	2.45	3600	0.00	0.00	3600	0.00
	258	0.89	0.00	3600	0.00	0.00	3600	0.00	0.00	3600	0.00
	274	0.08	9.10	2724	0.25	8.80	2733	0.25	8.75	2980	0.20
	291	0.53	0.10	3600	0.00	0.00	3600	0.00	0.00	3600	0.00
	27	0.52	**9.00**	168	1.00	**9.00**	2956	0.25	**9.00**	369	1.00
	43	0.28	**7.85**	1478	0.70	**7.85**	1951	0.50	**7.85**	1881	0.50
	60	0.73	**24.20**	3600	0.00	23.75	3600	0.00	24.00	3600	0.00
	76	0.16	**7.95**	1642	0.55	**7.80**	1823	0.50	**7.95**	1637	0.55
	93	0.60	27.00	3600	0.00	24.30	3600	0.00	24.35	3600	0.00
	109	0.39	19.90	3600	0.00	17.60	3600	0.00	17.75	3600	0.00
	126	0.84	58.80	3600	0.00	38.85	3600	0.00	0.75	3600	0.00
	142	0.11	**9.00**	3241	0.10	**8.75**	3241	0.10	**8.85**	3242	0.10
WS	159	0.57	1.00	3600	0.00	4.05	3600	0.00	0.00	3600	0.00
	175	0.33	15.20	3600	0.00	20.25	3600	0.00	1.00	3600	0.00
	192	0.78	31.35	3600	0.00	6.65	3600	0.00	0.00	3600	0.00
	208	0.22	12.95	3600	0.00	17.30	3600	0.00	8.50	3600	0.00
	225	0.67	1.00	3600	0.00	0.05	3600	0.00	0.00	3600	0.00
	241	0.44	1.00	3600	0.00	1.35	3600	0.00	0.00	3600	0.00
	258	0.89	0.00	3600	0.00	0.00	3600	0.00	0.00	3600	0.00
	274	0.08	9.25	3600	0.00	9.40	3600	0.00	9.40	3600	0.00
	291	0.53	0.00	3600	0.00	0.00	3600	0.00	0.00	3600	0.00
	27	0.52	**7.00**	1	1	**7.00**	37	1.00	**7.00**	1	1.00
	43	0.28	**6.00**	1	1	**6.00**	459	0.90	**6.00**	14	1.00
	60	0.73	17.00	3600	0	16.80	3600	0.00	17.00	3600	0.00
	76	0.16	**6.00**	20	1	**6.00**	1219	0.85	**6.00**	249	1.00
	93	0.60	19.00	3600	0	17.95	3600	0.00	18.80	3600	0.00
	109	0.39	**14.10**	3600	0	13.05	3600	0.00	13.85	3600	0.00
	126	0.84	32.55	3600	0	29.65	3600	0.00	29.35	3600	0.00
	142	0.11	**6.95**	3600	0	**6.75**	3600	0.00	**6.90**	3600	0.00
BA	159	0.57	22.40	3600	0	22.35	3600	0.00	8.10	3600	0.00
	175	0.33	17.20	3600	0	14.10	3600	0.00	15.65	3600	0.00
	192	0.78	1.00	3600	0	13.90	3600	0.00	0.00	3600	0.00
	208	0.22	13.90	3600	0	12.30	3600	0.00	12.65	3600	0.00
	225	0.67	1.00	3600	0	2.00	3600	0.00	0.00	3600	0.00
	241	0.44	1.25	3600	0	3.00	3600	0.00	0.00	3600	0.00
	258	0.89	1.00	3600	0	0.70	3600	0.00	0.00	3600	0.00
	274	0.08	**8.05**	3600	0	**7.05**	3600	0.00	**7.05**	3600	0.00
	291	0.53	1.00	3600	0	1.80	3600	0.00	0.00	3600	0.00

