Computational Study of the Travelling Salesman Problem

Iva Babukova 2030458

December 15, 2016

1 Introduction and Problem statement

The Travelling Salesman Problem (TSP) can be formulated as follows. Given a set of n cities A with distance $d_{i,j}$ between each two cities i and j in A, A cycle of size n through each city in A is also called a *tour*. TSP asks for finding a tour with minimum total sum of the distances between each pair of consecutive cities.

TSP has direct application in multiple fields: transportation and logistics [Flood, 1956], planning inspections of remote sites [Mahalanobis, 1940], genome sequencing [Agarwala et al., 2000], computer chips manufacturing [Barbagallo et al., 1996], data clustering [William T. McCormick et al., 1972] and many others. This implies a high demand in finding an efficient algorithm to solve TSP.

Flood [1956] conjectures that "...there may well be no general method for treating the problem." This statement is later proved by Garey and Johnson [1979] who show that TSP is NP-hard. The complexity of TSP and the high applicability of the problem in multiple fields has led to the development of more than 60 years old research in this area.

This work is a survey of six algorithms for TSP that represent the variety of existing algorithms for TSP. The fundamental work of Dantzig et al. [1954] is first discussed in Section 2. Section 3 is a review of branch and bound algorithms, where two particular techniques are discussed in Section 3.1 and Section 3.2. Section 4 presents a Constraint Programming algorithm for TSP. Section 5 discusses a technique for repeatedly improving a sub-optimal tour within a given time limit. Section 6 presents a method for reducing TSP to another NP-hard problem.

2 Dantzig, Fulkerson and Johnson

Dantzig et al. [1954] represent TSP as a system of linear integer inequalities, based on earlier work by Heller [1953] and Kuhn [1955a]. The formulation is as follows.

Let p be a tour in a TSP instance π and let $x_{i,j}$ be 1 if i is succeeded by j in the tour or 0 otherwise, for $i, j \in A$. Since TSP asks for a cycle of the cities in A, each city in p has to be both preceded and succeeded by one city. Dantzig et al. [1954] enforce this with the following constraint:

$$\begin{aligned} \sum_{j \in A} x_{i,j} &= 1, \qquad i \in A, \\ \sum_{i \in A} x_{i,j} &= 1, \qquad j \in A, \end{aligned} \tag{1}$$

Note that constraint (1) allows for cycles in p of size $n_1 < n$, also called *subtours*, since each city can still have only one preceding and one succeeding city in such arrangements. To filter out subtours, Dantzig et al. [1954] add the *subtour elimination* constraint:

$$\sum_{S} x_{i,j} \leq n_1 - 1, \qquad \forall S \subset A;\, n_1 = |S| > 0;\, i, j \in S \tag{2}$$

Equation (2) restricts that the sum of $x_{i,j}$ over all n_1 cities in a subset S of A has to be less than n_1, which combined with constraint (1) restricts all cities in S to be connected with at least one city not in S.

TSP is then formulated as the problem of minimising:

$$\sum_{i \in A} \sum_{j \in A} d_{i,j} x_{i,j} \tag{3}$$

subject to constraints (1) and (2).

The solution procedure is explained in the next paragraphs. It follows a branch and bound technique. Although the branch and bound method was coined by Little et al. [1963], it is believed that Dantzig et al. [1954] is the first work that uses it [Lawler et al., 1987]. Interestingly, the paper does not emphasise on this invention, but focuses more on one of the procedures in the algorithm, called the *cutting planes* method, which is described later.

Let π' be the problem consisting of minimising function (3) subject to constraint (1) only. From the properties of subtours, each solution to π as a solution to π', however a solution to π' need not be a solution to π [Dantzig et al., 1954]. Dantzig et al. [1954] call π' a *relaxation* of π and use it as part

of the branch and bound procedure to compute the lower bound on the tour length, using the algorithm of Kuhn [1955b] which has complexity $\mathcal{O}(\backslash^{\ni})$.

