Ann Oper Res (2006) 144: 83–97
DOI 10.1007/s10479-006-0014-y

Solving the asymmetric traveling purchaser problem*

Jorge Riera-Ledesma · Juan-José Salazar-González

Published online: 24 May 2006

Abstract The Asymmetric Traveling Purchaser Problem (ATPP) is a generalization of the Asymmetric Traveling Salesman Problem with several applications in the routing and the scheduling contexts. This problem is defined as follows. Let us consider a set of products and a set of markets. Each market is provided with a limited amount of each product at a known price. The ATPP consists in selecting a subset of markets such that a given demand of each product can be purchased, minimizing the routing cost and the purchasing cost. The aim of this article is to evaluate the effectiveness of a branch-and-cut algorithm based on new valid inequalities. It also proposes a transformation of the ATPP into its symmetric version, so a second exact method is also presented. An extensive computational analysis on several classes of instances from literature evaluates the proposed approaches. A previous work () solves instances with up to 25 markets and 100 products, while the here-presented approaches prove optimality on instances with up to 200 markets and 200 products.

Keywords Traveling purchaser problem · Traveling salesman problem · Branch-and-cut · Heuristics

This article concerns a generalization of the well-known *Asymmetric Traveling Salesman Problem* (ATSP). See, e.g., Fischetti, Lodi and Toth (2002) for recent references on the ATSP. The generalization is known as *Asymmetric Traveling Purchaser Problem* (ATPP) and is defined as follows. Let us consider a *depot* v_0, a set of *markets* $M := \{v_1, \ldots, v_n\}$, and a set of *products* $K := \{p_1, \ldots, p_m\}$. We will assume $n \geq 4$ and $m \geq 2$ for simplicity. Let $G = (V, A)$ be a directed graph where $V := \{v_0\} \cup M$ is the vertex set and $A := \{(v_i, v_j) : v_i, v_j \in V\}$ is the arc set. A purchaser originally in v_0 must select and visit a subset of markets to buy a given amount of each product in K. Each product p_k can be purchased at a subset

*Partially supported by "Ministerio de Ciencia y Tecnología" (TIC2003-05982-C05-02), and by Vicerrectorado de Investigación y Desarrollo Tecnológico de la Universidad de La Laguna.

J. Riera-Ledesma (✉)· J.-J. Salazar-González
Departamento de Estadística, Investigación Operativa y Computación,
Universidad de La Laguna, 38271 La Laguna, Spain
e-mail: jriera@ull.es

M_k of markets. Let d_k be the units of the product p_k that must be purchased, and let q_{ki} be the units of the product p_k that are available at market $v_i \in M_k$. We assume that q_{ki} and d_k satisfy $0 < q_{ki} \leq d_k$ and $\sum_{v_j \in M_k} q_{kj} \geq d_k$ for all $p_k \in K$ and $v_i \in M_k$. Let us denote the price of a unit of product p_k at market v_i by b_{ki}, and the travel cost of the arc $a = (v_i, v_j)$ (i.e., the cost of traveling from market v_i to market v_j) by c_a. The ATPP consists in determining a simple directed cycle (*circuit*) in G passing through the depot and a subset of markets so that all products are purchased at a minimum total cost obtained by adding the routing cost and the purchasing price.

This definition of the ATPP generalizes the standard version where the supplies are assumed to be unlimited, i.e., $q_{ki} = d_k$ for all i, k. This particular case, in which there is no restricted offer of a product at each market, is called *unrestricted ATPP*. It is based on the assumption that each product p_k can be totally purchased in each market $v_i \in M_k$, and can be seen as the ATPP with one-unit demand for each product.

The ATPP was introduced by Burstall (1966) and by Buzacott and Dutta (1971) in the scheduling context. Indeed, each product corresponds to a task to be performed by a flexible machine, and each market corresponds to a potential state of the machine. The offer q_{ki} is the resource available to perform task p_k when the machine is in state v_i, the cost b_{ki} is the time consumed to perform a unit of task p_k using state v_i, and c_a with $a = (v_i, v_j)$ is the changeover time from state v_i to state v_j. State v_0 is the initial and final state of the machine.

The unrestricted ATPP is $\mathcal{NP}$-hard in the strong sense since it is the TSP when each product can be purchased in exactly one market. Most of the previous works assume that the cost of traveling from a market v_i to a market v_j coincides with the cost of traveling from market v_j to market v_i, for all i and j. This version is known as *Symmetric Traveling Purchaser Problem* (STPP). Ramesh (1981) described an exact method based on a lexicographical search capable of handling symmetric instances with $n \leq 12$ and $m \leq 10$. Singh and van Oudheusden (1997) presented a branch-and-bound algorithm solving ATPP instances with $n \in \{10, 15, 20, 25\}$ and $m \in \{10, 30, 50, 100\}$, and STPP instances with $n \in \{10, 15, 20\}$ and $m \in \{15, 30, 50\}$. The bound in Singh and van Oudheusden (1997) is based on the *Uncapacitated Facility Location Problem* (UFLP), arising when the sequence requirement is relaxed. Indeed, the UFLP is also a particular case of the unrestricted ATPP by associating plants to markets, customers to products, and by setting $c_a = (f_i + f_j)/2$ for a being the arc from v_i to v_j and f_i being the cost of opening plant v_i. Laporte, Riera-Ledesma and Salazar-González (2003) deals with the STPP and presents a branch-and-cut algorithm for the exact solution of instances involving up to 200 markets and 200 products.

The literature is mostly directed toward the development of heuristic or near optimal methods for solving the STPP. One of these heuristic procedures was introduced by Golden, Levy and Dahl (1981) and named the *Generalized Saving Heuristic* (GSH). Their heuristic was, later on, modified by Ong (1982) who proposed the *Tour Reduction Heuristic*. Pearn and Chien (1998) suggested some improvements to the two previous works of Golden, Levy and Dahl (1981) and Ong (1982). Two of these improvements were related to the GSH of Golden, Levy and Dahl (1981). Another heuristic proposed by Pearn and Chien (1998) is called *Commodity Adding Heuristic*. The first metaheuristic approaches for the STPP based on dynamic tabu search and simulated annealing are presented in Voß (1996). That article proposes two dynamic strategies for the managing of the tabu list (the *Reverse Elimination Method* and the *Cancelation Sequence Method*), and their impact on the STPP is also studied. Other metaheuristic approaches for the symmetric version have been recently presented by Boctor, Laporte and Renaud (2003) and by Riera-Ledesma and Salazar-González (2005). The benchmark STPP instances have been created for both the unrestricted and the general versions, and for the particular case in which the markets are locations in the Euclidean plane.

