Improving Lin-Kernighan-Helsgaun with Crossover on Clustered Instances of the TSP

Doug Hains, Darrell Whitley, and Adele Howe

Colorado State University, Fort Collins CO, USA

Abstract. Multi-trial Lin-Kernighan-Helsgaun 2 (LKH-2) is widely considered to be the best Interated Local Search heuristic for the Traveling Salesman Problem (TSP) and has found the best-known solutions to a large number of benchmark problems. Although LKH-2 performs exceptionally well on most instances, it is known to have difficulty on clustered instances of the TSP. Generalized Partition Crossover (GPX) is a crossover operator for the TSP that efficiently constructs new solutions by partitioning a graph constructed from the union of two solutions. We show that GPX is especially well-suited for clustered instances and evaluate its ability to improve solutions found by LKH-2. We present two methods of combining GPX with multi-trial LKH-2. We find that combining GPX with LKH-2 dramatically improves the evaluation of solutions found by LKH-2 alone on clustered instances with sizes ranging from 3,000 to 30,000 cities.

1 Introduction

The Traveling Salesman Problem (TSP) can be stated as follows: Given n cities and an $n \times n$ cost matrix C, where entry C_{ij} is the cost of traveling between cities i and j, find a Hamiltonian circuit on the n cities which minimizes the sum of travel costs between cities on the route. We restrict our attention to symmetric instances, that is $C_{ij} = C_{ji}$.

Although the TSP is simply stated, the problem is NP-hard, necessitating the use of heuristics for larger instances. Lin-Kernighan-Helsgaun 2 (LKH-2) [2] is a state-of-the-art local search heuristic for the TSP based on the variable depth local search of Lin and Kernighan (LK-search) [4]. While LKH-2 performs exceptionally well on most instances of the TSP, its performance degrades on clustered instances [2].

Generalized Partition Crossover (GPX) [10,9] is a crossover operator for the TSP that produces offspring from a graph constructed from the union of two parent solutions. It has been shown that GPX has a high probability of producing locally optimal offspring if the parents are local optima [9]. We inspect local optima produced by LKH-2 on clustered instances and find they have a large number of common edges. As GPX relies on the common edges between two solutions to perform crossover, this indicates that the local optima produced by LKH-2 are especially well-suited for GPX.

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 388–397, 2012.

LKH-2 can also be used in a form of iterated local search (ILS), which Helsgaun calls *multi-trial LKH-2*. Multi-trail LKH-2 incorporates a form of crossover known as Iterative Partial Transcription (IPT) based on the work of Möbius et al. [6]. Instead of using a population, multi-trial LKH-2 keeps the best-so-far solution; when it converges to a new local optimum, it applies crossover to the current candidate solution and the previous best-so-far solution. If crossover yields an improved solution, this becomes the new candidate solution. In this way, LKH-2 does not require a population, or other tuning parameters associated with population-based heuristics.

To the best of our knowledge, there are no published studies on the effectiveness of using crossover in this way. We present empirical evidence that shows incorporating a crossover operator into LKH-2 greatly improves the evaluation of solutions found by LKH-2 alone. A theoretical analysis of the differences between the GPX and IPT crossover operators proves that GPX is superior and empirical studies show that GPX produces better solutions than IPT when crossing over local optima produced by LKH-2.

Finally, there is also an advantage to doing multiple (parallel) runs of multi-trial LKH-2. This creates additional opportunities to utilize crossover. This is not like a genetic algorithm, because the runs remain independent. We present two methods of using crossover that have not previously been utilized in combination with LKH-2. Our results show that both methods improve solution quality over that of multi-trial LKH-2 with no significant increase to runtime.

2 Lin-Kernighan-Helsgaun 2

Lin-Kernighan-Helsgaun 2 (LKH-2) is a variable depth search that is based on the well known Lin-Kernighan algorithm (LK-search) [4]. LKH-2 has found the majority of best known solutions on the TSP benchmarks at the Georgia Tech TSP repository that were not solved by complete solvers[1]. At each step of the search, LKH-2 removes and replaces k edges of a given solution. This is known as a k-opt move. LKH-2 chains together a variable number of k-opt moves to find a new solution with a better evaluation than the initial solution [2].

As we are not concerned with the inner workings of LKH-2, it is sufficient to think of LKH-2 as a 'black-box' that produces a locally optimal solution when given an arbitrary initial solution. LKH-2 has a number of parameters that influence its performance; for all our experiments, we use the same settings reported by Helsgaun to produce the best results on clustered instances [2].