This relaxation has great impact on the research in algorithms for NP-hard problems in general and it is still studied nowadays [Cheung, 2005]. Relaxations to other problems are defined and successfully used for various NP-hard problems [Fisher, 2004, Cruz-Chávez et al., 2010, Christofides et al., 1981]. A possible reason why the field of problem relaxations flourished could be that this method has good performance in general and it allows for the reuse of already existing efficient algorithms to solve the relaxation.

TSP is then solved as follows:

1. Solve π', obtain a path P'. Go to Step 2.

2. If P' does not contain a subtour: go to Step 3. If P' contains a subtour S: impose constraint (2) on S and obtain a new path P''. $P' = P''$. Go to Step 2.

3. Output P', it is the optimal solution and terminate the program.

Note that there were no computer resources available and Dantzig et al. [1954] executed this algorithm by hand for the 49-city instance. The procedure steps are outlined in the paper through a series of small examples and the authors prove that it outputs an optimal tour.

Step 2 is the *cutting planes* method. It consists of imposing the elimination constraint only to eliminate subtours of the initially computed path. Applegate et al. [2007] observe that "... the TSP literature from 1950s does not include any attempts to automate the Dantzig et al. [1954] solution procedure...". Dantzig et al. [1959] write: "... judging from the number of queries we have received from readers, this method was not elaborated sufficiently to make the proposal clear..." and present a thorough explanation of the cutting planes method. This suggests that the reason for the low utilization of the cutting planes method in subsequent work is that it was not well understood by other researches in the field.

The work of Dantzig et al. [1954] was a computational record for more than 20 years. One of the algorithms capable of solving instances with more than 2000 cities, introduced by Padberg and Rinaldi [1991] is based on the work of Dantzig et al. [1954].

The work of Dantzig et al. [1954] is described as a "breakthrough" that "...has reached far beyond the narrow confines of the TSP..." [Applegate et al., 2007]. This method sets a standard for modelling not only TSP instances, but also instances of other problems [Laporte et al., 1986, Baker, 1983, Agarwal et al., 1989].

3 Branch and Bound Algorithms

Lawler and Wood [1966] refer to branch and bound methods as "intelligently structured search of the space of all feasible solutions'. Starting with the set of all feasible solutions, the set is partitioned into successively smaller subsets, also called *subproblems*, according to a specific *branching rule*. A lower bound is calculated for the cost for each set of subproblems. Subproblems with bound that exceeds the lowest bound $l^\star$ found so far are discarded from the subproblem set. Otherwise, the subproblems are further partitioned. The procedure terminates when the subproblems can not be further partitioned and outputs the solution with bound equal to $l^\star$.

The performance of branch and bound algorithms depends on the choice of branching rule and bounding algorithm [Christofides et al., 1981, Lawler and Wood, 1966]. The next two sections present two different techniques for this.

3.1 The Branch and Bound method by Little, Murty, Sweeny and Caroline

Although Dantzig et al. [1954] use branch and bound algorithm as an extension of the cutting plane method, this technique is coined by Little et al. [1963].

Branching

The branching procedure can be represented as a rooted binary tree T, where the root of T represents all possible tours and each intermediate node is a partial tour. Each node X has two children: Y and $\neg Y$, which split the tours into two disjoint sets, based on whether a pair of cities is contained in the tours or not. For instance, let X be partitioned with respect to the pair of cities (i, j). Then Y represents all tours that visit city j immediately after city i and $\neg Y$ represents all other tours. The branching is done for every pair of cities and it will eventually yield complete tours.

Since there are n cities, there are $n!$ possible tours, which is equal to the number of leaf nodes in T. From properties of binary trees, the number of nodes in T in the worst case is $2n! - 1$. Therefore, if the lower bound on the tour distance is not tight enough and if the choice of the next pair of cities to be branched on is poor, the branching procedure will visit $2n! - 1$ states.

Let D be an $n \times n$ matrix containing the distance between each pair of cities in A. To avoid the worst case scenario, Little et al. [1963] split X, such that Y is likely to include an optimum and $\neg Y$ unlikely to include it

by choosing next pair of cities (i, j) for Y that have smallest distance in the reduced version of D. The reduction procedure is explained in the next section.

