Sets of interacting scalarization functions in local search for multi-objective combinatorial optimization problems

Madalina M. Drugan
Artificial Intelligence Lab,
Vrije Universiteit Brussel,
Pleinlaan 2, B-1050 Brussels, Belgium
Email: mdrugan@vub.ac.be

***Abstract*—Searching in multi-objective search spaces is considered a challenging problem. *Pareto local search* (PLS) searches directly into the multi-objective search space maintaining an archive of best non-dominated solutions found so far, the *non-dominated archive*. PLS' advantage is the exploitation of relationships between solutions in the non-dominated archive at the cost of high maintenance costs of the archive. The *scalarized local search* (SLS) uses *scalarization functions* to transform the multi-objective search space into a single objective search space. SLS is faster because it is searching in a single objective search space but the independent scalarization functions do not systematic exploit the structure of the multi-objective search space.**

We improve the performance of SLS algorithms by allowing interactions between scalarization functions. The *adaptive scalarization functions* select frequently the scalarization function that generates well performing SLS. The *genetic scalarization functions* assume that the scalarization functions have commonalities that can be exploited using genetic like operators. We experimentally show that the proposed techniques can improve the performance of local search algorithms on correlated bi-objective QAP instances.

I. Introduction

Many real-world applications are or can be modelled as multi-objective combinatorial optimization problems: scheduling, investment planning, vehicle routing [6]. *Multi-objective search spaces* have two or more objectives to optimize simultaneously.

There are two main approaches for solving multi-objective problems: *scalarization* and *Pareto search*. *Scalarization* functions [7] transform the multi-objective problem into a single objective function using some weighing scheme to combine the different values of multiple objectives. The choice of the scalarization functions is very important for efficiency of the search [5], [9]. Because the result of this optimization is a single solution, the optimization process needs to be repeated several times with different scalarization functions in order to generate the set of best solutions. Some examples of scalarization functions are presented in Section II.

Pareto search [15] is performed directly in the multi-objective space using the dedicated dominance measures to compare the quality of solutions. This method does not scale well for search spaces with many dimensions. When there are many incomparable solutions in the search space, such an algorithm has problems with managing incomparable solutions.

Local search [8] is a successful and powerful method for optimizing combinatorial optimization problems. Local search applies an exploration strategy to iteratively move from a current solution to a neighbour that improves upon the current solution. A local search stops in a local optimum. The simplest way to restart a local search is using a randomly generated solution. Stochastic LS [8] considers the structure of the search space in order to speed up the search.

Pareto local search (PLS) [13], [11] is local search using Pareto search. A PLS stops in a *Pareto local optimum set* that is a set of incomparable solutions that does not have improving solutions in their neighbourhood. *Stochastic PLS* [4] restarts PLS from solutions generated using genetic operators.

Main contributions. We introduce sets of scalarization functions that interact in order to explore correlations between multiple objectives of a search space. An *adaptive strategy* pursuits the reduced space that generates the Pareto local search with the best performance. We assume that the performance of scalarization functions can vary a lot in different regions of the multi-objective search space. Thus, some scalarization functions can be more useful in the beginning of the search whereas other scalarization functions can be performant in the end of the search. We consider that this technique is an *intensification technique* in searching for the right search direction [12], where no new scalarization functions are proposed. Instead, a scalarization function that best explore a specific region of the search space is calculated.

A *genetic strategy* generates weights for scalarization functions using real-coded genetic operators. The best performing scalarization functions are selected to create offspring. We reward the most the scalarization functions which with small changes in weights improves the Pareto local optimum set. We consider that this technique diversifies the search directions in order to fill up the gaps in the Pareto set.

Although the proposed techniques are applicable for two or more objective search spaces, they are experimentally tested only on two objectives.

Experimental settings. We compare Pareto and scalarized local search algorithms on various instances of the bi-objective *quadratic assignment problems* QAP instances. QAPs are NP-hard problems modelling many real-world applications that require to optimally allocate n facilities to n locations using

relation	objective vectors	
dominance	$\mathbf{f}(a) \prec \mathbf{f}(b)$	$\exists i, f_i(a) < f_i(b)$ and $\forall j \neq i, f_j(a) \leq f_j(b)$
weakly dom	$\mathbf{f}(a) \preceq \mathbf{f}(b)$	$\forall i, f_i(a) \leq f_i(b)$
incomparable	$\mathbf{f}(a) \\| \mathbf{f}(b)$	$\mathbf{f}(a) \npreceq \mathbf{f}(b)$ and $\mathbf{f}(b) \npreceq \mathbf{f}(a)$
non-domin by	$\mathbf{f}(a) \nsucc \mathbf{f}(b)$	$\mathbf{f}(a) \preceq \mathbf{f}(b)$ or $\mathbf{f}(a) \\| \mathbf{f}(b)$

relation	sets of objective vectors	
dominance	$A \prec B$	$\forall \mathbf{f}(b) \in B, \exists \mathbf{f}(a) \in A$ such that $\mathbf{f}(a) \prec \mathbf{f}(b)$
weakly dominates	$A \preceq B$	$\forall \mathbf{f}(b) \in B, \exists \mathbf{f}(a) \in A$ such that $\mathbf{f}(a) \preceq \mathbf{f}(b)$
incomparable	$A \\| B$	$A \npreceq B$ and $B \npreceq A$
non-domin by	$A \nsucc B$	$A \preceq B$ or $A \\| B$