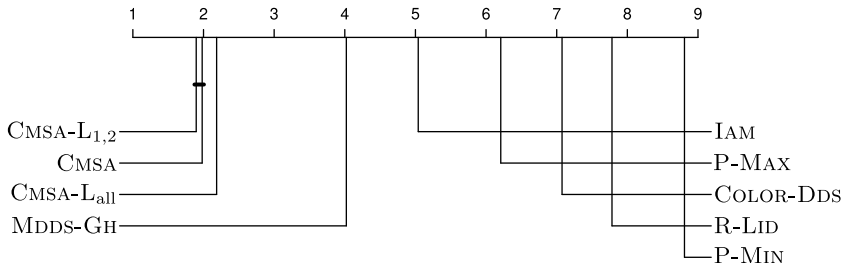

Fig. 10. Critical difference plot comparing all heuristic algorithms on all 3600 problem instances.

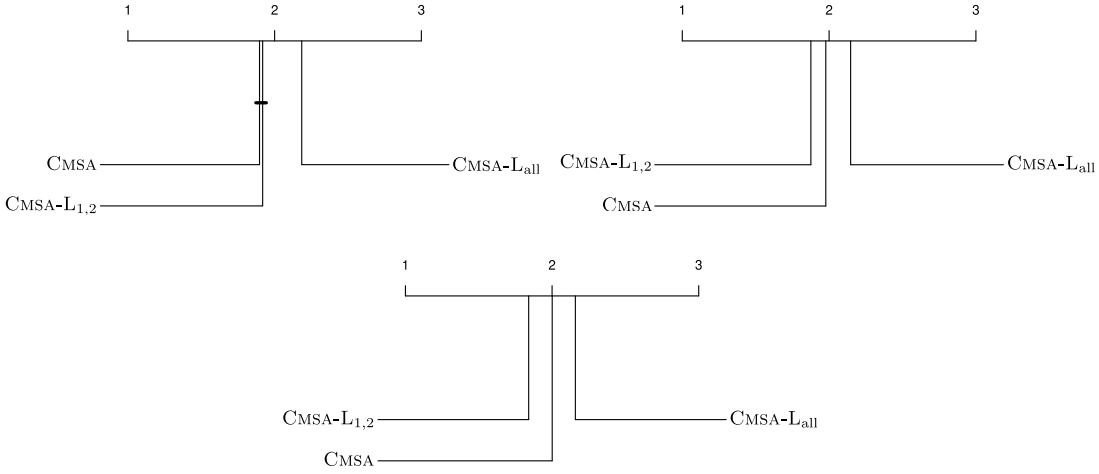

Fig. 11. Comparison depending on graph size. Top left: small graphs, top right: medium-sized graphs, bottom: large graphs.

Table 8
Performance of the algorithms and the ILP models for different types of networks, in terms of the number of instance groups for which the respective algorithms obtain the best average solution.

	RG	WS	BA	Sum	% best
CMSA	23	30	21	74	41.1%
CMSA-L$_{ALL}$	18	14	12	44	24.4%
CMSA-L$_{1,2}$	34	39	43	116	64.4%
CMSA (any)	58	57	55	170	94.4%
ILP	5	5	6	16	8.8%
ILP-SM$_1$	4	4	5	13	7.2%
ILP-SM$_2$	4	4	5	13	7.2%
MDDS-GH	0	0	0	0	0.0%
IAM	0	0	0	0	0.0%
P-MAX	0	0	0	0	0.0%
P-MIN	0	0	0	0	0.0%
R-LID	0	0	0	0	0.0%
COLOUR	0	0	0	0	0.0%

rank of the corresponding algorithm. Bold horizontal lines connecting algorithm markers indicate that the corresponding algorithms perform statistically equivalent – i.e. the critical difference is not greater than the significance level of 0.05 – concerning the considered set of problem instances.

The CD plot from Fig. 10 shows the following. First, the three CMSA variants outperform, with statistical significance, all six greedy heuristics, on the whole set of graphs. Second, the performance differences between all pairs of greedy heuristics are statistically significant. Third, CMSA-L$_{1,2}$ performs better – with statistical significance – than CMSA-L$_{ALL}$. Fourth, even though the average ranking of CMSA-L$_{1,2}$ is better than the one of CMSA and the average ranking of CMSA is better than the one of CMSA-L$_{ALL}$, no statistically significant difference can be found between CMSA-L$_{1,2}$ and CMSA.

