Pesquisa Operacional (2018) 38(3): 499-522

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
doi: 10.1590/0101-7438.2018.038.03.0499

MODELING AND SOLVING THE TRAVELING SALESMAN PROBLEM WITH PRIORITY PRIZES

Vitoria Pureza[1], Reinaldo Morabito[1*] and Henrique P. Luna[2]

Received November 20, 2017 / Accepted August 15, 2018

ABSTRACT. This paper addresses the Traveling Salesman Problem with Priority Prizes (*TSPPP*), an extension of the classical *TSP* in which the order of the node visits is taken into account in the objective function. A prize p_{ki} is received by the traveling salesman when node i is visited in the k-th order of the route, while a travel cost c_{ij} is incurred when the salesman travels from node i to node j. The aim of the *TSPPP* is to find the maximum profit n-node tour. The problem can be seen as a *TSP* variant with a more general objective function, aiming at solutions that in some way consider the quality of customer service and the delivery priorities and costs. A natural representation for the *TSPPP* is here grounded in the point of view of Koopmans and Beckmann approach, according to which the problem is seem as a special case of the quadratic assignment problem (*QAP*). Given the novelty of this *TSP* variant, we propose different mixed integer programming models to appropriately represent the *TSPPP*, some of them based on the *QAP*. Computational experiments are also presented when solving the MIP models with a well-known optimization software, as well as with a tabu search algorithm.

Keywords: Traveling Salesman Problem with Priority Prizes, mixed integer linear programming, quadratic assignment problem, routing with priorities, flow formulations, tabu search.

1 INTRODUCTION

The Traveling Salesman Problem (*TSP*) is among the most widely studied problems in network optimization and has a wide variety of practical applications [2, 5, 20, 24]. The problem consists in defining an optimal tour that visits customers from a depot and since the pioneering work of Dantzig and collaborators [7, 8], several models and methods have been proposed to effectively represent and solve large problem instances. Many variants of the *TSP* have been studied in the literature and seminal mathematical programming models are often the basis of most of these variants [4, 11, 22, 28].

*Corresponding author.
[1]Departamento de Engenharia de Produção, Universidade Federal de São Carlos, Via Washington Luiz km. 235, 13565-905 São Carlos, SP, Brazil. E-mails: vpureza@dep.ufscar.br; morabito@ufscar.br
[2]Instituto de Ciências da Computação, Universidade Federal de Alagoas, Av. Lourival Melo Mota, s/n, 57072-900 Maceió, AL, Brazil. E-mail: henrique.luna@pq.cnpq.br

The Traveling Salesman Problem with Priority Prizes (*TSPPP*) is an extension of the classic *TSP*, in which all customers (nodes) have to be visited by the traveling salesman and the order of the customers visits are directly taken into account in the objective function. A prize p_{ki} is received by the traveling salesman if customer i is visited in the k-th order of the tour, while a travel cost c_{ij} is incurred when traveling from customer i to customer j in the tour. Note that p_{ki} can include a prize that the traveling salesman gets when visiting node i, independently of the order k in which i is visited, plus a priority prize that he/she collects when visiting node i at the k-th order of the route, which is dependent on his/hers visit order. The objective of the *TSPPP* is to find a maximum profit sequence of the n-customer visits, considering the prizes and costs involved in the tour.

The problem can be seen as a *TSP* variant with a more general objective function with opposite criteria, aiming at solutions that in some way consider the quality of service to the customers and the delivery priorities, maximizing the collected prizes while minimizing the delivery costs. A simple example of the *TSPPP* is a van or minibus that picks up a group of tourists at an airport and delivers each tourist to his/her hotel. Instead of equally sharing the cost of the minimum cost tour, some tourists are willing to pay more if their hotels were visited at the first positions of the tour, while other tourists are not. The driver would like to choose the most profitable tour, however, complying with these priority prizes may increase the total distance traveled and hence, the tour cost. Another example arises in the context of machine scheduling problems. Single Machine Scheduling with sequence-dependent set up costs is a referential problem in production planning and it is a well-known application of the *TSP* in which the machine owner seeks the minimum set up cost production plan. When job priority prizes are also taken into account, the problem becomes an application of the *TSPPP*.

We are not aware of other studies that have directly dealt with the *TSPPP*, although related problems can be found in the literature. One example is the Minimum Latency Problem (*MLP*) [14], also known as the Traveling Repairman Problem, the Traveling Deliveryman Problem or the Traveling Salesman Problem with Cumulative Costs [4, 6, 10, 22, 28]. In the *MLP*, the traveling salesman is required to visit each customer in such a way that the overall waiting time for all customers is minimized (in this case costs are given by $c_{ijk} = (n-k)c_{ij}$ if arc (i, j) is the k-th arc traversed in the tour). The *MLP* can also be interpreted as a Single Machine Scheduling Problem with sequence-dependent processing times, in which one seeks to minimize the total flow time of the jobs [6]. It can also be seen as a special case of the Time-Dependent Traveling Salesman Problem (*TDTSP*) [15, 22, 25], in which the cost of traversing an arc between two nodes i and j may vary with the position of this arc (i, j) along the Hamiltonian tour [22] (i.e., $d_{ikj(k+1)} = c_{ijk}$). In fact, the *TDTSP* is strongly related to the *TSPPP* of the present study, as discussed in next section. An extension of the *MLP* is the Single Vehicle Delivery Problem. Instead of working with a general time-dependent cost function as in the *TDTSP*, this extension requires a different amount of demand d_i to be delivered at each node i (note that it is reduced to the *MLP* when the demand for each customer is a single unit, i.e., $d_i = 1$).

Other related *TSP* variants are the Multicommodity Traveling Salesman Problem, which encompasses heterogeneous node demands d_k and also includes unitary flow costs c_{ijk} that are specific

to the type of commodity k being transported across an arc (i, j); the Target Visitation Problem, in which one seeks a tour that visits all nodes in order to maximize a function representing the preference of visiting a node i before (but not necessarily immediately before) another node j, while minimizing travel costs [17]; and the Traveling Salesman Problem with Profits [9], where a profit p_i is associated to each customer i (node) and the traveling salesman does not need to visit all customers. The problem consists of finding a tour on a subset of customers in order to maximize a profit measure while minimizing travel costs. In other words, the aim is to find a route that maximizes the collected profit minus the transportation cost. When penalty terms for unvisited nodes are also added to the objective function, the problem is known as the Prize-Collecting Traveling Salesman Problem [3, 9]. Other two related problems in which customers have to be selected are the Traveling Purchaser Problem and the Generalized Traveling Salesman Problem. In the first, each node holds commodities, each with its own associated purchase cost. In the second, nodes are partitioned in clusters and the aim is to find a minimum cost tour visiting at least one node from each cluster ([9]).

In this study, to cope with the node visitation order, we first explore a representation for the *TSPPP* grounded on the Quadratic Assignment Problem (*QAP*) in Koopmans and Beckmann [18, 19]. We present different mixed integer linear programming (MIP) models derived from the *QAP* as well as other models to deal with the *TSPPP*. We also present an adaptive tabu search algorithm that systematically changes selected tabu elements based on the analysis of search trajectories patterns. Given the novelty of this *TSP* variant, the main contributions of this paper are threefold: the presentation of MIP formulations that appropriately represent the problem, the presentation of an effective metaheuristic capable to solve larger instances, and the analysis of some numerical experiments with the models and the metaheuristic, comparing their computational performances and highlighting an interesting problem insight: as the p_{ki} values become larger and with higher dispersion and the c_{ij} become smaller and with smaller dispersion, the *TSPPP* becomes easier to optimally solve as it tends to a linear assignment problem.

