International Journal of Computational Intelligence Systems, Vol. 4, No. 4 (June, 2011), 530-549

An Enhanced MOGWW for the bi-objective Quadratic Assignment Problem

Everardo Gutiérrez [1] , Carlos Brizuela [2]

[1] *CICESE Research Center*
Km. 107 Carr. Tijuana-Ensenada
Ensenada, B.C., México
E-mail: egutierr@cicese.mx
[2] *E-mail: cbrizuel@cicese.mx*

Abstract

This paper proposes an enhanced Multi-objective Go with the Winners (MOGWW) algorithm to solve multi-objective combinatorial optimization problems. The original MOGWW algorithm is equipped with the well known Pareto Local Search (PLS) procedure. In order to assess the performance of the hybridization, the non-dominated solutions it generates are compared with the ones generated by each of its components. The algorithms are applied to benchmark instances of the bi-objective Quadratic Assignment Problem. Experimental results show that the hybridized version outperforms both its components, i.e. the original MOGWW algorithm and a PLS variant.

Keywords: Multi Objective Go With the Winners, Bi-objective QAP, Pareto Local Search, Greedy Non-dominated Local Search.

1. Introduction

Most of the important real-world combinatorial optimization problems are actually multi-objective (MO) in nature, however, they are usually reduced to a single-objective or to a family of single objective problems. In this way the widely developed theory and techniques for single objective optimization can be used.[1] Many of these single objective combinatorial problems are hard to solve.[2] Furthermore, it is well-known that some combinatorial problems that can be solved in polynomial time (*i.e.* belong to the class P[3]), in their single-objective version, become NP-hard[a] when an extra objective or a constraint is included.[4] For these Multi-Objective COmbinatorial (MOCO) problems, we need efficient methods that can solve them while considering all their conflicting objectives at the same time.[5,6,7]

Many approaches have been proposed to deal with MOCO problems, most of them are techniques that can be classified as: local search, metaheuristics, and in some cases a hybridization of both of them. Preliminary results of a recently proposed metaheuristic referred to as Multi-Objective Go With the Winners (MOGWW) show that this approach can be an interesting alternative for solving MOCO problems. The MOGWW is an algorithm that uses an acceptance criterion based on the component-wise ordering of the objective value vectors. This algorithm is an extension of the Go With the Winners which was introduced by Aldous and Vazirani[8] as an attempt to give a rigorous expla-

[a]NP-hard: This complexity class contains problems for which we do not know any polynomial time algorithm to solve them.[3]

nation of the behavior of some methods based on a non crossover "survival of the fittest" paradigm. The MOGWW variant is a population based algorithm which needs only three parameters: the number of solutions to be analyzed at the same time, the length "*L*" of a random walk the method needs to perform, and the threshold set to define the fronts the algorithm will get rid of at each iteration. Our contribution here is to give the rationale behind the MOGWW algorithm along with a proposal for an improved hybrid version. The improvement consists of adding a multi-objective local search algorithm once the MOGWW converges. We analyze the hybrid MOGWW over a set of instances of the well-known Quadratic Assignment Problem (QAP) with different input distributions, in particular over instances with different correlation degrees between their flow matrices. We do this because it is well-known that the input's distribution has a strong influence on the algorithm's performance.[9,10]

The QAP is a very important combinatorial optimization problem from a practical[11,12] and theoretical point of view.[3] In the context of location theory, it is the problem of finding an optimal assignment of a set of facilities to a set of locations with given distances between locations and flows between all pair of facilities. Since QAP belongs to the *NP-hard* class,[3] many alternatives such as Local Search (LS) have been proposed to deal with it.[11,12] Moreover, the multi-objective version of this problem has received much attention in the last years, mainly because it models many real problems and because it is becoming a benchmark for new multi-objective heuristics.[13,9,14,15,10,16,17,18,19,20,21] It is worth noting that most of these approaches report their best results when using a combination of heuristics, resulting in what is known as a "hybridization", rather than when using a single one.[14,15,10,16,17,20,21]

Since MOGWW has shown a good performance,[19] even when, in some cases, it could stop prematurely. We modify this algorithm in order to generate a set of solutions that will allow the algorithm to continue the search. Then, we propose to improve the MOGWW, adding a multi-objective local search. It is worth noting that our goal with the hybridization of a MO algorithm is two-fold: first, as in any standard hybridization with local search we want to locally improve solution's quality; second, we use such mechanism to help the algorithm start anew. We study the algorithm's performance over the bi-objective version of the QAP (bQAP). The problem instances come from a generator proposed by Knowles and Corne,[9] for which we have one distance matrix and two flow matrices. We compare the results obtained by the proposed hybridization with each of its components and with the ones belonging to a reference set for this problem.[10]

The remainder of the paper is organized as follows. Section 2 presents some basic definitions about multi-objective problems. Section 3 states the bi-objective QAP and describes previous results. Section 4 explains the MOGWW algorithm, while Section 5 describes its hybrid version. After that, Section 6 shows the experimental setup and results. Finally, Section 7 states the conclusions of this work and some ideas for future research.

2. Multi-objective optimization and performance measures

Most of the important real-world problems are multi-objective (MO) in nature. To deal with them many heuristics have been proposed mainly as extensions of their single-objective counterparts.[6,7] Although the single objective versions have proven to work well in practice, it is not easy to infer that their MO extensions will also perform well. This motivates the analysis of multi-objective algorithms behavior along with the characterization of MO problems difficulty. Assuming, without loss of generality, a minimization problem, MO problems can be formulated as[22,1,5]:

$$minimize\{z_1 = f_1(x), ..., z_J = f_J(x)\} \qquad (1)$$

where J is the number of objective functions, a solution $x = [x_1, ..., x_l] \in \mathbf{X}$ is a vector of discrete variables and $\mathbf{X}$ is the set of feasible solutions. The *image* of a solution in the decision space is a *point*, $z = [z_1, ..., z_J] \in \mathbf{Z}$. Under this setting, there is not a single optimal solution to MO problems but many

"non-dominated" solutions, known as Pareto optimal solutions.[22,1,5] If we assume again minimization problems, the definition of dominance can be stated as follows[22,1,5]:

A point $z^1 \in \mathbf{Z}$ dominates $z^2 \in \mathbf{Z}$, denoted as $z^1 \preceq z^2$, if $\forall_j z^1_j \leqslant z^2_j$, and there is at least one j for which $z^1_j < z^2_j$. Solution x^1 dominates x^2, $x^1 \preceq x^2$, if the image of x^1 dominates the image of x^2. A solution $x \in \mathbf{X}' \subseteq \mathbf{X}$ is *non-dominated* if there is no $x' \in \mathbf{X}'$ such that $x' \preceq x$. A set of solutions $\mathbf{ND} \subseteq \mathbf{X}'$ is referred to as a **non-dominated set** if each $x \in \mathbf{ND}$ is non-dominated. If $\mathbf{X}'$ is the set of all feasible solutions then **ND** is known as **Pareto optimal set**. In this case, the image of **ND** is defined as the **Pareto front**.

Another important concept we will refer to is the **Pareto Local Optima**, which is adapted from Miettinen[1] to deal with combinatorial problems: a feasible solution $x^* \in \mathbf{X}$ is locally Pareto Optimal if, given a neighborhood $\mathbf{N}(x^*)$, x^* is Pareto optimal in $\mathbf{X} \cap \mathbf{N}(x^*)$. Similar definitions can be found elsewhere.[23,24]

When comparing two or more sets of non-dominated solutions, we need to use performance measures that can give us information about the dominance relations between those sets. Many alternatives have been proposed (see the work of Zitzler *et al.*[25] and references therein), however, the problem of deciding which non-dominated set is better is itself an MO problem. We are going to use two of the main performance measures, which are described as follows:

Coverage.[26,25] Let $\mathbf{A}$ and $\mathbf{B}$ be two sets. The function C maps the ordered pair $(\mathbf{A},\mathbf{B})$ to the interval $[0,1]$:

$$C(\mathbf{A},\mathbf{B}) := \frac{|\{b \in \mathbf{B}; \exists a \in \mathbf{A} : a \preceq b\}|}{|\mathbf{B}|}. \qquad (2)$$

A value $C(\mathbf{A},\mathbf{B}) = 1$ means that all points in $\mathbf{B}$ are dominated by or equal to points in $\mathbf{A}$, while $C(\mathbf{A},\mathbf{B}) = 0$ means that none of the points in $\mathbf{B}$ are covered by the set $\mathbf{A}$. $\mathbf{A}$ dominates $\mathbf{B}$ when $C(\mathbf{A},\mathbf{B}) = 1$ and $C(\mathbf{B},\mathbf{A}) = 0$. We will refer to this measure in terms of the percentage of coverage between the sets, *i.e.* $\%cov(\mathbf{A},\mathbf{B}) = C(\mathbf{A},\mathbf{B}) * 100\%$.

