A BEAM SEARCH FOR THE EQUALITY GENERALIZED SYMMETRIC TRAVELING SALESMAN PROBLEM

Ibtissem Ben Nejma¹ and Rym M’Hallah²,*

Abstract. This paper studies the equality generalized symmetric traveling salesman problem (EGSTSP). A salesman has to visit a predefined set of countries. S/he must determine exactly one city (of a subset of cities) to visit in each country and the sequence of the countries such that s/he minimizes the overall travel cost. From an academic perspective, EGSTSP is very important. It is NP-hard. Its relaxed version TSP is itself NP-hard, and no exact technique solves large difficult instances. From a logistic perspective, EGSTSP has a broad range of applications that vary from sea, air, and train shipping to emergency relief to elections and polling to airlines’ scheduling to urban transportation. During the COVID-19 pandemic, the roll-out of vaccines further emphasizes the importance of this problem. Pharmaceutical firms are challenged not only by a viable production schedule but also by a flawless distribution plan especially that some of these vaccines must be stored at extremely low temperatures. This paper proposes an approximate tree-based search technique for EGSTSP. It uses a beam search with low and high level hybridization. The low-level hybridization applies a swap based local search to each partial solution of a node of a tree whereas the high-level hybridization applies 2-Opt, 3-Opt or Lin-Kernighan to the incumbent. Empirical results provide computational evidence that the proposed approach solves large instances with 89 countries and 442 cities in few seconds while matching the best known cost of 8 out of 36 instances and being less than 1.78% away from the best known solution for 27 instances.

Mathematics Subject Classification. 90-08, 90-B06, 90-B10, 90-C06, 90-C08, 90-C27, 90-C59.

Received March 1, 2021. Accepted September 18, 2021.

1. Introduction

Logistics in general and transportation in particular are the cornerstones of modern life. Their importance emanates from their multi-fold repercussions on the cost of goods, profit margins of transportation companies, clients’ service quality, drivers’ well being, and air pollution. In fact, they involve several parties: end clients, manufacturers, distributors, drivers, stock holders, etc. In addition, they require the scheduling of several inter-related tasks that are dynamic in nature and constrained in time and space. The economic and temporal constraints augment their complexity. Solving them requires the migration of tools from diverse disciplines including information technology, optimization, and vehicle routing.

Keywords. Generalized traveling salesman, beam search, symmetric traveling salesman, k-opt, Lin-Kernighan heuristic.

¹Logistics Department, Institut Superieur du Transport et de la Logistique de Sousse, University of Sousse, Sousse, Tunisia.
²Engineering Department, Kings College London, FNMES, Department of Engineering, Strand, London WC2R 2LS, UK.
*Corresponding author: rym.mhallah@kcl.ac.uk

© The authors. Published by EDP Sciences, ROADEF, SMAI 2021
Among the most widely studied transportation problems is the traveling salesman problem (TSP). A traveler has to visit a finite number of countries starting from one country and returning back to it, and visiting every country exactly once. The objective is to find a minimal cost route, where the cost can be total duration, travel distance, etc. TSP’s importance emanates from its occurrence as a subproblem of complex real life problems in the transport of passengers/goods and in scheduling. For these problems, TSP identifies a minimal-cost itinerary for each salesman. For example, TSP is a special case of the equality generalized symmetric traveling salesman problem (EGSTSP), where the salesman chooses exactly one of many cities of a country to visit; i.e., a TSP with a covering constraint.

Formally, consider a set of nodes that are divided into clusters. EGSTSP searches for the shortest route that visits exactly one node from every cluster starting and ending at the same cluster. EGSTSP is more difficult than TSP because of the combinatorial aspect added by the sizes of the clusters. EGSTSP occurs in several real-life applications such as maritime ship routing, distribution of medical supplies, urban waste management, telecommunication networks, logistics, rapid post dispatching, VLSI, circuit designs, and in laser cutting to determine the trajectory of a laser cutter [18, 21]. During the COVID-19 pandemic, EGSTSP has drawn a lot of attention. With reduced air-traffic and disrupted logistic chains, the procurement and dispatching of goods to confined customers and isolated cities has become a true challenge. In addition, the availability of a vaccine raises the issue of its fair distribution and of health care equity. Some of these vaccines impose a cold chain that can’t be broken. In such cases, optimizing the distribution plan is of prime importance. This optimization is equivalent to solving a large scale EGSTSP. Evidently, its exact solution may be challenging. However, the continuous advancement of the computing technologies provides near-optimal solutions to such difficult problems. They are allowing approximate methods to undertake a more extensive search; thus obtaining nearer-global optima in shorter times.

EGSTSP has been solved by exact approaches (such as dynamic programming, branch and bound, and branch and cut), and by approximate ones such as local improvement heuristics (\(k\)-opt, swap, insertion, etc.), and meta-heuristics (tabu search, ant colony, genetic algorithms, etc.). This paper proposes a new approximate hybrid approach for the EGSTSP. Hybrid heuristics have identified the best known solutions to several complex combinatorial optimization problems. They are powerful search methods because they tackle two competing goals: exploration and exploitation. Exploration is a diversification of the search. It investigates the solution space in order to determine the part that has a higher chance of containing the global optimum. Exploitation refines (or intensifies) the search on the part of the space that has a high potential of containing the global optimum.

The proposed hybrid heuristic is a beam search (BS) (i.e., a truncated branch and bound) that is augmented with improvement techniques. It ensures exploration via a standard width-first BS and exploitation via local search heuristics. BS strives for global optimization while local search heuristics strive for local optimization in the global optimum’s neighborhood. That is, BS can be assimilated to evolution while local search to learning. Generally, synchronization of evolution and learning yields efficient hybrid heuristics. Specifically, the proposed hybrid BS embeds

- a low-level hybridization, which addresses the functional composition of BS by subjecting the partial solution at each node to a local search; and
- a high level hybridization that maintains BS self containing by subjecting the incumbent of BS to a \(k\)-opt type of search.