The paper is organized as follows. Section (2) introduces a pure integer quadratic programming model to represent the *TSPPP* based on the *QAP* in Koopmans and Beckmann [18]. Section (3) is devoted to the development of linearisations of this model for the *TSPPP*, thus providing two MIP formulations and a third two-assignment formulation. As these formulations are able to solve only problems of moderate size, Section (4) describes the tabu search implementation to cope with larger problem instances. Section (5) describes and analyses some numerical experiments from solving the MIP formulations using the optimization sofware CPLEX and by applying the heuristic algorithm. Finally, section (6) presents our final remarks and some perspectives for future research.

2 INTEGER QUADRATIC PROGRAM FOR *TSPPP*

A natural formulation for the *TSPPP* can be conceived as an integer quadratic programming problem with a quadratic objective function and linear constraints. The problem is to find a maximal profit Hamiltonian circuit over a directed graph $G(N, A)$, where N is the set of n nodes

(customers) and A is the set of m arcs (conecting the customers). The model parameters are provided by two square matrices, P and C, both of order n. Each entry $p_{ki} \geq 0$ of matrix P is the prize received by the traveling salesman when node i is visited in order k of the route. Note that p_{ki} can be viewed as a "semi-net" revenue from the visit of node i in order k, which may include a prize the traveling salesman gets when visiting node i (say, p_i), independently of order k, plus a priority prize that he collects when visiting node i at the k-th order of the route (say, p'_{ki}), which is dependent on the visit order. It is assumed that $p_{ki} = p'_{ki} + p_i$ is independent of the prizes of other nodes.

Each entry $c_{ij} > 0$ of the non-symmetric matrix C evaluates the transportation cost of the salesman when traveling from node i to node j, and it is defined only for the arcs in A. For the diagonal of C and for nodes i and j not linked by a direct arc in A, the entries are either disregarded or considered as $c_{ii} = \infty$ and $c_{ij} = \infty$, $(i, j) \notin A$. The assumption of positive coefficients c_{ij} is here made to simplify the discussion, but it can be relaxed in other applications. It is also assumed that the costs c_{ij} satisfy the triangular inequality, i.e., $c_{ij} \leq c_{ih} + c_{hj}$, and that they are independent of the order assignments.

Unless stated otherwise, the graph is complete, with a number of $m = n(n-1)$ arcs. The model size depends upon the number of nodes n and the number of arcs m. The binary variables are defined by n^2 variables x_{ki}. Each variable x_{ki} is equal to 1 if node i is visited in order k, and 0 otherwise. Without loss of generality, we assume that the origin of the salesman (depot) is node 1 and this node is also the last node to be visited, so that variable x_{n1} is fixed to 1 and variables $x_{k1}, k = 1, \ldots, n-1$, are null. As a consequence, the problem has in fact only $(n-1)^2$ variables to be determined. The quadratic traveling salesman problem with priority prizes can be defined as:

$$\max \sum_{k=1}^{n} \sum_{i=1}^{n} p_{ki}\, x_{ki} - \sum_{k=1}^{n-1} \sum_{(i,j)\in A} c_{ij}\, x_{ki}\, x_{(k+1)j} - \sum_{(1,j)\in A} c_{1j}\, x_{1j} \tag{1}$$

subject to:

$$\sum_{k=1}^{n-1} x_{ki} = 1 \quad \forall\, i = 2, \ldots, n \tag{2}$$

$$\sum_{i=2}^{n} x_{ki} = 1 \quad \forall\, k = 1, \ldots, n-1 \tag{3}$$

$$x_{ki} \in \{0, 1\} \quad \forall\, k = 1, \ldots, n \quad \forall\, i = 1, \ldots, n \tag{4}$$

The objective function (1) subtracts the transportation costs incurred in the sequence of node visits from the received priority prizes, that is, it is the total net revenue of the traveling salesman ("semi-net" revenue minus transportation costs). We remark that the last parcel accounts for the transportation cost incurred in the first node visit of the salesman, i.e. from the depot to the first visited node. Equalities (2) and (3) are the assignment constraints: equalities (2) assure that each node i is visited in a unique order, whereas equalities (3) state that each visit order k is

assigned to a unique node (assuming $x_{n1} = 1$ and $x_{k1} = 0, k = 1, \ldots, n-1$). Constraints (4) prescribe that the components of vector x are binary variables. We remind that the visit order n is previously fixed for the depot node 1 (i.e., $x_{n1} = 1$), indicating that the traveling salesman returns to the depot 1 at the tour end.

Koopmans and Beckman [18] studied problems of assigning plants to locations, considering "semi-net" revenues p_{ki} from the operation of plant k in location i, transportation costs c_{ij} between locations i and j, and nonegative numbers b_{kl} representing required commodity flows from plant k to plant l. These assignment problems were modeled as *QAP* maximizing the total net revenue, where the classic *TSP* can be interpreted as a particular case when all p_{ki} and b_{kl} are null, except for $b_{k(k+1)} = 1$ and $b_{n1} = 1$ (depot). We note that the integer quadratic formulation (1)-(4) is also a particular case of Koopmans and Beckman's *QAP* which, instead of assuming a null linear parcel as in the *TSP*, it also includes the term $\sum_{k,i} p_{ki}\, x_{ki}$ in the objective function of the traveling salesman problem. A plant k of the original *QAP* formulation in [18] is interpreted as the visit order k for the *TSPPP*, and the flows among the plants (orders) are maintained but restricted to subsequent orders. The starting point (depot) is node 1 that should be visited by the traveling salesman again in the last order n, while each other node should be visited at some order $k < n$.

Model (1)-(4), based on the input parameters p_{ki} and c_{ij}, is also directly related to quadratic assignment models for the *TDTSP* based on input cost parameters d_{ikjl} ([21,25]), as well as for one-machine scheduling problems, in which the cost of crossing an arc between two nodes may vary with the position of this arc along the Hamiltonian tour [15,25]. By defining $d_{ikj(k+1)} = c_{ijk}$ for $i, j = 1, \ldots, n, i \neq j, k = 1, \ldots, n-1$, as the "cost" incurred when the salesman travels from node i to node j at the k-th position of its route and $d_{ikjl} = 0$ for the remaining cases, both models are related by minimizing this "cost" c_{ijk} defined as $c_{ij} - p_{ki}$.

We observe that as the p_{ki} values become large and the c_{ij} values become small, one would expect that model (1)-(4) becomes easier to solve, as it tends to the Linear Assignment Problem (*LAP*) (with the linear "semi-net" revenue term of the objective function (1) dominating the last two terms), which is an easy problem in class P. On the other hand, as p_{ki} becomes small and c_{ij} becomes large, it is expected that the problem becomes difficult to solve as it tends to the classic *TSP*, with the quadratic transportation cost term of the objective function (1) dominating the first term.

