An Approximate MIP-DoM Calculation for Multi-objective Optimization using Affinity Propagation Clustering Algorithm

COIN Report Number 2021007

Claudio L. d. V. Lopes
Postgraduate Program in Mathematical and Computational Modeling, CEFET-MG
BH, MG, Brazil
claudiolucio@gmail.com

Flávio V. Cruzeiro Martins
Computer Department at the CEFET-MG
BH, MG, Brazil 30510-000
flaviocruzeiro@cefetmg.br

Elizabeth F. Wanner
Computer Department at the CEFET-MG,
BH, MG, Brazil
efwanner@cefetmg.br

Kalyanmoy Deb
Department of Computer Science and Engineering,
Michigan State University
Michigan, East Lansing, USA
kdeb@egr.msu.edu

ABSTRACT

In multi- and many-objective optimization, an effective, accurate, and meaningful comparison of multiple solution sets is an important task. Dominance move (DoM) is a quality indicator that compares two solution sets from a Pareto-optimal sense. DoM measures the minimum move in elements of $\boldsymbol{P}$ needed such that every element of $\boldsymbol{Q}$ is dominated or identical to at least one element of the moved set $\boldsymbol{P'}$. The indicator presents some relevant features as it is Pareto compliant. It does not need a prior reference point or set; it does not demand any normalization, and it is not affected by dominance-resistant solutions. However, the DoM computation involves a combinatorial explosion of moves. A recent paper proposes a mixed-integer programming (MIP) approach for computing DoM that exhibits a computational complexity that is linear to the number of objectives and polynomial to the number of solutions. Even with this property, considering practical situations, the MIP-DoM calculation on some problems may take many hours or even days. This paper presents an approximation method to deal with the combinatorial nature of the DoM calculation using a cluster-based divide-and-conquer strategy. Considering the original MIP-DoM formulation, some classical problems sets are tested, showing that the affinity propagation cluster based-algorithm is computationally much faster and makes a small percentage error from the original DoM value.

KEYWORDS

Multi-objective optimization, Evolutionary multi-objective optimization, Computationally expensive optimization, Machine Learning, Cluster algorithms

1 INTRODUCTION

Problems with more than one objective functions to be optimized are referred as multi-objective optimization problems (MOOP). An MOOP involves a decision variable vector, $\mathbf{x}$, from a feasible decision space $\Omega \subseteq \mathbb{R}^N$, and a set of M objective functions. The minimization of an MOOP can be simply defined as

$$\text{Minimize} \quad F(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})]^T, \quad \mathbf{x} \in \Omega. \tag{1}$$

$F : \Omega \to \Theta \subseteq \mathbb{R}^M$ is a mapping from the feasible decision space Ω to vectors in the M-dimensional objective space Θ. In this paper, we are interested in the evaluation of these objective vectors (loosely referred here as solution sets), and a comparison among them. Typically, these objective vectors are in conflict with each other. Many real-world problems are formulated in this manner, and the goal is to obtain a set of trade-off solutions known as Pareto-optimal solutions.

The comparison between two or more solution sets has great importance in the EMO community [8]. Quality indicators play a relevant role in this sense, and it can be used to compare the outcomes of multi-objective algorithms or even assist an algorithm during the search for non-dominated candidates. Hypervolume (HV) [3, 13], inverted generational distance plus (IGD+) [6], and ϵ-indicator [16] are some quality indicators, just to name a few.

Dominance move (DoM) is a binary quality indicator used in multi-objective and many-objective optimization to compare two solution sets obtained from different algorithms [7]. The DoM measures the minimum 'effort' that one solution set has to make in trying to dominate the second set, more specifically, the absolute sum of the movements (i.e., Manhattan distance) in objective directions needed to make the first set dominant. Compared to ϵ-indicator, it has the advantage of considering all solutions of both sets. Moreover, it does not require any reference point or a reference set (such as in IGD+ and HV), it does not demand any normalization, and it is also not affected by dominance-resistant solutions.

However, the DoM formulation brings a combinatorial issue in its calculation. The original proposers [7] presented an exact calculation only for the bi-objective case. Another work [4] tried to master the problem by treating it as an assignment problem. Some experiments showed that the DoM assignment formulation is correct and in accordance with its predecessor. However, the solution sets' cardinality still impose a handicap, preventing its use in solution sets with more than 20 solution candidates.

In a recent paper, a mixed integer programming (MIP) approach was proposed to calculate DoM [5]. In the bi-objective space, experiments presented the model's correctness. Other experiments using up to 400 solutions and three to 30 objectives showed how the model behaves in higher-dimensional cases. The paper also stated that the MIP-DoM indicator had a positive Pearson correlation with HV. Moreover, its time calculation was found to be polynomial to the cardinality of sets and linear to the number of objectives (M).

The MIP-DoM time spent is highly affected by the number of solutions in the set. In [5], the experiments present a relevant variability when the number of solutions is increased. Some models take some minutes, but others took 78 hours, for example. It is observed that this fact is inherent to the set distribution, the model, and some inner characteristics involving the two solution sets, $\boldsymbol{P}$ and $\boldsymbol{Q}$. Without loss of generality, let us assume that the number of elements in $\boldsymbol{P}$ and $\boldsymbol{Q}$ are equal and given by L. The time complexity function is defined by $T = \alpha L^{\beta}$, in which α and β are two coefficients obtained using a linear regression in a log-log plot with T and L [5]. The primary motivation is to explore some methodology that can reduce T. The initial hypothesis is related to a 'divide and conquer' approach to deal with the combinatorial nature of DoM. Using a cluster strategy, it is possible to establish the Proposition 1.1:

Proposition 1.1. *Let $\boldsymbol{P}$ and $\boldsymbol{Q}$ be two solution sets with an equal number of elements defined by L, a DoM($\boldsymbol{P}, \boldsymbol{Q}$) calculation with a complexity function defined by $T = \alpha L^{\beta}$, in which α and β are two positive coefficients. Assuming C as the number of non-overlapping clusters formed by elements from $\boldsymbol{P}$ and $\boldsymbol{Q}$. Then, the new complexity time function can be defined as $T_C = C\left[\alpha\left(\frac{L}{C}\right)^{\beta}\right]$. If $\beta \geq 1$, then $T_C \leq T$.*

Proof. By contraposition, suppose that $T_C > T$. In this way,

$$\begin{aligned} T_C > T \Rightarrow & \alpha\left(\frac{L}{C}\right)^{\beta} C > \alpha L^{\beta} \\ & \left(\frac{L^{\beta}}{C^{\beta}}\right) C > L^{\beta} \\ & \frac{1}{C^{\beta-1}} > 1 \\ & C^{1-\beta} > 1. \end{aligned}$$

Then, we have that $1 - \beta > 0 \Rightarrow \beta < 1$. ∎

The Proposition 1.1 represents the rationale behind the improvement in the MIP-DoM time spent calculation. The MIP-DoM calculation for some problem sets presents $\beta >> 1$ [5]. Some research questions (RQs) and their implementation details regarding the Proposition 1.1 must be addressed:

(1) **RQ1:** How to cluster the solutions sets $\boldsymbol{P}$ and $\boldsymbol{Q}$?
(2) **RQ2:** How to associate clusters of two $\boldsymbol{P}$ and $\boldsymbol{Q}$?
(3) **RQ3:** How to compute an approximate DoM value from individual cluster-wise DOM values?

