



Research Article

An Efficient Heuristic Tackling the Service Location Cloud Problem SLCP Exemplified at the Generalized Traveling Salesman Path Problem GTSP

Peter Hagen Richter* 

Department Development, O&S Consultancy, Berlin, Germany

Abstract

This paper gives an in-depth analysis of a heuristic tackling the Service Location Cloud Problem SLCP, whose efficient implementation is a necessary prerequisite to solve the Generalized Traveling Salesman Path Problem GTSP so far based on the recommended well-trying decomposition like Figure 1. For a given weighted directed graph, a partial assignment of clusters to graph nodes, a start s and a target t , the GTSP reads as follows: Find a shortest s - t -walk that visits each cluster at least once! And for the Asymmetric Steiner Traveling Salesman Path Problem ASTSP that doesn't need a cluster assignment but a given cloud Ω : Find a shortest s - t -walk that visits each member of Ω at least once! That means, if there is a SLCP heuristic tackling the objective "Find a cloud of nodes each assigned to a different cluster that enables a shortest s - t -walk through all clusters" then we can heuristically solve the GTSP: Find a good SLCP cloud and pass it to an ASTSP heuristic! Fortunately, there is already an efficient ASTSP heuristic we can rely on to test the performance of the SLCP algorithm ASLCP given below in the interest of good GTSP solutions. However, the SCLP's objective to find (a) a cloud Ω of different cluster nodes (b) that enables a shortest s - t -walk corresponds in (b) with the ASTSP that has to find a shortest s - t -walk through a given cloud Ω . I.e. from the NP-hardness of the ASTSP follows the NP-hardness of the SLCP. That means that we have to replace the sophisticated SLCP objective by a simplified one that hopefully brings about satisfiable results. This alternative objective reads as follows: Find a least cost "pivot" node x that minimizes the total length $s \rightarrow x \rightarrow t$ plus the sum of the minimum lengths $s \rightarrow x \rightarrow y \rightarrow t$ or $s \rightarrow y \rightarrow x \rightarrow t$ considering all nodes y each belonging to a different cluster! This pivot node x refers to a cloud of cluster nodes nearest situated to s , t , and x , that surprisingly well approaches the SLCP objective. Assured by extensive tests on random graphs, the performance analysis reveals a remarkable sample standard deviation SSD= 1,19% ascribed to SLCP-algorithm ASLCP. Accepting higher time consumption, a newly found second optimization feature improves the results further to an astonishing SSD= 0,22%!

Keywords

Applied Mathematics, Traveling Salesman Path Problem, Combinatorial Optimization, Meta-heuristics, Algorithm and Computational Design

*Correspondence: Peter Hagen Richter (peterhrichter@online.de)

Received: 5 October 2026; **Accepted:** 15 October 2026; **Published:** 28 October 2026

Copyright: © The Author(s), 2026. Published by Science Publishing Group. This is an **Open Access** article, distributed under the terms of the Creative Commons Attribution 4.0 License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

1. Introduction and Related Work

The SLCP reads as follows: Given a directed graph $G = [V_G, E_G]$, arc lengths $\lambda: E_G \rightarrow \mathbb{R}_{\geq 0}$, a pair (start s , target t) $\in V_G^2$, a set of services (clusters, classes,...) $\mathcal{C}$, a partial function $f: V_G \rightarrow \mathcal{C}$ uniquely assigning to some or all nodes services of $\mathcal{C}$: Find a cloud $\Omega \subseteq V_G$ of locations assigned to different clusters obeying $|\Omega| = |\mathcal{C}|$ and $f(\Omega) = \mathcal{C}$ that enables a shortest s - t -walk $\tilde{W} \subset G$ with $\Omega \subset V_{\tilde{W}}$. Correspondingly, the GTSP is the following: Determine a shortest s - t -walk $\tilde{W} \subset G$ such that $f(V_{\tilde{W}}) = \mathcal{C}$. And the ASTSP: For a given set $\Omega \subset V_G$, find a shortest s - t -walk $\tilde{W}$ such that $\Omega \subset V_{\tilde{W}}$.

SLCP is at least as hard as the ASTSP because it has to determine for an intermediary solution candidate Ω a shortest s - t -walk encountering each $x \in \Omega$ at least once what corresponds to the ASTSP. It follows that, like ASTSP, SLCP is NP-hard. Figure 1 shows the proposed decomposition of the GTSP into SLCP and ASTSP. The latter relies on the following metaheuristics [29-31]:

ASST “Advanced Scan of spanning Trees” scans a Steiner tree T spanning $\Omega \subset V_G$ for a first solution candidate π . T will be built by ASTSP algorithm A_{ASTSP} .

TSA “Tree Structure Adaption” refines T to an “improved” tree T' to get a routing π' with mostly less cost if T' will be scanned by ASST again.

CCE “Confined Complete Enumeration” directly improves routing π by partial permutations limited to small connected sequences of π .

This paper refers to the ASTSP publications [29, 30] and to the GTSP publication [28] describing the corresponding heuristics A_{GTSP} and A_{ASTSP} . The GTSP has been defined by Rice & Tsotras [25]. In contrast to our definition and implementation theirs one don't allow zero arc cost and they gave an exact algorithm (combinatorial explosion) and not a heuristic one. As they state, their algorithm solves many real-world problems in a matter of seconds on large-scale GTSP instances. Rice & Tsotras [25, 26] gave a solution approach tackling the Generalized Shortest Path GSP that looks for a variant of constrained shortest paths visiting at least one location each cluster in a specified order. Based on a new dynamic programming formulation the authors engineered a fast and scalable GSP query algorithm for use on large, real-world road networks with over 50 million edges. Rice & Tsotras [27] presented an exact and an approximate “fixed-parameter tractable” GTSP algorithm for real-world road networks. Astonishingly, the authors redefine here their (non-Hamiltonian) GTSP definition [25] to a Hamiltonian one: “The GTSP involves finding a least-cost tour through exactly one location from each of a set of generalized location categories”?! We adhere to their first definition [25] due to its wider range of applications. The GTSP goes back to the *Traveling Salesman Path Problem* TSP that was introduced by Hoogeveen [13]. It aims to find shortest Hamiltonian s - t -path through all nodes V_G in a complete undirected graph G . The author presented an

algorithm with an approx. ratio $k = \frac{5}{3}$, later improved by An [1] to $k = \frac{1 + \sqrt{5}}{2}$ and then by Zenklusen [42] to $k = \frac{3}{2}$. For the metric TSP (symmetric edge cost satisfying the triangle inequality), Seb ö [34] gave a $k = \frac{8}{5}$ approximation algorithm.