Finally, we repeated the CD plot analysis limiting the set of considered instances to small, medium and large-sized graphs. The resulting CD plots (only concerning the CMSA variants) can be found in Fig. 11. They show that, while no statistical difference can be found between CMSA-L$_{1,2}$ and CMSA in the context of small graphs, CMSA-L$_{1,2}$ is found to outperform CMSA with statistical significance in the context of and medium-sized and large graphs. This confirms our observations from the numerical results (Tables 4, 5 and 6), and in part, from the analysis provided in Figs. 6 and 7.

In addition to the statistical comparison, in Fig. 12 we show the absolute improvements obtained by CMSA-L$_{1,2}$, which is the best performing CMSA, over the best greedy result. This is done for the three considered types of graphs. As we commented previously, the improvement is more noticeable on graphs with higher densities. For this reason, the graph density is shown on the x-axis, while the y-axis shows the improvement, measured as the difference between the dominating sets found by CMSA-L$_{1,2}$ and the dominating sets in the best greedy result. For each density point, a boxplot shows the distribution of the obtained improvements.

7. Conclusions

In this work, we presented state-of-the-art algorithms for solving the maximum disjoint dominating sets problem (MDDSP). The proposed algorithms are variants of a recent matheuristic called Construct, Merge, Solve & Adapt (CMSA). In addition to a standard variant of CMSA, we proposed the Multi-Constructor CMSA as a framework for integrating multiple construction heuristics into CMSA. This algorithm