These algorithms are tested on instances up to $m \leq 200$ and $n \leq 200$. To our knowledge, a similar extensive analysis of algorithms has not been conducted on asymmetric instances, even if they were the original motivation to study the problem (see Burstall, 1966).

This article is organized as follows. Section 1 presents an Integer Linear Programming (ILP) model to formulate the ATPP, and introduces new valid inequalities to strengthen its linear relaxation. Section 2 describes a branch-and-cut approach for the exact solution of ATPP instances, empathizing some procedures to find violated valid inequalities when required. Section 3 shows an alternative approach to find optimal solutions of ATPP instances using an available exact algorithm for solving the symmetrical version. Extensive computational experiments are analysed in Section 4 to compare the two approaches. In our experiments, both approaches were able to prove optimality solving unrestricted ATPP instances with up to 200 markets and 200 products, and restricted ATPP instances with up to 100 markets and 200 products.

1. Mathematical model

We present in this section an Integer Linear Programming (ILP) formulation for the ATPP, and some valid inequalities strengthening the continuous linear programming relaxation. All the elements are used in the exact algorithm describe in Section 2.

1.1. ILP formulation for the ATPP

To formulate the problem, we first need some notation. For $S \subset V$, let us denote $A(S) := \{(v_i, v_j) \in A : v_i, v_j \in S\}$, $\delta^+(S) := \{(v_i, v_j) \in A : v_i \in S, v_j \in V \setminus S\}$ and $\delta^-(S) := \{(v_i, v_j) \in A : v_i \in V \setminus S, v_j \in S\}$. Also let us define

$$M^* := \{v_0\} \cup \left\{ v_i \in M : \text{ there exists } p_k \in K \text{ such that } \sum_{v_j \in M_k \setminus \{v_i\}} q_{kj} < d_k \right\},$$

the set of vertices that must necessarily be part of any feasible ATPP circuit, and

$$K^* := \left\{ p_k \in K : \sum_{v_i \in M_k} q_{ki} = d_k \right\},$$

the set of products without market decision options.

For a mathematical formulation we use three types of decision variables:

$$x_a := \begin{cases} 1 & \text{if arc } a \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } a \in A;$$

$$y_i := \begin{cases} 1 & \text{if vertex } v_i \text{ belongs to the solution} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } v_i \in V;$$

$z_{ki} :=$ the amount of product p_k purchased at market v_i,

for all $p_k \in K$ and all $v_i \in M_k$.

The ATPP formulation is now the following:

$$w^{OPT} := \min \sum_{a \in A} c_a x_a + \sum_{p_k \in K} \sum_{v_i \in M_k} b_{ki} z_{ki} \tag{1}$$

subject to

$$\sum_{a \in \delta^+(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V \tag{2}$$

$$\sum_{a \in \delta^-(\{v_i\})} x_a = y_i \qquad \text{for all } v_i \in V \tag{3}$$

$$\sum_{a \in \delta^+(S)} x_a \geq y_i \qquad \text{for all } S \subset M \text{ and all } v_i \in S \tag{4}$$

$$\sum_{v_i \in M_k} z_{ki} = d_k \qquad \text{for all } p_k \in K \tag{5}$$

$$z_{ki} \leq q_{ki} y_i \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k \tag{6}$$

$$x_a \in \{0, 1\} \qquad \text{for all } a \in A \tag{7}$$

$$y_i \in \{0, 1\} \qquad \text{for all } v_i \in M \setminus M^* \tag{8}$$

$$y_i = 1 \qquad \text{for all } v_i \in M^* \tag{9}$$

$$z_{ki} \geq 0 \qquad \text{for all } p_k \in K \text{ and all } v_i \in M_k. \tag{10}$$

Constraints (2) and (3) are the assignment equations, and impose that the in-degree and out-degree, respectively, of each visited vertex must be equal to one. Constraints (4), called *YSEC*$^+$, impose the strong connectivity on the route. Because of (2) and (3), inequalities (4) can be equivalently re-written as

$$\sum_{a \in A(S)} x_a \leq \sum_{v_j \in S \setminus \{v_i\}} y_j \qquad \text{for all } S \subset M \text{ and all } v_i \in S.$$

These constraints ensure that a visited market in a subset $S \subseteq M$ must be connected to the depot through a path. By also using Eqs. (2) and (3), these constrains are also equivalent to:

$$\sum_{a \in \delta^-(S)} x_a \geq y_i \qquad \text{for all } S \subset M \text{ and all } v_i \in S.$$

Inequalities (5) guarantee that the exact amount of product p_k is purchased. Inequalities (6) mean that it is not possible to purchase a product p_k in a market v_i if it is not visited, and that it is not possible to purchase more than its offer q_{ki} if it is visited. Constraints (7)–(10) impose bounds and integrality conditions on the variables.

1.2. Strengthening the linear relaxation

When using model (1)–(10) in a cutting-plane approach for the exact solution of the ATPP, it is observed that the basic linear relaxation needs additional valid inequalities to close the integrability gap. To this end this section shows some considerations leading to a tighter linear relaxation.

A first observation is based on the fact that constraints (4) can be trivially strengthened if there is a product p_k that cannot be totally purchased outside markets in S. More precisely, if S is a subset of markets such that $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ for a product $p_k \in K$, then the inequality:

$$\sum_{a \in \delta^+(S)} x_a \geq 1 \tag{11}$$

is valid for all ATTP solutions. This inequality imposes the requirement that the purchaser must necessarily visit a market in S.

A second observation is that each solution of the ATPP is a simple circuit in a directed graph, hence all valid inequalities presented in Balas and Oosten (2000) are also valid inequalities for ATPP solutions. To illustrate the idea, we restrict our attention adapting to the so-called D_k^+ inequalities.

Let us consider an ordered subset $S = \{v_{i_1}, \ldots, v_{i_l}\} \subset M$ $(3 \leq l \leq n)$. Then

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq \sum_{h=1}^{l} y_{i_h} - y_{i_2} \tag{12}$$

is a valid inequality for ATPP solutions. See Balas and Oosten (2000) and observe that $v_0 \notin S$, thus a feasible ATPP circuit could be totally outside S but not totally inside S. These inequalities were proposed by Grötschel and Padberg (1985) for the Asymmetric TSP, and a separation algorithm for the ATSP was proposed by Fischetti and Toth (1997).