2.1 LKH-2 and Clustered Instances

Papadimitriou proved that LK-search solves a Polynomial Local Search (PLS) complete problem [8,3] by constructing graphs that force LK-Search to take an exponential number of steps. Papadimitriou constructed a graph containing 'bait edges' that lead LK-search into an extensive search for an improved solution.

[1] `http://www.tsp.gatech.edu/data/index.html`

It is thought that the edges between clusters of a clustered TSP instance act in a similar manner as the bait edges of Papadimitriou's proof [7]. Empirical experiments show that LKH-2 performs significantly worse on random clustered instances than uniform random instances [2]. We conjecture that its performance on clustered instances could be improved by exploiting crossover.

3 Generalized Partition Crossover

Generalized Partition Crossover (GPX) is a crossover operator for the TSP with a number of interesting properties. When given local optima as parents, GPX is highly likely to produce locally optimal children [10,9]. GPX is "respectful", meaning that any common edges in the parents are inherited by the offspring; and it "transmits alleles", meaning that *all* edges found in the offspring are directly inherited from the parents. When used in a hybrid genetic algorithm with Chained Lin-Kernighan (Chained LK) [1], GPX is able to produce higher quality solutions than Chained LK alone [10].

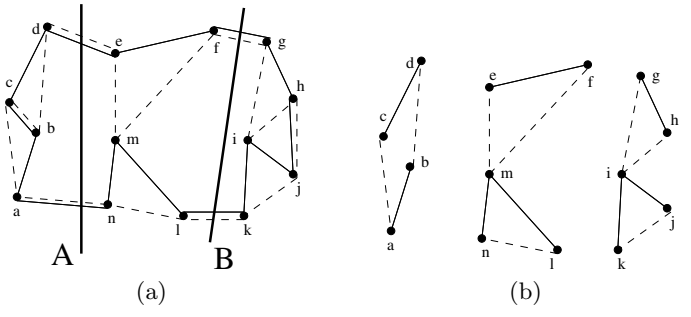


Fig. 1. An example of (a) the graph constructed by GPX from the union of two solutions, S_a (dashed edges) and S_b (solid edges), and (b) how the removal of shared edges creates subgraphs. The heavy dark lines show two partitions of cost 2, A and B. Once GPX partitions the graph, solutions are constructed from the shared edges in (a) and the dashed or solid edges from each subgraph in (b).

GPX works by first constructing a graph $G = (V, E)$ from two solutions, S_a and S_b, where V is the set of cities in the TSP instance and E is the union of edges in the two solutions (see Figure 1(a)). To partition G, the edges in G that are shared by both solutions are removed to create graph G_u as in Figure 1(b). Breadth first search is then applied to G_u to identify the connected components. Note that any connected component in G_u can be disconnected from G by removing shared edges. We refer to the connected components in G_u as *partition components*. We define the *cost* of a partition of graph G as the minimal number of shared edges that must be removed to disconnect a component from G. Whitley et al. prove that if G contains at least one partition of cost 2, it is always possible to create at least two Hamiltonian circuits distinct from the parents in $O(n)$ time [9].

After one or more partitions of cost 2 is found, GPX will recombine the parent solutions S_a and S_b to one or more offspring. First, the common edges between S_a and S_b are inherited. Then, the best possible offspring is obtained by greedily selecting the lowest cost path through each partition component of graph G. Additional offspring can be obtained by making non-greedy choices.

GPX is an ideal candidate for clustered instances of the TSP. As local optima tend to have a large number of edges in common, it is likely that some of these edges will be between clusters. This would allow GPX to partition across edges between clusters and recombine the lowest cost paths through the clusters on either side of the partition. If a partition component captures several clusters, GPX will also recombine the lowest cost edges between clusters.

To determine if this is the case, we produced one hundred different local optima with LKH-2 on a randomly generated clustered instance with 3162 cities. On average, there were 2.10 ± 0.04 partitions of cost 2 when pairing together these solutions. Thus, GPX will likely be able to find multiple partition components of cost 2 when using two local optima produced by LKH-2 as parents.

3.1 Iterative Partial Transcription and GPX

Iterative Partial Transcription [6] (IPT) is a form of crossover used by LKH-2. When LKH-2 reaches a local optimum, it executes a random restart while retaining the best-so-far-solution. LKH-2 uses IPT to recombine the best-so-far solution with the new local optimum [2]. If IPT finds a solution better than the best-so-far, the best-so-far solution is replaced. Note that this does not require a population of solutions.