Lower Bounds

The lower bound on the tours $\omega(\star)$ is obtained using the following reduction of the distance matrix D. Let l be a variable, initially set to zero. For each row in D, the smallest element h in the row is added to l and substracted from each element in the row. This process is repeated until D contains at least one zero element in each row and column and $\omega(\star) = l$. The lower bound of the root of the branching tree T is equal to $\omega(\star)$.

The reduction of D is correct, since in a tour each city is used only once as incoming and outgoing. If a given constant h is substracted from each element of a row in D, the total distance of a tour obtained under the new distance matrix will be h less than under the old matrix.

Little et al. [1963] give a thorough explanation of their TSP algorithm, representing each of its steps in a transition diagram. In summary, the procedure continuously reduces D until it becomes a 2×2 matrix and outputs the best tour $t^\star$ found so far. In the branching tree T, for a given node X that branches to Y and $\neg Y$ with respect to (i, j), the lower bound $\omega(\neg Y)$ of $\neg Y$ is equal to $\omega(X) + d(i, j)$, where $d(i, j)$ is the distance between i and j in the reduced D. For every branch containing (i, j), row i and column j are deleted in D and the matrix is further reduced. During the reduction, a *subtour blocking* procedure is executed that assigns $d(p, q) = \infty$ for each pair of cities (p, q) that creates subtours in the partial tours. Then, $\omega(Y) = \omega(X) + c$, where c is the sum of the substracted constants during the reduction. Each node X is partitioned if and only if $\omega(X) < t^\star$.

There are similarities between the work of Dantzig et al. [1954] and this TSP algorithm. For instance, obtaining Y from X is equivalent to assigning $x_{i,j} = 1$ and $\neg Y$ is the same as assigning $x_{i,j} = 0$. Repeatedly deleting rows and columns from D enforces constraint (1). The subtour blocking procedure is equivalent to the subtour elimination constraint.

Little et al. [1963] propose several improvements of this algorithm: a procedure that stores small number of nodes instead of the entire search tree, exploiting symmetry of the city distances and some improvements to the branching rule. They observe a trade-off between storage and computation time.

The experiments of Little et al. [1963] show that the computation time grows exponentially with respect to the number of cities in the TSP instance. The maximum size of the studied problems is 40 cities and they take a little

over 8 minutes to solve. The smallest problems have 10 cities and they are solved for few seconds. Compared with novel algorithms for that time, this is an improvement.

3.2 Lower bound based on Min Weight Spanning trees

This section discusses a method for computing lower bounds based on the minimum weight spanning tree introduced by Held and Karp [1971]. The lower bound is used as part of a branch and bound procedure.

Minimum-weight 1-tree and TSP

A tree is an undirected graph with no cycles. A 1-tree is a tree with one extra edge that forms a cycle. If the cycle goes through all vertices, then the 1-tree is equivalent to a tour. The minimum-weight 1-tree is the minimum cost 1-tree among all possible 1-trees formed from a given set of vertices and edges. The cost is the sum of all edges in the 1-tree. In particular, every tour is a 1-tree, but not all 1-trees are tours. Therefore, if a minimum-weight 1-tree is a tour, then it is a tour of minimum weight. Therefore, the cost of the minimum-weight 1-tree in a TSP instance can serve as a lower bound on the cost of the optimal tour. This idea is similar to the relaxation of TSP by Dantzig et al. [1954] discussed in Section 2.

Flood [1956] notes that for every n-vector $\alpha = (\alpha_1, ..., \alpha_n)$, if the weight w_{ij} of the edge between i and j is changed, such that $w_{ij} = w_{ij} + \alpha_i + \alpha_j$, then the TSP instance remains equivalent, but this changes the minimum 1-tree. This is used for deriving a lower bound on the cost of the optimal tour $C^\star$. At the heart of this idea lies the method of Little et al. [1963] of deriving lower bounds based on the reductions of the distance matrix, described in the previous section.