In the particular case that there is a product p_k that cannot be totally purchased outside markets in S, then constraints (12) can be trivially strengthened. More precisely, if S is a subset of markets such that $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ for a product $p_k \in K$, then the inequality:

$$x_{i_l i_1} + \sum_{h=1}^{l-1} x_{i_h i_{h+1}} + 2\sum_{h=3}^{l} x_{i_1 i_h} + \sum_{j=4}^{l}\sum_{h=3}^{j-1} x_{i_j i_h} \leq \sum_{h=1}^{l} y_{i_h} - 1 \tag{13}$$

is a valid inequality for ATPP solutions.

As observed by Singh and van Oudheusden (1997), the subproblem defined by (5), (6), (8) and (10) corresponds to a generalization of the UFLP with upper bounds on the customer-facility variables. Valid inequalities for this subproblem when $q_{ki} = d_k$ can be obtained from the *Set Covering Problem* polytope (see e.g., Balas and Ng, 1989). To illustrate the idea, let us consider a subset of products $L \subseteq K$ with $3 \leq |L| \leq |K| - 1$. Define $M'(L) := \cap_{p_k \in L} M_k$ the set of markets each one selling all products in L and $M''(L) := \cup_{p_k \in L} M_k$ the set of markets each one selling at least one products in L. We can then impose the *cover inequality*

$$2\sum_{v_i \in M'(L)} y_i + \sum_{v_i \in M''(L) \setminus M'(L)} y_i \geq 2 \tag{14}$$

which stipulates that at least two markets in $M''(L)$ must be visited if no market in $M'(L)$ is visited. On our computational experiment we did not find any advantage of considering these inequalities.

The previous ideas have exploited in a separated way two relaxations of the ATPP: on one side, the routing part; on another side, the set covering part. It is also possible to derive additional valid inequalities by combining both relaxations. Indeed, the constraints

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \frac{1}{d_k} \sum_{v_i\in S\cap M_k} z_{ki} \quad \text{for all } S \subseteq M \text{ and } p_k \in K \tag{15}$$

state that at least two edges must be incident to a subset S whenever some amount of any product p_k is purchased in a market of $S \cap M_k$. Clearly, if $\sum_{v_i\in M_k\setminus S} q_{ki} < d_k$ for some product p_k, then constraints (15) are dominated by constraints (11). Again, constraints (15) can be strengthened by

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \frac{\sum_{v_i\in S\cap M_k} z_{ki}}{\min\left\{d_k, \sum_{v_i\in S\cap M_k} q_{ki}\right\}} \quad \text{for all } S \subseteq M \text{ and } p_k \in K. \tag{16}$$

These constraints are named *ZSEC*$^+$. Constraints (16) coincide with inequalities (6) when $S = \{v_i\}$.

Another procedure to generate additional valid inequalities is inspired by the concept of *symmetric inequalities* for the ATSP (see Fischetti and Toth, 1997; Fischetti, Lodi and Toth, 2002). An inequality $\alpha x + \beta y + \gamma z \leq \alpha_0$ is called *symmetric* when $\alpha_{ij} = \alpha_{ji}$ for all $(v_i, v_j) \in A$. Symmetric inequalities valid for the ATPP can be easily derived from valid inequalities for the STPP. To this end, let us consider an undirected graph $\bar{G} = (V, E)$ where there is an edge $e = [v_i, v_j] \in E$ if $(v_i, v_j) \in A$ or $(v_j, v_i) \in A$, and let us associate each edge $e = [v_i, v_j] \in E$ with a variable $\bar{x}_e$ by setting

$$\bar{x}_{[v_i,v_j]} := x_{(v_i,v_j)} + x_{(v_j,v_i)}.$$

Then, a symmetric valid inequality for ATPP can be trivially derived from each inequality $\sum_{[v_i,v_j]\in E} \alpha_{ij}\bar{x}_{ij} + \beta y + \gamma z \leq \alpha_0$ valid for the STPP (see Laporte, Riera-Ledesma and Salazar-González, 2003) by replacing the x variables. In some cases the obtained symmetric inequality is dominated by another ATPP inequality but, as it is shown in Section 4, the general idea of considering this procedure is helpful to strengthen the linear relaxation in practice.

Some of the presented inequalities define facets of the ATPP polytope under certain conditions. The basic idea for the proofs are based on the fact that the ATPP polytope is a face of the ATSP polytope, and the results can be obtained by using the classical sequential lifting introduced by Padberg (1975) in a similar way done in Laporte, Riera-Ledesma and Salazar-González (2003) for the STPP. See Riera-Ledesma (2002) for details.

2. Algorithm

The ILP model and the valid inequalities presented in the previous section can be used to derive a branch-and-bound algorithm to find optimal ATPP solutions, where the lower bound is obtained by solving the linear relaxation iteratively strengthened with the additional constraints. The whole procedure is known as *branch-and-cut*, and it is based on an effective management of the constraints (see, e.g., Jünger, Reinelt and Rinaldi, 1995 for details). This section describes how some inequalities can be generated at each iteration to strengthen the linear relaxation. The idea is to find violated constraints by the optimal solution (x^*, y^*, z^*) of the current linear relaxation, and it is known as *separation problem*. Other classical ingredients (such as *initial* and *primal heuristics*, *branching phase*, *pricing scheme*, etc.) must be also implemented to produce an effective branch-and-cut code. In our implementation, these ingredients are trivial adaptations of the proposals of previous authors for similar routing problems, thus we do not give here details. For example, the primal heuristic builds a feasible ATPP circuit by iteratively selecting arcs a with the larger values of x_a^*; the circuit is completed by using a nearest neighborhood rule, and enlarged with the maximum-saving criterion markets to guarantee feasibility; a cleaning procedure tries to remove each market at a time to reduce the total cost; the heuristic ends with a 3-optimality scheme on the arcs in the circuit.

