Hybridizing Different Local Search Algorithms with Each Other and Evolutionary Computation: Better Performance on the Traveling Salesman Problem

Yuezhong Wu, Thomas Weise*, and Weichen Liu
UBRI, School of Computer Science and Technology, University of Science and Technology of China
Hefei, Anhui, China, 230027, tweise@mail.ustc.edu.cn. * Dr. Weise is the corresponding author.

ABSTRACT

We propose the new concept of hybridizing different local search algorithms with each other for the TSP. The new hybrids outperform their component algorithms. We then hybridize them with an Evolutionary Algorithm and Population-based ACO. The resulting EC-LS-LS hybrids perform even better.

Keywords

Local Search; Hybrid Algorithms; Traveling Salesman Problem; Memetic Algorithm; Ant Colony Optimization

1. INTRODUCTION

The Traveling Salesman Problem (TSP) [2] is defined as follows: Given n cities, a salesman departs from a start city, visits each city exactly once, and then returns back to the start city. The task is to find the city visiting order resulting in the minimal overall travel distance. Many algorithms have been applied to the TSP, from Evolutionary Computation (EC) [35] to exact methods like Branch and Bound [23]. We make the following contributions: We introduce the new concept of the LS-LS hybrids (for the TSP), i.e., hybrid algorithms combining two local search (LS) methods. We explore several LS-LS hybrids combining Multi-Neighborhood Search (MNS) [36], the Lin-Kernighan Heuristic (LK) [26], and FSM** [28]. We find that they almost always significantly outperform their LS components. We further hybridize our LS-LS hybrids with both Population-based ACO (PACO) [15] and Evolutionary Algorithms (EAs) [35]. We find that the resulting EC-LS-LS algorithms perform better than any other method in our experiments and in [28, 36, 38].

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.
DOI: http://dx.doi.org/10.1145/2908961.2909001

2. INVESTIGATED ALGORITHMS

LS algorithms for the TSP start at a random or heuristically-generated solution (tour). They remember the best solution discovered so far and try to improve it step by step. If a local optimum is reached, the LS applies a larger random modification. MNS is a LS that, in each iteration, scans several neighborhoods of the current solution and enqueues all possible improving moves. The best move is carried out. All invalidated intersecting moves are dropped and the remaining best move is applied. If the queue becomes empty, another scan is performed. If no improving moves can be found, a random sub-sequence of the current tour is randomly shuffled, which we refer to as soft restart. The LK heuristic dominates today's TSP research. We use the implementation from [38]. The Ejection Chain Method (ECM) FSM** [28] is an improvement of [32]. Although both LK and FSM** outperform MNS, the hybrids of MNS with Evolutionary Algorithms (EAs) and the Population-based Ant Colony Optimization (PACO) outperform similar hybrids based on them [28, 38].

Research on hybrid ("Memetic") algorithms is almost entirely focused on combining global and local search algorithms (EC-LS), as done in [28, 36, 38, 40]. However, LS algorithms can already exhibit different behaviors which might complement each other. MNS can find relatively good solutions quickly but often gets stuck in local optima. LK and FSM** initially are slower but find better final results [28, 38]. We pairwise hybridize LK, FSM**, and MNS with each other. We apply one LS approach until it cannot improve the solution anymore. The resulting tour is then passed as starting point to the other LS. Once this second LS gets trapped in a local optimum, we use its result as starting point again for the first LS. This is repeated until both LS methods cannot find an improvement, in which case we apply the same soft restart method as above. This is a generalization of Variable Neighborhood Search (VNS) [18], which explores the neighborhood of the current solution until it reaches a local optimum. It then uses another search operator to escape from it. With our new LS-LS hybrids, we extend the concept to whole LS algorithms instead of just search operators.

We investigate MA(μ $\dot{+}$ λ)-LS and MA(μ $\dot{+}$ λ)-LS-LS, which combine (μ $\dot{+}$ λ) EAs with LS and LS-LS, respectively. The first populations of our MAs stem from the Edge-Greedy, Double Minimum Spanning Tree, Savings, Double-Ended Nearest Neighbor, and Nearest Neighbor Heuristic [36]. We apply Edge Crossover [37] at a crossover rate of 1. The LS component of the MA is applied to every so-

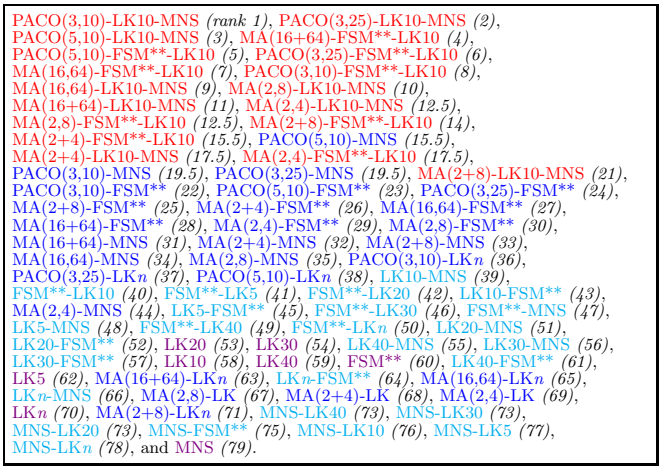


Figure 1: Algorithm ranking from best to worst, based on various performance measures and statistics (see [36] for details). The different algorithm types pure local search, LS-LS hybrid, EC-LS hybrid, and EC-LS-LS hybrid are highlighted.

lution generated, both by the initialization heuristics and crossover. We also hybridize PACO(k,l) with LS and LS-LS. Here, in each iteration, l tours are created in the same way as in the standard ACO. The "oldest" solution in the archive of size k is replaced by the best of the newly generated solutions. The pheromone on an edge is proportional to the number of times the edge is contained in the archive. The initial populations are again obtained heuristically, in the same way as in the MAs.

3. RESULTS AND DISCUSSION

We perform 30 independent runs for 79 algorithm setups on all 110 symmetric *TSPLib* [33] benchmark cases. We define the following LS setups: FSM**, MNS, and 6 setups of the LK heuristic, differing in their candidate set size $s =\in \{5, 10, 20, 30, 40, n\}$ and named LKs, i.e., LK5, LK10, ..., LKn, respectively. We compare them to 26 LS-LS hybrids, whose name consists of the two component LS algorithms in the cyclically applied sequence. We construct EC-LS and EC-LS-LS hybrids which are named after the EC method followed by the applied LS. In Figure 6, we present an abridge algorithm performance ranking generated by the *TSP Suite* [36].Our experiments have led us to four major conclusions:

1. The new LS-LS hybrids are better than their pure LS algorithm components. This means that the new idea of combining the strengths of different LS algorithms is very promising.
2. The LS-LS hybrids are still slightly worse than the EC-LS algorithms. This means that a global search component is necessary in a good TSP solver.
3. The new EC-LS-LS hybrids outperform the LS-LS algorithms as well as EC-LS hybrids. The best overall algorithm, PACO(3,10)-LK10-MNS, unites the global search strength of PACO, the ability to find good solutions of the LK heuristic, and the fast exploitation speed of MNS.
4. In [28, 36, 38, 40] as well as the present study, PACO is the significantly better host EC method for hybridization than an EA. However, we also find an exception to this rule, as MAs with FSM**-LK10 are better than the corresponding PACO versions.