TABLE I. RELATIONS BETWEEN OBJECTIVE VECTORS AND SETS OF OBJECTIVE VECTORS CONSIDERED IN THIS PAPER.

a distance and a flow matrix. *Multi-objective QAPs* [10] are an extension of QAPs with several flow matrices where the elements of the flow matrices are correlated. This problem is a minimization problem.

In our experiments, when the bi-objective QAPs have high correlation between flow matrices, multi-restart and stochastic PLS algorithms significantly outperform the corresponding SLS algorithms [5], [9]. We show that the exploration of the structure of the search space can be realized with the proposed adaptive scalarization techniques. The best performing algorithm is the stochastic genetic SLS algorithms that have similar performance like stochastic PLS.

Outline of the paper. Section II presents some background information. Section III introduces the adaptive scalarization functions. Section IV introduces genetic operators to evolve the weights of the scalarization functions. Section V compares the performance of Pareto and scalarized based local search algorithms. Section VI concludes the paper.

II. BACKGROUND

Let us consider the multi-objective function $\mathbf{f} = (f_1, \ldots, f_m)$, where $\mathbf{f} : \mathcal{S} \to E^m$ and E is a countable set. $\mathcal{S}$ is the set of solutions in a countable *solution space*. The image of the solution set $\mathcal{S}$ under $\mathbf{f}$ is denoted with the *objective space* $E^m = \mathbf{f}(\mathcal{S}) = \{y \mid \exists x \in \mathcal{S} : y = \mathbf{f}(x)\}$.

To define an optimization problem, we first consider a binary order relationship, $\preceq$, between any two elements in the objective space $\mathcal{O}$. Without loss of generality, only minimization problems are considered. In Table I, we present the relationships between objective vectors used in this paper [15].

The set of *Pareto optimal solutions* contains the solutions that are non-dominated by all the solutions in the multi-objective search space.

A. Stochastic Pareto local search

Pareto local search algorithms [13] search directly in the multi-objective search space. For ease of discussion, we consider a non-dominated archive A of unlimited size that can store *all* the solutions in a Pareto local optimum set. The pseudo-code for this algorithm is given in Algorithm 1.

The algorithm starts from an empty non-dominated archive A. Each solution s has a flag, $s.visited$, set on *false* at the initialization of s and set on *true* after its neighbourhood, $\mathcal{N}(s)$, is visited. $\mathcal{N}(s)$ depends on the problem it is applied on. For example, for single and multi-objective QAPs a popular neighbourhood function is the 2-exchange operator that exchanges any two positions in the permutation π associated with the solution s.

Algorithm 1: Stochastic Pareto local search (*SPLS*)

```
$A \leftarrow \{s\}$, where $s$ randomly generated
while a stopping criteria is met do do
  Generate a solution $s$ from $A$ to restart PLS
  $A' \leftarrow \{s\}$
  while $\exists s' \in A'$, with $s'.visited =$ false do
    for all $s'' \in \mathcal{N}(s')$ do
      if $\mathbf{f}(s'') \nsucc \mathbf{f}(s')$ then
        $s''.visited \leftarrow false$
        add $s''$ to $A'$
        remove the dominated solutions from $A'$
      end if
    end for
    $s'.visited \leftarrow$ true
  end while
  $A \leftarrow A \cup A'$
  remove the dominated solution from $A$
end while
return $A$
```

Pareto local search starts from a given solution s chosen uniform randomly from A. An archive A' storing the non-dominated solutions from a PLS run is initialized to s. Each iteration, a solution s', with $s'.visitied = false$, is uniform randomly chosen from A'. All solutions from $\mathcal{N}(s')$ are explored. Thus, we use the best neighbourhood exploitation strategy. A solution $s'' \in \mathcal{N}(s')$ is selected to be included in A' if it is non-dominated by the solution s'. Then, the solutions that are dominated by s'', if any, are removed from A'. The flag of the solution s' is set to $true$. While there are unexplored solutions in A', this process is repeated. Otherwise, A' is a Pareto local optimum set which is merged with the current archive A. The solutions that are dominated in the newly archive A are eliminated.

Restarting strategies. We can restart a PLS by generating new solutions from the solutions in the current non-dominated archive A using genetic operators. *Path-guided mutation* [1] selects at random two solutions from A and generates a solution on the path between the two solutions at a given distance. *Mutation* selects at random a solution from A and it generates a solution which is different from the current solution in few positions. In our experiments we select both operators with equal probability.