Although the *TDTSP* is a generalization of the *TSPPP* as stated in model (1)-(4), it requires the more general formulation of Lawler [21] for the *QAP*, thus departing from the original model of Koopmans and Beckmann [18] and hence loosing the appealing interpretation of matrices P, B and C of the original model. Here we prefer to maintain this original interpretation, which emphasizes the role of matrix P in the linear part of the model, as discussed with the following example. A classical *TSP* application concerns the one-machine scheduling problem with set up costs. A corresponding application of the *TSPPP* concerns the extension that consider priority prizes in this class of scheduling problems. Consider the problem where n jobs must be scheduled on a single machine. Job 1 denotes the initial state of the machine, which will be again in this

state after processing all the remaining $n-1$ jobs. A set of $n-1$ jobs, denoted by $2,\ldots,n$, are to be performed on the machine and a set up cost c_{ij} is incurred when job j is processed immediately after job i. A prize p_{ki} is received by the machine owner when job i is processed in order k. The objective is to find a maximal profit sequence of the n jobs, in such a way that the problem can be cast as a *TSPPP*.

A notable characteristic of the *TSPPP* is that it can be easily extended to cope with priority prizes in time dependent *TSP*. The *Time Dependent TSPPP* (*TDTSPPP*) is a generalization of the *TSPPP* where the cost of any given arc depends on its position in the tour. In terms of single machine scheduling, instead of c_{ij}, in this case a set up cost c_{ijk} is incurred when job j is processed immediately after job i in order k. The quadratic time dependent traveling salesman problem with priority prizes is:

$$\max \sum_{k=1}^{n}\sum_{i=1}^{n} p_{ki}\, x_{ki} \; - \sum_{k=1}^{n-1}\sum_{(i,j)\in A} c_{ijk}\, x_{ki}\, x_{(k+1)j} \; - \sum_{(1,j)\in A} c_{1j}\, x_{1j} \tag{5}$$

subject to the assignment constraints (2) – (4).

The simple procedure of introducing the set up cost c_{ijk}, instead of c_{ij}, enables the use of the Koopmans and Beckmann's point of view to cope with time dependence in extensions of both the classical *TSP* and the *TSPPP* here studied. Apart this observation, we do not pursue further in the question of time dependence in this paper. The idea is to focus on the priority prizes as the main aspect to be studied in the original quadratic *TSP*.

3 MIXED INTEGER LINEAR PROGRAMS FOR *TSPPP*

This section discusses how to obtain equivalent mixed integer linear programming models for the *TSPPP*. The idea here is to adapt and apply for the *TSPPP* the same linearisation suggested by Koopmans and Beckman in [18] for the more general *QAP*. The flow balance equations that enable a linear formulation for the *QAP* are given by:

$$b_{kl}x_{ki} + \sum_{(h,i)\in A} f_{hi}^{kl} = b_{kl}x_{li} + \sum_{(i,j)\in A} f_{ij}^{kl} \quad \forall\, k,l = 1,\ldots,n \;\forall\, i = 1,\ldots,n \tag{6}$$

where b_{kl} is defined in previous section and f_{ij}^{kl} is the flow from location i to location j of the commodity which is supplied by plant k to plant l. These equations specify that total inflow of an "intermediate commodity" (k,l) to a location i ($\sum_{(h,i)\in A} f_{hi}^{kl}$), added to its "production" at that location ($b_{kl}x_{ki}$), equals the total outflow of (k,l) from i ($\sum_{(i,j)\in A} f_{ij}^{kl}$), plus its "consumption" at that same location ($b_{kl}x_{li}$) (Figure 1). Here we rewrite these equations for the particular $b_{k(k+1)} = 1$ corresponding to the *TSPPP* and as a consequence, $f_{ij}^{k(k+1)}$ is simply the unit flow from location i to location j of an "intermmediate commodity" supplied by the "plant" at order k to the "plant" at the consecutive order $k+1$, and all the remaining f_{ij}^{kl} are null. In the words of Koopmans and Beckmann, the one and only one "intermediate commodity" now is a traveling salesman who is required to call once at each location and return to his point of departure.

Inspired by this interpretation, the following two subsections present flow formulations to appropriately represent the *TSPPP* problem. The third subsection presents a mixture of two linear assignment problems which also represents the *TSPPP*.

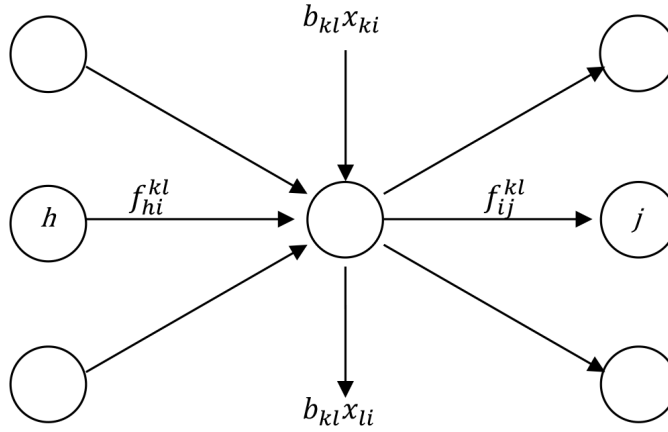

Figure 1 – Flow balance equations.

3.1 Transshipment Flow Formulation for *TSPPP*

In a refined interpretation, the "commodity" of a salesman visit at each node i is differentiated by the corresponding visit order k, as far as node i offers different prizes p_{ki} for its alternative visit orders. In the *TSPPP*, a flow variable f_{hi}^{k} equal to one indicates a visit order $k-1$ for node h in a tour where node i is the immediate sucessor of node h. For instance, if i is the first node to be visited, then one unit of outflow f_{hi}^{1} emerges from the predecessor node h and constitutes one unit of inflow to node i, where "commodity 1" (i.e., visit order 1) is consumed at prize p_{1i}. An outflow of one unit of "commodity 2" then emerges from that node i, thus forcing a corresponding unit of inflow to some other node j where this commodity (visit order 2) is consumed. The tour goes on to nodes with visit orders 3, 4 and so on. At last, a node l is visited in order $n-1$, from which emerges one unit of flow f_{l1}^{n} that will be consumed at the origin node $h=1$ (the depot in visit order n). As before, we assume that the visit order n is previously fixed for node 1 (i.e., $x_{n1}=1$), indicating that the traveling salesman returns to the origin 1 at the end of the tour.

A number of n^3 flow variables f_{ij}^{k} is here required, instead of working with the n^4 flow variables of the *QAP* in [18] (we note that $f_{ii}^{k}=0$). Our quadratic formulation (1-4) can then be linearised with the introduction of such related flow variables. A MIP model for the traveling salesman problem with priority prizes can be defined as (assuming $x_{n1}=1$ and $x_{k1}=0, k=1,\ldots,n-1$):

$$\max \sum_{k=1}^{n}\sum_{i=1}^{n} p_{ki}\, x_{ki} - \sum_{k=1}^{n}\sum_{(h,i)\in A} c_{hi}\, f_{hi}^{k} \tag{7}$$

subject to the assignment constraints (2) – (4) and to:

$$x_{ni} + \sum_{(h,i)\in A} f_{hi}^{1} = x_{1i} + \sum_{(i,j)\in A} f_{ij}^{1} \quad \forall\, i = 1, \ldots, n \tag{8}$$

$$x_{(k-1)i} + \sum_{(h,i)\in A} f_{hi}^{k} = x_{ki} + \sum_{(i,j)\in A} f_{ij}^{k} \quad \forall\, k = 2, \ldots, n\ \forall\, i = 1, \ldots, n \tag{9}$$

$$f_{hi}^{k} \geq 0\ \forall\, (h,i) \in A\ \forall\, k = 1, \ldots, n \tag{10}$$

The linear objective function (7) sums the received priority prizes less the transportation costs incurred in the sequence of node visits. Observe that now, in a tour where node h is the immediate predecessor of node i, instead of a quadratic term, a linear term $c_{hi}\, f_{hi}^{k}$ accounts for the transportation cost c_{hi} that is incurred when visit order k is assigned to node i. As in the previous quadratic formulation, the assignment constraints (2) - (4) refer only to the x variables. Equalities (2) assure that each node is visited in an unique order and equalities (3) state that a generic visit order k is assigned for an unique node. Constraints (4) mean that the components of vector x assume values 0 or 1.

