PasMoQAP: A Parallel Asynchronous Memetic Algorithm for solving the Multi-Objective Quadratic Assignment Problem

Claudio Sanhueza*, Francia Jimenez*, Regina Berretta* and Pablo Moscato*
*School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW, Australia.
Emails: {claudio.sanhuezalobos, francia.jimenezfuentes}@uon.edu.au, {regina.berretta, pablo.moscato}@newcastle.edu.au

***Abstract*—Multi-Objective Optimization Problems (MOPs) have attracted growing attention during the last decades. Multi-Objective Evolutionary Algorithms (MOEAs) have been extensively used to address MOPs because are able to approximate a set of non-dominated high-quality solutions in a reasonable time. The Multi-Objective Quadratic Assignment Problem (mQAP) is a MOP. The mQAP is a generalization of the classical QAP which has been extensively studied, and used in several real-life applications. The mQAP is defined as having as input several flows between the facilities which generate multiple cost functions that must be optimized simultaneously. In this study, we propose PASMOQAP, a parallel asynchronous memetic algorithm to solve the Multi-Objective Quadratic Assignment Problem. PASMOQAP is based on an island model that structures the population by creating subpopulations. The memetic algorithm on each island individually evolve a reduced population of solutions, and they asynchronously cooperate by sending selected solutions to the neighboring islands. The experimental results show that our approach significatively outperforms all the island-based variants of the multi-objective evolutionary algorithm NSGA-II. We show that PASMOQAP is a suitable alternative to solve the Multi-Objective Quadratic Assignment Problem.**

I. INTRODUCTION

Mathematical models based on Multi-Objective Optimization Problems (MOPs) have been extensively used to address real-world applications [3], [17]. In MOPs, the task we face is to simultaneously satisfy multiple and possibly conflicting objectives. Multi-Objective Evolutionary Algorithms (MOEAs) have received especial attention because they are well-suited to tackle a wide variety of MOPs [28], [30]. The *Pareto optimal set*[1] is approximated using MOEAs by evolving individuals (solutions) typically organized in populations. A population is improved by applying evolutionary operators such as recombination, mutation, and selection. MOEAs augmented with local search operators have attracted growing attention due to their success when applied for solving complex optimization problems. Memetic Algorithms (MAs) are computational methods which combine Evolutionary Algorithms and individual local search techniques in order to efficiently address optimization problems [19].

Despite their advantages, Memetic Algorithms may be computationally more expensive. In order to approximate the Pareto optimal set, the algorithms should explore large search spaces. By increasing the number of objectives, the size of the search space grows dramatically and the number of solutions belonging to the Pareto set will increase accordingly. Therefore, longer convergence times are expected and more evaluations must be performed to find the final Pareto set. This is even more critical in real-world optimization problems for which the evaluation of a given solution is computationally expensive. In general, most of the time in MAs is devoted to the local search procedures which explore user-defined neighborhoods [19]. In the case of combinatorial optimization problems, this neighborhood might be enormous; therefore, an exhaustive exploration is unfeasible.

These drawbacks in MAs can be addressed in two different ways. First, we can limit the neighborhood exploration by relaxing the acceptance criteria. For example, instead of an exhaustive exploration to find the best solution, we could explore the neighborhood until finding a solution that is better than the current one. Second, an expensive operation when exploring neighborhoods is the computation of the objective functions. In this case, for example, we might define surrogate models of the fitness function or we could compute the cost of a neighbor solution based on the method applied to obtain the neighborhood of the current solution.

Typically, parallel and distributed schemes are used to speed up the search. Several parallel approaches for multi-objective optimization have been proposed [16]. Including parallelism is not just a technique to accelerate the search process, but also for developing effective search methods. For example, a population can be divided into subpopulations to explore and exploit different regions of the search space, improving the quality of the obtained Pareto set.

In this study, our contributions are:

- We propose the PASMOQAP, an asynchronous parallel multi-objective memetic algorithm based on island models to solve the multi-objective quadratic assignment problem.
- We evaluate our method using twenty-two benchmark instances. To perform a fair comparison between the techniques, we consider the structure of each instance by varying the correlation between the flows.
- Our results show a significant difference between our approach and the state-of-the-art algorithm to solve multi-

[1]The Pareto optimal set corresponds to a set of non-dominated solutions which may be considered as the answer of a Multi-Objective Optimization Problem.

objective optimization problems. Our method outperforms the parallel variants of the NSGA-II algorithm at a significance level of 5% ($\rho < 0.05$).

The rest of the paper is organized as follows. In Section II, we describe the Multi-Objective Quadratic Assignment Problem and we discuss related works. We present PASMOQAP in Section III. We outline our experiments in Section IV, and we present the results in Section V. We finish with our conclusions and a discussion about future work in Section VI.

II. THE MULTI-OBJECTIVE QUADRATIC ASSIGNMENT PROBLEM

The Multi-Objective Quadratic Assignment Problem (mQAP) is a generalization of the well-known Quadratic Assignment Problem (QAP) [10]. The QAP was first introduced by Koopmans et al. [12] and it has been largely studied [15]. The problem belongs to the $\mathcal{NP}$-hard class and it has proven to be difficult even for small instances. It can be presented as the problem of allocating a set of facilities to a set of locations, with the cost being a function of the distance between locations and the flows between facilities. The goal is to assign each facility to a location such that the total cost is minimized. The multi-objective variation considers more than one flow between any pair of facilities. This leads to the joint minimization of several objective functions.

