Journal of Heuristics manuscript No.
(will be inserted by the editor)

Survey and Unification of Local Search Techniques in Metaheuristics for Multi-objective Combinatorial Optimisation

Aymeric Blot · Marie-Éléonore Kessaci · Laetitia Jourdan

Received: date / Accepted: date

Abstract Metaheuristics are algorithms that have proven their efficiency on multi-objective combinatorial optimisation problems. They often use local search techniques, either at their core or as intensification mechanisms, to obtain a well-converged and diversified final result. This paper surveys the use of local search techniques in multi-objective metaheuristics and proposes a general structure to describe and unify their underlying components. This structure can instantiate most of the multi-objective local search techniques and algorithms in literature.

Keywords Multi-objective optimisation · Combinatorial optimisation · Metaheuristics · Unification · Local search algorithms

1 Introduction

Metaheuristics are widely used algorithms for solving large and complex multi-objective optimisation problems (Gendreau and Potvin, 2010). Indeed, most of such problems are $\mathcal{NP}$-hard and require approximation mechanisms to obtain good solutions in a reasonable time. A common point of many multi-objective metaheuristics is the use of local search techniques, which are either hybridised inside of an existing algorithm or used as an essential building part of it.

A. Blot
Université de Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, Cité Scientifique, 59655 Villeneuve d'Ascq, France
E-mail: aymeric.blot@univ-lille.fr
ORCID: 0000-0003-0485-5279

M-É. Kessaci
E-mail: mkessaci@univ-lille.fr
ORCID: 0000-0002-4372-5162

L. Jourdan
E-mail: laetitia.jourdan@univ-lille.fr
ORCID: 0000-0002-4170-6830

Local search algorithms are originally single-objective algorithms very effective on hard combinatorial optimisation problems (Hoos and Stützle, 2004). Although they have been sometimes directly used as single-objective procedures of multi-objective algorithms, many Multi-Objective Local Search (MOLS) algorithms have also been specifically designed. In this paper, we focus on these MOLS techniques. Two types of MOLS algorithms can be distinguished: the direct extensions of well-known single-objective local search algorithms and those that have been designed together with evolutionary algorithms. Both types of local search algorithms nevertheless share a common underlying structure. Many generalisations of MOLS algorithms have already been proposed, such as the Dominance-based Multi-objective Local Search (DMLS) generalisation based on several Pareto Local Search (PLS) strategies by Liefooghe et al (2012). However, as far as we know, no unification of both types of MOLS algorithms has been carried out.

In this paper, we will propose a new basic local search structure that unifies most, if not all, of the MOLS algorithms in the literature and will discuss a number of strategies that can be integrated inside this structure. We will also carry out a chronological survey on the development of MOLS algorithms and will detail how the main MOLS algorithms in the literature are instantiated in our unification.

The rest of the paper is organised as follows. Section 2 reviews the general notions regarding multi-objective optimisation and local search mechanisms. Section 3 presents a survey of the use of local search techniques in multi-objective algorithms. Section 4 presents the keystones of our unification. Section 5 describes our unification of MOLS structure and gives a detailed instantiation of the most prominent MOLS algorithms. Finally, Section 6 concludes this paper.

2 Preliminaries

In this section, we give the notions required to understand the following topics of the paper. First, definitions of multi-objective optimisation are given. Then, local search algorithms are briefly introduced together with the concept of neighbourhood.

2.1 Multi-objective Combinatorial Optimisation

Optimisation problems consist in finding optimal solutions within a space of possible solutions. In this paper, we focus on multi-objective optimisation problems, in which the quality of solutions is compared using multiple criteria (or objective functions) to be optimised. Formally, let X be the *search space* (i.e., the space of all possible solutions) and let $F = \{f_1, \ldots, f_n\}$ be a vector of n objective functions. A multi-objective combinatorial optimisation

problem can then be defined as follows:

$$\begin{array}{ll} \text{optimise} & F(x) = \{f_1(x), \ldots, f_n(x)\} \\ \text{subject to} & x \in X. \end{array} \quad (1)$$

Each of the n objective functions is, in regard to the problem, to be minimised or maximised. In the following, we suppose, without loss of generality, that they are all to be minimised.

For the comparison of two solutions s_1 and s_2, the Pareto dominance is often used: s_1 is said to *dominate* s_2 (denoted by $s_1 \succ s_2$) if and only if s_1 is better than or equal to s_2 according to all criteria and if there exists at least one criterion according to which s_1 is strictly better than s_2:

$$s_1 \succ s_2 \iff \forall 1 \leq k \leq n, f_k(s_1) \leq f_k(s_2) \wedge \exists 1 \leq k \leq n, f_k(s_1) < f_k(s_2) \quad (2)$$

Two mutually non-dominated solutions are considered *incomparable*. A solution that is not dominated by any other possible solution is a *global* or a *Pareto optimum*. A set of mutually incomparable solutions defines a *Pareto set*. Finally, the *Pareto optimal set* is defined as the (Pareto) set of all Pareto optima.

In addition or for the replacement of the Pareto dominance, various alternatives can also be used such as an aggregation of the different criteria (e.g., Serafini (1994)), multi-objective indicators (e.g., Zitzler and Künzli (2004)), a lexicographical ordering or the use of weakened definitions of the Pareto dominance (e.g., Laumanns et al (2002)). Common scalarising functions include, but are not limited to, the simple weighted sum ($\sum_{k=1}^{n} \lambda_k f_k(x)$) or the weighted Chebyshev norm ($\max_{1 \leq k \leq n}\{\lambda_k | f_k(x) - f_k(z)|\}$, where z is a reference point, e.g., the ideal point).

2.2 Local Search Algorithms

In this paper, we focus on combinatorial optimisation problems in which the search space is a *finite set* of solutions. For solving $\mathcal{NP}$-hard problems, exhaustive numeration of the search space is generally not feasible and, therefore, local search algorithms that exploit the structure of the search space to iteratively find better and better solutions are used. The idea is that small modifications in the representation of a solution may lead to either a small improvement or a small deterioration of its initial quality. The notion of *neighbourhood* is defined from this idea. A *neighbourhood operator* is a function that modifies part of a given solution that produces a new solution called a *neighbour*. The set of neighbours that can be generated from a given solution x defines the neighbourhood $\mathcal{N}(x)$ of x. This concept of neighbourhood gives a structure to the search space by connecting close solutions.

Several definitions result from this concept. When the neighbourhood $\mathcal{N}(x)$ of a solution x contains no improving neighbour, x then constitutes a *local optimum*. Note that a solution x can be a local optimum for a neighbourhood $\mathcal{N}_1$ without being a local optimum for another neighbourhood $\mathcal{N}_2$. Furthermore,

there is no guarantee for a local optimum to be a global optimum, and it is generally not one. On the contrary, a global optimum is always a local optimum regardless of the neighbourhood considered since it is of the best possible quality.

Local search algorithms iteratively use neighbourhood operators to reach better and better solutions. However, local optima are fundamentally detrimental to local search algorithms as there is usually no means to distinguish between local and global optima. Thus, to continue the search after having converged to a local optima, researchers have proposed multiple techniques, including either performing multiple random moves over the search space called kicks (e.g., Iterated Local Search, Lourenço et al (2010)), accepting to move temporarily to worse solutions (e.g., Simulated Annealing, Kirkpatrick et al (1983), doing a Tabu Search, Glover et al (1993)) or switching between several neighbourhood structures (e.g., variable neighbourhood search, Mladenović and Hansen (1997)).

