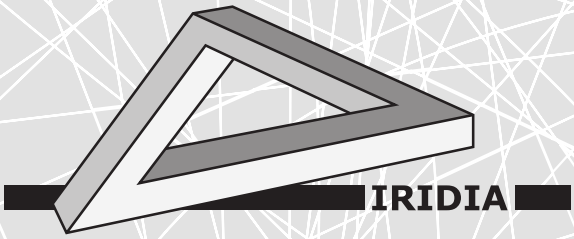

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Combining Two Search Paradigms for Multi-objective Optimization: Two-Phase and Pareto Local Search

Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2012-004

March 2012

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2012-004

Combining Two Search Paradigms for Multi-objective Optimization: Two-Phase and Pareto Local Search

Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle

Abstract In this chapter, we review metaheuristics for solving multi-objective combinatorial optimization problems, when no information about the decision maker's preferences is available, that is, when problems are tackled in the sense of Pareto optimization. Most of these metaheuristics follow one of the two main paradigms to tackle such problems in a heuristic way. The first paradigm is to rely on Pareto dominance when exploring the search space. The second paradigm is to tackle several single-objective problems to find several solutions that are non-dominated for the original problem; in this case, one may exploit existing, efficient single-objective algorithms, but the performance depends on the definition of the set of scalarized problems. There are also a number of approaches in the literature that combine both paradigms. However, this is usually done in a relatively ad-hoc way. In this chapter, we review two conceptually simple methods representative of each paradigm: Pareto local search and Two-phase local search. The hybridization of these two strategies provides a general framework for engineering stochastic local search algorithms that can be used to improve over the state-of-the-art for several, widely studied problems.

1 Introduction

Optimization problems appear in many different real-world situations of high social, environmental or economic relevance. Often, such problems are evaluated according to various conflicting objectives. Not surprisingly, multi-objective optimization is attracting significant efforts from researchers, such that nowadays it has become a mature field of research. Many early studies on the development of methods and algorithms for multi-objective problems use an *a priori* approach where several

Jérémie Dubois-Lacoste, Manuel López-Ibáñez, Thomas Stützle
IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium.
e-mail: {jeremie.dubois-lacoste,manuel.lopez-ibanez,stuetzle}@ulb.ac.be

objectives are aggregated into a single objective function. However, when no information is known about the decision maker's preferences one must rely on an *a posteriori* approach. In this case, the possible solutions to the problem are evaluated in the *Pareto sense*, using what is called Pareto dominance (see Section 2). When optimizing a multi-objective problem in the Pareto sense, the goal is to return a set of mutually non-dominated solutions, among which the decision maker can then choose the final solution to implement.

In this chapter, we focus on problems that are *combinatorial*. Solutions for such problems are made of discrete components and analytical methods cannot be used to solve them. For many relevant problems, no algorithm is known to find the optimum in polynomial time w.r.t. the size of the problem. Formally speaking, these problems are NP-hard [29]. Multi-objective combinatorial optimization problems (MCOPs) are always at least as hard as their single-objective counterparts. Often, multi-objective versions of a problem are NP-hard even if the single-objective version of the problem is not. An example is the shortest path problem, where the single-objective variant can be solved in polynomial time but the multi-objective version is NP-hard [28]. When dealing with such NP-hard problems, the typical sizes of the instances prevent the use of exact algorithms, and one must rely on *heuristic* algorithms. A heuristic algorithm is designed to return solutions in polynomial time (better said, quickly enough to be practical), without ensuring that they are optimal.

We discuss heuristic methods for tackling MCOPs with an a posteriori approach. From a very abstract perspective, most algorithms follow one of two main search paradigms: algorithms can be *dominance*-based or *scalarization*-based. Dominance-based methods tackle multi-objective problems by using some form of Pareto dominance relationship among solutions. Scalarization-based methods transform the multi-objective problem into a set of single-objective problems. By solving these single-objective problems, scalarization-based algorithms can provide solutions to the original multi-objective problem. In addition, several algorithms also combine some elements from these two search paradigms, that is, they are hybrid search methods.

For each of the two search paradigms, we present a representative local search framework. As a representative of dominance-based methods we discuss Pareto local search (PLS) [51], and as a representative of scalarization-based methods we present two-phase local search (TPLS) [53]. We describe these two methods together with their latest associated developments as well as similar methods from the literature; we show that their hybridization provides a general framework for multi-objective combinatorial optimization that leads to high performing algorithms.

This chapter is structured as follows. We introduce in Section 2 the basics of multi-objective optimization in the Pareto sense. Next, we present the two different search paradigms and the PLS and TPLS frameworks, in Sections 3 and 4, respectively. In Section 5, we explain how TPLS and PLS can be hybridized into a general framework and we review some hybrid algorithms in the literature. Section 6 reviews results that have been obtained by this framework. We conclude and highlight some directions for future research in Section 7.

2 Preliminaries

When tackling a multi-objective problem in the Pareto sense, the notion of optimal solution from single-objective optimization does not apply anymore. A solution s in the multi-objective case is better than another solution r if s is better than r for at least one objective and not worse for any of the remaining ones. If none of the two solutions is better than the other, they represent two different trade-offs of the objectives values that, without knowledge of the decision maker's preferences, are considered to be indifferent. The goal of an algorithm tackling a multi-objective problem in the Pareto sense is then to return all solutions representing different trade-offs among which the decision maker can choose the preferred one.

More formally, let us consider an MCOP with p objectives, all to be minimized. We call $\mathbf{f}(s)$ the vector of the objective function values of solution s; $f_k(s)$ denotes the specific value of objective k, that is, the k-th component of the objective function vector $\mathbf{f}(s)$. We call $N(s)$ the set of all neighbors of solution s.

Definition 1 (Dominance). A solution s_1 is said to dominate a solution s_2 ($s_1 \prec s_2$) if and only if $f_k(s_1) \leq f_k(s_2)\ \forall k = 1,\ldots,p$ and $\exists\, j \in \{1,\ldots,p\}$ such that $f_j(s_1) < f_j(s_2)$.

Definition 2 (Weak dominance). A solution s_1 is said to weakly dominate a solution s_2 ($s_1 \preceq s_2$) if and only if $f_k(s_1) \leq f_k(s_2)\ \forall k = 1,\ldots,p$.

Definition 3 (Incomparable solutions). Solutions s_1 and s_2 are said to be incomparable ($s_1 \parallel s_2$) if and only if neither $s_1 \npreceq s_2$ nor $s_2 \npreceq s_1$, and $s_1 \neq s_2$.

The fact that solutions can be incomparable is a fundamental difference to single-objective optimization, where a total ordering of the solutions exists.

Definition 4 (Pareto global optimum solution). Let S denote the set of all feasible solutions. A solution $s_1 \in S$ is a Pareto global optimum if and only if $\nexists\, s_2 \in S$ such that $s_2 \prec s_1$. Such solutions are also called *efficient*.

