European Journal of Operational Research 208 (2011) 221–232

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

# Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem

D. Karapetyan *, G. Gutin

*Royal Holloway London University, Egham, Surrey, TW20 0EX, United Kingdom*

ARTICLE INFO

*Article history:*
Received 26 March 2010
Accepted 11 August 2010
Available online 17 August 2010

*Keywords:*
Heuristics
Lin–Kernighan
Generalized traveling salesman problem
Combinatorial optimization

ABSTRACT

The Lin–Kernighan heuristic is known to be one of the most successful heuristics for the Traveling Salesman Problem (TSP). It has also proven its efficiency in application to some other problems.

In this paper, we discuss possible adaptations of TSP heuristics for the generalized traveling salesman problem (GTSP) and focus on the case of the Lin–Kernighan algorithm. At first, we provide an easy-to-understand description of the original Lin–Kernighan heuristic. Then we propose several adaptations, both trivial and complicated. Finally, we conduct a fair competition between all the variations of the Lin–Kernighan adaptation and some other GTSP heuristics.

It appears that our adaptation of the Lin–Kernighan algorithm for the GTSP reproduces the success of the original heuristic. Different variations of our adaptation outperform all other heuristics in a wide range of trade-offs between solution quality and running time, making Lin–Kernighan the state-of-the-art GTSP local search.



## 1. Introduction

One of the most successful heuristic algorithms for the famous Traveling Salesman Problem (TSP) known so far is the Lin–Kernighan heuristic (Lin and Kernighan, 1973). It was proposed almost 40 years ago but even nowadays it is the state-of-the-art TSP local search (Johnson and McGeoch, 2002).

In this paper, we attempt to reproduce the success of the original TSP Lin–Kernighan heuristic for the generalized traveling salesman problem (GTSP), which is an important extension of the TSP. In the TSP, we are given a set $V$ of $n$ vertices and weights $w(x \to y)$ of moving from a vertex $x \in V$ to a vertex $y \in V$. A feasible solution, or a tour, is a cycle visiting every vertex in $V$ exactly once. In the GTSP, we are given a set $V$ of $n$ vertices, weights $w(x \to y)$ of moving from $x \in V$ to $y \in V$ and a partition of $V$ into $m$ nonempty clusters $C_1, C_2, \ldots, C_m$ such that $C_i \cap C_j =$ for each $i \neq j$ and $\bigcup_i C_i = V$. A feasible solution, or a tour, is a cycle visiting exactly one vertex in every cluster. The objective of both TSP and GTSP is to find the shortest tour.

If the weight matrix is symmetric, i.e., $w(x \to y) = w(y \to x)$ for any $x,y \in V$, the problem is called *symmetric*. Otherwise it is an *asymmetric* GTSP. In what follows, the number of vertices in cluster $C_i$ is denoted as $|C_i|$, the size of the largest cluster is $s$, and $Cluster(x)$ is the cluster containing a vertex $x$. The weight function $w$ can be used for edges, paths $w(x_1 \to x_2 \to \cdots \to x_k) = w(x_1 \to x_2) + w(x_2 \to x_3) + \cdots + w(x_{k-1} \to x_k)$, and cycles.

Since Lin–Kernighan is designed for the symmetric problem, we do not consider the asymmetric GTSP in this research. However, some of the algorithms proposed in this paper are naturally suited for both symmetric and asymmetric cases.

Observe that the TSP is a special case of the GTSP when $|C_i| = 1$ for each $i$ and, hence, the GTSP is NP-hard. The GTSP has a host of applications in warehouse order picking with multiple stock locations, sequencing computer files, postal routing, airport selection and routing for courier planes and some others (see, e.g., Fischetti et al., 1995, 1997; Laporte et al., 1996; Noon and Bean, 1991) and references therein.