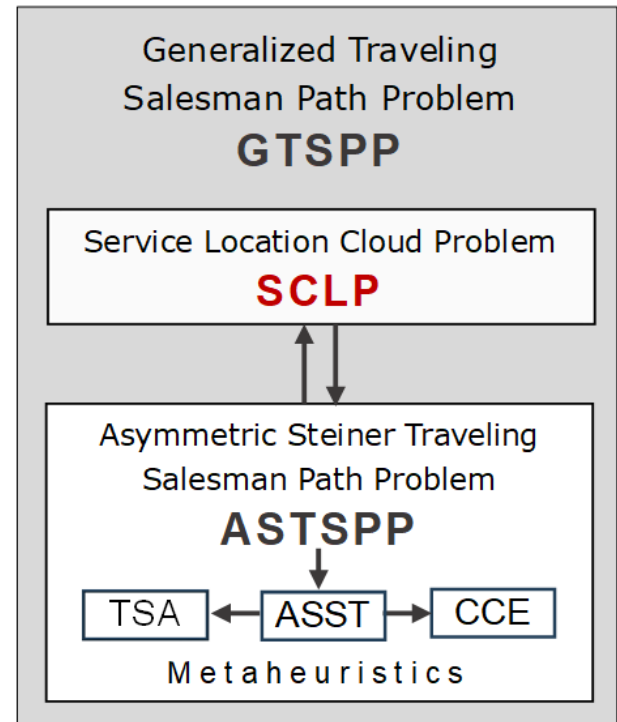


Figure 1. The decomposition of the GTSP.

The GTSP is referable to the Generalized Traveling Salesman Problem GTSP, Laporte & Nobert [16], that looks in complete graphs with nodes subdivided into disjoint clusters for a minimum cost Hamiltonian cycle (“International GTSP” or only GTSP). A similar version of this problem is called E-GTSP also known as “Equality GTSP”. It corresponds to the problem above with the exemption that each cluster must be visited exactly once. The E-GTSP was introduced by Henry-Labordere [12], Srivastava [37], Saksena [32], Noon and Bean [20] and has been substantiated by Laporte [17]. Both problems have many applications, Laporte [17], Fischetti [5, 6]. They are NP-hard because they reduce to the TSP if there is only one cluster assigned to all graph nodes.

Noon & Bean [21], Ben-Arieh [2], Dimitrijevic & Saric [4], Laporte [18] for the E-GTSP, and Lien et al. [19] for the GTSP transformed the problems to the classical TSP. However, Karapetyan & Gutin [14] allude that the obtained TSP instances have to produce exact solutions because even a near-optimal solution of such TSP may correspond to an infeasible GTSP solution. Still, Helsgaun [11] applied the Lin-Kernighan-Helsgaun-algorithm (LKH: Tinos [40]) and transformed E-

GTSP instances into asymmetric TSP instances using the Noon-Bean transformation [21] bringing all E-GTSP benchmark instances of the GTSPLIB to optimality in reasonable time. Fischetti [6] devised a GTSP branch-and-cut exact algorithm for $n=442$ and 89 clusters. The authors enlarged their studies [7] for both symmetric GTSP and E-GTSP based on exact branch-and-cut methods. Integer Linear Programming Formulations ILPF for the GTSP have been proposed by Laporte & Nobert [16] outperforming corresponding dynamic programming approaches for large problem sizes. Noon & Bean [20] succeeded in a combined approach of bounding, elimination and enumeration solving the asymmetric E-GTSP. Renaud & Boctor [24] proposed a 3-phase E-GTSP heuristic leading to the optima in 56% of 36 problems. Snyder & Daskin [36] delineated an E-GTSP algorithm based on a genetic algorithm combined with a local tour improvement. Karapetyan & Gutin [14] gave an efficient local search approach for the E-GTSP based on successive Cluster Optimization CO traced back to Fischetti [6]. Using Ant Colony Optimization some meta-heuristics have been published by Pintea [22] for the E-GTSP and Yanga [41] for the GTSP. Smith & Imeson [35] presented an effective E-GTSP algorithm based on adaptive large neighborhood search outperforming approaches particularly on harder instances that are non-metric. As to DNA Based Biological Approaches, Ren [23] proposed an $O(n^2)$ GTSP algorithm relying on the Adleman-Lipton model (described in [23]) applying strong parallel computing but requiring large storage. Tackling the GTSP, Lien [19] transformed graph G into another one G' with disjoint clusters to get a TSP instance mostly leading to the optima. A Memetic Algorithm MA (population based search combined with a neighborhood search) for the E-GTSP were given by Bontoux [3] whose approach uses a large neighborhood search up to 442 nodes. Gutin & Karapetyan [8, 10] outperformed the known MA heuristics at that time. For the GTSP, an MA heuristic based on a Breakthrough Local Search BLS has been suggested by Krari & Ahiod [15] with near-optimal results. Gutin & Karapetyan [9] introduced for the symmetric E-GTSP an $O(n^3)$ vertex and edge reduction algorithm significantly shrinking computation time. This can also be achieved by pre-processing procedures e.g. in integer and linear programming as shown by Savelsbergh [33]. Tan [39] gave a Genetic Algorithm GA converting the GTSP into the E-GTSP succeeding in solution quality and less run-time.

We only mention a different GTSP definition proposed by Sun & Gutin [38] that starts from a given complete graph G , $s, t \in V_G$, subsets $V' \subseteq V_G$ and $E' \subseteq E_G$. Its goal is to find a Hamiltonian s - t -path through all nodes V' traversing E' .

2. Further Notations and Definitions

$\xi \in \mathbb{N}$ is the random number starting seed.

$\lambda: E_G \rightarrow \mathbb{R}_0^+$ are the arc cost dependent on ξ .

$\mathcal{C}$ is a set of clusters, classes, services, categories partially

assigned to V_G of digraph $G=[V_G, E_G]$.

$\hat{f}: V_G \rightarrow \mathcal{C}$ implicitly given with the input of G and $\mathcal{C}$. Because s and t are always within the routing, their clusters are not to consider:

$\mathcal{C} = \mathcal{C} \setminus \{f(s), f(t)\}$ is the set of *Feasible Clusters*.

$f: V_G \rightarrow \mathcal{C}$ is the Feasible Assignment of Services to V_G arising from the Original Cluster Assignment

$\Phi = \text{dom}(f)$ is the set of all nodes of V_G assigned to any cluster of $\mathcal{C}$.

$n_{\mathcal{C}} = |\mathcal{C}|$ is the Number of all Clusters.

$n_{\mathcal{C}} = |\mathcal{C}|$ is the Number of Feasible Clusters.

$P_G(s, t) \subseteq G$ is a *Directed Shortest Path* from s to t .