Binary ε-indicator.[25] Gives a factor by which a non-dominated set of objective value vectors is worse than another, with respect to all objectives, and is calculated, for a bi-objective problem, as

$$I_\varepsilon(\mathbf{A},\mathbf{B}) := \begin{array}{cc} max & min \\ b \in \mathbf{B} & a \in \mathbf{A} \end{array} max\left(\frac{a_1}{b_1}, \frac{a_2}{b_2}\right), \qquad (3)$$

where a_i and b_i are the ith objective function values of solutions $a \in \mathbf{A}$ and $b \in \mathbf{B}$, respectively. If $I_\varepsilon(\mathbf{A},\mathbf{B}) > 1$ and $I_\varepsilon(\mathbf{B},\mathbf{A}) \leqslant 1$, then we say that the set $\mathbf{B}$ completely dominates $\mathbf{A}$. This measure indicates the minimum ε such that for any solution in $\mathbf{B}$ there is at least one solution in $\mathbf{A}$ that is not worse by a factor of ε in all objectives. This measure can be computed in time $O(n|\mathbf{A}||\mathbf{B}|)$. When comparing more than one pair of sets, we will measure the percentage of times when there is a complete dominance in the comparison. We will refer to this measure as $d_1\%$ if comparing $\mathbf{A}$ with $\mathbf{B}$, and $d_2\%$ if comparing $\mathbf{B}$ with $\mathbf{A}$.

These measures will be used to compare our results with the ones generated by the original MOGWW, the implemented PLS variant, and with the best known up-to-date solutions.

3. The Quadratic Assignment Problem

The Quadratic Assignment Problem, originally proposed by Koopmans *et al.*,[27] is a combinatorial optimization problem and can be described, in the context of location theory, as the problem of assigning a set of facilities to a set of locations with given distances between locations and flows between all pair of facilities. The goal is to find the best assignment of facilities to locations such that the sum of the product between flows and distances is minimal. Many practical problems can be formulated as QAP which can be stated as follows:

Given N facilities, N locations, and two N x N matrices D and E for distances and flows, respectively, then the QAP can be stated as[9]:

$$minimize \quad C(\pi) = \sum_{i=1}^{N}\sum_{j=1}^{N} e_{ij} d_{\pi_i \pi_j}, \qquad (4)$$

where e_{ij} is the flow between facilities i and j, π_i gives the location of facility i in permutation $\pi \in \Pi$, where Π is the set of all permutations of n numbers, $d_{\pi_i\pi_j}$ is the distance between locations π_i and π_j, and $e_{ij}d_{\pi_i\pi_j}$ is the cost contribution of assigning facility i to location π_i and facility j to location π_j.

We consider the multi-objective version of this problem introduced by Knowles and Corne,[9] such that we have one distance matrix D and more than one flow matrices E. This version can be stated as follows:

$$minimize \quad C(\pi) = \{C^1(\pi), C^2(\pi), \cdots, C^m(\pi)\} \quad (5)$$

where

$$C^k(\pi) = \sum_{i=1}^{N}\sum_{j=1}^{N} e_{ij}^k d_{\pi_i\pi_j}, k \in \{1, \cdots, m\}, \quad (6)$$

e_{ij}^k is the kth flow from facility i to facility j, and *minimize* means to obtain all the Pareto optimal solutions. This is an example of a MOCO problem,[5] here the set of feasible solutions X is given by a permutation space Π.

3.1. Related work

Some hybrid metaheuristics have been proposed recently to deal with the single-objective version of QAP.[28,29,30,31,32] Also, QAP instances are used as benchmark to analyze the behavior and performance impact of some specific mechanisms.[29,33] A Cooperative Parallel Tabu Search is proposed by James *et al.*[28] to deal with QAP, using a set of instances from the well known set of benchmark instances QAPLIB[b]. Their conclusions give some guidelines for a better exploitation of parallel mechanism to obtain competitive alternatives when dealing with this problem. The same set of instances is used by the authors in a different work[28] to analyze the impact of a Multi-Start Tabu Search mechanism in terms of diversification and exploitation. The main outcome is that it is more convenient to apply strategic diversification than relying on randomization. Although the ideas of hybridization are central to most of these works, all of them deal with the single-objective version of the QAP. In the following the most relevant works related to the multi-objective version are presented.

A procedure which allows to generate a wide range of instances with different distributions, as well as instances with different degrees of correlation between the flow matrices[c] is proposed by Knowles and Corne.[9] They conjecture that the behavior of different search strategies depends on the parameters used for generating the instances.

A set of bi-objective instances (bQAP)[d] with different correlations in the flow matrices using the generator from Knowles and Corne[9] is generated by Paquete and Stützle.[10] The performance of two different types of stochastic local search (SLS) algorithms, represented by PLS[34] and TPLS[35], for this set of instances are studied. The non—dominated solutions from 500 runs of the multi-objective version of RoTS (W-RoTS)[11] are used as a reference set. This algorithm is one of the best performing algorithms for single-objective QAP,[11,36] however the multi-objective version is computationally expensive.[10] It is concluded by the authors that, the correlation between the flow matrices of the bQAP strongly affects the performance of SLS algorithms. In general, PLS performs better for lower correlation between the flow matrices, however the computation time increases considerably, although not as much as it does for W-RoTS. TPLS also performs better with lower correlation between the flow matrices, and its computation time does not increase as much as with PLS, however the solutions given by TPLS are worse, in most instances, than those given by PLS. Table 1 shows the results obtained by Paquete and Stützle[10] using PLS. Every run of their algorithm is compared with a reference non-dominated front, the one obtained by W-RoTs,[10] and the mean Binary ε-indicator values and the percentage of times that their fronts were completely dominated ($\%dom$)

[b] http://www.opt.math.tu-graz.ac.at/qaplib/
[c] http://dbkgroup.org/knowles/mQAP/
[d] http://eden.dei.uc.pt/p̃aquete/qap/

are reported.

Variants of a multi-objective ant colony optimization (MO-ACO) algorithm and of the Strength Pareto Evolutionary Algorithm (SPEA2) hybridized with an iterative improvement and the Robust Tabu Search (RoTS) algorithm are presented by López-Ibáñez *et al.*[17] Their experimental results show the usefulness of the hybrid algorithms if the available computation time is not too limited. The SPEA2 hybridized with very short tabu search runs is identified as the most promising variant. Moreover, it was also found that the algorithms' performance depends strongly on the instance structure, being SPEA2 the one that showed a higher level of robustness.

Table 1: W-RoTS dominance over PLS $\%dom$, and binary indicators: $\varepsilon_1 = I_\varepsilon(PLS, W-RoTS)$, $\varepsilon_2 = I_\varepsilon(W-RoTS, PLS)$ average running time in seconds $avgt(s)$, and, average number of solutions in the output front $avgn$.

ρ	N	PLS				
		$\%dom$	ε_1	ε_2	$avgt(s)$	$avgn$
0.75	25	100	1.043	0.966	0.0	2.8
	50	100	1.031	0.975	0.0	4.2
	75	100	1.023	0.982	0.1	5.4
0.50	25	100	1.041	0.982	0.0	11.0
	50	100	1.029	0.986	0.1	22.5
	75	100	1.021	0.992	0.6	33.1
0.25	25	100	1.038	0.990	0.0	25.6
	50	100	1.030	0.992	0.3	55.7
	75	100	1.020	0.995	1.5	89.2
0.00	25	100	1.047	0.992	0.0	41.1
	50	98	1.028	0.995	0.5	101.5
	75	91	1.021	0.997	2.8	158.8
−0.25	25	92	1.037	0.996	0.1	69.5
	50	85	1.026	0.998	1.0	160.4
	75	82	1.020	0.999	5.7	273.1
−0.50	25	68	1.036	0.999	0.1	112.2
	50	42	1.030	1.000	2.5	283.3
	75	18	1.021	1.001	16.0	490.3
−0.75	25	0	1.034	1.005	0.4	286.6
	50	0	1.026	1.005	11.4	707.9
	75	0	1.018	1.004	79.2	1195.2

Two multi-objective fast messy genetic algorithms, MOMGA-II and MOMGA-IIa, to solve an application problem that can be modeled as the multi-objective quadratic assignment problem (mQAP) are introduced by Day and Lamont.[15] A set of twenty three instances from Knowles and Corne[9] of three different sizes 10, 20, and, 30, respectively, with two and three objectives are used. A comparison of the results from MOMGA-II and MOMGA-IIa with the ones obtained by a local search,[9] and an exhaustive search approach is presented. The outcome was that MOMGA-IIa finds all Pareto optimal points for problem instances with sizes less than 20. Moreover, it is also concluded that the reason for the dominance of the MOMGA-IIa over the MOMGA-II has to do with the generation and selection mechanisms that allow better multi-objective building blocks to be found.