Acknowledgments We acknowledge support from the Fundamental Research Funds for the Central Universities and the National Natural Science Foundation of China under Grants 61150110488 and 71520107002.

References

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton, NJ, USA: Princeton University Press, 2007.

[15] M. Guntsch and M. Middendorf. Applying population based aco to dynamic optimization problems. In *3rd Intl. Workshop on Ant Colony Optimization (ANTS'02)*, pages 111–122, Brussels, Belgium, Sept. 12–14, 2002. Berlin, Germany: Springer-Verlag GmbH.

[18] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *Europ. Journal of Operational Res. (EJOR)*, 130(3):449–467, 2001.

[23] Y. Jiang, T. Weise, J. Lässig, R. Chiong, and R. Athauda. Comparing a hybrid branch and bound algorithm with evolutionary computation methods, local search and their hybrids on the tsp. In *IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS'14)*, Orlando, FL, USA, Dec. 9–12, 2014. Los Alamitos, CA, USA: IEEE Computer Society Press. doi:10.1109/CIPLS.2014.7007174.

[26] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, Mar.–Apr. 1973.

[28] W. Liu, T. Weise, Y. Wu, and R. Chiong. Hybrid ejection chain methods for the traveling salesman problem. In *10th Intl. Conf. Bio-Inspired Computing – Theories and Applications (BIC-TA'15)*, volume 562 of *Communications in Computer and Information Science*, pages 268–282, Hefei, Anhui, China, Sept. 25–28, 2015. Berlin/Heidelberg: Springer-Verlag. doi:10.1007/978-3-662-49014-3_25.

[32] C. Rego. Relaxed tours and path ejections for the traveling salesman problem. *European Journal of Operational Research*, 106(2):522–538, 1998.

[33] G. Reinelt. Tsplib – a traveling salesman problem library. *ORSA Journal on Comp.*, 3(4):376–384, 1991.

[35] T. Weise. *Global Optimization Algorithms – Theory and Application*. Germany: it-weise.de (self-published), 2009. URL http://www.it-weise.de/projects/book.pdf.

[36] T. Weise, R. Chiong, K. Tang, J. Lässig, S. Tsutsui, W. Chen, Z. Michalewicz, and X. Yao. Benchmarking optimization algorithms: An open source framework for the traveling salesman problem. *IEEE Computational Intelligence Magazine (CIM)*, 9(3):40–52, Aug. 2014. doi:10.1109/MCI.2014.2326101.

[37] L. D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In *3rd Intl. Conf. on Genetic Algorithms (ICGA'89)*, pages 133–140, Fairfax, VA, USA, June 4–7, 1989. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[38] Y. Wu, T. Weise, and R. Chiong. Local search for the traveling salesman problem: A comparative study. In *Proc. of 14th IEEE Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC'15)*, pages 213–220, Beijing, China, July 6–8, 2015. doi:10.1109/ICCI-CC.2015.7259388.

[39] Y. Wu, T. Weise, and W. Liu. Hybridizing different local search algorithms with each other and

evolutionary computation: Better performance on the traveling salesman problem. In *Proceedings of the 18th Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM Press: New York, NY, USA, July20–24, 2016. ISBN 978-1-4503-4323-7/16/07. doi:10.1145/2908961.2909001. URL http://www.it-weise.de/research/publications/WWL2016HDLSAWEOAECBPOTTSP/WWL2016HDLSAWEOAECBPOTTSP.pdf. accepted for publication as short paper.

[40] D. Xu, T. Weise, Y. Wu, J. Lässig, and R. Chiong. An investigation of hybrid tabu search for the traveling salesman problem. In *10th Intl. Conf. Bio-Inspired Computing – Theories and Applications (BIC-TA'15)*, volume 562 of *Communications in Computer and Information Science*, pages 523–537. Berlin/Heidelberg: Springer-Verlag, Hefei, Anhui, China, Sept. 25–28, 2015. doi:10.1007/978-3-662-49014-3_47.

Unpublished Full Version of Short Paper [39]: Hybridizing Different Local Search Algorithms with Each Other and Evolutionary Computation: Better Performance on the Traveling Salesman Problem

ABSTRACT

The Traveling Salesman Problem (TSP) is one of the most well-studied combinatorial optimization problems. The best heuristics for solving TSPs today are based either on the Lin-Kernighan (LK) heuristic or are Ejection Chain Methods (ECMs), both of which are local search (LS) algorithms. Multi-Neighborhood Search (MNS) is another LS algorithm and especially suitable for hybridization with Evolutionary Computation (EC) algorithms. Existing studies on hybrid TSP solvers focus mainly on Memetic Algorithms (MAs), i.e., EC-LS hybrids. We introduce the new concept of LS-LS hybrids in order to combine the different strengths of multiple LS algorithms. We pairwise hybridize several setups of the three state-of-the-art local searches above. We then further hybridize these LS-LS algorithms with Evolutionary Algorithms (EAs) and Population-based Ant Colony Optimization (PACO), i.e., obtain EC-LS-LS algorithms. We conduct a large-scale experimental study with 79 different algorithm setups on all 110 symmetric instances of the *TSPLib* benchmark set. We find that most of the new LS-LS hybrids have better performance than their components alone and that the new EC-LS-LS hybrids are the strongest TSP solvers in our study. They outperform both LS-LS algorithms and hybrids that combine only one LS with an EC algorithm.

Keywords

Local Search, Hybrid Algorithms, Traveling Salesman Problem, Memetic Algorithm, Ant Colony Optimization

1. INTRODUCTION

The Traveling Salesman Problem (TSP) [2, 16, 25] is maybe the most important $\mathcal{NP}$-hard problem, both in terms of practical applications as well as being a test bed for novel optimization approaches [29]. Given n cities, a salesman departs from a start city, visits each city exactly once, and then returns back to the start city. The task is to find the city visiting order resulting in the minimal overall travel distance. In other words, given a cost matrix $D = (D_{i,j})$, where $D_{i,j}$ is the distance of going from city i to city j $(i, j \in 1 \dots n)$, the goal is to find a permutation t of the integers from 1 to n minimizing the sum $D_{t[1],t[2]} + D_{t[2],t[3]} + \cdots + D_{t[n],t[1]}$. The focus of this study is the symmetric TSP, where $D_{i,j} = D_{j,i}$ holds.