B. Scalarized functions

Scalarization functions transform a multi-objective problem into a single objective objective using some function [7]. Since, we deal with minimization problems, the scalarization functions are also minimized. In this study, we consider two popular scalarization functions. These scalarization functions have a set of weights $w_i \in [0, 1]$, $i = 1, \ldots, m$, and $\sum_{i=1}^{m} w_i = 1$.

The *linear scalarization* assigns to each objective a weight. It is equal with the sum over all the objectives of products

Algorithm 2: Stochastic adaptive SLS (*ASLS*)

```
Require: $\mathbf{g} = (g_1, \ldots, g_\ell)$
  $t \leftarrow 0$; $\mathcal{P}_i^{(t)} \leftarrow 1/\ell$; $\mathcal{Q}_i^{(t)} \leftarrow 0.5$, $\forall i$
  while a stopping criteria met do
    Select $g_i$ from $\mathbf{g}$ proportional with $\mathcal{P}_i^{(t)}$
    Generate a solution $s$ from $A$ to restart LS
    $A' \leftarrow \mathsf{SLS}(s, g_i)$
    Update the reward distribution $\mathcal{R}^{(t)}$
    $\mathcal{P}_i^{(t)} \leftarrow \mathsf{AP}(\mathbf{g})$
    $A \leftarrow A' \cup A$
    remove dominated solution from $A$
  end while
  return $A$
```

Algorithm 3: Scalarized local search (SLS)

```
Require: $g_i$, $s$ with $s.visited =$ false
  $A' \leftarrow \{s\}$
  while $\exists s' \in A'$, with $s'.visited =$ false do
    for all $s'' \in \mathcal{N}(s')$ do
      if $g_i(s'') \leq g_i(s')$ then
        $s''.visited \leftarrow$ false
        add $s''$ to $A'$
        remove dominated solutions from $A'$
      end if
    end for

  end while
  return $A'$
```

between weights and objectives. For a solution s we have, $g_L(s) = w_1 \cdot f_1(s) + \ldots + w_m \cdot f_m(s)$.

The *Tchebycheff scalarization*, beside weights, has an m-dimensional reference point, $\mathbf{z} = (z_1, \ldots, z_m)$ which, for minimization problems, should be, in all objectives, smaller than $\mathbf{f}(s)$. For a solution s, we have $g_C(s) = \max_j \{w_j \cdot (f_j(s) - z^j)\}$. where $\mathbf{z}$ is a ideal point that dominates all the other solutions. We can easily consider $\mathbf{z}$ as the minimum value in each objective, j, of solutions from A minus a small variable ϵ, $z^j = \min_{s \in A} f_j(s) - \epsilon$.

The linear scalarization functions are very intuitive, but they can only find solutions in a convex Pareto optimal set. The Tchebycheff functions can find solutions in a non-convex Pareto optimal set. In the next sections we introduce two local search algorithms that use scalarization functions.

III. Adaptive scalarized local search (ASLS)

In this section, we design an algorithm which frequently selects scalarization functions that most improve the performance of the algorithm. *Pursuit allocation strategies*(AP) [14] are on-line operator selection algorithms that adapt a selection probability distribution such that the operator with the maximal estimated reward is pursued. In [1], the pursuit algorithm is used to select operators in stochastic Pareto local search algorithms.

Here, we use the pursuit algorithm to select more often the scalarization functions which lead to local search algorithms that perform well. Note that, in time, the preference for a scalarization function can change with the increase/decrease in its performance in a particular region of the search space.

We call the algorithm *adaptive scalarization local search* (ASLS). The pseudo-code for the *stochastic adaptive SLS* (SASLS) is given in Algorithm 2. The pseudo-code for the scalarized local search is given in Algorithm 3. Algorithm 4 shows the pseudo-code for the adaptive pursuit algorithm.

A. The algorithm

In Algorithm 2, consider a fixed set of scalarization functions, $(g_1, \ldots, g_\ell)$. Consider a probability vector $\mathcal{P}^{(t)} = \{\mathcal{P}_1^{(t)}, \ldots, \mathcal{P}_\ell^{(t)}\}$, where $\mathcal{P}_i^{(t)}$ is the probability to choose the i-th scalarization function, g_i, and $\forall t : 0 \leq \mathcal{P}_i^{(t)} \leq 1; \sum_{i=1}^{\ell} \mathcal{P}_i^{(t)} = 1$. The selection target distribution $\mathcal{P}^{(t)}$ does not change in time and it is a step like function with one large value p_M and the rest small values p_m, where $p_M + \sum_{j=1}^{\ell-1} p_m = 1$. The quality vector (or estimated reward) $\mathcal{Q}^{(t)} = \{\mathcal{Q}_1^{(t)}, \ldots, \mathcal{Q}_\ell^{(t)}\}$ indicates the performance of a local search using the i-th scalarization function, g_i. A reward $\mathcal{R}_i^{(t)}$ is updated after applying the scalarized function g_i.