In a multi-objective combinatorial optimisation context, this notion is extended by defining *Pareto Local Optima* (PLO) as the solutions whose neighbourhood contains no dominating neighbour, i.e., the solutions whose every neighbour is either dominated or incomparable. Likewise, there is no guarantee for a PLO to be a Pareto optimum, although Pareto optima are always PLO regardless of the neighbourhood considered. Furthermore, many, if not all, of the above techniques used to escape local optima in a single-objective context have been extended to multi-objective ones to deal with PLO (see Section 3.1).

3 Development of Multi-Objective Local Search Algorithms

Historically, local search algorithms have been initially designed to solve single-objective combinatorial optimisation problems and, therefore, are themselves single-objective algorithms (Hoos and Stützle, 2004). Multi-objective local search (MOLS) algorithms are used on the same combinatorial problems, e.g., multi-objective travelling salesman problems, multi-objective scheduling problems (Jaszkiewicz, 2002; Basseur and Burke, 2007; Liefooghe et al, 2012; Dubois-Lacoste et al, 2015), and bioinformatics problems (Abbasi et al, 2015). The majority of the literature works focuses on bi-objective and tri-objective problems, while very fewer works tackle more than three objectives simultaneously. This is due to the nature of the induced search space; indeed, in these *many-objective* problems (Ishibuchi et al, 2008) solutions are much more often incomparable to each others, thus majorly hindering the neighbourhood exploration of MOLS algorithms.

The development of MOLS algorithms has occurred simultaneously from two different directions. On the one hand, they were directly extended from well-known and established single-objective algorithms (e.g., Serafini (1994); Ulungu et al (1995); Czyzak and Jaszkiewicz (1996); Hansen (1997)). On the other hand, they were either integrated into evolutionary algorithms as inner components or used as post-processing algorithms (e.g., Ishibuchi and Murata

(1996); Knowles and Corne (1999); Talbi et al (2001)). Nowadays, the prominent MOLS algorithms in the literature have grown into the PLS algorithms, which are derived from the second type of MOLS algorithms.

In the following, we detail chronologically the development of these two algorithmic families before summarising their common characteristics.

3.1 Extensions of Single-Objective Search Algorithms

Since local search algorithms have been originally designed for single-objective optimisation, they are single-trajectory algorithms, meaning that they follow a single solution (i.e., the *current* solution). Unsurprisingly, the first MOLS algorithms were extensions of these single-objective local search algorithms.

Simulated Annealing (SA) (Kirkpatrick et al, 1983) is a local search procedure that optimises a single solution, using a decreasing parameter, the temperature, to slowly converge to the global optimal solution. Serafini (1994), Fortemps et al (1994) and Ulungu et al (1995, 1999) have independently proposed the same algorithm, *Multi-Objective Simulated Annealing* (MOSA). Like in the original single-objective algorithm, a single *current* solution is considered and moved through the search space, while a subsidiary set is used to store the potential Pareto optimal solutions. The current solution is updated by evaluating a single random neighbour and potentially accepting it with regard to rules based on probabilities, which are themselves based on whether the neighbour dominates, is dominated by or is incomparable to the current solution, and on weighted projections of the fitness function. Czyzak and Jaszkiewicz (1996, 1998) proposed the *Pareto Simulated Annealing* (PSA), in which, rather than a single current solution, a set of current solutions is used to converge into multiple optima at the same time. The diversity of having multiple current solutions is also used to guide the parallel searches in diverse directions. Engrand (1998) and then Suppapitnarm and Parks (1999) proposed another MOSA variants, in which the current solution is periodically replaced by one of the archived solutions (SMOSA). Other variants also include the *Pareto Archived Simulated Annealing* (PASA) by Suresh and Mohanasundaram (2004), the *Archived Multi-Objective Simulated Annealing* (AMOSA) by Bandyopadhyay et al (2008) and those based on both Pareto Dominance (PDMOSA) and Weights (WMOSA) proposed in literature reviews by Suman (2003); Suman and Kumar (2006).

Tabu Search (TS) (Glover et al, 1993) is a local search algorithm that uses an auxiliary set of solutions, the *tabu list*, to guide the search and escape local optima by preventing a *backward* move on the search space by banning the acceptance of neighbours too similar to recent considered solutions. In a TS local search, when a solution is explored, each of the solution's neighbours is evaluated and the best non-tabu one is selected to replace the current solution, even if it has a worse quality. The first multi-objective algorithm based on TS is the *Multi-Objective Tabu Search* (MOTS) proposed by Hansen (1997). It uses a set of *current* solutions and independently explores their neighbour-

hoods using an aggregation of the multiple objectives. After a given number of iterations, a *drift* strategy is applied: the set of solutions is updated by replacing one of the solutions by another one, both uniformly selected at random in it, to explore the whole front and not merely to focus on one part of the objective space. Other TS algorithms have been proposed, such as the one by Baykasoglu et al (1999), which is based on a local search with an intensification memory to restart from when no more improving move is available, or the two algorithms proposed by Jaeggi et al (2004, 2008), based on the Hooke and Jeeves move (MOTS) and path-relinking (PRMOTS), respectively. Multi-objective variants of scatter search using TS and path-relinking have also been proposed (Beausoleil, 2001; Molina et al, 2007).

Greedy Randomised Adaptive Search Procedure for Maximum Independent Set (GRASP) is a procedure originally presented by Feo et al (1994) for single-objective optimisation problems. It has been extended by Vianna and Arroyo (2004) for multi-objective optimisation problems (GRASP-MULTI). Both algorithms are multi-start metaheuristics that alternate two phases, the first being the construction of an initial solution, and the second being the iterative improvement of that solution through a local search procedure; the best overall solution (or set of solutions, for the multi-objective version) is then returned. The local search procedure simply replaces the current solution by any improving neighbour until there is no more improving neighbour. In the case of the multi-objective version, the different local search iterations use different aggregation weights and a list of all potential Pareto optimal solutions is automatically kept up to date. An extensive survey of multi-objective GRASP can be found in Martí et al (2015), in which the authors propose a multi-objective GRASP with path-relinking.

Lastly, Variable Neighbourhood Search (VNS) (Mladenović and Hansen, 1997) is a local search algorithm that solves the *local* optima problem by simply considering multiple other neighbourhoods. Indeed, a PLO regarding a given neighbourhood may have dominating neighbours regarding other neighbourhoods. Geiger (2008) proposed a Multi-Objective Variable Neighbourhood Search (MOVNS) based on PLS-2 and the VNS methodology. Like in PLS, an archive is used to store all potential Pareto optimal solutions. In every iteration, both a solution and a neighbourhood, if they have not been explored yet, are selected uniformly at random, and then the solution is entirely explored and the Pareto set is updated using all the neighbours. Arroyo et al (2011) proposed an interesting alternative to the MOVNS algorithm by adding a *shaking* mechanism: instead of generating the neighbourhood of a solution of the Pareto set, their algorithm generates the neighbourhood of a neighbour of the solution of the Pareto set.

3.2 Local Search Techniques in Evolutionary Algorithms

In addition to the single-objective local search algorithms, the development of MOLS algorithms also occurred at the same time jointly with the multi-objective evolutionary algorithms.

Evolutionary Algorithms (EAs) constitute a class of metaheuristics based on the iterative improvement of a set of solutions (namely, the *population*), which is often used to tackle multi-objective optimisation problems. EAs usually iterate both crossover and mutation techniques to improve the population. There are two types of hybridisation of multi-objective EAs with local search mechanisms. The first type integrates the local search inside the EA, either complementing or replacing the mutation, by using a local search on every solution of the population at each iteration. The local search is then generally a single-objective algorithm, based on an aggregation of the different objectives. The second type uses a MOLS as a post-processing procedure at the end of the EA.