A lot of attention was paid in the literature to solving the GTSP. Several researchers (Ben-Arieh et al., 2003; Laporte and Semet, 1999; Noon and Bean, 1993) proposed transformations of the GTSP into the TSP. At first glance, the idea to transform a little-studied problem into a well-known one seems to be natural; however, this approach has a very limited application. On the one hand, it requires exact solutions of the obtained TSP instances because even a near-optimal solution of such TSP may correspond to an infeasible GTSP solution. On the other hand, the produced TSP instances have quite an unusual structure which is difficult for the existing solvers. A more efficient way to solve the GTSP exactly is a branch-and-bound algorithm designed by Fischetti et al. (1997). This algorithm was able to solve instances with up to 89 clusters. Two approximation algorithms were proposed in the literature, but both of them are unsuitable for the general case of the problem, and the guarantied solution quality is unreasonably low for

* Corresponding author.
*E-mail addresses:* daniel.karapetyan@gmail.com (D. Karapetyan), gutin@cs.rhul.ac.uk (G. Gutin).

0377-2217/$ - see front matter Crown Copyright © 2010 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.08.011

real-world applications (see Bontoux et al., 2010) and references therein.

In order to obtain good (i.e., not necessarily exact) solutions for larger GTSP instances, one should use the heuristic approach. Several construction heuristics and local searches were discussed in Bontoux et al. (2010), Gutin and Karapetyan (2010), Hu and Raidl (2008), Renaud and Boctor (1998), Snyder and Daskin (2006) and some others. A number of metaheuristics were proposed by Bontoux et al. (2010), Gutin and Karapetyan (2010), Gutin et al. (2008), Huang et al. (2005), Pintea et al. (2007), Silberholz and Golden (2007), Snyder and Daskin (2006), Tasgetiren et al. (2007), Yang et al. (2008).

In this paper, we thoroughly discuss possible adaptations of a TSP heuristic for the GTSP and focus on the Lin–Kernighan algorithm. The idea of the Lin–Kernighan algorithm was already successfully applied to the Multidimensional Assignment Problem (Balas and Saltzman, 1991; Karapetyan and Gutin, in press). A straightforward adaptation for the GTSP was proposed by Hu and Raidl (2008); their algorithm constructs a set of TSP instances and solves all of them with the TSP Lin–Kernighan heuristic. Bontoux et al. (2010) apply the TSP Lin–Kernighan heuristic to the TSP tours induced by the GTSP tours. It will be shown in Section 3 that both of these approaches are relatively weak.

The Lin–Kernighan heuristic is a sophisticated algorithm adjusted specifically for the TSP. The explanation provided by Lin and Kernighan (1973) is full of details which complicate understanding of the main idea of the method. We start our paper from a clear explanation of a simplified TSP Lin–Kernighan heuristic (Section 2) and then propose several adaptations of the heuristic for the GTSP (Section 3). In Section 4, we provide results of a thorough experimental evaluation of all the proposed Lin–Kernighan adaptations and discuss the success of our approach in comparison to other GTSP heuristics. In Section 5, we discuss the outcomes of the conducted research and select the state-of-the-art GTSP local searches.

## 2. The TSP Lin–Kernighan heuristic

In this section, we describe the TSP Lin–Kernighan heuristic ($\mathsf{LK}_{\mathrm{tsp}}$). It is a simplified version of the original algorithm. Note that Lin and Kernighan (1973) was published almost 40 years ago, when modest computer resources, obviously, influenced the algorithm design, hiding the main idea behind the technical details. Also note that, back then, the 'goto' operator was widely used; this affects the original algorithm description. In contrast, our interpretation of the algorithm is easy to understand and implement.

$\mathsf{LK}_{\mathrm{tsp}}$ is a generalization of the $k$-opt local search. The $k$-opt neighborhood $N_{k\text{-opt}}(T)$ includes all the TSP tours which can be obtained by removing $k$ edges from the original tour $T$ and adding $k$ different edges such that the resulting tour is feasible. Observe that exploring the whole $N_{k\text{-opt}}(T)$ takes $O(n^k)$ operations and, thus, with a few exceptions, only 2-opt and rarely 3-opt are used in practice (Johnson and McGeoch, 2002; Rego and Glover, 2006).