For the initialization step, to ensure that all the scalarization functions are tried at least once, we set, $\forall i$, $\mathcal{P}_i^{(0)} = 1/\ell$, $\mathcal{Q}_i^{(0)} = 0.5$, and $\#improve \;\; g_i = 1$ and $\#trials \;\; g_i = 2$. Until a stopping criteria is met, a scalarization function g_i is selected from the set of scalarizations to be proportional with the target distribution $\mathcal{P}_i^{(t)}$. A solution s is generated using a uniform random distribution or genetic operators, like for the stochastic PLS. The scalarized local search $\mathsf{SLS}(s, g_i)$ is returning a non-dominated archive A' that is merged with the solutions from the current non-dominated archive. To keep A consistent, the dominated solutions are deleted. We update the values of distributions $\mathcal{R}^{(t)}$ and $\mathcal{P}^{(t)}$.

Scalarized local search. It is a simplification of the Pareto local search from Algorithm 1 in the sense that there are few solutions in a neighbourhood that are non-dominated. Because we want to further integrate this local search into the framework in Algorithm 2, the explored solutions are stored into a non-dominated archive A' initialized with a given solution s, where $s.visited = false$. While there exist solutions in A' that have the visited flag set to false, SLS generates the neighbourhood of this solution, $\mathcal{N}(s')$. If there exists in $\mathcal{N}(s')$ a solution s'' with a scaralized value smaller than the scalarized value of s', then the solution s'' is added to A'. The solutions that are dominated in A' are removed. The algorithm returns A'.

Adaptive pursuit. Each measurement of performance for the scalarized LS using g_i, $\mathsf{SLS}(\cdot, g_i)$, has a time stamp, t'. Note that, in Algorithm 2, the reward $\mathcal{R}_i^{(t)}$ is connected with the dominance relationship between the archive A' resulting from running an SLS and the archive A over all the runs. An improvement of using $\mathsf{SLS}(\cdot, g_i)$ means that the resulting A' contains at least one new solution, $s \in A'$, that dominates at least one other solution from the current archive A, $\exists s' \in A$

Algorithm 4: Adaptive pursuit (AP)

```
Require: $\mathbf{g} = (g^1, \ldots, g^\ell)$
  $\mathcal{R}_i^{(t)} \leftarrow$ GetRewardApplying($g_i$)
  Update $\mathcal{Q}_i^{(t+1)}$ with reward $\mathcal{R}_i^{(t)}$
  High-rank $\mathcal{Q}^{(t+1)}$ and set the indexes in $r$
  for i = 1 to $\ell$ do
    $\mathcal{P}_i^{(t+1)} \leftarrow \mathcal{P}_i^{(t)} + \beta \cdot (p_M - \mathcal{P}_i^{(t)})$
  end for
  $t \leftarrow t + 1$
  return $\mathcal{P}^{(t)}$
```

such that $\mathbf{f}(s) \prec \mathbf{f}(s')$. Let's denote

$$\mathcal{Q}_i^{(t)} = \frac{\#\ improve\ g_i}{\#\ trials\ g_i} \tag{1}$$

where $\#improve\ g_i$ is the number of improvements for g_i, and $\#trials\ g_i$ are the number of trials for g_i. If applying g_i results in an improvement, then $\mathcal{Q}_i^{(t)}$ is increasing

$$\mathcal{Q}_i^{(t)} < \mathcal{Q}_i^{(t+1)} \Leftrightarrow \frac{\#improve\ g_i}{\#trials\ g_i} < \frac{\#improve\ g_i + 1}{\#trials\ g_i + 1}$$

Otherwise, $\mathcal{Q}_i^{(t)}$ decreases.

The update of $\mathcal{P}^{(t)}$ is started by ranking the quality vector $\mathcal{Q}^{(t)}$. The rank vector, $\mathbf{r}$, ranks the most rewarded operator with 1, the operator with maximum $\mathcal{Q}^{(t)}$.

Let $\beta \in [0, 1]$ be the learning rate that determines the speed with which the algorithm converges to the maximum and minimum probabilities values. A large β value means faster convergence of the algorithm to the target probabilities, which means a poorer use of certain scalarization functions. A small β value means slower convergence and thus more chances for the less rewarded scalarization functions to be tested.

To conclude, the adaptive scalarized local search adaptively chooses the scalarization function that best explore the current region of the search space. Note the difference with the adaptive Pareto local search [1] that adapts the exchange rate of the recombination-like mutation, i.e. the path guided mutation. The resemblance between the two algorithms in the same adaptive pursuit mechanism and their use in meta-heuristics for multi-objective combinatorial optimization problems.

IV. Genetic scalarized local search (GSLS)

We propose an algorithm which is able to generate and select a set of scalarization functions that best improve the performance of stochastic local search algorithms. This strategy generates new weights from the weights of the current scalarization functions using real-coded genetic operators [2]. We assume that the current weights explore the search space well and thus the novel weights which inherit their properties explore the search space well. The advantage in generating new weights is that they are still different and they could explore the other parts of the search space. We call this algorithm *genetic scalarized local search (GSLS)*.