$d_G(s, t) = \sum_{e \in P_G(s, t)} \lambda(e)$ is the *Length of Path* $P = P_G(s, t)$.

$\pi: \{0, 1, \dots, |\Omega|+1\} \rightarrow \Omega \cup \{s, t\}$ with $\pi(0) = s, \pi(|\Omega|+1) = t$ is the *Routing* with cost $C(\pi) = \sum_{i=0}^{|\Omega|} d_G(\pi(i), \pi(i+1))$.

T is a tree spanning Ω . Cost $C(T) \leq 2 C_{\text{opt}}(1 - \frac{1}{|\Omega|})$, [34].

A – array with extension $n_{\mathcal{C}} \times 2$. $A[\mathcal{C}_y, 1]$ is that node y_1 of cluster $\mathcal{C}_y$ that is nearest situated to pivot x , $A[\mathcal{C}_y, 2]$ is that node y_2 of cluster $\mathcal{C}_y$ that is second- nearest situated node to x after y_1 .

$\mathbb{B}$ –array with extension $n_{\mathcal{C}} \times 2$: $\mathbb{B}[\mathcal{C}_y, 1]$ – connection cost $\kappa(x) + \check{\kappa}(x, y_1)$, $\mathbb{B}[\mathcal{C}_y, 2]$ – connection cost $\kappa(x) + \check{\kappa}(x, y_2)$.

Ω with $|\Omega| = n_{\mathcal{C}}$ and $f(\Omega) = \mathcal{C}$ is a service location cloud built by SLCP-algorithm A_{SLCP} .

Q_x is a least cost cloud around node x (x called pivot node) w. r. t. s and t such that $f(Q_x) = \mathcal{C}$ and $|Q_x| = n_{\mathcal{C}}$.

$\mathcal{H}$ is a binary heap to contain all $x \in \Phi$ w. r. t. their cost $C(Q_x)$, taking advantage of heap operations *InsertHeap*($x, C(Q_x)$) - Inserts x and $C(Q_x)$ into $\mathcal{H}$;

FindMinDel() – Finds and deletes from $\mathcal{H}$ that x with minimal cost $C(Q_x)$ running with $O(|\Phi| \log |\Phi|)$.

$\text{opt} \in \{1, 2\}$ (1 default) is a parameter steering the optimization level for SLCP algorithm A_{SLCP} , explained by Table 1.

3. SLCP-Algorithm A_{SLCP}

A_{SLCP} is expected to find a cloud Ω with $|\Omega| = |\mathcal{C}|$ and $f(\Omega) = \mathcal{C}$ whose nodes ensure a shortest s - t -walk visiting all nodes of Ω at least once. This discerning task contains not the determination of a corresponding layout! However, A_{SLCP} uses a substitute objective that meets the defined SLCP objective surprisingly well:

Find a node $x \in \Phi$ called pivot node, that represents at best a center for nearest situated nodes belonging to different clusters within the vicinity of x, s, t . building cloud Q_x with $|Q_x| = n_{\mathcal{C}}$ and $f(Q_x) = \mathcal{C}$.

We start with distance functions κ and $\check{\kappa}$ describing how good one node $x \in \Phi$ or two nodes ($x, y \neq x$) are situated between s and t for a shortest connection as to

$$s \rightarrow x \rightarrow t \text{ and } \begin{matrix} s \rightarrow x \rightarrow y \rightarrow t \text{ or} \\ s \rightarrow y \rightarrow x \rightarrow t \end{matrix}$$

The connection costs of a pivot node x is

$$\kappa(x) = d_G(s, x) + d_G(x, t). \quad (1)$$

The connection costs of x and another node y is

$$\check{\kappa}(x, y) = \min \left\{ \begin{array}{l} d_G(s, x) + d_G(x, y) + d_G(y, t) \\ d_G(s, y) + d_G(y, x) + d_G(x, t) \end{array} \right\}. \quad (2)$$

Table 1. Algorithm A_{SCLP} corresponding to its implementation.

1	$A_{SCLP} \{$	Algorithm A_{SCLP} builds clouds Ω with $ \Omega = n_{\mathbb{C}}$ and $f(\Omega) = \mathbb{C}$.
2	$\Phi' := \Phi; \eta := \frac{ \Phi \cdot limit}{100\%}; \text{Make-Heap}(\mathcal{H}(\Phi); j := 0; C_{min} = -1;$	Φ' - all nodes assigned to a cluster; η - limits the heap for the η -best pivot nodes; $\mathcal{H}$ - binary heap for pivot nodes sorted by their connection cost.
3	$\text{while}(\Phi' \neq \emptyset) \{$	As long as Φ' (a copy of Φ) is not empty:
4	$x \in \Phi'; \Phi' \setminus = \{x\};$	Take any node x from Φ' and remove it from Φ' .
5	$C(Q_x) = \text{Cloud}(x);$	Call $\text{Cloud}(x)$ for pivot x to get the cost $C(Q_x)$.
6	$\mathcal{H}.Insert(x, C(Q_x));$	Insert $(x, C(Q_x))$ into heap $\mathcal{H}$.
7	$\}$	
8	$\text{while}(x := \mathcal{H}.FindMinDel()) \{$	As long as $\mathcal{H}$ is not empty ($x \neq 0$), get the current least cost pivot x .
9	$Q_x := \text{Cloud}(x); k = 1;$	$\text{Cloud}(x)$ determines the least cost cloud w. r. t. κ and $\check{\kappa}$.
10	$(\pi, C(\pi)) := \text{App}(Q_x);$	Let the application determine the s - t -walk π and its cost $C(\pi)$.
11	$\text{if}(C_{min} = -1 \vee C(\pi) < C_{min}) \{$	Update the best cloud Ω that enables $\text{App}(Q_x)$ building the present least cost routing π .
12	$C_{min} := C(\pi); \Omega := Q_x \}$	
13	$\text{if}(opt = 2) \text{ do } \{$	Optimization $opt = 2$ improves results only if better than $opt = 1$.
14	$\text{if}(k = f(s) \vee k = f(t)) \text{ continue};$	If s or t belong to a cluster these clusters are not to consider.
15	$x' = A[k, 1]; A[k, 1] = A[k, 2];$	The current best node $[k, 1]$ retained for cluster k will be replaced by its second best $A[k, 2]$ in the original cloud Q_x .
16	$Q_x := \bigcup_{c \in \mathbb{C}} \{A[c, 1]\};$	The original cloud will be restored.
17	$A[k, 1] := x';$	
18	$(\pi, C(\pi)) := \text{App}(Q_x);$	Let the application determine the s - t -walk π and cost $C(\pi)$.
19	$\text{if}(C(\pi) < C_{min}) \{$	Update the current best cloud Ω .
20	$C_{min} := C(\pi); \Omega := Q_x \}$	
21	$\text{while}(++k \leq n_{\mathbb{C}}) \}$	All clusters $1 \leq k \leq n_{\mathbb{C}}$ are to consider ($k = 1$ initialized step 9).
22	$\text{if}(++j \geq \eta) \text{ break};$	$limit > 50\%$ rarely brings a better routing π than $limit \leq 50\%$.
23	$\}$	
24	$\text{App}(\Omega);$	Finally, call the envisaged application with the best cloud Ω .
25	$\}$	We used for "Application" App algorithm A_{ASTSP} !
26		
27	$\text{Cloud}(x) \{$	$\text{Cloud}(x)$ returns a cloud of $n_{\mathbb{C}}$ nodes of all clusters.
28	$\Phi'' := \Phi;$	Keep a copy Φ'' of the set of all nodes assigned to a cluster.
29	$\forall c \in \mathbb{C}: \{A[c, 1] := A[c, 2] := 0;$	Initialize each cluster the spot for the 1. and 2. best node y for x .
30	$A[f(x), 1] := x; B[f(x), 1] := \kappa(x);$	Store x with its connection cost $\kappa(x)$ into cluster shelf $[f(x), 1]$.
31	$\text{while}(\Phi'' \neq \emptyset) \{$	As long as Φ'' is not empty:
32	$y \in \Phi''; \Phi'' \setminus = \{y\};$	Consider each $y \in \Phi''$ and remove it from Φ'' .