Many optimization algorithms have been applied to the TSP, including Local Search (LS) algorithms [13, 26], Evolutionary Algorithms (EAs) [3, 7, 35], Ant Colony Optimization (ACO) [8, 9, 11], and Branch and Bound (BB) [23, 27]. Today, the best known LS approaches for the TSP are either based on the Lin-Kernighan (LK) heuristic [26] or Ejection Chain Methods (ECM) [13]. Memetic Algorithms (MAs) [31], hybrid algorithms which combine the global search ability of an EA with the exploitation strength of such a LS, are known to be especially efficient [29].

Using other Evolutionary Computation (EC) approaches as host global search algorithms for hybridization is less common, although promising: Wu et al. [38] compared several hybrid and pure versions of the LK heuristic both with EAs and Population-based ACO (PACO) [15]. The PACO-LK performed best. Liu et al. [28] proposed the Fundamental Stem and Cycle Method** (FSM**), an improved ECM based on the P_SEC algorithm [32]. Similar EA and PACO hybrid methods were defined and those with PACO performed best. They also outperformed the above-mentioned PACO-LK [28]. Multi-Neighborhood Search (MNS) [36], another LS, is especially suitable for hybridization. Combined with PACO, it outperformed both of the above algorithms [28, 38].

To the best of our knowledge, however, there exists no work combining different LS algorithms to exploit their mutual advantages. Using the TSP as testbed, we explore this idea. With the present work, we make the following contributions:

1. We introduce the new concept of the LS-LS hybrids.
2. We explore several LS-LS hybrids combining MNS, LK, and FSM**. We find that they almost always significantly outperform their LS components.
3. We then hybridize our LS-LS hybrids with both PACO and EAs. We find that the resulting EC-LS-LS algorithms perform better than any other method in our experiments and in [28, 36, 38].
4. We conduct an in-depth statistically comparison of all the above algorithms based on a large-scale experimental study, where 79 algorithm setups are applied to all 110 symmetric benchmark instances from *TSPLib* [33]. We apply runtime-behavior based statistics which provide more information than simple end-result comparisons. This study gathers (and combines from related works) more precise information, includes more experiments, and provides more statistically sound evaluations than any other previous studies along this line of research.
5. In this large-scale study, we find that a combination of PACO, the LK heuristic, and MNS, i.e., a EC-LS-LS hybrid algorithm, performs best. It unites the global search strength of PACO, the ability to find good solutions of the LK heuristic, and the fast exploitation speed of MNS.

The remainder of this paper is organized as follows. In Section 2, we introduce the investigated LK, FSM**, and MNS algorithms, as well as our new LS-LS and EC-LS-LS hybrids. We then present our experimental study and discuss its results in Section 3. Finally, the paper ends with conclusions and plans for future work in Section 4.

2. INVESTIGATED ALGORITHMS

2.1 Local Search

LS algorithms for the TSP start at a random or heuristically-generated solution (tour). They remember the best solution discovered so far and try to improve it step

by step. A tour can be considered as cyclic path consisting of n edges. A search operation which replaces m of these n edges is called m-opt move [36]. Exchanging of two cities in the tour corresponds to replacing of (at most) four edges (4-opt) [24, 30]. Rotating a sub-sequence of cities to the left or right is a 3-opt [10, 24] move. The reversal of a sub-sequence of a tour, the maybe most common operator in a LS for the TSP, is a 2-opt move [21, 24]. If the LS concludes that it cannot further improve its best tour, it may apply a larger random modification in order to escape from the local optimum, while remembering the best overall solution in an additional variable. This process is repeated until a termination criterion is reached.

2.2 The Lin-Kernighan Algorithm

The LK heuristic is a LS approach published by Lin and Kernighan [26] in 1973. Its derivatives dominate today's TSP research and many improvements have been proposed. When the Chained Lin-Kernighan (CLK) algorithm [1] arrives at a local optimum from which it cannot escape, it generates a new solution by a random 4-opt move instead of restarting at a random solution. CLK performs particularly well on TSPs with a large number n of cities. Meanwhile, the Lin-Kernighan-Helsgaun (LKH) algorithm [19, 20] may be the most efficient LK variant.

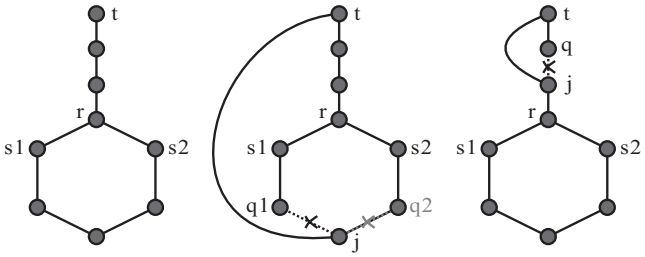
LK can be considered as a variable m-opt LS. For ascending values of m, the algorithm tests whether replacing m edges may achieve a shorter tour. Let T be the current tour and N the set containing all cities. In each iteration, the algorithm constructs the sets $X = \{X_1, \ldots, X_m\}$ of edges to be deleted from T and $Y = \{Y_1, \ldots, Y_m\}$, the set of edges to be added to T, such that the resulting tour would be valid and shorter. The interchange of these edges is then a m-opt move. In the beginning, X and Y are empty. Pairs of edges are added to X and Y such that the end node of the edge added to X is the starting node of the edge added to Y, whose end node will then become the starting node of the edge added to X in the next iteration, if any.

A necessary but not sufficient condition that the exchange of edges in X and Y results in a valid tour is that the chain is closed, i.e., that the end node of Y_m is the start node of X_1. The sets X and Y must further be disjoint, i.e., no added edge is deleted again and no deleted edge is added.

We use the LK implementation from [38] in our experiments. It replaces the current tour as soon as an improvement was found (and then begins to construct new sets X and Y). This is more efficient [19, 38] than the original LK heuristic, which continues exploring all possible moves to find an even shorter tour. We furthermore use candidate sets [26], i.e., limit the choices of neighbors for any city in a tour to speed up the algorithm. Finally, when reaching a local optimum, the algorithm uses the soft restart approach described in [36, 38], where a randomly chosen sub-sequence of the current tour is randomly shuffled. An in-depth description of the LK heuristic used in our investigation can be found in [38], while [19, 20] provides further explanations of the standard LK heuristic and its improvements.

