Solution of asymmetric traveling salesman problems combining the volume and simplex algorithms

Ana Maria A.C. Rocha* Edite M.G.P. Fernandes*
João Luís C. Soares†

10th December 2004

Abstract

In this paper we investigate the usage of the Lagrangian approach as a mechanism of speeding up the optimal basis identification in simplex methods. More precisely, we have combined the volume algorithm (a variation of the subgradient algorithm) with a simplex implementation like CPLEX on a strong linear programming formulation of the asymmetric traveling salesman problem known by extended disaggregated flow formulation. We compare the efficiency of our combined approach on a selection of test problems from the TSPLIB collection and on randomly generated instances. Numerical results also include heuristic finding of integer solutions and results are compared against some well-known heuristic methods.

1 Introduction

Some linear programs arising from real-world applications have such a large number of variables and/or constraints that they cannot be dealt with simplex type algorithms or even interior point methods. While the critical aspect for interior-point methods is memory requirements, in simplex methods it is often the initial basis choice. This paper is part of an ongoing research project in which the authors are engaged to study the usage of fast approximation schemes to speeding up the optimal basis identification in simplex algorithms. Problems of particular interest are the linear programming relaxations of difficult combinatorial optimization problems, namely, those that possess the following characteristics: (a) are large in the number of variables and/or constraints, and (b) have an embedding complicating characteristic that if removed makes the problem 'easy'.

The Asymmetric Traveling Salesman problem (ATSP) is a notorious NP-complete problem that fits in this group. The ATSP is the problem of finding the cheapest tour in a given directed graph $G = (V, E)$ with a real cost c_{ij} associated with each arc (i, j). Its algebraic formulation requires a large number of variables - at least $\mathcal{O}(n^2)$, where n is the number of nodes - and without the 'no subtours are allowed' group of constraints the problem becomes solvable in $\mathcal{O}(n^3)$ operations.

*Departamento de Produção e Sistemas, Universidade do Minho, {arocha;emgpf}@dps.uminho.pt
†Departamento de Matemática, Universidade de Coimbra, jsoares@mat.uc.pt

In practice, the seek of an optimal solution for the ATSP is complicated by the large number of variables. One of its first formulations was proposed by Dantzig, Fulkerson and Johnson (DFJ) [11] and still is the formulation of choice for state-of-art codes like CONCORDE [2] (in the symmetric case) and the one used in [12]. In spite of this formulation having an exponential number of constraints (the subtour elimination constraints), its linear programming relaxation can be solved in polynomial time by the ellipsoid algorithm. In practice, it is often solved by constraint generation algorithms [12].

Several authors have proposed formulations for the ATSP that possess a non-exponential number of constraints at the expense of a (non-exponential) increase in the number of variables. This is the case with those formulations proposed in [26, 13, 9, 14] to name a few. Surveys on formulations for the ATSP include [27, 24]. None of these linear programming formulations dominates the DFJ formulation in strength or adequacy to the solution by branch-and-bound methods.

The fast determination of an approximated solution to the linear programming relaxation of the (symmetric) TSP was initiated by Held and Karp [15, 16] through Lagrangian relaxation. In general, using Lagrangian relaxation means minimizing a convex and piecewise linear function over a polyhedral set. Typically, this nondifferentiable problem is solved by the projected subgradient algorithm. An inherent difficulty with Lagrangian relaxation lies in the fact that, by halting the subgradient algorithm, the solution found may be far from primal feasibility and quite often it is not clear how it may help in finding the original optimum (integer or fractional).

Recently, a variation of the subgradient algorithm, known by volume algorithm has proven to produce 'close-to-optimal' solutions that are quite close to primal feasibility [5]. Motivated by this improvement on the subgradient algorithm, we conjectured that by first applying the volume algorithm it would be possible to jump start the solution of the fractional ATSP by a simplex solver like CPLEX. A straightforward application of the volume algorithm on the Held and Karp dual problem would suffer from the fact that it is not efficient to reoptimize shortest spanning trees. Thus, instead of considering the symmetric case we considered the unsymmetric version of the Traveling Salesman Problem for which reoptimization of the Lagrangian function is, as we shall see, efficient.

In this work we approach the solution of the fractional ATSP through applying Lagrangian relaxation on a formulation known as extended disaggregated flow formulation. Primal and dual information obtained through the volume algorithm is used to guess a good initial basis for CPLEX. The reduced costs are also used to guide heuristic algorithms in the search of integer solutions. Computational results were obtained for a selection of problems from *TSPLib* and others were randomly generated. The article is structured as follows. Section 2 introduces the disaggregated flow formulation. Section 3 presents the Lagrangian problem. Section 4 recalls the volume algorithm especially adapted to our study. Section 5 presents numerical experiments and in Section 6 we conclude and identify potential areas of further study.

2 The extended disaggregated flow formulation

Let $G = (V, E)$ be a directed graph with n vertices and m arcs. The set of characteristic vectors of tours in G is a subset of the set of integer vectors $x \in \mathbb{R}^m$ that satisfy the

following system of equalities and inequalities

$$\left.\begin{array}{rl} \sum\limits_{(i,j)\in\delta^+(i)} x_{ij} & = 1 \quad (i \in V) \\ \sum\limits_{(i,j)\in\delta^-(j)} x_{ij} & = 1 \quad (j \in V) \\ x_{ij} & \geq 0 \quad ((i,j) \in E) \end{array}\right\} \tag{1}$$

where $\delta^-(j)$ denotes the set of incoming arcs into vertex j and $\delta^+(i)$ denotes the set of outgoing arcs from vertex i. We denote the polyhedron defined by (1) as P^{ASS} because the constraints define a (restricted) assignment problem. The characteristic vectors of tours are precisely the set of integer vectors x that satisfy (1) and for which there are vectors $y^2, y^3, \ldots, y^n \in \mathbb{R}^m$ satisfying the following system of equalities and inequalities

$$\left.\begin{array}{rl} \sum\limits_{(i,j)\in\delta^+(i)} y^k_{ij} - \sum\limits_{(j,i)\in\delta^-(i)} y^k_{ji} = \left\{\begin{array}{ll} -1 & \text{if } i = 1 \\ 0 & \text{if } i \neq 1, k \\ 1 & \text{if } i = k \end{array}\right\} & (i \in V, k \in V_1) \\ 0 \leq y^k_{ij} \leq x_{ij} & ((i,j) \in E, k \in V_1) \end{array}\right\} \tag{2}$$