A variant of the GRASP metaheuristic to solve the mQAP is proposed by Li and Landa-Silva,[37] the method's performance on 18 benchmark instances[9] is analyzed. The results are compared with the ones obtained by other state-of-the-art approaches like NSGA-II, SPEA2, and MOEA/D.[38] The main outcome is that their result are competitive with the ones produced by MOEA/D and outperforms those produced by NSGA-II and SPEA2.

There are two previous works[39,19] closely related to this one. The preliminary ideas for the MO version of the GWW (MOGWW) and its application to the bi-objective permutation flowshop scheduling problem are given by Brizuela and Gutiérrez.[39] On

the other hand, the influence of flow matrices correlation on the performance of the MOGWW, for the set of QAP's instances proposed by Paquete and Stützle,[10] is analyzed in Gutiérrez and Brizuela.[19] The algorithm performance as a function of two design parameters: the random walk length (L) and the number of solutions (P) is studied. The main outcome of these studies is that for instances with positive flow matrices correlation the parameter controlling the algorithm performance is the random walk length L, while the main parameter for negative correlation instances is the number of solutions P. It has been shown that the algorithm shows the disadvantage of stopping when all solutions become non-dominated even though when these solutions contain, in their neighborhoods, better solutions. Therefore, to overcome this situation a combination of the MOGWW algorithm with a multi-objective local search, like PLS,[34] in order to obtain a better meta-heuristic to deal with MOCO problems like the bQAP is proposed.

The objective here is two fold: first, to better explain the rationale behind MOGWW, and second, to analyze whether the MOGWW can be improved by adding a local search procedure. In this case we are strict in the term "improve" which means equal solution's quality at lower computation cost or better quality at the same computation cost.

It is worth noting that there have been many proposals to hybridize multi-objective algorithms with local searches, their main objective has been to reach a local optimum to improve the solution's quality. For instance Jaszkiewicz[40] hybridizes an evolutionary algorithm with a Local Search. The main goal is to improve the quality of offspring solutions after crossover. Another interesting work[41] uses different neighborhood sizes according to the solutions' quality. A more advanced hybridization for the bi-objective QAP is given by López-Ibáñez *et al.*,[17] their SPEA2 hybridized with short taboo search runs obtains one of the most competitive results for this problem. As in the previous case, the aim is to improve the quality of solutions. In our case the main goal is also to get better solutions, however, the local search has also another important goal which is to allow the algorithm to continue for one or more iterations. It does this by getting solutions that are part of different non-dominated fronts, i.e. solutions with different Pareto ranks, which is a condition that forces the algorithm to iterate.

The next section gives a detailed explanation of the MOGWW algorithm and describes the proposed hybridization with a variant of the Pareto Local Search algorithm proposed by Paquete and Stützle.[10]

4. The Multi-Objective Go With the Winners algorithm (MOGWW)

The MOGWW algorithm was proposed as a multi-objective version of the GWW,[8,42] which was originally designed to search graphs with a tree structure. Dimitriou and Impagliazzo[42] introduced a variant of this algorithm that can be used to search graphs that are not trees. With the same philosophy Brizuela and Gutiérrez[39] applied such strategy to design a MO version of the algorithm. However, the rationale behind the algorithm was not completely clarified. Here we deepen in the understanding of this algorithm clarifying and simplifying its description.

Algorithm 1. MOGWW.
Input: A set of P initial solutions. A number L of steps for the random walk. A set of k fronts to be eliminated at each step.
Output: A non-dominated set of P solutions.

> **Step 1**. **Initialization**. Generate P random solutions. Classify the solutions according to their dominance relations, this step resembles to the non-dominated sorting procedure proposed in NSGA-II.[43]
>
> **Step 2**. Until all solutions are non-dominated do:
>
> **Step 3**. **Restriction**. Decrease the threshold by eliminating the k selected fronts
>
> **Step 4**. **Redistribution**. Let q be the number of solutions deleted in the previous step. Make q copies of randomly

selected members of the remaining population to replace the solutions deleted in the previous step.

Step 5. **Randomization**. For every duplicate made in the previous step, execute the following random walk with non-dominated threshold for L steps; select a random neighbor; if the neighbor is not dominated by any solution in the worst remaining front then move to it, otherwise stay at the current solution. Note that selecting a neighbor but not moving to it still counts as one of the L steps in the random walk.

Step 6. Reclassify the solutions according to their dominance relations. Continue to the next stage (Step 2).

This algorithm works with the same principles as its single-objective version introduced by Dimitriou and Impagliazzo[42] does. The main difference with the single-objective version is the introduction of a proper threshold condition (Step 3) and a corresponding non-dominated random walk (Step 5).

The main idea regarding the threshold is as follows. At each stage the solutions are classified in non-dominated fronts, the non-dominated front is assigned the index 1, and the poorest front is assigned the highest index. The simplest threshold would be to select the poorest front and delete all solutions which are in it and replace them with copies of solutions in fronts with lower indices (*i.e.* better ones). An option to generalize, illustrated in **Step 3** of **Algorithm 1**, is to do the replacement of the k selected fronts with copies of solutions coming from the remaining fronts. Another possible approach, not presented in **Algorithm 1**, is to obtain such copies only from a particular set of the remaining fronts (*e.g.* coming only from the j best fronts). The original GWW algorithm decreases the threshold in one unit steps, this is not possible in many problems, therefore we propose instead to select k fronts to be eliminated.

Each of the generated duplicates is the starting point of a length L random walk (steps), at each step the selected neighbor is compared with solutions in the current worst front among the non deleted ones. If the neighbor is not dominated by any solution in the worst front then this become the new solution, otherwise stays at the current solution. In this way it is ensured that the new solutions will be at least as good as the solutions in the current worst front and better than solutions in any of the deleted fronts, in terms of dominance relations. The random walk is the mechanism that allows the algorithm to keep a diverse set of solutions,[e] which is a highly desirable property for an algorithm when dealing with MO problems.[7] The random walk is a sampling mechanism and the main exploration power for the MOGWW.

Points against the algorithm have to do with the length L of the random walk that can make the algorithm to be computationally expensive in problems where the steady state of the random walk can only be reached after a very long walk. Another important problem has to do with its stopping criterion, i.e. it stops when all solutions are non-dominated. The problem with this is that when we deal with big instances, it may be the case that even initial random solutions are non-dominated among each other. This will force the algorithm to stop in the first step producing low quality solutions. Notice that this can happen even when these non-dominated solutions are not locally Pareto optimal. This happens mainly because the non-dominance comparisons consider only the current solutions and not their neighbors, then the set of solutions in the last non-dominated front may have better solutions in their neighborhoods that will not be considered by the algorithm. That is why we propose to add a local search which will help to produce more than one non-dominated fronts, allowing the algorithm to continue the search.

5. Hybrid MOGWW

The MOGWW algorithm iterates until the set of P solutions form a non-dominated set (Step 2, Algorithm 1), without guaranteing that this non-dominated set is, at least, a locally Pareto optimal set. The idea is to add a multi-objective local search

[e] Notice, however, that this mechanism is not as specialized as the ones used in some evolutionary algorithms like NSGA-II[43] or SPEA2[44]

that will help the algorithm to improve the quality of its solutions and to allow it to continue the search. Such procedure will search in the neighborhoods of non-dominated solutions for better ones. This may lead to a new composition of the set of solutions regarding their rankings, i.e. to the generation of solutions of different levels of dominance which will allow the algorithm to continue the search. That is, every time the algorithm reaches a non dominated front (Step 2, Algorithm 1) a local search procedure helps to restart the MOGWW main procedure. The new algorithm will stop if after applying the multi-objective local search it is the case that all solutions are non-dominated. That is, until all the solutions find themselves in a Pareto Local optima. We propose to use a local search based on the Pareto Local Search (PLS) proposed by Paquete *et al.*[34] once the MOGWW stops.

PLS is an extension of local search algorithms for single objective problems.[34] The main modifications from the single objective to the multi-objective case concern the acceptance criterion of new solutions in the local search. The original PLS algorithm is described as follows.

Algorithm 2. PLS.[34]
Input: An initial solution s.
Output: A set F of non-dominated solutions.

> **Step 1**. Set $F = \{s\}$, and set the $visit - bit$ of s to $false$;
>
> **Step 2**. Randomly select a solution s of F whose $visit - bit$ is set to $false$, according to a uniform distribution and evaluate all neighboring solutions $s' \in N(s)$;
>
> **Step 3**. If a s' encountered in the neighborhood of s is non-dominated by any solution in F, it is added to F with its $visit - bit$ set to $false$;
>
> **Step 4**. Once all neighboring solutions of s are examined, the $visit - bit$ of s is set to $true$;
>
> **Step 5**. If solutions exist with the $visit - bit$ set to $false$, repeat **Step 2**, otherwise output F.