The pseudo-code for stochastic GSLS is given in Algorithm 5, where the pseudo-code for SLS was already presented in Algorithm 3. The pseudo-code for the genetic scalarization generator is given in Algorithm 6.

Algorithm 5: Genetic scalarized local search (*GSLS*)

```
Require: $\mathbf{g} = (g_1, \ldots, g_\ell)$
  while a stopping criteria met do
    Generate $g_i'$ from $\mathbf{g}$ using $RGA(\mathbf{g})$
    Generate a solution $s$ from $A$ to restart LS
    $A' \leftarrow \mathsf{SLS}(s, g_i)$
    if $\exists s' \in A'$, $\exists s \in A$ such that $\mathbf{f}(s') \preceq \mathbf{f}(s)$ then
      $g_i \leftarrow g_i'$
      $\#improve\ i \leftarrow \#improve\ i + 1$
    end if
    $\#trials\ i \leftarrow \#trials\ i + 1$
    $A \leftarrow A' \cup A$
    remove dominated solution from $A$
  end while
  return $A$
```

A. The algorithm

In Algorithm 5, to select the scalarization functions that perform well, we use a a quality vector (or estimated reward) $\mathcal{Q}^{(t)} = \{\mathcal{Q}_1^{(t)}, \ldots, \mathcal{Q}_\ell^{(t)}\}$ that indicates the performance of a local search using the i-th scalarization function, g_i. The quality vector is updated as described in Equation 1. This type of selection resembles roulette wheel selection from evolutionary computation.

To generate new scalarization functions, we use some of the real-coded operators proposed in [2]. The goal is to generate scalarization functions whose weights are similar with the weights of the current scalarization functions. If the newly generated scalarization function improves the current non-dominated archive A, it will replace it in the population of scalarization functions but the quality vector is kept because we assume the child scalarization will be even better than the parent scalarization. If this is not the case, the quality vector is degrading, the scalarization function being considered of a poor quality.

Until a stopping criteria is met, a scalarization function g_i' is generated using the genetic operators from Algorithm 6. A solution s is generated using the genetic operators or a uniform random distribution from A. A scalarized local search is run with the generated parameters s and g_i'. If the SLS using the proposed scalarization function g_i' contains at least a solution that is improving over at least another solution from the current archive A, then the scalarization function g_i' will replace the current scalarizer g_i. The quality vector of the scalarization function i is updated. In this way, the best performing scalarization functions are passed to the next generation.

For comparison with stochastic ASLS algorithms, in Algorithm 5, the scalarization functions are initiated with a set of scalarizarion functions $\mathbf{g} = (g_1, \ldots, g_\ell)$ that are played few times. Like in the stochastic ASLS from Section III, the scalarization functions are selected according to their performance. Unlike for the SASLS algorithm, the weights of the scalarization functions are adapted over time.

Algorithm 6: Real-coded GA (*RGA*)

```
Require: $\mathbf{g} = (g_1, \ldots, g_\ell)$
    Select the main parent scalarization $g_i$ proportional with $\mathcal{Q}^{(t)}$
    Select the rest of $m$ scalarizations proportional with $\mathcal{Q}^{(t)}$, $\{g_{j_1}, \ldots, g_{j_m}\}$
    Switch ($\alpha$)
    case $\alpha < \alpha_R$: $g'_i \leftarrow$ Rotation($g_i$, $g_{j_1}$,…,$g_{j_m}$)
    case $\alpha < \alpha_T$: $g'_i \leftarrow$ Transl($g_i$,$g_{j_1}$,…,$g_{j_m}$)
    default: $g'_i \leftarrow$ Mutation($g_i$)
    endswitch
    return $g'_i$
```

Generating new scalarization functions. In Algorithm 6, let's consider a set of scalarization functions $\mathbf{g} = (g_1, \ldots, g_\ell)$ and each scalarization function g_i a real valued string with values between 0 and 1 associated with its weights. Let's denote the weights of the scalarization function g_i with $\{w_{i1}, \ldots, w_{im}\}$, where $0 \leq w^i_j \leq 1$, $1 \leq i \leq m$, and $j = 1, 2$, and $\sum_{i=1}^{m} w^i_j = 1$. Then, $\mathbf{g}$ is a population of scalarization functions.

The mutation operator, Mutation(g_i), mutates each weight of a scalarization function independently using a normal distribution.

Two recombination operators considered the most representative for linear transformation are implemented. These recombination operators needs $m + 1$ parents when the string representing a solution is of size m. Let's consider a scalarization function $g_i = (w_{i1}, \ldots, w_{1m})$ chosen to be the main parent. We select the rest of m parents without replacement $\{g_{j_1}, \ldots, g_{j_m}\}$. Only one scalarization function is generated, g'_i, at the time.