33	if($x = y$) continue;	x has already been stored above, step 4.
34	$c_y := f(y)$;	$c_y := f(y)$ is the cluster y has been assigned for.
35	$\delta := \check{k}(x) + \check{k}(x, y)$; $k := 0$;	δ is the cost connecting $\check{k}(x) + (s \rightarrow x \rightarrow y \rightarrow t)$ or $(s \rightarrow y \rightarrow x \rightarrow t)$.
36	while($++k \leq opt$) {	Consider both optimizations $opt = 1$ (default) and $opt = 2$
37	if($A[c_y, k] = 0 \vee \delta < B[c_y, k]$) {	
38	$A[c_y, k] := y$; $B[c_y, k] := \delta$;	If optimization feature $opt = 2$ then y will be stored as second-best c_y - partner for pivot x only if its cost doesn't undercut that of the first best c_y - partner. An underbidding control for ($[c_y, 2], B[c_y, 2]$) is omitted if $opt = 1$ or if ($A[c_y, 1], B[c_y, 1]$) was just improved (13: "break").
39	break;	
40	}	
41	}	
42	}	
43	$\hat{Q}_x := \bigcup_{c \in \mathcal{C}} \{A[c, 1];$	$\hat{Q}(x)$ is a cloud of nodes of different clusters assigned to pivot node x . $C(\hat{Q}_x)$
44	$C(\hat{Q}_x) := \sum_{c \in \mathcal{C}} B[c][1]$;	are the corresponding connection costs. Both $\hat{Q}(x)$ and $C(\hat{Q}_x)$ are to be made available for $ASLCP$: 5 and 9.
45	}	

I.e. for a fixed pivot $x \in \Phi$ only that node $y \in f^{-1}(c)$ of cluster c with $f(x) \neq c = f(y)$ is best situated to x , s and t that causes the least distance

$$\check{k}(x, c) = \min_{y \in f^{-1}(c)} \left\{ d_G(s, x) + d_G(x, y) + d_G(y, t) \right\} \quad (3)$$

The best node y of cluster c (via emerging function g) is

$$y_{x,c}^* = g(x, c) = \arg \min_{y \in f^{-1}(c), f(x) \neq f(y)} \{ \check{k}(x, c) \}. \quad (4)$$

Accordingly, a complete cloud Q_x for pivot x consisting of all cluster nodes nearest situated to x , s , t is

$$Q_x = \bigcup_{c \in \mathcal{C}} \{g(x, c)\} \subset V_G \text{ with cost} \quad (5)$$

$$C(Q_x) = \check{k}(x) + \sum_{c \in \mathcal{C}} \check{k}(x, c). \quad (6)$$

Thus, the best pivot node is

$$\bar{x} = \arg \min_{x \in Q} (\check{k}(x) + \sum_{c \in \mathcal{C}} \check{k}(x, c)). \quad (7)$$

Hence, the least cost cloud is

$$Q_{\bar{x}} = \bigcup_{c \in \mathcal{C}} \{g(\bar{x}, c)\} \text{ with cost} \quad (8)$$

$$C(Q_{\bar{x}}) = \check{k}(\bar{x}) + \sum_{c \in \mathcal{C}} \check{k}(\bar{x}, c). \quad (9)$$

However, $Q_{\bar{x}}$ meets not directly the SLCP objective to enable a shortest s - t -walk (visiting each cloud member at least once), what is confirmed by algorithm A_{ASTSPP} serving here as

an arbitrary application $App(Q_x)$, Table 1. As the tests show, there are clouds Q_x having cost $C(Q_x) < C(Q_{\bar{x}})$. Further, there are few cases, were not the first but the second best neighbor $y_{x,c}^{**}$ outperforms $y_{x,c}^*$ providing a better routing π . Therefore, the following measures proved very efficient:

The introduction of a binary heap, to get only the best 30 – 50 % clouds Q_x whose real route length has to be proved by $App(Q_x)$, see Table 1: $ASLCP$ step 21.

The determination and storing not only the best neighbor $y_{x,c}^*$ each cluster for pivot x but also the second best one $y_{x,c}^{**}$. This feature controlled by parameter $opt = 2$ (default $opt = 1$) induces $ASLCP$ to additionally replace $y_{x,c}^*$ by $y_{x,c}^{**}$ each shelf $[k, 1]$ of cluster k , see Table 1: $ASLCP$ 13- 20.

Doing so (and anticipating results),

reaches a Sample Standard Deviation SSD of 1,19%.

reaches an SSD even of 0,22% accepting that the run-time considerably increases with the number $n_{\mathcal{C}}$ of clusters, Figure 3.