Ishibuchi and Murata (1996) first proposed the *Multi-Objective Genetic Local Search* (MOGLS), which hybridises a genetic algorithm with a single-objective local search by performing a local search on every solution generated at every iteration. During the local search, at most k neighbours of the current solution are produced and any improving neighbour is accepted. The crossover and the mutation strategies generate new solutions where the local search is applied using a new aggregation, i.e., the weights are randomly chosen. A *cellular* variant, called C-MOGLS, was proposed by Murata et al (2000), which divides solutions into cells associated with weight vectors to guide the selection and local search procedures of the MOGLS. Jaszkiewicz (2002) proposed another Genetic Local Search (GLS) where the main differences are the way the solutions are selected for recombination and the local search aggregation, which uses a weight vector selected at random from a set of possible weight vectors. Knowles and Corne (1999, 2000a) proposed the *Pareto Archived Evolutionary Strategy* (PAES), an EA without crossover that only relies on local search techniques. PAES was presented as "the simplest non-trivial approach to a multi-objective local search procedure", and three versions were introduced. All versions maintain a single *current* solution to be explored, while the population takes the role of a Pareto set. In the simple (1+1)-PAES, the current solution is explored by generating a single neighbour. The neighbour replaces the current solution either if it dominates the latter or if it is in a less crowded region of the population. The population itself is updated in such a way that any dominated solution is discarded and the solutions of less crowded spaces replace the solutions of more crowded spaces. The (1+λ)-PAES variant generates λ neighbours of the current solution at every iteration, which are all considered for updating the population and replacing the current solution. The (μ+λ)-PAES variant replaces the current solution with a list of μ current solutions, one of which is explored, selected using a binary tournament. Knowles and Corne (2000b) also proposed the memetic-PAES (M-PAES), a variant periodically employing a crossover.

Talbi et al (2001) also proposed a genetic algorithm hybridised with a MOLS procedure. The execution of the algorithm is divided into two separate steps, beginning with the execution of a genetic algorithm. Once the GA finishes, every solution of the final population is then explored by generating and archiving every possible non-dominated neighbour, a procedure that is then iterated until the end of the algorithm.

Similarly, Moslehi and Mahnam (2011) proposed a hybridisation of a *Multi-Objective Particle Swarm Optimisation* algorithm with a single-objective local search (MOPSO+LS).

3.3 Pareto Local Search Algorithms

Following the steps of algorithms such as PAES (Knowles and Corne, 1999) and PLS-2 (Talbi et al, 2001), which are based on Pareto dominance and use the population of the EA as a Pareto set, many more algorithms solely based on local search techniques have been designed that are not simple extensions of known single-objective local search algorithms. These simple extensions can still be seen as either single-trajectory algorithms or *multiple*-trajectory algorithms, as multiple solutions are simultaneously iteratively improved. On the contrary, the notion of trajectory is more blurred in the following algorithms, which are based more on improving iteratively the full archive (originally the evolutionary population) than on focusing on single solutions.

Paquete et al (2004) and Angel et al (2004) simultaneously proposed the first standalone local search algorithms: the *Pareto Local Search* (PLS) and the *Bi-criteria Local Search* (BLS). Both algorithms are very similar and are both known as PLS (or PLS-1 in comparison to the PLS-2 algorithm, the local search algorithm of the EA proposed by Talbi et al (2001) that is sometimes referred to under this name). Unlike in the previous EAs, in PLS algorithms, a *population* is called an *archive* and always consists of a Pareto set. At every iteration of the PLS algorithm, a single solution not yet considered is taken from the archive to explore its neighbourhood and all of its neighbours are used to update the archive.

Aguirre and Tanaka (2005) proposed the *multiple multi-objective random bit climbers* (moRBC), which also follows a local search scheme. At each iteration, all the possible moves of the neighbourhood are generated. They are all successively applied to the current solution, which can be immediately replaced when a dominating neighbour is found. These authors proposed multiple versions of moRBC wherein incomparable neighbours may be accepted and a separate archive may be used for crowding or restarting purposes.

Using the idea of employing a separate standalone procedure to generate the initial solutions of the local search, Paquete and Stützle (2003) proposed the Two-Phase Local Search procedure (TPLS) for bi-objective optimisation problems. First, an initial solution is generated (originally, using a local search), considering the first objective only. Then, a local search is performed, starting from the resulting solution, but using an aggregation of the

objectives slightly more oriented towards the second objective. This step is then repeated until the final local search considers the second objective only. Many variants of the TPLS procedure have been proposed, among which is the 2-Phase Pareto Local Search (2PPLS) procedure by Lust and Teghem (2010), which hybridises the first step by constructing *potentially extreme supported efficient solutions* as the initial set of a PLS algorithm and an adaptive version of TPLS, likewise hybridised with a PLS algorithm, by Dubois-Lacoste et al (2011).

Instead of using the Pareto dominance or an aggregation-based comparison, the Indicator-Based Multi-Objective Local Search (IBMOLS) (Basseur and Burke, 2007; Basseur et al, 2012) accepts neighbours that are better than any solution of the population, by using a binary multi-objective indicator, such as the hypervolume indicator (Zitzler and Thiele, 1999). The population size of IBMOLS is fixed, the worse solution being replaced as soon as a new neighbour is accepted. The authors also proposed an iterative version of IBMOLS, in which the new initial Pareto set is obtained by applying random noise to a given number of solutions of the Pareto set. If the Pareto set is not big enough, additional solutions randomly generated are considered.

Drugan and Thierens (2012) proposed a multi-restart version of PLS with the Iterated PLS (IPLS). IPLS follows the PLS-2 algorithm, but associates with every solution a Boolean flag that is turned off after the solution neighbourhood is explored. When all solutions are flagged, the search first restarts from a new solution that is randomly generated. After a given number of PLS runs, instead of considering a new solution, IPLS uniformly selects at random a solution from the archive, applies a mutation and restarts from the resulting solution.

Two separate generalisations of the PLS algorithms have since been independently proposed: the Dominance-based Multi-objective Local Search (DMLS) (Liefooghe et al, 2012) and the Stochastic Pareto Local Search (SPLS) (Drugan and Thierens, 2012). The DMLS generalisation uses an archive of solutions and includes multiple strategies related to the selection of solutions to explore and to the exploration of the neighbourhood. $DMLS(\alpha \cdot \beta)$ denotes that the DMLS uses the selection strategy α (with $\alpha \in \{1, \star\}$ for the selection of a single random solution and all solutions, respectively) and the exploration strategy β (with $\beta \in \{1, 1_{\not\prec}, 1_{\succ}, \star\}$ for the acceptance of a single neighbour at random, a single non-dominated neighbour at random, a single dominating neighbour at random and all neighbours, respectively). The SPLS generalisation also uses an archive of solutions from which at each iteration a solution is selected uniformly to be explored. Similarly, multiple exploration strategies are discussed. Furthermore, like in IPLS, a Boolean flag is associated with each solution to avoid exploring it multiple times and to enable faster termination and restarts when the exploration is not performed exhaustively. Indeed, the aforementioned authors also proposed a more generic process to restart PLS algorithms, together with a hybrid genetic PLS algorithm. Moalic et al (2013) also proposed the Fast Local Search (FLS), which behaves like SPLS in that the exploration of a solution neighbourhood stops as soon as a neighbour not

dominated by the archive is found. Tricoire (2012) also proposed the *multi-directional local search* (MDLS), loosely based on PLS, in which at every iteration a solution from the archive is taken at starting point of a subsidiary local search, before merging the resulting archives by filtering dominating solutions.

