Iterated Local Search

Thomas Stützle and Rubén Ruiz

Abstract

Iterated local search is a metaheuristic that embeds an improvement heuristic within an iterative process generating a chain of solutions. Often, the improvement method is some kind of local search algorithm and, hence, the name of the metaheuristic. The iterative process in iterated local search consists in a perturbation of the current solution, leading to some intermediate solution that is used as a new starting solution for the improvement method. An additional acceptance criterion decides which of the solutions to keep for continuing this process. This simple idea has led to some very powerful algorithms that have been successfully used to tackle hard combinatorial optimization problems. In this chapter, we review the main ideas of iterated local search, exemplify its application to combinatorial problems, discuss historical aspects of the development of the method, and give an overview of some successful applications.

Keywords

Stochastic local search • Metaheuristics • Iterated local search • Local search • Perturbation • Acceptance criterion

Contents

Introduction 2
Iterated Local Search 3
Local Search Heuristics 3
Iterated Local Search Framework 5

T. Stützle (✉)
IRIDIA, Université Libre de Bruxelles (ULB), Brussels, Belgium
e-mail: stuetzle@ulb.ac.be

R. Ruiz
Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Universitat Politécnica de València, València, Spain
e-mail: rruiz@eio.upv.es

© Springer International Publishing AG 2017

R. Martí et al. (eds.), *Handbook of Heuristics*,
DOI 10.1007/978-3-319-07153-4_8-1

Some Examples of Iterated Local Search Algorithms 8
TSP Example 9
QAP Example 10
Permutation Flow-Shop Scheduling Example 11
Historical Development of Iterated Local Search 12
Relationship of ILS to 14
. . . Simple SLS Methods 14
. . . Hybrid SLS Methods 15
. . . Population-Based SLS Methods 16
Applications 17
Iterated Local Search for Routing Problems 17
Iterated Local Search for Scheduling Problems 18
Iterated Local Search for Other Problems 20
Conclusions 21
Cross-References 22
References 22

Introduction

One important rationale of a large number of metaheuristics is to overcome the limitations of local optimality incurred by iterative improvement methods [40]. Several concepts for this task have been considered in the prolific literature on metaheuristics that include the occasional acceptance of worsening moves within a local search process, an idea underlying methods such as simulated annealing [20, 64] or tabu search [40], the usage of populations as a simple means for increasing the exploration of the search space as in evolutionary algorithms [7] or ant colony optimization [29], or the introduction of penalties in the search process as in dynamic local search methods [53].

Certainly, one of the simplest ideas is to call iterative improvement methods several times with new starting solutions to sample possibly new and better local optima. Doing so by generating new starting solutions uniformly at random from the search space is known to be an ineffective approach, especially when instance size increases [100]. Iterated local search (ILS) instead exploits the idea of generating a chain of solutions by creating new starting solutions from a perturbation of some of the previously found solutions; in this way, it creates a biased sampling of starting solutions [74]. An acceptance criterion determines whether the new locally optimal solution is taken as the new incumbent or whether the previous solution is maintained. This basic idea underlying ILS has been implemented in a number of early papers on the method [10, 11, 60, 79, 80]. The method has been extended in several directions in follow-up research, and a number of new variants and generalizations have been considered. In fact, a first generalization is that the underlying improvement heuristic need not be necessarily an iterative improvement algorithm but can be any method that takes some solution as input and returns a possibly improved solution. Other variants concern the introduction of rather elaborate perturbation mechanisms or the consideration of different acceptance criteria. Nevertheless, what characterizes an ILS algorithm is that a chain of

solutions is built through a process that involves as main sub-procedures some form of solution perturbation, an improvement method of whatever type, and an acceptance criterion.

In this chapter, we review the main ideas underlying ILS and illustrate its main principles using some example applications to well-known combinatorial optimization problems in section "Some Examples of Iterated Local Search Algorithms". We then shortly highlight, in section "Historical Development of Iterated Local Search", the main developments on the method from a historical perspective, which also shows that ILS is among the oldest metaheuristic techniques. We discuss the most relevant links ILS has to other metaheuristics in section "Relationship of ILS to . . .". We end with an overview of various noteworthy applications of ILS, which complements some earlier reviews on ILS [74,75], and some concluding remarks in section "Conclusions".

Iterated Local Search

Here, we first introduce basic notions of local search and then elaborate on the main mechanisms ILS uses to steer the search process.

Local Search Heuristics

Many heuristics for tackling hard optimization problems rely on (perturbative) local search [1, 53]. The idea underlying local search is to replace the current candidate solution s within a search space S of complete candidate solutions by some neighboring candidate solution. The neighborhood $N(s)$ of a candidate solution $s \in S$ is defined by all candidate solutions that can be obtained by applying specific modifications or "movements" to s.

Neighborhoods are problem specific, but for many problems, standard neighborhood concepts can be applied. One of these standard neighborhood concepts is the so-called k-exchange neighborhoods, where two candidate solutions are neighbored if they differ in at most k solution components. For implementing such neighborhoods, one needs to identify, for a given problem, appropriate solution components. Intuitively, a solution component corresponds to an atomic relationship, assignment, or selection that characterizes a candidate solution. For example, in the well-known traveling salesman problem (TSP), solution components refer to the fact that a node j is visited directly after a node i in a tour, that is, to an edge (i, j); in the quadratic assignment problem (QAP), a solution component would correspond to the assignment of a facility to a specific location, that is, an individual assignment. In a k-exchange move for the TSP, one would change a set of k edges by a different set of k edges, and for the QAP, one would change at most k individual assignments of facilities to locations.

Of critical importance is the size of neighborhoods: large neighborhoods have the advantage that better-quality candidate solutions may be reached in one step

of the algorithm, which comes, however, with the disadvantage of increasing the time it takes to actually search for such candidate solutions in large neighborhoods. To balance these two aspects, over the years a large set of techniques have been proposed that try to either define special-purpose neighborhood structures that allow to design algorithms for searching them efficiently or to heuristically explore very large neighborhoods trying to identify good candidate solutions there but without guarantee of finding the best one. For an overview of very large neighborhood search methods, we refer to Ahuja et al. [2].

The widest spread perturbative local search method is probably iterative improvement (also called iterated descent, hill-climbing, etc. in the literature). It replaces a current candidate solution by an improving neighboring one and repeats these steps until a locally optimal candidate solution has been found, that is, a candidate solution without any improving neighbor. In the following, we will indicate by $s^* \in S^*$ a locally optimal candidate solution from the set of locally optimal candidate solutions S^*; more in general, we will refer to the set S^* as the possible output set of search methods that are not necessarily iterative improvement methods. Various variants of this basic algorithm exist differing in the order in which the neighborhood of a candidate solution is searched, which neighboring candidate solution replaces the current one, and so on. The most common mechanisms for these rules are the so-called best- and first-improvement rules, which, respectively, replace the current candidate solution by a most-improving candidate solution in $N(s)$ or by the first one encountered when scanning the neighborhood. The specific choice for these pivoting rules can have a significant impact on the quality of the local optima generated and the computational effort it takes to reach local optima. Similarly, for local search algorithms, speed-up techniques such as incremental move evaluations are of crucial importance for their efficiency [1,53].

Local optima have an ambivalent importance in optimization. When compared to the candidate solutions in the full search space S, locally optimal solutions are of, on average, much better quality, and there are much less candidate solutions in S^* than in S. Hence, identifying local optima by an iterative improvement algorithm may seem a priori a good idea. However, iterative improvement algorithms are stuck in local optima, and even when several candidate solutions in S^* are sampled independently, the so sampled local optima may still be of rather poor quality [100]. Thus, other mechanisms than independent sampling in S^* are required to allow a highly performant search process. The search for such mechanisms led to a large number of proposals and studies of general mechanisms to pursue the local search beyond the traps of local optimality [40] or on the development and study of rather different search mechanisms, such as populations, which try to focus on a more global search behavior. These mechanisms are commonly referred to as metaheuristics [38, 39] or general-purpose stochastic local search methods [53]. Within the context of ILS, of particular relevance are trajectory-based metaheuristics that at each step modify a single solution. This includes methods such as tabu search, simulated annealing or guided local search. Such metaheuristics can be used in a straightforward way as an improvement method in ILS algorithms.

Iterated Local Search Framework

The main principle of ILS is to generate a sequence of candidate solutions that is obtained by iterating through solution perturbations and subsequent applications of a (local) improvement method. An additional acceptance criterion decides from which candidate solution seen during the run of the algorithm this sequence is continued. Typically, the improvement method is an iterative improvement method or a trajectory-following metaheuristic such as tabu search or simulated annealing. The goal of the perturbation is to introduce a modification into the current candidate solution that is larger than the modifications that are done in the local search phase, thus effectively allowing the search to escape local optima or specific search space regions.