Formally, the Multi-Objective Quadratic Assignment Problem can be presented as:

$$\underset{\pi \in P_n}{\text{minimize}} \;\; C(\pi) = \{C^1(\pi), C^2(\pi), ..., C^m(\pi)\}$$

$$C^r(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} f^r_{\pi(i)\pi(j)},\; r = 1, ..., m$$

where $f^r_{\pi(i)\pi(j)}$ represents the flow between the facility $\pi(i) = k$ and $\pi(j) = l$ of the r-th flow and d_{ij} is the distance between location i and j. P_n represents the set of all permutations $\pi : N \rightarrow N$. The product $d_{ij}\, f^r_{\pi(i)\pi(j)}$ is the r-th cost of locating facility $\pi(i) = k$ to the location i and facility $\pi(j) = l$ to the location j. The difference between this definition and the original QAP is that we consider different types of flows (m flows in this case) and we are minimizing each of the cost functions simultaneously.

A. Related work

The assumption that the performance of evolutionary algorithms can always be improved in practice by including some kind of local search procedure motivates the study of Garrett and Dasgupta [6]. In their work, a memetic algorithm (SPEA2 + RoTS) to solve mQAP is compared against a multi-objective tabu search. The authors found that there is a correlation between the advantage obtained by hybridizing the MOEA (i.e. SPEA2 + RoTS) versus a simple iterated local search algorithm and the distribution of offspring solutions generated via recombination.

Later, the same authors contributed an empirical comparison of a number of strategies to solve the mQAP [7]. Their results show that both the number of objectives and the specific structure of the search space have a strong impact in the performance of different MOMAs. As the number of objectives is increased, the performance of MOMAs change, with algorithms that stimulate exploration tending to perform better.

In [9], Gutiérrez et al. presented an enhanced variant of Multi-Objective Go with Winners (MOGWW). The MOGWW algorithm was hybridized using a modified version of a Multi-Objective Pareto Local Search (mPLS) [21]. The original Go With Winner (GWW) [1] was improved by including two mechanisms: a proper threshold condition and a non-dominated random walk. With the threshold condition at each stage the solutions are classified in non-dominated fronts. The random walk mechanism allows the approach to maintain a diverse set of solutions, which is a strongly recommended property for an algorithm addressing MOPs.

III. THE PARALLEL MEMETIC ALGORITHM TO SOLVE THE MQAP

In this section, we introduce the proposed PASMOQAP to solve the Multi-Objective Quadratic Assignment Problem. Our method is based on the island model which uses several memetic algorithms to evolve populations of individuals in parallel. The memetic algorithms asynchronously communicate to each other selected solutions as a way to improve the quality of the final Pareto set.

A. Overview

We design each component of PASMOQAP as separate modules which can be easily set by the users. The design allows the creation of flexible island models by just specifying configurations files for each island. Therefore, we can create parallel cooperative strategies, for example, between different algorithms with ease. We identify three main components in our approach. First, we define a topology using an island model that organizes how the memetic algorithms communicate to each other. Second, we design our memetic algorithm, using particular recombination and mutation operators. Also, we implemented an archiving strategy to store the non-dominated solutions found during the execution of the algorithm. Finally, we develop a local search operator which enhances the evolution process of our algorithm. In the following sections, we discuss the details of each component involved in our algorithm.

B. The Parallel Island Model

The parallel island model belongs to the *self-contained parallel cooperation* approach described in [26]. In the self-contained parallel cooperation each processor executes an independent algorithm using a sub-population. Typically, the cooperation between the algorithms is performed by selecting and sending migrants to the other processors. These algorithms are suitable to address problems with large search space. The population is divided into several subpopulations, called *islands* or *demes*. Each island computes an algorithm during

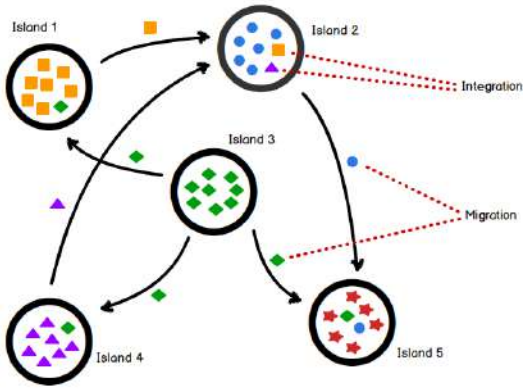

Fig. 1. A representation of an island model. Five islands represented by the circumferences evolves solutions (shapes) and periodically and asynchronously they communicate selected solutions to the neighboring islands.

a given period called *epoch*. After the epoch is reached, a *selection criteria* is applied to choose the individuals that will be migrated between islands. A *migration path* is defined by using a *island-network topology* which determines how the individual can be moved to other islands. Moreover, each island defines its *integration strategy*. The integration strategy is applied every time new individuals arrive from a migration procedure, and it determines if the immigrants should be included in the current island. An example of the island model is depicted in Figure 1.

C. Our approach: PASMOQAP

PASMOQAP is based on ideas from the island model, where the overall population is split into subpopulations, called *islands*. Each island evolves independently using a Memetic Algorithm. In our method, the islands are logically connected to each other, asynchronously migrating solutions at a fixed rate. The immigrant solutions received by the islands are evaluated against the members of the current population. The final goal of our method is to solve the Multi-Objective Quadratic Assignment Problem by creating a cooperation model to improve the quality of the solutions.