To the best of the authors’ knowledge, this is the first application of BS to EGSTSP. In addition, the hybridization explores the success of local search to assess the nodes of the tree and to estimate their potential. It subsequently chooses the nodes with the best potential to branch on and prunes the non-promising ones; thus, it explores the search space’s parts that contain near-global optima while it discards the others. It then applies a 2-opt, a 3-opt or the notorious improved Lin-Kernighan (LK) heuristic [9] to its incumbent. The computational investigation provides computational evidence of the good performance of the hybridized BS within a reduced
runtime. Its deviation from the best known solution is less than 0.0578% for half of the instances and less than 1.78% for three quarters of them.

Section 2 defines the problem. Section 3 reviews the literature on EGSTSP. Section 4 details the proposed approaches. It presents the algorithm of a standard BS, its adaptation to EGSTSP, the low-level hybridized BS, and the high-level hybridized BS when applying each of the three local improvement methods: 2-Opt, 3-Opt and LK. Section 5 presents the computational results, which assess the efficiency of the methods in terms of solution quality and runtime and highlights the utility of the hybridizations. Finally, Section 6 summarizes the findings and gives potential research extensions.

2. Problem definition

EGSTSP is an NP-hard combinatorial optimization problem. It consists of finding the optimal path of a salesman who has to travel through a set of countries while visiting exactly one city from each country and visiting every country once. The optimal path minimizes the total traveled cost. Hence, the salesman must determine for each country the city s/he will visit and the order of visit of the countries. EGSTSP is more complex than TSP. For TSP, each country consists of a single town while EGSTSP has the additional complexity of choosing a city from each country. Because it extends TSP, which is NP-hard, EGSTSP is also NP-hard.

Herein, we define EGSTSP using the notation of \cite{6,7} and present their integer linear program (ILP). Formally, consider a complete non-oriented graph \( G = (N, E) \) where \( N = \{1, \ldots, n\} \) is a set of nodes that are divided into \( m \) mutually exclusive clusters \( C_h, \ h = 1, \ldots, m, \) and \( m \geq 3 \). \( E = \{[i, j] : i \in N, j \in N, i \neq j \} \) denotes the set of edges \( e \) connecting pairs \((i, j)\) of distinct nodes \( i \in N \) and \( j \in N, i \neq j \). The cost of traveling through edge \( e \in E \) is \( d_e \). This cost may be assimilated to a linear function of the Euclidean distance between \( i \) and \( j \).

The objective of EGSTSP is to determine a minimal cost cycle \( T \subseteq E \) such that \( T \) includes exactly one city from each cluster, and each cluster is visited once.

To define the ILP model of EGSTSP, we introduce the following notation. For a subset \( S \subseteq N \), \( E(S) := \{[i, j] \in E : i \in S, j \in S\} \) denotes the set of edges with both endnodes in \( S \) and \( \delta(S) := \{[i, j] \in E : i \in S, j \notin S\} \) the set of edges with exactly one end node in \( S \). For simplicity, we denote \( \delta\{v\} \) by \( \delta(v) \), for \( v \in N \).

ILP uses two types of binary variables: \( x_e = 1 \) if the salesman travels through edge \( e \in E \) and 0 otherwise, and \( y_v = 1 \) if the salesman visits node \( v \in N \) and 0 otherwise. Using the aforementioned notation and these two sets of binary variables, EGSTSP can be formulated as follows.

\[
\begin{align*}
\min z &= \sum_{e \in E} d_e x_e \quad & (2.1) \\
\text{s.t.} & \sum_{e \in \delta(v)} x_e \leq 2y_v & v \in N \quad (2.2) \\
& \sum_{v \in C_k} y_v = 1 & k = 1, \ldots, m \quad (2.3) \\
& \sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1) & S \subseteq N, 2 \leq |S| \leq n - 2, i \in S, j \in N \setminus S \quad (2.4) \\
x_e \in \{0, 1\} & e \in E \quad (2.5) \\
y_v \in \{0, 1\} & v \in N \quad (2.6)
\end{align*}
\]

The objective function, given by equation (2.1), minimises the total travel cost. Equation (2.2) preserves the flow through every node. A node is visited if it has both a predecessor and a successor node; therefore the righthand side is 2; otherwise, the righthand side must be zero. Equation (2.3) ensures that the tour includes exactly one city from each cluster. Equation (2.4) guarantees the connectivity of the solution: Each cut separating two visited nodes \( i \) and \( j \) must be crossed at least twice. Finally, equations (2.5) and (2.6) determine the nature of the decision variables.
Because EGSTSP is NP-hard, solving large instances of EGSTSP using ILP is difficult. Herein, we are interested in efficiently solving large instances of EGSTSP using heuristic methods, and in comparing the heuristics’ solutions to the ILP results that are readily available in the literature (and given by $z_{\text{lit}}$ in the computational section and in the Appendix).

3. Literature review

Small instances of the equality generalized TSP (EGTSP) were solved exactly using dynamic programming [26], branch and bound [14], and branch and cut [6]. Large instances have been tackled approximately; for example, Noon and Bean [20] applied the TSP’s closest neighborhood heuristic. Lien et al. [16] assimilated EGTSP to a TSP whose number of nodes is three times as large as the number of clusters. Dimitrijevic and Saric [5] developed an alternative transformation that had fewer nodes; i.e., using twice as many nodes as the number of clusters of the original EGTSP. Ben-Arieh et al. [2] opted for a transformation that had as many nodes as the number of clusters of EGTSP using the ‘exact’ Noon–Bean, and two modifications of the non-exact Fischetti–Salazar–Toth transformation. Helsgaun [9] transformed EGSTSP into a classic TSP and applied LK to the transformed TSP. Karapetyan and Gutin [11] proposed an LK heuristic for EGSTSP. Smith and Imeson [25] applied an iterative remove and insert heuristic for EGTSP. They opted for three insertion mechanisms: the furthest node, the cheapest, and random insertion. Karapetyan and Gutin [12] also designed a large neighborhood search for EGSTSP. Renaud et al. [23] proposed an Initialization, Insertion and Improvement heuristic that further generalized. Khachai and Neznakhina [13] developed a dynamic programming based heuristic for EGSTSP.

