# Double Archive Pareto Local Search

Oded Maler
CNRS-VERIMAG
University of Grenoble Alpes, France
Email: oded.maler@imag.fr

Abhinav Srivastav
VERIMAG
University of Grenoble Alpes, France
Email: abhinav.srivastav@imag.fr

***Abstract*—Many real-world problems have multiple, conflicting objectives and a large complex solution space. The conflicting objectives give rise to a set of non-dominating solutions, known as the Pareto front. In the absence of any prior information on the relative importance of the objectives, none of these solutions can be said to be better than others, and they should all be presented to the decision maker as alternatives. In most cases, the number of Pareto solutions can be huge and we would like to provide a good representative approximation of the Pareto front. Moreover, the search space can be too large and complex for the problem to be solved by exact methods. Therefore efficient heuristic search algorithms are needed that can handle such problems.**

**In this paper, we propose a double archive based Pareto local search. The two archives of our algorithm are used to maintain $(i)$ the current set of non-dominated solutions, and $(ii)$ the set of promising candidate solutions whose neighbors have not been explored yet. Our selection criteria is based on choosing the candidate solutions from the second archive. This method improves upon the existing *Pareto local search* and *queued Pareto local search* methods for bi-objective and tri-objective quadratic assignment problem.**

## I. Introduction

The multi-criteria approach to decision making is based on the observation that many real-world optimization problems admit multiple conflicting objectives, and hence, different solutions can be incomparable and show different possible trade-offs between the objectives. Without any a priori information about the decision maker's preferences and weights, multi-objective problems are usually addressed by providing the set of all non-dominated solutions. Specifically, a solution $s$ is said to *dominate* solution $s'$ only if it is at least as good as $s'$ in all cost criteria and is strictly better in at least one criterion. The set containing all such non-dominated (and mutually incomparable) optimal solutions is known as the *Pareto front*.

For hard problems having a large solution space, computing the Pareto front is intractable. In such cases, one has to be content with a sub-optimal solution, a set of mutually incomparable solutions that approximates the Pareto front. Providing such an approximation is the general aim of the multi-objective evolutionary algorithms (MOEAs) [14], [25], [24], [23], [4], [12] and multi-objective local search algorithms [19], [2], [3], [18], [1], [6], [8], [9]. MOEAs are widely studied under different scenarios and have been applied to a large variety of multi-objective problems like scheduling, mapping, vehicle routing, etc. Stochastic local search algorithms, which are quite successful in solving a wide variety of $\mathcal{NP}$-hard single objective combinatorial optimization problems, have also been extended to multi-objective problems. *Pareto Local Search* [19], [20] (PLS) algorithm is one such kind of extension of an iterative improvement algorithms to multi-objective case.

The PLS algorithm employs a simple heuristic to find a reasonably good approximate Pareto front in a short amount of time. It maintains the set of all non-dominated solutions found so far in its archive and explores the neighborhood (search space) of each such solution. Upon finding a new non-dominated solution, the archive is updated. This process is repeated until no such improvement is possible in the neighborhood of the current archive. Since the size of archive is not limited, this method can be applied to a variety of problems where the cardinality of the optimal Pareto front is large. From our perspective PLS can be viewed as a greedy algorithm since it can remove some promising (unexplored) solutions from the archive if they became dominated by a newly discovered solution. Such exclusions can lead to premature convergence of the PLS algorithm to a local optimum. Moreover when PLS is used in conjunction with MOEA to speed up the latter, as in the case of the GSPLS algorithm [5], ignoring these points may reduce the genetic diversity and limit the effectiveness of combining the two methods.

To counter this problem, Inja et al. [13] proposed a new algorithm, QPLS which stores promising solutions in a queue. When such a solution is popped off the queue, QPLS performs strict Pareto improvement, better in all objectives, until no such improvement is possible. It is then compared to the Pareto archive. Unlike PLS algorithm, the queued approach doesn't remove dominated solutions until they are improved to locally optimal ones. Inja et al. [13] combined their algorithm with a genetic algorithm framework (GQPLS) where QPLS was restarted with a set of new solutions, consisting of mutations and recombinations of solutions obtained from the previous QPLS run. The major drawback of the QPLS algorithm is that it performs only strict Pareto improvement. The neighborhood of a solution may consist of many dominating solutions and in this case, QPLS selects a single dominating solution while discarding the rest.