Let the cost of the k^{th} 1-tree be T^k. Then, a new cost of the optimal tour $C^{\star\prime}$ can be obtained: $C^{\star\prime} = C^\star + 2\sum_{i=1}^{n} \alpha_i$, where the cost of the k^{th} tree is $T^{k\prime} = T^k + \sum_{i=1}^{n} \alpha_i d_{ik}$, where d_{ik} is the degree of vertex i in the k^{th} tree. Thus

$$C^\star \geq \min_k [T^{k\prime} - 2\sum_{i=1}^{n} \alpha_i d_{ik}] = w(\alpha)$$

The lower bound of the tour is the maximum value of $w(\alpha)$ over all values of α Held and Karp [1971]. This is computed via an iterative procedure, called the *ascent method*. This procedure is called for every subproblem that is generated after the branching step of a branch and bound algorithm, which is not discussed by Held and Karp [1971].

Held and Karp [1971] give a thorough explanation of the ascent method and prove its correctness mathematically. In summary, the procedure computes a sequence of n vectors according to a recursive formula, based on the minimum-weight 1-tree at a previous point. In addition, the authors show theoretically that the performance of the ascent method may be better when the problem involves large number of linear inequalities.

Computational results

Held and Karp [1971] conduct an empirical study on the ascent method using 18 TSP instances of different nature with size ranging from 20 up to 64 cities. The experiments allow for only one iteration of the ascent method. For 6 of the instances, the computed lower bound was equal to the length of an optimal tour. For the remaining instances, the ascent method derived a lower bound, which is very close to the length of an optimal tour. This suggests that this procedure produces tight lower bounds, as expected.

Held and Karp [1971] observe that the computation time for each instance is considerably large. Considering the size of the search trees, this is an indication of the high complexity of the ascent method. Improvement techniques for the ascent method procedure are suggested that would decrease the total computation time. It should be noted that the improvement techniques introduce a trade-off between the size of the explored search space and the computational cost of visiting each state of the search. This may mean that the running time is not improved, although less number of states are visited.

4 TSP and Constraint Programming

Constraint Programming (CP) is a method to solve optimisation problems by expressing the problem as a set of *decision variables*, each with a set of possible values, called its *domain* and a set of rules, called *constraints* concerning the assignment of domain values to variables. This section discusses the work of Caseau and Laburthe [1997] that constructs a set of CP techniques for solving TSP of small to moderate size, arguing that limited research is done on small travelling salesman problems, although they are often encountered in practise.

Caseau and Laburthe [1997] construct a CP model based on the problem relaxation discussed in Section 2. Each city i is a variable that can be assigned as value any other city. Having $i = j$ means that the next visited city after i is j. There are several options for choosing the next variable to be assigned a value. A rule that determines the variable of choice is called a *branching*

rule and it has great impact on the algorithm efficiency [Beck et al., 2003, Gent et al., 1996]. Let a and b be the two cities closest to i. Caseau and Laburthe [1997] introduce the notion of *regret* of i, which is equal to the difference between a and b, that is $|a - b|$. They propose as branching rules several combinations of largest regret and the fail-first principle [Haralick and Elliott, 1980] [1]. Caseau and Laburthe [1997] prune suboptimal partial solutions using a lower bound on the tour distance equal to the sum of the distances between each city and its nearest city.

Caseau and Laburthe [1997] reduce the number of partial solutions that lead to subtours by restricting the domains of certain variables upon each assignment with the following procedure. Upon every assignment $i := j$, all paths adjacent to i and j are checked. Let a be a path of size p that ends in i and b be a path of size q that starts from j. If $p + q < n - 2$, then the variable at the end of b can not be assigned the city at the start of a as value, because that would lead to a cycle of size less than n.

Note that the aforementioned restrictions can still lead to subtours. Caseau and Laburthe [1997] propose a *lookahead* technique that predicts when certain partial variable assignments will inevitably lead to an invalid tour by detecting assignments that are necessary to the cyclic property of a TSP tour.