An algorithmic outline of ILS is given in Fig. 1. After generating an initial candidate solution and improving it by a local search, the main loop is invoked. Within the main loop, first a (typically locally optimal) candidate solution $s^* \in S^*$ is perturbed leading to a candidate solution $s' \in S$. After improving this solution and obtaining a new solution $s^{*\prime} \in S^*$, an acceptance criterion decides whether to continue the search from s^* or $s^{*\prime}$. It is also possible to consider in the perturbation and the acceptance criterion aspects of the search history, for example, to adjust the perturbation strength or the choice done in the acceptance criterion; in that case, one would extend the corresponding procedures to Perturbation(s^*, *history*) and AcceptanceCriterion(s^*, $s^{*\prime}$, *history*).

An important advantage of ILS when compared to repeatedly starting the LocalSearch method from random initial candidate solutions is that (i) intermediate candidate solutions obtained through the perturbation are often of better quality and can lead to better locally optimal solutions, and (ii) in the case of iterative improvement methods, the subsequent local search may terminate more quickly than when starting from random initial candidate solutions, leading to the exploration of a larger number of local optima within a given computation time. Hence, more local optima and local optima of a better average quality may be generated in an ILS algorithm, increasing the chance to find very high-quality solutions.

A further advantage of ILS is its ease of a first implementation, especially if an improvement method is already available. A first version of an ILS algorithm can be obtained by adding perturbations in some higher-order neighborhoods than the ones used in the local search and a reasonable first choice is to accept only better or

Fig. 1 Algorithmic outline of Iterated Local Search (ILS)

```
procedure Iterated Local Search
    s_0 = GenerateInitialSolution
    s* = LocalSearch(s_0)        % optional
    repeat
        s' = Perturbation(s*)
        s*' = LocalSearch(s')
        s* = AcceptanceCriterion(s*, s*')
    until termination condition met
end
```

equal quality solutions in the acceptance criterion. Such a version can be obtained by adding few lines of code to an existing improvement method and be the basis for an algorithm engineering effort to generate a high-performing ILS algorithm. In fact, ILS is very malleable and can range from very straightforward implementations to rather well-engineered algorithms when problem-specific details are taken into account.

Generally speaking, ILS is a modular approach and the behavior of an ILS algorithm is determined by the four procedures GenerateInitialSolution, Perturbation, LocalSearch, and AcceptanceCriterion and their interaction. In what follows, we discuss the main issues arising for these procedures.

GenerateInitialSolution. This procedure determines the starting point of the search and it may have a significant impact during the initial search phase. Generally, candidate solutions generated by a good constructive heuristic should be preferable as often the better the quality of the initial candidate solution, the better is the quality of the local optima encountered. In addition, an iterative improvement algorithm requires usually less steps to reach a local optimum when starting from a better-quality solution. However, if the other procedures are well designed, the influence of the initial candidate solution is expected to be little especially for longer run-times.

Perturbation. The perturbation transforms one complete candidate solution into another complete candidate solution. Its task is to modify a current candidate solution and to generate a new, promising starting solution for the next local search application. To allow an effective escape from local optima, the perturbation should be larger than or at least different in nature from the modifications that are applied in the local search algorithm.

An important factor is the size of the perturbation, which can be measured by the number of solution components that are changed. If the perturbation is too small, it may quickly be undone by the next local search. If the perturbation is too strong, that is, too many solution components are modified, then much of the quality and the structure of the current candidate solution may be lost; in that case, the algorithm can become akin to a random restart algorithm, which is known to be usually of poor performance. Unfortunately, it depends typically on the problem and often on the specific problem instances how strong perturbations should be [74]. One possibility to adapt the perturbation strength may be either to do a proper parameter tuning, an adaptive scheme following ideas of reactive search [9], or to adapt the perturbation strength using simple schemes such as those of basic variable neighborhood search [45].

The simplest way of perturbing a candidate solution is by applying random perturbation moves. A disadvantage of choosing the perturbing moves uniformly at random is that one may lose significantly in solution quality. An alternative is to use biased perturbations and to introduce problem knowledge in this way. Biased perturbations may make use of heuristic information on the solution components that are changed. This can be done in various forms, for example, by removing

preferentially solution components that have a high contribution to the cost of a candidate solution, or by introducing new solution components that have a low cost contribution. Closely related to ILS is the iterated greedy method that relies on a destruction and reconstruction cycle that is managed through constructive heuristics [98]; in fact, this cycle may be seen as a perturbation in the ILS sense. A different possibility is to introduce more complex perturbations by re-optimizing parts of a candidate solution [73] or applying data perturbations [21].

LocalSearch. The local search procedure is of crucial importance for the performance of an ILS algorithm, which will benefit strongly from good choices and implementations for this procedure. A first crucial issue is the quality of the solutions that the local search algorithm produces. A priori, one may expect that the better the quality of the solutions that are generated by the local search, the better the results of the ILS algorithm. However, this reasoning would not take into account the execution time of the local search. In fact, a crucial aspect for the appropriate choice of a local search method is the trade-off between the quality of the solutions it generates and the computation time it takes to find these solutions. Given a fixed computation time, a fast local search may be applied more often and, thus, generate a larger number of candidate solutions that may be of less average quality than candidate solutions obtained by a more time-consuming local search procedure. Which choice is best, depends on the particular problem or type of problem instances and has to be determined on an experimental basis.

Another important aspect is whether the local search is used as a black-box or whether it is open and can be integrated in possibly more profitable ways into an ILS algorithm. One example is the exploitation of the don't look bit technique [14, 61]; we refer to section "TSP Example" for more details on this technique and how to integrate it with the perturbation. Another example requiring access to the local search is the adoption of techniques of tabu search by declaring solution components changed by the perturbation as tabu for a number of local search steps; this may help to avoid immediately undoing the perturbation.

The local search in an ILS algorithm is not limited to an improvement method, but any (even non-local-search) method that takes as input a complete candidate solution and returns a potentially improved candidate solution upon completion could be used. Common examples include the usage of tabu search, simulated annealing, or dynamic local search algorithms as the local search. One can even think of using an ILS algorithm as the local search leading to a hierarchy of ILS algorithms [55, 74]. However, to allow a sufficient number of iterations of the ILS algorithm, one needs to define a stopping criterion for the local search method, for example, by limiting the number of local search steps that are applied or the number of steps without improvement.

Finally, an advantage of a strong local search is that the sensitivity of the algorithm to the settings of numerical parameters is often reduced, making in this way the algorithm itself more robust. In this sense, ILS also benefits from the further development of local search techniques and the increased availability of powerful implementations based on very large neighborhoods [2].

AcceptanceCriterion. One interpretation of the search behavior of ILS is that of a biased random walk in S^*, the space of local optima. The acceptance criterion has a crucial impact on the nature of this walk and, thus, the trade-off between exploration and exploitation. As the two extremes, one can have the acceptance criterion that only accepts improved solutions, leading to a search that can be characterized as a randomized iterative improvement algorithm in S^*. On the other extreme is the acceptance criterion that accepts any new solution independent of its quality, leading to a random walk in S^*. Clearly, there are many intermediate choices between these extremes, and various have been explored in the literature. One example is the Metropolis condition [85], which accepts a new, worse solution $s^{*\prime}$ with a probability $\exp\{f((s^*) - f(s^{*\prime}))/T\}$, where T is a parameter; a better or equal quality solution is accepted always. The parameter T may be left fixed throughout a run of ILS or be varied as done in simulated annealing.

Various other choices have been made such as combining short random walks with occasional backtracks to the best candidate solution found so far [22], resulting in a specific usage of the search history within the acceptance criterion. Another simple example for the usage of the search history in the acceptance criterion is the occasional restart of the search from a new initial candidate solution, which can lead to much improved algorithm behavior [103, 105].

If one is to reach high performance with an ILS algorithm, the interactions among the components need to be taken into account. The two main interactions are the following. First, the perturbation should complement the local search and ideally introduce moves that cannot be easily done by the local search. This results in two main advantages: (i) it is easier to avoid returning to the previous locally optimal candidate solution, and (ii) the structure of the candidate solution may be changed in a way that is not possible for the local search to do. Second, the acceptance criterion and the perturbation should be well balanced. In fact, both components can be biased toward a more explorative or more exploitative search behavior: large vs. small perturbations or non-strict (random-walk) vs. strict (improvement) acceptance criteria. Finding the right balance between intensification and diversification of the search is one of the key issues in the design of any effective SLS algorithm, and one advantage of ILS is that the impact specific choices of the algorithm components have on this balance is rather easy to grasp. Thus, ILS is a method for which the SLS algorithm engineering process [107] can follow a well-guided direction.