These questions are addressed in this study. The first work's contribution is related to clustering the solution sets. We have tested some different algorithms such as affinity propagation [12], mean-shift [14], and K-means [10]. However, only the affinity propagation algorithm presents some relevant context properties: numerical stability and deterministic behavior. The second contribution is related to assigning each cluster from $\boldsymbol{P}$ to each cluster from $\boldsymbol{Q}$, we solve the problem as an assignment problem. The final contribution shows a two-phase approach in which MIP-DoM approximation maintains the original MIP-DoM values and properties. It can generate values with an error estimate of less than 0.40% on average, and a time spent improvement until 5400 times, based on the experiments done.

This paper is organized as follows: Section **2** presents some important concepts to our final approach: the MIP-DoM quality indicator and its properties, some cluster algorithms justifying our final choice related to affinity propagation algorithm, the assignment problem formulation, and the two-phased proposal to the problem. In Section **3** our algorithm is presented with a further discussion and some relevant details that deal with the approximate MIP-DoM calculation. Section **4** brings some multi-objective experiments to validate the approximate MIP-DoM using affinity propagation with the optimal MIP-DoM value and compare the time spent by the models. The final considerations and comments are presented in Section **5**.

2 BACKGROUND

2.1 MIP-DoM quality indicator

Dominance move is a measure for comparing two solution sets, being classified as a binary indicator. It considers the minimum overall movement of points in one set needed to *dominate* each and every member of the other set. The DoM does not need *a priori* problem knowledge, parameters, or a reference set. DoM can be defined as follows.

Definition 2.1. DoM [7]: Consider that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are sets of points, with $\boldsymbol{p}_i$ points $i \in \{1, \ldots, |\boldsymbol{P}|\}$ and $\boldsymbol{q}_j$ points $j \in \{1, \ldots, |\boldsymbol{Q}|\}$. The dominance move of $\boldsymbol{P}$ to $\boldsymbol{Q}$, $DoM(\boldsymbol{P}, \boldsymbol{Q})$, is the minimum total distance of moving points of $\boldsymbol{P}$, such that each and every point in $\boldsymbol{Q}$ is dominated by at least one point in $\boldsymbol{P}$. In fact, the problem aims to obtain $\boldsymbol{P'} = \{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_{|\boldsymbol{P}|}\}$ from $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{|\boldsymbol{P}|}\}$ such that $\boldsymbol{P'}$ dominates $\boldsymbol{Q}$ and the total move from $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{|\boldsymbol{P}|}\}$ to $\{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_{|\boldsymbol{P}|}\}$ must be minimum.

The formal expression of DoM can be stated as [5]:

$$DoM(\boldsymbol{P}, \boldsymbol{Q}) = \min_{P' \prec Q} \sum_{i=1}^{|\boldsymbol{P}|} d(\boldsymbol{p}_i, \boldsymbol{p}'_i), \tag{2}$$

in which $d(\boldsymbol{p}_i, \boldsymbol{p}'_i)$ can be the Euclidean or the Manhattan distance between $\boldsymbol{p}_i$ to $\boldsymbol{p}'_i$.

DoM reflects how far one solution set $\boldsymbol{P}$ needs to move to dominate another solution set $\boldsymbol{Q}$. Thus, $DoM(\boldsymbol{P}, \boldsymbol{Q})$ is always larger than or equal to zero. A small value suggests that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are close, and a large value indicates that $\boldsymbol{P}$ performs worst than $\boldsymbol{Q}$.

However, the computational cost to calculate DoM presents a combinatorial nature [4, 8] and its original formulation presents an algorithm for the bi-objective case only. In a recent paper [5], a mixed-integer programming (MIP) is used to calculate DoM for any number of objectives and the resulting model is dubbed MIP-DoM.

MIP-DoM applies the DoM definition and uses as input the $\boldsymbol{P}$ and $\boldsymbol{Q}$ sets of points, with $\boldsymbol{p}_i$ points $i \in \{1, \ldots, |\boldsymbol{P}|\}$ and $\boldsymbol{q}_j$ points

$j \in \{1, \ldots, |\boldsymbol{Q}|\}$, respectively. Each $\boldsymbol{p}_i$ is composed of $p_{(i,m)}$ components, $1 \leq m \leq M$ in which M is the number of objective functions. Analogously, each $\boldsymbol{q}_j$ is composed of $q_{(j,m)}$ components.

Given $\boldsymbol{P}$ and $\boldsymbol{Q}$, $\boldsymbol{P'}$ is a set of points, in which each $\boldsymbol{p}'_i$, generated from $\boldsymbol{p}_i$, can have updates in one or more objectives. The model output is composed by the $\boldsymbol{P'}$ and the final DoM value as expressed in Equation (2).

In [5], some experiments using up to 400 solutions and three to 30-objectives are performed to show how the model behaves in higher-dimensional cases. The DTLZ and WFG families have been used in the experiments. As an example, the complexity time function obtained for DTLZ1 is $T \approx O(L^{3.311})$, $O(L^{2.888})$ and $O(L^{2.833})$, for $M = 5$, 10 and 15, respectively [5]. Considering DTLZ2 there is a $T \approx O(L^{4.408})$, $O(L^{4.311})$ and $O(L^{3.632})$. These two examples confirm the motivation behind the Proposition 1.1 and emphasize the importance of an approximated approach for calculating MIP-DoM.

2.2 RQ1: Clustering Problem

We discuss the first research question on how to cluster the solutions sets $\boldsymbol{P}$ and $\boldsymbol{Q}$. Clustering aims to divide the data points into subsets in such a way that the elements belonging to the same subset are similar to each other. Alternatively, points belonging to different subsets are as dissimilar as possible. There are some different types of clustering paradigms such as hierarchical, density-based, graph-based, and others [15].