IPT constructs a graph $G = (V, E)$ from two solutions, S_1 and S_2, in the same way as GPX. IPT will attempt to partition V into two disjoint sets A and B such that the number of edges between sets is exactly 2. Let $E(A)$ be the set of edges incident only to vertices in A and $E(B)$ the set of edges incident only to vertices in B. Offspring are formed by removing the edges in S_1 that are also in $E(A)$ and replacing them with the edges found in both S_2 and $E(A)$. The same process is repeated with $E(B)$. This is identical to using Partition Crossover utilizing only a single partition of cost 2 [9].

Given G with k partition components of cost 2, GPX will return the best solution out of a possible $2^k - 2$ unique solutions [10]. As IPT uses only a single partition, it can reach only $2k - 2$ of the $2^k - 2$ solutions processed by GPX. We subtract 2 as we count only solutions different from the two parent solutions. Note that $k \geq 2$ since 1 partition break the graph into 2 partition components.

For $k > 2$, the solutions reachable by IPT are a subset of the solutions reachable by GPX. Therefore, it follows that the offspring generated by GPX is guaranteed to be equal or better than the solution generated by IPT. When k is larger, GPX will find a better solution with greater probability.

3.2 Effect of Crossover on LKH-2

To the best of our knowledge, there have been no published experiments to determine the effect of using LKH-2 with and without IPT. We therefore conduct an experiment to validate two hypotheses: 1. The addition of a crossover operator such as IPT can improve solutions over that of LKH-2, and 2. GPX can further improve solutions over IPT without a significant increase in run time.

For this experiment, we used six instances from the 8th DIMACS TSP Challenge[2]: C3k.0, C3k.1, C10k.0, C10k.1, C31k.0 and C31k.1 with sizes 3162, 3162, 10000, 10000, 31623 and 31623, respectively. For each instance, we ran three versions of LKH-2 with 10 random restarts: LKH-2 without crossover, LKH-2 with IPT and LKH-2 with GPX. When running LKH-2 with either crossover operator, the crossover operator is applied to the best-so-far solution and the local optimum produced after each restart.

As the optimal solutions are not known for all instances, we normalize the results in Table 1 by reporting the percentage above the Held-Karp bound (HKB). The HKB is guaranteed to be within 2/3 of the optimal evaluation on Euclidean instances, but in practice is often within 0.01% of optimal [3]. The HKB for each instance can be found at the DIMACS TSP Challenge homepage. We find that crossover significantly improves the solutions found by LKH-2 and that GPX can further improve over the solutions found by IPT.

Table 2 reports the average CPU time of a single call to IPT and a single call to GPX. We also report the percentage of the total CPU time per run of LKH-2 accounted for by the crossover operators. For instances of 10,000 cities and lower, IPT is slightly faster than GPX but both account for less than 0.1% of the overall running time. GPX is faster than IPT on larger problems. Therefore GPX can be used in place of IPT with no significant increase to the overall run time while increasing the potential to find improved solutions.

Table 1. The minimum percentage above the Held-Karp Bound for several clustered instances of the TSP of solutions found by ten random restarts of LKH-2 without crossover, with IPT and with GPX. Best values for each instance are in boldface. The p-value of a one-way ANOVA test are shown in the final row for each instance. A significant value is denoted with (*).

	Instance	C3k.0	C3k.1	C10k.0	C10k.1	C31k.0	C31k.1
LKH-2	Min	0.660	0.863	1.143	1.009	1.489	1.538
	Avg	0.772	1.584	1.671	1.597	1.760	1.907
	Max	1.387	2.051	2.339	2.658	2.169	2.397
LKH-2 w/ IPT	Min	**0.622**	0.656	1.040	0.873	1.280	1.274
	Avg	0.665	1.329	1.160	1.022	1.433	1.595
	Max	0.786	2.051	1.396	1.419	1.663	2.397
LKH-2 w/ GPX	Min	**0.622**	**0.651**	**1.031**	**0.872**	**1.270**	**1.267**
	Avg	0.660	1.326	1.159	1.021	1.426	1.591
	Max	0.786	2.051	1.396	1.419	1.663	2.397
p-value		0.112	0.484	<0.001(*)	<0.001(*)	<0.001(*)	<0.001(*)

[2] `http://www2.research.att.com/~dsj/chtsp/`

Table 2. Average run time in seconds of IPT and GPX over the 10 runs used to produce Table 1. The numbers in parentheses represent the percentage of the total run time of LKH-2 accounted for by the crossover operators. P-values for a standard t-test are shown with significant values denoted by (*).