where $V_1 \equiv V \setminus \{1\}$. Constraints (1) and (2) characterize a polyhedron lying in $\mathbb{R}^{n\times m}$ whose projection into the m-dimensional space of the x variables is a polyhedron denoted P_L^{ATSP}. It may be shown that the convex hull of characteristic vector of tours, denoted P^{ATSP}, satisfies $P^{\text{ATSP}} = P_L^{\text{ATSP}} \cap \mathbb{Z}^m$ and $x \in P_L^{\text{ATSP}}$ if and only if x satisfies (1) and

$$\sum_{(i,j)\in E(S)} x_{ij} \leq |S| - 1 \quad (S \subseteq V_1) \tag{3}$$

where $E(S)$ denotes the set of arcs of the induced graph $G[S]$. Constraints (1) and (2) are the *extended disaggregated flow formulation* of the ATSP for its interpretation in the terms of flows. The underlying interest in using (2) instead of (3) has to do, as it will become apparent in the next section, with the number of constraints and its matrix structure.

3 The Lagrangian problem

As it was explained in the previous section, the linear optimization problem $z_L^* \equiv \min\{cx \colon x \in P_L^{\text{ATSP}}\}$, and its dual, may be formulated as follows

$$z_L^* = \left\{\begin{array}{lllll} \min & cx & & & \\ \text{s.t.} & Ax & = & \mathbb{1} & \\ & By^k & = & b^k & (k \in V_1) \\ & x - y^k & \geq & 0 & (k \in V_1) \\ & y^k & \geq & 0 & (k \in V_1) \\ & x \geq 0 & & & \end{array}\right\} \tag{4}$$

$$= \left\{\begin{array}{lllll} \max & w\mathbb{1} + \sum_{k\in V_1} \pi^k b^k & & & \\ \text{s.t.} & wA + \sum_{k\in V_1} \rho^k & \leq & c & \\ & \pi^k B - \rho^k & \leq & 0 & (k \in V_1) \\ & \rho^k & \geq & 0 & (k \in V_1) \end{array}\right\} \tag{5}$$

where A denotes the node-edge incidence matrix of the (undirected) bipartite graph $G' = (V \times V, E)$, $\mathbb{1}$ is a vector of ones, B denotes the node-arc incidence matrix of the directed graph G, and, for each $k \in V_1$, b^k is a vector of all zeros except for $b^k_1 = -1$ and $b^k_k = 1$. Now, consider the Lagrangian problem

$$\max \left\{ z\left(\pi\right) : \pi = \left(\pi^k\right) \in \mathbb{R}^{(|V|-1)\times|V|} \right\} \left(= z^*_L\right) \tag{6}$$

that arises from dualizing the constraints $By^k = b^k, k \in V_1$. Its objective function is defined at each $\pi = (\pi^k)$ by

$$z(\pi) \equiv \left\{ \begin{array}{ll} \min & cx + \sum_{k\in V_1} \pi^k \left(b^k - By^k\right) \\ \text{s.t.} & Ax = \mathbb{1} \\ & x - y^k \geq 0 \quad (k \in V_1) \\ & y^k \geq 0 \quad (k \in V_1) \\ & x \geq 0 \end{array} \right\} \tag{7}$$

$$= \left\{ \begin{array}{ll} \max & w\mathbb{1} + \sum_{k\in V_1} \pi^k b^k \\ \text{s.t.} & wA + \sum_{k\in V_1} \rho^k \leq c \\ & -\rho^k \leq -\pi^k B \quad (k \in V_1) \\ & \rho^k \geq 0 \quad (k \in V_1) \end{array} \right\}. \tag{8}$$

An optimal solution in (7) can be derived from solving a single assignment problem. In fact, for all fixed x, an optimal solution $\bar{y}(x) = [\bar{y}^k(x)]$ in the y variable space is an optimal solution of

$$\begin{array}{ll} \max & \sum_{k\in V_1} \sum_{(i,j)\in E} \left(\pi^k_i - \pi^k_j\right) y^k_{ij} \\ \text{s.t.} & 0 \leq y^k_{ij} \leq x_{ij} \ ((i,j) \in E, k \in V_1). \end{array} \tag{9}$$

An optimal solution for (9) is defined componentwise by

$$\bar{y}^k_{ij}(x) = \left\{ \begin{array}{ll} x_{ij} & \text{if } \pi^k_i - \pi^k_j \geq 0, \\ 0 & \text{if } \pi^k_i - \pi^k_j < 0. \end{array} \right\} \quad ((i,j) \in E, k \in V_1) \tag{10}$$

corresponding an optimal value of $\sum_{k\in V_1} \sum_{(i,j)\in E} \max\left(\pi^k_i - \pi^k_j, 0\right) x_{ij}$. Therefore, problem (7) is equivalent to the following assignment problem

$$\begin{array}{ll} \min & \bar{c}x \\ \text{s.t.} & Ax = \mathbb{1}, x \geq 0, \end{array} \tag{11}$$

where $\bar{c}$ is defined componentwise by $\bar{c}_{ij} = c_{ij} - \sum_{k\in V_1} \max\left((\pi^k_i - \pi^k_j), 0\right)$. More precisely, if $\bar{x}$ is an optimal solution of (11) then $(\bar{x}, \bar{y})$, with $\bar{y} = \bar{y}(\bar{x})$ defined by (10), is an optimal solution to the minimization problem in (7). Problem (11) can be solved by, for example, the Hungarian method in $\mathcal{O}(|V|^3)$ operations.

Moreover, if $\bar{x}$ is an optimal solution of (11) then $z(\pi) = \bar{c}\bar{x} + \sum_{k\in V_1} \left(\pi^k_1 - \pi^k_k\right)$, and an optimal solution $(\bar{w}, \bar{\rho})$ of the dual in (7) is characterized by an optimal dual solution $\bar{w} = (\bar{u}, \bar{v})$ in (11) and by setting $\bar{\rho}^k_{ij} = \max(\pi^k_i - \pi^k_j, 0)$, for all $k \in V_1$ and $(i,j) \in E$. Note also that the vector $(\pi, \bar{w}, \bar{\rho})$ is dual feasible in (4) because, for each $(i,j) \in E$,

$$\left[c - \left(\bar{w}A + \sum_{k\in V_1} \bar{\rho}^k\right)\right]_{ij} = c_{ij} - \left(\bar{u}_i + \bar{v}_j + \sum_{k\in V_1} \max(\pi^k_i - \pi^k_j, 0)\right) = \bar{c}_{ij} - (\bar{u}_i + \bar{v}_j) \geq 0,$$

and, for each $(i,j) \in E$ and $k \in V_1$,

$$\left[0 - \left(\pi^k B - \bar{\rho}^k\right)\right]_{ij} = -\left(\pi_i^k - \pi_j^k - \max(\pi_i^k - \pi_j^k, 0)\right) = \max(\pi_j^k - \pi_i^k, 0) \geq 0.$$

In the next section we will recall the volume algorithm and explain how it was particularly tuned to solving (6). The Lagrangian subproblems, to be solved in each iteration, are restricted assignment problems whose objective function changes very little from iteration to iteration. Hence, these subproblems can be reoptimized quite effectively.