We modify PLS in order to use it with the MOGWW once the latter stops. Since the MOGWW ends when all solutions are nondominated and we have P such solutions, then the set F is initialized with P solutions instead of one. The algorithm is also restricted to a final set of P solutions. Once a neighbor s' of s that is non dominated by F is found, we search for a solution in F that is dominated by s' in order to be replaced. If such a solution is not found, the neighbor is not accepted. Each time we finish exploring a solution's neighborhood, its $visit - bit$ is set to $true$, and every time a new neighbor replaces a solution its $visit - bit$ is set to $false$. These modifications allow us to keep a fix number of solutions as output, however this can cause the solutions to gather around the neighborhoods of the first initial solutions. We will refer to this variant of PLS as mPLS and its pseudocode is given as follows.

Algorithm 3. mPLS.
Input: An initial set S of P nondominated solutions produced by the MOGWW.
Output: A set F of P solutions.

> **Step 1**. Set $F = S$, and set the $visit - bit$ of each $s \in F$ to $false$;
>
> **Step 2**. Randomly select a solution s of F whose $visit - bit$ is set to $false$, according to a uniform distribution and evaluate a solution $s' \in N(s)$;
>
> **Step 3**. If s' is nondominated by any solution in F, and s' dominates a solution $s'' \in F$ then replace s'' by s' and set the $visit - bit$ of s' to false. If such a solution is not found then s' is not accepted and another solution of $N(s)$ is selected;
>
> **Step 4**. Once all neighboring solutions of s are examined, and a dominated solution s'' is not found then the $visit - bit$ of s is set to $true$;
>
> **Step 5**. If solutions exist with the $visit - bit$ set to $false$, repeat **Step 2**, otherwise output F.

With this variant the number of solutions is fixed and will not grow as in PLS. Notice, however, that

this modified version of PLS may end with a set of solutions belonging to different non-dominated fronts. At the end the non-dominated property of the solutions will be guaranteed by the MOGWW. The whole procedure that we name HyMOGWW is described by Algorithm 4.

Algorithm 4. HyMOGWW.
Input: A set of P initial solutions. A number L of steps for the random walk. A set of k fronts to be eliminated at each step.
Output: A non-dominated set of P solutions.

> **Step 1**. **Initialization**. Generate P random solutions. Classify the solutions according to their dominance relations, this step resembles to the non-dominated sorting procedure proposed in NSGA-II.[43]
>
> **Step 2**. Until all solutions are non-dominated do:
>
> **Step 3**. **Restriction**. Decrease the threshold by eliminating the k selected fronts
>
> **Step 4**. **Redistribution**. Let q be the number of solutions deleted in the previous step. Make q copies of randomly selected members of the remaining population to replace the solutions deleted in the previous step.
>
> **Step 5**. **Randomization**. For every duplicate made in the previous step, execute the following random walk with non-dominated threshold for L steps; select a random neighbor; if the neighbor is not dominated by any solution in the worst remaining front then move to it, otherwise stay at the current solution. Note that selecting a neighbor but not moving to it still counts as one of the L steps in the random walk.
>
> **Step 6**. Reclassify the solutions according to their dominance relations. If there is only one non-dominated front apply mPLS (**Algorithm 3**) to the set of solutions, take the output from mPLS as the new set of solutions and reclassify them according to their dominance relations. Continue to the next stage (Step 2).

6. Experimental setup and results

The set of instances proposed by Paquete and Stützle[10] is used to test the hybridization of MOGWW algorithm using three different values $N, 2N$, and $4N$ for both parameters: P and L. Then, a total of nine different combinations are used, and 50 runs are performed for each combination. The MOGWW is tuned to delete all solutions in every but the best non-dominated front at each iteration and the deleted solutions are replaced with perturbed copies of solutions in the best non-dominated front (Step 5, Algorithm 4). That is, we do not need to actually perform a non-dominated sorting, we need just to compute the non-dominated set of solutions at each iteration. The neighborhood for this particular problem is the one defined by the swap operator described in Paquete and Stützle,[10] which defines as a neighbor the solution given by swapping the location assignment of two facilities.

6.1. Hybrid MOGWW Analysis

Once the resulting fronts for HyMOGWW are obtained, we compare them with those generated by the original MOGWW, in terms of diversity, in order to analyze the impact of the hybridization over this property. We use the Spread[43] which is a diversity measure of the extent of spread achieved among the obtained solutions. For the most widely and uniformly spreadout set of non-dominated solutions, with respect to a reference set, this metric would take a value of zero. The spread for each resulting non-dominated front of each of the 50 runs and its average are calculated. In case of MOGWW and HyMOGWW we used the fronts obtained when using $P = L$ for $N, 2N, 4N$ and all instance sizes. The reference set is the front generated by W-RoTS.[10]

Table 2 shows the diversity results obtained for the fronts generated by MOGWW and HyMOGWW algorithms. We can see that MOGWW fronts obtained a better (lower) value of Spread for almost

all instance sizes and parameters' values. The differences, shown here as the ratio between HyMOGWW and MOGWW spread values, are closer to one for instances with strong positive correlation, and increase as we go from correlation 0.75 to 0.0 and negative values, finishing with a small decrease for a correlation of -0.75. We can see that the ratios range from 0.99, meaning that HyMOGWW obtained a slightly better value, up to 1.37 for instance size 75, correlation -0.25, and $P = L = 2N$. This diversity results tell us that, as expected, the solutions generated by HyMOGWW are less diverse than the ones generated by MOGWW, when using the W-RoTS as reference front. This is because the local search narrows the search towards specific regions of better objective function values.

Table 2: Spread diversity measure for MOGWW and HyMOGWW (using W-RoTs as reference front).

ρ	N	MOGWW spread			HyMOGWW spread			ratio		
		N	$2N$	$4N$	N	$2N$	$4N$	N	$2N$	$4N$
0.75	25	1.0637	1.1145	1.1363	1.0864	1.1476	1.2704	1.02	1.03	1.12
	50	1.0667	1.1006	1.1216	1.0689	1.1318	1.2498	1.00	1.03	1.11
	75	1.0682	1.0994	1.1150	1.0678	1.1392	1.2228	1.00	1.04	1.10
0.50	25	1.0824	1.2458	1.4152	1.1346	1.2994	1.4644	1.05	1.04	1.03
	50	1.1016	1.2451	1.3886	1.1324	1.2743	1.4379	1.03	1.02	1.04
	75	1.0794	1.2447	1.3917	1.1107	1.2336	1.4082	1.03	0.99	1.01
0.25	25	0.8856	1.0672	1.3069	1.0404	1.1538	1.3763	1.17	1.08	1.05
	50	0.9156	1.0747	1.3161	1.0804	1.1962	1.3561	1.18	1.11	1.03
	75	0.8983	1.0659	1.3089	1.0886	1.1927	1.3428	1.21	1.12	1.03
0.00	25	0.8382	0.9211	1.1676	1.0070	1.0606	1.2299	1.20	1.15	1.05
	50	0.8269	0.8949	1.1436	1.0514	1.0996	1.2304	1.27	1.23	1.08
	75	0.8361	0.8605	1.1174	1.0677	1.1288	1.2233	1.28	1.31	1.09
-0.25	25	0.8163	0.7869	0.9388	0.9709	0.9615	1.0809	1.19	1.22	1.15
	50	0.8279	0.7868	0.9175	1.0277	1.0469	1.0746	1.24	1.33	1.17
	75	0.8408	0.7924	0.8960	1.0444	1.0887	1.0902	1.24	1.37	1.22
-0.50	25	0.8227	0.7679	0.7937	0.9505	0.9433	0.9865	1.16	1.23	1.24
	50	0.8401	0.7962	0.7575	1.0068	1.0034	0.9818	1.20	1.26	1.30
	75	0.8607	0.8064	0.7482	1.0285	1.0541	1.0177	1.19	1.31	1.36
-0.75	25	0.8386	0.8086	0.7591	0.9176	0.9003	0.8850	1.09	1.11	1.17
	50	0.8754	0.8424	0.7882	0.9667	0.9455	0.9227	1.10	1.12	1.17
	75	0.8906	0.8558	0.7998	0.9952	0.9863	0.9634	1.12	1.15	1.20

6.2. HyMOGWW vs. MOGWW and HyMOGWW vs. mPLS

The coverage, and binary epsilon indicators between each one of the non-dominated fronts generated at each run by the original MOGWW, by the mPLS, and by the HyMOGWW are calculated. The main goal is to ensure that the hybridization (HyMOGWW) is better than any of its components (MOGWW and mPLS). The measures were calculated taking one by one the fronts from MOGWW and mPLS, and comparing them with every front from HyMOGWW (one by one). We calculate the average values, the dominance measure is calculated as the percentage of times solutions of one algorithm completely dominates the other's in terms of the binary epsilon indicators,[10] as it was explained in Section 2. We take the 50 averages obtained for each MOGWW front and report the average over these 50 values.