We propose a *double archive Pareto local search* (DAPLS) algorithm which maintains two archive of solutions, the first of which consists of the non-dominated solutions found so far and is presented to the decision maker at the end of computation. The second archive maintains a set of unvisited solutions. At each step, a solution is removed from second archive and its neighbors are generated. Upon finding improvements to the first archive, the newly improved solutions are added to both archives. This may remove some unexplored solutions from the first archive as they are dominated by the newly improved solutions. But all such unvisited solutions are "protected" in the

second archive and therefore they are not prematurely deleted. As in previous PLS and QPLS work, we embed DAPLS in a genetic scheme. We empirically compare Genetic DAPLS with GQPLS and GSPLS using multi-objective quadratic assignment problems. We show that DAPLS can produce better approximate fronts in comparison to both, PLS and QPLS algorithms.

The rest of the paper is organized as follows. In Section II we present the essential concepts of multi-objective optimization. Section III describes some related work on approximating Pareto fronts using heuristic based approaches. Then in Section IV, we revisit in detail the Pareto local search and the queued Pareto local search algorithms. Our local search algorithm is presented in Section V, followed by an experimental evaluation on several instances of quadratic assignment problems in Section VI. Conclusions and suggestions for future work close the paper.

## II. Multi-Objective Optimization

A *multi-objective optimization problem* can be viewed as a tuple $\varphi = (\mathcal{S}, \mathcal{C}, f)$ where $\mathcal{S}$ is the solution space represented using fixed number of variables, $\mathcal{C} \subseteq \mathbb{R}^d$ is the cost space and $f : \mathcal{S} \to \mathcal{C}$ is a $d$-dimensional cost function. We assume that $\mathcal{S}$ is a discrete solution space and a solution $s \in \mathcal{S}$ can be transformed to some other solution $s' \in \mathcal{S}$ by making a change in single decision variable. The distance between two solutions $s$ and $s'$ is defined as the smallest number of changes needed to transform $s$ into $s'$. The neighborhood of a solution $s$, denoted by $\mathcal{N}(s)$, consists of the solutions whose distance from $s$ is 1.

The domination relations on cost space $\mathcal{C}$ are (non-strict and strict) standard partial order relations on $\mathbb{R}^d$. Without loss of generality, we assume that each objective needs to be minimized.

*Definition 1: Domination*: For two solutions $s, s' \in \mathcal{S}$ we say that

1) Solution $s$ weakly dominates $s'$, denotes as $f(s) \preceq f(s')$, if $\forall i \in \{1, \ldots, d\} : f_i(s) \leq f_i(s')$.
2) Solution $s$ strictly dominates $s'$, denoted as $f(s) \prec f(s')$, if $f(s) \preceq f(s')$ and $\exists i \in \{1, \ldots, d\} : f_i(s) < f_i(s')$.

Two distinct solution solutions $s$ and $s'$ are considered *incomparable* iff $f_i(s) < f_i(s')$ and $f_j(s) > f_j(s')$ for some $i, j \in \{1, \cdots, d\}$.

The set of minimal solutions of some $S \subseteq \mathcal{S}$ is defined as the set of non-dominated solution

$$\min(S) = \{s \in S : \forall s' \in S \text{ and } \ f(s') \not\prec f(s)\}.$$

Note that the solutions in $\min(S)$ are mutually incomparable.

*Definition 2: Pareto Front*: The Pareto front of a multi-objective optimization problem $\varphi = (\mathcal{S}, \mathcal{C}, f)$ is defined as

$$P_\varphi = \min(\mathcal{S})$$

The outcome of an multi-objective optimization algorithm which visits some $S \subseteq \mathcal{S}$ is

$$\hat{P}_\varphi = \min(S).$$

This set will be different from $P_\varphi$ as it may miss minimal points and include points which are minimal only relative to $S$, hence it can be viewed as an approximation of $P_\varphi$.

## III. Related work

Several state-of-the-art local search algorithms have been extended to multi-objective problems under the heading of Pareto based local search (PLS). Paquete et al. [19], initially introduced the idea of using Pareto based techniques in combination with local search that maintained an archive of non-dominated solutions. They used exploration strategies that select a solution from the current archive and visit its neighborhood. Angel et al.[2] uses Dynasearch and dynamic programming to explore the neighborhood of the bi-objective travelling salesman problem. Liefooghe et al. [16] experimentally compared various instances of PLS algorithms using different parameters and settings on bi-objective instances of travelling salesman and scheduling problems. They considered two types of archiving methods: unbounded archive, where all the non-dominated solutions during the search are stored and bounded archive which stores only a subset of non-dominated solutions. Lust et al. [18] first generate solutions using a single objective solver and then use PLS to generate more diverse solutions which were missed during the initial phase. Other works like Guided PLS [1], Iterated PLS [6], Anytime PLS [9] are also based on the PLS algorithm and uses it as a key components. Recently Inja et al. [13] proposed the idea of *queued Pareto local search* where the unvisited solutions are protected in a separate queue. At each step, QPLS pops an unvisited solution from the queue, and performs iterative Pareto improvement strategy. We revisit PLS and QPLS algorithm in next section.