Table 1 shows how Cloud(x) builds a cloud Q_x for the passed pivot x . Cloud(x) considers all neighbors y of x to prove whether the cost $\delta := \check{k}(x) + \check{k}(x, y)$ are smaller than that of the stored $y' = A[c_y, 1]$ with cost $B[c_y, 1]$ in the shelves A and B of cluster $c_y = f(y) = f(y')$. If $opt = 2$ and δ is not smaller than $[c_y, 1]$ then Cloud(x) proves whether $\delta < B[c_y, 2]$ in order to update the second best neighbor of x , Table 1: Cloud(x) 10-15. If all neighbors y have been processed, cloud $\hat{Q}_x = \bigcup_{c \in \mathcal{C}} \{[c, 1]\}$ with cost $C(\hat{Q}_x) := \sum_{c \in \mathcal{C}} B[c][1]$ is passed to $ASLCP$. There, $(x, C(Q_x))$ will be set into a priority queue organized as binary heap $\mathcal{H}$, Table 1: $ASLCP$ 3-7. Then a limited number η of least cost clouds will be passed to the genuine application, here represented by $App(Q_x)$ that returns a layout π with cost $C(\pi)$.

4. Performance Analysis

4.1. Test Graphs

The classes $\mathcal{C}$ are uniformly distributed over V_G realized

by a random graph generator. Although the classes $\mathcal{C}$ are disjoint, the corresponding node regions $dom(\hat{f}) \subset V_G$ are overlapping! Graphs $G_1, \dots, G_7$, Table 2, consist of rectangular compact grids without inner holes and high density $D = \frac{2012/2}{527} = 1,9$.

Table 2. Test graphs applied for the performance analysis.

Digraphs $G_1, G_2, \dots, G_6$:	Digraph G_7 :
$ V_G = 527, E_G = 2010. \lambda: E_G \rightarrow [0, 40] \subset \mathbb{R}_{\geq 0}$	$ V_G = 527, E_G = 2010. \lambda: E_G \rightarrow [0, 40] \subset \mathbb{R}_{\geq 0}$.
$n_{\mathcal{C}} = 6$ clusters, limit= 31%	10 cases of different clusters $n_{\mathcal{C}} \in \{3, 6, 9, \dots, 30\}$
7 members each cluster.	5 members each cluster. limit= 31%
Settings: $s = t = \xi$ with $1 \leq \xi \leq 510$	Settings: $s = t = 219$, starting seed $\xi = 999$.
For each graph the following combination of meta-heuristics have been considered: opt= 1: a) ASST, b) ASST+CCE, c) ASST+CCE+TSA. opt= 2: d) ASST+CCE+TSA	
For each combination a),.. d) there are 10 columns assigned to each different $s = t = \xi$ with the results	For each combination a),.. d) there are 10 columns assigned to each different $n_{\mathcal{C}}$ with the results
• routing cost of π, • SSD in %, • runtime in ms.	• routing cost of π, • runtime in ms.

4.2. Results

Table 3 shows, that the cooperation of meta-heuristics ASST+CCE+TSA is a ripshorter lowering the average Sample Standard Deviation SSD from 5,27% to 1,19% with a time consumption 115 ms. However, ASST, CCE and TSA are used by A_{ASTSP} acting for the general application $App(Q_x)$ that has no influence on the determination of cloud Q_x . That is the task of SLCP algorithm A_{SLCP} . It co-determines the final routing π . Despite the good SSD of maximum 1,94 %, analyzing algorithm Cloud(x) revealed a genuine improvement possibility: Backtracking this observation to the cloud Q_x

showed that Cloud(x) had got the optimal cloud if it had not taken $y_{x,c}^*$ but the second-best neighbour $y_{x,c}^{**}$ for pivot x of class $\mathcal{C}$. Thus, a new selectable optimization feature has been introduced called “opt= 2”: Table 1: Cloud(x) 10 – 13, A_{SLCP} 13- 20. It decreases SSD further from 1,19% (opt= 1) to 0,22% for opt=2, Figure 2. But it increases complexity from $O(n_{\mathcal{C}} \cdot O(App))$ to $O(n_{\mathcal{C}}^2 \cdot O(App))$. Figure 3 (logarithmic presentation) confirms this: The time demand for opt= 2 confines real-time apps (≤ 2.000 ms) to use problem sizes $n_{\mathcal{C}} < 11$, but opt= 1 $n_{\mathcal{C}} < 26$. To tackle this dilemma, apps have to relinquish insisting on highest result quality by using opt= 1 with ASST+CCE.

Table 3. Performance of A_{SLCP} accessing 40% of heap $\mathcal{H}$.

	G1	G2	G3	G4	G5	G6	
	Sample Standard Deviation / %						
	opt= 1						
ASST	5,29	5,75	6,16	5,54	6,06	2,80	5,27
ASST+CCE	1,77	2,79	0,88	0,59	2,49	1,55	1,68
ASST+CCE +TSA	1,36	1,64	0,60	0,59	1,94	1,02	1,19
	opt= 2						
ASST+CCE +TSA	0,00	0,00	0,39	0,55	0,32	0,05	0,22

	G ₁	G ₂	G ₃	G ₄	G ₅	G ₆	
Nominal Time Effort / ms							
opt= 1							
ASST	26	28	32	33	30	25	29
ASST+CCE	27	32	37	32	32	29	31
ASST+CCE +TSA	107	111	108	132	122	108	115
opt= 2							
ASST+CCE +TSA	627	616	641	753	732	689	676