Taking our context into consideration, we would like to have some clustering method properties:

- *Parameters*: typically, clustering methods involve some parameters, as inputs. The difficulty is how to pick settings for those parameters. In this sense, we would like algorithms that have as fewer parameters as possible;
- *Numerical stability*: some cluster algorithms have a stochastic component (e.g., a random initialization or a sampling approach). Our preference is related to stable methods. If we run the same algorithm using the same parameters, we need to get the same clusters always. Still, if we vary some parameters, we want a change in a somewhat stable predictable manner. It is a fundamental requisite for our approximation method;
- *Performance*: a cluster algorithm with a computational complexity function better than our Proposition 1.1 is a suitable choice to the problem. Moreover, we should prefer a clustering algorithm that can scale up to large data-sets.

Based on those criteria, we review some clustering methods. K-Means is a representative-based algorithm that minimizes the squared distance of points from their respective cluster means (centroids) [10]. It performs clustering interactively, assigning each point to only one cluster and minimizing the sum of squared errors. One of the parameters of K-Means is the number of clusters. Another K-Means feature is its stochastic component in the initialization step (it does not have numerical stability). However, considering the performance, the time complexity is $O(C \cdot L \cdot M \cdot i)$, in which C is the cluster number, L is the number of data points, M the space dimension, and i is the number of iterations.

Mean shift is another cluster algorithm [2, 14]. The method assumes that exists some probability density function from which the data is obtained, and it tries to place centroids of clusters at the maxima of the density function. It does not require prior knowledge of the number of clusters. The number of clusters is obtained automatically by finding the centers of the densest regions in the space, the modes. It is an iterative technique and it estimates the modes of the multivariate distribution underlying the feature space. However, as it approximates the centroids via kernel density estimation techniques, the key parameter is then the bandwidth of the kernel used. Concerning the numerical stability we have a random initialization, as well. But, the performance may vary: if it uses a flat kernel and a ball tree to look for members of each kernel, the complexity is $O(L \cdot s \cdot log(s))$ in lower dimensions, and, $O(L \cdot s^2)$ in higher dimensions, with s the number of samples.

Algorithm 1 Affinity propagation

```
1:  function AFFINITY PROPAGATION($S$,$\lambda$,$preference$,$T$)
2:      Input: $S$,$\lambda$,$preference$,$T$
3:      Output: $\{c_1, \ldots, c_C\}$
4:      $\boldsymbol{R}$,$\boldsymbol{A} \leftarrow 0, 0$
5:      $t \leftarrow 0$
6:      $\forall k : s[k,k] \leftarrow preference$
7:      while $t < T$ do
8:                                              ▷ propagating responsibility
9:          $\forall i,j : \rho[i,j] \leftarrow \begin{cases} (i \neq j), & s[i,j] - \max_{\forall k: k \neq j} (a[i,k] + s[i,k]) \\ (i = j), & s[i,j] - \max_{\forall k: k \neq j} (s[i,k]) \end{cases}$
                                                ▷ propagating availability
10:         $\forall i,j : \alpha[i,j] \leftarrow \begin{cases} (i \neq j), & \min(0, r[j,j] + \sum_{\forall k: k \neq i,j} \max(0, r[k,j])) \\ (i = j), & \sum_{\forall k: k \neq i} \max(0, r[k,j]) \end{cases}$
11:                                             ▷ updating responsibility
12:         $\forall i,j : r[i,j] \leftarrow (1-\lambda)\rho[i,j] - \lambda r[i,j]$
13:                                             ▷ updating availability
14:         $\forall i,j : a[i,j] \leftarrow (1-\lambda)\alpha[i,j] - \lambda a[i,j]$
15:     $\forall i : c[i] \leftarrow \operatorname{argmax}_k (r[i,k] + a[i,k])$
16:     return $\{c_1, \ldots, c_C\}$
```

The two previously discussed methods do not present the numerical stability feature suitable for the MIP-DoM approximation. For our calculation approach, this is a fundamental requisite. As an alternative, affinity propagation clustering is a deterministic algorithm (regarding its initial parameters) and is described next.

The clustering algorithm treats all points as potential cluster center. The process starts with the similarity matrix ($\boldsymbol{S}$), which is constructed in a pairwise manner using a similarity function, such as the negative Euclidean distance (values near zero indicate similarity, otherwise the degree of dissimilarity between the points) [12].

Given $\boldsymbol{S}$, the method tries to find 'exemplars' that maximize the net similarity. The algorithm is a graph approach and can be viewed as a message-passing process through edges with two kinds of messages exchanged among data points. The responsibility matrix $\boldsymbol{R}$ with $r[i,j]$ elements in with $i, j \in \{1, \ldots, L\}$, means how well suited, in a accumulated way, is the point j to be an exemplar to i. The availability matrix $\boldsymbol{A}$ with $a[i,j]$ elements in with $i, j \in \{1, \ldots, L\}$, is a message from data point j to i reflecting how appropriate it would be for data point i to choose data point j as its exemplar.

The whole process is detailed in Algorithm 1. All responsibilities and availabilities are initialized at 0. The λ parameter is a damping factor used to avoid numerical oscillations. High value leads to slow run time but avoids numeric oscillations. There is an interval recommendation to $\lambda \subseteq [0.5, 1.0]$.

Another important parameter is the preference, which is used to fill up the matrix diagonal and indicates how appropriate the i-th point is to be selected as an 'exemplar'. High preference values will cause the method to find many clusters. Contrarily, low values will generate a small number of clusters. Preference can be set as the median similarities' value.

The propagating responsibility between point i and j is indicated by $\rho[i, j]$. The expression in line 7 of Algorithm 1 indicates the update using the similarity value between points i and j minus the best value for a point i. The diagonal matrix values are updated, as well, using the preference minus the second-best similarity value. Similarly, $\alpha[i, j]$ is the propagating availability. The diagonal matrix is updated, reflecting the accumulated evidence that the point j is suitable to be an 'exemplar', based on the positive responsibilities of k other elements. The messages are updated and kept until convergence is met or the iteration reaches a certain number. The original algorithm has a time complexity $O(L^2 \cdot T)$ to update massages. However, there are some variants that can improve the time complexity [12].

2.3 RQ2: Assignment Problem

The following paper question is: (2) Considering each cluster from $\boldsymbol{P}$ and $\boldsymbol{Q}$ how to assign the subsets to calculate the MIP-DoM for each subset assignment? The classical assignment problem is discussed in many works in mathematical programming and management sciences. In general, the problem deal with the question of how to assign n tasks to n agents. The objective function is related to minimize the total cost of the assignments. Some examples still involve other situations, such as assigning jobs to machines, tasks to workers, product to machines, students to teachers, clients to sellers, and many others.