Instance	C3k.0	C3k.1	C10k.0	C10k.1	C31k.0	C31k.1
IPT	0.001(0.04%)	0.001(0.04%)	0.005(0.06%)	0.005(0.06%)	0.047(0.17%)	0.052(0.18%)
GPX	0.002(0.08%)	0.001(0.04%)	0.007(0.08%)	0.006(0.07%)	0.026(0.09%)	0.029(0.1%)
p-value	0.016	0.005(*)	0.036	0.014	<0.001(*)	<0.001(*)

4 Crossover and Iterated Local Search

In Iterated Local Search (ILS) [5] a perturbation operator is applied to escape local optima. The local search is restarted on the perturbed solution, and the process is repeated for a fixed number of iterations. Using ILS with LK-search based heuristics has proven to be more effective than random restarts [3,1]. Helsgaun refers to the ILS version of LKH-2 as *multi-trial LKH-2* [2].

Multi-trial LKH-2 uses a pseudo-random restart influenced by the best-so-far solution when a local optimum is reached. The next iteration of LKH-2 is then biased by ignoring any k-opt moves beginning with edges in the best-so-far solution. Given the benefits of incorporating crossover in LKH-2 shown in Section 3.2, we construct two methods for incorporating crossover with multi-trial LKH-2: *GPX across runs* and *GPX across restarts*.

4.1 GPX across Runs

GPX across runs applies crossover to improve the local optima found by independent runs of multi-trial LKH-2. At each iteration i of multi-trial LKH-2 (i.e., when it reaches a local optimum), we form a population of the local optima found at iteration i of each independent run. We then apply GPX, crossing over the best solution in the population with each other solution. The best solution found will be stored, but not returned to the multi-trial LKH-2 runs. This preserves diversity between the runs. Figure 2 depicts 10 independent runs of multi-trial LKH-2; GPX across runs will crossover the solutions with the same letters.

4.2 GPX across Restarts

Another option is to crossover solutions from the *same* run. We could crossover the best-so-far solution with the local optimum found at each iteration of multi-trial LKH-2 like IPT. However, the local optimum is discarded if it is not better than the best-so-far solution. It is possible that by doing so, low cost edges that could potentially be used to improve the best-so-far solution are discarded. To remedy this, we designed Subroutine 1 to maintain a population of local optima and to crossover the population at each iteration of multi-trial LKH-2. The best solution from the crossover becomes the starting solution for the next iteration of multi-trial LKH-2.

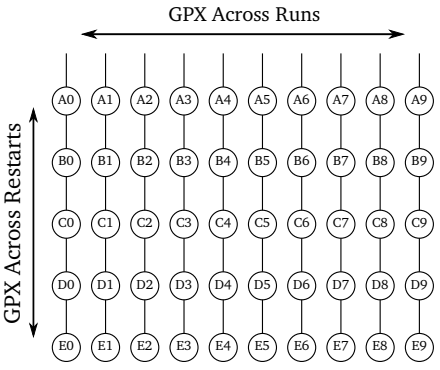


Fig. 2. A diagram depicting 10 runs of multi-trial LKH-2 run for 5 iterations per run. The circles represent local optima produced by LKH-2. GPX across runs crosses over solutions with the same letters. GPX across restarts crosses over solutions with the same numbers.

Subroutine 1. Given s^*, a local optimum passed from LKH-2; c, a cost function that sums the edge costs of a given solution, and P, a set of solutions. When the subroutine returns, control passes back to LKH-2.

1. If $P = \varnothing$, Let $P = \{s^*\}$ and return s^*.
2. Apply GPX with s^* and each solution in P.
3. Let s' be the offspring with best evaluation. If $c(s') < c(s^*)$, let $s^* = s'$.
4. If $|P| \neq$ popsize, let $P = P \cup \{s^*\}$.
5. Otherwise, let s be the solution with the poorest evaluation in P. If $c(s^*) < c(s)$, replace s in P with s^*.
6. Return s^*.

