Branch-and-Bound Algorithm for the Combinatorial Routing Problems

Jindřiška Deckerová
Tuesday - 16:15, parallel 103
Open Informatics
deckejin@fel.cvut.cz

ABSTRACT

My semestral project is focused on a generalization of the Traveling Salesman Problem called the Close-Enough TSP and the computation of lower bounds of this combinatorial problem by Branch-and-Bound algorithm proposed by Coutinho et al. in [1]. The project includes analysis of lower bounds obtained by the BnB algorithm and an extension to the Generalized Traveling Salesman Problem with Neighborhoods.

I. ASSIGNMENT

A. Problem Statement

The most popular and widely studied combinatorial problem is the Traveling Salesman Problem (TSP). The formulation of the TSP is following: Let us have a set of locations; the goal is to visit all of them in order to collect data from the locations, i.e., having a complete, undirected graph $G = (V, E)$, the goal is to determine the shortest Hamiltonian cycle.

In some cases, visiting the exact location is not necessary, thus the area around the location can be described as a region, and it is sufficient to visit the region in order to collect the data from the location. The generalization of the TSP is motivated by robotic manipulators and it is called the Traveling Salesman Problem with Neighborhoods (TSPN). A variant of the TSPN with a disk-shaped neighborhood motivated by collecting data using wireless communication is called the Close-Enough TSP (CETSP).

The goal of the CETSP is to determine the shortest path visiting all continuous disk-shaped neighborhoods with prespecified sensing radius. The problem formulation of the CETSP in this project is based on the formulation from [2]. Having a set of n locations to be visited, $S = \{s_1, \cdots, s_n\}$, where $s_i = (s_x^i, s_y^i) \in \mathbb{R}^2$, and the cost of travel between two locations s_i and s_j is the Euclidean distance, $\|(s_i, s_j)\| = \sqrt{(s_x^i - s_x^j)^2 + (s_y^i - s_y^j)^2}$. The goal of the TSP is to determine the shortest path visiting all locations, starting and ending at the same location s_1 (referred as a depot). In the CETSP, a location is considered visited if the path is in the distance less or equal to the sensing radius δ, a disk-shaped radius is formed around each location, except the depot. Each location s_i can have arbitrary radius δ_i. Then the goal of the CETSP is to determine a sequence of visits together with the most suitable point p_i such that each s_i has its point p_i and it holds $\|(s_i, p_i)\| = \delta_i$. The sequence of visits of locations can be described as a permutation of locations $\Sigma = (\sigma_1, \cdots, \sigma_n)$, where $\sigma_i \in [1, n]$ and $\sigma_i \neq \sigma_j$, $i \neq j$. The depot location visited first in the sequence Σ is denoted as $s_{\sigma_1} = s_1$, i.e., $\sigma_1 = 1$. So the location is visited at the σ_i time if a point p_i is determined. The CETSP can be formulated as a combinatorial optimization task:

$$\begin{aligned} \min_{\Sigma,P} \quad & L(\Sigma, P, S) = \sum_{i=1}^{n-1} \left\|(p_{\sigma_i}, p_{\sigma_{i+1}})\right\| + \|(p_{\sigma_n}, p_{\sigma_1})\| \\ & \text{subject to } \Sigma = (\sigma_1, \cdots, \sigma_n), 1 \leq \sigma_i \leq n, \quad \sigma_i \neq \sigma_j \text{ for } i \neq j \\ & S = \{s_1, \cdots, s_n\} \\ & P = \{p_1, \cdots, p_n\} \\ & s_{\sigma_1} = s_1 \quad \text{and} \quad \|(s_i, p_i)\| \leq \delta_i \quad \text{for } i \in [1, n] \end{aligned}$$

The continuous neighborhood of locations in the TSPN can be discretized, and the problem becomes a variant of the Generalized Traveling Salesman Problem (GTSP). Another generalization of the TSP referred to as Generalized Traveling Salesman Problem with Neighborhoods (GTSPN) is introduced in [7]. This generalization is motivated by tasks for redundant robotic manipulators or multi-goal aircraft missions. The main challenge of this problem is the extension of the TSPN with 3D regions and each neighborhood represented by a set of regions and forming a neighborhood set.