Some Examples of Iterated Local Search Algorithms

We introduce example applications of ILS algorithms to well-known combinatorial optimization problems, the traveling salesman problem (TSP), the quadratic assignment problem (QAP), and the permutation flow-shop scheduling problem (PFSP). Example implementations for these ILS algorithms are available from http://iridia.ulb.ac.be/~stuetzle/Code.

TSP Example

Traveling salesman problem (TSP). The TSP is given as a graph $G = (N, E)$ where N is the set of $n = |N|$ nodes and E is the set of edges that fully connects the nodes. To each edge (i, j) is associated a distance d_{ij}. Here we assume that the distance matrix is symmetric, that is, we have $d_{ij} = d_{ji}$ for all $(i, j) \in E$; this type of TSP instances are called symmetric and are among the most widely studied types of TSP instances. The objective in the TSP is to determine a Hamiltonian cycle of minimal length. Such a cycle can be represented by a permutation $\pi = \langle \pi(1), \ldots, \pi(n) \rangle$, where $\pi(i)$ is the node index at position i. The objective function to be minimized is

$$\min_{\pi \in S} \; d_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} \tag{1}$$

where S is the search space consisting of the set of all permutations.

The TSP is one of the most famous examples of ILS applications, which is due to the important role it plays in the historical development of the method [10, 11, 60, 79] and due to the very high performance ILS algorithms have reached for the TSP [3, 4, 49, 50, 61].

A basic ILS algorithm for the TSP could be the following. The initial candidate solution is generated by a constructive heuristic such as the nearest neighbor heuristic or the greedy heuristic. For the local search, a first-improvement method in a k-edge-exchange neighborhood (typically, k equal two or three) is used. The perturbation is implemented by the so-called double-bridge move, which is illustrated in Fig. 2; the double-bridge move removes four edges and replaces them with four other edges so that in the schematic view of the move, the characteristic double bridge is formed. The acceptance criterion forces the cost to decrease, that is, only better-quality candidate solutions are accepted.

Such a basic ILS algorithm for the TSP reaches high-quality tours, but its performance can be improved by alternative choices for the components. ILS algorithms for the TSP usually benefit from a local search with better performance. For example, by replacing the local search algorithm in the two- or three-edge-exchange neighborhood with the Lin-Kernighan heuristic [70], better performance is usually achieved. It is interesting to note that the double-bridge move was

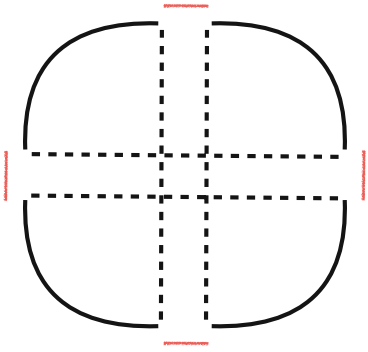

Fig. 2 Schematic view of the double-bridge move. Four edges (continuous, *red lines*) are removed from the current tour, and the four remaining tour segments are reconnected as shown by the *dashed line* in the schematic view of the move

considered within the Lin-Kernighan heuristic, but many implementations of it did not implement it. It is a type of move that cannot be easily achieved by a concatenation of moves in smaller neighborhoods or by the way complex moves in the Lin-Kernighan heuristic are generated, and, thus, it is complementary to standard local search implementations. A main advantage of the double-bridge move is that it only leads to a small increase of the tour length as only four edges are replaced and the subsequent local search typically needs only few improvement steps to reach a new local optimum. In practice, the double-bridge move has found to be rather effective even for very large-sized TSP instances with many thousands or millions of nodes [4, 61].

For speeding-up the search, direct access to the local search through the usage of the well-known don't look bit technique [14] was found to be useful. The don't look bit technique associates one such bit to each node in a TSP instance and starts a local search centered around a node for an improving move only if the don't look bit is set to one (turned on). If this local search is not successful, the don't look bit is turned off by setting it to zero. If a move improves a candidate solution, all end points of the edges involved will have set their don't look bit to one again. This technique leads to a particularly significant speedup, when integrated with the perturbation by re-setting only don't look bits to one for nodes that are directly affected by the perturbation [61] or additionally to some close-by ones [103].

An acceptance criterion that allows only moves to better or equal length tours leads to a strong intensification behavior of the search. While this results in very good performance for short and medium run-times, for longer ILS runs, a stronger diversification through, say, additional restarts improves algorithm performance [103, 105].

QAP Example

Quadratic assignment problem (QAP). In the QAP we are given two matrices F and D corresponding to some kind of flow exchanged between items and a distance matrix between possible locations of the items, respectively. Let f_{ij} be the flow exchanged between items i and j and d_{kl} be the distance between locations k and l. The task in the QAP is to assign items to locations so that the cost given by the sum of flows times the associated distances is minimized. Assuming the number of items is equal to the number of locations, a candidate solution to the QAP can be represented by a permutation π, where $\pi(i)$ gives the location to which item i is assigned. The objective function to be minimized for the QAP can then be written as

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij} \cdot d_{\pi(i)\pi(j)} \tag{2}$$

Similar to the TSP, the QAP has played a central role in combinatorial optimization. Differently from the TSP, for which optimal solutions can be found for very large instances (the largest being solved optimally having 85,900 nodes [5]), QAP instances can only be solved optimally for relatively small instance sizes of around 30; there are only few exceptions for very particularly structured instances for which an instance of size 128 could be solved [34].

A basic ILS algorithm for the QAP can be implemented as follows. An initial candidate solution for the QAP can be generated uniformly at random, as no constructive algorithms generate generally very high-quality candidate solutions. As local search algorithm, an iterative improvement algorithm in the two-exchange neighborhood, where at each step the locations of two items are exchanged, is used. More formally, two candidate solutions π and π' are neighbored if for exactly one index pair (i, j) we have $\pi'(i) = \pi(j)$ and $\pi'(j) = \pi(i)$ and for all other indices $l \neq i, j$ we have $\pi'(l) = \pi(l)$. As a perturbation, a random move in a k-exchange neighborhood, $k > 2$, can be chosen and, as acceptance criterion, one accepts only better-quality candidate solutions.

Such a straightforward ILS version was tested by Stützle [103] and Lourenço et al. [74], and it was shown that its relative performance with respect to other choices of the main ILS components depended on the specific type of QAP instances. In fact, the specific structure of QAP instances as identified by different distributions of the entries of the flow or the distance matrices can induce widely different behavior [110]. For example, the benchmark instances that have been generated randomly assuming uniform distributions of the matrix entries result in landscapes with many local optima that are spread across the whole search space. Such a landscape requires strong search diversification, which can be achieved in ILS by using an acceptance criterion that accepts every new local optimum independently of its quality and helps to improve the performance on such instances even when using only very small perturbations [74, 104]. Differently, for randomly generated instances that resemble real-life instances, which exchange most flow among relatively few items, accepting only better-quality candidate solutions combined with some occasional restarts was found to result in high performance [74, 104]. Further studies on the QAP have shown that plain ILS algorithms reach very high performance for a variety of QAP instances and that population-based extensions of ILS algorithms are state of the art for tackling various classes of QAP instances [104].

Permutation Flow-Shop Scheduling Example

Permutation flow-shop scheduling problem (PFSP). The PFSP arises in many practical situations as it is common to have production lines where machines are disposed in series. In the PFSP are given n jobs $J_1, J_2, \ldots J_n$, each consisting of m operations that are to be executed on m distinct machines $M_1, M_2, \ldots, M_m$. Each job needs to be processed on every machine and all jobs pass through the machines in the same machine order, that is, first on machine M_1, next on machine M_2, and so on until machine M_m. The non-negative, known and deterministic processing time of job J_i on machine M_j is denoted by p_{ij}. The PFSP enforces that all jobs are processed on every machine in the same order; therefore, a permutation of the job indices is enough to define the processing sequence. In the basic PFSP, one assumes that jobs are available for processing at time zero, pre-emption is not allowed, each machine can process only one job at a time and each job can be processed by at most one machine at a time, and that the capacity of buffers between machines is unlimited. A common objective is to minimize the completion time of the last job in the permutation, also known as makespan ($C_{\max}$).

The PFSP with the makespan objective is one of the most widely studied scheduling problems with hundreds of papers published; for reviews on it, see [33,35,48,98]. Furthermore, it is the basis of many other variants and extensions.