The flow balance equations (8) – (9) are the coupling constraints of variables x and f. These equations have a transshipment interpretation for each "commodity" at each different node, in the same spirit of the original *QAP* formulation in [18]. Observe that the flow balance equations (9) refers to the commodities $k = 2, \ldots, n$. The total inflow of an "intermediate commodity" to a location i ($\sum_{(h,i)\in A} f_{hi}^{k}$), added to its "production" at that location ($x_{(k-1)i}$), equals the total outflow from i ($\sum_{(i,j)\in A} f_{ij}^{k}$), plus its "consumption" at that same location (x_{ki}) (Figure 2).

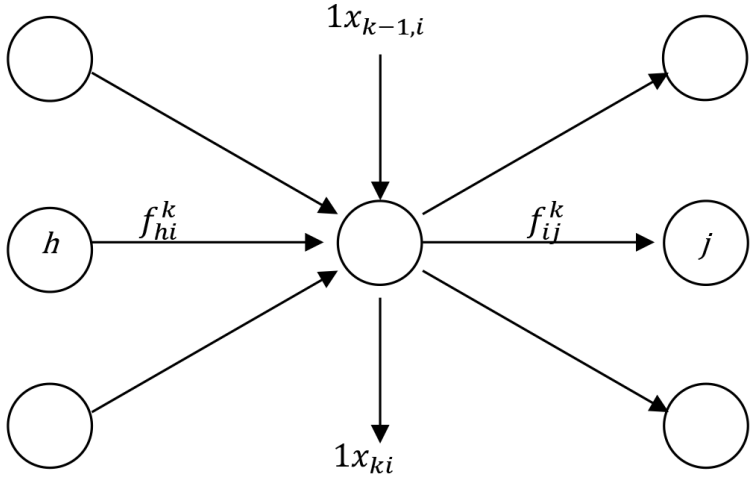

Figure 2 – Flow balance equations of the transshipment flow formulation.

Because only two subsequent visit orders relates the corresponding components of vectors x and f, observe that such coupling constraints (8) – (9) are quite sparse. With respect to a visit order $k \geq 2$, if node i in an optimal tour is visited before order $k-1$ or after order k, then (9) for pair k, i is trivially satisfied with all terms null. On the other hand, if node i is related to orders $k-1$ or k, then node i is either producer or consumer of commodity k, in such a way that equations (9) account for the correct balance. In complement, equations (8) refers to some node i that is served in order $k = 1$. We remark that this commodity 1 is not consumed and is specifically produced at some starting node h, from which an inflow $f^1_{hi} = 1$ is generated to the first served node i.

As stated, this transhipment flow formulation (7) – (10) may have alternative optimal solutions which involve subcycles in the flow variables f^k_{ij}. In order to avoid this, the following (redundant) valid inequalities can be added to the model which, together with the objective function, ensure that if $x_{k-1,h} = 1$ and $x_{ki} = 1$, then $f^k_{hi} = 1$ (similarly, if $x_{n1} = 1$ and $x_{1i} = 1$, then $f^1_{1i} = 1$):

$$f^1_{1i} \geq x_{n1} + x_{1i} - 1 \ \forall (1, i) \in A \tag{11}$$

$$f^k_{hi} \geq x_{(k-1)h} + x_{ki} - 1 \ \forall (h, i) \in A \ \forall k = 2, \ldots, n \tag{12}$$

The mixed integer linear formulation (7) – (10) (or (7) – (12)) enables the use of standard MIP packages to search for exact solutions for the problem. A variant of this model, which is a stronger formulation, is presented in next subsection. Large instances could eventually be solved in reasonable time to provide references for the faster TS algorithm that we present in Section 4. Decomposition strategies could also be devised to cope with memory problems in many of such large instances. All these possibilities are open for future work.

3.2 Transportation Flow Formulation for *TSPPP*

The linearisation (7) – (10) may reveal weak for many instances, since constraints (8) to (10) can be satisfied with zero flows between facilities. A similar linearisation was applied to a *QAP* in [18], for which the prize p_{ki} is equal to a constant, say p_k, regardless of node i. The linear programming relaxation yielded a trivial solution, with $x_{ki} = 1/n$ for each pair (k, i), consequently producing low quality linear programming lower bounds.

An alternative is to work with a bipartite flow formulation for the *TSPPP*, as suggested in [23] with respect to the *QAP* linearised Koopmans and Beckmann model. Instead of working with the balance equations (8) and (9), we rewrite each of these constraints as two equivalent sets. Note in (9) that as $x_{(k-1)i} + x_{ki} \leq 1$ (see 2) and $x_{(k-1)i} + x_{(k-1)h} \leq 1$ (see 3), then if $x_{(k-1)i} = 1$, it follows that $x_{ki} = 0$ and $\sum_{(h,i)\in A} f^k_{hi} = 0$ because $x_{(k-1)h} = 0$, which implies that $x_{(k-1)i} = \sum_{(i,j)\in A} f^k_{ij}$ for some j, for which $x_{kj} = 1$ (consumption). On the other hand, if $x_{(k-1)i} = 0$, then $\sum_{(i,j)\in A} f^k_{ij} = 0$, which implies that $\sum_{(h,i)\in A} f^k_{hi} = x_{ki}$. Similarly to the flow balance constraint (8). Moreover, if $x_{ki} = 1$, it follows that $x_{(k-1)i} = 0$ and $\sum_{(i,j)\in A} f^k_{ij} = 0$, which implies that $\sum_{(h,i)\in A} f^k_{hi} = x_{ki}$ for some h, for which $x_{(k-1)h} = 1$ (production). On the other hand, if $x_{ki} = 0$, then $\sum_{(h,i)\in A} f^k_{hi} = 0$, which implies that $x_{(k-1)i} = \sum_{(i,j)\in A} f^k_{ij}$.

We are interested to represent the flow balance at the origin and the destination points for each commodity k. By analogy with the classical transshipment and transportation problems, we are now interested in a transportation version of our problem based on a bipartite graph. The stronger MIP model for the *TSPPP* has the same objective function (7), subject to the assignment constraints (2) – (4) (assuming $x_{n1} = 1$, $x_{k1} = 0, k = 1, \ldots, n-1$, and $f_{ii}^k = 0$), to the non-negativity constraints (10) and to:

$$\sum_{(i,j)\in A} f_{ij}^1 = x_{ni} \quad \forall\, i = 1, \ldots, n \tag{13}$$

$$\sum_{(i,j)\in A} f_{ij}^k = x_{(k-1)i} \quad \forall\, k = 2, \ldots, n\ \forall\, i = 1, \ldots, n \tag{14}$$

$$\sum_{(h,i)\in A} f_{hi}^1 = x_{1i} \quad \forall\, i = 1, \ldots, n \tag{15}$$

$$\sum_{(h,i)\in A} f_{hi}^k = x_{ki} \quad \forall\, k = 2, \ldots, n\ \forall\, i = 1, \ldots, n \tag{16}$$

The above flow formulation imply non-zero flows and is able to produce better linear programming relaxation bounds when compared to the original (7) – (10) formulation. This alternative flow formulation has the same number of variables. On the other hand, it has the double of constraints. It also detaches information, sparing the cost and prize matrices, taking advantage of the demand matrix sparsity. The idea here is to obtain a well balanced mixed integer programming formulation, with a reasonable linear programming lower bound, being easy to solve.