Another surge of the EGSTSP literature came from hybrid approaches. Ardalan et al. [1] hybridized the Imperialist Competitive Algorithm with a local search. Lawrence and Daskin [15] hybridized a random key genetic algorithm with a local search. Their algorithm is quite fast. It identifies its best solution within the first two or three iterations. Its good performance is due to the utility of the local search in identifying the best solution. However, their algorithm is outperformed by the mimetic algorithm of [8], who combined the advantages of genetic algorithms and local search. Chira et al. [4] designed a “sensible” ant colony system that makes the ants sensitive to the pheromone level in their trail; thus, explore the most promising regions of the search space. Yang et al. [28] augmented ant colony optimization to EGSTSP with a mutation mechanism and a local search. They showed the importance of the local search, in particular, for instances with fewer than 200 nodes. Bontoux et al. [3] proposed a mimetic algorithm whose crossover operator is based on a large neighborhood search.

Different variants of EGSTSP have appeared recently. Sundar and Rathinam [27] applied a branch and cut and [30] extended Christofide’s TSP algorithm to the multi-depot EGSTSP where there are several travelers; each departing from a different depot (node). Mestria [19] considered the clustered TSP, where all nodes of a cluster must be visited in a contiguous manner. The author hybridized a variable neighborhood random descent with local search (for intensification) and with a greedy randomized adaptive search (for diversification). This latter consists of a constructive heuristic and a perturbation method. The author applied several variable neighborhood structures, in a random order. Jiang et al. [10] proposed a hybrid genetic ant colony algorithm for the multiple TSP, where each salesman departs from a specific depot and returns to it. Yuan et al. [29] studied the generalized TSP with time windows, where arrival to a city must occur within a time window. They proposed two integer linear programs and valid inequalities that are separated dynamically within a branch-and-cut algorithm. They initiated their branch and bound from a feasible solution built via a simple heuristic. They solved instances with up to 30 clusters within a one-hour runtime. Salman et al. [24] imposed precedence constraints on EGSTSP, developed a new branching rule, and adapted some existing bounds to the problem.

This literature review suggests that EGSTSP was never tackled via BS. It further suggests that hybridization is a key factor in the success of most approaches to TSP related problems. To explore these findings, this paper proposes a hybrid BS that employs local search at each node and applies a $k$-opt type of search to the incumbent.
Proposed approaches

We efficiently solve EGSTSP using hybridized BS-based algorithms. BS is a truncated tree search. It avoids exhaustive enumeration by branching on a subset of elite nodes, believed to lead to the optimum. They usually have minimal fitness values, which are either the cost of their partial solutions or their upper bounds. At each iteration, \( \omega \) nodes are selected for branching, where \( \omega \) is the beam width. The other nodes are permanently discarded, and no backtracking is performed. We enhance the performance of BS by hybridizing it at two levels. The low-level hybridization adds a local search phase at each node of the BS tree. The high-level hybridization applies 2-opt, 3-opt or LK heuristics to the best solution that BS obtains. Section 4.1 describes a standard BS. Section 4.2 explains our adaptation of BS to EGSTSP. Sections 4.3 and 4.4 present the low- and high-level hybridization.

4.1. Standard beam search

The pseudo code of a standard BS is given in Figure 1. It consists of an initialization step, an iterative step and a stopping criterion. The initialization step declares the set \( \mathcal{N} \) of current nodes of the tree to the root node \( \mu_0 \) and the set \( \mathcal{M} \) of offspring nodes to the empty set. When an initial feasible solution \( \mathbf{x} \) is available, this step further sets the incumbent \( \mathbf{x}^* \) and its value \( z^* = z(\mathbf{x}^*) \) to, respectively, this initial solution \( \mathbf{x} \) and its objective function value. When an initial feasible solution is not available, the upper bound \( z^* \) is set to \( \infty \).

The iterative step chooses a node from \( \mathcal{N} \), and sets it as the current node. It branches out of the current node, and adds all new nodes to \( \mathcal{M} \) except for leaves. Leaves constitute feasible solutions; thus, are candidate solutions. A leaf becomes the incumbent whenever its cost is less than \( z^* \). The iterative step appends the \( \omega \)
Corresponding to \( s \) path of its parent, and appends a cluster to the end of its parent’s path. Specifically, branching out of the node that is, out of a node of level \( \ell \), \( \ell = 1, \ldots, m \), is a sequence of cities \( i^1, i^2, \ldots, i^\ell \) all belonging to \( N \) and to different clusters. As all tree-search techniques, BS has three major steps: branching, assessment, and selection.

The branching of a node of the tree corresponds to appending a cluster to the partial solution of that node. That is, out of a node of level \( \ell \) emanate \( m-\ell \) branches; each leading to a different cluster. A node inherits the path of its parent, and appends a cluster to the end of its parent’s path. Specifically, branching out of the node corresponding to \( s^\ell \) consists in appending a city from a non-visited cluster to \( s^\ell \).

The assessment of the cost of a newly created node \( s^\ell \) is based on a straightforward/simple lower bound and on an upper bound. The lower bound is the cost \( z_{s^\ell} \) of the partial solution \( s^\ell \). It is the sum of the travel costs between the successive nodes of \( s^\ell \):

\[
z_{s^\ell} = d_{i^1,i^2} + d_{i^2,i^3} + \ldots + d_{i^{\ell-2},i^{\ell-1}} + d_{i^{\ell-1},i^\ell}.
\]

It is the sum of its parent node’s cost \( z_{s^{\ell-1}} = d_{i^1,i^2} + d_{i^2,i^3} + \ldots + d_{i^{\ell-2},i^{\ell-1}} \) and of the travel cost \( d_{i^{\ell-1},i^\ell} \) from its parent node to the appended cluster. The upper bound is a total-cost of a complete solution constructed by iteratively appending the closest city of a ‘not yet assigned’ cluster to the partial solution \( s^\ell \).