A first ILS algorithm for this problem has been proposed by Stützle [102]. It used the NEH heuristic [92] to generate an initial candidate solution. The NEH heuristic is a state-of-the-art constructive method for the PFSP with some recent variants being proposed only in 2014 [32]. It is a heuristic that builds a schedule by inserting at each step a next job in a position of the partial schedule where it least increases the objective function value. Using the speedup techniques proposed by Taillard [109], the NEH heuristic has a computational complexity of $O(n^2m)$. As an iterative improvement method in the ILS algorithm, the insert neighborhood was used, which exploits the same speedups as implemented for NEH. The insert neighborhood consists of all sequences that can be obtained by removing a job at position i and inserting it in all other possible positions; this results in a neighborhood of size $n \cdot (n-1)$. If one removes the job with index $\pi(i)$ from position i and inserts it at a position j, for $i < j$, one obtains $\pi' = (\pi(1), \ldots \pi(i-1), \pi(i+1), \ldots \pi(j), \pi(i), \pi(j+1), \ldots, \pi(n))$ and for $i > j$ one obtains $\pi' = (\pi(1), \ldots \pi(j-1), \pi(i), \pi(j), \ldots \pi(i-1), \pi(i+1), \ldots, \pi(n))$. The local search implemented in Stützle [102] used a kind of first-improvement neighborhood scan. It removes a job J_i from the sequence and checks all $n-1$ possible insertion positions; if several result in improvements, the position is taken that leads to the largest reduction in the makespan. One scan of the neighborhood consists in repetitions of this process for all possible positions i. The neighborhood scans are then repeated until a local optimum with respect to the insert neighborhood is found. As the perturbation, a mix of two contiguous swap moves, and one interchange move was proposed. In a contiguous swap move, a permutation $\pi = (\pi(1), \ldots, \pi(i), \pi(i+1), \ldots, \pi(n))$ is modified to $\pi' = (\pi(1), \ldots, \pi(i+1), \pi(i), \ldots, \pi(n))$; in an interchange-move $\pi = (\pi(1), \ldots, \pi(i), \ldots, \pi(j), \ldots, \pi(n))$ is modified to $\pi' = (\pi(1), \ldots, \pi(j), \ldots, \pi(i), \ldots, \pi(n))$. A further restriction was imposed on the positions involved in the interchange move by enforcing $|i-j| \leq \max\{n/5, 30\}$ to avoid too strong disruptions. Finally, as the acceptance criterion, the Metropolis condition with a fixed temperature was suggested.

The performance of this ILS algorithm was found to be superior to the best methods for the PFSP published until then. The excellent performance was confirmed in the extensive study of heuristics for the PFSP under makespan minimization by Ruiz and Maroto [98], where it was found to be the top performing heuristic.

Historical Development of Iterated Local Search

ILS has a long history, and the ideas underlying the method can be traced back to several approaches. The most influential have been the early developments for tackling the TSP. Baum [10, 11] has proposed an algorithm he called *iterated descent*, which used an iterative improvement method based on two-exchanges as the local search and random three-exchanges as the perturbation and forced the tour

length to decrease. His results were not very impressive, but inspired Martin et al. [79] to propose their *large-stop Markov chains* (LSMC) algorithm. The name of this algorithm stems from the acceptance criterion, which corresponds to the Metropolis condition. In first instantiations, LSMC used a three-exchange neighborhood in the local search but was later extended to use the Lin-Kernighan heuristic. One contribution of the LSMC algorithm was the introduction of the double-bridge move as the perturbation, which is used in many other ILS algorithms for the TSP. The results with the LSMC algorithm gave a major leap in the performance of TSP heuristics. Further refinements of this approach, in particular, by Johnson [60] and Johnson and McGeoch [61] on one side and Applegate et al. [3] on the other side have defined for a long time the state-of-the-art TSP heuristics. Differences in these implementations concern, on the high-level ILS side, mainly the choices for the acceptance criteria and different choices of the perturbation operators, but decisive are also implementation details of the local search and the data structures used for tackling large instances. Noteworthy is the experimental study by Applegate et al. [4] who performed tests on very large TSP instances with up to 25 million cities and the more recent work of Merz and Huhse [84], which is particularly well suited for very large TSP instances under severe time constraints.

In a number of papers, specific choices for ILS algorithms for the TSP have been considered and analyzed. Codenotti et al. [21] study complex perturbation schemes based on data perturbation. Hong et al. [51] examined the impact the perturbation strength has on ILS performance, indicating that for some instances perturbations larger than those introduced by the double-bridge move are beneficial; they also studied population-based extensions of ILS algorithms for the TSP. Katayama and Narihisa [63] proposed to use candidate solutions different from the incumbent one to generate directed perturbations and showed that this idea can lead to very competitive results when compared to an iterated Lin-Kernighan algorithm. Stützle [103] and Stützle and Hoos [105] have analyzed the run-time behavior of ILS algorithms for the TSP, showing that the most common implementations show stagnation behavior for long runs. As a simple remedy to this behavior, they have proposed occasional restarts of the ILS algorithms [103,105]. Finally, there are some approaches for TSP solving that in their iterated version do not follow necessarily the main ILS steps but are, say, inspired from it. The most notable example is probably Helsgaun's iterated version of his Lin-Kernighan implementation [49]. It generates new starting candidate solution through a constructive mechanism, which is strongly biased toward generating a candidate solution that is close to the incumbent candidate solution.

The ILS principle has been discovered before its first applications to the TSP. Baxter [12] proposed an algorithm for a location problem that made repeated use of a data perturbation technique to create new starting solutions for a local search algorithm. However, it appears that these initial approaches have not been the main inspiring source for other ILS approaches and that this work has been overlooked for quite some time. In fact, most early ILS adaptations to problems other than the TSP seem to have been inspired by the proposals of ILS algorithms for the TSP.

Among the first problems other than TSP tackled by ILS is graph bi-partitioning. Martin and Otto [77, 78] reported for specific types of graphs better performance with their ILS algorithm than for simulated annealing heuristics. Many other early applications of ILS algorithms have been for scheduling problems. Lourenço [73] presented an innovative idea that combined ILS with exact algorithms for job-shop scheduling. In her approach, the exact algorithm is used to solve exactly subproblems on one or two machines, keeping the sequence on other machines fixed; the so modified candidate solution is then used as the perturbed one.

It is important to stress that ILS algorithms have been published and made popular under a variety of different names, ranging from problem or algorithm specific names such as iterated Lin-Kernighan [60,61] or chained Lin-Kernighan [4] to generic names such as large-step Markov chains [79], chained local optimization [78], iterated descent [10, 11], iterated hill-climbing [90], or others. In the local search template proposed by Vaessens et al. [114], ILS was referred to as a multilevel point-based local search. Even today, apparent variations of the ideas underlying ILS are published under new names such as breakout local search [13], creating possibly confusion with the earlier proposed breakout method [88], which relies on a different mechanism for escaping from local optima. A review and unification of the various papers published in the literature has been proposed by Lourenço et al. [74] in their 2002 book chapter on ILS. This chapter has led to a more coherent view of the method, has discussed the main ingredients of ILS algorithms, and has shown trade-offs in the design of effective ILS algorithms.

Relationship of ILS to . . .

ILS, being a relatively old method, has a number of links to other well-known methods, which we will shortly discuss in this section. We focus our discussion on relationships to simple, hybrid, and population-based metaheuristics (or stochastic local search methods), following the classification of Hoos and Stützle [53]. Within this classification, ILS belongs to hybrid SLS methods, which can be characterized as methods that interleave search steps in different neighborhoods or that combine different types of procedures in an interleaved fashion. As such, most links of ILS are within that group.

. . . Simple SLS Methods

Simple SLS methods typically move within one type of neighborhood. They can be seen as direct extensions of perturbative local searches that avoid local optima either by occasionally accepting moves to worse candidate solutions or modifications of the evaluation function during the search. Well-known examples for the former are simulated annealing [20, 64] and tabu search [40], while a well-known example for the latter is guided local search [118].

A first relation is that any of these methods may play the local search part within an ILS algorithm, replacing simpler iterative improvement methods. While iterative improvement methods do have a natural stopping condition, namely, hitting a local optimum, for simple local search methods, in principle, arbitrary computation times can be invested. This leads to the task of balancing the intensification obtained through the simple SLS method and the diversification given through perturbation steps and the acceptance criterion. A main task therefore for the algorithm designer is to determine an appropriate computation time for the SLS method to obtain a good trade-off between the computation time and solution quality.

Another relation is the usage of diversification measures in methods such as tabu search. For example, the random shake-up ideas [38] are directly linked to perturbations in ILS. The well-known reactive tabu search algorithms invoke occasional sequences of random moves to provide a means for search diversification [8], which can directly be seen as a random perturbation whose size is adjusted in dependence of features of the search process. Several ideas from the area of tabu search and, in particular, the memory usage explored there [40] may provide a source of inspiration of how to improve ILS algorithms or to define schemes for adapting parameters at run-time [9].