In Algorithm 1, we present the pseudocode of PASMOQAP in a specific island. First, the external archive of the i-th island $\mathcal{P}_i^*$ is initialized and the subpopulation P is initiated at random. The main components of our memetic algorithm are executed during a fixed number of generations g_{max}. The CYCLECROSSOVER is executed with probability pb_c and the SWAPMUTATION with probability pb_m. In line 9, the archive is updated with the offspring O, and later the archive is used as input in our local search procedure called DOMBASEDLOCALSEARCH (line 10). Once the local search procedure is executed, the algorithm computes the fitness (DOMDEPTHFITASSIGNMENT) and diversity (FXFCROWDINGDIVASSIGNMENT) measures for each individual. An island receives, without blocking, the migrants M from the neighboring islands I, using the function CHECKMIGRANTS. In line 14, the external archive is updated, including

Algorithm 1 Pseudocode of PASMOQAP in a specific island

Input: population size n_p, set of neighboring islands I, island identification i, number of migrants m, epoch e, number of generations g_{max}, probability of crossover pb_c, probability of mutation pb_m
Output: Final subset of non-dominated solutions $\mathcal{P}_i^*$

1: $\mathcal{P}_i^* \leftarrow \emptyset$
2: $P \leftarrow$ RANDOMINIT(n_p) ▷ Initialize subpopulation P at random
3: $\mathcal{P}_i^* \leftarrow \mathcal{P}_i^* \cup P$ ▷ Initial set of non-dominated solutions
4: $g_c \leftarrow 1$
5: **while** $g_c \leq g_{max}$ **do**
6: $parents \leftarrow$ PARENTSTOURSELECTION(P)
7: $O \leftarrow$ CYCLECROSSOVER$(parents, pb_c)$
8: $O \leftarrow$ SWAPMUTATION(O, pb_m)
9: $\mathcal{P}_i^* \leftarrow \mathcal{P}_i^* \cup O$
10: $P \leftarrow$ DOMBASEDLOCALSEARCH$(\mathcal{P}_i^*)$
11: $P \leftarrow$ DOMDEPTHFITASSIGNMENT(P)
12: $P \leftarrow$ FXFCROWDINGDIVASSIGNMENT(P)
13: $M \leftarrow$ CHECKMIGRANTS(I) ▷ Checking the arrival of migrants from neighboring islands
14: $\mathcal{P}_i^* \leftarrow \mathcal{P}_i^* \cup P \cup M$
15: **if** $g_c \ \% \ e == 0$ **then**
16: $S \leftarrow$ TOURNAMENTSELECTION$(\mathcal{P}_i^*, m)$
17: SENDSOLUTIONS(S, I) ▷ Sending solutions to the neighboring islands
18: **end if**
19: $P \leftarrow$ ELITISTINTEGRATION$(P \cup M)$
20: g_c++
21: **end while**
22: **return** $\mathcal{P}_i^*$ ▷ Return the subset of non-dominated solutions for the island i

the current subpopulation as well as the immigrants. Everytime the epoch (e) is reached, the candidates S to be migrated are selected with the TOURNAMENTSELECTION, and a copy of them is sent to the destination island using SENDSOLUTIONS. In line 19, the subpopulation is updated, replacing some individuals with the immigrants. We employ ELITISTINTEGRATION, where the current population and the immigrants are combined, and then they are sorted using the Pareto dominance relation. This elitist method keeps the currently best solutions for the next iteration, assuring proximity to the optimal Pareto front. Finally, the external archives $\mathcal{P}_i^*$ of each island are collected and mixed to create a global Pareto set.

In the following sections, we present the details of each component of PASMOQAP.

1) Island model: We design PASMOQAP with independent modules which communicate to each other using ad-hoc interfaces to create a topology of islands. We develop in this way to facilitate re-usability of our implementations. Therefore, we can easily perform our experiments, and we can extend our approach for further studies.

a) Topology: We analyze PASMOQAP's performance with a range of islands. In our experiments, we test the

performance of 5, 8, 11, 16, and 21 memetic algorithms running in parallel. To setup a comparable scenario, we implemented a simple communication topology called *complete* where each island is connected to each other. The algorithms asynchronously communicate, sharing promising individuals at a fixed rate.

b) Migration: An island in PASMOQAP sends two promising solutions ($m = 2$) every five generations ($e = 5$). The migration process is performed using two steps. First, the migrants are selected using a selection operator. In particular, we use deterministic tournament [18] between the solutions (TOURNAMENTSELECTION). Second, the algorithm compares the solutions to apply the selection criteria. A *comparator* is an operator that is used to compare the quality of two solutions (i.e. to determine which one is better than the other). Our approach uses a comparator that first compares the fitness values of two solutions (i.e. convergence quality), and in the case of a tie, it compares the diversity values. This comparator is called *FitnessThenDiversity*. We present the details of the computation of the fitness and diversity values in the following sections.

c) Integration: Each time an island receives new individuals, the algorithm must decide whether to integrate the solutions in the current population. We choose to use an ELITISTINTEGRATION strategy that consists of keeping the best individuals in the population. The procedure updates the population by adding the migrant solutions, sorting them with the *FitnessThenDiversity* comparator and resizing the whole population obtained.