4 The volume algorithm

Practical experience has shown that the volume algorithm produces primal solutions that are quite close to feasibility (as well as good dual solutions). The denomination "volume" reflects the fact that primal values come from computing the volume below the faces of the dual problem. The direction of movement is also given by these volumes. Some convergence properties are studied in [4]. Successful numerical experiments were carried out on fractional set partitioning [5, 1, 6], Steiner tree problems [3], and facility location [7]. The solving of Steiner tree and Max-cut problems through branch-and-cut where the linear programming relaxations are solved by the volume algorithm is studied in [8].

Input: $\pi^0 \in \mathbb{R}^{(|V|-1)\times|V|}$.

Initialization: Solve (7) with $\pi = \pi^0$ to obtain (x^0, y^0).
Compute $v^0 = b - By^0 \in \partial z(\pi^0)$.
Set $\bar{\pi}^1 = \pi^0$, $(\bar{x}, \bar{y}) = (x^0, y^0)$, $\bar{w} = v^0$, $j = 1$ and $l = 1$.

Generic Iteration j:

Step 1: For some stepsize $s_j > 0$, compute $\pi^j = \bar{\pi}^l + s_j \bar{w}$.
Step 2: Let (x^j, y^j) be an optimal solution of (7).
Compute $v^j = [v^{k,j}] \in \partial z(\pi^j)$ with $v^{k,j} = b^k - By^{k,j}$.
Step 3: For some $\alpha_j \in [0,1]$, update

$$(\bar{x}, \bar{y}, \bar{w}) \;=\; \alpha_j \left(x^j, y^j, v^j\right) + (1 - \alpha_j)\,(\bar{x}, \bar{y}, \bar{w}) \qquad (12)$$

Step 4: If $z(\pi^j) > z(\bar{\pi}^l)$, then update $\bar{\pi}^{l+1} = \pi^j$ and set $l \leftarrow l + 1$.
Step 5: Test stopping criterion. Let $j \leftarrow j + 1$ and go to *Step 1.*

Figure 1: The volume algorithm (VA)

As observed in [4], the volume algorithm formally described in Figure 1 has some similarities with the bundle method. In particular, the idea of taking a convex combination between the new and previous direction in each iteration aims at avoiding the typical zig-zagging behavior of the subgradient algorithm and, in this way, accelerates solution convergence - though this is not always observable in practice. The description leaves open the definition of the scalars s_j and α_j involved in the redefinition of the primal

solution $\bar{x}$. Our choices for these two scalars are essentially empirical and similar to the ones used in [5]. The value of α_j is chosen to force primal feasibility of $(\bar{x}, \bar{y})$. As in [5], we first define

$$\alpha_{\text{opt}} \equiv \arg\min \left\| \alpha v^j + (1-\alpha)\,\bar{w} \right\|^2 = \arg\min \sum_{k \in K} \left\| b^k - B\left(\alpha y^{k,j} + (1-\alpha)\,\bar{y}^k\right) \right\|^2 .$$

Then, we set an upper bound to the value of α, denoted $\alpha_{\max}$, and finally set

$$\alpha_j = \begin{cases} \alpha_{\max}/10 & \text{if } \alpha_{\text{opt}} \leq 0, \\ \min(\alpha_{\text{opt}}, \alpha_{\max}) & \text{if } \alpha_{\text{opt}} > 0. \end{cases}$$

In our numerical experiments we used the implementation of the volume algorithm available at `http://www.coin-or.org` (the COIN-OR webpage). Thus, some choices for the values of the parameters are simple settings in the code. For example, $\alpha_{\max}$ is configured by setting the parameter `alphainit=0.01`. After a certain number of iterations (configured by the parameter `alphaint=80`), we test the progress of the dual function $z(\pi)$. If it has increased less than 1% and if $\alpha_{\max}$ is superior to a certain value (configured by the parameter `alphamin=0.0001`) then $\alpha_{\max}$ is multiplied by a certain quantity (configured by the parameter `alphafactor=0.5`).

As most of the implementations of the subgradient method, like the one in [17], the stepsize s_j is computed by the following formula:

$$s_j = \lambda_j \frac{T - z(\bar{\pi}^l)}{\|\bar{w}\|^2} \tag{13}$$

with $\lambda \in (0, 2]$ and T is a *target* value for the optimal value of (6). We start with a small value for T (for example, $T = z(\pi^0)$). Each time $z(\bar{\pi}^l)$ is below 5% of T, the value of T is increased by 5%. In each iteration, the setting of λ obeys to the following basic steps. The initial value of λ is 0.1 (`lambdainit=0.1`) and it is updated depending on three types of iterations: red, yellow and green.

- When $z(\pi^j) \leq z(\bar{\pi}^l)$ the iteration is referred to as red (an improvement in the dual value was not verified). After forty (`redtestinvl=40`) consecutive red iterations (suggesting the need for a smaller stepsize) λ is multiplied by 0.67, unless $\lambda < 0.0005$, in which case it is kept unmodified.

- When $z(\pi^j) > z(\bar{\pi}^l)$, an improvement in the dual value occurs. Hence, we compute

  $$d = \bar{w}\,(b - B\bar{y})\,.$$

  If $d < 0$, the iteration is referred to as yellow. It means that a longer step in the direction $\bar{w}$ would lead to a smaller value for $z(\pi^j)$. After a sequence of two (`yellowtestinvl=2`) consecutive yellow iterations we multiply λ by 1.1.

  If $d \geq 0$, the iteration is referred to as green. It suggests the need for a larger stepsize. After two (`greentestinvl=2`) consecutive green iterations λ is multiplied by 2 up to a maximum value of 2.