. . . Hybrid SLS Methods

Hybrid SLS methods combine more than one basic search strategy into an overall method. For example, ILS combines typically one search strategy in the local search with a different type of search strategy in the perturbation. From a high-level perspective, one point ILS has in common with many other hybrid SLS methods is that it combines a local optimization with search steps oriented toward diversification. These diversifying steps may be based on constructive or perturbative search steps.

Considering hybrid SLS methods that obtain diversification through constructive procedures, the closest to ILS is the iterated greedy algorithm, if the latter uses a local search phase. In iterated greedy methods, a new candidate solution is obtained by removing from a complete candidate solution $s \in S$ some solution components, obtaining a partial solution s_p, from which in turn a solution construction process is performed. While such iterated greedy algorithms may result in reasonable performance without improving the newly constructed candidate solution by a local search phase, in many cases, an additional perturbative local search improves performance. In that latter case, the destruction–construction cycle may be seen as a (directed) perturbation in the ILS sense. Another link between the two methods is to see both as creating an iterative process that in the ILS case iterates across local search applications and in the iterated greedy case iterates across applications of constructive algorithms. A different perspective on the mechanism of iterated greedy algorithms is taken by large neighborhood search techniques [101]. They consider also the removal of solution components, but then the subsequent completion of the resulting partial solution may be done with exact techniques, originally taken from constraint programming, or also constructive-type approaches. In any case,

large neighborhood search interprets the completion of a partial solution as the exploration of a large neighborhood, hence the name of the method. In a sense, the application of one iteration of LNS would correspond in ILS terms to one (complex) perturbation that moves a solution $s \in S$ (typically a local optimum) to another complete candidate solution $s' \in S$. More details on iterated greedy and further related methods such as large neighborhood search can be found in the chapter on this topic by Stützle and Ruiz in this handbook. The use of constructive mechanisms to provide search diversification also underlies greedy randomized adaptive search procedures (GRASP) [31, 96], which in their basic form create many independent starting points for a local search through randomized greedy constructive searches. GRASP is clearly distinct from the basic principle underlying ILS, as it does not create a biased walk in the space of local optima. Later extensions of GRASP add various mechanisms with the goal of making it more performing. In fact, some extensions reuse parts of candidate solutions and, hence, move GRASP more toward mechanisms that underlie ILS; see Resende and Ribeiro [96] for an overview of GRASP.

The hybrid SLS method that shares most similarities with ILS is (basic) variable neighborhood search (VNS) [45, 46, 87]. VNS is based on the principle of changing the neighborhood during the search to avoid getting trapped in local minima with respect to one specific neighborhood. While the rationale of the main search mechanism underlying VNS and ILS is rather different, many instantiations of VNS, in particular those related to basic, skewed, or general VNS, can be seen directly as specific instantiations of ILS algorithms. For example, in basic VNS, an iterative improvement process is performed in the smallest neighborhood. Larger neighborhoods are explored randomly, and this random exploration is interleaved with the iterative improvement search in the smallest neighborhood. This loop corresponds to the perturbation and local search steps in ILS algorithms, where the main specificity of VNS is to modify the strength of the perturbation following some fixed scheme in which the neighborhoods are explored: the scheme could be in an increasing or decreasing order and additionally taking into account step sizes. In early instantiations of a basic VNS, only candidate solutions that improve on the incumbent candidate solution after the local search phase are accepted, while in skewed VNS worse candidate solutions are accepted depending on how far the new candidate solution is from the incumbent one. Finally, general VNS is a variation of the basic VNS schemes, where the underlying local search can be a variable neighborhood descent algorithm.

. . . Population-Based SLS Methods

Population-based SLS methods are characterized by the use of a population of candidate solutions to drive the search. As such, they are clearly distinct from ILS, which is based on modifying a single search point. In addition, several population-based search methods such as evolutionary algorithms [7] and ant colony optimization [29] do not necessarily make use of local search algorithms. However,

several links exist. In fact, many population-based algorithms can profit from the introduction of a local search phase as it was clearly shown for ant colony optimization [29], for scatter search [41], and for many evolutionary algorithms resulting in methods such as genetic (or evolutionary) local search [90, 112] or memetic algorithms [89]. ILS as well as the population-based methods can in that case be seen as sampling candidate solutions in the space of local optima.

ILS and memetic algorithms are linked by taking an extreme parameterization of the latter, in particular, using a memetic algorithms with a population size of one. In that case, the mutation operator takes over the role of the perturbation and the selection scheme used for population replacement the role of the acceptance criterion. The usage of only mutation to generate candidate solutions has been called parthenogenetic algorithm [60]. In fact, even when using a population size of more than one individual and excluding recombination operations, in the context of memetic algorithms good performance results are reported for some problems [83]; mutation-only candidate solution modifications are also often used in evolution strategies.

Considering the usage of populations as a convenient means of providing search space exploration, several authors have explored population-based variants of ILS algorithms [51, 103, 104, 111]. The approaches proposed by Hong et al. [51] and Stützle [103] maintain the usual perturbation scheme, applying perturbation at each iteration to a single candidate solution. The usage of time-varying distance bounds among the candidate solutions in the population as a specific diversity mechanism was explored by Stützle [104], resulting in high solution quality for the quadratic assignment problem. Thierens [111] uses the population to (i) perturb only solution components in which pairs of candidate solutions differ, similar to what was done in the genetic transformations proposed by Katayama and Narihisa [63], and (ii) to reduce the neighborhood size during the local search.

Applications

This section summarizes some noteworthy applications of ILS in addition to those mentioned in sections "Some Examples of Iterated Local Search Algorithms" and "Historical Development of Iterated Local Search". As the number of applications of ILS has strongly increased over the recent years, we do not give here an exhaustive list but mainly refer to recent publications for illustrating the progress of ILS. Here, we mention only papers where the authors have identified their algorithms explicitly as ILS algorithms, even though a larger number of published algorithms would fit the framework of ILS without this being made explicit in the respective papers.

Iterated Local Search for Routing Problems

Among the first explicit ILS algorithms for vehicle routing problems (VRPs) are those proposed by Ibaraki et al. [58] and Hashimoto et al. [47]. In the first paper,

the authors tackle the VRP with time windows, where a dynamic programming approach minimizes the penalties for time window violations. The ILS algorithm, which used various neighborhoods in the local search, was shown to reach high performance on instances with up to 1000 customers. Hashimoto et al. [47] tackle a VRP with time windows and time-dependent travel times and costs. They introduce various speedups and neighborhood restrictions and show the effectiveness of their algorithm compared to previous proposals in the literature. Vaz Penna et al. [117] consider a variant where the fleet of vehicles is heterogeneous, that is, it consists of vehicles with different characteristics such as capacities. Melo Silva et al. [82] consider a VRP where split deliveries are allowed, that is, a customer's demand may be satisfied by splitting deliveries across various vehicles or tours; they report very high performance for their algorithm improving a large number of benchmark instances. Palhazi Cuervo et al. [94] tackle the vehicle routing problems with backhauls, where in addition to the consumers who get delivered from the depot, there are also suppliers that send goods to the depot. The authors propose a structurally simple ILS algorithm, where the main ingredient is a local search algorithm that uses multiple neighborhoods and allows oscillations between feasible and infeasible candidate solutions. Michallet et al. [86] consider a periodic VRP with time windows under the consideration of malevolence acts, and the goal becomes spreading the repeated visits to customers across their time windows. A VRP with multiple, incompatible commodities and multiple trips per work day is considered by Cattaruzza et al. [19]. They propose an effective ILS algorithm for this problems, which is shown to perform better than previous approaches. Nguyen et al. [93] consider a two-echelon location-routing problem, where two types of trips arise. One type of trips serves from a main depot a number of subordinate depots, which have to be located appropriately, and a second type of trips delivers goods to customers from the subordinate depots. Vansteenwegen and Mateo [115] solve a cyclic inventory routing problem for a single vehicle. The goal is to minimize the costs of the distribution and the inventory costs at the customers. An efficient ILS algorithm for this problem was shown to outperform previous approaches for this problem. Laurent and Hao [68] considered a multiple depot vehicle scheduling problem, which arises in public transport. Related to routing problems is the team orienteering problem, which arises commonly as a problem faced by tourists when planning their trips. Vansteenwegen et al. [116] have tackled a variant of this problem, which considers time windows, with an effective ILS algorithm that could reach high solution quality in relatively short computation times and improve, for 31 benchmark instances, the best candidate solutions known at that time.

Iterated Local Search for Scheduling Problems

Scheduling problems have been among the first problems tackled by ILS algorithms, and by now a large number of high-performing ILS algorithms have been proposed for many variants of scheduling problems including single and parallel machine scheduling problems as well as scheduling problems of flow- and job-shop type.