3.3 Two-assignment Formulation for *TSPPP*

A pure integer linear program for the *TSPPP* can be written with a mixure of the two well known linear assignment problems that support the *TSP*. The idea is to include the previously defined x variables and the assignment variables y_{ij} equal to 1 if arc (i, j) is included in the route and 0 otherwise, together with appropriate coupling constraints. A small and weak pure integer linear program for the *TSPPP* can be stated as (assuming $x_{n1} = 1$, $x_{k1} = 0, k = 1, \ldots, n-1$, and $y_{ii} = 0$):

$$\max \sum_{k=1}^{n}\sum_{i=1}^{n} p_{ki}\, x_{ki} - \sum_{(h,i)\in A} c_{hi}\, y_{hi} \tag{17}$$

subject to the assignment constraints (2) – (4) and to:

$$\sum_{j=1}^{n} y_{ij} = 1 \quad \forall\, i = 1, \ldots, n \tag{18}$$

$$\sum_{i=1}^{n} y_{ij} = 1 \quad \forall\, j = 1, \ldots, n \tag{19}$$

$$y_{1j} \geq x_{n1} + x_{1j} - 1 \quad \forall\ (1, j) \in A \tag{20}$$

$$y_{ij} \geq x_{(k-1)i} + x_{kj} - 1 \quad \forall\ (i, j) \in A \ \ \forall\, k = 2, \ldots, n \tag{21}$$

$$y_{ij} \in \{0, 1\} \quad \forall\, i = 1, \ldots, n \ \ \forall\ j = 1, \ldots, n \tag{22}$$

As before, the linear objective function (17) sums the received priority prizes less the transportation costs incurred in the sequence of node visits. The assignment constraints (2) – (4) refer only to the x variables, while the assignment constraints (18) – (19) to the y variables. Equalities (18) assure that the outflow from each node is unique, whereas equalities (19) state that the inflow to each node is unique. Constraints (20) – (21) are the coupling constraints of variables x and y in a similar way as constraints (11) – (12). And constraints (22) define the domain of the variables; we note that the integrality of the y-variables can be relaxed without loss of generality, given that the integrality of the x-variables is maintained in (4), and in this case the y-variables are viewed as flow variables.

Some of the existing *TSP* formulations are so-called extended formulations and provide information about the orderings of the nodes in addition to an optimal tour by means of auxiliary variables. One famous example is the MTZ formulation [26]. Model (17)-(22) can be modified to cope with the MTZ formulation by replacing the coupling constraints (20) - (21) by the constraints: $u_i - u_j + (n-1)y_{ij} \leq n-2$ for all $i,\ j = 2, \ldots, n$, $1 \leq u_i \leq n-1$ for all $i = 2, \ldots, n$ and $kx_{ki} \leq u_i$ for all $k, i = 1, \ldots, n$, where u_i is a positive auxiliary variable that returns the ordering of node i in the optimal tour. This is also a valid formulation for the *TSPPP*; however, different of model (17)-(22), the integrality of the y-variables can no longer be relaxed. As the linear relaxation of this model was not tighter than model (17)-(22) and its computational performance was not better than model (17)-(22) when solving the problem instances of Section 5, it was disregarded in this work. We also tested the linear ordering constraints $y_{ij} + y_{ji} \leq 1$ and $y_{ij} + y_{jl} + y_{li} \leq 2$ for all $i, j, l = 1, \ldots, n$ and $i \neq j, i \neq l, j \neq l$ in model (17)-(22), but the results did not improve.

Other MIP formulations presented in the literature for the *TDTSP* can also model the *TSPPP* by making some adjustments in their objective functions based on the redefinition of the time-dependent cost c_{ijk} as $c_{ij} - p_{ki}$, as discussed before. For instance, [25] presented a three-index formulation for the *TDTSP* based on binary variables y_{ijk} that specify whether node i is in position k and is followed immediatly by node j in the traveling salesman route. Four other different MIP models were proposed in [11] (pages 1019 and 1020) and in [15] (pages 72 and 74) also based on variables y_{ijk} (we note that the second model in [15] – model NO2 – becomes similar to the transportation flow formulation of previous subsection after some adjustments in its objective function). Some of these models will be considered in the computational experiments of Section 5, together with the MIP models presented in this section.

4 TABU SEARCH ALGORITHM FOR *TSPPP*

Given the intrinsic difficulties in solving larger *TSPPP* instances by exact methods, we describe a tabu search algorithm. The *TSPPP* can be seen as a permutation problem and our motivation for choosing a tabu search metaheuristic, instead of other possible metaheuristics proposed in the literature for solving permutation problems, is our previous experience with tabu search and its good performance when solving other permutation problems, as in [12] for the capacitated clustering problem and in [27] for the vehicle routing problem with time windows and multiple deliverymen. Our goal was to produce an algorithm capable to provide high quality solutions in short computational times for all instances tackled in this paper (Section 5) with few changes in parameters values. In fact, except for the tabu tenure, we use the single setting presented in this section. The convergence to a single setting required extensive experiments; even though other settings seem particularly favorable to specific instances, they do not work well for others.

The algorithm is based on the adaptive tabu search approach proposed in [12] and [27]. The approach consists of an integrated intensification/diversification mechanism that changes the tabu activation rule [13] according to search trajectory patterns (the curve of solution value vs. iteration) during local search. Basically, it assumes that general search trajectory patterns reflect the restrictiveness level imposed by tabu parameters values, e.g. by the tabu tenure and the tabu activation rule. Therefore, the main goal of the adaptive tabu search algorithm is to alter restrictiveness levels in order to intensify the exploration when trajectories identify possibly promising regions, and promote diversification if improvements seem to be miminal.

In order to identify the current trajectory pattern, the search process is dynamically divided into stages, and for each two consecutive stages $g-1$ and g, the average solution value in stage g (μ_g) is compared to the average solution value in stage $g-1$ (μ_{g-1}). If these average values are approximately the same and the coefficient of variation (ratio of the standard deviation σ to the mean of the two stages) is close to zero, the search describes a stagnated trajectory. On the other hand, if μ_g is larger than μ_{g-1}, an ascent trajectory is taking place. Similarly, a descent trajectory is identified if μ_g is smaller than μ_{g-1}.