Similarly to $k$-opt, $\mathsf{LK}_{\mathrm{tsp}}$ tries to remove and insert edges in the tour but it explores only some parts of the $k$-opt neighborhood that deem to be the most promising. Consider removing an edge from a tour; this produces a path. Rearrange this path to minimize its weight. To close up the tour we only need to add one edge. Since we did not consider this edge during the path optimization, it is likely that its weight is neither minimized nor maximized. Hence, the weight of the whole tour is probably reduced together with the weight of the path. Here is a general scheme of $\mathsf{LK}_{\mathrm{tsp}}$:

1. Let $T$ be the original tour.
2. For every edge $e \to b \in T$ do the following:
   (a) Let $P = b \to \cdots \to e$ be the path obtained from $T$ by removing the edge $e \to b$.
   (b) Rearrange $P$ to minimize its weight. Every time an improvement is found during this optimization, try to close up the path $P$. If it leads to a tour shorter than $T$, save this tour as $T$ and start the whole procedure again.
   (c) If no tour improvement was found, continue to the next edge (Step 2).

In order to reduce the weight of the path, a local search is used as follows. On every move, it tries to break up the path into two parts, invert one of these parts, and then rejoin them (see Fig. 1). In particular, the algorithm tries every edge $x \to y$ and selects the one which maximizes the gain $g = w(x \to y) - w(e \to x)$. If the maximum $g$ is positive, the corresponding move is an improvement and the local search is applied again to the improved path.

Observe that this algorithm tries only the best improvement and skips the other ones. A natural enhancement of the heuristic would be to use a backtracking mechanism to try all the improvements. However, this would slow down the algorithm too much. A compromise is to use the backtracking only for the first $\alpha$ moves. This approach is implemented in a recursive function *ImprovePath*$(P, depth, R)$, see Algorithm 1.

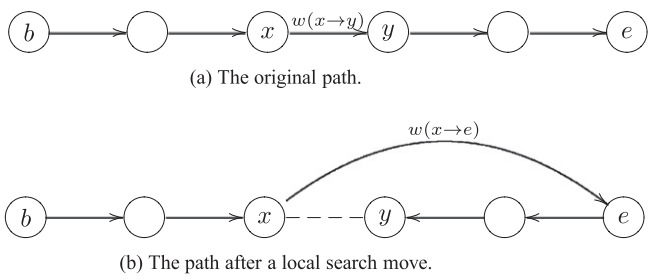


**Fig. 1.** An example of a local search move for a path improvement. The weight of the path is reduced by $w(x \to y) - w(x \to e)$.

**Algorithm 1.** *ImprovePath*$(P, depth, R)$ recursive algorithm ($\mathsf{LK}_{\mathrm{tsp}}$ version). The function either terminates after an improved tour is found or finishes normally with no profit

**Require**: The path $P = b \to \cdots \to e$, recursion depth *depth* and a set of restricted vertices $R$.
**if** $depth < \alpha$ **then**
  **for** every edge $x \to y \in P$ such that $x \notin R$ **do**
    Calculate $g = w(x \to y) - w(e \to x)$ (see Fig. 1b).
    **if** $g > 0$ **then**
      **if** the tour $b \to \cdots \to x \to e \to \cdots \to y \to b$ is an improvement over the original one **then**
        Accept the produced tour and **terminate**.
      **else**
        *ImprovePath*$(b \to \cdots \to x \to e \to \cdots \to y, depth + 1, R \cup \{x\})$.
**else**
  Find the edge $x \to y$ which maximizes $g = w(x \to y) - w(e \to x)$.
  **if** $g > 0$ **then**
    **if** the tour $b \to \cdots \to x \to e \to \cdots \to y \to b$ is an improvement over the original one **then**
      Accept the produced tour and **terminate**.
    **else**
      **return** *ImprovePath*$(b \to \cdots \to x \to e \to \cdots \to y, depth + 1, R \cup \{x\})$.