Upon termination the following two conditions are imposed. One is based on the checking of the maximum number of iterations allowed

$$\text{number of iterations} \leq \{500, 1000, 5000, 10000\} (= \texttt{maxsgriters}).$$

The other is the conjunction of two conditions:

1. the absolute maximum value of the constraints violation

$$\|\bar{w}\| \leq 0.01 \ (=\texttt{primal\_abs\_precision})$$

2. the relative or absolute difference between the lower bound $z(\bar{\pi}^l)$ and the value of primal solution $c\bar{x}$

$$\frac{\left|c\bar{x} - z(\bar{\pi}^l)\right|}{|z(\bar{\pi}^l)|} \leq 0.01 (=\texttt{gap\_rel\_precision})$$

or

$$\left|c\bar{x} - z(\bar{\pi}^l)\right| \leq 0.05 \ (=\texttt{gap\_abs\_precision}).$$

5 Computational results

This section summarizes the numerical experiments. All the codes were implemented in C++ and all the computational tests were performed on a PC with a 2.66GHz Pentium IV microprocessor and 512Mb of memory running RedHat Linux 8.0.

Table 1 contains some of the characteristics of the selected instances from *TSPLib* library [28]. The first columns of the table identify the problem and the last columns report the performance of the dual simplex method (dualopt) and the interior point method (baropt), as implemented in software CPLEX version 7.0 [20] with default settings, in the solving of (4). Note that for the largest instances, the dual-simplex method is much slower than the interior point method. The superior performance of the interior point method was expected because of the very large number of variables and constraints for these instances. A drastical example is instance p43, for which the interior point method spent only 3,5% of the time spent by the dual simplex method.

Motivated by the difficulty of the dual-simplex method on identifying the optimal basis it occurred to us the idea of using the volume algorithm to some advantage in three aspects. First, use the solution output from the volume algorithm as a means of deriving a good initial basis. Second, use the solution output from the volume algorithm as a mechanism of identifying the arcs that do not participate in the integer optimal solution. Last, use the reduced costs computed in the course of the volume algorithm, to generate integer solutions through heuristic methods.

5.1 Solving the linear relaxation

In this subsection, we analyze the strategy of using the solution output from the volume algorithm as a means of deriving an initial basis for the dual-simplex method. We observed that the volume algorithm has a very differentiated performance for slightly different values of its parameters which was not a surprise for it being an algorithm of the

ATSP INSTANCE					CPLEX (dualopt)		CPLEX (baropt)
ID	$\|V\|$	$\|E\|$	Int. Optimal	Frac. Optimal	simplex It.	Time (sec.)	Time (sec.)
br17	17	272	39	39.0	3055	2.2	0.4
ftv33	34	1122	1286	1286.0	24231	42.7	18.2
ftv35	36	1260	1473	1457.3	31627	76.2	23.5
ftv38	39	1482	1530	1514.3	41240	121.9	32.2
ftv44	45	1980	1613	1584.9	68004	278.9	59.1
ftv47	48	2256	1776	1748.6	100650	627.1	106.5
ftv55	56	3080	1608	1584.0	137580	1035.9	237.7
ftv64	65	4160	1839	1807.5	245467	2762.7	511.2
ftv70	71	4970	1950	1909.0	316196	3719.2	791.0
p43	43	1806	5620	5611.0	131426	2117.6	60.9
ry48p	48	2256	14422	14289.3	98816	600.2	107.0
ft53	53	2756	6905	6905.0	127702	827.1	175.2
ft70	70	4830	38673	38652.5	291086	3657.6	753.4

Table 1: ATSP instances and CPLEX performance.

subgradient class. The parameter values that were used are the ones reported in Section 4. In each iteration of the volume algorithm, the assignment problem (11) is solved by the dual-simplex method. Preliminary computational experiments enabled us to conclude that this algorithm was faster than the network primal simplex method.

Table 2 summarizes our results. Each instance was solved twice by the dual-simplex method, after a small and large number of iterations of algorithm VA, respectively. For the smaller instances we chose a maximum of VA iterations of 500 and 1000; for the medium-sized instances the choice was 1000 and 5000; and, for the larger instances, 5000 and 10000 iterations were used. The initial information supplied to CPLEX is realized through the use of the function CPXcopystart(*env*,*lp*,NULL,NULL,x,NULL,NULL,y), where *env* and *lp* are pointers to the CPLEX environment and problem, respectively, x indicates the primal vector and y indicates the dual vector as they are output from the volume algorithm. CPLEX is left the task of identifying an initial basis.

Column 1 of Table 2 identifies the ATSP instance. Columns 2-6 report the VA results, namely: the maximum number of iterations, the lower bound (based on dual information), the primal objective function value at the last primal solution found; the l_∞ constraint violation value at this last solution, and, the time spent. Columns 7-8 have to do with CPLEX and are as follows: the number of simplex iterations required by the dual simplex method and the time in seconds. The last two columns are the total time spent running the two algorithms and the percentage of the gain in time with this combined strategy (VA+CPLEX) as compared to the execution of CPLEX (dualopt) reported in Table 1. A positive value indicates an improved performance. The registered times of all the tables were rounded to one decimal place.

Some conclusions may be drawn. In all cases, a larger number of iterations of the volume algorithm implies a smaller number of iterations of the dual-simplex method. The reduction is drastic in a few cases. For example, after 5000 iterations of the VA, the