2.1. Separating the YSEC$^+$ inequalities (4) and (11)

The separation problem of inequalities (11) appears when solving the classical ATSP, and the separation of inequalities (4) when solving the Circuit Problem. Hence an efficient algorithm for separating these inequalities is already known in literature (see, e.g., Fischetti, Lodi and Toth, 2002). The idea consists in considering a network $G^* = (V, A)$ where each arc $a \in A$ is associated with a capacity x_a^*. Then, for each $v_i \in M$ with $y_i^* > 0$, a most violated YSEC$^+$ constraint (4) corresponds to a minimum-capacity directed cut $(S, V^* \setminus S)$ with $v_i \in S$ and $v_0 \notin S$. Therefore, the separation problem can be solved by applying several max-flow computations. Whenever $\sum_{v_i \in M_k \setminus S} q_{ki} < d_k$ for a product $p_k \in K$ then the stronger inequality (11) is generated instead of the dominated inequality (4). Since the separation procedure tends to produce different subsets S, we shrink each subset S so the final family of subsets is a nested family. This idea has several practical advantages also notice on other routing problems (see, e.g., Fischetti, Salazar and Toth, 1997).

2.2. Separating the D_k^+ inequalities (12) and (13)

A heuristic procedure to solve the separation problem of the D_k^+ inequalities can be done by performing partial enumeration search on the support graph induced by the current fractional solution. This idea, in a clever way, has been successfully implemented by Fischetti, Lodi and Toth (2002) for the ATSP. The basic idea is to develop a greedy mechanism to select and sequence a subset of markets $v_{i_1}, \ldots, v_{i_l}$, $3 \leq l \leq n$ maximizing

$$x_{i_l i_1}^* + \sum_{h=1}^{l-1} x_{i_h i_{h+1}}^* + 2 \sum_{h=3}^{l} x_{i_1 i_h}^* + \sum_{j=4}^{l} \sum_{h=3}^{j-1} x_{i_j i_h}^* - \sum_{h=1}^{l} y_{i_h}^*.$$

Indeed, on the network G^* previously defined, we start by selecting the arcs with biggest capacity, and whenever a circuit is created among the selected arcs, an enumerative search looks for a violated constraint in (12) and (13). As it is shown in Section 4, this simple approach did not required a high computational effort but was successful in generating some violated constraints in our computational experiments.

2.3. Separating the ZSEC$^+$ inequalities (15) and (16)

The separation problem of constraints (15) can be also solved in a similar way to the separation problem of the YSEC$^+$ constraints. Indeed, observe that by using Eqs. (5) finding a set S defining a most violated inequality in (15) corresponds to finding a set S with minimum value of:

$$\sum_{a\in\delta^+(S)} x_a^* + \sum_{v_i\in M_k\setminus S} \frac{z_{ki}^*}{d_k}.$$

Therefore, the separation problem of (15) for each product $p_k \in K$ is equivalent to compute a max-flow on a network $G' = (V', A')$ defined as follows. The vertex set V' contains the depot v_0, the market set M and a dummy vertex v_{n+1}. The arc set A' contains all the arcs $a \in A$ with capacity x_a^*, and all arcs (v_i, v_{n+1}) with capacity z_{ki}^*/d_k. Then, the max-flow must go from the source v_0 to the destination v_{n+1}. If the capacity of the max-flow is bigger than or equal to 1, then all inequalities in (15) hold; otherwise, the minimum-capacity cut defines a most violated inequality in (15). Whenever $\sum_{v_i\in S\cap M_k} q_{ki} < d_k$ for a product $p_k \in K$ then the stronger inequality (16) is generated instead of the dominated inequality (15).

This procedure solves exactly the separation problem of inequalities (15) and is a heuristic approach for the separation problem of inequalities (16). Indeed, the procedure can end with the proof that no inequality in (15) exits, but still a violated constraint in (16) could exits. To improve this basic heuristic idea we have also implemented a procedure to check for violation subsets S' generated by inserting and removing a market from a previous generated subset S.

2.4. Separating symmetric constraints for the ATPP

If no one of the above separation procedures generate a violated inequality, then we remove the orientation of each arc in the current solution and construct an undirected fractional solution. This fractional solution is given as input to the separation procedures described in Laporte, Riera-Ledesma and Salazar-González (2003). If a violated inequality is generated, then a violated symmetric constraint is constructed and considered to strengthen the current linear relaxation. Among the several inequalities we consider the following *2-matching constraints*:

$$\sum_{e\in T} x_e - \sum_{e\in\delta(H)\setminus T} x_e \le |T| - 1 \tag{17}$$

for all $H \subset V$ and T being an odd set of disjoint edges in $\delta(H)$, $|T| \ge 3$.

3. Transformation of the ATPP into the STPP

The previous section proposed a direct approach to solve the ATPP. We now introduce an alternative proposal to find an optimal solution of an ATPP instance when an exact algorithm for solving STPP instances is available. The aim is to transform the asymmetric instance into a symmetric one in the spirit of similar works done for similar routing problems. In particular, inspired by the *3-node transformation* of Karp (1972) and the *2-node transformation* of Jonker and Volgenant (1983), both for the ATSP, we next propose a transformation for the ATPP into the STPP.

If the original ATPP instance is given on the directed graph $G = (V, A)$, we built a new STPP instance on the following undirected graph $G''' = (V''', E''')$. The vertex set V''' contains three copies of each vertex in V. We denote by v_i', v_i'', v_i''' the three vertices in V''' associated to each v_i in V. Let N a large enough number. For each vertex $v_i \in V$, we consider the edge $[v_i', v_i'']$ in E''' with cost equal to $-N$, and the edge $[v_i'', v_i''']$ in E''' with cost equal to 0. Associated to each arc $a = (v_i, v_j) \in A$, we consider the edge in $[v_i''', v_j'] \in E'''$ with cost $c_a + N$. The remaining edges in E''' have cost equal to ∞. For each vertex $v_i \in V$, the vertex $v_i'' \in V'''$ represents the market selling the same products of v_i at the same prices, while v_i' and v_i'' do not sell any product. Then, a minimum-cost cycle in G''' solving the STPP corresponds to a minimum-cost circuit in G solving the ATPP, and viceversa. Indeed, an optimal cycle cannot use two consecutive edges with negative cost and it will alternate positive and negative costs, so the value of N will not affect the total cost. This alternation guarantees that the arcs associated to the used edges define a circuit in G. Trivially, each pair of vertices $\{v_i'', v_i'''\}$ can be shrunk into a single vertex, so the previous transformation involves to solve a STPP in a graph with only a double number of markets.