Definition 5 (Pareto local optimum set). A set S' of solutions is a Pareto local optimum if $\forall s \in S', \forall\, s_1 \in N(s), \exists s_2 \in S'$ verifying $s_2 \preceq s_1$.

Definition 6 (Pareto front). Let S denote the set of all feasible solutions. A set S' is a Pareto global optimum set if and only if it contains all the Pareto global optimum solutions of S and only these solutions. The set of objective vectors of the Pareto global optimum is called the Pareto front.

Several Pareto global optimum solutions can have the same objective vector in the Pareto front. In practice, it is common to return only one solution for each objective vector. Such a set is also called strict Pareto global optimum set [52] or strictly Pareto optimal set [21].

The dominance relations can be extended to sets of solutions. We present here the weak dominance relation on sets, which we use in this chapter. In the following, A and B denote two sets of solutions.

Definition 7 (Weak dominance on sets). A set A is said to weakly dominate a set B ($A \lhd B$) if and only if $\forall\, s_i \in B$, $\exists\, s_j \in A$, such that $s_j \preceq s_i$, and $A \neq B$.

Definition 8 (Incomparable sets). A and B are said to be incomparable ($A \parallel B$) if and only if neither $A \lhd B$ nor $B \lhd A$, and $A \neq B$.

In the next two sections we present the two search paradigms more in detail and we review representative methods in the literature, focusing on PLS and TPLS.

3 Dominance-based Multi-Objective Optimization

We say that algorithms are dominance-based if they use some form of Pareto dominance for acceptance decisions on solutions. When comparing solutions using Pareto dominance, solutions may be mutually non-dominated; thus, there is only a partial order defined over solutions, which is a fundamental difference to the single-objective case. Also for this reason, dominance-based algorithms keep an archive of solutions instead of only a single solution as the best one found so far.

There are many algorithms for MCOPs that are purely dominance-based. We restrict our discussion to methods that are based on the iterative improvement of the set of non-dominated solutions by performing local search (or mutation) of solutions one at a time. We do not consider here population-based algorithms such as multi-objective evolutionary algorithms [10, 8] or multi-objective ant colony optimization algorithms [5, 27]. However, it should be noted that these algorithms also often make direct or indirect use of Pareto dominance for directing the search, in particular, in acceptance or selection decisions on solutions.

Pareto local search (PLS) [51] is a paradigmatic representative of dominance-based multi-objective algorithms that improve solutions one at a time by use of neighborhood search [52]. While the original motivation for proposing PLS was to study the connectedness of solutions [51], PLS turned out to be also an effective local search method for multi-objective problems. Independently, a very similar algorithm was proposed by Angel et al. [4].

PLS extends iterative improvement procedures from the single-objective case to the multi-objective case by changing the acceptance criterion. While in the single-objective case an iterative improvement algorithm accepts a new solution if it is better than the current one, in the multi-objective case PLS accepts a new solution to enter the archive only if it is not dominated by any solution in the archive. PLS takes care that the archive contains only non-dominated solutions by filtering out dominated ones.

Algorithm 1 illustrates the general framework of PLS. It is initialized by an initial set A of mutually non-dominated solutions, called *archive*. These solutions are initially marked as unexplored (line 2). PLS then iteratively applies the following steps. First, a solution s is selected among all unexplored ones (*selection step*, line 5). Then, some (or all) of the neighbors of s, are explored (*neighborhood exploration*) and all the neighbors that are accepted (*acceptance criterion*) w.r.t. the archive A are

Algorithm 1 Pareto Local Search

```
1: Input: An initial set of non-dominated solutions A
2: explored(s) := FALSE   ∀s ∈ A
3: A_0 := A
4: repeat
5:    s := SelectSolution(A_0)
6:    repeat
7:       s' := NeighborhoodExploration(s)
8:       if AcceptSolution(A, s') then
9:          explored(s') := FALSE
10:         A := Update(A, s')
11:      end if
12:   until (termination criterion)
13:   explored(s) := TRUE
14:   A_0 := {s ∈ A | explored(s) = FALSE}
15: until A_0 = ∅
16: Output: A
```

added to A (lines 8 to 11). Solutions in A that are dominated by the newly added solutions are removed (procedure Update in line 10). Once the termination criterion for the exploration of the neighborhood of s is met, s is marked as explored (line 13). When all solutions have been explored, and no more new non-dominated solutions can be discovered, the algorithm stops in a Pareto local optimum. Algorithm 1 is a generic outline and different variants of PLS can be obtained by different instantiations of the components SelectSolution, AcceptSolution and NeighborhoodExploration. In the original PLS algorithm, as proposed in [51], the three main components are implemented as follows.

Selection step. The next solution to be explored is selected uniformly at random from the unexplored ones in the archive.

Neighborhood exploration. The neighborhood of a solution is always explored entirely. This corresponds to the "best-improvement" neighborhood exploration in single-objective local search, i.e., all neighborhood solutions are explored before exploring the neighborhood of a new solution.

Acceptance criterion. The original PLS accepts any non-dominated solution for inclusion in the archive.

The method proposed by Angel et al. [4] is very similar to PLS. The difference lies in the selection step: in the method of Angel et al., contrary to PLS, the neighborhoods of all unexplored solutions are explored before updating the archive. It has a stronger exploration capability than PLS because the archive is not updated immediately after the neighborhood of a single solution has been explored. Due to this, neighbors of solutions that otherwise would have become dominated can be examined in addition to those of non-dominated solutions.

Liefooghe et al. [43, 42] study the performance of some variants of the PLS algorithm. They test variants of the selection step that are obtained by restricting the overall exploration with a limit on the number of solutions to be selected, and vari-

ants of the neighborhood exploration itself, combining different ways of scanning the neighborhood and different acceptance criteria. The several variants are compared experimentally by the authors. In their experimental setup, they choose the same, predefined computation time limit for all variants. Variants that would finish before this computation time limit are then restarted from scratch and a variant is judged by the final aggregate non-dominated set found across the multiple restarts. As a result, they highlight variants that are the most effective if the computation time is known a priori and the PLS algorithms is launched several times.

Another recent study of PLS is carried out by Dubois-Lacoste et al. [20]. The aim is to examine variants for the algorithmic components of PLS with a main focus on their impact on the *anytime behavior* [64] of PLS. Several variants are tested for the three main components of PLS. It is shown that some of the resulting combinations can significantly speed-up the convergence of PLS towards good Pareto front approximations.