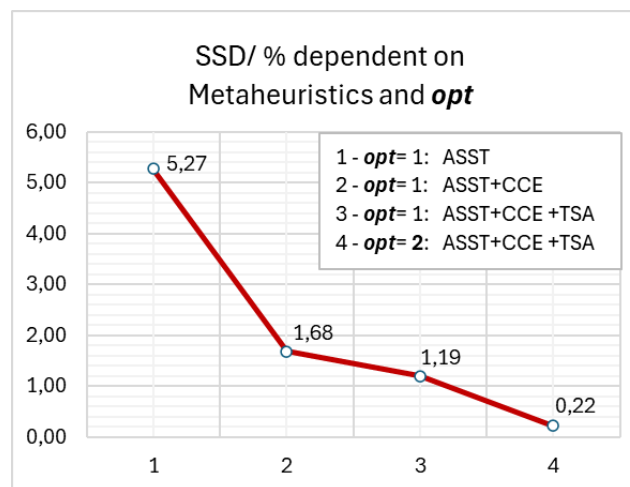
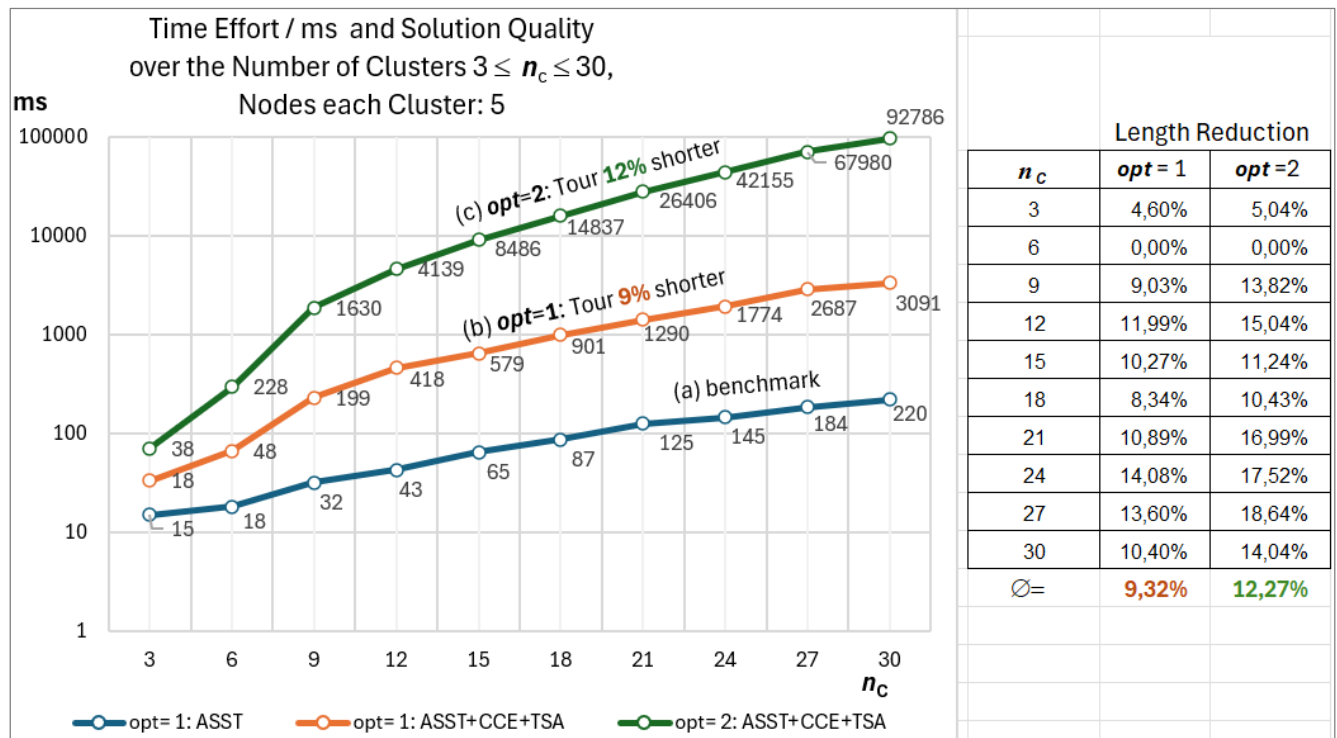


Figure 2. SSD curve related to Table 3.

Figure 3. Nominal time effort of AG_{TSPP} running on graph G_7 dependant on SLCP parameter *opt*.

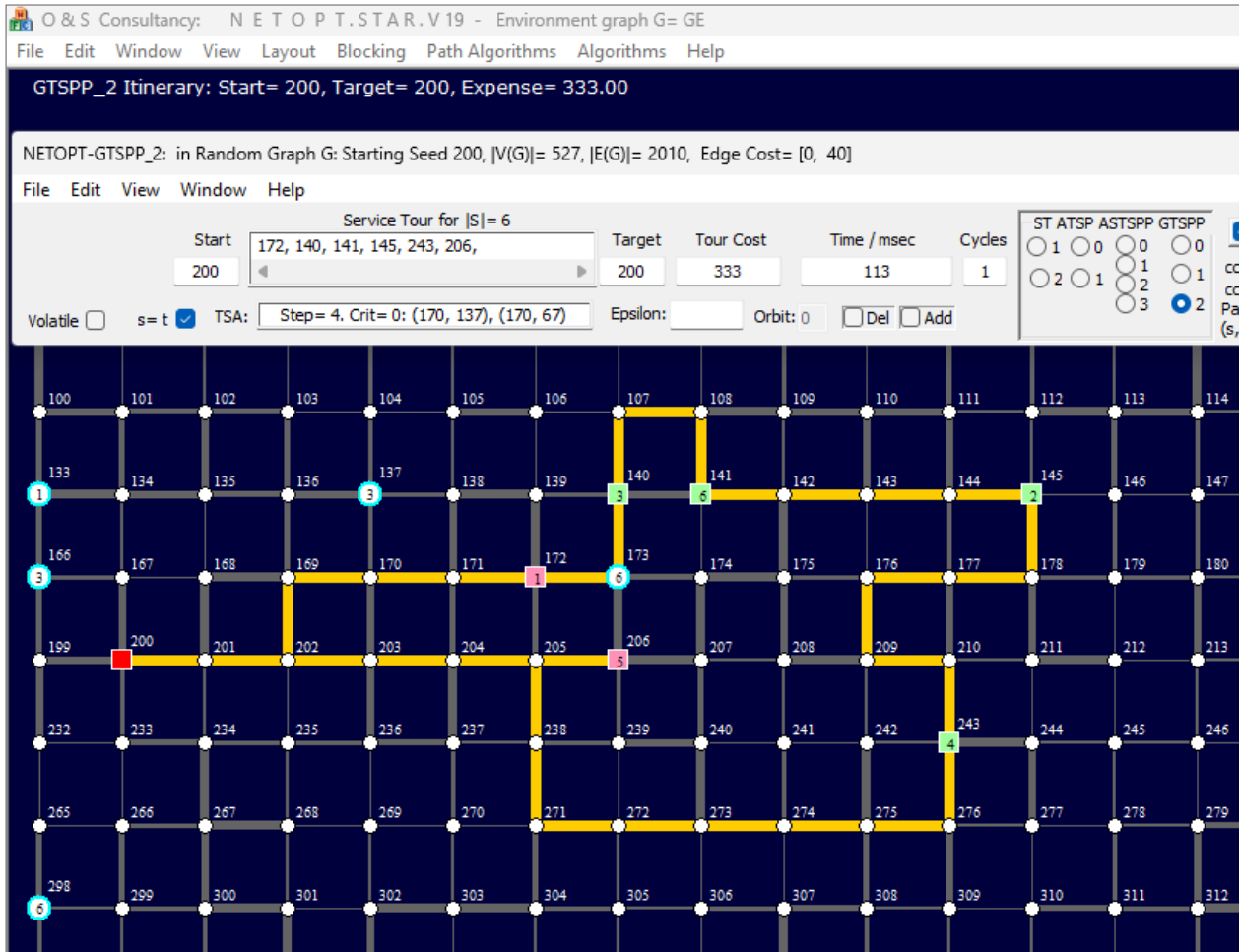


Figure 4. GTSP tour $200 \rightarrow 172 \rightarrow 140 \rightarrow 141 \rightarrow 145 \rightarrow 243 \rightarrow 206 \rightarrow 200$ got with $opt = 1$ and $limit = 40\%$ on graph G_1 with ASST+ CCE + TSA: Cloud $Q_x = \{172, 140, 141, 145, 243, 206\}$ derived from $ASLCP$ covers the clusters $\{1, 2, \dots, 6\}$ and enables A_{ASTSP} to find an optimal tour ($SSD = 0\%$) of length 333 needing a runtime 113 ms. In the interest of a shortest walk, node 173 / cluster 6 is not bypassed but 141 / cluster 6 belongs to the solution.