Table 3 shows the results obtained when comparing the MOGWW algorithm with HyMOGWW, using $P = L = N$. The table contains the coverage and dominance relations, the binary indicators, and the ratio between the computation times for different correlations between flow matrices.

The HyMOGWW algorithm shows a strong dominance and coverage over the original algorithm on all instances, however, MOGWW uses only from 5% up to 41% of the time required by the hybrid version. Then, we increase the parameter values for MOGWW in order to make a fair comparison with

HyMOGWW, regarding the computation time. Table 3 shows the results obtained when comparing HyMOGWW using $P = L = N$ and MOGWW using $P = L = 2N$. We can see that HyMOGWW clearly outperforms MOGWW in terms of coverage, dominance and binary indicator values. Moreover, using these parameter values, the computation time required by HyMOGWW was lower for 12 out of 21 instances, bigger for 8 instances, and approximately the same in one. That is, using $P = L = N$ for HyMOGWW and $P = L = 2N$ for MOGWW, the former clearly outperforms the latter using less computation time for more than half of the instances.

We know already that the HyMOGWW outperforms MOGWW, however, can this be due to the contribution of mPLS only and not to the hybridization? To answer this question we compare the results of mPLS with those of HyMOGWW. The results of this comparison are shown in Table 4. The HyMOGWW algorithm shows a strong dominance and coverage over mPLS on all instances, however, mPLS uses less computation time than HyMOGWW for 16 out of 21 instances. Then, in order to balance the computation effort we increase the parameter values for mPLS. Table 4 shows the results obtained when comparing HyMOGWW using $P = L = N$ and mPLS using $N, 2N$ and $4N$. We can see that HyMOGWW clearly outperforms mPLS in terms of coverage, dominance and binary indicator values. The values for $\%d_2$ are all 0.00 except for four instances with correlation $\rho = 0.75$, resulting $\%d_2 = 6$ for two instances of size $N = 25$, and $\%d_2 = 2$ for two instances of size $N = 50$. Moreover, the computation time required by HyMOGWW was lower for 14 out of 21 instances, and 18 out of 21, when mPLS uses a number of solutions equals to $2N$ and $4N$, respectively.

6.3. Comparing HyMOGWW with W-RoTs

Since HyMOGWW outperforms both of its components, we will compare this variant with a well known reference front for the set of instances defined by Paquete and Stützle,[10] the same that we use along this section. Since there is only one reference front from W-RoTs, we calculated the measures by comparing the result from each HyMOGWW run with it and take the average values, following the approach of Paquete and Stützle.[10] The dominance measure represents the percentage of times the W-RoTs front completely dominates the fronts generated by the HyMOGWW in terms of the binary epsilon indicators, as explained in Section 2.

Table 5 shows the results obtained when comparing HyMOGWW using $P = L = \{N, 2N, 4N\}$ and W-RoTs. Since our algorithm neither covered ($\%c_2$) nor dominated ($\%d_2$) the reference front, obtaining 0% for all instances, we do not show those columns in this table. We can see that the coverage over our algorithm slightly decreases by using larger values for the parameters P and L, reaching a minimum $\%c_1$ value of 79.82%, for $P = L = 4N$. On the other hand, in terms of dominance most values keep over 90%, for $\%d_1$, for $P = L = 4N$ values, and only for instances with correlation -0.75 there is a significant reduction of $\%d_1$ reaching even 0% for a couple of instances. Moreover, for 7 out of 21 instances the dominance is 100%, and for 14 of them is bigger than 95%.

Table 3: Comparison results between the HyMOGWW and the original MOGWW algorithm: HyMOGWW coverage over MOGWW $\%c_1$, MOGWW coverage over HyMOGWW $\%c_2$, HyMOGWW dominance over MOGWW $\%d_1$, MOGWW dominance over HyMOGWW $\%d_2$, binary indicators: $\varepsilon_1 = I_\varepsilon(MOGWW, HyMOGWW)$; $\varepsilon_2 = I_\varepsilon(HyMOGWW, MOGWW)$, and, the ratio ($r$) between MOGWW and HyMOGWW average computation times.

ρ	N	MOGWW ($P=L=N$) vs. HyMOGWW ($P=L=N$)							MOGWW ($P=L=2N$) vs. HyMOGWW ($P=L=N$)						
		$\%c_1$	$\%c_2$	$\%d_1$	$\%d_2$	ε_1	ε_2	r	$\%c_1$	$\%c_2$	$\%d_1$	$\%d_2$	ε_1	ε_2	r
0.75	25	88.03	5.40	88.00	0.00	1.021	0.988	0.29	70.12	18.95	64.00	10.00	1.012	0.997	0.93
	50	95.87	1.90	100.00	0.00	1.016	0.990	0.10	83.31	9.81	84.00	2.00	1.009	0.996	0.60
	75	97.49	1.21	98.00	0.00	1.013	0.991	0.17	90.87	4.76	92.00	0.00	1.008	0.996	0.48
0.50	25	97.72	0.38	98.00	0.00	1.032	0.984	0.36	68.23	17.11	34.00	0.00	1.014	1.004	1.29
	50	99.58	0.02	100.00	0.00	1.034	0.984	0.27	88.71	2.28	48.00	0.00	1.012	1.000	0.94
	75	99.89	0.00	100.00	0.00	1.019	0.990	0.41	87.81	0.66	46.00	0.00	1.010	1.000	0.82
0.25	25	99.74	0.00	100.00	0.00	1.043	0.982	0.33	80.26	2.01	6.00	0.00	1.020	1.010	1.67
	50	99.93	0.00	100.00	0.00	1.034	0.984	0.32	82.24	0.04	2.00	0.00	1.016	1.006	1.32
	75	99.92	0.00	100.00	0.00	1.027	0.989	0.34	80.62	0.06	2.00	0.00	1.013	1.005	1.19
0.00	25	99.65	0.00	98.00	0.00	1.053	0.979	0.40	81.15	0.27	0.00	0.00	1.026	1.015	2.20
	50	99.99	0.00	100.00	0.00	1.040	0.982	0.29	79.85	0.03	0.00	0.00	1.019	1.009	1.71
	75	100.00	0.00	100.00	0.00	1.035	0.986	0.26	78.73	0.00	0.00	0.00	1.017	1.009	1.51
−0.25	25	99.75	0.00	100.00	0.00	1.050	0.982	0.30	85.44	0.00	2.00	0.00	1.029	1.014	1.70
	50	99.93	0.00	100.00	0.00	1.041	0.983	0.26	86.48	0.00	0.00	0.00	1.024	1.008	1.61
	75	99.98	0.00	100.00	0.00	1.036	0.985	0.23	86.90	0.00	0.00	0.00	1.022	1.006	1.48
−0.50	25	99.67	0.00	98.00	0.00	1.049	0.981	0.18	88.96	0.00	2.00	0.00	1.031	1.011	1.45
	50	99.95	0.00	100.00	0.00	1.039	0.986	0.17	91.10	0.00	2.00	0.00	1.027	1.006	1.18
	75	100.00	0.00	100.00	0.00	1.033	0.987	0.14	91.54	0.00	0.00	0.00	1.024	1.004	1.00
−0.75	25	99.15	0.00	84.00	0.00	1.033	0.992	0.20	99.70	0.00	12.00	0.00	1.025	1.007	0.80
	50	99.84	0.00	100.00	0.00	1.024	0.991	0.07	94.64	0.00	0.00	0.00	1.019	1.004	0.48
	75	99.97	0.00	100.00	0.00	1.020	0.992	0.05	95.92	0.00	2.00	0.00	1.017	1.003	0.37

Table 4: Comparison results between HyMOGWW and our variant of PLS: HyMOGWW coverage over mPLS $\%c_1$, mPLS coverage over HyMOGWW $\%c_2$, HyMOGWW dominance over mPLS $\%d_1$, binary indicators: $\varepsilon_1 = I_\varepsilon(mPLS, HyMOGWW)$; $\varepsilon_2 = I_\varepsilon(HyMOGWW, mPLS)$, and, the ratio ($r$) between mPLS and HyMOGWW average running times.