2.3 Ejection Chain Method: FSM**

Most LS approaches applied to the TSP directly search in the space of candidate solutions, i.e., genotypes and phenotypes are the same. This is, for instance, the case in the LK heuristic described above. ECMs, introduced by Glover [13] in 1992, are different. The ECM explored in this work, FSM**, internally works on a Stem-and-Cycle (S&C) reference structure, as illustrated in Figure 2a. This structure consists of a path (called stem) attached to a cycle of nodes. The common node of stem and cycle is called root r and its two adjacent nodes on the cycle are called sub-roots (s_1 and s_2). The root r marks one end of the stem. The other end is called the tip t.



(a) An example of a S&C reference structure. (b) Rule 1 of the Fundamental S&C ECM. (c) Rule 2 of the Fundamental S&C ECM.

Figure 2: Examples of a S&C reference structure and the two rules of the Fundamental S&C ECM.

If the stem is degenerated to become a single node (i.e., $r = t$), the S&C structure becomes a tour. Otherwise, the S&C structure can be transformed to two candidate tours by removing the edge between one of the sub-roots si and the root r, and then re-connecting si to the tip t. The better one of these two trial solutions can be chosen.

During the search, the S&C structure is iteratively refined by two search operators (called "rules") [14, 32]:

1. Choose a node j on the cycle. Let the two nodes adjacent to it be q_1 and q_2. Select one of them and refer to it as q. Delete the edge between q and j and then add edge (t, j).

2. Choose a node j on the stem. Let the node adjacent to j and farther away from r than the other adjacent node be called q. Connect t to j and delete edge (j, q).

In both cases, q becomes the new tip t (see Figures 2b and 2c) but the root r remains unchanged.

In the P_SEC ECM by Rego [32], in each step, the rule and j are chosen which can minimize the overall edge length of the S&C. Glover [13] and Rego [32] found that choosing a different root after a relatively small number of such steps leads to better performance. Therefore, initially, a set R containing all n nodes is created. At the beginning and each time the root is to be changed, a new root is randomly extracted from R.

We investigate the FSM** [28], which improves P_SEC in several aspects. P_SEC defines a tabu criterion that forbids adding an edge again to the S&C which previously was deleted. FSM** applies the less restrictive criterion of not allowing an edge to be deleted a second time (i.e., being deleted, added, and then deleted again from the S&C). The maximum number steps before a root change is set to $0.45n$ and after $0.15n$ nodes have been used as root, the algorithm applies the same soft restart method as the tested LK implementation [36]. Each of these modifications has been confirmed to improve the overall performance [28].

2.4 Multi-Neighborhood Search

Another efficient LS approach for the TSP is the MNS algorithm. In each iteration, MNS performs an $\mathcal{O}(n^2)$ scan that investigates four neighborhoods (city swap, sub-sequence rotate left, sub-sequence rotate right, and reversal) of a tour at once. It tests all pairs $\{i, j\}$ as potential indexes for cities to swap or start and end indexes of sub-sequence rotations and reversals. For each pair $\{i, j\}$, the gain is computed and all discovered improving moves enter a queue. The access to the distance matrix D is minimized by remembering (and updating) the lengths of all n edges in the current tour and avoiding the checking of redundant moves (swapping the cities at index i and $i+1$ is equivalent to a reversal of the sub-sequence from i to $i+1$, for instance). After the scan, the best discovered move is carried out. Doing this may invalidate some other moves in the queue, e.g., if a sub-sequence reversal that overlaps with a potential sub-sequence left rotation was performed. After pruning all invalidated moves from the queue, the remaining best move is carried out, if any. If the queue becomes empty, another scan of the current solution is performed, as new moves may have become possible. During this scan, only moves that at least intersect with the previously modified sub-sequence(s) of the current best solution need to be considered (to speed up the search). If no improving moves can be found anymore, a random sub-sequence of the current tour is randomly shuffled.

This algorithm has performed the best among all of the methods tested in [36]. It is outperformed by both LK and FSM** in their pure form, but its hybrid versions with PACO outperform theirs [28, 38].

2.5 Hybrid Local Search: LS-LS

Research on hybrid ("Memetic") algorithms is almost entirely focused on combining global and local search algorithms (EC-LS), as done in [28, 29, 36, 38, 40], for instance. However, LS algorithms can already exhibit different behaviors which might complement each other. Some LS approaches (like MNS) are efficient to find good solutions quickly but can easily get stuck in local optima. Others (LK, FSM**) might initially be slower but find better final results [28, 38]. ECMs are considered to be able to reach parts of search space which cannot be reached by LK [12].

In order to take advantages of their different abilities, we pairwise hybridize LK, FSM**, and MNS with each other. We therefore apply one LS approaches until it cannot improve the solution anymore. Instead of performing a soft restart, the resulting tour is passed as starting point to the other LS. Once this second LS gets trapped in a local optimum, we use its result as starting point again for the first LS. This is repeated until both LS methods cannot find an improvement, in which case we apply the same soft restart method mentioned before.

In some sense, this idea can be considered as a generalization of Variable Neighborhood Search (VNS) [18]. VNS explores the neighborhood of the current solution, spanned by a search operator such as 2-opt, until it reaches a local optimum. It then uses another search operator (such as 3-opt) to escape from this trap. With LS-LS hybrids, we extend the concept to whole LS algorithms instead of just search operators.

We define six LS-LS hybrids, namely LK-FSM**, FSM**-LK, LK-MNS, MNS-LK, FSM**-MNS, and MNS-FSM**. We do not explore VNS as component here, as it was already found in [36] that MNS can outperform VNS based on the same operators on the TSP.

2.6 Evolutionary and Memetic Algorithms

EAs are the most well-known EC approaches [4, 6]. EAs first generate a set of λ random solutions. Out of these, the best $\mu \leq \lambda$ solutions will be selected as "parents" of the second generation. λ offspring solutions are created by applying either a unary (mutation) or binary (crossover) operator to the parents. From then on, the μ best individuals are selected from the λ offspring solutions and their μ parents in each generation in the case of a $(\mu + \lambda)$-EA. A (μ, λ)-EA selects only from the λ offspring.

We investigate MAs which combine such EAs with LS. The first population of our MAs is not generated randomly, but instead stems from the Edge-Greedy, Double Minimum Spanning Tree, Savings, Double-Ended Nearest Neighbor, and Nearest Neighbor Heuristic, in order to improve their performance [36]. We apply Edge Crossover [37], which generates a new solution by picking edges belonging to either of its two parents, as recombination operator. It is considered to be one of the best crossover operators for the TSP [36]. The crossover rate is set to 1. The LS component of the MA is applied to every solution generated, both by these initialization heuristics and crossover. We propose two families of MAs: MA($\mu \dot{+} \lambda$)-LS and MA($\mu \dot{+} \lambda$)-LS-LS.