Once the trajectory pattern is identified, changes in restrictiveness levels are prescribed for τh iterations, where h is an integer randomly generated within a pre-specified range $[h_{min}, h_{max}]$ and τ is a tuning factor associated to the observed trajectory pattern. Prior to trajectory evaluation, it is verified if any improvement with respect to the best solution found so far was obtained in the last stage; in this case, levels of restrictiveness and the tuning factor value are also set. That is, the approach reacts to improvement phases, stagnation, ascent trajectories and descent trajectories, which means that four possible tuning factor values are used in order to define the duration of restrictiveness levels changes. The period of application of a new setting corresponds to a new stage $g+1$ and once stage $g+1$ ends, a new trajectory pattern evaluation is performed using the average values of stages g and $g+1$, and so forth. Figure 3 illustrates the identification of a descent trajectory and the changes of restrictiveness levels suggested by the analysis. For the *TSPPP*, ascent trajectories and improvement phases indicate promising regions for which no

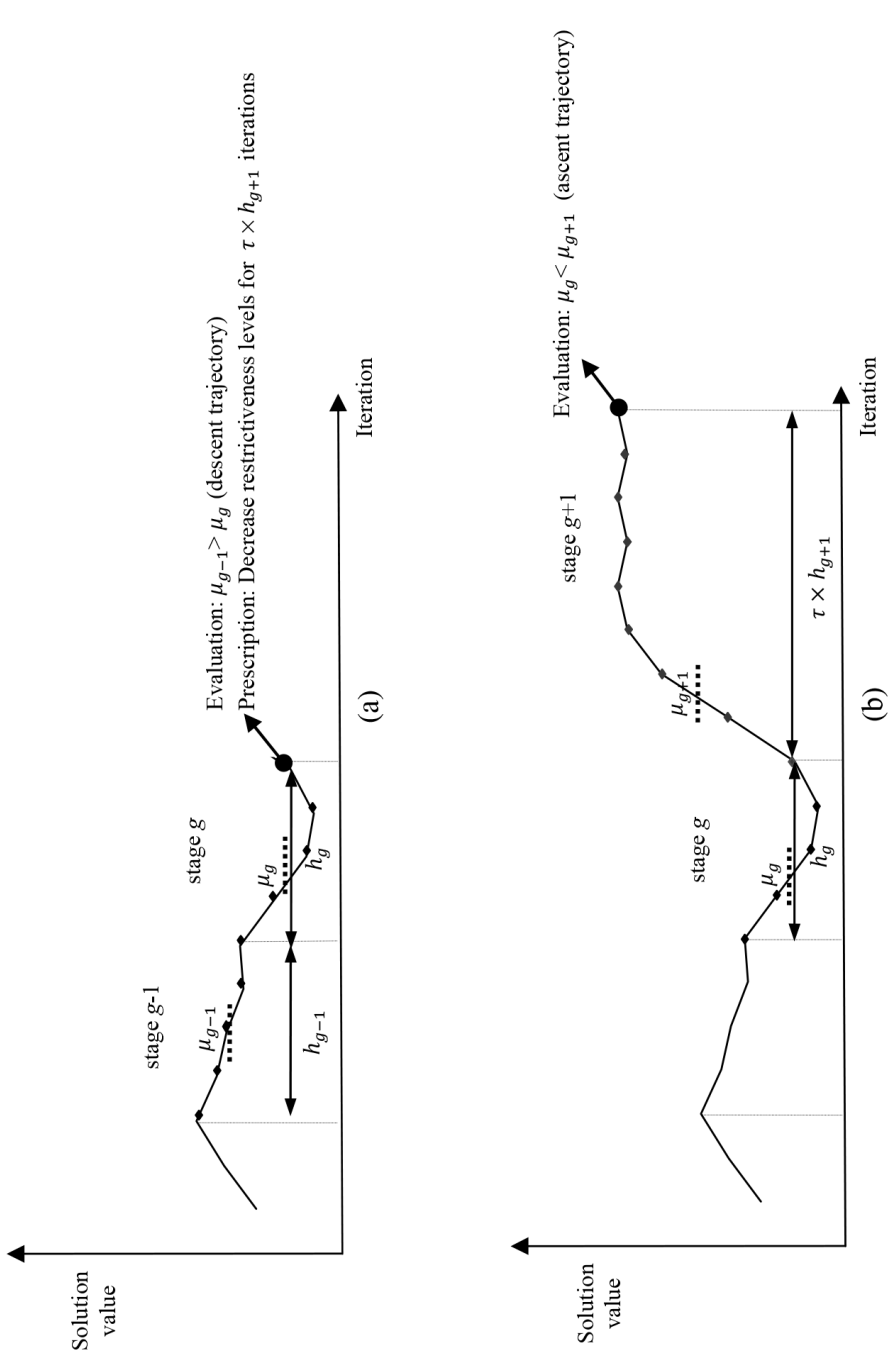

Figure 3 – Diversification/intensification mechanism: (a) Identification of a descent trajectory, resulting in the decrement of restrictiveness levels for $\tau \times h_{g+1}$ iterations; (b) Impact of decreasing restrictiveness levels in stage $g+1$. When a new pattern evaluation is performed, μ_{g+1} is compared to μ_g, indicating an ascent trajectory.

changes or lower restrictiveness levels are prescribed, while under descent trajectories, tabu restrictions are mildly relaxed to stop diversification. Finally, when search stagnation is observed, it is imposed high restrictiveness levels aiming at promoting diversification.

A high-level description of the Tabu Search algorithm (henceforth called TS) for the T*SPPP* is presented below:

1. Read the input data. Let n be the number of nodes to be routed.

2. (Tour construction) Starting from a partial tour S comprising a randomly chosen node i ($i \neq 1$) and node 1 fixed in the n-th order of the tour, expand S by selecting the unrouted node j and its insertion position that provides the greatest profit. Repeat the step until all nodes are routed.

3. Initialize the current iteration, set standard parameters and apply local search for two consecutive stages $g - 1$ and g.

4. Repeat until the maximum runtime T_{max} elapses:

4.1 If no solution improvement is obtained in stage g, identify the current trajectory pattern described by the stages $g - 1$ and g (as discussed in previous paragraphs of this section).

4.2 Set tabu parameters according to the trajectory pattern (discussed in Section 4.1) or improvement phase. Apply local search for the prescribed number of iterations, obtaining a new search stage $g + 1$. Make $g - 1 = g$ and $g = g + 1$.

5. Return the best found solution.

4.1 Local Search and the Adaptive Approach

Starting tours produced by the tour construction heuristic (step 2) are improved by applying two move types: (a) two-opt arc exchange and (b) order exchange of two nodes. Even though other types such as three-opt moves and the move of a single node to a different order in the tour have been tried, they usually delay or impede the convergence to the best solution found during the experiments in some instances.

Added (removed arcs) in recent moves are labeled as tabu-active for the adopted tabu tenure whenever a new move prescribe their removal (addition), and as mentioned in the previous section, restrictiveness levels are manipulated by the tabu activation rule. This rule prescribes the maximum number of tabu-active arcs for each of the two move types in a given search stage, here called tolerance. Note the smaller the tolerance, the more constrained the search process is. Tolerance values for two-opt moves are given by parameter TL, while tolerance values for exchange moves are given by TE. In step 3, initial tolerance values $(TL, TE) = (1, 1)$ are applied through the first two search stages. The first stage comprises all solutions between the starting solution and the first local optimum, while the second stage starts with the solution that follows

the first local optimum and ends with the solution found h iterations ahead and drawn from the range $[h_{min}, h_{max}] = [50, 100]$, also used to generate other stage lengths. The tabu tenure t for each added or deleted arc in a move is also randomly drawn from the range $[t_{min}, t_{max}]$, whose lower and upper limits depend on the range of the instance size (see Section 5).

In step 4.2, if any solution improvement is obtained in the last stage, tolerance values (TL, TE) and tuning factor τ are set to (1, 2) and 0.1, respectively. These tolerance values aim to intensify the exploration in possibly promising regions. Although they do not seem high, larger values almost invariably lead to cycling. We use low τ (resulting in relatively small number of iterations) because for some instances there is usually just a few subsequent improving moves.