The formulation of the problem in this project is similar to the description proposed in [3] to remain readability and consistency of the problem due to its complex formulation. The problem formulation is following: Having a set of neighborhoods represented as neighborhood sets, i.e., each neighborhood consists of multiple individual regions, and the goal is to determine the shortest path visiting all neighborhood sets. Let $\mathcal{S}$ be a set of n neighborhood sets $\mathcal{S} = \{S_1, \cdots, S_n\}$, where each set $S_i \in \mathcal{S}$ consists of m_i regions $S_i = \{Q_{i,1}, \cdots, Q_{i,m_i}\}$, each region $Q_{i,k} \subset \mathbb{R}^3$, $i \in [1, n]$ and $k \in [1, m_i]$, is one of three types: polyhedron, ellipsoid or combination of both. The goal is to determine a sequence of visits $\Sigma = (\sigma_1, \cdots, \sigma_n)$,

where $\sigma_i \in [1, n]$ and the point of the visit $p_i \in \mathbb{R}^3$ that is inside the region $Q_{i,k} \in S_i$, i.e., $\bigcup_{k=1}^{m_i}(\{p_i\} \cap Q_{i,k}) \neq \emptyset$. Then the GTSPN can be formulated as follows:

$$\begin{aligned} \min_{\Sigma,P} \quad & L(\Sigma, P, S) = \sum_{i=1}^{n-1} \left\|(p_{\sigma_i}, p_{\sigma_{i+1}})\right\| + \|(p_{\sigma_n}, p_{\sigma_1})\| \\ & \text{subject to } \Sigma = (\sigma_1, \cdots, \sigma_n), 1 \leq \sigma_i \leq n, \quad \sigma_i \neq \sigma_j \text{ for } i \neq j \\ & P = \{p_1, \cdots, p_n\}, \bigcup_{k=1}^{m_i}(\{p_i\} \cap Q_{i,k}) \neq \emptyset \end{aligned}$$

B. Problem Categorization

The TSP is known to be NP-hard problem. In [4] it is proven that the TSPN is APX-hard and cannot be approximated within a factor of 391/390 in polynomial time, unless P=NP.

II. Related works

This section presents brief outline of approaches for solving the CETSP and the GTSPN and an outline of existing approaches for computing the lower bounds of the CETSP.

A. Approaches for solving the CETSP and the GTSPN

The CETSP has been first introduced by Gulczynski et al. [5] together with several heuristic approaches, all consisting of three steps. In the first step, a set of feasible supernodes (points within the radius of location) is created. The next step is to find a feasible tour visiting all supernodes. The third step is called an economization routine; the tour found in step two is optimized to reduce the length of the tour while remaining feasibility. The heuristics differ in the first step; different methods are used for determination of the supernodes set, e.g., hexagonal tiling, binary matrix representation, or Steiner zones.

Another approach for solving the CETSP is based on the unsupervised learning procedure of the Growing Self-Organizing Array (GSOA) from [2]. The GSOA is motivated by the Self-Organizing Maps (SOMs) based solution of multi-goal path planning problems. The key element of the heuristic is having a representation of locations as a set of nodes organized into a one-dimensional structure called ring. The nodes are associated with the particular locations to be visited and a point called waypoint that represents the representative point within the radius δ for the location. The learning procedure starting with the depot location and iteratively determines for each location its node and its waypoint. The ring is then adapted towards the waypoints. The algorithm ends when the solution is no longer improving. The GSOA has been deployed in a solution of the CETSP in 3D [6].

The GTSPN was first introduced by Vicencio et al. in [7] and solved by a genetic algorithm called the Hybrid Random-Key Genetic Algorithm (HRKGA) proposed in [7]. The neighborhood sets are encoded into chromosomes, and an initial population of chromosomes is created. Then selection and crossover operations are repeatedly applied to the generation of the chromosomes together with mutation of chromosomes. The Lin-Kernighan heuristic is used to improve the tour.

In [3], the GSOA with a fast post optimization has been deployed in the solution of the GTSPN. Besides the post optimized GSOA, a new heuristic based on the decoupled approach using a solution of the GTSP is proposed, and it is further improved by post-processing procedure.