5. Discussion

The objective defined for the SLCP to find a shortest s - t walk $\tilde{W} \subset G$ visiting nodes of V_G assigned to different clusters sufficing $f(V_{\tilde{W}}) = \mathcal{C}$ deviates from the objective of problem SLCP that wangles about with a densest cloud of nodes nearest situated among each other as well as to start s and target t . Further analyses could avoid the trick calling the application $App(Q_x)$ getting the result π directly. This avoidance would reduce run-time considerably with regard to $opt = 2$ and higher problem sizes.

6. Conclusion

Tackling the Generalized Traveling Salesman Path Problem GTSP via the proposed decomposition, the decisive task for each GTSP heuristic is the efficient determination of a set ("cloud") $\Omega \subset V_G$ with $f(\Omega) = \mathcal{C}$ meeting the SLCP objective. The proposed algorithm A_{SLCP} tries to approach this objective

efficiently determining a densest set of service locations (cluster members) Ω integrating all classes such that the sum of the nodes' distances among each other as well as to start s and target t is minimal. This is efficiently realized by a novel algorithm Cloud(). Algorithm A_{SLCP} has been enhanced further by a new optimization feature " $opt = 2$ " that decreases the SSD of the hitherto used algorithm A_{SLCP} ($opt = 1$) to 1/5! However, Figure 3 shows that for $n_{\mathcal{C}} = 30$ the runtime demand for (c) $opt = 2$ is 30-times magnified than that for (b) $opt = 1$. I.e. real-time capability (≤ 2.000 ms) for $opt = 2$ can only be granted for problem sizes $n_{\mathcal{C}} \leq 10$, but $n_{\mathcal{C}} \leq 25$ for $opt = 1$, see Figure 3.

Algorithm A_{GTSP} based on A_{SLCP} and A_{ASTSP} meets the following requirements:

- 1) Graph G must not necessarily be complete.
- 2) Overlapping cluster regions are permitted.
- 3) Asymmetric instead of symmetric arc cost.
- 4) $\Phi \subseteq V_G$ is generally a proper subset of V_G .
- 5) Service locations might be visited more than once.
- 6) Routes from a start s to a target $t \neq s$.
- 7) Graphs must not suffice the triangle inequality.

- 8) Crossings' turn restrictions are to consider.
- 9) Open or closed routes to build with same efficiency.
- 10) Drawing advantage from using planar graphs

The test graphs used, as well as the proposed algorithms fulfill all the 10 demands above.

Abbreviations

A_{ASTSP}	Algorithm Tackling the ASTSP
APP	An Arbitrary App for the ASTSP
A_{SLCP}	Algorithm Tackling the SLCP
ASST	Advanced Scan of Spanning Trees
ASTSP	Asymmetric Steiner Traveling Salesman Path Problem
CCE	Confined Complete Enumeration
$dom(f)$	Domain of a Function f
E_G	Set of All Edges of Graph G or Arcs of digraph G
E-GTSP	Equality GTSP
G	Directed Graph
GSP	Generalized Shortest Path
GTSP	Generalized Traveling Salesman Problem
GTSP-Lib	Collection of Instances for the GTSP
GTSPPP	Generalized Traveling Salesman Path Problem
ms	Milliseconds
NP	Non-deterministically in Polynomial-time Solvable
$O(expr)$	Worst Case Complexity of Expression x
opt	Optimization level 1 $\leq opt \leq 2$ for ASLCP
SLCP	Service Location Cloud Problem
SSD	Sample Standard Deviation
TSA	Tree Structure Adaption
TSP	Traveling Salesman Problem
V_G	Set of all nodes of graph G

Author Contributions

Peter Hagen Richter: Investigation, Conceptualization, Methodology, Project administration, Data curation, Software, Visualization, Writing original draft, Writing review and editing

Conflict of Interest

The author declares no conflict of interest.

References

- [1] AN H-C, Kleinberg R and Shmoys D-B, "Improving Christofides' algorithm for the s-t path TSP". In: Proceedings of the 44th annual ACM Symposium on Theory of computing, pp. 875-886, 2012.
- [2] Ben-Arieh D., Gutin G., Penn M., Yeo A., and Zverovitch A. "Transformations of generalized ATSP into ATSP", Operations Research Letters 31, pp. 357–365, 2003.
- [3] Bontoux B., Artigues C., Feillet D. "A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem". Computers & Operations Research 37, pp. 1844–1852, 2010.
- [4] Dimitrijevic V., Saric Z. "An Efficient Transformation of the Generalized Traveling Salesman Problem into the Traveling Salesman Problem on Digraphs". Inf. Sci. 102(1-4): pp. 105-110, 1997.
- [5] Fischetti M., Salazar Gonzales J. J., Toth P. "The Symmetric Generalized Traveling Salesman Polytope", Networks 26, pp. 113–123, 1995.
- [6] Fischetti, M., Salazar Gonzales J. J., Toth, P. "A Branch-and-Cut Algorithm for the Symmetric Generalized Travelling Salesman Problem"; Oper. Res. 45, 3, pp. 378-394, 1997.
- [7] Fischetti, M., Salazar Gonzales J. J., Toth, P. "The Generalized Traveling Salesman and Orienteering Problem". In The Traveling Salesman Problem and Its Variations (Gutin G., Punnen A., eds.), Kluwer, Dordrecht, 2002.
- [8] Gutin G., Karapetyan D., and Krasnogor N. "A memetic algorithm for the asymmetric generalized traveling salesman problem". In Proc. NISCO 2007, Studies in Computational Intelligence 129, Springer, pp. 199–210, 2008.
- [9] Gutin G., Karapetyan D. "Generalized Traveling Salesman Problem Reduction Algorithms", Algorithmic Operations Research 4, pp. 144-154, 2009
- [10] Gutin G., Karapetyan D. "A memetic algorithm for the generalized traveling salesman problem". Natural Computing 9(1), pp. 47-60, 2010
- [11] Helsgaun K. "Solving the equality generalized traveling salesman problem using the Lin-Kernighan-Helsgaun Algorithm", Computer Science Mathematical Programming Computation, K.: <https://doi.org/10.1007/s12532-015-0080-8> Corpus ID: 17808092, 2015
- [12] Henry-Labordere "The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem", RAIRO Operations Research, B2, pp. 43-49, 1969.
- [13] Hoogeveen J-A. "Analysis of Christofides' heuristic: some paths are more difficult than cycles". Operations Research Letters, 10(5) pp. 291-295, 1991.
- [14] Karapetyan D., and Gutin G. "Efficient local search algorithms for known and new neighbourhoods for the generalized traveling salesman problem", EJOR 219, pp. 234-251, 2012.
- [15] Krari M. E., Ahiod B. "A Memetic Algorithm Based on Breakout Local Search for the Generalized Travelling Salesman Problem", Applied Artificial Intelligence, Volume 34, Issue 7, 2020.
- [16] Laporte, G., Nobert, Y. "Generalized traveling salesman problem through n sets of nodes: An integer programming approach"; INFOR 21, 1, pp. 61–75, 1983.