The assignment problem has many variations, such as altering some constraints, changing the possible assignments between the sets, or even adding some new data in the objective function, [1].

In Figure 1 we can see a bipartite graph with two sets $\boldsymbol{C^P}$ and $\boldsymbol{C^Q}$ which are the clusters centroids/'exemplars' created from $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively. Our problem leads to an unbalanced version of the assignment problem. Here, our objective is to assign to all members of $\boldsymbol{C^Q}$ one member from $\boldsymbol{C^P}$ with the minimum cost, represented by the edges. The unbalanced version indicates that a member of $\boldsymbol{C^P}$ could not be used in the final solution. Figure 1 presents an example, and a valid assignment from $\boldsymbol{C^P}$ to $\boldsymbol{C^Q}$ representing our case.

The mathematical model for the assignment problem presented in Figure 1 can be given by the Equation 3. Concerning the cluster number, it is given by $|C^P|$ and $|C^Q|$. The $x_{ij} = 1$ indicates if the i-th member of $\boldsymbol{C^P}$ is assigned to the j-th member of $\boldsymbol{C^Q}$, and 0 otherwise. The c_{ij} represents the distance to j-th assigned to the i-th. The first constraint ensures that every member in $\boldsymbol{C^Q}$ is assigned to only one member of $\boldsymbol{C^P}$.

$$\begin{aligned} &\text{Minimize} \quad \sum_{i=1}^{|C^P|} \sum_{j=1}^{|C^Q|} c_{ij} x_{ij} \\ &\text{subject to:} \begin{cases} \sum_{i=1}^{|C^P|} x_{ij} = 1 & \forall j \in \{1, 2, \ldots, |C^Q|\}, \\ x_{ij} = 0 \text{ or } 1. \end{cases} \end{aligned} \tag{3}$$

It is worth emphasizing that x_{ij} is a binary variable.

The time complexity in the assignment problems can vary due to problem specificity. The linear model, such as detailed in Equation 3, has the $O(|C^P| \cdot |C^Q| \cdot r + r^2 \log(r))$ in which $r = \min(|C^P|, |C^Q|)$ [11] .

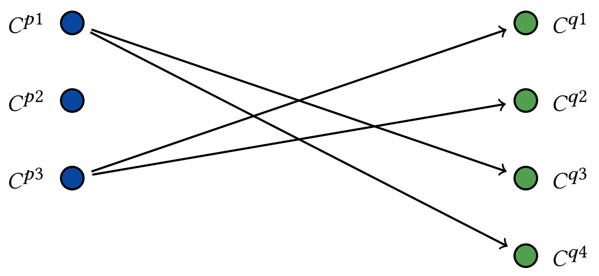

Figure 1: One possible example of assignment between C^P and C^Q . The edges indicating the assignment can be computed using the minimum distance from members C^P to C^Q.

2.4 RQ3: A Two-phase Approximate DoM Approach

The last question is: (3) How to use the MIP-DoM values in a 'divide and conquer' approach and guarantee that it preserves the original MIP-DoM value? Based on the combinatorial nature of DoM [5] a method that uses clustering followed by an assignment step to calculate DoM values will probably, generate a solution with a little chance of being a good approximation. The hypothesis is to improve the intermediary values from the 'divide' phase turning the final value even better in a 'conquer' phase. We present more details in the next section. In this sense, it makes sense to observe that the approximation should always be greater than the original MIP-DoM value.

3 AFFINITY PROPAGATION WITH A TWO-PHASE MIP-DOM CALCULATION

The MIP-DoM model, affinity propagation, and the assignment formulation are the building blocks of our approximation method. Hereafter, all the referenced methods will be put together to explain the two-phase approach to the problem.

3.1 Geometric Intuition

A simple example can illustrate our approximation proposal. Consider two solution sets $\boldsymbol{P}$ and $\boldsymbol{Q}$ in the bi-objective space, and with four solutions in each set, see Figure 2 marked 'plot 1'. Our intention is to calculate MIP-DoM($\boldsymbol{P}$, $\boldsymbol{Q}$) in a two-phase manner, which mimics a 'divide and conquer' approach.

In Phase 1, the first step is to apply the clustering algorithm in both sets. For that task, affinity propagation cluster is used, separately, in $\boldsymbol{P}$ and $\boldsymbol{Q}$. In Figure 2, plot 1, it is possible to observe that we have three clusters generated by the method. Let us call $|\boldsymbol{C^P}|$ and $|\boldsymbol{C^Q}|$ the number of clusters generated, in plot 2, $|\boldsymbol{C^P}| = |\boldsymbol{C^Q}|$, but it is not a requirement of our proposal. Lets call $\{\boldsymbol{P}^1, \cdots, \boldsymbol{P}^{|\boldsymbol{C^P}|}\}$ as a set formed with all the subsets created in the cluster procedure. The assertion in which $\boldsymbol{P}^1 \bigcup \boldsymbol{P}^2 \bigcup \cdots \bigcup \boldsymbol{P}^{|\boldsymbol{C^P}|} = \boldsymbol{P}$ is valid. The similar assertion is true to $\boldsymbol{Q}$.

After the cluster method we have $L_{P^1} + \cdots + L_{P^{|C^P|}} = L_P$ and $L_{Q^1} + \cdots + L_{Q^{|C^Q|}} = L_Q$. In Figure 2, phase 1, plot 1, we can observe three clusters from $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively, $\{\boldsymbol{P}^1, \boldsymbol{P}^2, \boldsymbol{P}^3\}$ and $\{\boldsymbol{Q}^1, \boldsymbol{Q}^2, \boldsymbol{Q}^3\}$. Each solution set member, now is a member belonging to a cluster. Differently of K-Means, affinity propagation elects a solution as an 'exemplar' to represent the cluster. However, we decide to use the ideal point to represent each cluster: constructed by each objective's best value for all solutions owned by the cluster. The ideal points are depicted in Figure 2, phase 1, plot 2.

The next step is how to assign the clusters 'exemplars' using $\boldsymbol{C^P}$ and $\boldsymbol{C^Q}$ sets. Here, the assignment problem does not use the generated 'exemplars'; instead, the ideal point coming from each cluster is used. In plot 2 these points are emphasized. With such points, the assignment is done using the Equation 3. The c_{ij} is calculated using the Manhattan distance and represent the distance between the i-th cluster ideal points from $\boldsymbol{P}$ to j-th cluster ideal points from $\boldsymbol{Q}$.