The fact that PLS stops upon finding a Pareto local optimum set can be a disadvantage if the algorithm finishes while there is still computation time available. A possibility is to keep the non-dominated solutions found in an external archive and to restart PLS "from scratch" (as done in [42]). There were other extensions that aim at obtaining a more efficient search. Alsheddy and Tsang [1] proposed an extension of PLS that continues the search when a Pareto local optimum set is found without restarting from different solutions. The idea is based on the *guided local search* [63] strategy in the single-objective case: a penalty is applied to worsen the components of the objective vectors of solutions in the current archive, allowing the algorithm to escape from a Pareto local optimum set. Other strategies to continue the search focus on generating good solution(s) to restart the search. In particular, solutions mutated from the ones in the Pareto local optimum set can be used, resulting, in some sense in an extension of *iterated local search* [47] for single-objective problems. A study of such strategies was done by Drugan and Thierens [12]. In that paper, the authors showed that the best results on the bi-objective quadratic assignment problem are attained when restarting PLS from new solutions on a "path" between two solutions in the Pareto local optimum set (this path is constructed in a manner similar to *path-relinking* [32]). Geiger [30] proposed to apply a different neighborhood operator (w.r.t. the one used during the search) when PLS converges to a Pareto local optimum, allowing PLS to find possibly new non-dominated solutions. This idea can be seen as an extension of *variable neighborhood search* [36].

Some evolutionary algorithms are similar to PLS. PAES has been proposed by Knowles and Corne [40] as an algorithm whose simplicity should make it a baseline for comparison to more complex evolutionary algorithms. In PAES, a solution is selected in the current archive of non-dominated solutions and a mutation operator is applied to obtain a new candidate solution. This new candidate is potentially inserted in the archive, which is then updated to keep only non-dominated solutions. The archive size is kept limited by using an archive bounding strategy. Contrary to PLS, this algorithm does not have a natural stopping criterion since solutions are never marked as explored. Laumanns et al. proposed the SEMO algorithm [41], which is similar to PAES but solutions are selected from an archive whose size is not

limited. Differently from PAES, SEMO marks solutions as explored analogously to PLS. The authors also test variants of SEMO that differ in the selection step. These variants tend to balance the number of times solutions are selected for mutation, or they try to focus on the most recently found solutions.

An advantage of dominance-based algorithms is that they deal with an archive of solutions rather than a single solution. Therefore, they can return quickly numerous non-dominated solutions to the problem. However, this can also be a drawback since dealing with possibly many solutions can make the exploration of the search space slower in terms of the closeness to the Pareto front. In the next section, we present the scalarization-based search paradigm for multi-objective optimization, whose aim is to provide quickly few high-quality solutions whose objective vectors are close to the Pareto front.

4 Scalarization-based Multi-Objective Optimization

Scalarization-based algorithms rely on solving several single-objective problems in order to find various non-dominated solutions for a multi-objective problem. To do so, the multiple objective functions are *aggregated* into a single scalar function. In this way, the solutions can be compared by scalar values, resulting in a total ordering of solutions. In other words, an aggregation transforms the multi-objective problem into a (*scalarized*) single-objective problem, often called simply *scalarization*.

There are many ways of scalarizing multi-objective problems. However, most commonly few standard methods are used for their simplicity and desirable properties. We present here the most common methods in use in the context of heuristic algorithms; for other possibilities, we refer the interested reader to [21, 8].

- **Linear aggregation.** The weighted sum method defines a linear aggregation of the objectives that is commonly used to define the preferences of the decision maker with an a priori approach. It is also used to define scalarizations for tackling problems with an a posteriori approach and we also use it in this chapter in the experimental part. A weight vector is used to give a relative importance to each objective. Let us consider a solution s whose objective function vector is $\mathbf{f}(s) = (f_1(s), f_2(s), \ldots, f_p(s))$, and a weight vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$. We assume, without loss of generality, that the components of $\mathbf{f}(s)$ are non-negative. The scalar value for this solution and this weight is then:

$$f_\lambda(s) = \sum_{1 \leq i \leq p} \lambda_i \cdot f_i(s).$$

  Since the different components of the weight vector have an effect that is relative to the value of each other, there exist infinitely many different weight vectors that define the same scalarization. Therefore, it is common to use normalized weight vectors, whose components sum up to one. The advantage of considering a weighted sum is that an optimal solution for the scalarized problem is a

supported non-dominated solution, that is, its objective vector is located on the convex hull of the Pareto front.

- **Tchebycheff aggregation** The Tchebycheff method requires to define a *reference point*, r, that dominates any feasible solution. Some weights, additionally, can be assigned to each objective with a weight vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$. The scalar value for a solution s is then:

$$f_\lambda(s) = \max\{\lambda_i \cdot |f_i(s) - f_i(r)|\}, \quad i = 1 \ldots p.$$

  Hence, the goal becomes to find a solution as close as possible to the reference point r, using the Tchebycheff distance as a measure of "closeness".
- **Lexicographic ordering**. This method requires to define an order of the objectives, in decreasing importance, and, thus, it defines a total order of the solutions: If some solutions have the same value for objectives 1 to k, objective $k+1$ is used to break ties. The number of different orders of the objectives is limited, and therefore is also the number of solutions that can be obtained with this method. For instance, for bi-objective problems there are only two possible orderings of the objectives and thus only two different scalarized problems can be defined, preventing to return more than two solutions. Thus, when tackling a problem in the Pareto sense, a lexicographic ordering can be used to provide some initial solutions only, and must be used in combination with another technique.

Solving scalarized problems is often done when the multi-objective problem is tackled a priori, that is, the decision maker is able to define the components of the weight vector before the optimization. If the problem is considered using an a posteriori approach, multi-objective algorithms can make use of scalarizations, but the weight vector must be varied *during* the optimization process by the algorithm. An advantage of scalarization-based algorithms is that they can make use of any algorithm known to solve the scalarized problems effectively, and make use of its effectiveness in the multi-objective context. The drawback is that solutions are found one at a time, and the total number of non-dominated solutions returned is at most the number of scalarizations considered, which may be small compared to what is desirable.

Two-phase local search (TPLS) is a representative example of a scalarization-based algorithm [53]. TPLS is a general algorithmic framework that, as the name suggests, is composed of two phases. In the first phase, a single-objective algorithm generates a high-quality solution for one or all objectives, and then one of these high-quality solutions serves as the starting point of the second phase, where a sequence of scalarizations is tackled. Each scalarization uses the best solution found by the previous scalarization as the solution to start from. TPLS will be successful if the underlying single-objective algorithms are high-performing, and if solutions that are close to each other in the solution space have also objective function vectors that are close to each other in the objective space.

Algorithm 2 presents the general framework of TPLS for a bi-objective problem. First, high-quality solutions are generated for each objective (lines 1 and 2) using dedicated single-objective algorithms SLS_1 and SLS_2, and added to the archive

Algorithm 2 General Framework for Two-Phase Local Search

1: $s_1 := \mathsf{SLS}_1()$
2: $s_2 := \mathsf{SLS}_2()$
3: $A := \mathsf{Update}(A, s_1)$
4: $A := \mathsf{Update}(A, s_2)$
5: **repeat**
6: $\quad \lambda := \mathsf{ChooseWeight}(A)$
7: $\quad s' := \mathsf{ChooseSeed}(\lambda, A)$
8: $\quad s' := \mathsf{SLS}_\Sigma(s', \lambda)$
9: $\quad A := \mathsf{Update}(A, s')$
10: **until termination criterion**
11: $\mathsf{Filter}(A)$
12: **Output:** A

(lines 3 and 4). Then, a sequence of scalarizations is solved (lines 5 to 10), based on strategies to generate a weight vector (procedure ChooseWeight on line 6) and to define how the previous solutions can be used as seed for further scalarizations (procedure ChooseSeed on line 7). Solutions are generated using a single-objective algorithm that tackles scalarized problems, SLS_Σ. The archive is updated with the solutions obtained for each scalarization (line 9). Note that the archive could include more information than just the solutions themselves, e.g., it could also include the fact that solutions have already been used as seeds, for which scalarization they have been obtained, etc.