The anytime behaviour of PLS algorithms has been investigated by Dubois-Lacoste et al (2012, 2015), who proposed variants that optimise not just the quality of the final archive only, but also the quality of intermediate archives. They proposed the *Optimistic HyperVolume Improvement* (OHVI), an alternative mechanism for selecting the solution of the archive whose neighbourhood will be explored, and, more importantly, they showed that changing the exploration strategy during the search could improve the performances of the PLS algorithm.

Finally, Inja et al (2014) proposed the *Queued Pareto Local Search* (QPLS), another restart scheme using a queue to avoid premature convergence. Starting from the initial solutions, QPLS recursively explores every solution of the queue by using dominating neighbours to finally obtain a single final solution. If this final solution is not dominated by the archive, it is merged and k incomparable neighbours are added to the queue. The authors also proposed the *Genetic Queued Pareto Local Search* (GQPLS), which hybridises genetic algorithm techniques to update the queue.

3.4 Condensed Literature Summary

All of the MOLS algorithms outlined above share a common structure, in which a Pareto set of solution is iteratively improved by considering either a solution or a set of solutions as *current*, which is then explored to merge some or all of their neighbouring solutions to the Pareto set. Table 1 summarises the main local search algorithms in the literature, according to the five following local search attributes.

Current solutions A single current solution or a current set of multiple solutions is used by the local search.

Archive The local search keeps track of a separate current set, or the current solutions can be directly selected from the archive.

Neighbourhood exploration A single neighbour, the full neighbourhood or only a subset of the neighbourhood (if a stopping criterion is used) is evaluated.

Acceptance criterion Incomparable and dominated neighbour may be accepted and returned after the neighbourhood exploration, either as the stopping criterion of the exploration or in addition to the final neighbour.

Quality The comparison of the quality of two neighbours is done by considering either an aggregation or the Pareto dominance.

Reference During the neighbourhood exploration, neighbours are compared either to the current solution or to other solutions such as the full Pareto set.

Table 1 Condensed Literature Summary

"X": the algorithm possesses the given characteristic
"C": the algorithm may be configured to possess the given characteristic

	Current		Archive		Neighbours			Acceptance Criterion		Quality		Reference	
Algorithm	Single current solution	Multiple current solutions	Separate archive and current solution	Current solution(s) from the archive	Single neighbour explored	Partial neighbourhood exploration	Full neighbourhood exploration	Accept if incomparable	Accept if dominated	Aggregation	Pareto dominance	Compare to current solution	Compare to Pareto set
MOSA	X		X		X			X	X	X		X	
PSA		X	X		X			X	X	X		X	
MOTS		X	X				X	X		X		X	X
MOVNS	X			X			X	X			X		X
MOGLS	X			X	X			X		X			X
PAES	X	C	X		C	C		X		X		X	
PLS-2		X		X			X	X			X		X
PLS-1	X			X			X	X			X		X
moRBC	X		X	X		X		C			X	X	
IBMOLS		X		X		X		X	X				X
DMLS	C	C		X	C	C	C	C			X	C	C
SPLS	X			X	C	C	C	C			X	C	C
FLS	X			X	X			X			X		X

MOSA (Serafini, 1994; Fortemps et al, 1994); PSA (Czyzak and Jaszkiewicz, 1996); MOTS (Hansen, 1997); MOVNS (Geiger, 2008); MOGLS (Ishibuchi and Murata, 1996); PAES (Knowles and Corne, 1999, 2000a); PLS-2 (Talbi et al, 2001); PLS-1 (Paquete et al, 2004; Angel et al, 2004); moRBC (Aguirre and Tanaka, 2005); IBMOLS (Basseur and Burke, 2007); DMLS (Liefooghe et al, 2012); SPLS (Drugan and Thierens, 2012); FLS (Moalic et al, 2013)

In Table 1, an "X" means that the algorithm possesses the corresponding characteristic, possibly depending of the context during the resolution (e.g., SA algorithm accepting dominated solutions by means of the temperature), whereas a "C" means that the characteristic is only present in some particular variant of the algorithm (e.g., the DMLS structure is able to instantiate many different local search algorithms).

3.5 Analysis and Discussion

Table 1 shows a trend between the two algorithmic families of the MOLS algorithms, where extensions of single-objective local search algorithms generally separate the archive and the current solutions and use aggregations, and the family of the PLS algorithms, which generally directly select the current solutions from the archive and use Pareto dominance.

One of the apparent weakness of MOLS algorithms relates to the possible number of solutions included in the archive and thus the size and shape of the optimal Pareto front. Indeed, if a MOLS algorithm does not use any mechanism to bound the size of its archive, exploration of too many solutions (and furthermore exhaustive neighbourhood explorations) can become prohibitively computationally expensive slowing the convergence of the algorithm to a halting point (Liefooghe et al, 2012). MOLS are similarly much weakened when using too large neighbourhood, especially when explorations are performed exhaustively. Another current weakness of MOLS algorithms is that there is usually no explicit handling of the intensification/diversification trade-off. If some works focus on preserving diversity at the cost of some convergence speed (Blot et al, 2015), in most of the MOLS algorithms only intensification is rewarded and diversification is delegated as a side-effect of the archiving process. Furthermore, some variants of MOLS algorithms may require long computational time to reach high-quality approximations of the Pareto fronts and result on poor solutions if stopped early. Anytime mechanisms for MOLS algorithms have been proposed to deal with this particular limitation (Dubois-Lacoste et al, 2015).

The two DMLS and SPLS generalisations can be configured to instantiate a large range of PLS strategies, but are not compatible with many extensions of single-objective strategies (and do not claim to be). The first fundamental limitation is that these generalisations do not use an explicit set of current solutions that is conveyed through the iterations of the local search, but instead select new current solutions from the archive every iteration. This also implies that the current solutions are always non-dominated. They can, through the use of an activation/deactivation scheme, emulate to some extent some trajectory-based strategies by keeping track of the selection of the previous iteration, but without the flexibility of keeping a separate set of current solutions, which allows, for example, to easily perform explorations outside their current archive (e.g., to explore dominated neighbours or when the algorithm allows some deterioration of the current solutions). The second main limitation is that the use of an archive as the main set of solutions leads to the use of the Pareto dominance (or a weakened version) for quality comparison, which leaves out the use of scalar-based comparisons in the exploration procedure.

To overcome these limitations, to allow more flexibility and to incorporate more diverse strategies, we propose a new MOLS generalisation, which is detailed in Sections 4 and 5. Its main characteristics are the use of two explicit sets of solutions (namely, the set of current solutions and the archive), the separation of acceptance and stopping criteria in the exploration strategy, the

possibility of using a simple set and not a Pareto set for the set of current solutions, the possibility of using scalar-based acceptance criteria and, finally, the use of an explicit reference during neighbourhood comparisons.

4 Multi-Objective Local Search Strategies

In this section, we describe different sets and strategies of the MOLS algorithms through examples from the literature review of the previous section. They are the basic components of our unification of MOLS that will be presented in Section 5.

4.1 Set of Potential Pareto Optimal Solutions (Archive)

The *archive* is the Pareto set at the core of all MOLS algorithms. It holds potential Pareto optimal solutions, i.e., solutions not yet dominated by any other found solutions. This is the set of solutions finally returned by the procedure.

Depending on the problem considered, the size of the archive can become very large. Unless this size is kept unbounded, a mechanism such as a diversity criterion (e.g., crowding, relaxed dominance, etc.) or a basic filtering mechanism may be used to remove the less important potential Pareto optimal solutions once a given size is reached (Liefooghe et al, 2012).