ATSP ID	**VOLUME**					CPLEX		**Total Time**	**Time Gain**
	VA It.	L (dual)	P (primal)	Max. violation	Time (sec.)	Simplex It.	Time (sec.)		
br17	500	33.5	40.4	0.00416	0.7	1208	2.3	3.0	-34.1%
br17	1000	38.3	40.1	0.00237	1.4	1208	2.4	3.8	-70.5%
ftv33	500	1246.2	1304.1	0.04462	2.8	4421	19.1	21.9	48.8%
ftv33	1000	1281.9	1311.1	0.03612	5.5	2700	13.8	19.3	54.7%
ftv35	500	1435.6	1464.2	0.13534	3.2	12449	59.4	62.6	17.8%
ftv35	1000	1454.6	1466.0	0.11960	6.4	2190	13.4	19.8	74.1%
ftv38	500	1496.1	1513.5	0.11292	3.6	8999	59.6	63.2	48.2%
ftv38	1000	1512.8	1514.6	0.08486	7.1	2926	20.8	27.9	77.1%
ftv44	1000	1577.1	1586.0	0.10168	10.8	8217	82.5	93.3	66.5%
ftv44	5000	1584.7	1585.8	0.05515	52.4	2408	27.8	80.3	71.2%
ftv47	1000	1740.3	1750.5	0.09320	14.1	27997	347.9	362.1	42.3%
ftv47	5000	1746.9	1749.8	0.05170	70.6	12957	174.6	245.2	60.9%
ftv55	5000	1583.9	1585.6	0.04191	101.7	38867	940.5	1042.1	-0.6%
ftv55	10000	1583.9	1584.7	0.01919	205.7	20	16.8	222.5	78.5%
ftv64	5000	1807.3	1813.9	0.03699	161.8	18252	584.4	746.2	73.0%
ftv64	10000	1807.3	1810.6	0.01694	309.7	9876	349.9	659.5	76.1%
ftv70	5000	1908.4	1910.3	0.02956	198.0	36051	1215.8	1413.8	62.0%
ftv70	10000	1908.4	1909.6	0.01351	409.9	35	42.2	452.2	87.8%
p43	1582	5578.3	5633.8	0.00717	28.8	175194	6424.6	6453.4	-204.7%
p43	5000	5580.1	5625.1	0.00420	76.7	86710	3041.5	3118.3	-47.3%
ry48p	3328	14280.1	14422.9	0.00455	47.0	13181	187.7	234.8	60.9%
ry48p	5000	14280.1	14393.0	0.00351	66.3	6625	87.9	154.3	74.3%
ft53	5000	6904.0	6948.9	0.01414	82.5	1572	38.9	121.4	85.3%
ft53	7243	6904.0	6945.0	0.01000	127.2	74	14.7	141.9	82.8%
ft70	5000	38640.1	8831.2	0.03572	208.9	32930	983.4	1192.3	67.4%
ft70	10000	38640.1	38735.1	0.01637	432.3	236	46.3	478.6	86.9%

Table 2: Results of applying CPLEX after VA.

solution of the ftv55 instance is not accelerated much but, after 10000 iterations CPLEX required only 20 iterations (16.8 seconds) to get an optimal solution. Another example is the ftv70 instance. After 5000 VA iterations CPLEX time still is the large bulk to the total time. After 10000 iterations the gain in time is far greater, 87.8%.

By comparing the total times of Tables 1 and 2 - see Figure 2 - we observe an overall advantage in using the VA before invoking the dual simplex method of CPLEX. In some cases, ftv70 for example, the combined strategy is even superior to the interior point method. Nevertheless, there are instances, br17 and p43, for which the combined strategy did not work.

Since the role of the dualized constraints in this Lagrangian strategy is to avoid the subtours, we also claimed that the number of subtours in the Lagrangian subproblems would decrease as the volume algorithm stabilizes, becoming more and more similar to tours. This claim was not completely validated but, we observe a tendency. Figure 3 reports the number of subtours found in 1000 VA iterations for the ftv70 instance.

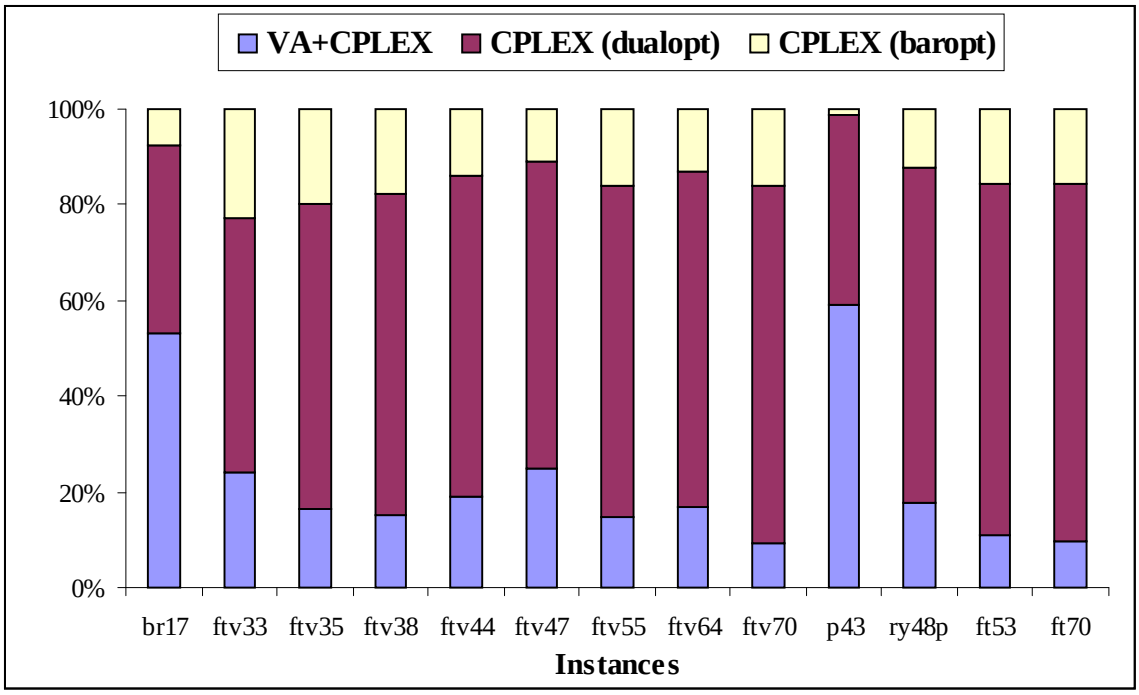

Figure 2: Comparing solution times.