Caseau and Laburthe [1997] use a set of TSP instances with 17 to 120 cities in order to identify the most effective combination of the proposed techniques. They argue that the effectiveness of each technique depends on the problem characteristics. The evaluation confirms the hypothesis that additional constraints make the problem easier by drastically restricting the solution space. This result is not surprising. The success of the fail-first principle and its wide application in the existing literature has already confirmed this statement [Golomb and Baumert, 1965, Brélaz, 1979].

In summary, Caseau and Laburthe [1997] show that CP is an effective choice to solve small TSPs by investigating a set of CP-based techniques. They propose a guide of choosing the best solution approach, based on the nature of the TSP instance.

5 The Lin-Kernighan algorithm

This Section discusses the work of Lin and Kernighan [1973] that belongs to a family of *tour-improvement* algorithms. Given a tour T, an upper bound

[1] The fail-first principle is a famous "rule of thumb", also known as heuristic, that says that the next variable chosen by the branching rule should be the one which is most likely going to lead to invalid solution

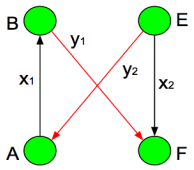

Figure 1: Illustration of Step 2(b) of 2-opt

$u(t)$ of the running time, tour-improvement algorithms repeatedly modify T in order to obtain a tour of smaller length until the elapsed time reaches $u(t)$. This approach is motivated by the fact that often it more important to find "good answers" in "feasible running times" [Lin and Kernighan, 1973].

Tour improvement procedure

Let E be the set of all possible pairs of consecutive cities in a given TSP instance. The algorithm executes the following steps:

1. Generate a random tour with set pairs of consecutive cities $T \subset E$.
2. (a) Set $i = 1$.
   (b) Select (x_i, y_i): $x_i \in T$ and $y_i \in E \setminus T$ such that exchanging $x_1 ... x_i$ with $y_1 ... y_i$ is also a tour and it leads to *maximum improvement*.
   (c) If no such (x_i, y_i) exists, go to Step 3. Otherwise, increment i and repeat Step 2(b).
3. If $\sum_{i=1}^{k} x_i > \sum_{i=1}^{k} y_i$: update T by exchanging all x_i with their corresponding y_i and go to Step 2. Otherwise, go to Step 4.
4. If computation time is not exceeded and search has more to explore: go to Step 1. Otherwise: return T.

Figure 5 is an example for k=2, where (x_1, y_1) and (x_2, y_2) are the pairs selected by Step 2(b). If $x_1 + x_2 > y_1 + y_2$, then x_1 and x_2 will be exchanged with y_1 and y_2 at Step 3. [Lin and Kernighan, 1973] perform several experiments and show that the value of k has substantial influence on the performance of the algorithm and that the best choice for k depends on the size of the TSP instance. Due to the importance of k, the algorithm is often referred to as *k-opt*.

Additional refinements are added on top of the outlined steps of the algorithm. One of the refinements aims to minimise *checkout time*, which is the time spent between first encounter of an optimal tour T and the second time

the algorithm finds T after exploring most options for tour improvement. The experiments have shown that this refinement is highly valuable, as on average 30 to 50 percent of computation time is spend on checkout time.

Empirical Analysis

The algorithm was tested on 36 instances with up to 110 cities each[2] and it was let to run for time proportional to $\mathcal{O}(n^2)$ iterations, where n is the number of cities. Lin and Kernighan [1973] observe that the frequency of finding an optimal tour is close to 1 for instances with up to 42 cities and for 100-city problems it drops to 0.2-0.3 on average[3].

The algorithm was always able to find local optimum for all instances. This is a remarkable result. It shows that this algorithm is highly useful for cases when computation time is crucial and near-optimal solution is acceptable.

The experiments show that as the allowed computation time is increased, the algorithm gives better solutions. This suggests that there is a trade off between running time and tour quality.

The authors comment on the hardness of TSP instances involved in their experiment, a subject of research in later years for other NP-hard problems [Cheeseman et al., 1991, McCreesh et al., 2016]. Lin and Kernighan [1973] report that all problems are relatively easy, where random problems with distances not under the triangle are the hardest and the classical problems are the easiest.