The first applications of ILS to single- and parallel-machine scheduling problems are due to Brucker et al. [16, 17]; they have used two neighborhood structures that are nested, where the outer neighborhood essentially is used for perturbation. ILS algorithms have reached particularly excellent results for the well-known single-machine total weighted tardiness problem (SMTWTP), for which the iterated dynasearch algorithms by Congram et al. [22] and the extension thereof by Grosso et al. [43] are state-of-the-art algorithms. den Besten et al. [26] have applied an ILS with a variable neighborhood descent local search also to the SMTWTP. Later ILS algorithms have been applied to the SMTWTP with additional sequence-dependent setup times in two independent works, reaching high performance when compared to other heuristics [108, 120]. A recent application of ILS to the parallel-machine total weighted tardiness problem has been presented by Della Croce et al. [25]. They use generalized interchange moves, ideas from dynasearch, and new large-scale machine-based neighborhoods for the local search to improve over the current state of the art.

ILS algorithms have been proposed to tackle a variety of flow-shop scheduling problems, starting with the best-known variant that considers the minimization of the makespan. The first ILS algorithm for this problem [102] has already been described in section "Permutation Flow-Shop Scheduling Example". Some issues in the design of ILS algorithms for the permutation flow-shop problem have been considered by Juan et al. [62]. ILS algorithms have been adapted by other researchers to variants of the flow-shop scheduling problem and shown high performance for the flow-shop problem with flowtime objective [27]. Later, the authors of [27] further improve their ILS algorithm by including a multi-restart perturbation strategy, where the ILS is continued from the best of a set of perturbations obtained from a local optimum [28]. State-of-the-art results for the same flowtime objective are obtained by [95] with ILS variants, including versions that entail populations. ILS has been used to solve more complex variants of the flow-shop problem. Yang et al. [121] presented an ILS algorithm for a flow-shop variant with several stages in series, where at each stage a number of machines is available for processing the jobs. Ribas et al. [97] deal with the blocking flow-shop problem and propose an ILS procedure where the local search combines moves in different neighborhoods as does the perturbation step. M'Hallah [91] tackles the flow-shop scheduling problem with the objective of minimizing the earliness and tardiness, where due dates are distinct. An ILS algorithm is presented, which uses a variable neighborhood descent in the local search phase. Geiger [36] proposes the usage of an ILS algorithm for tackling the multi-objective flow-shop scheduling problem, obtaining promising solution quality despite the simplicity of the proposed approach. Finally, complex hybrid flexible flowline problems are studied by [113], where ILS approaches are intermingled with other schemes to obtain high-quality candidate solutions for scheduling problems that are very close to real settings found at production floors.

As mentioned before, the first scheduling problem tackled by ILS was the well-known job-shop scheduling problem [73]. The ILS algorithm by Kreipl [66] for the total weighted tardiness job-shop scheduling problem reached high performance,

being surpassed only quite some time later by an evolutionary algorithm that integrated an ILS algorithm as the local search operator [30]. An ILS algorithm is underlying the generic approach proposed by Mati et al. [81], which tackles job-shop problems with regular objectives, that is, objectives whose values increase along with an increase of the completion times of the jobs. The proposed ILS algorithm, which uses a perturbation with a size chosen uniformly at random and an acceptance criterion that accepts every new candidate solution, was shown to reach very high performance for a number of criteria, including weighted tardiness and weighted completion time.

Iterated Local Search for Other Problems

ILS algorithms have been applied to tackle many other problems, illustrating the wide range of possible applications of the ILS principle. Corte and Sörensen [24] use an ILS algorithm for designing a water distribution network, obtaining excellent computational results despite the fact that the proposed algorithm has a much simpler structure than many of the other proposed methods for the same task. Grosso et al. [44] propose an ILS algorithms for the maximin Latin hypercube design problem, which consists in assigning positions to n points in a k-dimensional grid such that no two points have a same coordinate and the distance between the closest pair of points is maximized. A possibility to improve the state estimation for the monitoring of power systems is to insert power measure units into the network at appropriate positions. Hurtgen and Maun [54] propose an ILS algorithm for minimizing the size of the configuration to reach full observability of the network. While virtually all ILS applications have been to deterministic problems, Grasas et al. [42] consider adaptations for ILS to integrate simulation in order to be able to tackle stochastic combinatorial optimization problems. The approach is illustrated with some example application of the proposed SimILS framework. The integration of information from lower bounding techniques into an ILS algorithm is considered by Buson et al. [18] for solving the fixed-charge transportation problem, which extends the transportation problem by the consideration of fixed costs for sending a flow from some origin to a destination. The reduced costs from the lower bounding are used to guide a restart phase of the algorithm. Lai and Hao [67] apply ILS to obtain high-quality candidate solutions to the maximally diverse grouping problem, where the goal is to partition the vertices of an edge-weighted and undirected complete graph under some constraints on the group size. Wolf and Merz [119] develop an ILS algorithm to find a symmetric connectivity topology with minimum power consumption in wireless ad hoc networks. Benchmark results on large instances with up to 1000 nodes show that their algorithm outperforms other heuristics that were previously used to solve this problem. ILS has been applied to image analysis tasks and Cordón and Damas [23] applied it to image registration, obtaining promising initial results. Imamichi et al. [59] apply ILS as a method to improve the placement of irregular polygons in a rectangular surface with the goal of minimizing, for a fixed width, the length of the container so that no polygon

overlaps with any other or protrudes the container. The proposed approach allows overlaps but tries to minimize them using nonlinear programming methods and swaps between polygons. Few papers have considered the adoption of the ideas underlying ILS for tackling continuous optimization problems. Kramer [65] has explored a simple continuous optimization algorithm, which embedded Powell's methods into an ILS for continuous optimization. Liao and Stützle [69] use ILS as one of the component algorithms in their hybrid, competition-based approach for continuous optimization. This approach decides in a preliminary competition phase which of two continuous optimizers, an ILS algorithm or IPOP-CMAES [6] to execute and then, in a second phase, continues with the execution of the winning algorithm. This approach was a winner of the CEC 2013 benchmark competition for real-parameter optimization.

Conclusions

In the initial phases of metaheuristic research, many efforts were dedicated toward developing and refining new metaheuristic methods and on studying their performance and behavior. As such, the focus was on rather pure applications of the respective methods to tackle computationally hard problems. Iterated local search contributed by relying on a clear principle that is easy to identify and that leads usually to high-performing algorithms. In addition, ILS algorithms are relatively intuitive to design and at the same time malleable. Therefore, iterated local search served often as a basis for a larger algorithm engineering effort if high-performing heuristics are desired. This approach was very successful, as shown by the various problems for which ILS algorithms have been or still are state of the art. In later research efforts in the metaheuristics area, often hybrid methods were proposed that mix in one algorithm concepts from different principles to derive effective algorithms [15, 76]. Even if this trend somehow blurs the differences behind the methods, it is nevertheless important to have methods that rely on clear principles and to have knowledge of their particular strengths, which can be exploited in the design of hybrid methods.

The research in heuristic search methodologies and metaheuristics, in particular, has now reached a mature state, and fundamentally new ideas appear more and more rarely. Within this context, ILS has a clear role as one of the main methods that offers an excellent trade-off between simplicity and flexibility. As future trends we expect on one side that the number of applications of ILS increases further, and that more complex problems and also non-combinatorial problems may be tackled by it. On the other side, as the development of effective heuristic algorithms is becoming increasingly more streamlined, we expect that the clear principles underlying ILS support well a sound algorithm engineering effort. While traditionally such an algorithm engineering effort relied in part on the manual configuration and tuning of the algorithms, over the recent years, the advent of automatic algorithm configuration tools [52, 106] such as ParamILS [56], SMAC [57], or irace [71, 72] has helped to alleviate the algorithm designer from the manual algorithm design

and parameter tuning tasks. In fact, we foresee that in the future, the development of effective ILS algorithms and, more in general, of any other metaheuristic algorithms will be strongly based on the exploitation of automatic algorithm configuration techniques.

Acknowledgements This work received support from the COMEX project within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a senior research associate. Rubén Ruiz is partially supported by the Spanish Ministry of Economy and Competitiveness, under the project "SCHEYARD – Optimization of Scheduling Problems in Container Yards" (No. DPI2015-65895-R) financed by FEDER funds.