At a given level \( \ell \) of the tree, the selection chooses the \( \omega \) best nodes among all generated child nodes for further branching at the next level \( \ell+1 \) of the tree. These iterative branching, evaluation and selection processes are repeated until \( \ell = m \); that is, until all clusters are visited. Herein, BS\(_0\) is started with a feasible solution \( \tilde{c} \) that minimizes the distance from \([1,\ldots,\ell]\) to \([h-1,\ldots,\ell]\); i.e.,

\[
d_{[h-1],v^*} + d_{v^*,[h+1]} = \min_{v \in C_h} \{d_{[h-1],v} + d_{v,[h+1]}\}.
\]

When applied to a node \( s^\ell \), the local search has \( O(\ell c) \) complexity (\( c = \bar{c} \)), where \( \bar{c} = \max_{h=1,\ldots,m} \{|C_h|\} \) is the maximum number of cities among all clusters. Because it is applied to all \( \sum_{\ell=3}^m \ell (m-\ell) \) nodes of the tree, the
local search increases the complexity of BS\(_1\) to at worst \(O(\omega m^2 c)\). Yet, it allows BS\(_1\) to attenuate the myopic nature of BS\(_0\); i.e., BS\(_0\) may miss the global optimum when it selects the \(\omega\) best nodes of a level and permanently prunes the others.

### 4.4. High-level hybridized beam search

The high-level hybridized BS, denoted BS\(.,\) = 2, 3, 4, applies a 2-Opt, a 3-Opt, or LK heuristic to the best solution obtained by BS\(_1\). Because the hybridization is high-level, the worst time complexity of BS\(.,\) = 2, 3, 4 is the sum of the complexity of BS\(_1\) and of the adopted hybridization approach.

The 2-Opt has an \(O(m^2)\) complexity where \(m\) is the number of clusters of the tour. It chooses two clusters of the tour randomly and reverses the flow between them. It is repeated as long as the solution is improved. For instance, consider a tour \([1, 2, \ldots, i - 1, i, i + 1, \ldots, j - 1, j, j + 1, \ldots, m, 1]\), where \([i]\) denotes the \([i]\)th cluster of the tour. When 2-Opt chooses randomly clusters \([i]\) and \([j]\), it generates the new solution \([1, 2, \ldots, i - 1, j, j - 1, \ldots, i + 1, [i], [j], [j + 1], \ldots, [m], 1]\). The 3-Opt has an \(O(m^3)\) complexity where \(m\) is the number of clusters of the tour. For a tour \([1, 2, \ldots, i - 1, [i], i + 1, \ldots, [j - 1, [j], j + 1, \ldots, [k - 1, [k], [k + 1, \ldots, [m], 1]\), 3-Opt chooses randomly three clusters \([i]\), \([j]\) and \([k]\) of the tour, and generates the new solution \([1, 2, \ldots, i, [k], [k - 1, \ldots, [j + 1, [i + 1], \ldots, [j], [k + 1, \ldots, [m], 1]\). It repeats this process as long as the solution is improved.

LK yields near-global optima when started from a large number of initial solutions. An LK perturbation of a solution causes, on average, increases of the order of 10 to 15% of its cost. It is one of the best heuristics for the symmetric TSP because of its adaptive nature. Indeed, it swaps a number of partial sequences of the tour. This solution is the sum of the complexity of BS\(_1\) and of the adopted hybridization approach.

4.4. High-level hybridized beam search

The high-level hybridized BS, denoted BS\(.,\) = 2, 3, 4, applies a 2-Opt, a 3-Opt, or LK heuristic to the best solution obtained by BS\(_1\). Because the hybridization is high-level, the worst time complexity of BS\(.,\) = 2, 3, 4 is the sum of the complexity of BS\(_1\) and of the adopted hybridization approach.

The 2-Opt has an \(O(m^2)\) complexity where \(m\) is the number of clusters of the tour. It chooses two clusters of the tour randomly and reverses the flow between them. It is repeated as long as the solution is improved. For instance, consider a tour \([1, 2, \ldots, i - 1, i, i + 1, \ldots, j - 1, j, j + 1, \ldots, m, 1]\), where \([i]\) denotes the \([i]\)th cluster of the tour. When 2-Opt chooses randomly clusters \([i]\) and \([j]\), it generates the new solution \([1, 2, \ldots, i - 1, [j], j - 1, \ldots, i + 1, [i], [j], [j + 1], \ldots, [m], 1]\). The 3-Opt has an \(O(m^3)\) complexity where \(m\) is the number of clusters of the tour. For a tour \([1, 2, \ldots, i - 1, [i], i + 1, \ldots, [j - 1, [j], j + 1, \ldots, [k - 1, [k], [k + 1, \ldots, [m], 1]\), 3-Opt chooses randomly three clusters \([i]\), \([j]\) and \([k]\) of the tour, and generates the new solution \([1, 2, \ldots, i, [k], [k - 1, \ldots, [j + 1, [i + 1], \ldots, [j], [k + 1, \ldots, [m], 1]\). It repeats this process as long as the solution is improved.

LK yields near-global optima when started from a large number of initial solutions. An LK perturbation of a solution causes, on average, increases of the order of 10 to 15% of its cost. It is one of the best heuristics for the symmetric TSP because of its adaptive nature. Indeed, it swaps a number of partial sequences of the tour. This number is not predetermined; yet, it offers a good tradeoff between solution quality and runtime. While 2-opt and 3-opt break 2 and 3 edges of the tour, LK chooses the number of edges to be broken such that this number yields a minimal cost tour. In this sense, LK may be perceived as a variable- and 3-opt break 2 and 3 edges of the tour, LK chooses the number of edges to be broken such that this number symmetric TSP because of its adaptive nature. Indeed, it swaps a number of partial sequences of the tour. This solution is the sum of the complexity of BS\(_1\) and of the adopted hybridization approach.