2.7 Ant Colony Optimization

The ACO algorithm is an EC approach introduced by Dorigo [8] in 1992. We consider the state-of-the-art ACO variant PACO [15] in our experiments. Different from normal ACO, PACO has linear memory requirements since it does not use a full pheromone matrix. Instead, it maintains a population of k solutions.

The pheromones are defined by the edges occurring in these tours: In each algorithm iteration, l tours are created in the same way as in the standard ACO. The "oldest" solution in the population is replaced by the best of the newly generated solutions. The pheromone on an edge is proportional to the number of times the edge is contained in the population. Only few hybrid ACO approaches have been applied to the TSP, although it was shown in [36] that they perform particularly well. We compare hybrid PACO(k, l)-LS with PACO(k, l)-LS-LS algorithms. The initial populations are again obtained heuristically, in the same way as in the MAs.

3. EXPERIMENTS AND RESULTS

We conduct a large-scale experimental study in which we perform 30 independent runs for 79 algorithm setups on all 110 symmetric *TSPLib* [33] benchmark cases. The maximum computational budget is set to 1 hour per run.

Most metaheuristics, including EAs, MAs, ACO, as well as all LS approaches are anytime algorithms [5]. Even several exact algorithms, such as BB [23], fall into this category. Anytime algorithms can provide a best guess of what the optimal solution of a problem could be at *any time* during their run. This means that there are no "final" solutions, as the point in time when the algorithm is stopped may be arbitrarily chosen (here: after 1h). Thus, anytime algorithms cannot just be characterized by a final solution and runtime requirement, but should be compared based on their whole runtime behavior [36].

We therefore use the *TSP Suite* [36] to execute our experiments, gather data about algorithm runtime behavior, and to evaluate these data. The *TSP Suite* also addresses the problem of properly measuring the time consumed by an algorithm: Runtime measurements presented in terms CPU seconds can capture all the complexities and the overhead of the algorithm implementations, but are strongly machine-dependent and inherently incomparable. Even if normalized runtimes (NT) are calculated based on machine performance factors, they remain problem specific and may not represent the utility of black-box metaheuristics in general. Counting the number of generated solutions (i.e., objective function evaluations, or *FE*s in short) is the most-often used alternative in benchmarking. However, it neglects the fact that one *FE* in (plain) ACO and PACO has quadratic complexity whereas in a LS algorithm using 2-opt operations, it may be in $\mathcal{O}(1)$. The *TSP Suite* thus measures runtime in four different measures, CPU time, normalized CPU time, FEs, and the number DE of accesses to the distance matrix D, in order to provide a balanced overview on algorithm performance.

The *TSP Suite* furthermore ranks algorithms according to their overall performance. This ranking includes statistical comparisons of results at different runtimes, empirical cumulative distribution functions (ECDFs) [17, 22, 34] for different goal values (see the following section), the progress of algorithms over time, estimated running time (ERT) [17] curves, final results, as well as the expected runtime to reach the optimum, amongst others. It therefore represents algorithm performance and robustness from several different angles.

3.1 LS and LS-LS Performance

First we will investigate LS and the LS-LS hybrids. We therefore define 8 LS setups, namely FSM**, MNS, and 6 setups of the LK heuristic, differing in their candidate set size (5, 10, 20, 30, 40, and n). We call the LK setups LK5, LK10, ..., LKn, respectively. We compare them to 26 LS-LS hybrids, namely 6 setups of LK-FSM**, 6 setups of FSM**-LK, 6 setups of LK-MNS, 6 setups of MNS-LK, the FSM**-MNS, and the MNS-FSM**.

According to the automated ranking obtained from the *TSP Suite* (see Figure 6 at the end of the experiment section), LK10-MNS, FSM**-LK10, and FSM**-LK5 have the best performances amongst the LS and LS-LS algorithms. In Figure 3, we plot the ECDFs of selected setups for different goal errors F_t and runtime measures. The ECDF illustrates the fraction of runs that have discovered a (best) solution with $F_b \leq F_t$, where F_b corresponds to the relative excess length of the tour length compared to the global optimum. $F_b = 0.01$ would correspond to a tour which is 1% longer than the globally optimal tour of a given benchmark instance[1]. The illustrated ECDFs are aggregated over all 110 benchmark instances. An algorithm is the better the faster and higher its ECDF rises. For the sake of readability, we only include the pure LS and six representative hybrid versions in the diagram.

The ECDF of FSM**-LK10 in Figure 3a, based on the normalized CPU runtime measure NT, reaches 0.45 for $F_t = 0$. In other words, a global optimum can be reached in about 45% (of the runs) of all benchmark cases under the given computational budget. Its ECDF curve rises quicker and reaches higher than those of LK10 and FSM**, which both converge at 0.4. This means that FSM**-LK10 outperforms both of its pure components. Similar observations can be made for LK10-MNS and FSM**-MNS. This confirms that hybridizing LS algorithms with each other therefore can improve the performance.

Not *all* LS-LS hybridizations lead to a better performance. MNS-LK10 and MNS-FSM**, which first execute MNS followed by LK10 and FSM**, respectively, outperform MNS but not their second component LS. Their ECDFs are higher only for a very small time budget, probably due to the fast initial progress of MNS, but converge around 0.175 and 0.2. This is better than MNS, which can only solve 10% of the problem instances. LK10 and FSM**, however, can find the global optima in 40% of their runs. One possible reason for the influence of LS execution order on performance might be that MNS produces a local optimum from which the subsequent LS steps cannot easily escape. MNS itself may be able to improve the results of the other methods, as it exhaustively scans a relatively large neighborhood.

In the other sub-figures of Figure 3, we increase the goal error F_t to 0.025, i.e., illustrates the fraction of runs finding a solution that is up to 2.5% longer than the optimum. This goal can be reached more often. LK10-FSM** and LK10-MNS now can reach marginally higher ECDF values than FSM**-LK10 and FSM**-MNS.

In Figure 3c, we change the runtime measure from normalized CPU time NT to the number of objective function evaluations, i.e., *FE*s. This measure counts the search steps but disregards their algorithmic complexity, which is to the disadvantage of the hybrids starting with LK. The hybrids beginning with FSM** and MNS may need fewer *FE*s to solve some of the problems, which leads to higher ECDF values earlier on in this chart. From Figure 3b we know, however, that there is no big difference in terms of the actually consumed CPU times. This shows that it is *not* sufficient to just count *FE*s when experimentally investigating optimization algorithms[2]...

We can furthermore find that a candidate set size around 10 for hybrid LK perform best, although pure LK tends to perform better with larger candidate set sizes (except for LKn, see Figure 6 at the end of this section). This shows that the best parameter settings of a pure LS are not necessarily the best settings for its hybrid forms. A set of experiments to find good setups is therefore always necessary, even if we hybridize algorithms which have been extensively tested in their separate, pure forms.