ρ	N	mPLS (N) vs. HyMOGWW						mPLS ($2N$) vs. HyMOGWW						mPLS ($4N$) vs. HyMOGWW					
		$\%c_1$	$\%c_2$	$\%d_1$	ε_1	ε_2	r	$\%c_1$	$\%c_2$	$\%d_1$	ε_1	ε_2	r	$\%c_1$	$\%c_2$	$\%d_1$	ε_1	ε_2	r
0.75	25	82.04	10.62	80.00	1.017	0.992	0.29	78.59	13.07	76.00	1.014	0.995	0.50	77.38	14.31	72.00	1.016	0.993	0.93
	50	86.55	6.92	90.00	1.012	0.993	0.27	84.89	8.06	90	1.011	0.993	0.44	87.43	5.68	98	1.013	0.993	0.82
	75	91.58	3.34	98.00	1.010	0.993	0.22	94.42	2.55	100.00	1.011	0.992	0.39	90.87	4.52	98.00	1.009	0.994	0.89
0.50	25	79.16	9.71	76.00	1.022	0.994	0.36	76.65	10.22	68.00	1.022	0.996	0.71	85.91	5.60	86.00	1.023	0.993	1.29
	50	85.90	4.47	92.00	1.015	0.995	0.36	86.01	4.58	92	1.015	0.994	0.70	90.13	1.96	94	1.016	0.994	1.39
	75	90.11	2.10	90.00	1.012	0.994	0.36	89.71	3.49	88.00	1.011	0.995	0.69	92.40	1.36	98.00	1.012	0.994	1.37
0.25	25	75.74	11.55	70.00	1.024	0.997	0.58	73.59	10.12	60.00	1.023	0.999	0.92	72.17	10.77	52.00	1.022	0.999	1.83
	50	84.24	2.04	82.00	1.018	0.996	0.48	82.02	4.95	78	1.018	0.997	1.05	81.23	2.93	76	1.018	0.997	2.08
	75	80.60	1.10	82.00	1.016	0.997	0.51	73.07	2.20	70.00	1.014	0.999	1.10	78.16	2.09	72.00	1.015	0.998	1.97
0.00	25	64.67	10.53	30.00	1.030	1.002	0.70	65.96	10.00	44.00	1.030	1.001	1.30	70.31	7.29	40.00	1.029	1.001	3.00
	50	75.96	2.54	66.00	1.020	0.999	0.66	71.95	4.34	54	1.018	1.000	1.32	72.35	3.95	52	1.019	0.999	2.96
	75	66.00	3.10	50.00	1.016	1.000	0.73	65.94	2.68	54.00	1.016	1.000	1.50	67.47	1.45	44.00	1.016	1.000	3.03
−0.25	25	68.69	6.64	28.00	1.030	1.002	0.80	59.69	9.87	32.00	1.028	1.004	1.70	52.68	15.39	18.00	1.024	1.004	4.10
	50	67.39	3.29	30.00	1.021	1.001	0.90	67.45	3.13	34	1.021	1.001	1.72	61.89	5.77	26	1.020	1.0024	4.55
	75	55.20	2.23	16.00	1.017	1.002	0.99	54.95	1.85	12.00	1.017	1.002	2.19	50.39	3.35	12.00	1.017	1.003	5.01
−0.50	25	50.44	15.49	14.00	1.029	1.004	0.91	56.10	13.12	12.00	1.029	1.005	1.82	52.88	15.18	8.00	1.029	1.006	4.73
	50	64.12	2.68	12.00	1.024	1.003	1.04	55.93	5.73	8	1.021	1.004	2.50	55.84	5.62	6	1.022	1.004	6.57
	75	57.10	2.52	10.00	1.019	1.002	1.32	53.19	3.65	0.00	1.018	1.003	2.94	50.99	4.69	0.00	1.018	1.003	8.10
−0.75	25	52.15	9.57	4.00	1.024	1.003	1.30	42.31	16.04	0.00	1.020	1.005	2.90	31.50	26.68	0.00	1.016	1.006	8.30
	50	56.70	6.56	2.00	1.020	1.002	1.28	56.40	6.20	2	1.020	1.002	2.89	49.18	11.54	2	1.017	1.003	8.69
	75	51.27	3.20	0.00	1.018	1.003	1.27	45.45	4.58	0.00	1.017	1.003	3.28	48.11	4.17	0.00	1.017	1.003	9.06

Table 5: Comparison results between W-RoTs and HyMOGWW: W-RoTs coverage over HyMOGWW $\%c_1$, W-RoTs dominance over HyMOGWW $\%d_1$, binary indicators: $\varepsilon_1 = I_\varepsilon(HyMOGWW, W-RoTs)$; $\varepsilon_2 = I_\varepsilon(W-RoTs, HyMOGWW)$, and, the ratio ($r$) between HyMOGWW and W-RoTs average running times.

ρ	N	HyMOGWW ($P=L=N$) vs. W-RoTs					HyMOGWW ($P=L=2N$) vs. W-RoTs					HyMOGWW ($P=L=4N$) vs. W-RoTs				
		$\%c_1$	$\%d_1$	ε_1	ε_2	r	$\%c_1$	$\%d_1$	ε_1	ε_2	r	$\%c_1$	$\%d_1$	ε_1	ε_2	r
0.75	25	100.0	100.0	1.034	0.974	0.003	100.0	100.0	1.027	0.981	0.013	100.0	100.0	1.020	0.988	0.059
	50	100.0	100.0	1.026	0.980	0.010	100.0	100.0	1.021	0.984	0.039	99.9	98.0	1.017	0.988	0.158
	75	100.0	100.0	1.019	0.986	0.018	100.0	100.0	1.016	0.988	0.071	100.0	100.0	1.012	0.992	0.284
0.50	25	100.0	100.0	1.043	0.981	0.003	98.68	100.0	1.033	0.988	0.013	98.86	98.0	1.025	0.991	0.058
	50	100.0	100.0	1.036	0.983	0.009	99.66	98.0	1.028	0.988	0.036	100.0	100.0	1.022	0.990	0.151
	75	100.0	100.0	1.027	0.989	0.016	100.0	100.0	1.021	0.992	0.065	100.0	100.0	1.016	0.993	0.263
0.25	25	100.0	100.0	1.058	0.985	0.003	99.96	100.0	1.039	0.990	0.012	99.18	98.0	1.032	0.992	0.053
	50	100.0	100.0	1.047	0.988	0.007	100.0	100.0	1.034	0.991	0.034	100.0	100.0	1.026	0.994	0.147
	75	100.0	100.0	1.035	0.991	0.013	99.75	98.0	1.024	0.994	0.060	100.0	100.0	1.018	0.996	0.255
0.00	25	100.0	100.0	1.084	0.987	0.002	100.0	100.0	1.060	0.990	0.011	99.80	98.0	1.044	0.993	0.054
	50	100.0	100.0	1.056	0.989	0.006	100.0	100.0	1.040	0.993	0.031	99.82	98.0	1.026	0.995	0.146
	75	100.0	100.0	1.049	0.993	0.011	100.0	100.0	1.032	0.994	0.054	100.0	100.0	1.021	0.997	0.254
−0.25	25	100.0	100.0	1.093	0.990	0.002	99.84	98.0	1.064	0.994	0.010	99.34	88.0	1.040	0.997	0.054
	50	100.0	100.0	1.069	0.993	0.006	100.0	100.0	1.049	0.994	0.027	99.69	96.0	1.030	0.997	0.144
	75	100.0	100.0	1.059	0.995	0.009	100.0	100.0	1.041	0.995	0.045	99.29	88.0	1.024	0.998	0.247
−0.50	25	99.84	98.0	1.101	0.993	0.003	99.96	98.0	1.077	0.995	0.009	99.38	90.0	1.047	0.997	0.051
	50	100.0	100.0	1.093	0.995	0.006	100.0	100.0	1.072	0.996	0.023	99.87	96.0	1.047	0.998	0.131
	75	100.0	100.0	1.078	0.997	0.009	100.0	100.0	1.062	0.997	0.036	99.63	90.0	1.040	0.999	0.216
−0.75	25	93.60	40.0	1.132	1.001	0.002	90.32	2.0	1.110	1.002	0.008	87.66	0.0	1.082	1.004	0.039
	50	96.84	66.0	1.111	1.000	0.007	96.12	26.0	1.094	1.000	0.022	89.75	2.0	1.075	1.003	0.095
	75	86.72	14.0	1.094	1.001	0.013	86.88	2.0	1.083	1.001	0.035	79.82	0.0	1.066	1.002	0.166

When we compare the results of PLS (Table 1) with ours (Table 5) for $P = L = 4N$, we can see that the PLS results regarding dominance (Table 1, third column; and Table 5, 14th column) are better than ours, however when comparing the epsilon indicators our results are better for positive correlation and some negative ones (Table 1, columns 4 and 5; and Table 5, columns 15 and 16). We can then conclude, in relative terms, that our solutions are of comparable quality to those of PLS. Unfortunately, the PLS results are not available for a direct comparison. It is worth noting that the computation resources of Paquete and Stützle [10] and ours are comparable.

We need now to see how the different correlation values influences the performance measures we have chosen when different sets of P and L values are considered. Figure 1 shows W-RoTS coverage over HyMOGWW and binary indicators between their results. The figures show the results for $P = L = \{N, 2N, 4N\}$ and all instances with different correlations between flow matrices. We can see that for almost all correlations and instance sizes, we can improve the epsilon indicator results, although slightly, by increasing the values of P and L. The improvements given by these changes are more significant for instances with strong negative correlation when we consider the binary epsilon indicator, while for coverage the improvements are greater for small instances.