d) Fitness assignment: Different schemes exist to assign fitness values to the individuals contained in the population at each generation. We must note that this measure aims to represent the quality of an individual regarding convergence (according to the other solutions of the population or not) and it should not be confused with the objective values of a particular solution. PASMOQAP uses the dominance based fitness assignment strategy called *Dominance Depth fitness assignment* [4], [24]. In DOMDEPTHFITASSIGNMENT, given a population of solutions, the procedure computes the ranking of a solution p by counting the number of solutions C_p (*domination count*) that dominate p, and S_p a set of solutions that the solution p dominates. Thus, the procedure creates groups of non-dominated solution where the first group will have their domination values as zero. Next, for each solution p with $C_p = 0$, the method visits each member $q \in S_p$ and reduces its domination count by one. In doing so, if for any member q the domination count becomes zero the member is saved in a new list Q. These members belong to the second non-dominated group. The procedure is repeated with the members stored in Q for identifying the third group. The process continues until all the groups are found. At the end, each solution will have a ranking value that indicates the position in the ranking of a solution.

e) Diversity assignment: Likewise, different schemes exist to assign diversity measures to the individuals contained in the population at each generation. The measure aims to represent the quality of an individual in terms of diversity. PASMOQAP uses a *front-by-front crowding diversity assignment* strategy which in Algorithm 1 we called FXFCROWDINGDIVASSIGNMENT. The computation of this measure requires sorting the population according to each objective function value in ascending order of magnitude. After that, for each objective function, the boundary solutions (solutions with smallest and largest objective values) are assigned with an infinite distance value. All other intermediate solutions are assigned a distance value equal to the normalized absolute difference in the function values of two adjacent solutions. The computation is repeated in each objective. The overall crowding diversity measure is calculated as the sum of individual distance values corresponding to each objective. The method normalizes each objective value between 0 and 1 before calculating the diversity measure.

2) Memetic Algorithm: In the following sections, we present the details of each component in the memetic algorithm. First, we discuss the details of the evolutionary aspect of our approach. Later, we explain the details of our local search procedure. In particular, we describe the current set selection strategy, the neighborhood exploration strategy, and the stopping criteria.

a) Population: The population on each experiment with 5, 8, 11 islands was defined as $n_p = \lceil 100/N_i \rceil$ where N_i represents the number of islands. We also perform experiment on large island models each one with 16 and 21 islands. To avoid small populations in this case, we define the population size as the mean of our previous experiment i.e. $n_p = 13$.

b) Recombination and mutation: In this study, we test several recombination operators such as *order crossover* (OX) [8], *cycle crossover* [20] (CX), and *partial mapped crossover* (PMX) [23]. The best results were obtained with the *cycle crossover*. The CYCLECROSSOVER preserves the position of the elements in the parent by identifying *cycles* between the two parents. A cycle corresponds to a sequence of elements which is obtained by alternately visiting elements of each parent. The method works as follows. First, randomly select a parent and a cycle starting point. Next, the element at the cycle starting point of the selected parent is inherited by the child. Later, the element in the same position in the other parent cannot then be placed in this position. Then, its position is found in the selected parent and is inherited from that position by the child. Repeat the process until the cycle is completed by finding the first item in the non-selected parent. To select the two parents for applying the recombination operator we use the tournament selection procedure.

To form the children, the cycles are copied from the respective parents in alternating order, i.e. in cycle 1, the elements of parent 1 are copied to child 1, while in cycle 2 the elements of parent 1 are copied to child 2, and so on. It is important to note that cycle crossover always keeps the position of items from one parent or the other without any alteration. We applied the crossover operator with a probability of $pb_c = 0.9$. An example of the operator is depicted in Figure 2.

To explore the search space of solutions, we use the

Cycle	Parents	Elements	Children
1	P1: 8 4 7 3 6 2 5 1 9 0 P2: 0 1 2 3 4 5 6 7 8 9	8, 0, 9	C1: 8 - - - - - - - 9 0 C2: 0 - - - - - - - 8 9
2	P1: 8 4 7 3 6 2 5 1 9 0 P2: 0 1 2 3 4 5 6 7 8 9	4, 1, 7, 2, 5, 6	C1: 8 1 2 - 4 5 6 7 9 0 C2: 0 4 7 - 6 2 5 1 8 9
3	P1: 8 4 7 3 6 2 5 1 9 0 P2: 0 1 2 3 4 5 6 7 8 9	3	C1: 8 1 2 3 4 5 6 7 9 0 C2: 0 4 7 3 6 2 5 1 8 9

Fig. 2. Cycle Crossover (CX) operator example. We highlight the elements to indicate how the values from the parents are copied to the children in every cycle.

SWAPMUTATION operator with probability $pb_m = 0.01$. The operator consists of exchanging two randomly selected elements so that each occupies the location formerly occupied by the other.

c) Archive: PASMOQAP uses an external archive during the optimization process. An archive is a secondary population that stores non-dominated solutions found since the beginning of the optimization process. It aims first at preserving these solutions by updating the archive at each generation. But, it is also possible to include archive's members during the selection phase of the evolutionary algorithm, to save the archive's objective vectors into a file every generation, and/or to compute performance metrics on this archive. We must bear in mind that the default dominance relation used to update the archive is the Pareto dominance relation, but other relations (typically relaxations of the Pareto dominance) can be specified. In particular, our approach employs an archive with a limited number of solutions to avoid an exponential grow in the size of the archive. The new non-dominated solutions are included in the archive according to the Pareto dominance relation.