Algorithm 2 is a generic outline that covers both the original TPLS and recent developments, depending on how the procedures ChooseWeight and ChooseSeed are implemented. TPLS in its original form considers a regular sequence of weights. Two main weight-setting strategies have been originally proposed to define the order in which these weights are selected. The simplest way to define a sequence of scalarizations is to use a regular sequence of weight vectors from the first objective to the second (for instance $\lambda_1 = (1, 0.8, 0.6, 0.2, 0)$ with $\lambda_2 = 1 - \lambda_1$) or from the second objective to the first one. However, this introduces a bias towards the region of the objective space where the first scalarizations are performed, and against the region where the last ones are performed [53]. To avoid this effect, a *double* weight setting strategy has been proposed: first a sequence of scalarizations is performed from one objective to the other and then a second sequence of scalarizations is performed in the opposite direction.

Recently, an *adaptive anytime* strategy (AA-TPLS) has been proposed [16, 19] to further improve the TPLS method. Inspired by the dichotomic scheme of Aneja and Nair [3], AA-TPLS has been shown to lead to better results by adapting to the shape of the Pareto front. Moreover, AA-TPLS shows a very good anytime behavior, providing a high-quality approximation to the Pareto front at any time without requiring a predefined computation time limit, as the original TPLS does. To do so, the selection of the next scalarization to be performed is based on the *optimistic hypervolume improvement*, which measures the potential improvement that can be expected in terms of hypervolume [66].

TPLS is a general framework that relies on a simple idea: run several times a single-objective algorithm on weighted sum scalarizations and obtain several non-dominated solutions, one for each weight. Another general-purpose method based on scalarizations has been proposed by Borges [7]. This method uses the Tchebycheff distance, but not in the standard way explained earlier in this section. Instead, the goal for each single-objective problem is to find a new solution that maximizes the Tchebycheff distance between this new solution and the closest one in the archive. The idea is, as in TPLS, to obtain a set of well-spread solutions in the objective space.

Other methods have been proposed in the literature, more specific than the general idea behind TPLS. They are, however, general-purpose and can be applied to many different problems. These are extensions of single-objective metaheuristics to the multi-objective case. An example is Multi-Objective Tabu Search (MOTS), which is an extension of tabu search [31] to multi-objective problems proposed by Hansen [35]. MOTS keeps a set of non-dominated solutions and tries to improve each solution in a direction that moves its objective vector away from other non-dominated solutions. To do so, it updates the weights for a given solution based on all other non-dominated solutions (the closer solutions are, the higher is their mutual influence). The purpose of this behavior is to obtain a set of solutions as spread as possible in the objective space along the Pareto front. The optimization of solutions toward different directions is performed using tabu search principles, each solution dealing with its own tabu list.

There have been several adaptations of the Simulated Annealing (SA) principle to the multi-objective case. They usually use several runs of single-objective SA algorithms, and mainly differ by the acceptance rules of new solutions. The first SA for multi-objective algorithm, proposed by Serafini [58], uses the following acceptance criterion. If the new solution dominates the current one, this new solution is accepted to replace the current one. Otherwise, the acceptance probability is computed on a weighted sum of the objectives. Several runs are performed using different weight vectors, and some small random variations are applied to them each time a solution is considered. A similar method is MOSA, proposed by Ulungu et al. [61]. MOSA uses the same type of rule for the acceptance criterion and a similar set of predefined weight vectors to define the single-objective problems. However, MOSA is not only returning one solution per weight vector: every time a solution is accepted as the new current one, it is potentially inserted in a set of non-dominated solutions. Each run of the single-objective SA maintains its own set of non-dominated solutions, and the sets are merged and filtered in a last step. Suppapitnarm et al. [59] proposed another adaptation of the SA principle to the multi-objective case. The acceptance criterion is different from other SA adaptations. Their proposal uses a multiplicative function of the objectives, instead of a weighted sum, and a different *temperature* for each objective. The setting of the temperature does not follow a pre-scheduled decrease, but is automatically updated based on the variance of each objective among already accepted solutions. The algorithm is then restarted several times to provide several solutions.

Finally, various population-based methods such as evolutionary algorithms, have used scalarizations to direct the search towards the Pareto front. An early example is VEGA [57]; other examples include the algorithms proposed by Ishibuchi and Murata [37] and MOGLS of Jaszkiewicz [38]. Also ACO algorithms frequently use some form of scalarized aggregation, for example, for combining pheromone (or heuristic) information specific to each objective [5, 27, 46]. However, an overview of such population-based methods is beyond the scope of this chapter.

5 Hybridization of Search Paradigms

We have presented in Sections 3 and 4 two search paradigms for multi-objective optimization. Each of them has its particular advantages and drawbacks. Dominance-based algorithms can return quickly a large number of non-dominated solutions; however, they progress rather slowly towards the Pareto front and they may require a long computation time before reaching high-quality approximations to the Pareto front. Scalarization-based algorithms can exploit effective single-objective algorithms and they find quickly high-quality approximations to the Pareto front. However, they return only relatively few solutions and they may not be able to approximate well certain types of solutions. For example, heuristic algorithms based on weighted-sum scalarizations are not designed to identify non-supported solutions and, thus, they may leave "gaps" in the Pareto front approximation. Thus, combining both search paradigms can be profitable, in order to exploit their respective advantages and to avoid as much as possible their respective disadvantages.

Hybrid algorithms combining TPLS and PLS elements have been considered in the literature and have shown high performance. (Examples on the performance of such hybrids are given in Section 6.) The natural way of combining TPLS and PLS is to first use TPLS to generate a set S' of (few) non-dominated solutions that are a high-quality approximations to the Pareto front and then use the solutions in S' to seed PLS. In fact, such a hybrid TP+PLS algorithm is straightforwardly obtained from existing TPLS and PLS algorithms. A rudimentary form of a TP+PLS algorithm has been studied by Paquete and Stützle [53]; they use a restricted form of PLS, the component-wise step. Later, Lust and Teghem apply a TP+PLS algorithm that runs the PLS phase to completion [49]. More recently, Dubois-Lacoste et al. [15, 18] have presented applications of TP+PLS algorithms to bi-objective flow-shop problems and also considered the automatic configuration of TP+PLS [17]. In their applications, PLS is terminated based on a bound on the available computation time.