The binary indicator results seem to contradict the ones obtained for coverage relations. This is because coverage relations only tell us that a set of solutions is covering another set of solutions, they do not give us information about the distance between them. On the other hand, the binary epsilon indicator only gives information about how close the sets are from one another.

The improvement achieved by increasing P and L values lead to larger computation times, as can be seen in Figure 2. The effort grows roughly 10 times from $P = L = N$ to $P = L = 4N$ for the three instance sizes.

After analyzing all these results we can better understand the hybridization effects. The problem we see when using HyMOGWW is that, the main characteristic of the MOGWW is to keep a uniformly distributed set of solutions at each step. Therefore, even in the case we improve each solution, there may be cases where the improved solution is not far from its seed and then, all solutions will be again in a non-dominated front. In this case, however, all solutions are guaranteed to be locally Pareto optimal.

It is worth noting that the instances generator, with high positive correlations produces a Pareto front with few solutions that are close to each other, resembling a single-objective problem. It will be very helpful to generate instances with a Pareto front with few solutions that are far away from each-other, since generating few solutions close to each other induces a similar difficulty to that of single-objective problems.

An analysis of dominance and binary indicators for different correlation values in the flow matrices is also reported in previous works. [10,17] For negative correlations there are many Pareto-optimal solutions and it is easy for an algorithm to find one, however, it is hard to find all of them. Therefore it is likely that two algorithms find non-dominated fronts that are non-dominated between them, resulting in small coverage values and large epsilon binary indicators. On the contrary, for positive correlations there are only few Pareto-optimal solutions and it is difficult to find them. That is, although an algorithm finds very good solutions it is likely to be dominated by a Pareto-optimal solution. This lead to large coverage values and small epsilon binary indicators. Our results then shows that W-RoTS outperforms HyMOGWW in terms of coverage and binary epsilon indicators for all instances. The most favorable results for HyMOGWW occur when we deal with negative correlations and large instances.

It is worth discussing here the relative performance we should expect from the proposed approach, regarding state-of-the-art hybridized approaches like those based on GA or ACO. Given the simplicity of the HyMOGWW it is expected that state-of-the-art approaches [17] (ACO and GA) should outperform it, in terms of solution quality, *i.e.* better non-dominated fronts. This is mainly because the HyMOGWW is a general framework where specific mechanisms can still be inserted. For instance, the random walk mechanism ensures a uni-

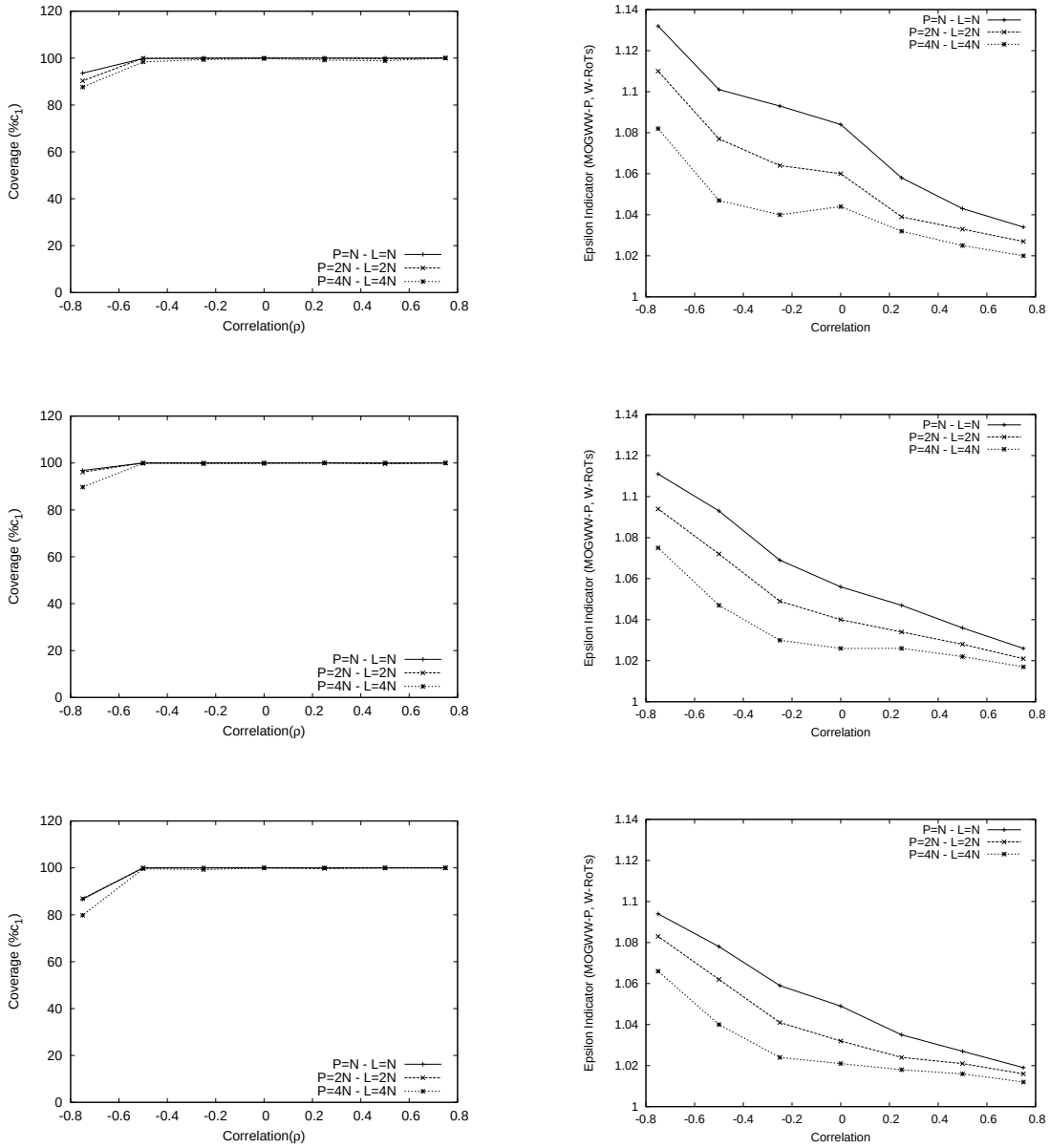

Figure 1: Percentage of W-RoTS coverage over HyMOGWW (left) and Binary Epsilon Indicator $I_\varepsilon(HyMOGWW, W-RoTS)$ (right), instance sizes 25 (top), 50 (middle), and 75 (bottom).

form sampling of the solution space, then we can add a mechanism to better exploit promising regions. The mechanism to perform each random walk step can also be modified to exploit any knowledge we may have on the problem. With the introduction of these modifications we should expect competitive results with those obtained by the state-of-the-art hybridized approaches.

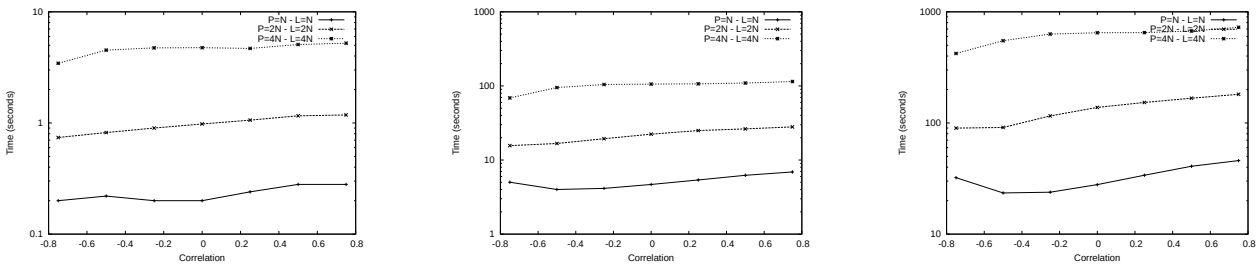

Figure 2: Mean computation time for HyMOGWW, instance sizes 25, 50, and 75.

7. Conclusions

A complete description of the Multi-Objective GWW is presented, and a hybrid version of it has been proposed. The hybridization was focused on adding a multi-objective local search once all MOGWW solutions are in the same non-dominated set. The hybridization is done using a modified version of the PLS algorithm, with a fixed number of solutions. The algorithm was applied to instances of the bi-objective version of the well-known Quadratic Assignment Problem.

The hybrid version outperforms both of its components: the original MOGWW algorithm and the PLS variant it uses, in terms of coverage and binary epsilon indicators (dominance). The results show that, the correlation between flow matrices influences the algorithm's performance as it was previously reported in the literature. For large positive correlations, MOGWW variant was fully dominated by W-RoTS, while for large negative correlations, HyMOGWW is slightly less dominated by W-RoTS. However, for large positive correlation the binary epsilon indicators show that our results were at a small factor from the W-RoTS results.