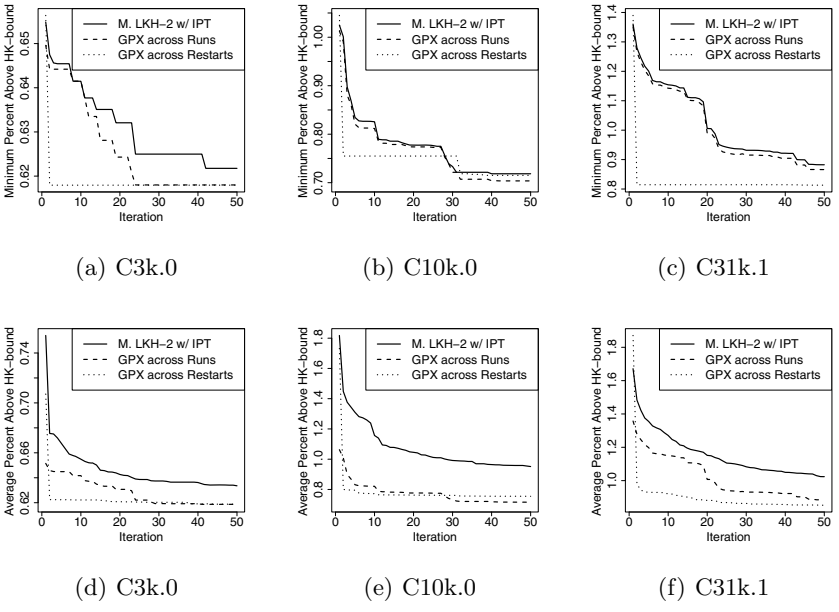
4.3 Effect of Crossover on Multi-trial LKH-2

We hypothesize that incorporating crossover with multi-trial LKH-2 should further improve solution quality, especially when GPX is the crossover operator. To test this, we ran 10 independent runs of multi-trial LKH-2 with three different methods of crossover for 50 iterations per run on the same clustered instances as before. Method one was multi-trial LKH-2 with IPT. IPT was applied at each iteration to the most recent local optimum and the best-so-far solution. If IPT produced a better solution than the best-so-far solution, it is replaced. Method two was GPX across runs. At each iteration, GPX was applied to the 10 local optima found by each run. Method three was GPX across restarts. Subroutine 1 with popsize = 10 was called at each iteration. When applying GPX across restarts, the population was set to empty at the beginning of each run.

Table 3 reports the minimum, maximum and average evaluation above the HKB after the final iteration. The data in Table 3 shows that Multi-trial LKH-2 with IPT is better than LKH-2 with IPT (see Table 1) in every problem. The

Table 3. Minimum, average, and maximum percentage of evaluation above the Held-Karp bound for solutions after 50 iterations of 10 runs of multi-trial LKH-2 using different crossovers. '*' signifies the known global optimum was found. Best solutions are in boldface. The p-value of a one-way ANOVA test are shown in the final row for each instance. All values were significant.

	Instance	C3k.0	C3k.1	C10k.0	C10k.1	C31k.0	C31k.1
M. LKH-2 w/ IPT	Min	0.6218	0.6153	0.7184	0.7061	0.6922	0.8824
	Avg	0.6336	1.0822	0.9514	0.8890	0.9639	1.0237
	Max	0.6432	1.5692	1.3341	1.2489	1.2077	1.1417
GPX Across Runs	Min	**0.6180***	0.6153	**0.7037**	**0.7036**	**0.6879**	0.8660
	Avg	0.6183	0.6156	0.7144	0.7048	0.7012	0.8843
	Max	0.6190	0.6183	0.7223	0.7064	0.8053	0.9409
GPX Across Restarts	Min	**0.6180***	**0.6150***	0.7151	0.7912	0.9725	**0.8131**
	Avg	0.6188	0.7117	0.7529	0.8615	1.0199	0.8525
	Max	0.6254	1.5350	0.7691	1.1518	1.1430	0.9376
p-values		< 0.001	0.001	0.002	0.008	< 0.001	< 0.001



(a) C3k.0 (b) C10k.0 (c) C31k.1

(d) C3k.0 (e) C10k.0 (f) C31k.1

Fig. 3. Minimum (top row) and average (bottom row) evaluation above the Held-Karp bound at each iteration across 10 runs of multi-trial LKH-2 using various crossover methods on three instances.

same is true for the two methods using GPX. Thus, multi-trial LKH-2 with crossover does provide a significant benefit. Comparing the results for IPT to that of the two GPX based methods shows that GPX generally improves over IPT. GPX Across Restarts finds the global optimum in two cases where IPT did not. GPX Across Runs consistently improves upon IPT.