4.2 Set of Current Solutions (Memory)

In addition to the archive, the *current set*, a second set of solutions, is used to keep all the solutions whose neighbourhood may be explored. These solutions are taken either from the archive or from previous iterations and may possibly be dominated by some solutions of the archive. To avoid using the same term (i.e., *current*) for both the current set and the current solutions it contains, we propose to call this set *memory*.

We identify three categories of strategies concerning the usage of the memory. First, as a direct extension of the single-objective local search algorithms, the memory can contain a single current solution (e.g., MOSA algorithm (Ulungu et al, 1999)). Iteration after iteration, the current solution is explored, potentially replaced by one of its neighbours, while the archive is automatically updated. If the current solution appears to be a PLO, a restart can then be performed from one of the other potential Pareto optimal solutions. However, considering a single current solution means focusing on a single trajectory in the search space, whereas the multi-objective setting requires optimising the whole Pareto front. Thus, the second category of strategies includes algorithms that keep a set of multiple *current* solutions and explores it sequentially, with the direct consequence of an improved diversity since each of the separate trajectories can then focus on the subset of the Pareto front (e.g., PSA (Czyzak and Jaszkiewicz, 1998), MOTS algorithms (Hansen, 1997), etc.). Finally, the

third category includes algorithms that do not keep track of the trajectory, but rather directly select and explore solutions from the archive (e.g., PAES (Knowles and Corne, 1999), PLS (Paquete et al, 2004), DMLS algorithms (Liefooghe et al, 2012), etc.).

Note that, like in the archive, the size of the memory may become very large, and, therefore, the same bounding mechanisms may be used. However, as such mechanisms were proposed for algorithms in which the memory and the archive were joined, it may be advantageous to bound only the memory and keep the archive unbounded.

We may envision a new exploration strategy where multiple solutions could be explored at the same time by combining their neighbourhoods. In that case, without loss of generality, the *current* object would be itself a set of solutions and the *memory* would be a set of sets of solutions.

4.3 Exploration Strategies

The exploration of the current solution consists in the construction of its neighbourhood, i.e., the generation of its neighbours.

Like in the single-objective case, two types of exploration strategy are distinguished: the *best improvement strategy* and the *first improvement strategy*. The *best* strategies compare every neighbour to the current solution or to the reference so that only the best non-dominated neighbours are accepted. On the contrary, the *first* strategies generate neighbours one by one and stop when a given stopping criterion is reached. Of course, the latter strategies are not limited to stopping after a single accepted neighbour. In both the *best* and the *first* strategies, the exploration procedure generates some neighbours, accepting some of them, and then returns the set of *accepted* neighbours. For each of these neighbours, three questions arise: (i) Should it be included into the archive? (ii) Should it replace the current solution? (iii) Should the exploration continue or stop in regard to its quality?

The quality of a neighbour can be a function of either the current solution or a part or the totality of the archive. Figures 1 and 2 show how the objective space is divided into dominating solutions, incomparable solutions and dominated solutions, regarding a single solution x and multiple solutions x, u, v and w, respectively. Solutions in the *"c"* space are dominated by the current solutions and are generally ignored, whereas exploration strategies usually consider solutions in the *"a"* or *"a+b"* spaces. Considering the neighbouring solutions enables to make better-informed decisions, e.g., distinguishing between the *"α"*, *"β"*, and *"γ"* spaces; the main drawback, however, is the added cost (e.g., computational time) of an overall more expensive exploration procedure. An alternative to using the Pareto dominance criterion is to aggregate the objectives, to obtain a scalar value subsequently used to either rank neighbours or compute probabilities. The weights of the aggregation can be either globally set, associated with the current solution or updated automatically in regard to the state of the archive.

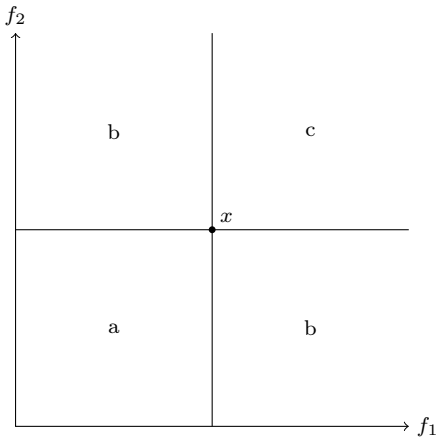

Fig. 1 Objective space around x

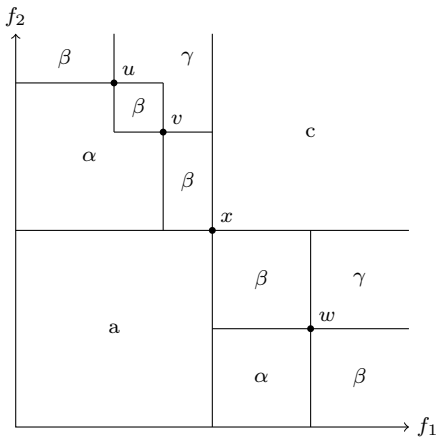

Fig. 2 Objective space around x, taking into account the surrounding solutions u, v and w

"a", "b" and "c": objective space partitions in which the solution respectively dominates, is incomparable with and is dominated by the solution x.
"α", "β" and"γ": subdivisions of the objective space partition "b" in which the solution respectively dominates, is incomparable with and is dominated by the solution u, v or w.

The archive (the set of potential Pareto optimal solutions) can be updated directly either during the exploration of a current solution or after the exploration of all current solutions has been performed. In the direct update, the explorations of the remaining current solutions may be impacted, i.e., the reference set is modified on the fly.

Similarly, the memory (the set of current solutions) can be updated during the exploration to replace the current explored solutions (e.g., in trajectory-based local search algorithms (Serafini, 1994; Fortemps et al, 1994; Czyzak and Jaszkiewicz, 1996; Hansen, 1997)) or to include promising new neighbours directly (Blot et al, 2015).

If the memory contains multiple solutions, they are all explored before the search continues unless an early stopping criterion is met. Note that, if multiple solutions are explored and either the memory or the archive is updated during the exploration, the order in which the solutions of the memory are explored can strongly impact the performance.

4.4 Selection Strategies

After the exploration step has been completed, the solutions of the memory will have been explored and the archive will have been updated with the accepted neighbours. The memory has to be updated for the next iteration. Generally, the solutions are taken from the archive (e.g., randomly, with regard to a crowding or sharing property (Deb, 2001), to an individual contribution

(Dubois-Lacoste et al, 2012) or to the order of insertion in the archive (Blot et al, 2017a)). However, in trajectory-based local search algorithms, the memory is unchanged since it has been updated during the previous exploration step.

4.5 Termination Criteria

The local search has a natural termination criterion, which is reached when the memory becomes empty, meaning that no more solution is to be explored. Such an event generally means that every solution of the archive is a PLO. This situation also arises when the algorithm intentionally removes partially explored solutions from the memory, for example, to force a quick convergence or ensure diversification. Other commonly used termination criteria include the whole computational time; the total number of iterations, explorations or evaluations; and the number of successive iterations without improvement.

4.6 Escaping Local Optima

In single-objective optimisation, local search algorithms are generally trapped in local optima. However, various mechanisms (e.g., SA, TS, etc.) can be used to converge further towards a global optimum. Likewise, the basic instantiations of the procedures detailed in this paper will generally be trapped in sets of PLO. Likewise, the same various mechanisms can be and have been adapted for MOLS procedures to converge further towards the set of Pareto optima.