TP+PLS is an example of a sequential hybridization of algorithms from the dominance-based and the scalarization-based search paradigms. A second class of hybridizations considers iterative hybridizations where elements of the two search paradigms are alternately applied. In the following, we give a concise overview of some representative examples of sequential and iterative hybrids. For a more complete review, we refer the interested reader to [23].

5.1 Sequential Hybridization

Combining a scalarization-based and a dominance-based component by switching from one to the other is the most straightforward way of hybridizing the two search paradigms. This switching forms the basis of a *sequential* hybridization. A common usage of sequential hybrids is to first use an exact algorithm to solve scalarized problems to optimality and, thus, to provide some (or all) of the supported solutions. Then, in a second phase, a dominance-based component aims at finding some non-supported solutions. In the heuristic case, a scalarization-based components can provide a small set of high-quality solutions (not necessarily supported ones), and in a second step, a dominance-based component improves this set of solutions further. We describe next some representative examples of such sequential hybrids.

Hamacher and Ruhe [34] combined the two search paradigms to tackle the bi-objective minimum spanning tree problem. A sequence of scalarizations is solved to optimality in the first phase; this is well feasible given that the minimum spanning tree problem is polynomially solvable. In a second phase, the neighborhood of all solutions obtained from the scalarizations is explored to search for additional non-dominated solutions. Andersen et al. [2] proposed a similar approach. They tested restrictions that consider only solutions that are neighbors of *two* different solutions in the set, and show that it may be useful for large scale problems since the number of solutions to consider is small.

Ulungu and Tehgem [60] proposed the *two-phases method*. This is a scheme for exactly solving MCOPs that works as follows. In a first phase, the whole set of supported solutions is determined using weighted sum scalarizations defined by the dichotomic scheme of Aneja and Nair [3]. In a second phase, this set of supported solutions is used to provide bounds to algorithms such as branch & bound, to find all non-dominated solutions. Despite being developed for exact solving, this approach has also inspired developments for heuristic solvers [49, 19].

Gandibleux et al. [26] proposed an algorithm for the bi-objective assignment problem that combined the two search paradigms as follows. First, an exact algorithm finds several supported solutions (a polynomial-time algorithm is known for the scalarized problems), and then the set of solutions obtained is improved further by seeding with this set a dominance-based evolutionary algorithm, which is run for few iterations.

Parragh et al. [55] designed an hybrid algorithm to solve the multi-objective dial-a-ride problem. A variable-neighborhood search algorithm is used to tackle weighted sum scalarizations defined by a regular sequence of weight vectors. In a second, dominance-based phase, a path-relinking step is used to further improve the set of solutions.

Delorme et al. [11] combined a greedy randomized adaptive search procedure (GRASP) [24] with the strength Pareto evolutionary algorithm (SPEA) [67] to tackle the bi-objective set packing problem. GRASP is used to tackle a sequence of weighted sum scalarized problems, and then SPEA is used to improve further the set of solutions returned by the GRASP.

5.2 Iterative Hybridization

The second possibility is to use both search paradigms in an *iterative* way. In that case, typically a scalarization-based component is used for a specific step within a dominance-based algorithm. Such iterative algorithms are often implicit hybrids of the two search paradigms: researchers seek the best possible performance and include a scalarization-based component within a dominance-based algorithm (or vice-versa), without making explicit the general concept behind this combination.

Gandibleux et al. [25] proposed an algorithm called MOTS (not to be confused with the MOTS algorithm mentioned in the previous section) that uses the tabu search principle to push solutions towards local *ideal* points. Once a solution becomes the new current one in the tabu search process, its neighborhood is explored using Pareto dominance to add possible non-dominated solutions to the archive.

Czyzzak and Jaszkiewicz [9] proposed the Pareto Simulated Annealing (PSA). Several runs of a single-objective SA are performed *in parallel*. Each run of the single-objective SA is tackling a problem defined by a weight vector and when a new solution is accepted to be the current one, its neighborhood is explored to search for non-dominated solutions. The weight vectors that define the scalarized problems are updated to "escape" from the current solutions of the other runs (taking only the closest ones into account).

López-Ibáñez et al. [44] tested the combination of a tabu search algorithm with a multi-objective ACO, and another combination with an evolutionary algorithm (SPEA2 [65]). The ACO and the SPEA2 algorithms use the tabu search algorithm at each iteration to improve individual solutions by tackling scalarized problems defined from a regular sequence of weight vectors.

6 Experimental results of TP+PLS

In this section, we present some exemplary computational results with a TP+PLS algorithm for bi-objective flowshop scheduling problems and we give an overview of other recent experimental results. The flow-shop scheduling problem [39] is of high relevance since it models a common type of machine scheduling environment in industry. In the flow-shop scheduling problem, a set of n jobs is processed on m machines. The most common objective is to minimize the *makespan* (denoted by $C_{\max}$), that is, the completion time of the last job on the last machine. Other common objectives are the minimization of the *total completion time* (denoted by *SFT*) and the minimization of the *total tardiness*, denoted by *TT*, or the *total weighted tardiness*, denoted by *WT*. The bPFSP is NP-hard and it is one of the most widely studied scheduling problems. (To be more exact, minimizing makespan is NP-hard for three or more machines, minimizing the total completion time is NP-hard for two or more machines, and minimizing the total tardiness is NP-hard already for a single machine [29, 13].) For more details on this problem we refer to [50, 18].

In our research, we have developed a TP+PLS algorithm for five bi-objective permutation flowshop scheduling problems (bPFSP) [14, 15, 18] that correspond to all combinations of the above mentioned objectives with the exception of for the combination of *TT* and *WT*. We have carefully engineered the various algorithmic components of TPLS and PLS. The TPLS part of the algorithm exploits an underlying *iterated greedy algorithm* that is state-of-the-art for the PFSP with makespan minimization [56] and that has been adapted to tackle efficiently the other objectives and the weighted sum problems resulting from the scalarizations. The TPLS version that we used is the most recent one: AA-TPLS [16, 19]. We showed that its adaptive behavior yields better results than the classical TPLS even when used inside the hybrid algorithm since it can adapt the search to the shape of the Pareto front, which is rather irregular for bPFSPs. The PLS version that we used is the original one [51]. It uses two different types of neighborhood operators, which was found to be useful for the bPFSPs [14]. For more details on the TP+PLS framework for bPFSP, the interested reader can refer to [18].