B. Approaches for computing lower bounds for the CETSP

The lower and upper bounds on the solution of a problem give us information about instances and the optimal solution, and we can measure the performance of algorithms by computing a gap between a solution of a tight lower bound.

A discretization scheme is proposed to compute lower bounds and the upper bounds of the CETSP in [8]. Each disk-shaped neighborhood is discretized using a finite set of discretization points. The disk is divided into arcs, and each point on the arc is associated with a point inside the region - the discretization point. The problem is then solved as the GTSP on the discretization points. The set of points is reduced by selected only points with the maximum discretization error measured as a distance between a discretization point and the point on the arc. When the graph is reduced, the GTSP is solved as a Mixed Integer Problem (MIP). This gives an upper bound; the lower bound is computed from the upper bound by removing the discretization error for each discretization point included in the tour.

For solving the lower bound of the CETSP, the Branch-and-Bound (BnB) algorithm is introduced in [1]. The BnB is described in Section III.

III. Problem solution

A. Design

The Branch-and-Bound (BnB) is an algorithm for solving combinatorial problems, such as scheduling or the TSP. In general, the goal is to find a solution to the given problem in a set of possible solutions called a solution space, a solution in the solution space is called the partial solution. The BnB consists of two phases: *branching* and *bounding*. During the *branching*, the solution space is recursively split into smaller spaces, and the problem is solved in the smaller solution space. In the *bounding*,

the solution space is pruned by eliminating the partial solutions that will not contain the optimal solution, e.g., in the scheduling the cost of the removed partial solution is already larger that deadlines of remaining tasks, or in the TSP, the length of the removed partial solution is larger than the given upper bound on the total problem length.

The BnB for the CETSP has been introduced in [1], where the general algorithm is modified to fit the CETSP. The solution space is represented as a tree, and each node of the tree is associated with the partial solution, i.e., a partial tour - a sequence of locations in particular order, and a set of yet not covered locations. The BnB algorithm starts by creating a root node. The root node consists of three locations, the depot location S_1 and the last location S_n are always included in the root. The third location is selected by solving the Second Order Cone Programming (SOCP) problem of the depot, the last location, and each remaining location. The location associated with the maximum length of the problem is selected for the root node. The SOCP solves GTSPN for the given sequence and associates waypoints with the locations in the sequence. If the partial solution of the root node is feasible, i.e., the partial tour covers every neighborhood of every location, then the optimal solution is found, and the algorithm is terminated. Otherwise, the branching phase starts. The solution space is split into smaller solution spaces. A new solution space is created by adding a location from the set of uncovered locations to the partial solution. Then the SOCP problem is solved for the new partial solution. In the bounding phase, the solution space is pruned by eliminating the nodes having the total length of the partial tour is larger than the given upper bound.

B. Implementation

The SOCP problem for the CETSP is formulated according to [1] as follows:

$$\text{minimize} \quad \sum_{i=0}^{n} d_i \tag{1}$$

$$\begin{aligned}
\text{subject to} \quad & w_i = p_x^{\sigma_{i-1}} - p_x^{\sigma_i} \quad \forall i \in [2,n] && (2)\\
& v_i = p_y^{\sigma_{i-1}} - p_y^{\sigma_i} \quad \forall i \in [2,n] && (3)\\
& u_i = p_z^{\sigma_{i-1}} - p_z^{\sigma_i} \quad \forall i \in [2,n] && (4)\\
& d_i^2 \geq w_i^2 + v_i^2 + u_i^2 \quad \forall i \in [1,n] && (5)\\
& x_i = s_x^i - p_x^i \quad \forall i \in [1,n] && (6)\\
& y_i = s_y^i - p_y^i \quad \forall i \in [1,n] && (7)\\
& z_i = s_z^i - p_z^i \quad \forall i \in [1,n] && (8)\\
& x_i^2 + y_i^2 + z_i^2 \leq \delta_i^2 \quad \forall i \in 1,n] && (9)\\
& d_i \geq 0 \quad \forall i \in [1,n] && (10)
\end{aligned}$$

The goal is to select a waypoint p_i for each location s_i that the waypoint is within the radius δ_i. Then the sequence of waypoints is arranged so the overall distance is minimized.