The original MIP-DoM [5] is calculated for each assigned pair of clusters, generating intermediary MIP-DoM values. The method returns the MIP-DoM value and $\boldsymbol{P'}$ for each assigned cluster pair. In plot 3, Figure 2, the values generated by each MIP-DoM execution are shown. In a 'divide and conquer' approach, the first estimate appears in phase 1. Considering the problem, MIP-DoM is calculated by summing the parts obtained from assignments. The first value is 4.4.

In plot 4, phase 2 is started, and now the $\boldsymbol{P}$ solution set is substituted by the $\boldsymbol{P'}$ coming from all MIP-DoM cluster executions in phase 1. It is relevant to note that the MIP-DoM offers not only a quality indicator value. The $\boldsymbol{P'}$ is a new solution set that can or can not dominate $\boldsymbol{Q}$ completely (it can not dominate totally because the cluster approximation makes MIP-DoM 'blind' for the whole solution set). However, if we re-execute MIP-DoM using the new solution set $\boldsymbol{P'}$ and the distance coming from phase 1, it is possible to improve the final solution better. An approximate MIP-DoM receives an additional parameter, the intermediary value, indicated by $\boldsymbol{Distance}$ in plot 4.

Finally, the approximate MIP-DoM is calculated. It is shown in Figure 2, phase 2, plot 5, with the final solution (using the $\boldsymbol{Distance}$ MIP-DoM value and the new $\boldsymbol{P'}$). Considering the MIP-DoM model presented in [5], a new component is added to the objective function. It works as a penalization strategy using the MIP-DoM execution values as a parameter in this approximated MIP-DoM run (associated with each $\boldsymbol{P'}$ subset comes from the first execution, we have a distance that represents a penalization for the movement already done). The results from the approximated MIP-DoM are the same as before. It generates the new, improved value (now with 3.4) and the new $\boldsymbol{P'}$ which now dominates $\boldsymbol{Q}$ completely.

3.2 Proposed algorithm

With the algorithm's intuition explained, it is possible to discuss some other details. Additional aspects are presented in Algorithm 2. The $\boldsymbol{P}$ and $\boldsymbol{Q}$ solution sets are the first two input parameters, and the *preference*, λ, and T are the last ones, and related to affinity propagation clustering.

The first step is related to the similarity matrix calculation: *SIMILARITYMATRIX*. We use the negative Euclidean distance, as suggested in the affinity propagation algorithm. The total number of pairwise comparisons to calculate the pairwise distance matrix is $\lceil L(L-1)/2 \rceil$, and each comparison can be a vector operation with M summations.

Secondly, we create the clusters using the affinity propagation, as detailed in Algorithm 1: *AFFINITYPROPAGATION*. We use the similarity matrix obtained from the last step. As we want the numerical stability, in a try and error approach we decide the damping factor λ to 0.9. The *preference* is another clustering parameter received in the approximation MIP-DoM function, which will be better discussed afterward. The final affinity propagation parameter is the maximum number of iterations. It is used as a default value of $T = 200$. As discussed in Section 2 the affinity propagation method has the worst time complexity as $O(L^2 \cdot T)$.

The next step is how to assign the clusters from $\boldsymbol{P}$ to every cluster from $\boldsymbol{Q}$. The *ASSIGNMENT* method has two parameters related to clusters from $\boldsymbol{P}$ and $\boldsymbol{Q}$ which are formed by the elements in each cluster, including the 'exemplars'. The *ASSIGNMENT* method calculates the ideal points in each cluster. The number of clusters is $|\boldsymbol{C^P}|$ and $|\boldsymbol{C^Q}|$ meaning the number of clusters suggested by the affinity propagation clustering method. Assuming $|\boldsymbol{C^P}| = |\boldsymbol{C^Q}|$, the time complexity for this assignment in the worst case is $O(|\boldsymbol{C^P}|^2 \cdot \log(|\boldsymbol{C^P}|))$. It is important to note that $|\boldsymbol{C^P}| << L$, which emphasize the use of the assignment formulation.

From the assignment step, we have tuples which indicate the clusters from $\boldsymbol{P}$ assigned to every cluster from $\boldsymbol{Q}$. For each tuple in the assignment, it is possible to execute the model to calculate the MIP-DoM value: *MIPDoM* function in line 17 Algorithm 2, as in [5]. D_c is the intermediary MIP-DoM value, and $\boldsymbol{P'_c}$ are the solutions sets generated by MIP-DoM. These values and solution sets are retained for the next and final step.

The approximate MIP-DoM step involves the use of $\boldsymbol{D}$ values, and $\boldsymbol{P'}$ solution set:*APPROXIMATEMIPDoM* in line 22 . The cardinality of $\boldsymbol{P'}$ is much smaller than $\boldsymbol{P}$ [5], in other words, $L_{P'} \ll L_P$. The *APPROXIMATEMIPDoM* differ from *MIPDoM* function in line 17, just by the $\boldsymbol{D}$ vector added in the objective function, all the other details remained the same. It is relevant to note that if at least one solution in $\boldsymbol{P'}$ already dominate all $\boldsymbol{Q}$, the approximate MIP-DoM value will be the same as the MIP-DoM value previously executed, using the $\boldsymbol{D}$ values to dominate $\boldsymbol{Q}$. Here, the time complexity is decreased due to the $\boldsymbol{P'}$ cardinality.

Concerning all complexity time component functions in the Algorithm 2, we expect that our approximate approach has in the worst case a $O(L^2)$ complexity time.