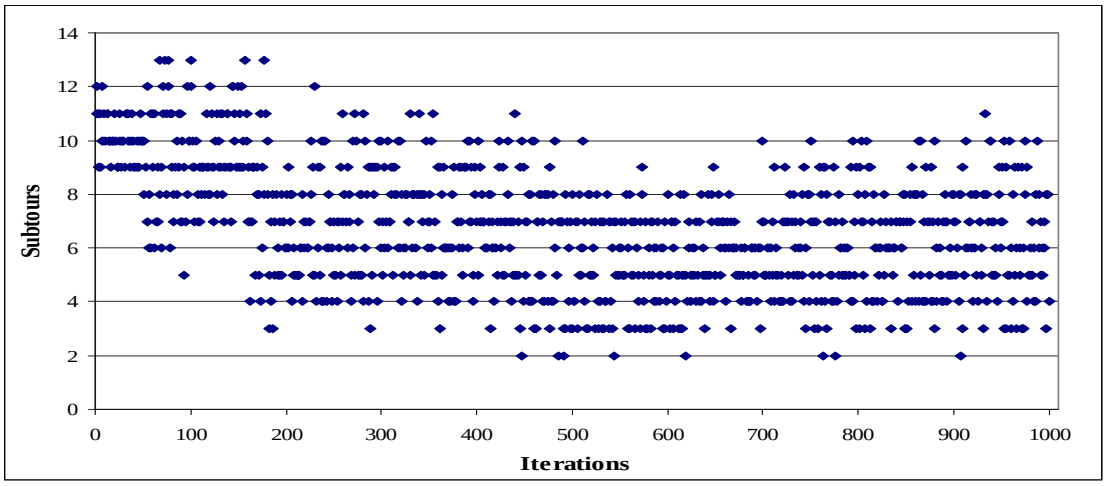

Figure 3: Number of subtours generated throughout VA for the ftv70 instance.

5.2 Variable fixing before branch-and-bound

In this subsection, we analyze the strategy of using the solution output from the volume algorithm to guess which variables are zero at an integral optimum. Table 3 presents the results of a straight call to Cplex MIP without providing any initial information for four of the small instances.

Table 4 reports some of the results obtained for a combined strategy of running the volume algorithm, decide which x variables to fix to zero and, then, invoke Cplex MIP. All the primal variables whose value is less or equal to 0.0005 and whose reduced cost

ATSP ID	Cplex MIP	
	Simplex It.	Time (sec.)
br17	27349	162.2
ftv33	24226	56.8
ftv35	288329	2094.9
ftv38	2403409	25536.7

Table 3: Results of Cplex MIP on some small instances.

is greater than or equal to 0.5 are fixed to zero. The first part of the table duplicates information presented before. The second part contains the performance of Cplex MIP. The column before last displays the total time spent (i.e., VA+ Cplex MIP). The last column displays the gain of time as compared to a straight call to Cplex MIP.

ATSP ID	**VOLUME**		Cplex MIP			**Total Time**	**Time Gain**
	VA It.	Time (sec.)	Final Solution	Simplex It.	Time (sec.)		
br17	500	0.7	39	48837	158.4	159.1	1.9%
br17	1000	1.4	39	35331	121.4	122.8	24.3%
ftv33	500	2.8	1286	2508	1.1	3.9	93.1%
ftv33	1000	5.5	1286	2879	1.4	6.8	88.0%
ftv35	500	3.2	1473	5028	27.9	31.1	98.5%
ftv35	1000	6.4	1473	4106	24.4	30.8	98.5%
ftv38	500	3.6	1530	5997	36.9	40.5	99.8%
ftv38	1000	7.1	1530	11662	64.0	71.2	99.7%

Table 4: Results of Cplex MIP after VA on a few small instances.

For the br17 instance the time spent with Cplex MIP is still large after 500 iterations of VA. However, an increase in the number of VA iterations allowed to fixing 1071 variables (out of 4624) lead to a time gain of 24.3%. This performance contrasts with the performance of the solving of the br17 fractional model. For the remaining instances, the time gains is in the order of 90%. For the ftv33 instance, after fixing 10 709 variables (out of 38148), an integer solution was found in 3.9s total time, instead of 56.8s. However, an increase in the number of VA iterations did not lead to an improvement in the total time. For the ftv35 instance, 500 VA iterations allowed the fixing of 5990 variables out of 45360 variables but, 1000 VA iterations allowed for the fixing of 6132 variables and the extra VA time was compensated by the reduction in Cplex MIP time. The ftv38 instance is a counterexample to this good behavior. 500 VA iterations allowed the fixing of 6972 variables, out of 57798, but 1000 VA iterations allowed the fixing of 6827 variables and the time required by Cplex MIP worsened considerably.

5.3 Heuristic integer solution finding

There are essentially two types of heuristic methods for the ATSP. Construction heuristics generate a solution adding individual components (nodes, arcs, variables, etc) step by step until a feasible solution is found. A simple example is the nearest neighbor heuristic which

works in the following way. One starts at a randomly chosen vertex and then, successively, connects each vertex to nearest unvisited vertex until a tour is formed. Often, this is not a suitable technique because the last connections tend to have rather large costs.

Improvement heuristics (also called local search and neighborhood search) depart from a feasible tour and successively generate neighboring tours through exchange rules. Updates occur when tours of better quality are found. A simple example is the 2-Opt heuristic which essentially builds a neighboring tour by taking a pair of arcs in a tour and exchange their endpoints. A generalization of this very simple principle forms the basis of the successful Lin-Kernighan heuristic [25].

In this subsection, we study the strategy of running an off-the-shelf heuristic method for the ATSP every once in a while during the course of the volume algorithm. The cost (c_{ij}) is changed to $(c_{ij} - (\bar{u}_i + \bar{v}_j))$, where $\bar{w} = (\bar{u}, \bar{v})$ are the optimal dual variables of the current assignment subproblem. The set of optimal solutions for the ATSP for the new cost remains the same but the behavior of the heuristic methods can be quite different. Each 100 iterations of the VA we run an heuristic method. The best upper bound is reported after 1000 or 5000 iterations of the VA.

Table 5 describes some additional test problems. The first eight problems are part of the *TSPLib* collection. We have also made some numerical experiments on some larger instances randomly generated through a code available from the *DIMACS Implementation Challenge* webpage [22]. These instances belong to the class of *Random Asymmetric Matrices (tmat)*. First, each arc distance c_{ij} is randomly chosen in $\{0, 1, \ldots, 10^6\}$. Then, whenever $c_{ij} > c_{ik} + c_{kj}$ we set $c_{ij} = c_{ik} + c_{kj}$. Repeat until no changes can be further made. These instances, the last eleven of Table 5, satisfy the triangular inequality.

We have tried four different heuristics, as they were implemented in the software *tsp_solve* [19] namely: *addition*, *assign*, *loss* and *patching*. The heuristic *addition* is similar to the nearest neighbor heuristic. A description is found in [21]. The heuristic *assign* first solves an assignment problem and then gathers the subtours into a single tour heuristically. The heuristic *loss* is an implementation of the method proposed in [10]. The heuristic *patching* first solves an assignment problem and then gathers the subtours into a single tour through operations of patching [23].