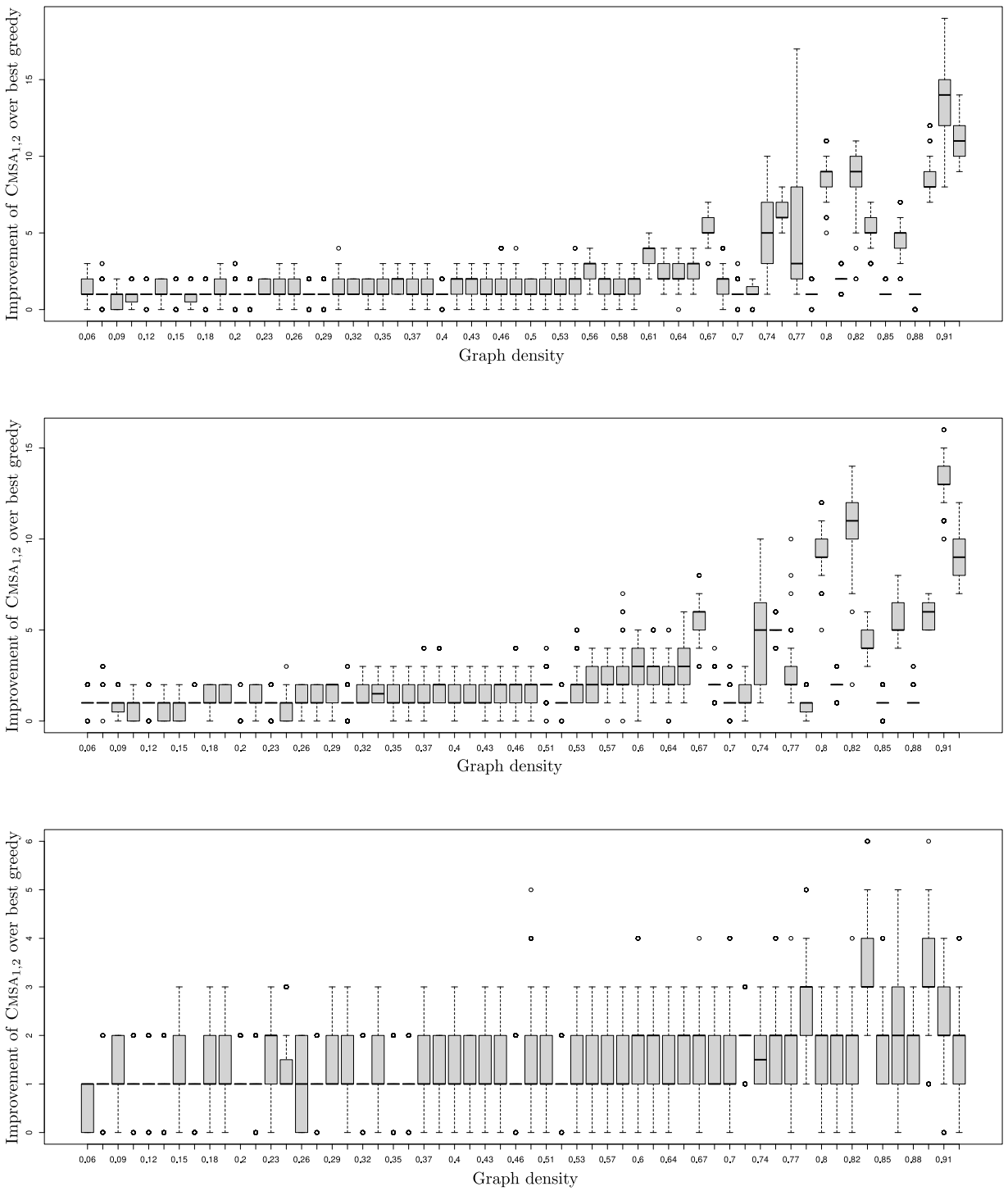

Fig. 12. Improvement obtained by CMSA-$L_{1,2}$ over the best greedy result, for instance groups, ordered by increasing density. From top to bottom: random graphs, Watts–Strogatz networks, Barabási-Albert networks.

uses a stochastically biased selection criterion and a reinforcement learning technique for learning the best probabilities for the constructors during the search process. Two variants of the Multi-Constructor CMSA were tested: the first one, CMSA-L_{all}, makes use of all available constructors, while the second one, CMSA-$L_{1,2}$, only utilises the two most promising constructors. We were able to show that all our CMSA variants outperform the greedy heuristics from the related literature on three distinct models of graphs: random, Watts–Strogatz and Barabási-Albert. Moreover, CMSA-$L_{1,2}$ generally performs best, with a statistically significant advantage over the other two CMSA variants in the context of medium-sized and large graphs, which are the most challenging ones. We also made an attempt to solve the MDDSP directly by means of applying a MIP solver to three ILP models, two of which implement symmetry breaking constraints. However, this did not yield satisfactory results, with the exception of very sparse graphs. Overall, CMSA-$L_{1,2}$ performs best on 116 out of 180 instance groups, which corresponds to 64.4% of the instances. When considering all CMSA variants together, they perform best on 94.4% of the instances, that is, 170 out of 180 instance groups. The remaining 10 instance groups are small and sparse graphs for which the ILP models perform best. We executed all the

experiments on a new set of graphs that we have generated from an instance space spanned by graph size (number of vertices) and graph density. This was done with a statistically sound procedure that guarantees that all regions of the instance space are represented in the sample.

Future work will consist in trying to come up with new constructors for the problem, and in investigating different reward functions or different learning schemes. For example, we might consider rewarding also the time that a constructor employs to generate the dominating sets, in order to favour those constructors that are faster. Another option would be to use an ϵ-greedy criterion for the diversification of the constructor choices, instead of a minimum threshold. We consider also evaluating the effects of setting fixed probabilities for the constructors and tuning them with a traditional offline tuning procedure, instead of employing a learning scheme. Another idea is a sequential selection criterion, in comparison with a stochastic one. Finally, we plan to extend the Multi-Constructor CMSA to other optimisation problems for which constructors of different types exist. Consider, as an example, job scheduling problems or packing problems.

CRediT authorship contribution statement

Roberto Maria Rosati: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Visualization. **Salim Bouamama:** Investigation, Software, Writing – review & editing. **Christian Blum:** Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link to the data is shared in the article.