Below we mention some relevant work on *multi-objective evolutionary algorithm* (EA). The common feature of our approach and MOEAs is the maintenance of a set (population) of solutions which are mutated and crossed over.

One of the earlier work of Schaffer [22] presents a multi-modal evolutionary algorithm that performs selection for each objective separately. It partitions the mating pool randomly into $d$ parts. After the selection procedure, the mating pools are shuffled and crossed over and mutations are performed. They use multiple fitness functions for the selection procedure. Hajela et al. [11] used the weighted sum scalarization method for fitness assignment. The scalar fitness values is then calculated by summing up the weighted objective values. The diversity of the weight combination is based on phenotypic fitness sharing.

Zitzler et al. [25] introduced Strength Pareto Evolutionary Algorithm (SPEA) that combines the several features of the above algorithms in a unified manner. It maintains the non dominated points externally in a second continuously updated population and the fitness of an individual depends on the number of external points it dominates. They preserve diversity using Pareto dominance. To keep the set of solutions small, they incorporated a clustering procedure. Another version of SPEA (SPEA-2) was proposed in [24], which includes a fine-grained fitness assignment strategy, a density estimation technique and an enhanced archive truncation method.

Another popular work in area of evolutionary algorithms is Non-dominated Sorting Genetic Algorithm (NSGA) [23]. Here

the pool of individuals is split into multiple fronts according to Pareto dominance. Individuals in first non dominated front are assigned highest rank, those in the second front are assigned a lower rank and so forth. The mating and environmental selection is made on the basis of this ranking. Their archive maintains both non dominated and dominated individuals and the worst $50\%$ of the pools of individual are truncated. Later, NSGA-2 [4] was proposed to enhance the efficiency of previous version. It uses a fast non-dominated sorting approach to alleviate complexity of previous version. Moreover, the mating pool is constructed by a selection operator applied to the best solutions.

Though all of these methods are known to give competitive results, neither of these algorithm employ local search algorithm for faster generation of better solutions. Therefore the running time of these algorithms are huge in comparison to evolutionary algorithm employing local search algorithms. Inja et al. [13] showed that GQPLS, which employs QPLS, outperforms both, SPEA-2 and NSGA-2. In this work, we show that our algorithm which also employs local search, achieves better solutions than GQPLS.

## IV. Revisiting Pareto based local search algorithms

In this section, we first revisit the Pareto local search (PLS) described in [19]. Algorithm 1 presents the pseudo code for the PLS algorithm. The input is an initial set $S_0$ of mutually incomparable solutions. All the solutions are marked as unvisited. The algorithm randomly selects an unvisited solution $s$ and explores its neighborhood $N(s)$. Line 10 attempts to insert $s' \in N(s)$ to the archive $P$. The operation $min$ is realized using a Pareto filter, a procedure that takes as input a set $K$ and a solution $c$ and returns a set consisting of the non-dominated points in $K \cup \{c\}$. The algorithm stops when all the solutions in the set $P$ are visited. Paquete et al. [20] showed that PLS stops in a Pareto local optimum set.

**Algorithm 1** Pareto Local Search (PLS) algorithm

```
1:  Input: An initial set of mutually incomparable solutions S_0
2:  P := S_0
3:  for each s ∈ P do
4:      visited(s) := False
5:  end for
6:  repeat
7:      s := select randomly a solution from P.
8:      for each s' ∈ N(s) do
9:          visited(s') := false
10:         P := min(P ∪ s')
11:     end for
12:     visited(s) := true
13: until ∀s' ∈ P : visited(s') = True
```

PLS has major advantages over other prior works. First, it maintains only non-dominated solution, hence provides fast convergence to a *Pareto local optimum set*. Moreover, since it maintains an unlimited archive, it can provide a large number of incomparable solutions. However, one of its major drawbacks is that good candidate solutions are removed from the archive if dominated by another solution. This premature removal may limit the diversity of exploration [13].

Inja et al. [13] introduced a queue based Pareto local search (QPLS) which prevents the premature deletion of promising candidate solutions by maintaining a queue of solutions, which leads to more diverse Pareto archive. The pseudo code of QPLS is presented in Algorithm 2.