3) Local Search: After applying the evolutionary operators, our approach runs a *Dominance Based Local Search* (DMLS) algorithm [14]. A DMLS algorithm defines both problem-related and problem-independent modules. Next, we discuss the current set selection strategy, the neighborhood exploration strategy, and the stopping criteria. These strategies belong to the problem-independent modules. We present the details of the local search procedure used in PASMOQAP in Algorithm 2.

a) Current set selection: The first step of a local search procedure deals with the selection of a set of solutions from which the neighborhood will be explored. In general, a DMLS model could apply two strategies: *exhaustive selection* and *partial selection*. In the former, the whole set of solutions stored in the archive is selected. On the other hand, only a subset of solutions is chosen in a partial selection. Such a set may be selected at random, or also with respect to a diversity measure. The local search procedure used in PASMOQAP applies an exhaustive selection of individuals. During each iteration of the local search procedure, an unvisited solution is selected at random from the archive using the SELECTSOLUTION procedure. This *current set* is the starting point for the next step in our local search procedure, the neighborhood exploration strategy.

Algorithm 2 Pseudocode of the DOMBASEDLOCALSEARCH procedure

Input: archive of the island $\mathcal{P}_i^*$
Output: A subpopulation P

```
Mark s unvisited ∀s ∈ P_i^*
P ← P_i^*
t_0 ← CURRENTTIME(clock)
t_c ← 0
while (t_c < t_max) or (P_i^* = ∅) do
    s ← SELECTSOLUTION(P_i^*)
    for each s' ∈ ORDERSWAPEXPLORATION(s) do
        if s' dominates s then
            s'.visited ← false
            P ← P ∪ s'
            s.visited ← true
            break
        end if
    end for
    P_i^* ← GETUNVISITED(P ∪ P_i^*)
    t_c ← CURRENTTIME(clock) − t_0
end while
return P
```

b) Neighborhood exploration: From the current set, candidate solutions must be generated using a neighborhood structure. The neighborhood is obtained by applying a local transformation to every solution belonging to the current set. We use the swap operator to define our neighborhood. This operator interchanges two selected facilities from their original positions. We use this operator because we can compute in $\mathcal{O}(m \times n)$ the objectives functions of a new solution by just computing the difference between the original and a new solution.

Given the solution π and the locations i and j of the elements to be exchanged, the difference δ in the r-th objective function value when exchanging the elements $\pi(i)$ and $\pi(j)$ is computed as follows [25].

$$\begin{aligned}\delta^r(\pi,i,j) = (d_{ii} - d_{jj})(f^r_{\pi(j)\pi(j)} - f^r_{\pi(i)\pi(i)}) + \\ (d_{ij} - d_{ji})(f^r_{\pi(j)\pi(i)} - f^r_{\pi(i)\pi(j)}) + \\ \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \Big((d_{ki} - d_{kj})(f^r_{\pi(k)\pi(j)} - f^r_{\pi(k)\pi(i)}) + \\ (d_{ik} - d_{jk})(f^r_{\pi(j)\pi(k)} - f^r_{\pi(i)\pi(k)})\Big)\end{aligned}$$

Our local search procedure uses an ORDERSWAPEXPLORATION strategy. In this case, the first selected pair of facilities are located in the positions 0 and 1, the second pair of facilities will be located in positions 0 and 2, and so on.

In general, we can identify two classes of neighborhood exploration. An *exhaustive neighborhood exploration* generates every possible move and the neighboring solutions are all added to the candidate set. A *partial neighborhood exploration* generates a subset of moves. The PASMOQAP local search

procedure uses the *FirstImproving* strategy which belongs to the partial exploration class. The method explores the neighborhood by transforming the current solution s, and incrementally computes the objective functions of the transformed solution s'. If the transformation generates a better solution (s' *dominates* s, line 8 in Algorithm 2), the method accepts the new solutions to be included in the subpopulation, and finishes the exploration for the solution s. Finally, the subpopulation is updated with the solutions of the archive, and the procedure keeps only the unvisited solutions for the next iteration (GETUNVISITED).

c) Stopping criteria: In PASMOQAP, the local search procedure is allowed to run during 5 seconds (t_{max}) or until all the solutions in the archive are visited ($\mathcal{P}_i^* = \emptyset$). In this way, we avoid the local search monopolize the execution of the algorithm during long periods.

IV. EXPERIMENTAL DESIGN

In this section, we provide an overview of the experimental design used in our study to evaluate PASMOQAP. We present the set of instances which we use to evaluate our method, the baseline algorithm that we use to perform the comparisons, and the metric that we compute to measure convergence of the different approaches.

A. Instances

We evaluate PASMOQAP with a set of twenty-two 60 facility benchmark instances defined in [7]. We re-create the instances using the instance generator presented in [11] with the parameters that are shown in Table I. The set of instances includes Multi-Objective Quadratic Assignment Problems instances with two, three and four objectives. We consider the problems' structure by varying the *correlation* between flow matrices. In *uniform* instances, the flow values are uniformly distributed across the matrices; hence, the instances represent less realistic `mQAPs`. On the other hand, the *real-like* instances include sparsity during the generation, representing more realistic instances of the `mQAP` [11]. We use this variety of instances to ensure that the algorithms are not being selected because of good performances on a small set of very similar benchmark instances.