Acknowledgements

The authors acknowledge Andrea Schaerf for his fruitful comments on the work and his help in reviewing the drafts of this manuscript. Additionally, the authors acknowledge TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215, for the support provided for this research. Furthermore, this work was supported by grant PID2022-136787NB-I00 funded by MCIN/AEI/10.13039/501100011033. Finally, the support provided by CINECA through grant ISCRA C IA4EVRP HP10CE285L is gratefully acknowledged by the authors. Finally, the authors acknowledge the anonymous reviewers for their constructive feedback that helped improve the quality of this work.

References

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. Wireless sensor networks: a survey. Comput. Netw. 38 (4), 393–422.
Alicastro, M., Ferone, D., Festa, P., Fugaro, S., Pastore, T., 2021. A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. Comput. Oper. Res. 131, 105272.
Balbal, S., Bouamama, S., Blum, C., 2021. A greedy heuristic for maximizing the lifetime of wireless sensor networks based on disjoint weighted dominating sets. Algorithms 14 (6), 170.
Barabási, A.L., Albert, R., 1999. Emergence of scaling in random networks. Science 286 (5439), 509–512.
Bengio, Y., Lodi, A., Prouvost, A., 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. European J. Oper. Res. 290 (2), 405–421.
Blum, C., Ochoa, G., 2021. A comparative analysis of two matheuristics by means of merged local optima networks. European J. Oper. Res. 290 (1), 36–56.
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.
Bouamama, S., Blum, C., Pinacho-Davidson, P., 2022. A population-based iterated greedy algorithm for maximizing sensor network lifetime. Sensors 22 (5), 1804.
Bruglieri, M., Cordone, R., 2021. Metaheuristics for the minimum gap graph partitioning problem. Comput. Oper. Res. 132, 105301.
Calvo, B., Santafé Rodrigo, G., 2016. Scmamp: Statistical comparison of multiple algorithms in multiple problems. R J. 8/1.
Canca, D., De-Los-Santos, A., Laporte, G., Mesa, J.A., 2017. An adaptive neighborhood search metaheuristic for the integrated railway rapid transit network design and line planning problem. Comput. Oper. Res. 78, 1–14.
Cardei, M., Du, D.Z., 2005. Improving wireless sensor network lifetime through power aware organization. Wirel. Netw. 11 (3), 333–340.
Cardei, M., MacCallum, D., Cheng, M.X., Min, M., Jia, X., Li, D., Du, D.Z., 2002. Wireless sensor networks with energy efficient organization. J. Interconnect. Netw. 3 (03n04), 213–229.
Cockayne, E.J., Hedetniemi, S.T., 1975. Optimal domination in graphs. IEEE Trans. Circ. Syst. 22 (11), 855–857.
Cockayne, E.J., Hedetniemi, S.T., 1977. Towards a theory of domination in graphs. Networks 7 (3), 247–261.
Dupin, N., Talbi, E.-G., 2021. Matheuristics to optimize refueling and maintenance planning of nuclear power plants. J. Heuristics 27 (1–2), 63–105.
Feige, U., Halldórsson, M.M., Kortsarz, G., Srinivasan, A., 2002. Approximating the domatic number. SIAM J. Comput. 32 (1), 172–195.
Ferrer, J., Chicano, F., Ortega-Toro, J.A., 2021. CMSA algorithm for solving the prioritized pairwise test data generation problem in software product lines. J. Heuristics 27, 229–249.
Fialho, A., Da Costa, L., Schoenauer, M., Sebag, M., 2010. Analyzing bandit-based adaptive operator selection mechanisms. Ann. Math. Artif. Intell. 60 (1), 25–64.
Garcia, S., Herrera, F., 2008. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J. Mach. Learn. Res. 9 (Dec), 2677–2694.
Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide To the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA.
Hammersley, J.M., Handscomb, D., 1964. Percolation processes. In: Monte Carlo Methods. Springer, pp. 134–141.
Hu, B., Raidl, G.R., 2006. Variable neighborhood descent with self-adaptive neighborhood-ordering. In: Proceedings of the 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics.
Irving, R.W., 1991. On approximating the minimum independent dominating set. Inform. Process. Lett. 37 (4), 197–200.
Islam, K., Akl, S.G., Meijer, H., 2009. Maximizing the lifetime of wireless sensor networks through domatic partition. In: 2009 IEEE 34th Conference on Local Computer Networks. IEEE, pp. 436–442.
Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.-G., 2022. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. European J. Oper. Res. 296 (2), 393–422.
Landete, M., Sainz-Pardo, J.L., 2022. The domatic partition problem in separable graphs. Mathematics 10 (4), 640.
Lewis, R., Thiruvady, D., Morgan, K., 2019. Finding happiness: an analysis of the maximum happy vertices problem. Comput. Oper. Res. 103, 265–276.
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.
Mak, N.H., Seah, W.K., 2009. How long is the lifetime of a wireless sensor network? In: 2009 International Conference on Advanced Information Networking and Applications. pp. 763–770.
Méndez-Díaz, I., Zabala, P., 2008. A cutting plane algorithm for graph coloring. Discrete Appl. Math. 156 (2), 159–179.
Mesbahi, M., Egerstedt, M., 2010. Graph Theoretic Methods in Multiagent Networks. Princeton University Press.
Mischek, F., Musliu, N., 2022. Reinforcement learning for cross-domain hyper-heuristics. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence. IJCAI-22, pp. 4793–4799.
Nagra, A.A., Han, F., Ling, Q.H., 2019. An improved hybrid self-inertia weight adaptive particle swarm optimization algorithm with local search. Eng. Optim. 51 (7), 1115–1132.
Nguyen, T.N., Huynh, D.T., 2007. Extending sensor networks lifetime through energy efficient organization. In: International Conference on Wireless Algorithms, Systems and Applications. WASA 2007, IEEE, pp. 205–212.
Ore, O., 1962. Theory of Graphs.
Pinacho-Davidson, P., Bouamama, S., Blum, C., 2019. Application of CMSA to the minimum capacitated dominating set problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 321–328.
Poon, S.H., Yen, W.C.K., Ung, C.T., 2012. Domatic partition on several classes of graphs. In: International Conference on Combinatorial Optimization and Applications. Springer, pp. 245–256.