Table 6 reports the best upper bound found after 1000 or 5000 iterations of the volume algorithm. Each one of these heuristics is run every 100 iterations by using the reduced costs described before. For each instance we present the total number of VA iterations and for each heuristic we present the best upper bound found, as well as the iteration number where this upper bound was hit for the first time. The heuristic *assign* always hit its upper bound after 100 VA iterations of the VA.

The heuristic *assign* found the integer optimum in all cases, even when the original costs were used. This heuristic cycles in instance p43 and reports an error message because of the dimension in instances tmat180 until tmat200.

The heuristics *addition*, *loss* and *patching* required, in general, more than 100 VA iterations to find the best upper bound. In most cases, the heuristics *addition* and *patching* did not perform well.

Figure 4 maps the upper bounds found by the heuristic *addition* for ftv44 instance throughout the 5000 VA iterations. The best upper bound was found in iterations 3400 and 4800. At iteration zero, the upper bound found was 1829. In this case, the reduced costs helped in finding a better upper bound.

The Lin-Kernighan heuristic [25] is generally considered one of the most effective

ATSP			
ID	$\|V\|$	$\|E\|$	Int. Optimal
ftv100	101	10100	1788
ftv110	111	12210	1958
ftv120	121	14520	2166
ftv130	131	17030	2307
ftv140	141	19740	2420
ftv150	151	22650	2611
ftv160	161	25760	2683
ftv170	171	29070	2755
tmat100	100	9900	
tmat110	110	11990	
tmat120	120	14280	
tmat130	130	16770	
tmat140	140	19460	
tmat150	150	22350	
tmat160	160	25440	
tmat170	170	28730	
tmat180	180	32220	
tmat190	190	35910	
tmat200	200	39800	

Table 5: Instances of larger dimensions.

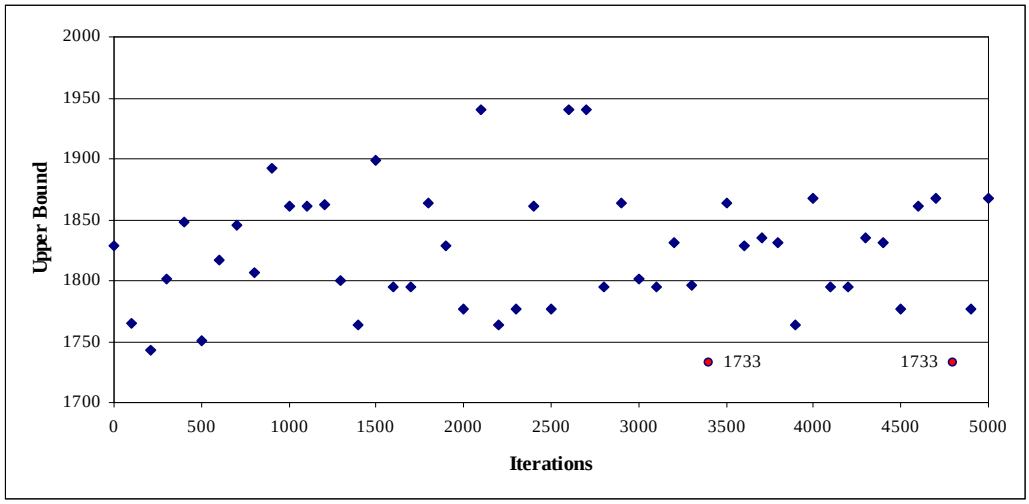

Figure 4: Upper Bounds found for ftv44 instance with *addition* heuristic.

methods to generate optimal or near-optimal solutions for the traveling salesman problem. It belongs to the class of local search heuristics. In this heuristic we search for adjacent tours to the current tour by performing λ-*opt* operations. A λ-*opt* operation corresponds to taking any combination of λ arcs and generate alternative paths between the endpoints to get a different tour. Whenever a better solution is generated, the current tour is updated

ATSP	VA	addition		assign	loss		patching	
ID	It.	UB	It.	UB	UB	It.	UB	It.
br17	1000	39	100	39	39	100	39	100
ftv33	1000	1482	200	1286	1372	600	1409	200
ftv35	1000	1491	100	1473	1508	300	1489	200
ftv38	1000	1634	200	1530	1547	700	1546	100
ftv44	5000	1733	3400	1613	1673	300	1699	100
ftv47	5000	1793	700	1776	1787	1600	1846	300
ftv55	5000	1781	3400	1608	1747	200	1657	100
ftv64	5000	2054	500	1839	1890	500	1871	900
ftv70	5000	2168	400	1950	2074	100	2004	600
ftv100	5000	2119	3000	1788	1969	900	1878	400
ftv110	5000	2335	800	1958	2156	4400	2036	500
ftv120	5000	2618	3000	2166	2402	100	2244	1100
ftv130	5000	2900	3300	2307	2519	100	2402	1000
ftv140	5000	2965	5000	2420	2590	300	2500	1200
ftv150	5000	3166	3400	2611	2971	200	2691	1500
ftv160	5000	3318	4200	2683	2839	100	2729	200
ftv170	5000	3509	4500	2755	2938	300	2809	600
kro124p	5000	40524	200	36230	41121	500	40106	100
p43	5000	5623	800		5638	1100	5640	200
ry48p	5000	14939	200	14422	15254	1000	14857	100
ft53	5000	8088	300	6905	7383	500	7847	100
ft70	5000	40566	500	38673	39065	1900	39197	100
tmat100	5000	1628486	1700	1377025	1379385	3500	1395601	500
tmat110	5000	1579919	200	1362004	1371800	700	1367459	100
tmat120	5000	1684696	2800	1390478	1396470	600	1419536	100
tmat130	5000	1716897	4500	1429864	1428864	3400	1444402	300
tmat140	5000	1719640	4800	1433486	1443556	300	1449585	100
tmat150	5000	1598899	3500	1364821	1364821	100	1389641	300
tmat160	5000	1855854	1300	1484992	1486164	2200	1487017	100
tmat170	5000	1674560	1600	1405055	1405273	4400	1405172	800
tmat180	5000	1941760	4400		1558217	500	1562008	100
tmat190	5000	1670791	500		1411658	1000	1424933	200
tmat200	5000	1968195	1400		1548893	400	1547919	100

Table 6: Solutions obtained with *tsp_solve* heuristics.

and the heuristic is halted when a better tour is not identifiable this way.