For the extension to the GTSPN, the Eq. (1) - (5), (10) stay the same and the formulation of Eq. (6) - (9) has been modified to fit the generalized problem with various-shaped neighborhoods.

Having three different region types: polyhedron, ellipsoid, and hybrid, the constraints to ensure the waypoint p_i is within the region $Q_{i,k}$ will vary according to the region type. The region types are well described in [3] as follows: A region $Q_{i,k}$ of the ellipsoid type is defined by the region center $c_{i,k}$ and the symmetric positive definite matrix $\mathbf{P}_{i,k} \in \mathbb{R}^{3,3}$:

$$(p_i - c_{i,k})^T \mathbf{P}_{i,k}^{-1} (p_i - c_{i,k}) \leq 1. \tag{11}$$

The polyhedron is given by 12 half planes and the region $Q_{i,k}$ of the polyhedron type is given by the matrix $\mathbf{A}_{i,k} \in \mathbb{R}^{12,3}$ and the vector $\mathbf{b}_{i,k} \in \mathbb{R}^{12,1}$:

$$\mathbf{A}_{i,k} \cdot p_i - \mathbf{b}_{i,k} \leq 0. \tag{12}$$

The hybrid type of the region is a combination of an ellipsoid given by its center $c_{i,k}$ and the matrix $\mathbf{P}_{i,k} \in \mathbb{R}^{3,3}$; and six half planes given by the matrix $\mathbf{A}_{i,k} \in \mathbb{R}^{6,3}$ and the vector $\mathbf{b}_{i,k} \in \mathbb{R}^{6,1}$. In [3] is assumed that all regions are convex and its center $c_{i,k}$ is inside the region.

In order to visit the neighborhood set S_i, one of the m neighborhoods $Q_{i,k}$, $k \in [1,m]$, has to be visited. In the SOCP model, we use the Big-M method to select only one of the m neighborhoods for the representation of the neighborhood set S_i. For example, having two regions in the neighborhood set S_i of the polyhedron type and the ellipsoid type, the constraints are following:

$$(p_i - c_{i,1})^T \mathbf{P}_{i,1}^{-1} (p_i - c_{i,1}) \leq 1 + M \cdot b_1 \tag{13}$$

$$\mathbf{A}_{i,2} \cdot p_i - \mathbf{b}_{i,2} \leq M \cdot b_2 \tag{14}$$

$$\sum_{i=1}^{m=2} b_i = m - 1 \tag{15}$$

The implementation of the BnB based on [9] is simplified and extended for the GTSPN.

IV. Experiments

A. Benchmark Settings

The proposed algorithm has been empirically evaluated on 3D instances of the GTSPN proposed in [7]. The instances has been modified to reduce the running time per each instance. In [7], the authors proposed overall 30 3D instances, each instance consists of n neighborhood sets, $n \in \{30, 35, 40, 45, 50\}$, and each neighborhood set is represented by six regions of various types: polyhedra, ellipsoids and hybrids. The modified instances were used for the evaluation. Each modified instance consists of six neighborhood sets, and each neighborhood set is represented by three regions of various types, see Fig. 1.

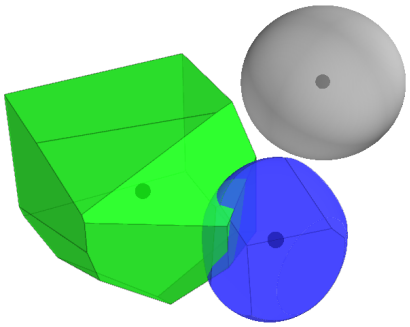

Fig. 1. Detail of a neighborhood set consisting of three neighborhoods: polyhedron (green), ellipsoid (gray), and hybrid (blue).