To further assess the differences between the three methods, Figure 3 shows the minimum and average evaluation at each iteration for several instances. Interestingly, GPX across restarts yields the largest gain in evaluation initially. After the second iteration, the local optima from the first iteration will be crossed over with the local optimum produced at the second iteration. As the search was biased away from investigating moves beginning with edges in the best-so-far solution, the local optimum it produces and the best-so-far solution may present ideal parents for GPX. In some cases, it finds the best solution in the second iteration and does not improve further. This may be a result of an interaction with the way multi-trial LKH-2 biases the search with edges in the best-so-far solution [2].

GPX across runs is capable of consistently improving the quality of solutions over that of IPT. The average evaluation across iterations shows larger differences of GPX over IPT. Note that the solutions produced by GPX across runs are not in any way used by LKH-2. Therefore, GPX across runs does not influence the local search. On the other hand, GPX across restarts does influence the search behavior of LKH-2. The results suggest some trade-offs in how crossover can be exploited that might offer further opportunity for improvement.

5 Conclusion

Clustered instances of the TSP are problematic for LKH-2 [2]. Examining the structure of clustered instances, it seems likely that crossover operators such as GPX and IPT will perform well on clustered instances. We examine both operators and show that they are able to significantly improve solution quality on clustered instances when combined with LKH-2.

Furthermore, GPX is a compelling replacement for IPT in LKH-2. GPX is able to find all partitions that IPT can find, but can utilize more of them when constructing offspring. This allows GPX to find higher quality solutions than IPT. GPX also has a computation cost comparable to IPT. Although IPT is slightly faster on smaller instances, both operators require less than than 0.2% of the overall run time of LKH-2. As the instance size grows, GPX scales better than IPT and GPX becomes faster than IPT on larger instances.

We introduce two methods of incorporating crossover with multi-trial LKH-2. GPX across restarts uses a subroutine to maintain a population of solutions as an alternative to applying crossover to the best-so-far solution and a local optimum produced by LKH-2. GPX across restarts produces better average solutions and finds better minimum solutions on the majority of instances tested. GPX across runs consistently improves the minimum solution quality over that of multi-trial LKH-2 w/ IPT. This method also finds the globally optimal solution for the two benchmark instances for which the global optimum is known.

Our results show that crossover offers significant benefits when incorporated with a state-of-the-art local search heuristic for the TSP. We conjecture that the benefits we observed in TSP can also be obtained in other applications using local search; crossover leverages information about good partial solutions which can be exploited in search after restarts.

Acknowledgments. This research was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number FA9550-11-1-0088. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

1. Applegate, D., Cook, W., Rohe, A.: Chained Lin-Kernighan for large traveling salesman problems. INFORMS Journal on Computing 15(1), 82–92 (2003)
2. Helsgaun, K.: General k-opt submoves for the Lin-Kernighan TSP heuristic. Mathematical Programming Computation 1(2), 119–163 (2009)
3. Johnson, D.S., Mcgeoch, L.A.: The traveling salesman problem: A case study in local optimization. In: Local Search in Combinatorial Optimization, pp. 215–310. John Wiley and Sons (1997)
4. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21(2), 498–516 (1973), http://dx.doi.org/10.2307/169020
5. Lourenço, H., Martin, O., Stützle, T.: Iterated local search. In: Handbook of Metaheuristics, pp. 320–353 (2003)
6. Möbius, A., Freisleben, B., Merz, P., Schreiber, M.: Combinatorial optimization by iterative partial transcription. Physical Review E 59(4), 4667 (1999)
7. Neto, D.: Efficient cluster compensation for Lin-Kernighan heuristics. Ph.D. thesis, University of Toronto (1999)
8. Papadimitriou, C.: The complexity of the Lin-Kernighan heuristic for the traveling salesman problem. SIAM Journal on Computing 21, 450 (1992)
9. Whitley, D., Hains, D., Howe, A.: Tunneling between optima: partition crossover for the traveling salesman problem. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 915–922. ACM (2009)
10. Whitley, D., Hains, D., Howe, A.: A Hybrid Genetic Algorithm for the Traveling Salesman Problem Using Generalized Partition Crossover. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 566–575. Springer, Heidelberg (2010)