**Algorithm 2** Queued Pareto Local Search (QPLS) algorithm

```
1:  Input: An initial queue Q
2:  P := ∅
3:  while Q ≠ ∅ do
4:      s := pop an element from Q
5:      s := PI(s, f)
6:      if ∃p ∈ P : f(s) ⊀ f(p) ∧ f(s) ≠ f(p) then
7:          P = min(P ∪ {s})
8:          N := {s' ∈ N(s) : f(s) ⊀ f(s')}
9:          Q.addK(N, k)
10:     end if
11: end while
12: return P
```

QPLS starts with an initial queue $Q$ of solutions and an empty Pareto archive. A candidate solution $s$ is popped from the queue and a recursive Pareto improvement function (PI) is applied. It improves the solutions by repeatedly selecting a dominating solution from the neighborhood, until no such improvements is possible. After a solution $s$ is found that is not weakly dominated by any of its neighbors, it is compared to the Pareto archive. If no solution in the current archive dominates $s$, then solution $s$ is added to $P$ using the $\min$ operator. Then a set of $k$ new incomparable candidate solutions are randomly selected from the neighborhood of $s$ and are added to the queue. This entire procedure is repeated until $Q$ is empty.

QPLS has a major advantage over PLS as the solutions which have not been optimized are stored in a queue. This prevents the premature deletions of solutions. However, a drawback of the QPLS algorithm is that it applies recursive Pareto improvement strategy. PI improves upon a solution by repeatedly selecting a single dominating solution from the neighborhood. Note that a neighborhood of a solution may consist of a set of dominating solutions which are incomparable. In this case, QPLS selects only one such solution from the neighborhood while discarding the rest.

## V. Double Archive Pareto Local Search Algorithm

In this section, we present our Double archive Pareto local search algorithm that maintains an additional archive $L$ which is maintained as a queue. Algorithm 3 depicts the general scheme of DAPLS.

Both archives $P$ and $L$ are initialized to a set $S_0$ of mutually incomparable solutions. While $L$ is not empty, a solution $s$ is selected and its entire (or partial) neighborhood $N(s)$ is generated. The current Pareto archive $P$ is updated with solutions from $N(s)$ using a $\min$ operator. If a solution $s' \in N(s)$ is present in the updated Pareto archive, it is added to archive $L$ in line 10 and remains there even if it

**Algorithm 3** DAPLS($S_0, f$)

1: **Input:** An initial set of incomparable solutions $S_0$
2: $P := S_0$
3: $L := S_0$
4: **while** $L \neq \emptyset$ **do**
5: $s :=$ select a solution from $L$
6: $L := L \setminus \{s\}$
7: $P := \min(P \cup N(s))$
8: **for each** $s' \in N(s)$ **do**
9: **if** $s' \in P$ **then**
10: $L := L \cup \{s'\}$
11: **end if**
12: **end for**
13: **end while**
14: **return** $P$

is later removed from $P$. This procedure is repeated for all the solutions in neighborhood of $s$ (lines 8-11).

Note that as in previous work of SPLS [5], one can distinguish between two neighborhood generation strategies: best-improvement and first improvement implementation. Note that in Algorithm 3, DAPLS applies improvement once per iteration and saves all dominating points from $P$ in $L$. Moreover, we maintain archive $L$ as a queue. Hence, a solution $s$ which is added to $L$ earlier (in sense of number of iteration) than some other solution $s'$, DAPLS explores the neighbors of $s$ before the neighbors of $s'$. This provides *fair* chance to the solutions for exploration and prevents premature convergence. Therefore, DAPLS can also be seen as *breadth first* exploration of search space using Pareto dominance criteria.

DAPLS admits some natural properties mentioned below.

*Property 1:* DAPLS is an iterative improvement algorithm with respect to its neighbours and strict non-dominance relation.

*Property 2:* Let $P_i$ and $L_i$ denote the archives $P$ and $L$, respectively at end of iteration $i$, then $\forall s \in L_i, \exists i' \leq i : s \in P_{i'}$.

*Property 3:* DAPLS terminates with a Pareto local optimum set.

Property 1 holds since $P$ is updated using the $\min$ operator. Thus at all times, $P$ consists of incomparable solutions. The essence of our approach lies in Property 2. Essentially all the solutions which are inserted to $P$ at iteration $i$ are also inserted to archive $L$ (line 10). At a later iteration $i'$, it may happen that a new solution is removed prematurely from $P$ in which case, $L$ protects the unvisited solution. Recall that a solution from $L$ can only be removed after its neighborhood has been explored. Unlike QPLS, DAPLS prevents all the dominating incomparable solutions from the neighborhood by inserting them in $L$.