The rotation recombination, Rotation(g_i,g_{j_1},…,g_{j_m}), considers that the scalarizations are positioned on a m-dimensional hypersphere. The scalarization generated also belongs to this hypersphere around the main parent, that is rotated with a small normally distributed angle. The translation recombination, Transl(g_i,g_{j_1},…,g_{j_m}), translates the main parent with a normally distributed variable. The direction is composed from the line between mean of all parents and the main parent, and $m - 1$ perpendiculars to it. These operators are explained in detail in [2].

The main difference between the two recombination operators is that rotation recombination is assuming that the weights of different scalarization functions have correlated values. The translation recombination updates the weights of each of the objective independently. The translation recombination is selected with probability α_T, the rotation recombination is selected with probability α_R, and the mutation is selected with probability $1 - \alpha_T - \alpha_R > 0$, where $\alpha_T, \alpha_R \in (0, 1)$. We set the variance of the normal distribution to be the same for all the operators, 0.05.

To conclude, the use of genetic operators to propose new directions for scalarization functions based on the current scalarization functions is novel. Up to date, there is only one other technique which generate new directions of search using the *dichotomic operator* [12], [5], but this technique adapts its search based on current solutions in the Pareto local optimal set. The drawback of the last technique is that the same direction is proposed if there are the same solutions in the Pareto local optimal set and thus the search will be stuck for a long time. Furthermore, this method is designed exclusively for two objectives, whereas our method can be applied for any number of objectives.

V. Experimental results

In this section, we experimentally compare the proposed algorithms and some of the related algorithms from the state of the art.

The tested problems. We compare the proposed algorithms on nine bi-objective QAP instances generated using the software of Knowles and Corne [10]. These bQAP instances have positive correlations $\rho = \{0.25, 0.5, 0.75\}$ and a large number of facilities $n = \{50, 75, 100\}$. To facilitate comparisons, six of these QAP instances, i.e. $n = \{50, 75\}$ are unstructured bQAP instances in Paquete's study [13] (*http://eden.dei.uc.pt/ paquete/qap/*). Three of the QAP instances, $n = 100$, are from [4].

For QAPs with a large number of facilities and high positive correlation, Paquete reported a poor performance of multi-restart PLSs. [4] reported good performance for stochastic PLS on the same bQAP instances because of the structure exploration.

The tested algorithms. We compare ten algorithms: two PLS based algorithms and eight SLS based algorithms.

1. *Multi-restart PLS* (*MPLS*) using a best improvement 2-exchange neighbourhood restarts M times. The number of swaps S is counted. All the algorithms are run for the same number of swaps S.

2. *Stochastic PLS* (*SPLS*) after randomly restarting 10 PLSs, runs S swaps and the restarting points are generated 50% with mutation and 50% with the path-guided mutation. The exchange rate for both mutation operators is uniform randomly chosen between 3 and $n/3$ based on the experimental study in [3].

3. *Multi-restart SLS* (*MSLS*) uniform randomly chooses scalarization functions from a set of $\ell = 11$ functions and restarts the SLS from a uniformly randomly generated solutions. The first scalarization function has the weights $(1, 0)$ and the other ten scalarization functions incrementally decreases/increases the first/second weight with 0.1.

4. *Stochastic SLS* (*SSLS*) uniform randomly chooses from the 11 scalarization functions and restart the SLS from solutions generated 50% with mutation and 50% with path guided mutation.

5. *Multi-restart adaptive SLS* (*MASLS*) uses random solutions to restart SLS and the adaptive pursuit to often select the best scalarization function. We have $\ell = 11$ scalarizations to choose from. We set $p_m = 0.075$, and then $p_M = 0.25$, and a low learning rate $\beta = 0.1$. We set the value for time window to $w = 500$.