TABLE I
BENCHMARK MULTI-OBJECTIVE QAP PROBLEM INSTANCES.

Objectives	Instance	Size	Type	Correlation
2	Gar60-2fl-1uni	60	uniform	-0.3
	Gar60-2fl-2uni	60	uniform	0
	Gar60-2fl-3uni	60	uniform	0.3
	Gar60-2fl-4uni	60	uniform	-0.8
	Gar60-2fl-5uni	60	uniform	0.8
	Gar60-2fl-1rl	60	real-like	-0.3
	Gar60-2fl-2rl	60	real-like	0
	Gar60-2fl-3rl	60	real-like	0.3
	Gar60-2fl-4rl	60	real-like	-0.8
	Gar60-2fl-5rl	60	real-like	0.8
3	Gar60-3fl-1uni	60	uniform	0
	Gar60-3fl-2uni	60	uniform	-0.5
	Gar60-3fl-3uni	60	uniform	0.5
	Gar60-3fl-1rl	60	real-like	0
	Gar60-3fl-2rl	60	real-like	-0.5
	Gar60-3fl-3rl	60	real-like	0.5
4	Gar60-4fl-1uni	60	uniform	0
	Gar60-4fl-2uni	60	uniform	-0.5
	Gar60-4fl-3uni	60	uniform	0.5
	Gar60-4fl-1rl	60	real-like	0
	Gar60-4fl-2rl	60	real-like	-0.5
	Gar60-4fl-3rl	60	real-like	0.5

B. Baseline

To evaluate PASMOQAP, we compare it against a well-known multi-objective evolutionary algorithm called `NSGA-II` [4]. This algorithm has been extensively used for tackling multi-objective optimization problems; therefore, it is considered the state-of-the-art algorithm [22], [27]. To be fair in our comparisons, we implemented a baseline island models using the `NSGA-II` algorithm on each island.

C. Performance assesment

To quantify the convergence of our method we used the *hypervolume* indicator proposed by Zitzler et al. [28], [30]. The hypervolume quantifies the volume, in the objective space, of the space dominated by the obtained non-dominated Pareto optimal solutions bounded by a *reference point*. The hypervolume has been proven to be Pareto compliant [5], [29], i.e., it does not contradict the order induced by the Pareto dominance relation. Therefore, higher values of the hypervolume indicator mean better performances.

V. COMPUTATIONAL RESULTS

We implemented PASMOQAP in C++ using the framework ParadisEO [2], [13]. The experiments were performed in the University of Newcastle's Research Compute Grid that contains a Cluster of 32 nodes Intel® Xeon® CPU E5-2698 v3 @ 2.30 GHz x 32 with 128 GB of RAM.

We execute 30 independent trials for each benchmark instance. The algorithms were allowed to run for 100 generations or during 300 seconds (5 min.). Then, we get a reference point per instance from a global Pareto set that we compute by combining the results of each trial. We use each reference point to measure the hypervolume in each computed Pareto set. Finally, we report the average normalized hypervolume metric. All the objectives values were normalized between 0 and 1 before performing the hypervolume computation.

In our experiments, we evaluate ten different island-based algorithms. Five correspond to the baseline `NSGA-II` island model using 5, 8, 11, 16, and 21 islands and five correspond to PASMOQAP with the same number of islands. The best hypervolume indicators for each instance are highlighted in Table II. Comparing the results between island models with the same number of islands, we observe that PASMOQAP consistently outperforms the baseline `NSGA-II` method in terms of the hypervolume with a just few exceptions where `NSGA-II` obtain better results.

TABLE II

MEAN OF THE NORMALIZED HYPERVOLUME INDICATOR. WE EVALUATED FIVE NSGA−II ISLAND MODELS USING 5, 8, 11, 16, AND 21 ISLANDS. WE ALSO EVALUATE FIVE PASMOQAP ISLAND MODELS USING THE SAME NUMBER OF ISLANDS. THE BEST RESULTS PER INSTANCE ARE HIGHLIGHTED.