First, a temperature can be used to compute the probabilities of accepting neighbours of lesser quality (Serafini, 1994; Fortemps et al, 1994; Czyzak and Jaszkiewicz, 1996). This temperature can be either a global parameter of the local search or a specific temperature that can be associated with each and every solution of the memory when the local search follows a set of solutions of fixed size. The Tabu paradigm can also be used to drive the search out of the PLO (Hansen, 1997). Similarly, a global tabu list or a set of tabu list can be used for each followed solution. Finally, it is possible to use an iterated local search scheme (Drugan and Thierens, 2012) to stop the local search early, before reaching a true set of PLO. In this case, a convergence condition is defined as, for example, a threshold in the convergence rate or a stagnation criterion. The search can then restart either from the new solutions selected uniformly in the search space or from the solutions in the close neighbourhood of the current or the best solutions, using a *kick*. In the single-objective case, a kick consists in taking a solution, either the current one or the best one, and performing a given number of random moves over the search space. In the multi-objective case, some solutions are selected (either a single one, a fixed number or a ratio of solutions, or all of them) from the memory or the archive; a single-objective kick is performed on each of them, and the resulting solutions are included in a new Pareto set, and then the algorithm restarts from it.

Procedure 1: LS(`memory`, `archive`)

Input: `memory`, a set of solutions to generate neighbourhoods
Input: `archive`, a Pareto set of solutions
Output: the updated `archive` set

```
repeat
    all_accepted ← ∅;
    repeat
        let current ∈ memory;
        ref ← REFERENCE(current, memory, archive, all_accepted);
        accepted ← EXPLORE(current, ref, archive);
        memory ← UPDATE(memory, current, accepted);
        all_accepted ← all_accepted ∪ accepted;
    until iteration stopping condition is met
       or every current ∈ memory has been considered;
    archive ← COMBINE(archive, all_accepted);
    memory ← SELECT(memory, archive, all_accepted);
until local search stopping condition is met
   or memory = ∅;
return archive;
```

5 Multi-Objective Local Search Unification

From the basic components presented in Section 4, we defined a unified structure of MOLS algorithms that can instantiate the algorithms in the literature (see Section 3) and, maybe, future designs.

5.1 Local Search Algorithm

Procedure 1 (LS) describes the main loop of the local search. This procedure takes an initial current set and an archive as input and returns the updated archive. It consists in iterating three steps (the names in parentheses are the names of the sub-procedures described as they appear in Procedure 1).

1. First, the solutions of the memory are explored one by one: for each, a reference is chosen to compare the neighbours with (REFERENCE), then some or all of the neighbours are accepted as candidates (EXPLORE), and, finally, the memory may be updated with the neighbours (UPDATE).
2. When all the current solutions have been explored, or when an early stopping condition is met, all accepted neighbours are used to update the archive (COMBINE). Note that it is possible to update the archive during the exploration, in which case the COMBINE procedure can still be used to bound its size.
3. Finally, the memory is set up with the new solutions to explore.

These three steps are iterated until the memory is empty or as soon as a given stopping condition is met.

Procedure 2: EXPLORE(`current`, `ref`, `archive`)

Input: `current`, a solution to generate the neighbourhood
Input: `ref`, a set of solutions to compare neighbours with
Input: `archive`, a Pareto set of solutions
Output: `accepted`, the set of accepted solutions
Side effect: modifies the `archive` set

```
accepted ← ∅;
repeat
    let neighbour ∈ 𝒩(current);
    accepted ← ACCEPT(accepted, neighbour, ref);
    current, ref, archive ← UPDATE(ref, accepted, current, archive,
     neighbour);
until exploration stopping condition is met
   or every neighbour ∈ 𝒩(current) has been considered;
return accepted;
```

5.2 Local Search Exploration

The exploration mechanism (EXPLORE) is described in Procedure 2. This procedure handles how neighbouring solutions are generated and accepted, and how the reference set is updated. It takes as input a solution to explore, which is used to generate the neighbourhood; a reference set to compare the neighbours with; and the archive of the local search. It returns a set of accepted neighbours of the input solution and possibly modifies the archive as a side effect.

The neighbours of the current solution are generated one by one, and for each new neighbour, the set of accepted neighbours is updated (ACCEPT). To implement some local search algorithms from the literature, it is possible to immediately update the current solution, the reference set and the archive (UPDATE). Neighbours are generated until every possible neighbour of the current solution has been generated or as soon as a given stopping condition is met.

5.3 Iterated Local Search Algorithm

The local search of Procedure 1 (LS) can eventually stop because either the archive contains only PLO or an early stopping condition has been met. One of the possible mechanisms to iterate the local search (LS) and continue the search is described in Procedure 3 (ITER). It follows the Iterated Local Search (ILS) scheme (Lourenço et al, 2003; Drugan and Thierens, 2010, 2012) where the final archive given by the local search is slightly modified and given again as input to the local search procedure.

First, the local search is performed once, which sets up `archive`*, the Pareto set that contains the overall best non-dominated solutions across local search iterations. Then, until the global stopping condition is met, new initial memory and archive are generated (PERTURB), subsequent local search

Procedure 3: ITER(`archive`)

Input: `archive`, a Pareto set of solutions
Output: the updated `archive`* set

```
archive ← LS(archive);
archive* ← archive;
repeat
    memory, archive ← PERTURB(archive, archive*);
    archive ← LS(memory, archive);
    archive* ← COMBINE(archive, archive*);
until global stopping condition is met;
return archive*;
```

are performed and the two archives are combined to update `archive`* (COMBINE).

5.4 Literature Instantiation

Following the unification presented in Section 5, Tables 2 and 3 detail how the main literature algorithms are instantiated in Procedure 1 and Procedure 2, respectively, of our structure. In Table 2, k designates a constant of the algorithm set beforehand, and in Table 3, the "$*$" symbol means that the memory size is variable.

Table 2 shows that many of the MOLS algorithms in the literature use the current solution as a reference. However, recent studies increasingly encourage the use of the archive as a reference since it leads to improved results (Blot et al, 2017a,b). The recombination column highlights that the recombination only makes sense when the exploration step returns a new Pareto archive; for trajectory-based local search algorithms, such a step is directly performed during the exploration, when a neighbour replaces the current solution in the memory. Not mentioned here is the possible bounding of the archive size, which is also performed on some problems after Pareto filtering (e.g., Liefooghe et al (2012)). The selection column mainly differentiates between trajectory-based algorithms, for which such a step is likewise irrelevant, and algorithms that do not use a memory mechanism but recreate the set of new solutions every iteration.

Lastly, Table 3 shows that, if the first MOLS algorithms predominantly accepted improving neighbours, newer MOLS algorithms have shown that considering incomparable neighbours leads to improved results.