Later work confirms the effectiveness of this algorithm for large TSPs [Applegate et al., 2003]. Applegate et al. [2007] write that "...at the heart of the most successful tour-finding approaches to date lies the simple and elegant algorithm of Lin and Kernighan [1973]."

6 TSP and the Longest Path Problem

A common way to solve a TSP instance π is to reduce it to another problem P for which there is a known efficient algorithm. Then, using the algorithm one solves P and this gives the solution for π. This procedure is not equivalent to relaxation, since every solution to P is a solution to π and vice versa. This section is a review of a work by Hardgrave and Nemhauser [1962] that makes a reduction from TSP to the Longest Path Problem (LPP).

[2]The upper bound on the city number is imposed due to computer storage limitations.

[3]These numbers are derived only for the classical problems, where the optimal solution is known.

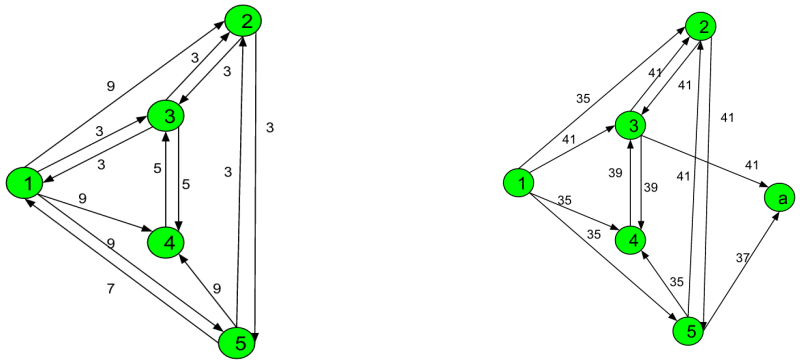

Figure 2: Construction of $L(G)$ from G, where $S = 9 + 9 + 9 + 9 + 7 = 43$ and $K = 44$

The Longest Path Problem (LPP) asks for finding a path between two specified vertices in a graph, such that the path has maximum length and does not contain any repeated vertices. LPP was not well researched at the time when Hardgrave and Nemhauser [1962] was published and its NP-hadness was yet to be established [Garey and Johnson, 1979]. Hardgrave and Nemhauser [1962] propose the following reduction from TSP to LPP with the hope that LPP "... will prove to be easier ..." than TSP.

A TSP instance can be represented as a directed labelled graph $G = (V, E, l)$, with a set of vertices $V = A$, a set of edges $E = \{(i, j) \,|\, i, j \in A \wedge i \neq j\}$ and a function l that assigns a label on each edge, such that $\forall (i, j) \in E$, $l(i, j) = d_{i,j}$. A tour in TSP is then equivalent to a path in G that visits each vertex once and returns to the starting city v_1. Hardgrave and Nemhauser [1962] reduce TSP to LPP by showing that for every G there exists a graph $L(G)$, such that a longest path in $L(G)$ corresponds to an optimal tour in G.

Graph $L(G) = (V', E', l')$ has a set of vertices $V' = V + a$, where a is an additional vertex, a set of edges E' that are the edges in E except from the edges from any vertex to the start vertex, that is (i, v_1) plus edges (i, v_1) if and only if $(i, v_1) \in E$, and an edge labelling function l':

$$l'(i, j) = \begin{cases} K - d_{i,j}, & \text{if } j \neq v_1 \\ K - d_{i,v_1}, & \text{if } j = v_1, \end{cases}$$

where K is a constant that is strictly greater than the sum S of the n greatest distances $d_{i,j}$.

Figure 6 is an example of $L(G)$, constructed from a particular graph G. An optimal tour in G is $(1, 4, 3, 2, 5)$ and it is equivalent to the longest path in $L(G)$, which is $(1, 4, 3, 2, 5, a)$.

Hardgrave and Nemhauser [1962] prove the correctness of the reduction and propose several procedures for solving LPP, which prove to be either not successful or inefficient. The proposed procedures are still useful for the research community as a guide of what "not to do" in order to solve LPP.