Next, we present a simple Genetic DAPLS, depicted in Algorithm 4. Our Genetic DAPLS escapes local optima by mutating and recombining the entire Pareto Archive. It starts with executing DAPLS on an initial set of solutions. Once a locally optimal Pareto archive P is obtained. It mutates (with probability $\alpha$) or recombines (with probability $1 - \alpha$) all the solutions in $P$. Essentially, the skeleton of Genetic DAPLS is

**Algorithm 4** Genetic DAPLS($S_0, \alpha, f$)

1: **Input:** An initial set of incomparable solutions $S_0$
2: $P :=$ DAPLS $(S_0, f)$
3: **while** NOT Termination **do**
4: $S := \emptyset$
5: **for each** $s \in P$ **do**
6: **if** $\alpha >$ rand$(0, 1)$ or $|P| < 2$ **then**
7: $s' :=$ mutate($s$)
8: **else**
9: Select $s'' \neq s$ from $P$
10: $s' :=$ Recombine(s", s)
11: **end if**
12: $S := \min(S \cup \{s'\})$
13: **end for**
14: $P := \min(P \cup$ DAPLS$(S, f))$
15: **end while**
16: **return** $P$

similar to Genetic QPLS where QPLS algorithm is replaced with DAPLS algorithm and update to $S$ is realized using non-dominance relation. This helps us to explore only a small number of candidate solutions. Note that Genetic DAPLS runs until some stopping criterion is met.

## VI. Experimental Results

We compare DAPLS to QPLS and PLS on the *multi-objective quadratic assignment problem* (MQAP) [15].

Single objective QAPs are NP-hard combinatorial optimization problems that model many real-world situations like the layout out of electrical circuits in computer aided design, scheduling, vehicle routing, etc. In fact, travelling salesman problem which is one of the most challenging problem in combinatorial optimization, is a special case of QAP. Intuitively, QAPs can be described as the assignment of $n$ facilities to $n$ locations where the distance between each pair of locations is given and for each pair of facilities, the amount of flow (or materials) transported is specified. The aim is to find an optimal assignment of facilities to locations that minimizes the sum of products between distance and flows.

In this work, we consider the multi-objective version of QAPs (MQAP) introduced by Knowles et al. [15], where the flows between each pair of facilities are multi-dimensional values. The values in flow matrices are correlated with factor $\rho$. If $\rho$ is strongly positive then the Pareto front is small and is closer to being convex. This makes the problem harder. On the other hand if the value of $\rho$ is small or negative, then there exists large number of Pareto optimal solution which are evenly spread out in the cost space.

Specifically, we are given $n$ facilities and $n$ locations such that $d_{pq}$ denotes the distance between location $p$ and location $q$. Moreover, we are provided with $d$ flow matrices $F^1, \cdots, F^d$, where $F^i_{jk}$ denotes the flow from facility $j$ to facility $k$ in the $i$-th dimension. The aim is to minimize:

$$C^i(\pi) = \sum_{a=1}^{n} \sum_{b=1}^{n} F^i_{ab}.d_{\pi(a),\pi(b)}, \quad \forall i \in \{1, \cdots, d\}$$

where $\pi(.)$ is a permutation from set of all permutations $\Pi(n)$ of $\{1, 2, \cdots, n\}$. Given a permutation $\pi$, it takes $O(n^2)$ to compute the above cost functions.

Below we present a description of the neighborhood relation for QAPs. Furthermore, we also define a mutation and a recombination operator applied in the genetic version of local search algorithms. Lastly, we present a simple time-based stopping criteria so as to have fair comparison of the performance of the algorithms.

*Neighborhood Relation*: MQAPs are permutation problems where a suitable neighborhood operator is $q$-exchange operator that swaps the locations of $q$-facilities. In this work, we the use a 2-exchange operator that swaps the location of two difference facilities. It has two major advantages: the size of neighborhood $\binom{n}{2}$, is relatively small and the time complexity of computing the incremental change in cost is linear [21].

*Mutation Operator*: We present mutation $q$-exchange mutation operator described in [7]. The $q$-exchange mutation randomly selects $q > 2$ locations $\{l_1, \cdots, l_q\}$, without replacement from a solution. A new solution is generated by exchanging these location from left to right or from right to left with equal probability. For example, when exchanges are made right to left, facility at $l_i$ is shifted to location $l_{i-1}$ where $i > 2$ and facility at location $l_1$ is shifted to location $l_q$. Note that a new solution is $q$-swaps apart from the original solution. Since our neighborhood operator is 2 exchange operator, we use $q > 2$ exchange mutation to escape from the local optima.