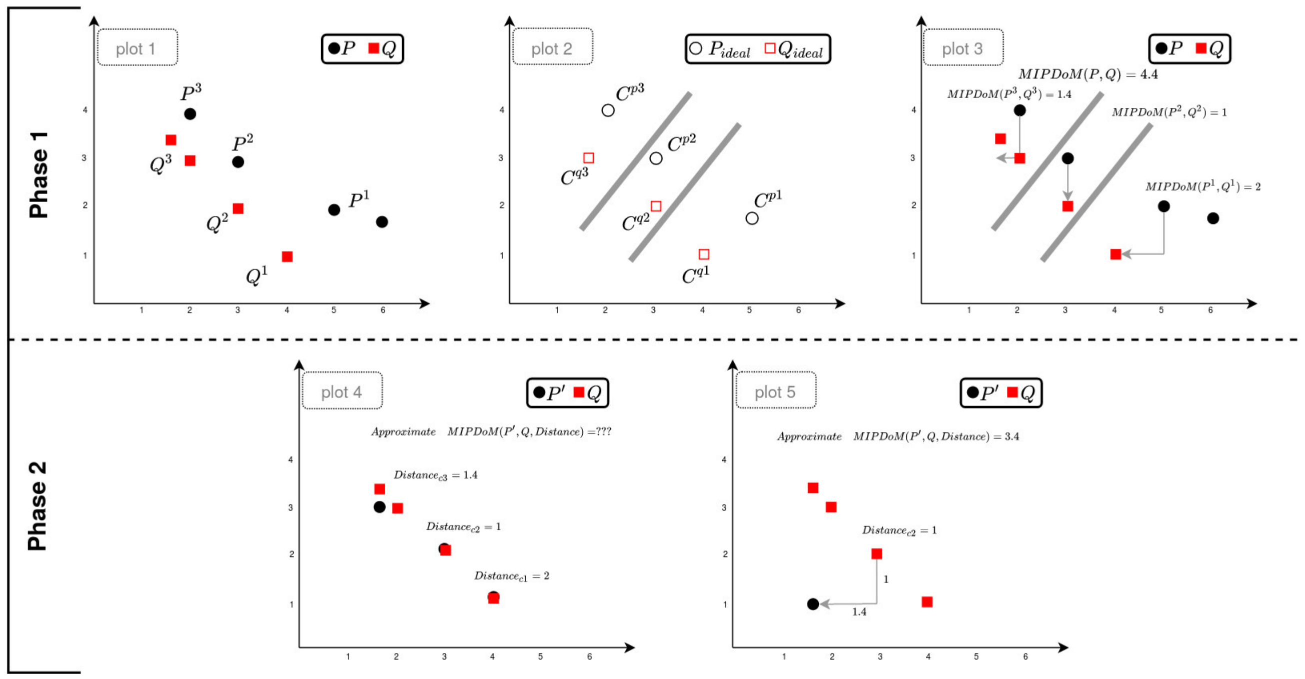

Figure 2: A bi-objective example of how the two-phase approximate MIP-DoM using a clustering algorithm works. There are two phases and five plots in order. Plot 1 defines the two solution sets P and Q and the clusters created for each one. In plot 2, using the clusters, the ideal points are generated, and the assignment is done using such ideal points: build by each objective's best value for all solutions owned by each cluster. Plot 3 shows the original MIP-DoM been calculated for each cluster. It generates a MIP-DoM value for each cluster. The generated MIP-DoM value is retained, and it is associated with each P' generated. Plot 4, phase two is started, and the approximate MIP-DoM will be calculated, but a new parameter is related to the intermediary MIP-DoM value, which comes from the first phase execution (plot 3). Finally, plot 5 presents the approximate MIP-DoM and the final solution calculated using the distance values.

Algorithm 2 Approximate MIP-DoM with affinity propagation

```
1: function MIP-DoM AFFINITY PROPAGATION($P$,$Q$,$preference$,$\lambda$,T)
2:     Input: $P$,$Q$,$preference$,$\lambda$,T
3:     Output: $ApproximateValue$
4:                                                  ▷ PHASE 1
5:                             ▷ calculating the similarity matrix
6:     $S_P \leftarrow SIMILARITYMATRIX(P)$
7:     $S_Q \leftarrow SIMILARITYMATRIX(Q)$
8:                         ▷ generating the clusters from $P$ and $Q$
9:     $P\ clusters \leftarrow AFFINITYPROPAGATION(S_P, \lambda, preference, T)$
10:    $Q\ clusters \leftarrow AFFINITYPROPAGATION(S_Q, \lambda, preference, T)$
11:                         ▷ Assigning clusters using the ideal points
12:    $Assignment \leftarrow ASSIGNMENT(P\ clusters, Q\ clusters)$
13:                 ▷ calculating the MIP-DoM for each assigned cluster
14:    $P' \leftarrow \emptyset$
15:    $D \leftarrow \emptyset$
16:    for each $\{(p_c, q_c)\} \in Assignment$ do
17:        $D_c, P'_c \leftarrow MIPDoM(p_c, q_c)$
18:        $P' \leftarrow P' \bigcup P'_c$
19:        $D \leftarrow D \bigcup D_c$
20:                                                  ▷ PHASE 2
21:                 ▷ final calculation using the approximate MIP-DoM
22:    $ApproximateValue \leftarrow APPROXIMATEMIPDoM(P', Q, D)$
23:    return $ApproximateValue$
```

4 EXPERIMENTS

Somehow to validate our propositions, some multi-objective benchmark problems are selected. We use the same test sets as in [5]. These problem test sets have three objectives and come from *DTLZ* and *WFG* families. We use DTLZ1, DTLZ2, DTLZ3, and DTLZ7, which are linear, concave/multimodality, concave, and disconnected, respectively. In the same manner, WFG1, WFG2, WFG3, and WFG9 are also chosen, presenting similar properties: convex/mixed, convex/disconnected, linear/degenerate, and concave, respectively.

Some algorithms are used to generate the approximated Pareto front and using the outcomes in the MIP-DoM indicator. We use IBEA, NSGA-II, NSGA-III, MOEA/D, and SPEA2 to generate our solution sets. In the experiments, the maximum number of fitness evaluations is set to 10,000, and each algorithm is executed 21 times.

We intend to compare two main factors: the time spent by the original MIP-DoM vs. our approximate approach and the error added at the value calculated by our approximation.

A critical parameter in the affinity propagation MIP-DoM approximation is the *preference* which highly influences the number of clusters. This parameter is input in Algorithm 2. In the initial tests, we observe that the preference value significantly influences the final result; in general, the median value is used. We decide to use a brute force strategy to assess the best testing of some percentile which can represent values near minimum, median, and the maximum value of *preference* is established: 0.01, 0.05, representing the minimum, 0.5 as the median, 0.95 and 0.99 percentile as the maximum (this value is calculated using the similarity matrix).

In Table 1, the experiment results are presented. This table is ordered by the best algorithm results obtained by each algorithm in each problem set. The columns labeled as 'Original' means the value obtained from the MIP-DoM model as in [5], and the 'Approx.' represents the result obtained from our best MIP-DoM approximation using affinity propagation. In the same manner, there is the time spent by each execution. The time spent in the 'Approx.' column is the summation of all preference percentiles.