Low tolerances $(TL, TE) = (0, 1)$ with $\tau = 0.5$ are imposed when search stagnation takes place, specifically, when the absolute value of the percent deviation of the solutions current stage average from the previous stage average is less than or equal to 5% and the coefficient of variation of the two stages average is less than or equal to 0.2. That is, stagnation is identified when:

$$|\frac{\mu_{g-1} - \mu_g}{\mu_{g-1}}| \leq 0.05, \quad \sigma/\mu \leq 0.2 \tag{23}$$

If the average solution value increases (an ascent trajectory and possibly promising region), no tolerance changes are applied and $\tau = 2$. Finally, if the average solution value decreases, tolerances are set to $(TL, TE) = (1, 1)$ and $\tau = 1$. Table 1 summarizes the adopted setting for all instances.

Table 1 – Parameter setting of algorithm TS used in the experiments.

Trajectory/phase	Identification	(TL, TE)	τ	$[h_{min}, h_{max}]$
Improving	Improvement	(1, 2)	0.1	[50, 100]
Stagnation	see (23)	(0, 1)	0.5	[50, 100]
Ascent	$\mu_{g-1} < \mu_g$	current	2	[50, 100]
Descent	$\mu_{g-1} > \mu_g$	(1, 1)	1	[50, 100]

5 COMPUTATIONAL RESULTS

In this section we present some computational results of the MIP models of Section 3 and the TS algorithm of Section 4 for solving *TSPPP* instances. For simplicity, the transhipment flow formulation, the transportation flow formulation and the two-assignment formulation are henceforth referred as models TF, TFr and KB-DFJ, respectively. This section also includes the results with the 3-index formulation in [25] (model 3PQ) and the first 3-index formulation in [11] (model FGG), both addressed to the *TDTSP* (the second formulation in [11] was not considered here as it has a weaker linear relaxation than FGG [15]). As discussed in Section 3, the objective function of these two models were appropriately modified in order to cope with the *TSPPP*.

For the analysis, we use some well-known examples from benchmark instances with 12 and 30 nodes (Nug12 and Nug30) provided in the QAPLIB database (`http://www.seas.upenn.`

edu/qaplib/inst.html) and with 51 and 100 nodes (Eil51 and KroA100) from TSPLIB95 (http://www.seas.upenn.edu/qaplib/inst.html), for which the *TSP* optimal solutions are known. The input data of these examples were slightly modified for the *TSPPP*: the original costs c_{ij} were maintained, either explicitly or computed from given node coordinates, while the prize values p_{ki}, depicted in Table 2, were generated. Although these literature examples have symmetric costs (i.e., $c_{ij} = c_{ji}$), the MIP models and the TS algorithm can be also applied to the more general non-symmetric case. Note that examples Nug12a and Nug12b are variations of Nug12 by simply modifying the values p_{ki} (similarly for Nug30a-Nug30d, and Eil51a-Eil51e). In order to impose node 1 as the last one visited (that is, at order $k = n$) in algorithm TS, p_{n1} was set to a sufficiently large number when compared to the other prizes (e.g., $> 1,000$).

Table 2 – Prize values in examples.

Example	Prizes
Nug12a	$p_{ki} = 2 \ \forall k = 1 \ldots 12, i = 1 \ldots, 12$
Nug12b	$p_{ki} = 2 \ \forall k = 1 \ldots 12, i = 1 \ldots, 12$, except $p_{1,7} = p_{2,10} = 5$
Nug30a	$p_{ki} = 2 \ \forall k = 1 \ldots 30, i = 1 \ldots, 30$
Nug30b	$p_{ki} = 2 \ \forall k = 1 \ldots 30, i = 1 \ldots, 30$, except $p_{1,10} = p_{2,20} = 5 \ p_{3,30} = 10$
Nug30c	p_{ki} randomly sorted in [2, 10] $\forall k = 1 \ldots 30, i = 1 \ldots, 30$
Nug30d	p_{ki} randomly sorted in [10, 100] $\forall k = 1 \ldots 30, i = 1 \ldots, 30$
Eil51a	$p_{ki} = 10 \ \forall k = 1 \ldots 51, i = 1 \ldots, 51$
Eil51b	$p_{ki} = 10 \ \forall k = 1 \ldots 51, i = 1 \ldots, 51$, except $p_{1,6} = 21$
Eil51c	p_{ki} randomly sorted in [10, 20] $\forall k = 1 \ldots 51, i = 1 \ldots, 51$
Eil51d	p_{ki} randomly sorted in [20, 100] $\forall k = 1 \ldots 51, i = 1 \ldots, 51$
Eil51e	p_{ki} randomly sorted in [100, 1000] $\forall k = 1 \ldots 51, i = 1 \ldots, 51$
KroA100	$p_{ki} = 300 \ \forall k = 1 \ldots 100, i = 1 \ldots, 100$

The modeling language GAMS with the optimization solver CPLEX 12.5 was used to implement and solve the MIP models. The CPLEX branch-and-cut was executed on Windows 7 with all its default parameters and within a runtime limit of 1000 seconds. TS was coded in Delphi 7 and run on Windows 7. Given the probabilistic nature of the algorithm, five runs starting from different initial solutions were performed within a runtime limit T_{max} of 100 seconds per run and under the parameter settings depicted in Section 4.1. The tabu tenure t for each added or deleted arc in a move was randomly drawn from the range $[t_{min}, t_{max}] = [10, 20]$ for all Nug12, Nug30 and Eil51 instances, and from the range $[t_{min}, t_{max}] = [50, 60]$ for Kroa100 instance. All experiments were conducted on a notebook Intel Core i7 2.00 GHz with 6 GB RAM.

Regarding examples Nug12a and Nug12b with $n = 12$ nodes, all models TF, TFr, KB-DFJ, FGG and 3PQ were able to find an optimal solution and prove its optimality in a few seconds. In Nug12a, the c_{ij} values vary from 0, 1, ..., 5 and all p_{ki} are equal to 2, i.e. they are constant and independent of order k and node i. Therefore, the optimal solution of Nug12a with value 12 correctly gets a total prize of $2n = 24$ and as expected, follows the minimum cost route

of the corresponding *TSP*, known as 12. In case of Nug12b, as $p_{1,7} = p_{2,10} = 5$, there are priority prizes of value 5 if the traveling salesman visits nodes 7 and 10 in the first and second orders of its route, respectively. In fact, the optimal solution of Nug12b with value 14 involves the collection of these higher prizes in nodes 7 and 10 in these required orders, obtaining a total prize of 30 but paying a higher transportation cost than before, equal to 16. Figure 4 illustrates the optimal solutions of examples Nug12a and Nug12b found with model TFr (there are many other alternative optimal solutions for these examples). These solutions were also obtained by algorithm TS in less than one second.