### 5. Computational results

The computational investigation assesses the performance of hybridization in general, and of its type, in particular, on the solution quality and on the runtime of BS. For this purpose, it uses five versions of BS:

- **BS\(_0\)** A standard width-first beam search of beam width \(\omega\);
- **BS\(_1\)** BS\(_0\) augmented with a local search at each node of the tree,
- **BS\(_2\)** BS\(_1\) with its best solution subject to a 2-opt,
- **BS\(_3\)** BS\(_1\) with its best solution subject to a 3-opt, and
- **BS\(_4\)** BS\(_1\) with its best solution subject to the LK heuristic with \(k\) up to 5.

It applies these five versions (coded in C and run on an Intel Core i3-4030U, 1.90 GHz, 4GB RAM) to 36 benchmark instances of EGSTSP, all available at [http://www.cs.rhul.ac.uk/home/zvero/GTSPLIB/](http://www.cs.rhul.ac.uk/home/zvero/GTSPLIB/). Let \(z^B\text{S}_i\) be the best known solution, and \(z^B\text{S}_i\), \(i = 0, 1, 2, 3, 4\) the corresponding BS\(_i\) solution value, for a beam width \(\omega = 1, 2, 3, 4, 5, 10\), obtained within runtime \(t^\omega\) (expressed in seconds). For this solution, the percent optimality gap \(\Delta^\omega = 100\% \frac{z^\omega - z^B\text{S}_i}{z^B\text{S}_i}\). Herein, we analyze the results, reported in A, focusing on the utility of the low- and high-level hybridization of BS. We then conclude with some useful remarks.
5.1. Utility of the low level hybridization

First, we compare the runtime and solution quality of BS$_0$ to that of BS$_1$; that is, of BS without and with local search at each node. (cf. Tabs. A.1 and A.2 for the detailed results.) We undertake this comparison to highlight the importance of the low-level hybridization undertaken at each node of each level $\ell$ of the search tree.

Figure 2, which displays the mean runtime of BS$_0, \ldots, BS_4$, suggests that the mean runtime of BS increases linearly as a function of the beam width $\omega$. Its average runtime (in seconds) can be estimated as a linear function of $\omega$: $\bar{t}_0 = 0.5454\omega - 0.0459$ and $\bar{t}_1 = 0.5473\omega + 0.0080$, with 99.03% and 99.98% respective coefficients of determination. This behavior is expected as a larger beam width requires more evaluations of partial solutions, of bounds, of sorting, stocking, and retrieving.

Figure 3, which displays box plots of the observed run times, further clarifies this tendency. Yet, it stipulates that the local search does not increase the run time. A statistical paired t-test infers that there is no difference between the mean run times of BS$_0$ and BS$_1$ at any level of significance while a paired statistical test infers that the mean $\Delta_{BS_1}$ is less than the mean $\Delta_{BS_0}$ at any level of significance and that the mean difference $\Delta_{BS_0} - \Delta_{BS_1}$ has a 4.84% point estimate and a 4.19% lower bound of a 95% confidence interval estimate. This difference is due to the local search, which enhances the search of BS, by investigating the neighborhood of the partial solution at each node. In fact, $\Delta_{BS_0} > \Delta_{BS_1}$ for all tested instances and for all beam widths. In addition, the average percent deviation $100\% \frac{\frac{1}{z^{lit}} - \frac{1}{z}}{\frac{1}{z}}$ is of the order of 26%; further highlighting the importance of the local search undertaken by BS$_1$ at every node. Because BS$_1$ is superior to BS$_0$ in terms of solution quality while being equally good in terms of runtime, it can be inferred that BS$_1$ is better than BS$_0$.

Figure 4 displays the box plots and means of the percent deviation of the solutions of BS$_0, \ldots, 4$, from $z^{lit}$. Zooming on the box plots and means of BS$_0$ and BS$_1$, we detect a seemingly counter-intuitive behavior for small $\omega$. Increasing $\omega$ from 1 to 4 does not decrease $\Delta_{BS_0}$ and $\Delta_{BS_1}$. This is most likely because it makes BS choose, at a level $\ell$, partial solutions that --despite their good quality at level $\ell$-- do not lead to near-optima. That is, the diversification brought up by the larger beam width focuses on areas of the search space that do not contain the global optimum. The local search undertaken at each node does not mitigate this glitch. On the other hand, increasing $\omega$ beyond 5 overcomes this issue. Setting $\omega = 10$ allows BS to obtain solutions that are closer to the global optimum. That is, it makes BS investigate areas of the search space that contain near-global
5.2. Utility of the high level hybridization

Second, we compare the performance of BS$_2$, BS$_3$, and BS$_4$. This comparison highlights the important impact of the high level hybridization, which requires a negligible additional runtime. (cf. Tabs. A.3–A.5 for the detailed results.)
As Figure 4 reveals, the improvements of the solution quality due the high-level hybridization are much larger than their counterparts due to the low-level hybridization, regardless of the beam width. These improvements occur at no additional runtime cost except for the last three instances when run with BS_4 and a beam width \( \omega = 10 \). These instances are marked as outliers in Figure 3, which displays the box plots and means of the observed run times of BS_0–BS_4. For all beam widths, the mean run time of any of the approaches is larger than its median; signaling the existence of outlier cases, corresponding to the last three instances. Despite the presence of these outliers, which drive the run time of BS_4 up for \( \omega = 10 \), paired t-tests infer that there is no statistical evidence to claim that the mean run time of any pair of hybridized versions of BS are different at a 5% significance level.

The lack of exploitation and of exploration of the search space makes BS_0 obtain better results for larger beam widths. This behavior persists for BS_1, which benefits from a local search at each of its nodes, and for BS_2, which benefits from an intensified 2-opt search around its best solution. However, for BS_3 and BS_4, the 3-opt and the LK intensification makes BS obtain its best solutions using a beam width \( \omega = 3 \), with a mean runtime less than 2 seconds. This is confirmed by Figures 2 and 5, which display respectively the mean percent deviation from \( z^{\text{lit}} \) and mean runtime as a function of beam width for BS_0 to BS_4.