Keld Helsgaum describes in [18] the implementation of a modified version of Lin-Kernighan heuristic. The new algorithm differs in many details from the original and its implementation is publicly available. The new algorithm uses larger and more complex search steps. The numerical experiences reported in [18] and others show that its implementation is quite efficient and considerably improves upon the original algorithm.

We have embedded this implementation within the volume algorithm as explained before for the other heuristics. Table 7 reports the best upper bound found on the

instances from *TSPLib*. In all these problems, the best upper bound was found in the iteration 100 except for ftv150, ftv170 and p43, for which the best upper bound was found in iterations 400, 500 and 400, respectively.

ATSP ID	UB
br17	39
ftv33	1286
ftv35	1473
ftv38	1530
ftv44	1613
ftv47	1776
ftv55	1608
ftv64	1839
ftv70	1950
kro124p	36230
p43	5620
ry48p	14422
ft53	6905
ft70	38673

ATSP ID	UB
ftv100	1788
ftv110	1958
ftv120	2166
ftv130	2307
ftv140	2420
ftv150	2611
ftv160	2683
ftv170	2755

ATSP ID	UB
tmat100	1377025
tmat110	1362004
tmat120	1390478
tmat130	1429864
tmat140	1433486
tmat150	1364821
tmat160	1484992
tmat170	1405055
tmat180	1554371
tmat190	1411658
tmat200	1545211

Table 7: Performance of the Lin-Kernighan heuristic, as implemented by K. Helsgaum.

It should be said that this implementation of the Lin-Kernighan heuristic also found the integer optimum for all these instances but for p43. Here, the original costs made the heuristic find a tour of value 5621, which is worse than the heuristic solution found by the same algorithm after 400 VA iterations.

6 Conclusions

Our study showed that the volume algorithm is a viable mechanism of deriving a good initial basis for the solving of a polynomial formulation of the ATSP like the extended disaggregated flow formulation by simplex algorithms. In a few cases our combined strategy is even faster than the interior point method. However, special caution is required on the volume algorithm stopping criterion. Halting too soon may be more damaging than beneficial.

Relatively to the usage of the reduced costs generated by the volume algorithm, we do not see as much an impact especially because Helsgaum implementation of Lin-Kernighan heuristic is so efficient. We have also observed that the volume algorithm can be quite effective in the identification of irrelevant arcs but our results are scarce.

Overall, we think that this combined strategy is viable and deserves to be further exploited. Of course our numerical experiments were limited and further testing is required especially on even larger models. A direction for future research in the context of the ATSP is, for example, to test the usage of the solution output from the volume algorithm as a means to speeding the solving of the fractional DFJ model (that has an exponential number of constraints) by a constraint generation algorithm. In the context

of other models, we think similar conclusions can be made when the reoptimization of the Lagrangian subproblems can be performed efficiently.

References

[1] R. Anbil, J. J. Forrest, and W. R. Pulleyblank. Column generation and the airline crew pairing problem. *Documenta Mathematica*, Extra Volume ICM III:677–686, 1998.

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. On the solution of traveling salesman problems. In *Proceedings of the International Congress of Mathematicians, Vol. III (Berlin, 1998)*, number Extra Vol. III, pages 645–656 (electronic), 1998.

[3] L. Bahiense, F. Barahona, and O. Porto. Solving steiner tree problems in graphs with lagrangian relaxation. *Journal of Combinatorial Optimization*, 3(7):259–282, 2003.

[4] L. Bahiense, N. Maculan, and C. Sagastizábal. The volume algorithm revisited: Relation with bundle methods. *Mathematical Programming*, 94:41–70, 2002.

[5] F. Barahona and R. Anbil. The volume algorithm: Producing primal solutions with a subgradient method. *Mathematical Programming*, 87:385–399, 2000.

[6] F. Barahona and R. Anbil. On some difficult linear programs coming from set partitioning. *Discrete Applied Mathematics*, 118(1-2):3–11, 2002.

[7] F. Barahona and F. A. Chudak. Near-optimal solutions to large scale facility location problems. Technical report, IBM Watson Research Center, 1999.

[8] F. Barahona and L. Ladanyi. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. Technical report, IBM Watson Research Center, 2001.

[9] A. Claus. A new formulation for the travelling salesman problem. *SIAM Journal of Algebraic and Discrete Methods*, 5:21–25, 1984.

[10] P. V. D. Cruyssen and M. Rijckaert. Heuristic for the asymmetric travelling salesman problem. *Journal of the Operational Research Society*, 29(7):697–701, 1978.

[11] G. Dantzig, D. Fulkerson, and S. Johnson. Solution of a large-scale traveling salesman problem. *Operations Research*, 2:393–410, 1954.

[12] M. Fischetti and P. Toth. A polyhedral approach to the asymmetric traveling salesman problem. *Management Science*, 43(11):1520–1536, 1997.

[13] K. Fox, B. Gavish, and S. Graves. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 28:1018–1021, 1980.

[14] L. Gouveia and J. M. Pires. The asymmetric travelling salesman problem: on generalizations of disaggregated Miller-Tucker-Zemlin constraints. *Discrete Applied Mathematics*, 112(1-3):129–145, 2001. Combinatorial Optimization Symposium (Brussels, 1998).

[15] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[16] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.

[17] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–68, 1974.

[18] K. Helsgaum. An effective implementation of the lin-kernighan traveling salesman problem. *European Journal of Operations Research*, 126:106–130, 2000. Code available at http://www.dat.ruc.dk/~keld/.

[19] C. Hurwitz and R. Craig. *GNU Tsp_solve*. Version 1.3.8, 1994.

[20] ILOG. Cplex *7.0*. ILOG, 2000.

[21] D. Johnson and C. Papadimitriou. Performance guarantees for heuristics. In E. Lawler, J. Lenstra, A. R. Kan, and D. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 145–180. John Wiley & Sons, Chichester, 1985.

[22] D. Jonhson, L. McGeoch, F. Glover, and C. Rego. Website for the DIMACS implementation challenge on the traveling salesman problem. http://www.research.att.com/~dsj/chtsp.

[23] R. Karp and J. Steele. Probabilistic analysis of heuristics. In E. Lawler, J. Lenstra, A. R. Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, pages 181–205. John Wiley & Sons, Chichester, 1985.

[24] A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9:127–132, 1990.

[25] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[26] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of ACM*, 7:326–329, 1960.

[27] M. Padberg and T. Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52:315–357, 1991.

[28] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991. Available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.