*Recombination Operator*: Drugan et al. [7] also introduced the idea of *path-guided mutation* for QAP problems where two solutions $s$ and $s'$ are selected from the current Pareto archive such that the distance is at least $q$. An offspring $s''$ is generated by copying the solution $s$. The set of common cycles for two solutions, $s''$ and $s'$ are identified. A cycle is a minimal subset of locations such that the set of their facilities is the same in both parent solutions. Then a cycle $c$ is randomly chosen. For $q-1$ times, choose at random a location $i$ in the cycle $c$ from solution $s''$, where $s''[i] = s'[j]$ and $i \neq j$. Exchange the facilities of $s''[i]$ and $s'[j]$. Thus, the distance between $s''$ and its first parent $s$, is increased by 1 and the distance between $s''$ and the second parent $s'$, is decreased by 1. If the size of $c$ is smaller or equal to $q$, a second cycle is chosen. This process of randomly selecting a cycle and swap locations is repeated until the distance between $s''$ and $s$ is $q$. If there are no parent solutions at distance larger or equal with $q$, we generate a solution with the mutation operator.

*Stopping Criteria*: In the majority of the MOEA literature, the algorithms are compared using the number of fitness function evaluation instead of time. The neighborhood generating operators do not perform full fitness evaluations. Instead they perturb many small changes to the solutions, which can be evaluated in a fractional amount of time. Therefore, we measure the outcome of different algorithm as a function of time. We run each algorithm for the same fixed amount of time for each instance.

*Methodology*

We generated multiple QAPs problem instances with different correlation factors and facilities. For bi-objective QAP problem, we choose a large number of facilities, $n = \{50, 75\}$ with correlation factors $\rho = \{-0.25, -0.75, 0.25, 0.75\}$. Since the number of non-dominated solutions in the Pareto front increases exponentially with dimension, we restricted our choice to rather small number of facilities $n = \{20, 25\}$ with the combination of correlation factor $\rho = \{0.25.0.75\}$ for tri-objective QAP. To have a fair comparison, we kept the runtime for each instance across all the algorithms a constant. For example, for a small bi-objective instance with 50 facilities and $\rho = 0.25$, all algorithms were executed for 20 minutes. Due to stochasticity of the algorithms, each experiment was executed 20 times for each instance. The comparison of performance is carried out using a hypervolume unary indicator [25]. This indicator measures the volume of the cost space that is weakly dominated by an approximating set.

| Bi-qap | Volume | | |
|---|---|---|---|
| $n, \rho$ | GSPLS | GQPLS | DAPLS |
| $50, -0.75$ | $0.92 \pm 0.008$ | $0.93 \pm 0.002$ | **0.94 ± 0.001** |
| $50, -0.25$ | $0.94 \pm 0.010$ | $0.97 \pm 0.011$ | **0.99 ± 0.008** |
| $50, +0.25$ | $0.93 \pm 0.015$ | $0.95 \pm 0.002$ | **0.98 ± 0.005** |
| $50, +0.75$ | $0.84 \pm 0.012$ | $0.82 \pm 0.015$ | **0.88 ± 0.006** |
| $75, -0.75$ | $0.80 \pm 0.007$ | $0.83 \pm 0.010$ | **0.86 ± 0.002** |
| $75, -0.25$ | $0.75 \pm 0.001$ | $0.79 \pm 0.007$ | **0.81 ± 0.009** |
| $75, +0.25$ | $0.81 \pm 0.006$ | $0.83 \pm 0.001$ | **0.86 ± 0.001** |
| $75, +0.75$ | $0.77 \pm 0.001$ | $0.77 \pm 0.013$ | **0.83 ± 0.014** |

TABLE I: Performance of GSPLS, GQPLS and Genetic DAPLS on 8 large bi-objective instances of QAP in terms of normalized hypervolume

| Tri-qap | Volume | | |
|---|---|---|---|
| $n, \rho_1, \rho_2$ | GSPLS | GQPLS | DAPLS |
| $20, 0.25, 0.25$ | $0.81 \pm 0.027$ | $0.84 \pm 0.003$ | **0.86 ± 0.001** |
| $20, 0.25, 0.75$ | $0.79 \pm 0.001$ | $0.81 \pm 0.001$ | **0.83 ± 0.003** |
| $20, 0.75, 0.25$ | $0.78 \pm 0.013$ | $0.80 \pm 0.001$ | **0.82 ± 0.002** |
| $20, 0.75, 0.75$ | $0.73 \pm 0.011$ | $0.75 \pm 0.002$ | **0.82 ± 0.001** |
| $25, 0.25, 0.25$ | $0.90 \pm 0.019$ | $0.92 \pm 0.003$ | **0.96 ± 0.002** |
| $25, 0.25, 0.75$ | $0.82 \pm 0.012$ | $0.84 \pm 0.006$ | **0.87 ± 0.001** |
| $25, 0.75, 0.25$ | $0.80 \pm 0.014$ | $0.82 \pm 0.008$ | **0.84 ± 0.001** |
| $25, 0.75, 0.75$ | $0.78 \pm 0.012$ | $0.82 \pm 0.023$ | **0.87 ± 0.001** |