- [17] Laporte G., Asef-Vaziri A., Sriskandarajah C. "Some Applications of the Generalized Traveling Salesman Problem", *Journal of the Operational Research Society* 47 pp. 1461-1467, 1996.
- [18] Laporte, G., Semet F. "Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem". *INFOR: Information Systems and Operational Research* 37(2) Taylor & Francis, pp. 114–20, 1999.
- [19] Lien Y., Ma E., Wah B.W.S. "Transformation of the Generalized Traveling Salesman Problem into the Standard Traveling Salesman Problem", *Information Sciences* 74, pp. 177-189, 1993.
- [20] Noon C. E., Bean J. C. "A Lagrangian based approach for the asymmetric generalized traveling salesman problem", *Operations Research*, Vol. 39, pp. 623-632, 1991.
- [21] Noon C. E., Bean J. C. "An Efficient Transformation of the Generalized Traveling Salesman Problem". *INFOR* 31, pp. 39–44, 1993
- [22] Pinteia Camelia-M., Pop Petrică C., Chira Camelia "The generalized traveling salesman problem solved with ant algorithms", *Complex Adapt Syst Model*, <https://doi.org/10.1186/s40294-017-0048-9> 2017.
- [23] Ren X., Wang X., Wang Z., Wu T. "Parallel DNA Algorithms of Generalized Traveling Salesman Problem-Based Bioinspired Computing Model", *International Journal of Computational Intelligence Systems*, Volume 14, Issue 1, pp. 228 – 237, 2021.
- [24] Renaud J., Boctor F. F. "An Efficient Composite Heuristic for the Symmetric Generalized Traveling Salesman Problem", *European Journal of Operational Research* 108(3), pp. 571-584, 1998.
- [25] Rice, M.N., Tsotras, V.J. "Exact Graph Search Algorithms for Generalized Traveling Salesman Path Problems". In: Klasing, R. (eds) *Experimental Algorithms*. SEA 2012. *Lecture Notes in Computer Science*, vol 7276. Springer, Berlin, Heidelberg, 2012. https://doi.org/10.1007/978-3-642-30850-5_30
- [26] Rice M.N., Tsotras V. J. "Engineering Generalized Shortest Path queries", *IEEE 29th International Conference on Data Engineering (ICDE)*, Brisbane, 2013.
- [27] Rice M. N., Tsotras V. J. "Parameterized algorithms for generalized traveling salesman problems in road networks", *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 114 – 123, 2013.
- [28] Richter P. H. "The Generalized Traveling Salesman Path Problem GTSP - An Impactful Construction Algorithm", *ECCO XXXVII CONFERENCE 2024 June 6th - 8th 2024*, KU Leuven - Ghent Belgium, available at ResearchGate and www.os-consultancy.de, 2024.
- [29] Richter P. H. "Multiple Destination Tours' and Paths' Efficient Determination in Directed Graphs", ISBN 978-1-63648-284-2, <https://doi.org/10.13140/RG.2.2.20932.14726> ELIVA PRESS, 2021.
- [30] Richter P. H. "The Asymmetric Steiner Traveling Salesman Path Problem ASTSPP - Open and Closed Tours' Efficient Determination in General Digraphs", *ecco2021*, Madrid 10.-11.06.2021, also at ResearchGate, 2021.
- [31] Richter P. H. "Present Approximate Algorithms for Steiner's Problem in Graphs – Classification and Two New Fast Approaches", *Systems Science (PL ISSN 0137-1223)*; Volume 17 - Number 2, pp. 2-28, 1991.
- [32] Saksena J. P. "Mathematical model of scheduling clients through welfare agencies", *Journal of the Canadian Operational Research Society*, Vol. 8, pp. 185-200, 1970.
- [33] Savelsbergh M.W.P. "Pre-processing and probing techniques for mixed integer programming problems". *ORSA Journal on Computing* 6, pp. 445–454, 1994.
- [34] Sebö A. "Eight fifth approximation for TSP paths". In: Goemans M., Correa J. (eds) *Integer Programming and Combinatorial Optimization 2013*, LNCS, vol 7801, Springer, Berlin, Heidelberg, pp. 362-374, 2013.
- [35] Smith S. L., Imeson F. "An effective large neighbourhood search heuristic for the generalized traveling salesman problem", *Computers & Operations Research*, vol. 87, pp. 1–19, 2017.
- [36] Snyder L. V., Daskin M. S. "A random-key genetic algorithm for the generalized Traveling salesman problem", *EJOR* 174, pp. 38-53, 2006.
- [37] Srivastava S. S., Kumar, Garg R. C., Sen P. "Generalized traveling salesman problem through n sets of nodes", *CORS Journal*, Vol. 7, pp. 97-101, 1969.
- [38] Sun J., Gutin G., Li P., Shid P., Zhange X. "An LP-based approximation algorithm for the generalized traveling salesman path problem", *Theoretical Computer Science* 941 pp. 180-190.
- [39] Tan Y., Hao Z.F., Huang H., Zhao S. (2013) Genetic algorithm of distance matrix remodelling for solving the second type of GTSP, *J. South China Univ. Technol.*, 2023.
- [40] Tinós R., Helsgaun K., Whitley D. "Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic". *PPSN XV, Proceedings I*, pp. 95-107, 2018.
- [41] Yanga, J., Shi, X., Marchese, M., Liang, Y. "An ant colony optimization method for generalized TSP problem". *Progress in Natural Science* 18(11), pp. 1417–1422, 2008.
- [42] Zenklusen R-A. "1.5-approximation for path TSP". In: *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1539-1549, 2019.