5.3. Remarks

LK is known to obtain good results when initialized from several random initial solutions. The proposed approach BS_4 provides evidence that it is possible to generate initial solutions for LK in a more systematic manner. Furthermore, the results infer that BS_3 with a beam width \( \omega = 3 \) yields, on average, better results than the other considered beam searches. However, it remains true that the incumbent of BS_1 can be subjected to three types of searches 2-opt, 3-opt, and LK, at a negligible additional runtime. In fact, there is no statistical difference between the runtime of BS_1 and BS_1, \( \cdot = 2, 3, 4 \); implying that the bulk of their runtime is caused by the BS component. Finally, even though \( \omega = 3 \) yields in general the best performance, running BS_1 with different beam widths constitutes a good diversification strategy. Using these two additional aforementioned intensification and diversification mechanisms reduces the percent deviation gaps of the BS solution to those observed in the literature; matching the best solution in 22.22% of the instances, and averaging a 0.01344%
Figure 6. Percent deviation of BS solutions from best known ones.

Figure 7. Box plot of percent deviation of BS solutions from best known ones.

deviation. The mean should be interpreted with care as it is affected by two outlier values, recorded for instances 40kroA200 and 80rd400, as shown in Figure 6. These outliers are clearly depicted in Figure 7, which displays the resulting box plot of percent deviations for this BS. The corresponding five-point summary of the percent deviation is (Minimum = 0, Q1 = 0.00063, Q2 = 0.00578, Q3 = 0.01779, Maximum = 0.07027), where Q1, Q2 and Q3 are the first, second and third quartiles. Ignoring the two outlier instances brings the largest deviation over the other 34 instances to 0.03925% and its average to 0.01020%.
6. Conclusion

This paper addressed EGSTSP via a beam search that obtains good solutions for large beam widths. However, to avoid the exponential increase of runtime associated with branch and bound, we opted for both a low- and a high-level hybridization of the beam search. First, we performed a local search at each node of the tree. This local search acts as a lookahead strategy. It allows the beam search to retain the partial solutions that could lead to near-global optima in lieu of selecting the lowest cost partial solutions. This local search improved the performance of the beam search without affecting its runtime. Second, we subjected the best solution of the beam search to each of three local search operators: 2-Opt, 3-Opt and Lin-Kernighan. This high level hybridization further improved the solution quality of the standard beam search by up to 70% without affecting its runtime. Applying the three search operators to the incumbent offers BS more exploration and exploitation power. The proposed hybridization can be applied to different variants of traveling related problems including vehicle routing, dial-a-ride, and delivery with time windows. Other types of search techniques can also be considered such as simulated annealing, variable neighborhood search, adaptive, and data-driven techniques.

Appendix A. Detailed computational results

The results of BS, ω = 0, . . . , 4 are reported in Tables A.1–A.5. The first column indicates the ‘.gts’ label of the instance whereas the second column reports its best known solution \( z_{\text{lit}} \), available in the literature. The next six triplets of columns report the BS solution value \( z_{\text{BS}} \), its percent optimality gap \( \Delta_{\omega} = 100\% \frac{z_{\omega} - z_{\text{lit}}}{z_{\text{lit}}} \), and its runtime \( t_{\omega} \) in seconds when the beam width \( \omega = 1, 2, 3, 4, 5, 10 \).
| Instance | Median | Maximum |
|----------|--------|---------|
| 1        | 267    | 320     |
| 2        | 304    | 336     |
| 3        | 326    | 352     |
| 4        | 347    | 375     |
| 5        | 368    | 397     |
| 6        | 382    | 406     |
| 7        | 401    | 427     |
| 8        | 421    | 447     |
| 9        | 438    | 465     |
| 10       | 450    | 478     |
| 11       | 467    | 497     |
| 12       | 488    | 521     |
| 13       | 505    | 538     |
| 14       | 529    | 566     |
| 15       | 553    | 593     |
| 16       | 574    | 615     |
| 17       | 602    | 644     |
| 18       | 629    | 673     |
| 19       | 651    | 697     |
| 20       | 673    | 721     |
| 21       | 701    | 750     |
| 22       | 729    | 782     |
| 23       | 758    | 815     |
| 24       | 788    | 849     |
| 25       | 819    | 883     |

**Table A.1.** BS Results.
| Instance | \(z\) | \(\Delta_1\) | \(\Delta_2\) | \(\Delta_3\) | \(\Delta_4\) | \(t\) |
|----------|----------|----------|----------|----------|----------|------|
| 1         | 11.10     | 2.71     | 0.0000  | 1.0000   | 0.0000   | 1.0000|
| 2         | 12.46     | 1.10     | 0.0000  | 1.0000   | 0.0000   | 1.0000|
| 3         | 13.42     | 1.10     | 0.0000  | 1.0000   | 0.0000   | 1.0000|
| 4         | 11.10     | 2.71     | 0.0000  | 1.0000   | 0.0000   | 1.0000|
| 5         | 12.46     | 1.10     | 0.0000  | 1.0000   | 0.0000   | 1.0000|
| 6         | 13.42     | 1.10     | 0.0000  | 1.0000   | 0.0000   | 1.0000|