4. Computational results

The here-proposed approaches have been implemented in C++ on a PC AMD 1333 MHz. ABACUS 2.2 linked with CPLEX 6.0 has been used as a framework (see Jünger and Thienel (1998) for details on this software). A time limit of two hours has been established for the running time of our algorithms.

To test the performances of our code, we have considered ATPP instances obtained by using the random generator described in Singh and van Oudheusden (1997), since this is the only today's article in which algorithms for the ATPP are tested. It is a generator of unrestricted ATPP instances in which the routing costs c_a are randomly generated in $[1, \tau]$, where τ is generated in [15,140]. Each market sells a number of products randomly generated in $[1, m]$, where $m = |K|$ is the number of products. Purchasing costs are randomly generated in $[0, \omega]$ where ω is generated in [5,75] for each market. We have defined instances with $|V| \in \{50, 100, 150, 200\}$ and $|K| \in \{50, 100, 150, 200\}$. For each type we have generated five instances by considering different seeds, thus our benchmark library contains 80 unrestricted ATPP instances.

In order to consider ATPP instances with restricted offers, the generator of Singh and van Oudheusden has been extended in the following way. For each product p_k and each market v_i, q_{ki} has been randomly generated in [1,15] and $d_k := \lceil \lambda \max_{v_i \in M_k} q_{ki} + (1 - \lambda) \sum_{v_i \in M_k} q_{ki} \rceil$ for $\lambda \in \{0.5, 0.8, 0.9, 0.95, 0.99\}$. Observe that parameter λ controls the demand of each product, and therefore it affects the number of visited markets in an optimal ATPP circuit: the smaller λ is, the bigger is the number of visited markets.

Tables 1–3 show statistical results from our computational experiments testing the branch-and-cut algorithm for the ATPP. The heading columns have the following meaning:

$|V|$*:* number of vertices (i.e., $n + 1$);
$|K|$*:* number of products (i.e., m);
λ*:* value of the parameter λ in the generation (only for restricted ATPP instances);
Solved: number of instances solved before the time limit (over 5 trials);
#: average number of vertices in the optimal solutions;
2sec: average number of constraints (11) separated;

Table 1 Solving unrestricted ATPP instances with the branch-and-cut algorithm

$\|V\|$	$\|K\|$	#	*Solved*	*2sec*	*ysec*	*zsec*	*2mat*	D_k^+	*Nodes*	*%UB*	*%LB*	*Root-t*	*Total-t*
50	50	9.2	5	506.8	60.6	848.4	8.0	8.6	13.8	0.47	0.33	1.0	5.6
	100	12.4	5	586.8	110.0	1357.2	6.4	2.8	17.0	0.58	0.20	1.6	8.8
	150	14.0	5	827.8	152.8	2345.2	5.4	4.0	19.4	0.38	0.12	1.8	15.8
	200	16.0	5	1329.4	220.4	3925.0	6.6	4.4	25.8	0.33	0.15	1.8	25.6
100	50	6.0	5	223.8	44.0	1062.0	5.2	2.8	7.8	0.12	0.17	17.0	36.8
	100	12.0	5	9378.2	1124.2	12619.4	32.8	18.6	76.2	0.79	0.47	21.2	411.0
	150	14.8	5	31966.6	4024.8	55081.8	124.2	43.2	265.0	0.76	0.48	25.4	1646.4
	200	17.2	5	44964.0	6431.6	97084.0	246.0	62.4	329.4	0.50	0.38	24.2	2237.6
150	50	7.6	5	5343.2	598.6	6544.2	20.8	28.2	49.4	0.90	0.66	87.2	812.2
	100	10.6	5	17261.4	1958.4	26797.4	34.4	17.8	93.4	0.77	0.57	99.0	2302.8
	150	14.4	5	13148.4	2301.2	31251.8	77.2	22.0	147.4	0.74	0.44	69.0	2247.4
	200	15.8	1	32748.6	4261.0	78140.4	132.0	31.4	300.2	0.68	0.68	100.4	1428.0
200	50	7.8	5	4186.6	326.8	6694.2	19.0	16.8	32.2	0.57	0.59	277.8	1484.8
	100	10.2	3	15039.2	2374.4	30172.0	64.2	25.8	117.4	0.51	0.61	172.6	3037.7
	150	13.2	1	25074.2	2315.8	41014.2	86.8	20.8	126.6	0.64	0.65	241.0	1605.0
	200	16.8	0	32170.0	2351.6	45997.8	48.4	7.6	119.8	0.65	0.78	307.8	–

Table 2 Solving ATPP instances with $|V| = 50$ with the branch-and-cut algorithm

$\|K\|$	λ	*Solved*	#	*2sec*	*ysec*	*zsec*	*2mat*	D_k^+	*Nodes*	*%UB*	*%LB*	*Root-t*	*Total-t*
50	0.50	5	50.0	636.2	1.8	35.0	90.2	133.4	2.2	0.01	0.00	0.0	0.6
	0.80	5	40.2	294.4	3.2	32.4	132.4	175.6	14.0	0.04	0.01	0.0	1.6
	0.90	5	27.4	390.8	22.4	179.0	712.8	857.6	41.2	0.07	0.08	0.0	15.2
	0.95	5	18.0	664.8	62.4	718.8	1017.4	2723.0	72.6	0.24	0.19	0.0	31.4
	0.99	5	10.0	484.0	20.0	370.2	115.2	1070.2	5.0	0.62	0.30	0.6	6.2
100	0.50	5	50.0	1325.2	2.4	84.2	166.6	525.2	7.4	0.00	0.00	0.2	3.0
	0.80	5	50.0	594.4	2.8	64.0	88.2	253.2	40.2	0.01	0.00	0.0	3.0
	0.90	5	42.8	668.8	19.4	257.0	739.8	1032.4	45.6	0.09	0.04	0.0	27.8
	0.95	5	28.6	12748.2	635.4	10177.4	15147.2	37812.2	543.8	0.21	0.18	0.2	800.8
	0.99	5	15.8	1436.8	40.0	1840.6	590.4	5039.8	17.4	0.34	0.24	1.6	31.6
150	0.50	5	50.0	1979.6	2.4	101.0	134.2	678.6	8.6	0.00	0.00	0.0	2.8
	0.80	5	50.0	909.8	3.4	133.6	176.2	650.0	8.0	0.01	0.00	0.8	3.2
	0.90	5	46.2	617.0	4.6	148.8	229.4	764.0	22.8	0.02	0.01	0.0	8.0
	0.95	5	34.2	5457.2	209.4	3176.8	5070.2	15189.8	212.8	0.12	0.10	0.2	424.4
	0.99	5	18.4	2669.4	70.8	2464.2	626.0	6527.4	28.8	0.42	0.18	2.4	71.4
200	0.50	5	50.0	2704.2	2.0	94.0	61.4	542.8	12.8	0.00	0.00	0.0	1.8
	0.80	5	48.8	1254.8	2.2	154.6	184.2	917.6	6.4	0.00	0.00	0.8	5.2
	0.90	5	48.8	779.6	3.8	154.2	145.8	692.2	8.0	0.02	0.01	0.8	6.4
	0.95	5	40.0	3652.8	124.4	2639.2	3538.4	11679.8	102.6	0.08	0.07	1.0	344.8
	0.99	5	21.4	7344.0	180.0	5862.6	1434.6	16103.6	67.0	0.29	0.20	2.6	221.6