6 Conclusions

The primary purpose of this paper was to propose a unification of the different local search techniques in multi-objective combinatorial optimisation. Indeed, local search algorithms are well known and often used to solve single-objective combinatorial optimisation problems; however, their transposition

Table 2 Condensed Literature Instantiation (LS Procedure)

imp.: improving (implies underlying scalarisation)
dom.: dominating
ndom.: non-dominated (either dominating or incomparable)

Algorithm	REFERENCE	UPDATE	COMBINE	SELECT	Miscellaneous
MOSA	current solution	replace if *imp.*	irrelevant	do nothing	
PSA	current solution	replace if *imp.*	irrelevant	do nothing	
MOTS	irrelevant	always replace	irrelevant	drift if long enough	
MOVNS	irrelevant	remove current if fully explored; add *ndom.* neighbours; filter	Pareto dominance	all solutions	
MOGLS	current solution	replace if *imp.*	Pareto dominance	do nothing	stop after a given number of explorations without improvement
$(1+1)$-PAES	current solution	replace if *dom.* or less crowded	Pareto dominance	do nothing	
$(1+\lambda)$-PAES	current solution	replace if *dom.* or less crowded	Pareto dominance	do nothing	
$(\mu+\lambda)$-PAES	current solution	replace if *dom.* or less crowded	Pareto dominance	do nothing	reference chosen via binary tournament
PLS-2	archive	do nothing	Pareto dominance	all solutions	
PLS-1	irrelevant	replace if *dom.*	Pareto dominance	single unexplored	
moRBC	current solution	replace if *imp.*	irrelevant	irrelevant	restart on local optima
IBMOLS	memory	do nothing	irrelevant	all solutions	
DMLS $(1\cdot\ )$	current solution	do nothing	Pareto dominance	single unexplored	
DMLS $(\star\cdot\ )$	current solution	do nothing	Pareto dominance	all solutions	
SPLS	current solution	do nothing	Pareto dominance	single non-flagged	
FLS	current solution	always replace; filter	Pareto dominance	do nothing	

MOSA (Serafini, 1994; Fortemps et al, 1994); PSA (Czyzak and Jaszkiewicz, 1996); MOTS (Hansen, 1997); MOVNS (Geiger, 2008); MOGLS (Ishibuchi and Murata, 1996); PAES (Knowles and Corne, 1999, 2000a); PLS-2 (Talbi et al, 2001); PLS-1 (Paquete et al, 2004; Angel et al, 2004); moRBC (Aguirre and Tanaka, 2005); IBMOLS (Basseur and Burke, 2007); DMLS (Liefooghe et al, 2012); SPLS (Drugan and Thierens, 2012); FLS (Moalic et al, 2013)

Table 3 Condensed Literature Instantiation (EXPLORE Procedure)

imp.: improving (implies underlying scalarisation)
dom.: dominating
ndom.: non-dominated (either dominating or incomparable)

Memory size: 1: single solution; k: constant; $*$: variable; other: other constant

Algorithm	Memory size	ACCEPT	UPDATE	Miscellaneous
MOSA	1	if *imp.*	update both	
PSA	$*$	if *imp.*	update both	
MOTS	$*$	best non-tabu neighbour	update both	
MOVNS	1	if *ndom.*	irrelevant	use an unexplored neighbourhood
MOGLS	$*$	first *imp.*	do nothing	stop after first *imp.* neighbour or k neighbours
$(1+1)$-PAES	1	if *dom.* or less crowded	do nothing	stop after 1 neighbour
$(1+\lambda)$-PAES	1	if *dom.* or less crowded	do nothing	stop after λ neighbours
$(\mu+\lambda)$-PAES	μ	if *dom.* or less crowded	do nothing	stop after λ neighbours
PLS-2	$*$	if *ndom.*	do nothing	
PLS-1	1	if *ndom.*	do nothing	
moRBC$(1+1)$	1	if *dom.*	replace `ref` if accepted	neighbours generated using the current reference
moRBC$(1+1)^*$	1	if *ndom.*	replace `ref` if accepted	neighbours generated using the current reference
moRBC$(1+1)^A$	1	if *dom.* or less crowded *ndom.*	replace `ref` if accepted	neighbours generated using the current reference
IBMOLS	$*$	if not worst w.r.t. the indicator	remove worst from `ref`	
DMLS $(\ \cdot 1)$	k	if *dom.* or *ndom.*	do nothing	stop after 1 neighbour
DMLS $(\ \cdot 1_{\not\prec})$	k	if *dom.* or *ndom.*	stop after 1 *ndom.*	
DMLS $(\ \cdot 1_{\succ})$	k	if *dom.* or *ndom.*	do nothing	stop after 1 *dom.*
DMLS $(\ \cdot \star)$	k	if *dom.* or *ndom.*	do nothing	
SPLS	1	if *dom.* or *ndom.*	do nothing	
FLS	$*$	if *ndom.*	do nothing	stop after first *ndom.* neighbour

MOSA (Serafini, 1994; Fortemps et al, 1994); PSA (Czyzak and Jaszkiewicz, 1996); MOTS (Hansen, 1997); MOVNS (Geiger, 2008); MOGLS (Ishibuchi and Murata, 1996); PAES (Knowles and Corne, 1999, 2000a); PLS-2 (Talbi et al, 2001); PLS-1 (Paquete et al, 2004; Angel et al, 2004); moRBC (Aguirre and Tanaka, 2005); IBMOLS (Basseur and Burke, 2007); DMLS (Liefooghe et al, 2012); SPLS (Drugan and Thierens, 2012); FLS (Moalic et al, 2013)

in the multi-objective context is not straightforward and has led to different algorithms. Multi-Objective Local Search (MOLS) algorithms can and have been successfully applied to continuous problems, generally through the use of ad-hoc neighbourhood structures such as sampling of close solutions, but they suffer even more of the MOLS weaknesses discussed in the paper.

In this survey, we carried out a large literature review that showed an evolution in the design of MOLS algorithms for multi-objective combinatorial optimisation. First, they were either extensions of single-objective local search algorithms (e.g., Tabu Search and Simulated Annealing) or evolutionary algorithms directly integrating single-objective local search algorithms using the aggregation of the objectives. Then, in the early 2000s, *pure* MOLS algorithms were designed using the Pareto dominance and the concepts of neighbourhood and archive. We highlighted that several concepts and strategies are shared by all MOLS algorithms and that only the instantiation is different. We propose therefore a unified structure that can instantiate most of the MOLS algorithms in the literature.

With a unified structure for algorithms, it is easy to implement both the many existing MOLS algorithms and the new ones. Many research works focus on the automatic design of algorithms that gives a natural perspective to this work: this unified structure paired with a multi-objective automatic algorithm configurator (e.g., MO-ParamILS (Blot et al, 2016)) may help in the design of efficient algorithms for any multi-objective combinatorial optimisation problems.

References

Abbasi M, Paquete L, Pereira FB (2015) Local search for multiobjective multiple sequence alignment. In: Guzman FMO, Rojas I (eds) Bioinformatics and Biomedical Engineering - Third International Conference, IWBBIO 2015. Proceedings, Part II, Springer, Lecture Notes in Computer Science, vol 9044, pp 175–182

Aguirre H, Tanaka K (2005) Random bit climbers on multiobjective MNK-landscapes: effects of memory and population climbing. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 88(1):334–345

Angel E, Bampis E, Gourvés L (2004) Approximating the pareto curve with local search for the bicriteria TSP (1, 2) problem. Theoretical Computer Science 310(1-3):135–146

Arroyo JEC, dos Santos Ottoni R, de Paiva Oliveira A (2011) Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. Electronic Notes in Theoretical Computer Science 281:5–19

Bandyopadhyay S, Saha S, Maulik U, Deb K (2008) A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Transactions on Evolutionary Computation 12(3):269–283

Basseur M, Burke EK (2007) Indicator-based multi-objective local search. In: IEEE Congress on Evolutionary Computation, IEEE, pp 3100–3107

Basseur M, Zeng RQ, Hao JK (2012) Hypervolume-based multi-objective local search. Neural Computing and Applications 21(8):1917–1929

Baykasoglu A, Owen S, Gindy N (1999) A taboo search based approach to find the pareto optimal set in multiple objective optimization. Engineering Optimization 31(6):731–748

Beausoleil RP (2001) Multiple criteria scatter search. In: 4th Metaheuristics International Conference, pp 534–539

Blot A, Aguirre H, Dhaenens C, Jourdan L, Marmion MÉ, Tanaka K (2015) Neutral but a winner! how neutrality helps multiobjective local search algorithms. In: Evolutionary Multi-Criterion Optimization. Proceedings, Part I, pp 34–47