	NSGA-II					PASMOQAP				
Instances	5	8	11	16	21	5	8	11	16	21
Gar60-2fl-1uni	0.8659	0.8745	0.8399	0.8570	0.8838	0.8790	0.8913	0.9128	0.8714	0.9080
Gar60-2fl-2uni	0.8529	0.8388	0.8421	0.8407	0.8551	0.8519	0.8749	0.8961	0.8587	0.9127
Gar60-2fl-3uni	0.8130	0.7893	0.8094	0.8136	0.8025	0.8415	0.8107	0.8673	0.8155	0.8868
Gar60-2fl-4uni	0.7846	0.7788	0.7656	0.7543	0.7613	0.8043	0.7944	0.8001	0.7677	0.8040
Gar60-2fl-5uni	0.3803	0.4031	0.3988	0.4472	0.4993	0.5037	0.3529	0.5188	0.3235	0.5255
Gar60-2fl-1rl	0.8952	0.8294	0.8598	0.9022	0.8757	0.9180	0.9169	0.9524	0.8697	0.9393
Gar60-2fl-2rl	0.8860	0.8657	0.8595	0.8307	0.8449	0.9259	0.9144	0.9473	0.8685	0.9419
Gar60-2fl-3rl	0.9010	0.8705	0.8852	0.8607	0.8852	0.8945	0.8812	0.9209	0.8797	0.8996
Gar60-2fl-4rl	0.8908	0.8700	0.8681	0.8840	0.9035	0.9307	0.9202	0.9428	0.9090	0.9325
Gar60-2fl-5rl	0.7734	0.8264	0.7456	0.7725	0.7241	0.8740	0.8376	0.8704	0.8032	0.8253
Gar60-3fl-1uni	0.8493	0.8332	0.8694	0.9090	0.9039	0.8550	0.8324	0.8910	0.8963	0.8843
Gar60-3fl-2uni	0.7633	0.7539	0.7703	0.7190	0.7308	0.8010	0.7838	0.7963	0.7398	0.7737
Gar60-3fl-3uni	0.8004	0.7531	0.7662	0.7789	0.7532	0.8494	0.8022	0.8730	0.7961	0.7268
Gar60-3fl-1rl	0.8802	0.8565	0.9272	0.9513	0.9194	0.8855	0.9009	0.9157	0.9460	0.9476
Gar60-3fl-2rl	0.8537	0.8506	0.8872	0.9324	0.8648	0.8703	0.8991	0.9006	0.9275	0.8744
Gar60-3fl-3rl	0.8579	0.8687	0.8719	0.8858	0.9107	0.8962	0.8876	0.9048	0.9107	0.8932
Gar60-4fl-1uni	0.7817	0.8123	0.8726	0.8252	0.8000	0.8134	0.8111	0.9083	0.8556	0.8251
Gar60-4fl-2uni	0.6892	0.7237	0.6920	0.6170	0.6133	0.7019	0.7184	0.6679	0.6480	0.6465
Gar60-4fl-3uni	0.7920	0.7764	0.7916	0.7893	0.7993	0.8017	0.7642	0.8729	0.7938	0.8995
Gar60-4fl-1rl	0.7264	0.8759	0.9181	0.8466	0.8488	0.8221	0.8684	0.9200	0.9018	0.8692
Gar60-4fl-2rl	0.8083	0.8599	0.9145	0.8354	0.8091	0.8338	0.8553	0.9131	0.8728	0.8640
Gar60-4fl-3rl	0.8015	0.8569	0.8914	0.8842	0.8557	0.8248	0.8275	0.9201	0.8783	0.8823

We apply the Wilcoxon rank sum test to the mean hypervolume indicator values to determine if there are significant differences between the algorithms at a significance level of 5%. First, we perform a pairwise comparison between NSGA−II and PASMOQAP when they use the same number of islands. Our results suggest that there are no significant differences between the algorithms with 5, 8, 16, and 21 islands. We found significant differences ($\rho = 0.0074$) between the approaches with 11 islands. Second, we use PASMOQAP with 11 islands to compare against the rest algorithms which use a different number of islands. Our statistical tests show that PASMOQAP with 11 islands significantly outperforms the whole set of NSGA−II island models. Moreover, PASMOQAP with 11 islands is significantly better than the other PASMOQAP with 5, 8, and 16 islands. No significant differences were found against PASMOQAP with 21 island. Therefore, our results suggest the best approach is PASMOQAP with 11 islands.

Moreover, our results show that at some point increasing the number of islands do not generate benefits during the approximation of the Pareto optimal set. However, our results also suggest that larger island models can be useful for addressing bi-objective uniform instances.

We show in Figure 3 two Pareto fronts of bi-objective instances. We compute the Pareto fronts combining the 30 trials of the uniform instance Gar60-2fl-1uni and the real-like instance Gar60-2fl-1rl. We can see in both cases how PASMOQAP computes better sets of non-dominated solutions.

In summary, we observe that PASMOQAP achieve a high performance compare to NSGA−II in uniform and real-like instances using the hypervolume indicator. For this reason, PASMOQAP is a promising alternative for solving multi-objective quadratic assignment problems.

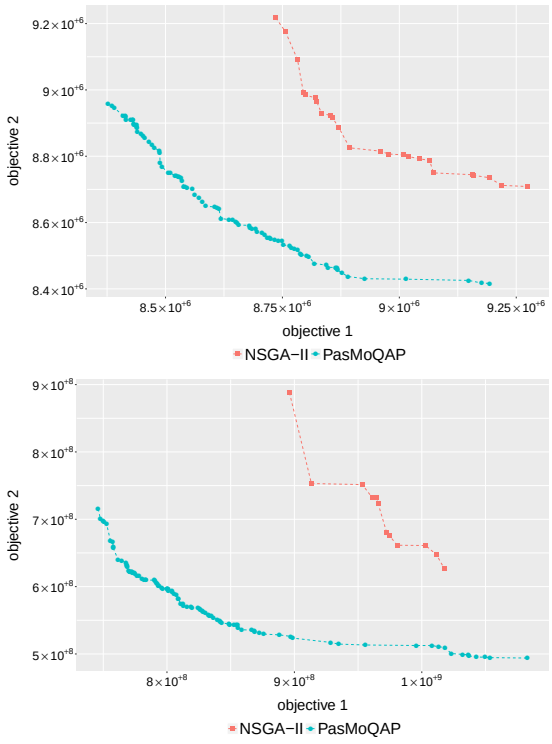

Fig. 3. The global Pareto fronts of (top) the bi-objective uniform instance Gar60-2fl-1uni and (bottom) the bi-objective real-like instance Gar60-2fl-1rl. In both cases, PASMOQAP computes better Pareto sets.