Table 3 Solving ATPP instances with $|V| = 100$ with the branch-and-cut algorithm

$\|K\|$	λ	*Solved*	#	*2sec*	*ysec*	*zsec*	*2mat*	D_k^+	*Nodes*	*%UB*	*%LB*	*Root-t*	*Total-t*
50	0.50	5	100.0	1546.8	122.2	136.2	1095.2	502.2	576.4	0.01	0.00	2.4	265.0
	0.80	5	91.2	714.2	28.8	78.2	752.8	292.0	269.4	0.02	0.00	2.2	83.0
	0.90	5	60.8	4998.4	272.0	2834.0	16540.6	8709.6	393.6	0.06	0.04	1.8	1311.6
	0.95	1	35.0	19694.2	1189.8	13954.4	65805.4	57756.2	1440.8	0.26	0.16	1.8	6767.8
	0.99	5	13.6	6407.6	262.4	8251.0	3857.8	17495.6	100.0	0.13	0.56	5.6	2029.6
100	0.50	5	100.0	2856.0	13.0	101.4	433.6	534.4	270.0	0.00	0.00	4.4	114.2
	0.80	5	90.4	1325.0	6.0	81.0	476.4	460.6	87.8	0.01	0.00	3.2	33.0
	0.90	5	71.6	1271.6	25.6	343.2	1320.4	1390.2	76.2	0.03	0.01	3.0	190.2
	0.95	0	–	–	–	–	–	–	–	0.14	0.13	4.0	–
	0.99	3	20.0	111815.4	3087.8	89527.4	31175.0	189650.4	988.4	0.74	0.33	10.8	5854.4
150	0.50	5	100.0	4542.6	23.4	108.4	388.2	670.6	347.8	0.00	0.00	2.6	185.4
	0.80	5	100.0	22563.2	1763.4	46.8	901.2	585.6	5.4	0.00	0.00	16.2	16.2
	0.90	5	90.4	3122.6	1120.8	152.8	2632.0	1672.0	96.2	0.02	0.01	8.8	8.8
	0.95	0	–	–	–	–	–	–	–	0.07	0.10	5.0	–
	0.99	5	18.4	108864.2	1837.0	25643.2	20542.6	174276.0	694.2	0.30	0.26	12.2	5623.6
200	0.50	5	100.0	5866.4	15.0	215.8	625.8	1181.0	203.4	0.00	0.00	5.0	97.6
	0.80	5	100.0	3338.4	77.4	347.0	676.6	1089.4	495.0	0.00	0.00	46.4	294.6
	0.90	5	95.0	2355.8	119.6	656.0	999.8	1580.8	617.8	0.01	0.01	9.4	7.2
	0.95	0	–	–	–	–	–	–	–	0.06	0.09	8.6	–
	0.99	2	22.0	103687.6	1666.8	46234.8	13034.6	146714.6	438.0	0.31	0.23	17.6	6675.6

ysec: average number of constraints (4) separated;
zsec: average number of constraints (15)–(16) separated;
2mat: average number of constraints (17) separated;
D_k^+*:* average number of constraints (12)–(13) separated;
Nodes: average number of nodes explored during the branch-and-cut execution;
%UB: average gap between the heuristic and the optimal solutions at the end of the root node, over the optimal solution value;
%LB: average gap between the fractional and the optimal solutions at the end of the root node, over the optimal solution value;
Root-t: average computational time in seconds at the end of the root node;
Total-t: average computational time in seconds for the whole branch-and-cut execution.

According to Table 1, the branch-and-cut algorithm above described was able to solve most of the 80 unrestricted ATPP instances. Only 15 instances have not been solved before the time limit. Only those instances solved before the time limit of two hours have been taken into account in the average computations. The difficulty of the problem grows clearly with the number of available markets and required products. The number of visited markets in an optimal solution tends to remain small, even when $|K| = 200$. This is mainly due to the ratio between the routing cost and the pricing cost in an optimal solution. In other words, the higher the pricing costs are, the bigger the number of markets in the optimal solution is.

Table 4 Branch-and-cut vs transformation for unrestricted ATPP instances

			Branch-and-cut				Transformation			
$\|V\|$	$\|K\|$	#	*Solved*	*%Gap*	*Root-t*	*Total-t*	*Solved*	*%Gap*	*Root-t*	*Total-t*
50	50	9.2	5	0.80	1.0	5.6	5	1.42	3.0	13.2
	100	12.4	5	0.78	1.6	8.8	5	1.01	2.8	20.0
	150	14.0	5	0.51	1.8	15.8	5	0.50	3.6	40.8
	200	16.0	5	0.49	1.8	25.6	5	0.45	3.6	61.0
100	50	6.0	5	0.29	17.0	36.8	5	0.55	30.2	57.6
	100	12.0	5	1.26	21.2	411.0	5	1.37	44.4	956.0
	150	14.8	5	1.23	25.4	1646.4	4	1.72	38.0	2137.0
	200	17.2	5	0.88	24.2	2237.6	4	1.05	28.2	1157.0
150	50	7.6	5	1.56	87.2	812.2	5	2.31	148.6	1371.4
	100	10.6	5	1.34	99.0	2302.8	5	1.69	204.4	3256.0
	150	14.4	5	1.19	69.0	2247.4	5	1.50	111.4	3880.0
	200	15.8	1	1.35	100.4	1428.0	1	1.58	131.0	2609.0
200	50	7.8	5	1.17	277.8	1484.8	5	1.23	290.6	1663.2
	100	10.2	3	1.13	172.6	3037.7	3	1.55	189.6	3511.7
	150	13.2	1	1.29	241.0	1605.0	1	1.47	279.2	2162.0
	200	16.8	0	1.43	307.8	–	0	1.74	418.0	–