The BnB algorithm is implemented in C++ and results were obtained using processor Inter Core i5-4460 running at 3.20 Hz and OS Ubuntu 18.04.2 LTS. The quality of the algorithm is measured by the Percentage Deviation (PDB) from from the reference solution of the best solution value among the performed trials as $\%\text{PDB} = \frac{(L-L_{ref})}{L_{ref}} \cdot 100$. L_{ref} is the best-known solution, L is the best solution obtained by the algorithm.

B. Results

The BnB has been compared with CENTROID$^+$ method and GSOA method, both introduced in [3]. The results indicate that the proposed BnB method for lower bounds does not perform very well. Although it does not diverse from the solutions obtained by GSOA and CENTROID$^+$ very much. The reason for this unsatisfying behaviour may be the numerical unstability caused by inappropriately chosen value of "big M", that is calculated for each region as the maximum distance between the region and its bounding box. It would explain the unfeasible solutions reported in Table 1.

Set	No. regions	GSOA		CENTROID+		**BnB**		
		L	T_{CPU} *	L	T_{CPU} *	LB	%PDB	T_{CPU} *
3D_6_3_f	18	1516.81	0.1	**1 513.86**	0.0	1544.67	2.04	0.0
3D_6_3_e	18	1743.76	0.0	**1 737.60**	0.0	1749.92	0.71	0.1
3D_6_3_c	18	**1413.41**	0.0	1431.06	0.0	- *	- *	0.0
3D_6_3_a	18	1573.97	0.0	**1 573.10**	0.0	1578.20	0.32	0.0
3D_6_3_b	18	1452.56	0.0	**1450.11**	0.0	NF**	NF**	0.0
3D_6_3_d	18	**1 498.86**	0.0	1500.37	0.0	1525.64	1.79	0.0

*No value means that the BnB algorithm failed ** Not Feasible

V. Conclusion

In this semestral project I proposed an extension of the well known Branch-and-Bound algorithm for the Close Enough Traveling Salesman Problem to fit the Generalized Traveling Salesmen Problem with Neighborhoods. The BnB has been modified and a new formulation of the SOCP problem for GTSPN has been proposed. The algorithm gives not very good results, although it could be promising with improved method for selecting the "big M".

References

[1] W. P. Coutinho, R. Q. do Nascimento, A. A. Pessoa, A. Subramanian, *"A branch-and-bound algorithm for the close-enough traveling salesman problem"*, INFORMS Journal on Computing, vol. 28, pp. 752–765, 2016.

[2] J. Faigl, *"GSOA: Growing Self-Organizing Array - Unsupervised learning for the Close-Enough Traveling Salesman Problem and other routing problems"*,Neurocomputing, vol. 312, pp. 120-134, 2018.

[3] J. Faigl, P. Vana and J. Deckerova, *"Fast Heuristics for the 3D Multi-Goal Path Plannning based on the Generalized Traveling Salesman Problem with Neighborhoods"*, IEEE Robotics and Automation Letters, pp. 1, 2019

[4] M. de Berga, J. Gudmundssonb, M. J. Katzc, C. Levcopoulosd, M. H. Overmarse, and A. F. van der Stappen, *"TSP with neighborhoods of varying size"*, Journal of Algorithms, vol. 57, no. 1, pp. 22–36, 2005.

[5] D.J. Gulczynski, J.W. Heath, C.C. Price, *"The Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics"*, Springer US, Boston, MA, pp. 271– 283, 2006

[6] J. Faigl, J. Deckerová, *"On Unsupervised Learning based Multi-Goal Path Planning for Visiting 3D Regions,"* Proceedings of the 2018 4th International Conference on Robotics and Artificial Intelligence, ACM, pp. 45-50, 2018.

[7] K. Vicencio, B. Davis, and I. Gentilini, *"Multi-goal path planning based on the generalized traveling salesman problem with neighborhoods"* , IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2985–2990, 2014

[8] F. Carrabs, C. Cerrone, R. Cerulli, M. Gaudioso, *"A novel discretization scheme for the close enough traveling salesman problem"* , Computers & Operations Research, vol. 78, pp. 163-171, 2017.

[9] W. P. Coutinho, R. Q. do Nascimento, A. A. Pessoa, A. Subramanian, *"BnB CETSP"* ,https://github.com/waltonpcoutinho/BnB_CETSP/, 2018.