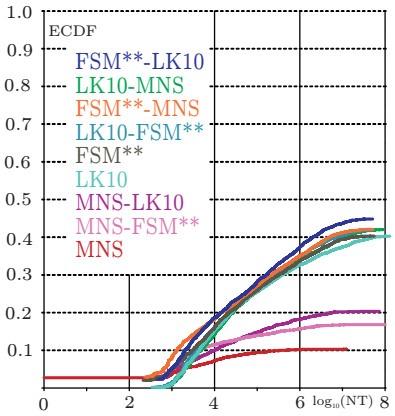
3.2 EC-LS and EC-LS-LS Hybrids

After confirming that LS-LS hybrids outperform pure LS and knowing that EC-LS hybrids outperform both pure EC and LS from [28, 36, 38], we now investigate combinations of a EC and our new LS-LS algorithms, i.e., EC-LS-LS hybrids. Since each single setup consumes more than one week of runtime on one node in our cluster, we chose the two best LS-LS methods, LK10-MNS and FSM**-LK10, for EC-LS-LS hybridization only.

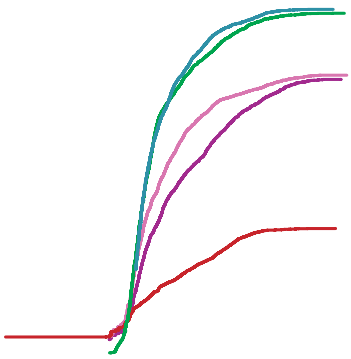
For each of LK10-MNS, FSM**-LK10, MNS, LKn, and FSM**, 9 additional setups were evaluated: 3 hybrids with PACO (using PACO(3,10), PACO(3,25), and PACO(5,10)))

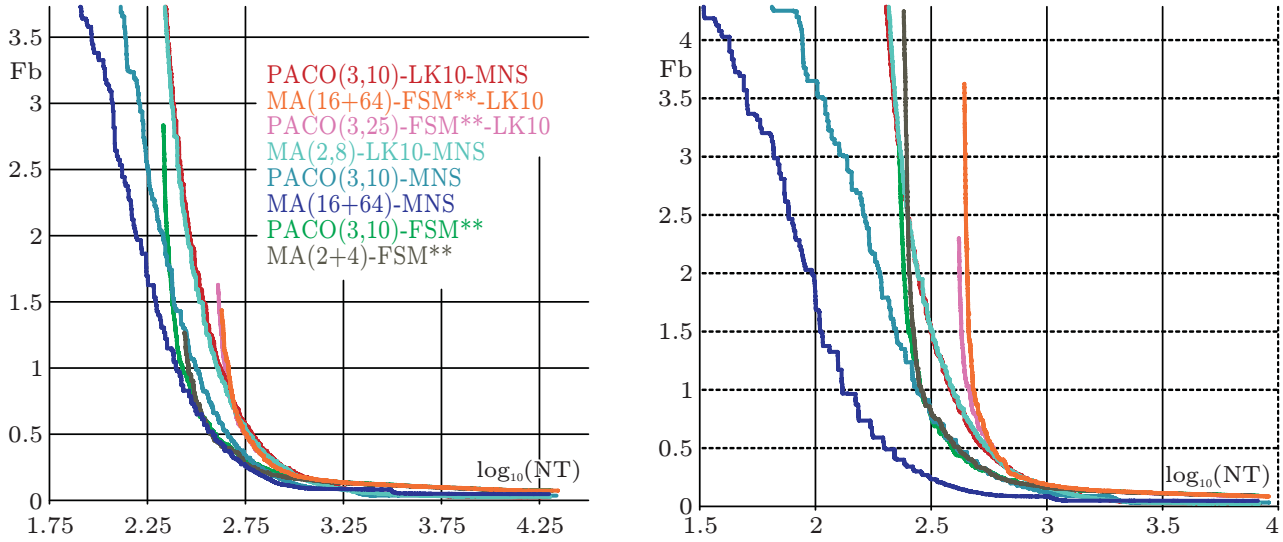
[1] All optima are known for *TSPLib*.

[2] ECDF charts based on time measured in *DE*s look similar to those using NT, signifying the *DE*s are a better machine-independent time measure for the TSP than *FE*s.



(a) ECDF for NT and $F_t = 0.0$.





(a) Progress in F_b over NT for $8192 \leq n < 16383$.

(b) Progress in F_b over NT for $16384 \leq n < 32768$.

Figure 5: Progress of the EC-LS and EC-LS-LS hybrids in terms of the best objective value in F_b discovered so far over the normalized runtime NT and different large problem scales.

PACO(3,10)-LK10-MNS *(rank 1)*,
PACO(3,25)-LK10-MNS *(2)*, PACO(5,10)-LK10-MNS *(3)*,
MA(16+64)-FSM**-LK10 *(4)*, PACO(5,10)-FSM**-LK10 *(5)*,
PACO(3,25)-FSM**-LK10 *(6)*, MA(16,64)-FSM**-LK10 *(7)*,
PACO(3,10)-FSM**-LK10 *(8)*, MA(16,64)-LK10-MNS *(9)*,
MA(2,8)-LK10-MNS *(10)*, MA(16+64)-LK10-MNS *(11)*,
MA(2,4)-LK10-MNS *(12.5)*, MA(2,8)-FSM**-LK10 *(12.5)*,
MA(2+8)-FSM**-LK10 *(14)*, MA(2+4)-FSM**-LK10 *(15.5)*,
PACO(5,10)-MNS *(15.5)*, MA(2+4)-LK10-MNS *(17.5)*,
MA(2,4)-FSM**-LK10 *(17.5)*, PACO(3,10)-MNS *(19.5)*,
PACO(3,25)-MNS *(19.5)*, MA(2+8)-LK10-MNS *(21)*,
PACO(3,10)-FSM** *(22)*, PACO(5,10)-FSM** *(23)*,
PACO(3,25)-FSM** *(24)*, MA(2+8)-FSM** *(25)*,
MA(2+4)-FSM** *(26)*, MA(16,64)-FSM** *(27)*,
MA(16+64)-FSM** *(28)*, MA(2,4)-FSM** *(29)*,
MA(2,8)-FSM** *(30)*, MA(16+64)-MNS *(31)*,
MA(2+4)-MNS *(32)*, MA(2+8)-MNS *(33)*,
MA(16,64)-MNS *(34)*, MA(2,8)-MNS *(35)*,
PACO(3,10)-LKn *(36)*, PACO(3,25)-LKn *(37)*,
PACO(5,10)-LKn *(38)*, LK10-MNS *(39)*, FSM**-LK10 *(40)*,
FSM**-LK5 *(41)*, FSM**-LK20 *(42)*, LK10-FSM** *(43)*,
MA(2,4)-MNS *(44)*, LK5-FSM** *(45)*, FSM**-LK30 *(46)*,
FSM**-MNS *(47)*, LK5-MNS *(48)*, FSM**-LK40 *(49)*,
FSM**-LKn *(50)*, LK20-MNS *(51)*, LK20-FSM** *(52)*,
LK20 *(53)*, LK30 *(54)*, LK40-MNS *(55)*, LK30-MNS *(56)*,
LK30-FSM** *(57)*, LK10 *(58)*, LK40 *(59)*, FSM** *(60)*,
LK40-FSM** *(61)*, LK5 *(62)*, MA(16+64)-LKn *(63)*,
LKn-FSM** *(64)*, MA(16,64)-LKn *(65)*, LKn-MNS *(66)*,
MA(2,8)-LK *(67)*, MA(2+4)-LK *(68)*, MA(2,4)-LK *(69)*,
LKn *(70)*, MA(2+8)-LKn *(71)*, MNS-LK40 *(73)*,
MNS-LK30 *(73)*, MNS-LK20 *(73)*, MNS-FSM** *(75)*,
MNS-LK10 *(76)*, MNS-LK5 *(77)*, MNS-LKn *(78)*, and
MNS *(79)*.