One of the ideas of Hardgrave and Nemhauser [1962] is to find the longest spanning tree in $L(G)$ between v_1 and a, which is similar to the bounding procedure by Held and Karp [1971], discussed earlier. Hardgrave and Nemhauser [1962] are not clear about why this approach is unsuccessful. One of the suggested reasons is a failure to establish a method for choosing an initial spanning tree.

In summary, Hardgrave and Nemhauser [1962] present an interesting idea of reducing an NP-hard problem to another NP-hard problem, with the hope that the latter problem will be easier to solve. They propose several procedures for solving LPP and give detailed explanation about why each of the procedures failed. Hardgrave and Nemhauser [1962] do not provide a successful method for tackling TSP and LPP, but their work is a lesson about how certain procedures can lead to a dead-end.

7 Conclusion

The Travelling Salesman Problem (TSP) is a combinatorial optimisation problem that is applicable to multiple fields. This work introduced six influential approaches for solving TSP. Most of the methods implement similar ideas to the work of Dantzig et al. [1954] [Caseau and Laburthe, 1997, Held and Karp, 1971, Little et al., 1963]. This shows that Dantzig et al. [1954] is one of the fundamental works on TSP. This work showed that branch and bound algorithms are very effective techniques for solving TSP and demonstrated the flexibility of Constraint Programming for modelling optimisation problems.

This work reviewed a tour improvement algorithm for TSP by Lin and Kernighan [1973], which showed that it is often the case that an optimal or near-optimal solution is found relatively fast and most of the computation effort goes for proving that the solution is an optimum. This shows that tour-improvement techniques can significantly reduce the computation time and enable to solve substantially larger TSP instances, provided that a proof of their optimality is not required.

The reader should note that this work does not aim to present a review of the entire field of TSP. Due to the volume of existing work on TSP, that would prove impossible to do within a document of this size. The interested reader should refer to Applegate et al. [2007], Lawler et al. [1987] for more

detailed overview of TSP algorithms.

References

Yogesh Agarwal, Kamlesh Mathur, and Harvey M. Salkin. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, 19(7): 731–749, 1989. doi: 10.1002/net.3230190702. URL `http://dx.doi.org/10.1002/net.3230190702`.

R. Agarwala, D. Applegate, D. Maglott, G. Schuler, and A. Schäffer. A fast and scalable radiation hybrid map construction and integration strategy. *Cold Spring Harbor Laboratory Press*, pages 350–364, 2000. ISSN 1088-9051.

David Applegate, William J. Cook, and André Rohe. Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003. doi: 10.1287/ijoc.15.1.82.15157. URL `http://dx.doi.org/10.1287/ijoc.15.1.82.15157`.

David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007. ISBN 0691129932, 9780691129938.

Edward K. Baker. Technical note - an exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):938–945, 1983. doi: 10.1287/opre.31.5.938. URL `http://dx.doi.org/10.1287/opre.31.5.938`.

S. Barbagallo, M. Lobetti Bodoni, D. Medina, F. Corno, P. Prinetto, and M. Sonza Reorda. Scan insertion criteria for low design impact. In *Proceedings of the 14th IEEE VLSI Test Symposium*, VTS '96, pages 26–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7304-4. URL `http://dl.acm.org/citation.cfm?id=832296.836276`.

J. Christopher Beck, Patrick Prosser, and Richard J. Wallace. Toward understanding variable ordering heuristics for constraint satisfaction problems. In *Proceedings of the 14th Irish Artificial Intelligence and Cognitive Science Conference*, pages 11–16, 2003.

Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

Yves Caseau and François Laburthe. Solving small tsps with constraints. In *Logic Programming, Proceedings of the Fourteenth International Con-*

ference on Logic Programming, Leuven, Belgium, July 8-11, 1997, pages 316–330, 1997.

Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, pages 331–337, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1-55860-160-0. URL `http://dl.acm.org/citation.cfm?id=1631171.1631221`.

Kevin KH Cheung. On lovász–schrijver lift-and-project procedures on the dantzig–fulkerson–johnson relaxation of the tsp. *SIAM Journal on Optimization*, 16(2):380–399, 2005.