All the separation procedures described in Section 2 succeeded in finding some violated constraints. Our computational experience shows that constraints (11), (4) and (15) are quite relevant, since the average computational time grows when their separation procedures are not available. This behaviour cannot be extended to constraints (17), (12) and (13): the computational time does not show an important increase when those constraints are not considered. Activating all the separated constraints, the lower bound at the end of root node has never been bigger than 1% with respect to the cost of an optimal solution. A similar result applies also to the upper bound obtained by using the initial and primal heuristics. Because of the gap between the lower and the upper bounds at the end of the root node, the exact algorithm requires a branching phase but despite of this, our approach has obtained the optimality proof before the time limit in most of the instances.

Tables 2 and 3 show the statistical results when the branch-and-cut code is used to solve the restricted ATTP instances. Columns represented by # show how sensitive is the number of markets in an optimal circuit with respect to λ, which also has a strong impact in the algorithm performance.

The hardness of solving the restricted instances with the branch-and-cut code is observed in Tables 2 and 3. The total computational time grows with the parameter λ (which is also related to the length of an optimal circuit). The computational time attains its maximum at $\lambda = 0.95$, and immediately the hardness begins to decrease. As it has also been observed in the unrestricted ATPP instances, the most relevant constraints are (11), (4) and (15), and the upper and lower bounds at the end of the root node are very close to the optimal solution value.

All instances of Table 2 (100 ATPP instances with $|V| = 50$) have been solved up to optimality before the time limit. However, 25 over 100 instances with $|V| = 100$ remain

Table 5 Branch-and-cut vs transformation for ATPP instances with $|V| = 50$

			Branch-and-cut				Transformation			
$\|K\|$	λ	#	*Solved*	*%Gap*	*Root-t*	*Total-t*	*Solved*	*%Gap*	*Root-t*	*Total-t*
50	0.50	49.0	5	0.01	0.0	0.6	5	0.03	0.2	6.0
	0.80	40.6	5	0.05	0.0	1.6	5	0.05	0.2	5.4
	0.90	27.4	5	0.15	0.0	15.2	5	0.20	0.0	37.6
	0.95	18.0	5	0.43	0.0	31.4	5	0.41	0.0	62.0
	0.99	10.0	5	0.93	0.6	6.2	5	1.35	1.4	18.6
100	0.50	50.0	5	0.00	0.2	3.0	5	0.01	1.0	2.8
	0.80	50.0	5	0.01	0.0	3.0	5	0.01	0.8	2.0
	0.90	42.8	5	0.14	0.0	27.8	5	0.12	0.4	74.0
	0.95	27.3	5	0.38	0.2	800.8	5	0.32	0.3	921.3
	0.99	15.8	5	0.58	1.6	31.6	5	0.79	2.0	82.0
150	0.50	50.0	5	0.00	0.0	2.8	5	0.01	1.2	5.4
	0.80	50.0	5	0.01	0.8	3.2	5	0.00	0.8	6.6
	0.90	46.4	5	0.04	0.0	8.0	5	0.03	0.8	17.4
	0.95	34.0	5	0.22	0.2	424.4	5	0.18	1.0	443.3
	0.99	18.2	5	0.61	2.4	71.4	5	0.52	3.0	222.8
200	0.50	49.8	5	0.00	0.0	1.8	5	0.00	2.2	6.8
	0.80	48.8	5	0.01	0.8	5.2	5	0.01	1.6	3.2
	0.90	49.0	5	0.03	0.8	6.4	5	0.03	1.2	9.2
	0.95	40.0	5	0.15	1.0	344.8	5	0.14	1.0	561.8
	0.99	21.4	5	0.48	2.6	221.6	5	0.50	2.0	520.2

unsolved with our time limit, as observed in Table 3. Notice that, only one over 20 instances with $\lambda = 0.95$ has been solved before the time limit in this table.

The transformation from the ATPP into the STPP presented in Section 3 has been also computationally tested on the two previous families of instances. Notice that a transformation of an ATPP instance with $|V|$ vertices produces a new STPP instance with $2|V|$ vertices and with a minor increment of the number of edges. Therefore, the size of the STPP instance is still reasonable for available exact algorithms.

Tables 4, 5 and 6 show a comparative study between the specific branch-and-cut algorithm for the ATPP and the above mentioned transformation. As in the previous tables, the three first columns show the cardinality of the instance and the number of markets involved in the optimal circuit generated. The next four columns, both for the specific branch-and-cut and for the transformation respectively, show the number of instances solved before the time limit (*Solved*), the percentage of the gap between the upper and lower bound over the upper bound at the root node (*%gap*), the computational time consumed at the root node (*Root-t*), and the total computational time taken by the optimal algorithm (*Total-t*).

Table 4 compares the original branch-and-cut algorithm against the transformation approach in the unrestricted ATPP instances. The specific branch-and-cut seems to be more efficient not only with respect to the running time but also with respect to the gap between the upper and lower bound. However, as long as the number of markets is increased, the different between these two approaches is reduced.