Figure 6: Algorithm ranking from best to worst, based on various performance measures and statistics (see [36] for details). The different algorithm types pure local search, LS-LS hybrid, EC-LS hybrid, and EC-LS-LS hybrid are highlighted.

pairwise combined the three state-of-the-art TSP solvers FSM**, MNS, and the LK heuristic. We hybridized the new LS-LS hybrids further with two EC approaches, namely an EA and PACO. We conducted a large-scale experimental study applying 79 different setups to all of the 110 benchmark instances from *TSPLib*. Our experiments have led us to four major conclusions:

1. The new LS-LS hybrids are better than their pure LS algorithm components. This means that the new idea of combining the strengths of different LS algorithms is very promising.

2. The LS-LS hybrids are still slightly worse than the above-mentioned EC methods hybridized with a single LS, i.e., EC-LS algorithms. This means that a global search component is still necessary in a good TSP solver.

3. The new EC-LS-LS hybrids outperform the LS-LS algorithms as well as EC-LS hybrids. The best overall algorithm, PACO(3,10)-LK10-MNS, unites the global search strength of PACO, the ability to find good solutions of the LK heuristic, and the fast exploitation speed of MNS.

4. In [28, 36, 38, 40] as well as the present study, PACO is the significantly better host EC method for hybridization than an EA. However, we also find an exception to this rule, as MAs with FSM**-LK10 are better than the corresponding PACO versions.

In our future work, we will hybridize more than two LS algorithms with each other. Since LK10-MNS and FSM**-LK10 both perform well, it is attractive to also investigate FSM**-LK10-MNS. We will construct LS-LS hybrids with the efficient Tabu Search-based TSP solvers introduced in [40]. Furthermore, instead of hybridizing an EC method with an LS-LS hybrid, we might instead hybridize it with several LS

algorithms at once. In each search step, it will randomly choose which LS to use for refining a new candidate solution. This choice will be adaptive and change according to the success rate of such refinements.

Finally, the concept of LS-LS and EC-LS-LS hybrid algorithms is also promising for other problem domains. We will therefore explore its utility on several other well-known combinatorial optimization problems.

Acknowledgments We acknowledge support from the Fundamental Research Funds for the Central Universities and the National Natural Science Foundation of China under Grants 61150110488 and 71520107002. The experiments reported in this paper were executed on the supercomputing system in the Supercomputing Center of University of Science and Technology of China.

References

[1] D. L. Applegate, W. J. Cook, and A. Rohe. Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing (JOC)*, 15(1):82–92, Winter 2003.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton, NJ, USA: Princeton University Press, 2007. ISBN 0-691-12993-2.

[3] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Computational Intelligence Library. New York, NY, USA: Oxford University Press, Inc., 1997. ISBN 0-7503-0392-1.

[4] C. Blum, R. Chiong, M. Clerc, K. A. De Jong, Z. Michalewicz, F. Neri, and T. Weise. Evolutionary optimization. In R. Chiong, T. Weise, and Z. Michalewicz, editors, *Variants of Evolutionary Algorithms for Real-World Applications*, chapter 1, pages 1–29. Berlin/Heidelberg: Springer-Verlag, 2011. doi:10.1007/978-3-642-23424-8_1.

[5] M. S. Boddy and T. L. Dean. Solving time-dependent planning problems. Technical Report CS-89-03, Providence, RI, USA: Brown University, Department of Computer Science, Feb. 1989.

[6] R. Chiong, F. Neri, and R. I. McKay. Nature that breeds solutions. In *Nature-Inspired Informatics for Intelligent Applications and Knowledge Discovery: Implications in Business, Science and Engineering*, chapter 1, pages 1–24. Hershey, PA, USA: Information Science Reference, 2009.

[7] K. A. De Jong. *Evolutionary Computation: A Unified Approach*, volume 4 of *Bradford Books*. Cambridge, MA, USA: MIT Press, 2006. ISBN 0262041944.

[8] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Milano, Italy: Dipartimento di Elettronica, Politecnico di Milano, Jan. 1992. In Italian.

[9] M. Dorigo, M. Birattari, and T. Stützle. Ant colony optimization – artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine (CIM)*, 1(4):28–39, Nov. 2006.

[10] D. B. Fogel. An evolutionary approach to the traveling salesman problem. *Biological Cybernetics*, 60(2):139–144, Dec. 1988.

[11] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric tsps by ant colonies. In *Proc. of IEEE Intl. Conf. on Evolutionary Computation (CEC'96)*, pages 622–627, Nagoya, Aichi, Japan, May 20–22, 1996. Los Alamitos, CA, USA: IEEE Computer Society Press.

[12] F. Glover. Ejection chains and combinatorial leverage for the traveling salesman problems. Technical report, University of Colorado-Boulder, Boulder, CO, USA, 1992.

[13] F. W. Glover. New ejection chain and alternating path methods for traveling salesman problems. In O. Balci and R. Sharda, editors, *Computer Science and Operations Research: New Developments in Their Interfaces*, pages 449–509. Oxford, UK: Pergamon Press, July 1992.

[14] F. W. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics – The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, 65(1-3):223–253, Mar. 1996.