Blot A, Hoos HH, Jourdan L, Marmion MÉ, Trautmann H (2016) MO-ParamILS: A multi-objective automatic algorithm configuration framework. In: LION 10

Blot A, Jourdan L, Kessaci-Marmion MÉ (2017a) Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 227–234

Blot A, Pernet A, Jourdan L, Kessaci-Marmion MÉ, Hoos HH (2017b) Automatically configuring multi-objective local search using multi-objective optimisation. In: Evolutionary Multi-Criterion Optimization, Proceedings, pp 61–76

Czyzak P, Jaszkiewicz A (1996) A multiobjective metaheuristic approach to the location of petrol stations by the capital budgeting model. Control and Cybernetics 25:177–187

Czyzak P, Jaszkiewicz A (1998) Pareto simulated annealing – a metaheuristic technique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis 7(1):34–47

Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. John Wiley & Sons

Drugan MM, Thierens D (2010) Path-guided mutation for stochastic Pareto local search algorithms. In: 11th International on Conference on Parallel Problem Solving from Nature - PPSN XI, Springer, pp 485–495

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

Dubois-Lacoste J, López-Ibáñez M, Stützle T (2011) A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. Computers & Operations Research 38(8):1219–1236

Dubois-Lacoste J, López-Ibáñez M, Stützle T (2012) Pareto local search algorithms for anytime bi-objective optimization. In: European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, pp 206–217

Dubois-Lacoste J, López-Ibáñez M, Stützle T (2015) Anytime pareto local search. European Journal of Operational Research 243(2):369–385

Engrand P (1998) A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management. Tech. rep., Electricite de France

Feo TA, Resende MGC, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. Operations Research 42(5):860–878

Fortemps P, Teghem J, Ulungu B (1994) Heuristics for multiobjective combinatorial optimization by simulated annealing. In: XIth International Conference on MCDM, pp 1–6

Geiger MJ (2008) Randomised variable neighbourhood search for multi objective optimisation. In: Proceedings of the 4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristics, November 4–5, Nottingham, United Kingdom, pp. 34-42, arXiv: 0809.0271

Gendreau M, Potvin JY (2010) Handbook of metaheuristics, vol 2. Springer

Glover F, Laguna M, Taillard E, de Werra D (1993) Tabu search. Baltzer Basel

Hansen MP (1997) Tabu search for multiobjective optimization: MOTS. In: Proceedings of the 13th International Conference on Multiple Criteria Decision Making, pp 574–586

Hoos HH, Stützle T (2004) Stochastic Local Search: Foundations & Applications. Elsevier / Morgan Kaufmann

Inja M, Kooijman C, de Waard M, Roijers DM, Whiteson S (2014) Queued pareto local search for multi-objective optimization. In: 13th International on Conference on Parallel Problem Solving from Nature - PPSN XIII, pp 589–599

Ishibuchi H, Murata T (1996) Multi-objective genetic local search algorithm. In: Evolutionary Computation, 1996., Proceedings of IEEE International Conference on, IEEE, pp 119–124

Ishibuchi H, Tsukamoto N, Nojima Y (2008) Evolutionary many-objective optimization: A short review. In: IEEE Congress on Evolutionary Computation, IEEE, pp 2419–2426

Jaeggi D, Asselin-Miller C, Parks G, Kipouros T, Bell T, Clarkson J (2004) Multi-objective parallel tabu search. In: 8th International on Conference on Parallel Problem Solving from Nature - PPSN VIII, Springer, pp 732–741

Jaeggi D, Parks GT, Kipouros T, Clarkson PJ (2008) The development of a multi-objective tabu search algorithm for continuous optimisation problems. European Journal of Operational Research 185(3):1192–1212

Jaszkiewicz A (2002) Genetic local search for multi-objective combinatorial optimization. European Journal of Operational Research 137(1):50–71

Kirkpatrick S, Gelatt CD, Vecchi MP, et al (1983) Optimization by simulated annealing. science 220(4598):671–680

Knowles J, Corne D (1999) The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, IEEE, vol 1, pp 98–105

Knowles J, Corne D (2000a) Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation 8(2):149–172

Knowles J, Corne D (2000b) M-PAES: A memetic algorithm for multiobjective optimization. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, IEEE, vol 1, pp 325–332

Laumanns M, Thiele L, Deb K, Zitzler E (2002) Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary computation 10(3):263–282

Liefooghe A, Humeau J, Mesmoudi S, Jourdan L, Talbi E (2012) On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. Journal of Heuristics 18(2):317–352

Lourenço H, Martin O, Stützle T (2003) Iterated local search. In: Handbook of Metaheuristics, Springer, pp 320–353

Lourenço H, Martin O, Stützle T (2010) Iterated local search: Framework and applications. In: Handbook of Metaheuristics, vol 2, Springer, pp 363–397

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

Martí R, Campos V, Resende MGC, Duarte A (2015) Multiobjective GRASP with path relinking. European Journal of Operational Research 240(1):54–71

Mladenović N, Hansen P (1997) Variable neighborhood search. Computers & operations research 24(11):1097–1100

Moalic L, Caminada A, Lamrous S (2013) A fast local search approach for multiobjective problems. In: International Conference on Learning and Intelligent Optimization, Springer, pp 294–298

Molina J, Laguna M, Martí R, Caballero R (2007) SSPMO: A scatter tabu search procedure for non-linear multiobjective optimization. INFORMS Journal on Computing 19(1):91–100

Moslehi G, Mahnam M (2011) A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. International Journal of Production Economics 129(1):14–22

Murata T, Ishibuchi H, Gen M (2000) Cellular genetic local search for multi-objective optimization. In: Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., pp 307–314

Paquete L, Stützle T (2003) A two-phase local search for the biobjective traveling salesman problem. In: Evolutionary Multi-Criterion Optimization, Springer, pp 69–69

Paquete L, Chiarandini M, Stützle T (2004) Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Metaheuristics for Multiobjective Optimisation, Springer, pp 177–199

Serafini P (1994) Simulated annealing for multi objective optimization problems. In: Multiple Criteria Decision Making, Springer, pp 283–292

Suman B (2003) Simulated annealing-based multiobjective algorithms and their application for system reliability. Engineering Optimization 35(4):391–

416

Suman B, Kumar P (2006) A survey of simulated annealing as a tool for single and multiobjective optimization. Journal of the Operational Research Society 57(10):1143–1160

Suppapitnarm A, Parks G (1999) Simulated annealing: an alternative approach to true multiobjective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), Morgan Kaufmann Publishers, pp 406–407

Suresh R, Mohanasundaram K (2004) Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives. In: Cybernetics and Intelligent Systems, 2004 IEEE Conference on, IEEE, vol 2, pp 712–717

Talbi EG, Rahoual M, Mabed MH, Dhaenens C (2001) A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop. In: International Conference on Evolutionary Multi-Criterion Optimization, Springer, pp 416–428

Tricoire F (2012) Multi-directional local search. Computers & OR 39(12):3089–3101

Ulungu B, Teghem J, Fortemps P (1995) Heuristic for multi-objective combinatorial optimization problems by simulated annealing. MCDM: Theory and Applications

Ulungu B, Teghem J, Fortemps P, Tuyttens D (1999) MOSA method: a tool for solving multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis 8(4):221

Vianna DS, Arroyo JEC (2004) A grasp algorithm for the multi-objective knapsack problem. In: Computer Science Society, 2004. SCCC 2004. 24th International Conference of the Chilean, IEEE, pp 69–75

Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: International Conference on Parallel Problem Solving from Nature, Springer, pp 832–842

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE TEVC 3(4):257–271