The TP+PLS framework has been extensively compared to MOSA, a multi-objective SA proposed by Varadharajan and Rajendran [62]. MOSA was shown to outperform other algorithms on bPFSPs for several combinations of objectives [50]. The experimental comparison between MOSA and TP+PLS has been done based on a re-implementation of MOSA under equal computation times. Exemplary results are given in Table 1, which shows the percentage of runs where the output set of the TP+PLS algorithm weakly dominates in the Pareto sense (see Def. 7; hereafter we say that a set that weakly dominates another is "better") the output set obtained by a run of MOSA, and, conversely, the average percentage of runs that the output set of MOSA is better than TP+PLS. These percentages are computed for each instance over 625 pairwise comparisons obtained from 25 runs for each algorithm, and averaged over the 10 instances of each size. The results given are clearly in favor of TP+PLS. While TP+PLS is better than MOSA for a large percentage of the comparisons, the opposite is very rarely the case. Note that a value of 0 means that MOSA is not able to produce in any run a non-dominated set better than the worst non-dominated set produced by TP+PLS in any of the 25 runs of the 10 instances of a given size. The percentages given in Table 1 show that for small instances of 20 jobs, MOSA and the TP+PLS algorithm are difficult to compare. The low percentages are explained by the fact that both algorithms often find the same non-dominated set, which is probably the optimal Pareto front. For these small instances, differences are not consistent across instances and combinations of objectives, and it cannot be said that any algorithm is clearly better than the other. Nevertheless, for all the remaining instances, Table 1 shows that TP+PLS clearly outperforms MOSA.

Despite the fact that the TP+PLS algorithm often dominates MOSA, it is interesting to study how different the Pareto front approximations provided by the two algorithms are. To do so we use a graphical tool, the empirical attainment function (EAF). The EAF gives the, empirically estimated, probability that an algorithm dominates an arbitrary area of the objective space [33]. By plotting the differences of the EAFs of two algorithms, one can graphically show where in the objective space, and how frequently, an algorithm performs better relative to the other. A

Table 1 For each bi-objective problem (denoted by the two objectives in parenthesis), the left column shows the percentage of runs (computed over 25 runs per instance and averaged over 10 instances of the same size) in which an output set obtained by TP+PLS is better in the Pareto sense than an output set obtained by MOSA. The right column shows the percentage of runs for which an output set of MOSA is better than an output set of TP+PLS .

	PFSP-($C_{\max}$, *SFT*)		*PFSP*-($C_{\max}$, *TT*)		*PFSP*-($C_{\max}$, *WT*)		*PFSP*-(*SFT*, *TT*)		*PFSP*-(*SFT*, *WT*)	
nxm	TP+PLS	MOSA	TP+PLS	MOSA	TP+PLS	MOSA	TP+PLS	MOSA	TP+PLS	MOSA
20x5	4.66	5.83	6.1	1.34	14.95	0.18	10.19	26.31	0.02	20.15
20x10	1.87	9.2	0.07	0.26	0.02	0.06	0.19	0.63	0.03	0.07
20x20	0.13	1.23	1.27	1.57	1.99	2.32	3.63	5.55	4.2	10.09
50x5	89.49	0	84.33	0	79.22	0	98.13	0.08	33.67	0
50x10	72.92	0	63.17	0	63.24	0	94.07	0	20.53	0
50x20	75.94	0	61.11	0	63.01	0	5.79	0	14.72	0
100x5	84.97	0	70.5	0	67.12	0	93.66	2.54	9.72	0
100x10	76.94	0.05	69.86	0	37.49	0	95.38	0.58	16.84	0
100x20	73.17	0	63.29	0	23.81	0	97.35	0	15.31	0
200x10	18.04	0.16	24.5	0	4.15	0	91.77	3.72	0.02	0
200x20	15.16	0	37.83	0	0.25	0	78.23	6.28	1.04	0.02

more detailed explanation of this graphical tool can be found in [45]. Fig. 1 shows the differences between the EAFs obtained by the hybrid TP+PLS algorithm and the MOSA algorithm, for one instance with makespan and total completion time minimization. A large gap is observed in favor of the TP+PLS algorithm over the one obtained by MOSA, indicating a clearly superior performance. Other instances and combinations of objectives show similar trends.

Table 1, Fig. 1 and additional results available in [18] show that the hybrid TP+PLS algorithm clearly outperforms the previously best-known algorithm, by a large gap, and for all combinations of makespan minimization, total completion time minimization, total and weighted tardiness minimization.

TP+PLS algorithms were also applied to the bi-objective traveling salesman problem (bTSP). The TSP and the bi-objective version of it are well-known NP-hard combinatorial problems widely used to assess the performance of optimization algorithms and metaheuristics [22, 54]. The goal in the bTSP is to find a Hamiltonian tour that minimizes the sum of the edge costs in the tour. In the bi-objective variant of the TSP, two cost values are assigned to each edge of a graph, and each of the two objective functions is computed with respect to the corresponding cost value.

Paquete and Stützle [53] proposed the Pareto double two-phase local search (PD-TPLS method) and applied it to the bTSP. In this sequential hybrid algorithm, the first phase uses the TPLS method (in its original form, that is, with a regular sequence of weights) to return a set of non-dominated solutions, and in the second phase the neighborhood of all these solutions is explored to find additional non-dominated solutions. In a sense, this so-called component-wise step implements a restricted version of PLS. The single-objective algorithm used to tackle the scalarized problems is an iterated local search algorithm, and the neighborhood operator

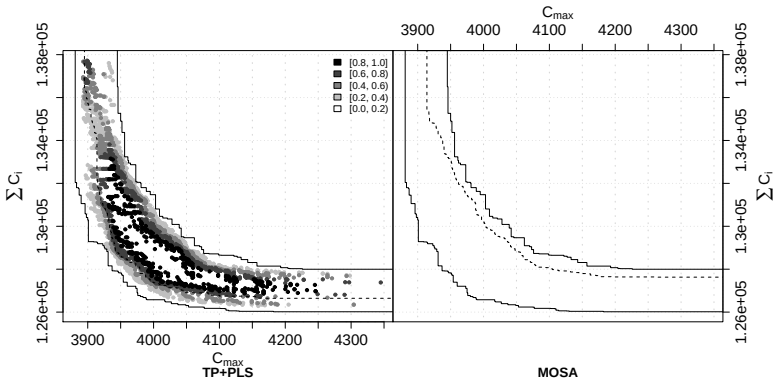

Fig. 1 Differences of the empirical attainment functions obtained over 25 runs in favor of the hybrid TP+PLS algorithm (left) and in favor of the MOSA algorithm (right), for the bPFSP with minimization of the makespan and of the total completion time, on an instance with 50 jobs and 20 machines.

is based on 2-opt moves. This hybrid algorithm has been compared favorably to the best algorithm known at that time, the MOGLS algorithm from Jaszkiewicz [38]. A more in-depth experimental study of PD-TPLS and other TPLS variants has been presented by Paquete and Stützle [54].

More recently, another TP+PLS algorithm has been applied to the bTSP by Lust and Teghem [49]. This algorithm is reported to outperform the PD-TPLS of Paquete and Stützle. The main differences are that (i) the single-objective algorithm used to tackle the scalarized problems is an effective implementation of the chained Lin-Kernighan heuristic [6], which is presumably more effective than the iterated local search algorithm used by the PD-TPLS, and (ii) a full version of PLS is used (more precisely, the version of Angel et al. [4]) instead of the restricted one in the PD-TPLS method. This algorithm is nowadays, to the best of our knowledge, the state of the art for the bTSP.