**Table A.2. BS Results**
### Table A.3. BS Results

| Instance | \( \Delta^1 \) | \( \Delta^2 \) | \( \Delta^3 \) | \( \Delta^4 \) | \( \Delta^5 \) | Average | Maximum |
|----------|----------------|----------------|----------------|----------------|----------------|---------|---------|
| 11st70   | 146            | 57             | 102            | 107            | 112            | 107.2   | 112     |
| 10kroA   | 84             | 34             | 52             | 57             | 62             | 52.2    | 62.0    |
| 20kroB   | 123            | 68             | 85             | 90             | 95             | 85.6    | 95.0    |
| 21lin    | 112            | 57             | 72             | 77             | 82             | 72.5    | 82.0    |
| 26bier   | 127            | 72             | 83             | 88             | 93             | 83.5    | 93.0    |
| 29pr     | 144            | 68             | 85             | 90             | 95             | 85.6    | 95.0    |
| 30kroA   | 95             | 46             | 62             | 67             | 72             | 62.5    | 72.0    |
| 30kroB   | 123            | 68             | 85             | 90             | 95             | 85.6    | 95.0    |
| 31pr     | 152            | 72             | 83             | 88             | 93             | 83.5    | 93.0    |
| 40d      | 198            | 102            | 107            | 112            | 117            | 107.5   | 117     |
| 40kroA   | 135            | 68             | 85             | 90             | 95             | 85.6    | 95.0    |
| 40kroB   | 135            | 68             | 85             | 90             | 95             | 85.6    | 95.0    |

**Summary**

Minimum: 51.12
Average: 85.52
Maximum: 95.00
| Instance | $\Delta t_1$ | $\Delta t_2$ | $\Delta t_3$ | $\Delta t_4$ | $\Delta t_5$ | $\Delta t_6$ | $\Delta t_7$ | $\Delta t_8$ | $\Delta t_9$ | $\Delta t_{10}$ |
|----------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 1st670   | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         |
| 1st870   | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         |
| 1st1070  | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         |
| 1st1270  | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         |
| 1st1470  | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         | 3.02         |

**Table A.4. BS Results.**
| Instance       | $\omega = 1$ | $\omega = 2$ | $\omega = 3$ | $\omega = 4$ | $\omega = 5$ | $\omega = 10$ |
|---------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 11el151       | 174          | 175          | 175          | 174          | 174          | 179          |
| 14st70        | 316          | 317          | 317          | 316          | 317          | 317          |
| 16el76        | 209          | 211          | 214          | 209          | 214          | 209          |
| 16pr76        | 64,925       | 66,783       | 67,082       | 64,994       | 65,646       | 67,281       |
| 20kra100      | 9711         | 9712         | 9712         | 9712         | 9712         | 9712         |
| 20kraB100     | 10,328       | 10,328       | 10,352       | 10,376       | 10,542       | 10,376       |
| 20kroC100     | 9554         | 10,189       | 10,449       | 9,986        | 9,554        | 9,660        |
| 20kroD100     | 9450         | 9462         | 9462         | 9450         | 9462         | 9450         |
| 20kroE100     | 9523         | 9770         | 9776         | 9750         | 9766         | 9750         |
| 20at90        | 497          | 517          | 529          | 506          | 506          | 506          |
| 20rd100       | 3650         | 3825         | 3926         | 3752         | 3873         | 3824         |
| 21el110       | 249          | 271          | 255          | 257          | 262          | 255          |
| 21lin105      | 8213         | 8295         | 8357         | 8224         | 8252         | 8252         |
| 22pr107       | 27,899       | 28,050       | 28,053       | 28,053       | 28,053       | 28,053       |
| 25pr124       | 36,305       | 38,295       | 37,462       | 37,462       | 37,542       | 37,542       |
| 26bier127     | 72,418       | 73,442       | 73,442       | 73,322       | 73,884       | 74,063       |
| 28pr136       | 42,526       | 45,028       | 43,262       | 43,262       | 43,526       | 43,526       |
| 29pr144       | 45,886       | 45,988       | 47,525       | 46,005       | 47,525       | 47,402       |
| 30kraA150     | 11,018       | 11,082       | 11,082       | 11,082       | 11,082       | 11,082       |
| 30kraB150     | 12,196       | 12,680       | 13,001       | 12,672       | 12,672       | 12,672       |
| 31pr152       | 51,576       | 53,350       | 52,700       | 52,907       | 52,907       | 52,830       |
| 32u159        | 22,664       | 23,036       | 23,017       | 23,015       | 23,360       | 23,442       |
| 39at195       | 854          | 887          | 933          | 887          | 941          | 944          |
| 40d198        | 10,557       | 10,771       | 10,664       | 10,674       | 10,674       | 10,619       |
| 40kroA200     | 13,406       | 13,722       | 13,650       | 13,698       | 13,698       | 13,379       |
| 40kroB200     | 13,113       | 13,722       | 13,650       | 13,698       | 13,698       | 13,379       |
| 45xs225       | 68,340       | 70,467       | 69,693       | 68,817       | 69,965       | 70,143       |
| 46pr226       | 64,007       | 64,139       | 64,007       | 64,007       | 64,007       | 64,007       |
| 53gil263      | 10,676       | 10,653       | 10,650       | 10,639       | 10,639       | 10,566       |
| 53pr264       | 29,549       | 30,131       | 30,056       | 30,135       | 30,135       | 29,983       |
| 60pr299       | 22,615       | 22,992       | 23,700       | 23,091       | 23,412       | 23,412       |
| 64lin318      | 20,765       | 21,842       | 21,618       | 21,905       | 21,905       | 21,897       |
| 80rd400       | 63,061       | 67,922       | 68,285       | 67,868       | 67,868       | 67,870       |
| 88pr459       | 60,099       | 62,942       | 61,518       | 62,751       | 62,751       | 62,751       |
| 89pc442       | 21,657       | 23,020       | 23,168       | 22,507       | 22,507       | 22,507       |