[15] M. Guntsch and M. Middendorf. Applying population based aco to dynamic optimization problems. In *From Ant Colonies to Artificial Ants – Proc. of the Third Intl. Workshop on Ant Colony Optimization (ANTS'02)*, volume 2463/2002 of *LNCS*, pages 111–122, Brussels, Belgium, Sept. 12–14, 2002. Berlin, Germany: Springer-Verlag GmbH.

[16] G. Z. Gutin and A. P. Punnen, editors. *The Traveling Salesman Problem and its Variations*, volume 12 of *Combinatorial Optimization*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN 0-306-48213-4.

[17] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking: Experimental setup. Technical report, Orsay, France: Université Paris Sud, INRIA Futurs, Équipe TAO, Mar. 24, 2012.

[18] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research (EJOR)*, 130(3):449–467, May 1 2001.

[19] K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. Datalogiske Skrifter (Writings on Computer Science) 81, Roskilde, Denmark: Roskilde University, Department of Computer Science, 1998.

[20] K. Helsgaun. General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation*, 1 (2-3):119–163, Oct. 2009.

[21] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: University of Michigan Press, 1975. ISBN 0-472-08460-7.

[22] H. H. Hoos and T. Stützle. Evaluating las vegas algorithms – pitfalls and remedies. In *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence (UAI'98)*, pages 238–245, Madison, WI, USA, July 24–26, 1998. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[23] Y. Jiang, T. Weise, J. Lässig, R. Chiong, and R. Athauda. Comparing a hybrid branch and bound algorithm with evolutionary computation methods, local search and their hybrids on the tsp. In *Proc. of the IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS'14), Proc. of the IEEE Symposium Series on Computational Intelligence (SSCI'14)*, Orlando, FL, USA, Dec. 9–12, 2014. Los Alamitos, CA, USA: IEEE Computer Society Press. doi:10.1109/CIPLS.2014.7007174.

[24] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Journal of Artificial Intelligence Research (JAIR)*, 13(2): 129–170, Apr. 1999.

[25] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, UK: Wiley Interscience, Sept. 1985. ISBN 0-471-90413-9.

[26] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21 (2):498–516, Mar.–Apr. 1973.

[27] J. D. C. Little, K. G. Murty, D. W. Sweeny, and C. Karel. An algorithm for the traveling salesman problem. Sloan Working Papers 07-63, Cambridge, MA, USA: Massachusetts Institute of Technology (MIT), Sloan School of Management, Mar. 1, 1963.

[28] W. Liu, T. Weise, Y. Wu, and R. Chiong. Hybrid ejection chain methods for the traveling salesman problem. In *Proc. of the 10th Intl. Conf. Bio-Inspired Computing – Theories and Applications (BIC-TA'15)*, volume 562 of *Communications in Computer and Information Science*, pages 268–282. Berlin/Heidelberg: Springer-Verlag, Hefei, Anhui, China, Sept. 25–28, 2015. ISBN 978-3-662-49013-6. doi:10.1007/978-3-662-49014-3_25.

[29] P. Merz and B. Freisleben. Memetic algorithms for the traveling salesman problem. *Complex Systems*, 13(4):297–345, 2001.

[30] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag GmbH, 1996. ISBN 3-540-58090-5.

[31] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program C3P 826, Pasadena, CA, USA: California Institute of Technology (Caltech), Caltech Concurrent Computation Program (C3P), 1989. URL http://www.each.usp.br/sarajane/SubPaginas/arquivos_aulas_IA/memetic.pdf.

[32] C. Rego. Relaxed tours and path ejections for the traveling salesman problem. *European Journal of Operational Research*, 106(2):522–538, 1998.

[33] G. Reinelt. Tsplib – a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, Fall 1991.

[34] D. A. D. Tompkins and H. H. Hoos. Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat. In *Revised Selected Papers from the Seventh Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT'04)*, volume 3542 of *LNCS*, pages 306–320, Vancouver, BC, Canada, May 10–13, 2004. Berlin, Germany: Springer-Verlag GmbH.

[35] T. Weise. *Global Optimization Algorithms – Theory and Application*. Germany: it-weise.de (self-published), 2009. URL http://www.it-weise.de/projects/book.pdf.

[36] T. Weise, R. Chiong, K. Tang, J. Lässig, S. Tsutsui, W. Chen, Z. Michalewicz, and X. Yao. Benchmarking optimization algorithms: An open source framework for the traveling salesman problem. *IEEE Computational Intelligence Magazine (CIM)*, 9(3):40–52, Aug. 2014. doi:10.1109/MCI.2014.2326101.

[37] L. D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In *Proc. of the Third Intl. Conf. on Genetic Algorithms (ICGA'89)*, pages 133–140, Fairfax, VA, USA: George Mason University (GMU), June 4–7, 1989. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[38] Y. Wu, T. Weise, and R. Chiong. Local search for the traveling salesman problem: A comparative study. In *Proc. of 14th IEEE Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC'15)*, pages 213–220, Beijing, China, July 6–8, 2015. doi:10.1109/ICCI-CC.2015.7259388.

[39] Y. Wu, T. Weise, and W. Liu. Hybridizing different local search algorithms with each other and evolutionary computation: Better performance on the traveling salesman problem. In *Proceedings of the 18th Genetic and Evolutionary Computation Conference (GECCO'16)*, pages 57–58. ACM Press: New York, NY, USA, July20–24, 2016. ISBN 978-1-4503-4323-7/16/07. doi:10.1145/2908961.2909001. published as short paper.

[40] D. Xu, T. Weise, Y. Wu, J. Lässig, and R. Chiong. An investigation of hybrid tabu search for the traveling salesman problem. In *Proc. of the 10th Intl. Conf. Bio-Inspired Computing – Theories and Applications (BIC-TA'15)*, volume 562 of *Communications in Computer and Information Science*, pages 523–537. Berlin/Heidelberg: Springer-Verlag, Hefei, Anhui, China, Sept. 25–28, 2015. ISBN 978-3-662-49013-6. doi:10.1007/978-3-662-49014-3_47.


```
@inproceedings{WWL2016HDLSAWEOAECBPOTTSP,
  author    = {Yuezhong Wu and Thomas Weise and Weichen Liu},
  title     = {Hybridizing Different Local Search Algorithms with
               Each Other and Evolutionary Computation: Better
               Performance on the Traveling Salesman Problem},
  booktitle = {Proceedings of the 18th Genetic and Evolutionary
               Computation Conference (GECCO'16)},
  publisher = {ACM Press: New York, NY, USA},
  year      = {2016},
  month     = jul # {20--24,~},
  location  = {Denver, CO, USA},
  pages     = {57--58},
  doi       = {10.1145/2908961.2909001},
  isbn      = {978-1-4503-4323-7/16/07},
  note      = {published as short paper},
},
```