Finally, Lust and Teghem [48] tackled multi-objective multi-dimensional knapsack problems. In these problems, two or more types of profits are associated to each item, each type of profit representing a different objective, and the goal is to determine a subset of items to place in a knapsack that maximizes, in the Pareto sense, the sum of profits for each objective. In the multi-dimensional version, the knapsack has more than one capacity constraint, and a feasible solution needs to satisfy all capacity constraints. The TPLS phase of the algorithm finds solutions for each weighted sum scalarization using a greedy constructive heuristic. In the second phase, PLS uses a very-large neighborhood search that starts from the set of solutions obtained by TPLS. They consider the bi-objective variant using regularly distributed weights for the scalarizations, and a three-objective variant using random weights for the scalarizations. For the bi-objective multi-dimensional knapsack problem the proposed TP+PLS algorithm is a new state-of-the-art approach, clearly outperforming its competitors on widely tested benchmark instances.

7 Conclusion

In this chapter, we have reviewed heuristic methods for solving multi-objective combinatorial optimization problems in the Pareto sense. In particular, we reviewed methods rooted in the scalarization-based and the dominance-based search paradigms. For each of the two search paradigms, we presented a representative method, namely Pareto local search (PLS) as a dominance-based method and two-phase local search (TPLS) as a scalarization-based method. We have detailed the main algorithmic components that are required to define PLS and TPLS algorithms, and discussed how PLS and TPLS can be combined into hybrid TP+PLS algorithms. Some exemplary computational results have illustrated the high potential of TP+PLS algorithms. In fact, for several well-known multi-objective problems, including various bi-objective flowshop problems, the traveling salesman problem with two and three objectives and the bi-objective knapsack problem, TP+PLS algorithms are currently state of the art.

An important direction for future research is certainly to extend the current generation of TPLS, PLS and TP+PLS algorithms to three and more objectives. We also believe that an iterative hybridization instead of a sequential one could be beneficial in terms of flexibility and performance. The TP+PLS framework is also a clear candidate for the automatic configuration of high-performing multi-objective algorithms. The feasibility of such an endeavor has already been proven [17]. Regarding applications, we believe the hybrid TP+PLS framework could be applied to other well-known and widely studied problems, providing high-quality results without requiring a large implementation effort.

Acknowledgments.

This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium, and by the MIBISOC network, an Initial Training Network funded by the European Commission, grant PITN–GA–2009–238819. Manuel López-Ibáñez and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a postdoctoral researcher and a Research Associate, respectively. The authors also acknowledge support from the FRFC project "*Méthodes de recherche hybrides pour la résolution de problèmes complexes*".

References

1. Alsheddy, A., Tsang, E.: Guided Pareto local search and its application to the 0/1 multi-objective knapsack problems. In: Caserta, M., Voß, S. (eds.) Proceedings of MIC 2009, the 8th Metaheuristics International Conference. University of Hamburg, Hamburg, Germany (2010)
2. Andersen, K., Jörnsten, K., Lind, M.: On bicriterion minimal spanning trees: An approximation. Computers & Operations Research 23(12), 1171–1182 (1996)

3. Aneja, Y.P., Nair, K.P.K.: Bicriteria transportation problem. Management Science 25(1), 73–78 (1979)
4. Angel, E.: Approximating the Pareto curve with local search for the bicriteria TSP(1,2) problem. Theoretical Computer Science 310(1-3), 135–146 (2004)
5. Angus, D., Woodward, C.: Multiple objective ant colony optimization. Swarm Intelligence 3(1), 69–85 (2009)
6. Applegate, D., Cook, W., Rohe, A.: Chained Lin-Kernighan for large traveling salesman problems. INFORMS Journal on Computing 15(1), 82–92 (2003)
7. Borges, P.C.: CHESS - changing horizon efficient set search: A simple principle for multiobjective optimization. Journal of Heuristics 6(3), 405–418 (2000)
8. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York, NY (2007)
9. Czyżżak, P., Jaszkiewicz, A.: Pareto simulated annealing – a metaheuristic technique for multipleobjective combinatorial optimization. Journal of Multi-Criteria Decision Analysis 7(1), 34–47 (1998)
10. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Chichester, UK (2001)
11. Delorme, X., Gandibleux, X., Degoutin, F.: Evolutionary, constructive and hybrid procedures for the bi-objective set packing problem. European Journal of Operational Research 204(2), 206–217 (2010)
12. Drugan, M.M., Thierens, D.: Path-guided mutation for stochastic Pareto local search algorithms. In: Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G. (eds.) Parallel Problem Solving from Nature, PPSN XI, Lecture Notes in Computer Science, vol. 6238, pp. 485–495. Springer, Heidelberg, Germany (2010)
13. Du, J., Leung, J.Y.T.: Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 15(3), 483–495 (1990)
14. Dubois-Lacoste, J.: A study of Pareto and Two-Phase Local Search Algorithms for Biobjective Permutation Flowshop Scheduling. Master's thesis, IRIDIA, Université Libre de Bruxelles, Belgium (2009)
15. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Effective hybrid stochastic local search algorithms for biobjective permutation flowshop scheduling. In: Blesa, M.J., Blum, C., Di Gaspero, L., Roli, A., Sampels, M., Schaerf, A. (eds.) Hybrid Metaheuristics, Lecture Notes in Computer Science, vol. 5818, pp. 100–114. Springer, Heidelberg, Germany (2009)
16. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Adaptive "anytime" two-phase local search. In: Blum, C., Battiti, R. (eds.) Learning and Intelligent Optimization, 4th International Conference, LION 4, Lecture Notes in Computer Science, vol. 6073, pp. 52–67. Springer, Heidelberg, Germany (2010)
17. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Automatic configuration of state-of-the-art multi-objective optimizers using the TP+PLS framework. In: Krasnogor, N., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 2019–2026. ACM press, New York, NY (2011)
18. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 38(8), 1219–1236 (2011)
19. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Improving the anytime behavior of two-phase local search. Annals of Mathematics and Artificial Intelligence 61(2), 125–154 (2011)
20. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Pareto local search algorithms for anytime bi-objective optimization. In: Hao, J.K., Middendorf, M. (eds.) Proceedings of EvoCOP 2012 – 12th European Conference on Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol. 7245, pp. 206–217. Springer, Heidelberg, Germany (2012)
21. Ehrgott, M.: Multicriteria optimization, Lecture Notes in Economics and Mathematical Systems, vol. 491. Springer, Berlin, Germany (2000)
22. Ehrgott, M., Gandibleux, X.: Approximative solution methods for combinatorial multicriteria optimization. TOP 12(1), 1–88 (2004)