**Summary**

- **Average**
  - $\omega = 1$: 3.03 ± 0.784
  - $\omega = 2$: 3.26 ± 1.636
  - $\omega = 3$: 2.20 ± 2.006
  - $\omega = 4$: 2.83 ± 3.106
  - $\omega = 5$: 3.07 ± 3.388
  - $\omega = 10$: 3.18 ± 3.299
- **Minimum**
  - $\omega = 1$: 0.00 ± 0.000
  - $\omega = 2$: 0.00 ± 0.000
  - $\omega = 3$: 0.00 ± 0.000
  - $\omega = 4$: 0.00 ± 0.000
  - $\omega = 5$: 0.00 ± 0.000
  - $\omega = 10$: 0.00 ± 0.000
- **Maximum**
  - $\omega = 1$: 8.84 ± 5.703
  - $\omega = 2$: 9.00 ± 13.422
  - $\omega = 3$: 7.05 ± 15.312
  - $\omega = 4$: 10.00 ± 24.915
  - $\omega = 5$: 10.19 ± 28.954
  - $\omega = 10$: 11.06 ± 81.625
- **Median**
  - $\omega = 1$: 2.73 ± 0.110
  - $\omega = 2$: 3.28 ± 0.211
  - $\omega = 3$: 1.90 ± 0.268
  - $\omega = 4$: 2.05 ± 0.344
  - $\omega = 5$: 2.62 ± 0.460
  - $\omega = 10$: 2.45 ± 0.87
References

[1] A. Ardalan, S. Karimi, O. Poursabzi and B. Naderi, A novel imperialist competitive algorithm for generalized traveling salesman problems. Appl. Soft Comput. 26 (2015) 546–555.

[2] D. Ben-Ariei, G. Gutin, M.Penn, A. Yeo and A. Zverovitch, Transformations of generalized ATSP into ATSP. Oper. Res. Lett. 31 (2003) 357–365.

[3] B. Bontoux, C. Artiges and D. Feillet, A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. Comput. Oper. Res. 37 (2010) 1844–1852.

[4] C. Chira, C.M. Pintea and D. Dumitrescu, Sensitive ant systems in combinatorial optimization. Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques, KEPT, Cluj-Napoca, Romania (2007) 185–192.

[5] V. Dimitrijevic and Z. Saric, An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem. Geogr. Inf. Sci. 102 (1997) 105–110.

[6] M. Fischetti, J.J. Salazar-González and P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper. Res. 45 (1997) 378–394.

[7] M. Fischetti, J.J. Salazar-González and P. Toth, The generalized traveling salesman and orienteering problems. In The traveling salesman problem and its variations, edited by G. Gutin, A.P. Punnen. Vol 12 of Combinatorial Optimization. Springer, Boston, MA (2007).

[8] G. Gutin, D. Karapetyan and N. Krasnogor, Memetic algorithm for the generalized asymmetric traveling salesman problem. Stud. Comput. Intell. 129 (2008) 199–210.

[9] K. Helsgaun, Solving the equality generalized traveling salesman problem using the Lin-Kernighan-Helsgaun-algorithm. Math. Program. Comput. 17 (2015) 1–19.

[10] C. Jiang, Z. Wan and Z. Peng, A new efficient hybrid algorithm for large scale multiple traveling salesman problems. Expert Syst. Appl. 139 (2020) 112867.

[11] D. Karapetyan and G. Gutin, Lin-Kernighan heuristic adaptations for the generalized traveling salesman problem. Eur. J. Oper. Res. 208 (2011) 221–232.

[12] D. Karapetyan and G. Gutin, Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. Eur. J. Oper. Res. 219 (2012) 234–251.

[13] M.Y. Khachai and E.D. Neznakhina, Approximation schemes for the generalized traveling salesman problem. Proc. Steklov Inst. Math. 299 (2017) 97–105.

[14] G. Laporte, H. Mercure and Y. Nobert, Generalized traveling salesman problem through n sets of nodes: The asymmetric case. Discrete Appl. Math. 18 (1987) 185–197.

[15] V.S. Lawrence and M.S. Daskin, A random-key genetic algorithm for the generalized asymmetric traveling salesman problem. Eur. J. Oper. Res. 174 (2006) 38–53.

[16] Y. Lien, E. Ma and B.W.S. Wah, Transformation of the generalized traveling salesman problem into the standard traveling salesman problem. Inf. Sci. 74 (1993) 177–189.

[17] T. Makarovskikh, A. Panyukov and E. Savitsky, Software development for cutting tool routing problems. Procedia Manuf. 29 (2019) 567–574.

[18] J. Renaud and F.F. Boctor, An efficient composite heuristic for the symmetric generalized travelling salesman problem. Eur. J. Oper. Res. 108 (1998) 571–584.

[19] J. Renaud, F.F. Boctor and G. Laporte, A fast composite heuristic for the symmetric traveling salesman problem. INFORMS J. Comput. 8 (1996) 134–143.

[20] R. Salman, F. Ekstedt and P. Damaschke, Branch-and-bound for the precedence constrained generalized traveling salesman problem. Oper. Res. Lett. 48 (2020) 163–166.

[21] S.L. Smith and F. Imeson, GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. Comput. Oper. Res. 108 (2019) 2320–2325.

[22] J. Renaud and F.F. Boctor, An efficient composite heuristic for the symmetric generalized travelling salesman problem. Eur. J. Oper. Res. 108 (1998) 571–584.

[23] R. Salman, F. Ekstedt and P. Damaschke, Branch-and-bound for the precedence constrained generalized traveling salesman problem. Oper. Res. Lett. 48 (2020) 163–166.

[24] S.L. Smith and F. Imeson, GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. Comput. Oper. Res. 87 (2017) 1–19.

[25] S.S. Srivastava, S. Kumar, R. Garg and P. Sen, Generalized traveling salesman problem through n sets of nodes. CORS J. 7 (1969) 97–101.

[26] K. Sundar and S. Rathinam, Generalized multiple depot traveling salesmen problem-Polyhedral study and exact algorithm. Comput. Oper. Res. 70 (2016) 39–55.

[27] J. Yang, X. Shi, M. Marchese and Y. Liang, An ant colony optimization method for generalized TSP problem. Prog. Nat. Sci. 18 (2008) 1417–1422.
[29] Y. Yuan, D. Cattaruzza, M. Ogier and F. Semet, A branch-and-cut algorithm for the generalized traveling salesman problem with time windows. Oper. Res. Lett. 286 (2020) 849–866.

[30] X. Zhou and B. Rodrigues, An extension of the Christofides heuristic for the generalized multiple depot multiple traveling salesmen problem. Oper. Res. Lett. 257 (2017) 735–745.