n	ρ	Multi-restart PLS					Stochastic PLS				
		MPLS	MASLS	MGSLS	MDSLS	MSLS	SPLS	SASLS	SGSLS	SDSLS	SSLS
50	0.25	0.98 $\pm$ 0.01	0.96 $\pm$ 0.03	0.98 $\pm$ 0.02	0.83 $\pm$ 0.01	0.84 $\pm$ 0.01	**1.02 $\pm$ 0.02**	0.96 $\pm$ 0.02	1.0 $\pm$ 0.02	0.89 $\pm$ 0.02	0.94 $\pm$ 0.02
	0.50	0.87 $\pm$ 0.03	0.85 $\pm$ 0.05	0.79 $\pm$ 0.02	0.65 $\pm$ 0.02	0.67 $\pm$ 0.02	0.96 $\pm$ 0.03	0.87 $\pm$ 0.04	0.91 $\pm$ 0.04	0.80 $\pm$ 0.04	0.81 $\pm$ 0.03
	0.75	0.62 $\pm$ 0.06	0.60 $\pm$ 0.10	0.58 $\pm$ 0.02	0.43 $\pm$ 0.04	0.44 $\pm$ 0.06	0.87 $\pm$ 0.08	0.80 $\pm$ 0.07	**0.89 $\pm$ 0.07**	0.78 $\pm$ 0.10	0.76 $\pm$ 0.09
75	0.25	0.96 $\pm$ 0.01	0.96 $\pm$ 0.03	0.90 $\pm$ 0.01	0.79 $\pm$ 0.01	0.81 $\pm$ 0.01	**1.02 $\pm$ 0.02**	0.96 $\pm$ 0.02	0.94 $\pm$ 0.03	0.90 $\pm$ 0.03	0.91 $\pm$ 0.03
	0.50	0.85 $\pm$ 0.03	0.86 $\pm$ 0.04	0.72 $\pm$ 0.06	0.64 $\pm$ 0.02	0.65 $\pm$ 0.01	**0.97 $\pm$ 0.04**	0.89 $\pm$ 0.04	0.95 $\pm$ 0.02	0.90 $\pm$ 0.04	0.88 $\pm$ 0.05
	0.75	0.44 $\pm$ 0.10	0.62 $\pm$ 0.09	0.57 $\pm$ 0.10	0.32 $\pm$ 0.04	0.33 $\pm$ 0.04	0.83 $\pm$ 0.09	0.74 $\pm$ 0.11	**0.86 $\pm$ 0.09**	0.72 $\pm$ 0.08	0.74 $\pm$ 0.09
100	0.25	0.99 $\pm$ 0.01	0.94 $\pm$ 0.02	0.90 $\pm$ 0.01	0.77 $\pm$ 0.01	0.79 $\pm$ 0.01	**1.07 $\pm$ 0.02**	0.98 $\pm$ 0.02	1.03 $\pm$ 0.01	0.93 $\pm$ 0.03	0.96 $\pm$ 0.02
	0.50	0.89 $\pm$ 0.02	0.91 $\pm$ 0.04	0.84 $\pm$ 0.01	0.61 $\pm$ 0.02	0.63 $\pm$ 0.02	**1.04 $\pm$ 0.03**	0.92 $\pm$ 0.04	0.96 $\pm$ 0.05	0.83 $\pm$ 0.04	0.86 $\pm$ 0.04
	0.75	0.55 $\pm$ 0.04	0.63 $\pm$ 0.08	0.63 $\pm$ 0.08	0.33 $\pm$ 0.03	0.35 $\pm$ 0.03	0.85 $\pm$ 0.10	0.76 $\pm$ 0.10	**0.90 $\pm$ 0.07**	0.55 $\pm$ 0.10	0.59 $\pm$ 0.09

TABLE II. THE HYPERVOLUME UNARY INDICATOR OF THE TEN LOCAL SEARCH ALGORITHMS ON NINE BQAP INSTANCES.

6. *Stochastic adaptive SLS* (*SASLS*) generates new solutions using genetic operators like *SSLS* and adaptively selects scalarization functions like MASLS.

7. *Multi-restart genetic SLS* (*MGSLS*) uses random solutions to restart SLS and the genetic operators to generate fit scalarization functions. We use the three operators equally, $\alpha_R = \alpha_T = 0.33$.

8. *Stochastic GPLS* (*SGSLS*) is a MGSLS that restarts the SLS from solutions generated with mutation and path-guided mutation.

9, 10. Finally, we compare the above algorithms with an algorithm that generates the scalarization functions using the dichotomic operator [5]. We integrate this operator in the GSLS algorithm and we call the corresponding algorithms the *multi-restart dichotomic SLS* (*MDSLS*) and the *stochastic dichotomic SLS* (*SDSLS*).

Each instance of the ten algorithms is run 50 times on each of the nine bQAP instances. For a fair comparison, for the same bQAP instance, all the algorithms are run the same number of exchanges that are equivalent of $M = 100$ runs of *MPLS*. We consider this stopping criteria equivalent with an equivalent running time for all the algorithms tested on a specific bQAP instance.

We show results only for linear scalarization function. We have also performed experiments with the Tchebycheff scalarization but the results where statistical equivalent with the results for the linear scalarization. We explain this with convex Pareto local optimal sets for the tested bQAP problems.

The performance of local search algorithms. The *hypervolume* indicator is a unary performance measure designed to compare the Pareto fronts given by multi-objective combinatorial optimization algorithms [15]. The larger the hypervolume, the larger is the volume between a reference point, in this case the worst point in the bi-variate normalized search space, and the Pareto front. The larger the hypervolume the better the algorithm performs.

For comparison purposes, a normalization function assigns to the best point(s) in a dimension the value 1 and to the worst point(s) the value 2. All the other points are scaled to a value between 1 and 2 in both dimensions, and the reference point is $\{2, 2\}$.

Multi-restart vs stochastic restarting strategies. In Table II, we present the hypervolume indicators for the ten tested algorithms. Note the four multi-restart algorithms (MPLS, MSLS, MASLS, and MGSLS), are performing significantly worst than the corresponding stochastic algorithms, (SPLS, SSLS, SASLS, and SGSLS). This means that restarting the Pareto and stochastic local search algorithms using the genetic operators that exploit the structure of the search space is considerably improving the performance of the local search algorithms. This difference in performance is larger for bQAP instances with larger number of facilities and stronger correlation between the flow matrices.