Cross-References

▸ GRASP
▸ Iterated Greedy
▸ Memetic Algorithms
▸ Tabu Search
▸ Variable Neighborhood Search

References

1. Aarts EHL, Lenstra JK (eds) (1997) Local Search in Combinatorial Optimization. John Wiley & Sons, Chichester, UK
2. Ahuja RK, Ergun O, Orlin JB, Punnen AP (2002) A survey of very large-scale neighborhood search techniques. Discret Appl Math 123(1–3):75–102
3. Applegate D, Bixby RE, Chvátal V, Cook WJ (1999) Finding tours in the TSP. Technical report 99885, Forschungsinstitut für Diskrete Mathematik, University of Bonn
4. Applegate D, Cook WJ, Rohe A (2003) Chained Lin-Kernighan for large traveling salesman problems. INFORMS J Comput 15(1):82–92
5. Applegate D, Bixby RE, Chvátal V, Cook WJ, Espinoza D, Goycoolea M, Helsgaun K (2009) Certification of an optimal TSP tour through 85,900 cities. Oper Res Lett 37(1):11–15
6. Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Proceedings of CEC 2005. IEEE Press, Piscataway, pp 1769–1776
7. Bäck T, Fogel DB, Michalewicz Z (1997) Handbook of evolutionary computation. IOP Publishing Ltd., Bristol
8. Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput 6(2):126–140
9. Battiti R, Brunato M, Mascia F (2008) Reactive search and intelligent optimization. Operations research/computer science interfaces, vol 45. Springer, New York
10. Baum EB (1986) Iterated descent: a better algorithm for local search in combinatorial optimization problems, manuscript
11. Baum EB (1986) Towards practical "neural" computation for combinatorial optimization problems. In: Neural networks for computing. AIP conference proceedings, pp 53–64
12. Baxter J (1981) Local optima avoidance in depot location. J Oper Res Soc 32(9):815–819
13. Benlic U, Hao JK (2013) Breakout local search for the quadratic assignment problem. Appl Math Comput 219(9):4800–4815
14. Bentley JL (1992) Fast algorithms for geometric traveling salesman problems. ORSA J Comput 4(4):387–411

15. Blum C, Puchinger J, Raidl GR, Roli A (2011) Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput 11(6):4135–4151
16. Brucker P, Hurink J, Werner F (1996) Improving local search heuristics for some scheduling problems—part I. Discret Appl Math 65(1–3):97–122
17. Brucker P, Hurink J, Werner F (1997) Improving local search heuristics for some scheduling problems—part II. Discret Appl Math 72(1–2):47–69
18. Buson E, Roberti R, Toth P (2014) A reduced-cost iterated local search heuristic for the fixed-charge transportation problem. Oper Res 62(5):1095–1106
19. Cattaruzza D, Absi N, Feillet D, Vigo D (2014) An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. Comput Oper Res 51:257–267
20. Cerný V (1985) A thermodynamical approach to the traveling salesman problem. J Optim Theory Appl 45(1):41–51
21. Codenotti B, Manzini G, Margara L, Resta G (1996) Perturbation: an efficient technique for the solution of very large instances of the Euclidean TSP. INFORMS J Comput 8(2):125–133
22. Congram RK, Potts CN, van de Velde S (2002) An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. INFORMS J Comput 14(1): 52–67
23. Cordón O, Damas S (2006) Image registration with iterated local search. J Heuristics 12(1–2): 73–94
24. Corte AD, Sörensen K (2016) An iterated local search algorithm for water distribution network design optimization. Networks 67(3):187–198
25. Della Croce F, Garaix T, Grosso A (2012) Iterated local search and very large neighborhoods for the parallel-machines total tardiness problem. Comput Oper Res 39(6):1213–1217
26. den Besten ML, Stützle T, Dorigo M (2001) Design of iterated local search algorithms: an example application to the single machine total weighted tardiness problem. In: Boers EJW et al (eds) Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2001. LNCS, vol 2037. Springer, Heidelberg, pp 441–452
27. Dong X, Huang H, Chen P (2009) An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. Comput Oper Res 36(5):1664–1669
28. Dong X, Ping, Huang H, Nowak M (2013) A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time. Comput Oper Res 40(2): 627–632
29. Dorigo M, Stützle T (2004) Ant Colony Optimization. MIT Press, Cambridge, MA
30. Essafi I, Mati Y, Dauzère-Pérés S (2008) A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Comput Oper Res 35(8):2599–2616
31. Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6:109–113
32. Fernandez-Viagas V, Framinan JM (2014) On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. Comput Oper Res 45:60–67
33. Fernandez-Viagas V, Ruiz R, Framiñan JM (2017) A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. Eur J Oper Res 257(3):707–721. doi:10.1016/j.ejor.2016.09.055
34. Fischetti M, Monaci M, Salvagnin D (2012) Three ideas for the quadratic assignment problem. Oper Res 60(4):954–964
35. Framinan JM, Gupta JN, Leisten R (2004) A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. J Oper Res Soc 55(12): 1243–1255
36. Geiger MJ (2011) Decision support for multi-objective flow shop scheduling by the Pareto iterated local search methodology. Comput Ind Eng 61(3):805–812
37. Gendreau M, Potvin JY (eds) (2010) Handbook of metaheuristics. International series in operations research & management science, vol 146, 2nd edn. Springer, New York
38. Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549

39. Glover F, Kochenberger G (eds) (2002) Handbook of metaheuristics. Kluwer Academic Publishers, Norwell
40. Glover F, Laguna M (1997) Tabu search. Kluwer Academic Publishers, Boston
41. Glover F, Laguna M, Martí R (2002) Scatter search and path relinking: advances and applications. In: [38], pp 1–35
42. Grasas A, Juan AA, Lourenço HR (2016) SimILS: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. J Simul 10(1):69–77
43. Grosso A, Della Croce F, Tadei R (2004) An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. Oper Res Lett 32(1):68–72
44. Grosso A, Jamali ARMJU, Locatelli M (2009) Finding maximin Latin hypercube designs by iterated local search heuristics. Eur J Oper Res 197(2):541–547
45. Hansen P, Mladenović N (2001) Variable neighborhood search: principles and applications. Eur J Oper Res 130(3):449–467
46. Hansen P, Mladenović N, Brimberg J, Pérez JAM (2010) Variable neighborhood search. In: [36], pp 61–86
47. Hashimoto H, Yagiura M, Ibaraki T (2008) An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. Discret Optim 5(2):434–456
48. Hejazi SR, Saghafian S (2005) Flowshop-scheduling problems with makespan criterion: a review. Int J Prod Res 43(14):2895–2929
49. Helsgaun K (2000) An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur J Oper Res 126:106–130
50. Helsgaun K (2009) General *k*-opt submoves for the Lin-Kernighan TSP heuristic. Math Program Comput 1(2–3):119–163
51. Hong I, Kahng AB, Moon BR (1997) Improved large-step Markov chain variants for the symmetric TSP. J Heuristics 3(1):63–81
52. Hoos HH (2012) Automated algorithm configuration and parameter tuning. In: Hamadi Y, Monfroy E, Saubion F (eds) Autonomous search. Springer, Berlin, pp 37–71
53. Hoos HH, Stützle T (2005) Stochastic local search—foundations and applications. Morgan Kaufmann Publishers, San Francisco
54. Hurtgen M, Maun JC (2010) Optimal PMU placement using iterated local search. Int J Electr Power Energy Syst 32(8):857–860
55. Hussin MS, Stützle T (2009) Hierarchical iterated local search for the quadratic assignment problem. In: Blesa MJ et al (eds) Hybrid metaheuristics. LNCS, vol 5818. Springer, Heidelberg, pp 115–129
56. Hutter F, Hoos HH, Leyton-Brown K, Stützle T (2009) ParamILS: an automatic algorithm configuration framework. J Artif Intell Res 36:267–306
57. Hutter F, Hoos HH, Leyton-Brown K (2011) Sequential model-based optimization for general algorithm configuration. In: Coello Coello CA (ed) LION 5. LNCS, vol 6683. Springer, Heidelberg, pp 507–523
58. Ibaraki T, Imahori S, Nonobe K, Sobue K, Uno T, Yagiura M (2008) An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. Discret Appl Math 156(11):2050–2069
59. Imamichi T, Yagiura M, Nagamochi H (2009) An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. Discret Optim 6(4):345–361
60. Johnson DS (1990) Local optimization and the traveling salesman problem. In: Paterson M (ed) 17th international colloquium on Automata, languages and programming. LNCS, vol 443. Springer, Heidelberg, pp 446–461
61. Johnson DS, McGeoch LA (1997) The traveling salesman problem: a case study in local optimization. In: Aarts EHL, Lenstra JK (eds) Local search in combinatorial optimization. John Wiley & Sons, Chichester, pp 215–310
62. Juan AA, Lourenço HR, Mateo M, Luo R, Castellà Q (2014) Using iterated local search for solving the flow-shop problem: parallelization, parametrization, and randomization issues. Int Trans Oper Res 21(1):103–126