Table 6 Branch-and-cut vs transformation for ATPP instances with $|V| = 100$

				Branch-and-cut				*Transformation*		
$\|K\|$	λ	#	*Solved*	*%Gap*	*Root-t*	*Total-t*	*Solved*	*%Gap*	*Root-t*	*Total-t*
50	0.50	99.0	5	0.01	2.4	265.0	5	0.02	6.6	35.4
	0.80	90.8	5	0.03	2.2	83.0	5	0.03	7.0	92.8
	0.90	64.0	5	0.10	1.8	1311.6	1	0.03	2.0	129.0
	0.95	35.0	1	0.42	1.8	6767.8	0	–	–	–
	0.99	13.5	5	0.69	5.6	2029.6	4	0.70	9.8	1947.0
100	0.50	97.2	5	0.00	4.4	114.2	5	0.01	8.6	101.6
	0.80	90.2	5	0.01	3.2	33.0	5	0.01	9.6	34.8
	0.90	69.0	5	0.05	3.0	190.2	4	0.05	3.3	535.8
	0.95	–	0	–	–	–	0	–	–	–
	0.99	19.0	3	1.07	10.8	5854.4	2	1.06	12.0	5736.5
				0.00						
150	0.50	100.0	5	0.00	2.6	185.4	5	0.00	21.4	21.4
	0.80	99.8	5	0.00	16.2	16.2	5	0.01	9.0	9.0
	0.90	90.2	5	0.03	8.8	8.8	5	0.03	7.8	7.8
	0.95	–	0	–	–	–	0	–	–	–
	0.99	19.0	5	0.56	12.2	5623.6	1	0.88	11.0	4729.0
				0.00						
200	0.50	100.0	5	0.00	5.0	97.6	5	0.00	30.4	122.2
	0.80	100.0	5	0.00	46.4	294.6	5	0.00	10.8	46.8
	0.90	95.4	5	0.02	9.4	7.2	5	0.02	6.0	7.2
	0.95	–	0	–	–	–	0	–	–	–
	0.99	22.0	2	0.54	17.6	6675.6	2	0.56	17.5	5584.0

Tables 5 and 6 are related to restricted TPP instances. Also on these harder instances, the branch-and-cut code shows better performance than the transformation approach, even if there are several exceptions when $|V| = 100$. The smaller gap of the direct approach is due to the new *ad hoc* inequalities and the heuristic approaches.

5. Conclusions

We have formulated and solved the Asymmetric Traveling Purchaser Problem (ATPP), a generalization of the well-known Asymmetric Traveling Salesman Problem. Two approaches have been proposed to find optimal solutions. A first one is based on a new Integer Linear Programming model, strengthened through several families of valid inequalities. It is a branch-and-cut algorithm using some separation procedures to manage the huge number of constraints. The second approach is based on a transformation of the asymmetric instances into a symmetric instances, to be solved with an exact method from literature. The two approaches have been implemented and tested on instances from literature, proving that both are able to find optimal solutions of instances with up to 200 markets and 200 products within reasonable computing time. When restricted offer are also considered, the two new exact approaches were able to solve some ATPP instances involving 100 markets and 200 products. From our experiments even if the proposed transformation was observed to be quite effective, the *ad hoc* branch-and-cut algorithm provided a better average performance.

References

Balas, E. and S.M. Ng. (1989). “On the Set Covering Polytope: I. All the Facets With Coefficients in {0, 1, 2}.” *Mathematical Programming*, 43, 57–69.

Balas, E. and M. Oosten. (2000). “On the Cycle Polytope of a Directed Graph.” *Networks*, 36, 34–46.

Boctor, F.F., G. Laporte, and J. Renaud. (2003). “Heuristics for the Traveling Purchaser Problem.” *Computers and Operations Research*, 30(4), 491–504.

Burstall, R.M. (1966). “A Heuristic Method for a Job Sequencing Problem.” *Operational Research Quarterly*, 17, 291–304.

Buzacott, J.A. and S.K. Dutta. (1971). “Sequencing Many Jobs on a Multipurpose Facility.” *Naval Research Logistics Quarterly*, 18, 75–82.

Fischetti, M., A. Lodi, and P. Toth. (2002). Exact Methods for the Asymmetric Traveling Salesman Problem.” Kluwer Academic Publishers, Dordrecht, chapter 4, 169–206.

Fischetti, M., J.J. Salazar, and P. Toth. (1997). “A Branch-and-Cut Algorithms for the Symmetric Generalized Traveling Salesman Problem.” *Operations Research*, 45, 378–394.

Fischetti, M. and P. Toth. (1997). “A Polyhedral Approach to the Asymmetric Traveling Salesman Problem.” *Management Science*, 43, 1520–1536.

Golden, B.L., L. Levy, and R. Dahl. (1981). “Two Generalizations of the Traveling Salesman Problem.” *Omega*, 9, 439–445.

Grötschel, M. and M.W. Padberg. (1985). *Polyhedral Theory.* Wyley, Chichester. chapter 8, pp. 251–305.

Jonker, R. and T. Volgenant. (1983). “Transforming Asymmetric into Symmetric Traveling Salesman Problems.” *Operations Research Letters*, 2, 161–163.

Jünger, M., G. Reinelt, and G. Rinaldi (1995). *The Traveling Salesman Problem.* North–Holland, Amsterdam chapter 4.

Jünger, M. and S. Thienel. (1998). “Introduction to ABACUS—a Branch-and-Cut System.” *Operations Research Letters*, 22, 83–95.

Karp, R.M. (1972). “Reducibility Among Combinatorial Problems.” In Miller, R.E. and J.W. Thatcher. (ed.), *Complexity of Computer Computations*. New York: Plenum Press, pp. 85–103.

Laporte, G., J. Riera-Ledesma, and J.J. Salazar-González. (2003). “A Branch-and-Cut Algorithm for the Undirected Traveling Purchaser Problem.” *Operations Research*, 51(6), 940–951.

Ong, H.L. (1982). “Approximate Algorithms for the Traveling Purchaser Problem.” *Approximate Algorithms for the Traveling Purchaser Problem* 1, 201–205.

Padberg, M.W. (1975). “A Note on Zero-One Programming.” *Operations Research*, 23, 833–837.

Pearn, W.L. and R.C. Chien. (1998). “Improved Solutions for the Traveling Purchaser Problem.” *Computers & Operations Research*, 25, 879–885.

Ramesh, T. (1981). “Traveling Purchaser Problem.” *Opsearch*, 18, 78–91.

Riera-Ledesma, J. (2002). *The Traveling Purchaser Problem*. PhD thesis, DEIOC, Universidad de La Laguna.

Riera-Ledesma, J. and J.J. Salazar-González. (2005). “A Heuristic Approach for the Travelling Purchaser Problem.” *European Journal of Operational Research*, 162, 142–152.

Singh, K.N. and D.L. van Oudheusden. (1997). “A Branch and Bound Algorithm for the Traveling Purchaser Problem.” *European Journal of Operational Research*, 97, 571–579.

Voß, S. (1996). “Dynamic Tabu Search Strategies for the Traveling Purchaser Problem.” *Annals of Operations Research*, 63, 253–275.