Future research is planned to explore other means to improve the hybridization. Additionally, a study of the HyMOGWW behavior when a large number of objective functions is considered, and a comparison with hybridizations like MO-ACO and SPEA2 with Taboo[17] are planned.

8. Acknowledgments

This research was supported by the CONACyT under grant C01-45811.

References

[1] K. M. Miettinen. *"Nonlinear Multiobjective Optimization"*. Kluwer Academic Publishers, Boston, 1999.

[2] C. H. Papadimitriou and K. Steiglitz. *"Combinatorial Optimization: Algorithms and Complexity"*. Prentice Hall, Englewood Cliffs, NJ, 1982.

[3] M. R. Garey and D. S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-completeness"*. W. H. Freeman, San Francisco, 1979.

[4] P. Serafini. "Some considerations about computational complexity for multi objective combinatorial problems". In J. Jahn and W. Krabs, editors, *Lecture Notes in Economics and Mathematical Systems, vol. 294*, pages 222–231, Berlin, 1986. Springer.

[5] M. Ehrgott, editor. *"Multicriteria Optimization"*. Springer, 2005.

[6] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, editors. *"Evolutionary Algorithms for Solving Multi-Objective Problems"*. Kluwer, New York, 2002.

[7] K. Deb, editor. *"Multi-Objective Optimization using Evolutionary Algorithms"*. Wiley, Chichester, 2001.

[8] D. Aldous and U. Vazirani. "Go with the winners algorithms". In *35th IEEE Symposium on Foundations of Computer Science*, pages 492–501, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[9] J. D. Knowles and D. Corne. "Instance generators and test suites for the multiobjective quadratic assignment problem". In C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multicriterion Optimization (EMO 2003), LNCS 2632*, pages 295–310, Berlin, 2003. Springer-Verlag.

[10] L. Paquete and T. Stützle. "A study of stochastic local search algorithms for the biobjective QAP with correlated matrices". *European Journal of Operational Research*, 169: 943–959, 2006.

[11] E. D. Taillard. "Comparison of iterative searches for the quadratic assignment problem". *Location Science*, 3:87–105, 1995.

[12] T. Stützle and M. Dorigo. "Local search and metaheuristics for the quadratic assignment problem". Technical Report AIDA-01-01, TU Darmstadt, Germany, 2001. URL `http://iridia.ulb.ac.be/~meta/newsite/downloads/qap.pdf`.

[13] J. D. Knowles and D. Corne. "Towards Landscape Analyses to Inform the Design of Hybrid Local Search for the Multiobjective Quadratic Assignment Problem". In Ajith Abraham, Javier Ruiz del Solar, and Mario Köppen, editors, *HIS*, pages 271–279. IOS Press, 2002.

[14] A. Acan and A. Ünveren. "Evolutionary multiobjective optimization with a segment based external memory support for the multiobjective quadratic assignment problem". In *IEEE Congress on Evolutionary Computation*, pages 2723–2729, Edinburgh, UK, 2005. IEEE.

[15] R. O. Day and G. B. Lamont. "Multiobjective Quadratic Assignment Problem Solved by an Explicit Building Block Search Algorithm - MOMGA-IIa". In Günther R. Raidl and Jens Gottlieb, editors, *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2005, LNCS 3448*, pages 91–100, Berlin, 2005. Springer-Verlag.

[16] I. Borgulya. "An Evolutionary Algorithm for the Biobjective QAP". In *Computational Intelligence, Theory and Applications*, pages 577–586, Berlin, 2006. Springer-Verlag.

[17] M. López-Ibáñez, L. Paquete, and Thomas Stützle. "Hybrid population-based algorithms for the bi-objective quadratic assignment problem". *Journal of Mathematical Modelling and Algorithms*, 5(1):111–137, 2006.

[18] E. Taillard and X. Gandibleux. "Sweep and Path Relinking for the bi-objective QAP". In *22nd European Conference on Operational Research EURO XXII*, 2007.

[19] E. Gutiérrez and C. A. Brizuela. "An experimental study of the multi-objective Go with the Winners algorithm on the biobjective QAP with correlated flow matrices". In *IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC.*, pages 1476–1481. IEEE, 2007.

[20] D. Garrett and D. Dasgupta. "An empirical comparison of memetic algorithm strategies on the multiobjective quadratic assignment problem". In *IEEE symposium on Computational intelligence in multi-criteria decision-making, 2009. mcdm '09.*, pages 80–87, 2009.

[21] C. Lezcano, D. Pinto, and B. Barán. "Team Algorithms Based on Ant Colony Optimization A New Multi-Objective Optimization Approach". In *Parallel Problem Solving from Nature PPSN. LNCS 5199*, pages 773–783, Berlin, 2008. Springer.

[22] R. E. Steuer. *"Multiple Criteria Optimization: Theory, Computation, and Application"*. John Wiley & Sons, New York, 1985.

[23] K. Deb. "Multi-objective genetic algorithms: Problem difficulties and construction of test problems". *Evolutionary Computation*, 7(3): 205–230, 1999.

[24] L. Paquete, T. Schiavinotto, and T. Stützle. "On Local Optima in Multiobjective Combinatorial Optimization Problems". *Annals of Operations Research*, 156(1):83–97, 2007.

[25] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. Fonseca. "Performance assessment of multiobjective optimizers: An analysis and review". *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[26] E. Zitzler and L. Thiele. "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach". *IEEE Transactions on Evolutionary Computation*, 4 (3):257–271, 1999.

[27] Tjalling C. Koopmans and Martin Beckmann. "Assignment Problems and the Location of Economic Activities". *Econometrica*, 25(1): 53–76.

[28] T. James, C. Rego, and F. Glover. "A cooperative parallel tabu search algorithm for the quadratic assignment problem". *European Journal of Operational Research*, 195(3):810–826, June 2009.

[29] T. James, C. Rego, and F. Glover. "Multistart Tabu Search and Diversification Strategies for the Quadratic Assignment Problem". *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(3):579–596, May 2009.

[30] C. Rego, T. James, and F. Glover. "An ejection chain algorithm for the quadratic assignment problem". *Networks*, 56(3):188206, October 2010.

[31] A. S. Ramkumar, S. G. Ponnambalam, N. Jawahar, and R. K. Suresh. "Iterated fast local search algorithm for solving quadratic assignment problems". *Robotics and Computer-Integrated Manufacturing*, 24 (3):392–401, June 2008.

[32] K. Y. Wong and P. C. See. "A hybrid ant colony optimization algorithm for solving facility layout problems formulated as quadratic assignment problems". *Engineering Computations*, 27(1):117–128, 2010.

[33] T. Y. Chen, H. Lin, R. Merkel, and D. Wang. "Verification of optimization algorithms: a case study of a quadratic assignment problem solver". In *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE 2008)*, pages 16–21, San Francisco, California, 2008.

[34] L. Paquete, M. Chiarandini, and T. Stützle. "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study". In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation, LNCS 535*, pages 177–200, Berlin, 2004. Springer-Verlag.

[35] L. Paquete and T. Stützle. "A two-phase local search for the biobjective traveling salesman problem". In C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary multi-criterion optimization (EMO 2003), LNCS 2632*, pages 479–493, Berlin, 2003. Springer-Verlag.

[36] E. M. Loiola, N. M. Maia de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido. "A survey for the quadratic assignment problem". *European Journal of Operational Research*, 176:657690, 2007.

[37] H. Li and D. Landa-Silva. "An Elitist GRASP Metaheuristic for the Multi-objective Quadratic Assignment Problem". In *EMO 2009, LNCS 5467*, pages 481–494, Berlin, 2009. Springer.

[38] Q. Zhang and H. Li. "MOEA/D: A multiobjective evolutionary algorithm based on decomposition". *IEEE Trans. on Evolutionary Computation*, 11(6):712–731, 2007.

[39] C. Brizuela and E. Gutiérrez. "Multi-objective Go with the Winners Algorithm: A Preliminary Study". In C. A. Coello Coello et al., editor, *EMO05, LNCS 3410*, pages 206–220, Berlin, 2005. Springer-Verlag.

[40] A. Jaszkiewicz. "Genetic local search for multi-objective combinatorial optimization". *European journal of operational research*, 137 (1):50–71, 2002.

[41] X. Hu, C. A. Coello Coello, and Z. Huang. "A new multi-objective evolutionary algorithm: neighbourhood exploring evolution strategy". *Engineering optimization*, 37(4): 351–379, 2005.

[42] T. Dimitriou and R. Impagliazzo. "Towards a rigorous analysis of local optimization algorithms". In *25th ACM Symposium on the Theory of Computing*, 1996.

[43] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[44] E. Zitzler, M. Laumanns, and L. Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization". In K.C. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.