The difference between the Original vs. Approximate value in some problem sets did not exist. This is the case for DTLZ1, DTLZ2, DTLZ3, WFG1, WFG2, and WFG3. The approximation value is always greater than the original values. The first difference appears to DTLZ7 for the IBEA algorithm, the difference is 2.72%, and for NSGA-II the percentual difference is 0.51%. The next one is related to WFG9 for IBEA, and SPEA2, with the values of 3.00%, and 9.82%. Regarding all the problem sets, the experiment average difference is 0.40%.

Regarding the time spent, there is just one problem set in which the 'Approx.' is higher than the Original time spent: DTLZ1 for NSGA-III and MOEA/D. The first observation is that DTLZ1, in general, does not represent a critical demand/improvement in the time spent considering the original MIP-DoM. However, the approximation approach maintained a similar time spent behavior compared with the original MIP-DoM. The differences are tiny, and the values are next to each other.

For all other problem sets, there are improvements in the time spent. The most time-consuming problem sets are DTLZ2, DTLZ7, and WFG3. Regarding this problem sets, there is an improvement range from ~ 9 to ~ 5400 times better, which happens to DTLZ7 for NSGA-III and DTLZ2 for IBEA, respectively.

As an approximation approach, the method is influenced by parameters. In the approximate MIP-DoM, this parameter comes from the affinity propagation algorithm: the preference. It is relevant to assess the behavior of this parameter in the experiments.

A brute force strategy is used with some preference percentiles search space. Table 2 tries to uncover the method behavior due to the preference parameter. The two time-consuming problem sets family of each family is exposed in more detail: DTLZ2 and WFG3.

Table 2 has three main columns exposing the MIP-DoM values, the number of clusters, and the time spent. All these columns are detailed by the respective percentile preference. The first observation related to preference is the non-monotonic behavior; neither MIP-DoM values, the number of clusters, or the time spent presents it. For example, in WFG3 for NSGA-III, the percentile 0.01 and 0.05 shows the same MIP-DoM value 3.034, it decreases with percentile 0.50, but it is increased again with percentile 0.99.

Another interesting observation is that sometimes different percentile generated the same number of clusters and produces the same MIP-DoM value. In fact, in these cases, the same clusters were generated, resulting in the same final solution. This kind of 'memory' is one of the improvements in the iterative call of the Algorithm 2. In other words, the *AFFINITYPROPAGATION* is called before, and if the method generates the same clusters in which the results are known, there is no necessity to call the Algorithm 2 again. This is a relevant fact; for example, let us consider the WFG3 for the IBEA algorithm, the total spent time presented in Table 1 do not

Table 1: Original MIP-DoM(P,Q) and Approximate MIP-DoM(P,Q) values for the *DTLZ* and *WFG* families for comparison among some algorithms. Observe that P is the solution set generated by the algorithm, and Q is the non-dominated solution set from all algorithms results combined. The two column's groups detail the MIP-DoM values (Original vs. Approximate) and time spent in seconds.

Problem	Algorithms	MIP-DoM		Time spent	
		Original	*Approx.*	*Original*	*Approx.*
DTLZ1	*IBEA*	0.312	0.312	32.59	**22.24**
	NSGA-III	0.551	0.551	**7.21**	10.87
	MOEA/D	1.630	1.630	**22.25**	27.08
	NSGA-II	6.422	6.422	4.49	**4.45**
	SPEA2	11.139	11.139	5.04	**2.47**
DTLZ2	*MOEA/D*	0.970	0.970	101138.66	**65.77**
	IBEA	1.000	1.000	976980.85	**286.50**
	NSGA-III	1.004	1.004	190875.98	**288.71**
	NSGA-II	1.013	1.013	39250.80	**253.25**
	SPEA2	1.022	1.022	107129.99	**72.73**
DTLZ3	*IBEA*	0.049	0.049	16.72	**5.00**
	NSGA-III	18.652	18.652	14.15	**7.86**
	MOEA/D	25.374	25.374	122.72	**3.94**
	NSGA-II	51.208	51.208	26.78	**5.55**
	SPEA2	150.870	150.870	12.63	**6.56**
DTLZ7	*NSGA-III*	1.402	1.402	3345.07	**364.33**
	IBEA	1.468	1.508	65875.00	**135.76**
	MOEA/D	1.695	1.695	2520.99	**23.90**
	NSGA-II	1.782	1.791	8260.20	**363.03**
	SPEA2	2.184	2.184	2040.71	**21.74**
WFG1	*IBEA*	1.230	1.230	60.27	**22.33**
	MOEA/D	1.557	1.557	158.81	**36.62**
	SPEA2	1.659	1.659	341.12	**78.79**
	NSGA-III	1.822	1.822	950.14	**126.53**
	NSGA-II	1.944	1.944	184.51	**60.14**
WFG2	*IBEA*	1.503	1.503	283.05	**39.53**
	NSGA-III	1.571	1.571	181.02	**50.57**
	MOEA/D	1.683	1.683	328.40	**17.25**
	NSGA-II	1.687	1.687	140.20	**50.57**
	SPEA2	1.859	1.859	879.41	**24.47**
WFG3	*IBEA*	1.889	1.889	1368.77	**269.54**
	NSGA-III	2.609	2.609	32534.96	**855.45**
	NSGA-II	2.630	2.630	3491.16	**440.10**
	MOEA/D	2.820	2.820	37238.82	**1009.81**
	SPEA2	3.178	3.178	5779.25	**425.97**
WFG9	*IBEA*	1.965	2.024	673.26	**23.05**
	NSGA-III	2.194	2.194	348.48	**25.25**
	MOEA/D	2.331	2.331	263.90	**20.80**
	NSGA-II	2.601	2.601	2898.88	**22.89**
	SPEA2	2.759	3.030	748.76	**27.74**

Table 2: Some details about the approximation algorithm search process due to the preference parameter. DTLZ2 and WFG3 are further presented, and they are the slowest experiments. Three column groups are presented with the respective percentile preference: MIP-DoM value, number of clusters suggested by the affinity propagation cluster, and the time spent for each case.