23. Ehrgott, M., Gandibleux, X.: Hybrid metaheuristics for multi-objective combinatorial optimization. In: Blum, C., Blesa, M.J., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics: An emergent approach for optimization, pp. 221–259. Springer, Berlin, Germany (2008)
24. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–113 (1995)
25. Gandibleux, X., Mezdaoui, N., Fréville, A.: A tabu search procedure to solve multiobjective combinatorial optimization problem. In: Caballero, R., Ruiz, F., Steuer, R. (eds.) Advances in Multiple Objective and Goal Programming. Lecture Notes in Economics and Mathematical Systems, vol. 455, pp. 291–300. Springer, Heidelberg, Germany (1997)
26. Gandibleux, X., Morita, H., Katoh, N.: Use of a genetic heritage for solving the assignment problem with two objectives. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) Evolutionary Multi-criterion Optimization (EMO 2003), Lecture Notes in Computer Science, vol. 2632, pp. 43–57. Springer, Heidelberg, Germany (2003)
27. García-Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. European Journal of Operational Research 180(1), 116–148 (2007)
28. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman&Co, San Francisco, CA (1979)
29. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129 (1976)
30. Geiger, M.J.: Decision support for multi-objective flow shop scheduling by the Pareto iterated local search methodology. Computers and Industrial Engineering 61(3), 805–812 (2011)
31. Glover, F.: Tabu search – Part I. INFORMS Journal on Computing 1(3), 190–206 (1989)
32. Glover, F.: A template for scatter search and path relinking. In: Artificial Evolution, pp. 1–51. Lecture Notes in Computer Science, Springer, Heidelberg, Germany (1998)
33. Grunert da Fonseca, V., Fonseca, C.M., Hall, A.O.: Inferential performance assessment of stochastic optimisers and the attainment function. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D. (eds.) Evolutionary Multi-criterion Optimization (EMO 2001), Lecture Notes in Computer Science, vol. 1993, pp. 213–225. Springer, Heidelberg, Germany (2001)
34. Hamacher, H.W., Ruhe, G.: On spanning tree problems with multiple objectives. Annals of Operations Research 52(4), 209–230 (1994)
35. Hansen, M.P.: Tabu search for multiobjective optimization: MOTS. In: Climaco, J. (ed.) Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM97), pp. 574–586. Springer Verlag (1997)
36. Hansen, P., Mladenovic, N.: Variable neighborhood search: Principles and applications. European Journal of Operational Research 130(3), 449–467 (2001)
37. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics – Part C 28(3), 392–403 (1998)
38. Jaszkiewicz, A.: Genetic local search for multi-objective combinatorial optimization. European Journal of Operational Research 137(1), 50–71 (2002)
39. Johnson, D.S.: Optimal two- and three-stage production scheduling with setup times included. Naval Research Logistics Quarterly 1, 61–68 (1954)
40. Knowles, J.D., Corne, D.: The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), pp. 98–105. IEEE Press, Piscataway, NJ (1999)
41. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. IEEE Transactions on Evolutionary Computation 8(2), 170–182 (2004)
42. Liefooghe, A., Humeau, J., Mesmoudi, S., Jourdan, L., Talbi, E.G.: On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. Journal of Heuristics 18(2), 317–352 (2011)

43. Liefooghe, A., Mesmoudi, S., Humeau, J., Jourdan, L., Talbi, E.G.: A study on dominance-based local search approaches for multiobjective combinatorial optimization. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics. SLS 2009, Lecture Notes in Computer Science, vol. 5752, pp. 120–124. Springer, Heidelberg, Germany (2009)
44. López-Ibáñez, M., Paquete, L., Stützle, T.: Hybrid population-based algorithms for the bi-objective quadratic assignment problem. Journal of Mathematical Modelling and Algorithms 5(1), 111–137 (2006)
45. López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Experimental Methods for the Analysis of Optimization Algorithms, pp. 209–222. Springer, Berlin, Germany (2010)
46. López-Ibáñez, M., Stützle, T.: The automatic design of multi-objective ant colony optimization algorithms. IEEE Transactions on Evolutionary Computation (2012), accepted
47. Lourenço, H.R., Martin, O., Stützle, T.: Iterated local search: Framework and applications. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 146, chap. 9, pp. 363–397. Springer, New York, NY, 2 edn. (2010)
48. Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. Arxiv preprint arXiv:1007.4063 (2010)
49. Lust, T., Teghem, J.: Two-phase Pareto local search for the biobjective traveling salesman problem. Journal of Heuristics 16(3), 475–510 (2010)
50. Minella, G., Ruiz, R., Ciavotta, M.: A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS Journal on Computing 20(3), 451–471 (2008)
51. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Gandibleux, X., et al. (eds.) Metaheuristics for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 177–200. Springer (2004)
52. Paquete, L., Schiavinotto, T., Stützle, T.: On local optima in multiobjective combinatorial optimization problems. Annals of Operations Research 156, 83–98 (2007)
53. Paquete, L., Stützle, T.: A two-phase local search for the biobjective traveling salesman problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) Evolutionary Multi-criterion Optimization (EMO 2003), Lecture Notes in Computer Science, vol. 2632, pp. 479–493. Springer, Heidelberg, Germany (2003)
54. Paquete, L., Stützle, T.: Design and analysis of stochastic local search for the multiobjective traveling salesman problem. Computers & Operations Research 36(9), 2619–2631 (2009)
55. Parragh, S., Doerner, K.F., Hartl, R.F., Gandibleux, X.: A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. Networks 54(4), 227–242 (2009)
56. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177(3), 2033–2049 (2007)
57. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette, J.J. (ed.) ICGA-85. pp. 93–100. Lawrence Erlbaum Associates (1985)
58. Serafini, P.: Simulated annealing for multiple objective optimization problems. In: Tzeng, G.H., Yu, P.L. (eds.) Proceedings of the 10th International Conference on Multiple Criteria Decision Making (MCDM91). vol. 1, pp. 87–96. Springer Verlag (1992)
59. Suppapitnarm, A., Seffen, K., Parks, G., Clarkson, P.: A simulated annealing algorithm for multiobjective optimization. Engineering Optimization 33(1), 59–85 (2000)
60. Ulungu, E., Teghem, J.: The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. Foundations of Computing and Decision Sciences 20(2), 149–165 (1995)
61. Ulungu, E., Teghem, J., Fortemps, P., Tuyttens, D.: MOSA method: a tool for solving multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis 8(4), 221–236 (1999)

62. Varadharajan, T.K., Rajendran, C.: A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. European Journal of Operational Research 167(3), 772–795 (2005)
63. Voudouris, C., Tsang, E.: Guided local search and its application to the travelling salesman problem. European Journal of Operational Research 113(2), 469–499 (1999)
64. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996)
65. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K., Tsahalis, D., Periaux, J., Papaliliou, K., Fogarty, T. (eds.) Evolutionary Methods for Design, Optimisation and Control. pp. 95–100. CIMNE, Barcelona, Spain (2002)
66. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben, A.E., et al. (eds.) Parallel Problem Solving from Nature, PPSN V. Lecture Notes in Computer Science, vol. 1498, pp. 292–301. Springer, Heidelberg, Germany (1998)
67. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)