Pareto vs scalarized local search. In general, Pareto local search based algorithms outperform the standard scalarized local search algorithms. The multi-restarted PLS (MPLS) outperforms the multi-restart SLS (MSLS) and the multi-restart DSLS (MDSLS), but MPLS is outperformed by stochastic version of the same algorithms, stochastic SLS (SSLS) and stochastic DSLS (SDSLS), where the structure of the search space is exploited with genetic like operators. The performance of multi-restart ASLS (MASLS) is comparable or higher than the performance of MPLS, meaning the the structure can be exploited also with scalarization functions. The performance of the multi-restart GSLS is higher than of the multi-restart SLS and DSLS but lower than the performance of multi-restart ASLS. The best performing algorithms are the stochastic PLS and the stochastic genetic SLS, the later with a slightly better performance.

Interacting vs non-interacting scalarization functions. Overall, the SLS algorithms using the interacting scalarization functions are outperforming the other non-interacting scalarization functions. Note that DSLS, like genetic SLS, adapts the weights of the scalarization functions. Unlike genetic SLS, DSLS uses a single scalarization function that adapts the scalarizer weights from the solutions in the current non-dominated archive A. However, if there are no newly solutions included in A, this technique will generate the same scalarization functions which could stagnate in improving A. Adaptive SLS (ASLS) is systematically better than the genetic SLS for multi-restart strategies, but ASLS is worst than GSLS for stochastic strategies.

Discussion. To conclude, stochastic local search outperforms the multi-restart local search indicating the exploitation of the problem commonalities with the genetic like operators. The local search algorithms, both multi-restart and stochastic, can profit from the use of the proposed interacting scalarization functions.

Let's now discuss the differences in performance between the two scalarization techniques. Intuitively, multi-restart local search are quite good in exploring the search space and a intensification technique for the search direction, i.e. adaptive scalarization functions, is performing the best. The stochastic local search on the other hand already explore the commonalities the search space and a diversification technique that feels the gaps in the Pareto local optimal set, i.e. genetic scalarization functions, are the most effective.

VI. Conclusion

We have proposed two techniques for scalarized local search algorithms that use a set of scalarization functions that interact in order to improve their performance. The adaptive pursuit is integrated into stochastic scalarized local search algorithms to select often scalarization functions that generates SLS runs with the best performance. Real-coded genetic operators are used to generate well performing scalarization functions in the neighbourhood of the current scalarization function. These two mechanisms can be used for any number of objectives of a combinatorial search space.

We experimentally compare the proposed algorithms with some of the state of the art algorithms on four instances of bQAP problems. In general, the Pareto local search based algorithms are outperforming, in the same settings, the scalarized local search based algorithms. The exception is the highly correlated problems where exploring commonalities of the search space has the highest impact on the performance of the algorithms. The stochastic genetic SLS algorithms are performing comparable with the stochastic PLS algorithms. We conclude that the interacting scalarization functions through their mechanisms of intensification/diversification of the search can complement and thus improve the restarting local search mechanisms.

References

[1] M. M. Drugan and D. Thierens. Generalized adaptive pursuit algorithm for genetic Pareto local search algorithms. In *Proc of Conf. on Genetic and Evol. Comp., GECCO'11*, pages 1963–1970, 2011.

[2] M.M. Drugan and D. Thierens. Geometrical recombination operators for real-coded evolutionary MCMCs. *Evolutionary Computation*, 18(2):157–198, 2010.

[3] M.M. Drugan and D. Thierens. Path-guided mutation for stochastic Pareto local search algorithms. In *Parallel problem solving from Nature (PPSN)*, volume LNCS, pages 485–495. Springer, 2010.

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

[5] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Improving the anytime behavior of two-phase local search. *Ann. Math. Artif. Intell.*, 61(2):125–154, 2011.

[6] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multi-objective combinatorial optimization. *OR Spekctrum*, 22:425–460, 2000.

[7] G. Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization*. Springer, 2008.

[8] H.H. Hoos and T. Stutzle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[9] A. Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137:50–71, 2002.

[10] J. Knowles and D. Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In *Proc. of. Evolutionary Multi-Criterion Optimization EMO 2003*, number 2632 in LNCS, pages 295–310. Springer, 2003.

[11] A. Liefooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E.-G. Talbi. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *J. Heuristics*, pages 1–36, 2011.

[12] T. Lust and J. Teghem. Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510, 2010.

[13] L. Paquete and T. Stützle. A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *European J. of Oper. Res.*, 169(3):943–959, 2006.

[14] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proc. of Conf. on Genetic and Evol. Comp., GECCO'05*, pages 1539–1546, 2005.

[15] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V.G. da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE T. on Evol. Comput.*, 7:117–132, 2003.