Problem	Algorithms	MIP-DoM *Original*	MIP-DoM Approx. values					Number of clusters					Time spent				
			Preference percentile														
			0.01	*0.05*	*0.50*	*0.95*	*0.99*	*0.01*	*0.05*	*0.50*	*0.95*	*0.99*	*0.01*	*0.05*	*0.50*	*0.95*	*0.99*
DTLZ2	*MOEA/D*	**0.970**	1.013	1.011	0.974	**0.970**	**0.970**	4	5	4	10	24	3.27	5.56	4.85	5.48	46.61
	IBEA	**1.000**	**1.000**	**1.000**	**1.000**	1.009	**1.000**	4	4	5	2	23	3.30	3.34	2.56	64.10	216.54
	NSGA-III	**1.004**	**1.004**	**1.004**	**1.004**	**1.004**	**1.004**	4	4	6	10	26	3.14	3.17	3.12	4.23	278.22
	NSGA-II	**1.013**	**1.013**	1.041	**1.013**	**1.013**	**1.013**	3	2	5	9	26	9.02	5.56	4.48	3.57	230.62
	SPEA2	**1.022**	**1.022**	**1.022**	**1.022**	**1.022**	**1.022**	3	3	4	9	24	11.02	10.17	4.26	2.19	55.26
WFG3	*IBEA*	**1.889**	**1.889**	**1.889**	2.072	**1.889**	2.268	3	3	4	1	7	4.51	5.10	0.94	263.65	0.44
	NSGA-III	**2.608**	3.034	3.034	**2.609**	**2.609**	3.099	2	2	1	1	2	233.94	234.84	621.46	603.89	0.05
	NSGA-II	**2.630**	**2.630**	**2.630**	2.888	2.999	3.081	1	1	4	7	2	423.63	428.06	9.78	6.64	0.05
	MOEA/D	**2.820**	3.058	3.058	**2.820**	3.058	3.232	2	2	1	2	3	237.87	237.59	771.83	238.65	0.11
	SPEA2	**3.178**	**3.178**	**3.178**	**3.178**	**3.178**	**3.178**	2	2	1	6	13	215.01	216.93	188.71	2.62	19.63

compute the percentile 0.05 , as it has generated the same clusters for percentile 0.01, as showed in Table 2.

In Table 2, it is possible to observe that in some cases the number of clusters is 1. It happens for WFG3 in all algorithms, but in different percentile preferences. It means that the original problem will be solved by the MIP-DoM with the same number of solutions in each set. This happens for the DTLZ1 as well, however, for the DTZ1 this is interesting behaviour, because the own problem is easily solved by the original MIP-DoM. This is not the case for WFG3. Due to this possibility, a early stopping procedure was adopted in the approximate MIP-DoM, specifically in Algorithm 2 line 17: if there is no improvement during 180 seconds the MIP-DoM terminates and returns the current solution.

One last interesting observation comes from the SPEA2 in both problem sets, and in DTLZ2 for NSGA-III algorithm. The original value is always obtained, and the percentile variance is innocuous considering the MIP-DoM value.

5 CONCLUDING REMARKS

DoM is a quality indicator that does not need a priori problem knowledge, parameters, or a reference set and is more suitable to multi- and many-objectives scenarios. This paper has analyzed and proposed an approximate MIP-DoM calculation using the affinity propagation cluster algorithm.

Our empirical results have shown that the proposed approximation is faster than the original MIP-DoM calculation, from ~ 9 to $\sim 5,400$ times better, and the average approximation error is 0.40%.

We also empirically presented the effects of the preference parameter. Using a brute force approach, our results demonstrate that this percentile parameter plays an essential role in the approximation method. In the future, a better way should perform different strategies to find a better preference, such as the irace approach [9]. It is still possible to try to predict these values based on some inner solution set features.

Similarly, a parametric study can further show the influence of λ and T, if any, in the approximation quality result.

This MIP-DoM approximation drastically reduced its computational cost. It can improve the DoM applicability in a way to become applicable and useful in the EMO community.

REFERENCES

[1] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. 2012. *Assignment Problems*. Society for Industrial and Applied Mathematics, USA.

[2] Dorin Comaniciu and Peter Meer. 2002. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 5 (May 2002), 603–619. https://doi.org/10.1109/34.1000236

[3] J. Deng and Q. Zhang. 2019. Approximating Hypervolume and Hypervolume Contributions Using Polar Coordinate. *IEEE Transactions on Evolutionary Computation* 23, 5 (Oct 2019), 913–918. https://doi.org/10.1109/TEVC.2019.2895108

[4] Claudio Lucio do Val Lopes, Flávio Vinícius Cruzeiro Martins, and Elizabeth Fialho Wanner. 2020. An Assignment Problem Formulation for Dominance Move Indicator. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. 1–8.

[5] Claudio Lucio do Val Lopes, Flávio Vinícius Cruzeiro Martins, Elizabeth Fialho Wanner, and Kalyanmoy Deb. 2020. Analyzing Dominance Move (MIP-DoM) Indicator for Multi- and Many-objective Optimization. *Submitted for consideration for publication in the IEEE Transactions on Evolutionary Computation* (2020). arXiv:cs.NE/2012.11557

[6] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. 2015. Modified Distance Calculation in Generational Distance and Inverted Generational Distance. (2015), 110–125.

[7] Miqing Li and Xin Yao. 2017. Dominance Move: A Measure of Comparing Solution Sets in Multiobjective Optimization. *CoRR* abs/1702.00477 (2017). arXiv:1702.00477 http://arxiv.org/abs/1702.00477

[8] Miqing Li and Xin Yao. 2019. Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey. *ACM Comput. Surv.* 52, 2, Article 26 (March 2019), 38 pages. https://doi.org/10.1145/3300148

[9] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3 (2016), 43 – 58. https://doi.org/10.1016/j.orp.2016.09.002

[10] D T Pham, S S Dimov, and C D Nguyen. 2005. Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 219, 1 (2005), 103–119. https://doi.org/10.1243/095440605X8298 arXiv:https://doi.org/10.1243/095440605X8298

[11] Lyle Ramshaw and Robert E Tarjan. 2012. On minimum-cost assignments in unbalanced bipartite graphs. *HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1* (2012).

[12] L. Wang, Z. Hao, and W. Sun. 2019. A Novel Self-Adaptive Affinity Propagation Clustering Algorithm Based on Density Peak Theory and Weighted Similarity. *IEEE Access* 7 (2019), 175106–175115. https://doi.org/10.1109/ACCESS.2019.2956963

[13] Kaifeng Yang, Michael Emmerich, André Deutz, and Thomas Bäck. 2019. Efficient computation of expected hypervolume improvement using box decomposition algorithms. *Journal of Global Optimization* 75, 1 (01 Sep 2019), 3–34. https://doi.org/10.1007/s10898-019-00798-7

[14] Yizong Cheng. 1995. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17, 8 (1995), 790–799. https://doi.org/10.1109/34.400568

[15] Mohammed J. Zaki and Wagner Meira, Jr. 2020. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (2 ed.). Cambridge University Press. https://doi.org/10.1017/9781108564175

[16] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation* 7, 2 (April 2003), 117–132. https://doi.org/10.1109/TEVC.2003.810758