TABLE II: Performance of GSPLS, GQPLS and Genetic DAPLS on 8 tri-objective instances of QAP in terms of normalized hypervolume

Table I shows the average performance, in terms of normalized hypervolume [1] for bi-objective QAP instances. For all instances, genetic DAPLS outperforms both algorithms. Moreover, we observe that for small and negative correlation factors the difference between genetic DAPLS and other methods are almost negligible. This can be explained by the fact that for such instances the solutions are evenly spread out in the cost space and all algorithms can find them easily. On the other hand, for strongly positive correlation factors, the number of solutions are small and rather restricted to some part of the cost space. This in turn increases the complexity of finding

[1]We used hypervolume generating tool from *http://lopez-ibanez.eu/hypervolume* for comparison.

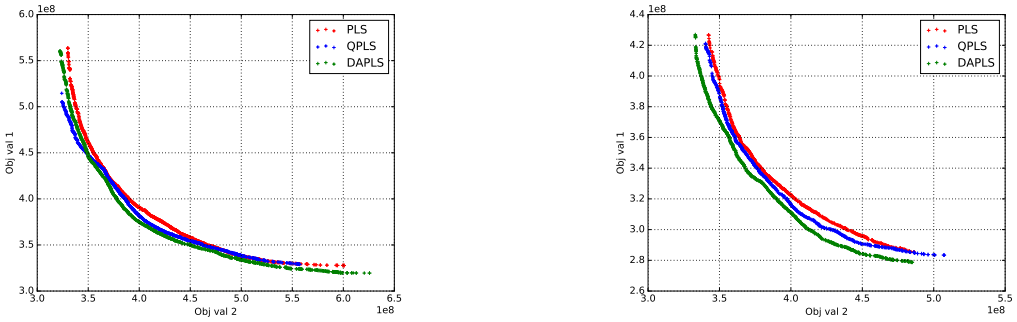


Fig. 1: Median attainment surfaces for $n = 50$ with $\rho = 0.25$ (on left) and $\rho = 0.75$ (on right)

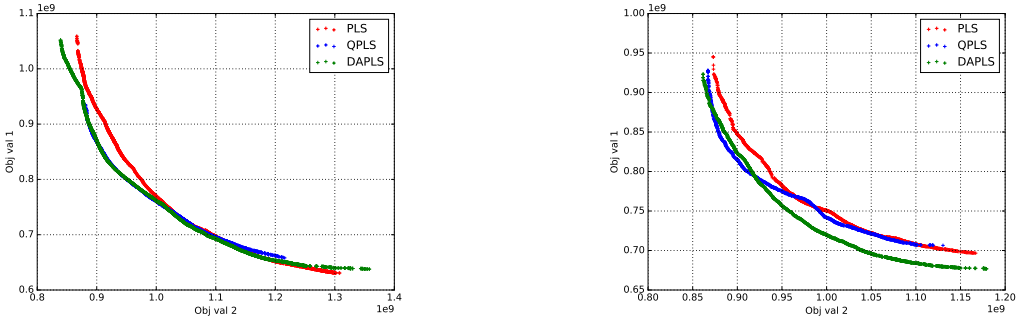


Fig. 2: Median attainment surfaces for $n = 75$ with $\rho = 0.25$ (on left) and $\rho = 0.75$ (on right)

solutions approximating the Pareto front for such instances. Our method improves upon the previous algorithms and finds a much better approximate Pareto front for these hard instances.

Similarly, Table II shows the average performance for tri-objective QAP instances. Here, $\rho_1$ represents the correlation factor between objective 1 and objective 2 while $\rho_2$ presents the correlation between objective 1 and objective 3. Like bi-objective case, genetic DAPLS outperforms all other methods in terms of hypervolume. As in bi-objective case, we observe that for large correlation factors ($\rho_1 = 0.75$ and $\rho_2 = 0.75$), DAPLS finds better approximate solutions.

Fonseca et al. [10] proposed the idea of empirical first-order attainment function (EAF). This measure estimates the probability that a random point in cost space is attained in one optimization run of the considered algorithm independently of attaining any other point in cost space. In this work, we also use visualization of EAFs from outcomes of multiple runs of an algorithm, for comparison.. An approximating set is $k\%$ approximation set if it weakly dominates exactly those solutions that have been attained in at least $k$ percent of runs. We show $50\%$-approximation set of EAFs for biobjective instances with positive correlation. These plots are generated using the $\mathcal{R}$-statistical tool with the library *EAF*. The details of the generating algorithm can be found in [17].