63. Katayama K, Narihisa H (1999) Iterated local search approach using genetic transformation to the traveling salesman problem. In: Banzhaf W et al (eds) Proceedings of GECCO 1999, vol 1. Morgan Kaufmann Publishers, San Francisco, pp 321–328
64. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
65. Kramer O (2010) Iterated local search with Powell's method: a memetic algorithm for continuous global optimization. Memetic Comput 2(1):69–83
66. Kreipl S (2000) A large step random walk for minimizing total weighted tardiness in a job shop. J Sched 3(3):125–138
67. Lai X, Hao JK (2016) Iterated maxima search for the maximally diverse grouping problem. Eur J Oper Res 254(3):780–800
68. Laurent B, Hao JK (2009) Iterated local search for the multiple depot vehicle scheduling problem. Comput Ind Eng 57(1):277–286
69. Liao T, Stützle T (2013) Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization. In: Proceedings of CEC 2013. IEEE Press, Piscataway, pp 1938–1944
70. Lin S, Kernighan B (1973) An effective heuristic algorithm for the traveling salesman problem. Oper Res 21(2):498–516
71. López-Ibáñez M, Dubois-Lacoste J, Stützle T, Birattari M (2011) The irace package, iterated race for automatic algorithm configuration. Technical report, TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles. http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf
72. López-Ibáñez M, Dubois-Lacoste J, Pérez Cáceres L, Stützle T, Birattari M (2016) The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspectives 3: 43–58
73. Lourenço HR (1995) Job-shop scheduling: computational study of local search and large-step optimization methods. Eur J Oper Res 83(2):347–364
74. Lourenço HR, Martin O, Stützle T (2002) Iterated local search. In: [38], pp 321–353
75. Lourenço HR, Martin O, Stützle T (2010) Iterated local search: framework and applications. In: [36], chap 9, pp 363–397
76. Maniezzo V, Stützle T, Voß S (eds) (2009) Matheuristics—hybridizing metaheuristics and mathematical programming. Annals of information systems, vol 10. Springer, New York
77. Martin O, Otto SW (1995) Partitioning of unstructured meshes for load balancing. Concurr Pract Exp 7(4):303–314
78. Martin O, Otto SW (1996) Combining simulated annealing with local search heuristics. Ann Oper Res 63:57–75
79. Martin O, Otto SW, Felten EW (1991) Large-step Markov chains for the traveling salesman problem. Complex Syst 5(3):299–326
80. Martin O, Otto SW, Felten EW (1992) Large-step Markov chains for the TSP incorporating local search heuristics. Oper Res Lett 11(4):219–224
81. Mati Y, Dauzère-Pèrés S, Lahlou C (2011) A general approach for optimizing regular criteria in the job-shop scheduling problem. Eur J Oper Res 212(1):33–42
82. Melo Silva M, Subramanian A, Ochi LS (2015) An iterated local search heuristic for the split delivery vehicle routing problem. Comput Oper Res 53:234–249
83. Merz P, Freisleben B (2000) Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Trans Evol Comput 4(4):337–352
84. Merz P, Huhse J (2008) An iterated local search approach for finding provably good solutions for very large TSP instances. In: Rudolph G et al (eds) PPSN X. LNCS, vol 5199. Springer, Heidelberg, pp 929–939
85. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller A, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21:1087–1092
86. Michallet J, Prins C, Yalaoui F, Vitry G (2014) Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. Comput Oper Res 41:196–207

87. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
88. Morris P (1993) The breakout method for escaping from local minima. In: Proceedings of the 11th National Conference on Artificial Intelligence. AAAI Press/MIT Press, Menlo Park, pp 40–45
89. Moscato P (1999) Memetic algorithms: a short introduction. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw Hill, London, pp 219–234
90. Mühlenbein H (1991) Evolution in time and space—the parallel genetic algorithm. In: Rawlins G (ed) Foundations of genetic algorithms (FOGA). Morgan Kaufmann Publishers, San Mateo, pp 316–337
91. M'Hallah R (2014) An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop. Int J Prod Res 52(13):3802–3819
92. Nawaz M, Enscore E Jr, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11(1):91–95
93. Nguyen VP, Prins C, Prodhon C (2012) A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. Eng Appl Artif Intell 25(1): 56–71
94. Palhazi Cuervo D, Goos P, Sörensen K, Arráiz E (2014) An iterated local search algorithm for the vehicle routing problem with Backhauls. Eur J Oper Res 237(2):454–464
95. Pan QK, Ruiz R (2012) Local search methods for the flowshop scheduling problem with flowtime minimization. Eur J Oper Res 222(1):31–43
96. Resende MGC, Ribeiro CC (2010) Greedy randomized adaptive search procedures: advances, hybridizations, and applications. In: [36], pp 283–319
97. Ribas I, Companys R, Tort-Martorell X (2013) An efficient iterated local search algorithm for the total tardiness blocking flow shop problem. Int J Prod Res 51(17):5238–5252
98. Ruiz R, Maroto C (2005) A comprehensive review and evaluation of permutation flowshop heuristics. Eur J Oper Res 165(2):479–494
99. Ruiz R, Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 177(3):2033–2049
100. Schreiber GR, Martin O (1999) Cut size statistics of graph bisection heuristics. SIAM J Optim 10(1):231–251
101. Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget JF (eds) CP'98. LNCS, vol 1520. Springer, Heidelberg, pp 417–431
102. Stützle T (1998) Applying iterated local search to the permutation flow shop problem. Technical report AIDA–98–04, FG Intellektik, FB Informatik, TU Darmstadt
103. Stützle T (1998) Local search algorithms for combinatorial problems—analysis, improvements, and new applications. PhD thesis, FB Informatik, TU Darmstadt
104. Stützle T (2006) Iterated local search for the quadratic assignment problem. Eur J Oper Res 174(3):1519–1539
105. Stützle T, Hoos HH (2001) Analysing the run-time behaviour of iterated local search for the travelling salesman problem. In: Hansen P, Ribeiro C (eds) Essays and surveys on metaheuristics. Kluwer Academic Publishers, Boston, pp 589–611
106. Stützle T, López-Ibáñez M (2015) Automatic (offline) configuration of algorithms. In: Laredo JLJ, Silva S, Esparcia-Alcázar AI (eds) GECCO (Companion). ACM Press, New York, pp 681–702
107. Stützle T, Birattari M, Hoos HH (eds) (2007). SLS 2007. LNCS, vol 4638. Springer, Heidelberg
108. Subramanian A, Battarra M, Potts CN (2014) An iterated local search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. Int J Prod Res 52(9):2729–2742
109. Taillard ÉD (1990) Some efficient heuristic methods for the flow shop sequencing problem. Eur J Oper Res 47(1):65–74

110. Taillard ÉD (1995) Comparison of iterative searches for the quadratic assignment problem. Locat Sci 3(2):87–105
111. Thierens D (2004) Population-based iterated local search: restricting the neighborhood search by crossover. In: Deb K et al (eds) GECCO 2004, part II. LNCS, vol 3103. Springer, Heidelberg, pp 234–245
112. Ulder NLJ, Aarts EHL, Bandelt HJ, van Laarhoven PJM, Pesch E (1991) Genetic local search algorithms for the travelling salesman problem. In: Schwefel HP, Männer R (eds) Proceedings of PPSN-I. LNCS, vol 496. Springer, Heidelberg, pp 109–116
113. Urlings T, Ruiz R, Stützle T (2010) Shifting representation search for hybrid flexible flowline problems. Eur J Oper Res 207(2):1086–1095
114. Vaessens RJM, Aarts EHL, Lenstra JK (1998) A local search template. Comput Oper Res 25(11):969–979
115. Vansteenwegen P, Mateo M (2014) An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. Eur J Oper Res 237(3):802–813
116. Vansteenwegen P, Souffriau W, Berghe GV, Oudheusden DV (2009) Iterated local search for the team orienteering problem with time windows. Comput Oper Res 36(12):3281–3290
117. Vaz Penna PH, Subramanian A, Ochi LS (2013) An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J Heuristics 19(2):201–232
118. Voudouris C, Tsang EPK (2002) Guided local search. In: [38], pp 185–218
119. Wolf S, Merz P (2009) Iterated local search for minimum power symmetric connectivity in wireless networks. In: Cotta C, Cowling P (eds) Proceedings of EvoCOP 2009. LNCS, vol 5482. Springer, Heidelberg, pp 192–203
120. Xu H, Lü Z, Cheng TCE (2014) Iterated local search for single-machine scheduling with sequence-dependent setup times to minimize total weighted tardiness. J Sched 17(3): 271–287
121. Yang Y, Kreipl S, Pinedo M (2000) Heuristics for minimizing total weighted tardiness in flexible flow shops. J Sched 3(2):89–108