Nicos Christofides, Aristide Mingozzi, and Paolo Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164, 1981.

Marco Antonio Cruz-Chávez, Alina Martinez-Oropeza, and Rafael Rivera López. Relaxation of job shop scheduling problem using a bipartite graph. In *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010*, pages 132–136. IEEE, 2010.

G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. On a linear-programming, combinatorial approach to the traveling-salesman problem. *Oper. Res.*, 7(1):58–66, February 1959. ISSN 0030-364X. doi: 10.1287/opre.7.1.58. URL `http://dx.doi.org/10.1287/opre.7.1.58`.

George B. Dantzig, D. Ray Fulkerson, and Selmer M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2(4):393–410, 1954. doi: 10.1287/opre.2.4.393. URL `http://dx.doi.org/10.1287/opre.2.4.393`.

Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 50(12-Supplement):1861–1871, 2004. doi: 10.1287/mnsc.1040.0263. URL `http://dx.doi.org/10.1287/mnsc.1040.0263`.

Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4 (1):61–75, 1956. doi: 10.1287/opre.4.1.61. URL `http://dx.doi.org/10.1287/opre.4.1.61`.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Bell Telephone Laboratories, Incorporated, United States of America, 1979.

Ian P. Gent, Ewan MacIntyre, Patrick Presser, Barbara M. Smith, and Toby Walsh. *An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem*, pages 179–193. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. ISBN 978-3-540-70620-5. doi: 10.1007/3-540-61551-2_74. URL http://dx.doi.org/10.1007/3-540-61551-2_74.

Solomon W Golomb and Leonard D Baumert. Backtrack programming. *Journal of the ACM (JACM)*, 12(4):516–524, 1965.

Robert M Haralick and Gordon L Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial intelligence*, 14(3):263–313, 1980.

William W Hardgrave and George L Nemhauser. On the relation between the traveling-salesman and the longest-path problems. *Operations Research*, 10(5):647–657, 1962.

Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Math. Program.*, 1(1):6–25, 1971. doi: 10.1007/BF01584070. URL http://dx.doi.org/10.1007/BF01584070.

I. Heller. On the problem of the shortest path between points. *Bulletin of the American Mathematical Society*, pages 551–551, 1953.

H. W. Kuhn. On certain convex polyhedra. *Bulletin of the American Mathematical Society*, pages 557–558, 1955a.

Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955b.

G. Laporte, H. Mercure, and Y. Nobert. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1):33–46, 1986.

E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966. doi: 10.1287/opre.14.4.699. URL http://dx.doi.org/10.1287/opre.14.4.699.

Eugene L. Lawler, David B. Shmoys, Alexander H. G. Rinnooy Kan, and Jan K. Lenstra. *The Traveling salesman problem : a guided tour of*

combinatorial optimization. Wiley interscience series in discrete mathematics and optimization. J. Wiley and sons, Chichester, New York, Brisbane, 1987. ISBN 0-471-90413-9. URL http://opac.inria.fr/record=b1117783.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.*, 21(2):498–516, April 1973. ISSN 0030-364X. doi: 10.1287/opre.21.2.498. URL http://dx.doi.org/10.1287/opre.21.2.498.

John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Oper. Res.*, 11(6):972–989, December 1963. ISSN 0030-364X. doi: 10.1287/opre.11.6.972. URL http://dx.doi.org/10.1287/opre.11.6.972.

P. C. Mahalanobis. A sample survey of the acreage under jute in bengal. *Sankhya: The Indian Journal of Statistics (1933-1960)*, 4(4):511–530, 1940. ISSN 00364452. URL http://www.jstor.org/stable/40383954.

Ciaran McCreesh, Patrick Prosser, and James Trimble. Heuristics and really hard instances for subgraph isomorphism problems, 2016. URL http://eprints.gla.ac.uk/119086/.

Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991. doi: 10.1137/1033004. URL http://dx.doi.org/10.1137/1033004.

Jr. William T. McCormick, Paul J. Schweitzer, and Thomas W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972. doi: 10.1287/opre.20.5.993. URL http://dx.doi.org/10.1287/opre.20.5.993.