Figure 1 shows the median attainment surfaces for bi-objective instance of 50-facilities with correlation factors $0.25$ and $0.75$. Similarly, Figure 2 shows the median attainment surface for 75 facilities for $\rho$ equal to $0.25$ and $0.75$. For instances with correlation factor $0.25$, it is clearly visible that genetic DAPLS achieves better spread of solutions (diversity) in the cost space than GSPLS and GQPLS. Similar improvements are also observed for the instances with correlation factor $0.75$, where genetic DAPLS not only achieves better diversification but also provides solutions which are closer to the Pareto front.

## VII. Conclusion and Future work

We developed a new local search algorithm for approximating Pareto fronts. Our algorithm is based on the principle that the neighbors of solutions that were non-dominated at some stage of the search process should be explored before they are discarded. Like PLS and QPLS, we embedded DAPLS in a genetic framework. We showed empirically that genetic DAPLS outperforms GSPLS and GQPLS algorithms on several instances of bi-objective and tri-objective quadratic assignment problems.

In the future, we would like to study DAPLS for higher dimensional cost spaces where the size of Pareto front is typically huge. This in turn causes an increase in the runtime of the inner loop of genetic DAPLS. Our next aim would be to use some heuristics to limit the size of neighborhood. We would also like to see how DAPLS performs on other multi-objective problems like knapsack, coordination graphs, etc.

## References

[1] A. Alsheddy and E. Tsang. Guided Pareto local search and its application to the 0/1 multi-objective knapsack problems. *Intl Conf. on Metaheuristics*, 2009.

[2] E. Angel, E. Bampis, and L. Gourvés. A Dynasearch neighborhood for the bicriteria traveling salesman problem. *Metaheuristics for Multiobjective Optimization*, pages 153–176, 2004.

[3] M. Basseur. Design of cooperative algorithms for multi-objective optimization: application to the flow-shop scheduling problem. *4OR*, pages 255–258, 2006.

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

[5] D. Drugan, M.M.and Thierens. Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies. *Journal of Heuristics*, 18(5):727–766, 2012.

[6] M.M. Drugan and D. Thierens. Path-guided mutation for stochastic Pareto local search algorithms. *Parallel Problem Solving from Nature*, pages 485–495, 2010.

[7] M.M. Drugan and D. Thierens. Path-guided mutation for stochastic pareto local search algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 485–495, 2010.

[8] J. Dubois-Lacoste, M. López-Ibanéz, and T. Stützle. Improving the anytime behavior of two-phase local search. *Annals of Mathematics and AI*, pages 125–154, 2011.

[9] J. Dubois-Lacoste, M. López-Ibanéz, and T. Stützle. Anytime pareto local search. *Euro. Journal of Operational Research*, pages 369 – 385, 2015.

[10] V. Fonseca, C.M. Fonseca, and A.O. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. In *International Conference on Evolutionary Multi-Criterion Optimization*, 2001.

[11] P. Hajela and C.Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, pages 99–107, 1992.

[12] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. *IEEE Conference on Evolutionary Computation*, pages 82–87, 1994.

[13] M. Inja, C. Kooijman, M. de Waard, D.M. Roijers, and S. Whiteson. Queued pareto local search for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 589–599, 2014.

[14] A. Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation*, pages 402–412, 2002.

[15] J. Knowles and D. Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. *Evolutionary Multi-Criterion Optimization*, pages 295–310, 2003.

[16] A. Liefooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E. Talbi. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, pages 317–352, 2012.

[17] M. López-Ibáñez, L. Paquete, and T. Stützle. Experimental methods for the analysis of optimization algorithms. *Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization*, pages 209–222, 2010.

[18] T. Lust and J. Teghem. Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, pages 475–510, 2010.

[19] L. Paquete, M. Chiarandini, and T. Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. *Metaheuristics for Multiobjective Optimisation*, 2004.

[20] L. Paquete, T. Schiavinotto, and T. Stützle. On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, pages 83–97, 2007.

[21] L. Paquete and T. Stützle. A study of stochastic local search algorithms for the biobjective {QAP} with correlated flow matrices. *European Journal of Operational Research*, pages 943–959, 2006.

[22] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. *International Conference on Genetic Algorithms*, pages 93–100, 1985.

[23] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Journal of Evolutionary Computation*, pages 221–248, 1994.

[24] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2001.

[25] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, pages 257–271, 1999.