Queiroz dos Santos, J.P., de Melo, J.D., Duarte Neto, A.D., Aloise, D., 2014. Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. Expert Syst. Appl. 41 (10), 4939–4949.

Riege, T., Rothe, J., 2005. An exact 2.9416^n algorithm for the three domatic number problem. In: International Symposium on Mathematical Foundations of Computer Science. Springer, pp. 733–744.

Riege, T., Rothe, J., Spakowski, H., Yamamoto, M., 2007. An improved exact algorithm for the domatic number problem. Inform. Process. Lett. 101 (3), 101–106.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Rosati, R.M., Bouamama, S., Blum, C., 2023a. Construct, merge, solve and adapt applied to the maximum disjoint dominating sets problem. In: Metaheuristics. pp. 306–321.

Rosati, R.M., Kletzander, L., Blum, C., Musliu, N., Schaerf, A., 2023b. Construct, merge, solve and adapt applied to a bus driver scheduling problem with complex break constraints. In: AIxIA 2022 – Advances in Artificial Intelligence. pp. 254–267.

Van-Rooij, J.M.M., 2010. Polynomial space algorithms for counting dominating sets and the domatic number. In: International Conference on Algorithms and Complexity. Springer, pp. 73–84.

Wang, R., Purshouse, R.C., Fleming, P.J., 2015. Preference-inspired co-evolutionary algorithms using weight vectors. European J. Oper. Res. 243 (2), 423–441.

Watkins, C.J.C.H., 1989. Learning from Delayed Rewards. King's College, Cambridge United Kingdom.

Watson, J.-P., 2010. An introduction to fitness landscape analysis and cost models for local search. In: Handbook of Metaheuristics. Springer US, Boston, MA, pp. 599–623.

Watts, D.J., Strogatz, S.H., 1998. Collective dynamics of 'small-world'networks. Nature 393 (6684), 440–442.

Welsh, D.J., Powell, M.B., 1967. An upper bound for the chromatic number of a graph and its application to timetabling problems. Comput. J. 10 (1), 85–86.