VI. CONCLUSIONS AND FUTURE WORK

We propose the PASMOQAP a parallel memetic algorithm for tackling the Multi-Objective Quadratic Assignment Problem. Our approach is based on the island model with small

subpopulations. Each island evolves independent subpopulations, communicating asynchronously the promising solutions to the neighboring islands according to a user-defined topology. We preserve the diversity through external archives of limited size. We compare PASMOQAP against a parallel version of the well-known multi-objective evolutionary algorithm `NSGA-II`. The results show that PASMOQAP significantly outperforms the baseline method. We observe that our proposal is a viable alternative for solving the Multi-Objective Quadratic Assignment Problem at a reasonable computational time.

Further studies are required using more benchmark instances and more objectives (i.e. many objectives). We are interested in analyzing with more details the effects of the parameters involved in the island models (i.e. topologies, migration and integration techniques) that might give us more insights into the design of new parallel memetic algorithms for addressing the `mQAP`. Moreover, we believe that these parallel models might be improved if we explore the search space with different algorithms. These methods are called heterogeneous island models, and they might help to find a better balance between the algorithms' exploration and exploitation features. More empirical studies in this direction may help to identify advantages and disadvantages of each approach and how they can collaborate to approximate the optimal Pareto set.

REFERENCES

[1] D. Aldous and U. V. Vazirani. "Go with the winners" algorithms. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 492–501, 1994.

[2] S. Cahon, N. Melab, and E. Talbi. ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *J. Heuristics*, 10(3):357–380, 2004.

[3] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

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

[5] C. Fonseca, J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. In *Third International Conference on Evolutionary Multi-Criterion Optimization*, volume 216, page 240, 2005.

[6] D. Garrett and D. Dasgupta. Analyzing the performance of hybrid evolutionary algorithms for the multiobjective quadratic assignment problem. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 1710–1717, 2006.

[7] 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, MCDM 2009, Nashville, TN, USA, March 30 - April 2, 2009*, pages 80–87. IEEE, 2009.

[8] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[9] E. Gutiérrez and C. A. Brizuela. An enhanced MOGWW for the biobjective quadratic assignment problem. *Int. J. Computational Intelligence Systems*, 4(4):530–549, 2011.

[10] J. D. Knowles and D. Corne. Towards landscape analyses to inform the design of hybrid local search for the multiobjective quadratic assignment problem. In A. Abraham, J. Ruiz-del-Solar, and M. Köppen, editors, *Soft Computing Systems - Design, Management and Applications, HIS 2002, December 1-4, 2002, Santiago, Chile*, volume 87 of *Frontiers in Artificial Intelligence and Applications*, pages 271–279. IOS Press, 2002.

[11] J. D. Knowles and D. Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings*, volume 2632 of *Lecture Notes in Computer Science*, pages 295–310. Springer, 2003.

[12] T. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.

[13] A. Liefooghe, L. Jourdan, T. Legrand, J. Humeau, and E. Talbi. ParadisEO-MOEO: A software framework for evolutionary multi-objective optimization. In *Advances in Multi-Objective Nature Inspired Computing*, volume 272 of *Studies in Computational Intelligence*, pages 87–117. Springer, 2010.

[14] A. Liefooghe, S. Mesmoudi, J. Humeau, L. Jourdan, and E.-G. Talbi. A study on dominance-based local search approaches for multiobjective combinatorial optimization. In *International Workshop on Engineering Stochastic Local Search Algorithms*, pages 120–124. Springer, 2009.

[15] E. M. Loiola, N. M. M. de Abreu, P. O. B. Netto, P. Hahn, and T. M. Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007.

[16] F. Luna and E. Alba. Parallel multiobjective evolutionary algorithms. In J. Kacprzyk and W. Pedrycz, editors, *Springer Handbook of Computational Intelligence*, pages 1017–1031. Springer, 2015.

[17] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[18] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[19] F. Neri, C. Cotta, and P. Moscato, editors. *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*. Springer, 2012.

[20] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge, MA, USA, July 1987*, pages 224–230. Lawrence Erlbaum Associates, 1987.

[21] L. Paquete, M. Chiarandini, and T. Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In *Metaheuristics for Multiobjective Optimisation*, pages 177–199. Springer, 2004.

[22] M. Pilát. Evolutionary multiobjective optimization: A short survey of the state-of-the-art. *Proceedings of the Contributed Papers Part I-Mathematics and Computer Sciences, WDS, Prague, Czech*, pages 1–4, 2010.

[23] S. N. Sivanandam and S. N. Deepa. *Introduction to genetic algorithms*. Springer, 2008.

[24] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3):221–248, Sept. 1994.

[25] E. D. Taillard. Comparison of iterative searches for the quadratic assignment problem. *Location science*, 3(2):87–105, 1995.

[26] E. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. A. C. Coello. Parallel approaches for multiobjective optimization. In *Multiobjective Optimization, Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*, pages 349–372. Springer, 2008.

[27] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[28] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer, 1999.

[29] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 862–876. Springer, 2007.

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