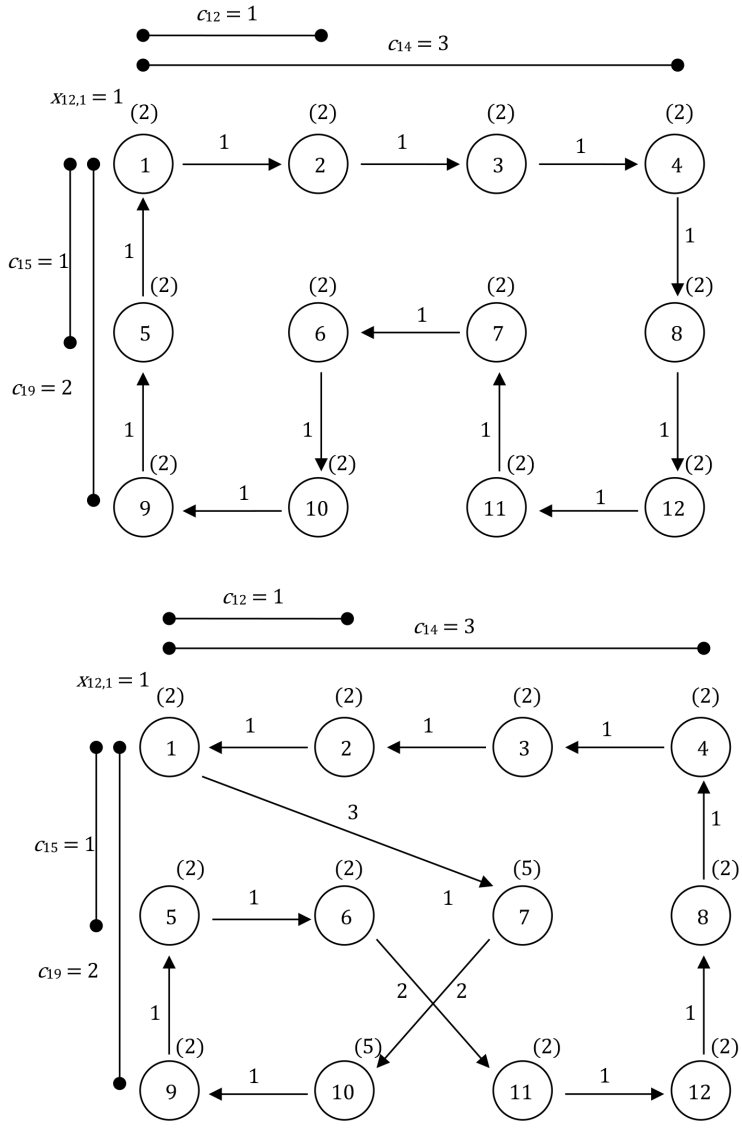

Figure 4 – Optimal routes of examples Nug12a and Nug12b.

As we move to the larger examples Nug30a and Nug30b with $n = 30$ nodes, all models were also able to find an optimal solution and prove its optimality, however, the computer required runtimes vary from a few seconds to several minutes or even hours. In Nug30a, the c_{ij} values vary from 0, 1, ..., 9 and all p_{ki} are equal to 2. Its optimal solution with value 30 correctly gets a total prize of $2n = 60$ and follows the minimum cost route of the corresponding *TSP*, known as 30. In case of Nug30b, as $p_{1,10} = p_{2,20} = 5$, $p_{3,30} = 10$, there are priority prizes of values 5, 5 and 10 if the traveling salesman visits nodes 10, 20 and 30 in the first, second and third orders of its route, respectively. The optimal solution of Nug30b with value 35 shows that it does not pay off to get the priority prize in node 10 in the first visit of the route, despite the high value of this prize. However, it is worth collecting the priority prizes in nodes 20 and 30 in the second and third visits of the route, obtaining a total prize of 71 but paying a higher transportation cost than before, equal to 36.

Table 3 summarizes the results obtained for examples Nug30a and Nug30b. Columns Total profit, Collected prizes, Travel costs, Gap and Time show the best solution values, the optimality gaps (%) and the CPU runtimes (in seconds) required by GAMS/CPLEX and TS when applied to these examples. The gaps were computed accordingly to the formula used by CPLEX (the difference between the best upper bound provided by CPLEX and the best solution value, divided by the best solution value) and the symbol "> 100" means gaps larger than 100%. The last two columnns LR and LRTime of the table show respectively the linear relaxation bounds and the CPU runtimes (in seconds) obtained by solving the linear formulations without their variable integrality constraints. The solution values for TS were already decremented by the adopted p_{n1}, so that they can be directly compared to GAMS/CPLEX results. The values in brackets correspond to the worst and best solution values found by TS in five runs, the gaps are computed based on the best upper bound found by CPLEX and the runtimes reported are the five TS run average. Note that besides TS, only models TFr and 3PQ were able to optimally solve both examples in a few seconds, which are also the formulations with best RL bounds for these and the other examples.

At the end of section 2 we have observed that as the p_{ki} values become larger and with higher dispersion and the c_{ij} values become smaller and with smaller dispersion, one would expect that the *TSPPP* becomes easier to solve, as it tends to the *LAP*, which is an easy problem. In order to verify this comment, Table 4 compares the model performances when solving examples Nug30c and Nug30d with randomly generated p_{ki} values in the intervals [2,10] and [10,100], respectively (the sorted values were then rounded to the nearest integer). Note that the p_{ki} values of example Nug30d are relatively large if compared to the values of example Nug30c, while the c_{ij} values of both examples are the same. Then we expect that Nug30d should be less difficult to solve than Nug30c, and that Nug30c should be less difficult to solve than Nug30a and Nug30b, which are confirmed by the model results of Table 4. Again, models TFr and 3PQ had better performances than the other models. In fact, these results would be expected because: (i) model TF has a weaker linear relaxation than model TFr as discussed in section 3, (ii) model FGG has a weaker linear relaxation than model 3PQ, which is equivalent to model NO2 (presented in [15] for the *TDTSP*) in terms of linear relaxations, as pointed out in [15], (iii) models NO2 and TFr

Table 3 – Results of examples Nug30a and Nug30b (*Proven optimal solution).

Example	Solution approach	Total profit	Collected prizes	Travel costs	Gap (%)	Time (sec)	LR (%)	LRTime (sec)
Nug30a	TF	22	60	38	> 100	1000	50.6	0.1
Nug30a	TFr	30*	60	30	0.0	7.2	30.0	0.4
Nug30a	KB-DFJ	30*	60	30	0.0	385.9	30.0	0.2
Nug30a	FGG	30*	60	30	0.0	4.3	30.0	0.2
Nug30a	3PQ	30*	60	30	0.0	3.6	30.0	0.7
Nug30a	TS	[30*,30*]	60	30	0.0	0.1		
Nug30b	TF	29	71	42	8.2	1000	54.0	0.1
Nug30b	TFr	35*	71	36	0.0	10.5	35.0	0.3
Nug30b	KB-DFJ	35*	71	36	0.0	995.6	38.0	0.4
Nug30b	FGG	33	71	38	6.0	1000	40.6	0.2
Nug30b	3PQ	35*	71	36	0.0	5.5	35.0	0.9
Nug30b	TS	[35*,35*]	71	36	0.0	0.1		

Table 4 – Results of examples Nug30c and Nug30d (*Proven optimal solution).

Example	Solution approach	Total profit	Collected prizes	Travel costs	Gap (%)	Time (sec)	LR (%)	LRTime (sec)
Nug30c	TF	232*	286	54	0.1	1000	247.8	0.2
Nug30c	TFr	232*	286	54	0.0	12.4	238.8	0.2
Nug30c	KB-DFJ	232*	286	54	0.9	1000	258.0	0.1
Nug30c	FGG	227	279	52	7.8	1000	261.4	0.2
Nug30c	3PQ	232*	286	54	0.0	5.7	238.8	0.4
Nug30c	TS	[232*,232*]	286	54	0.0	0.1		
Nug30d	TF	2750*	2830	80	0.0	0.6	2755.2	0.1
Nug30d	TFr	2750*	2830	80	0.0	0.4	2751.5	0.1
Nug30d	KB-DFJ	2750*	2830	80	0.0	18.6	2785.4	0.1
Nug30d	FGG	2750*	2830	80	0.0	19.5	2799.2	0.1
Nug30d	3PQ	2750*	2830	80	0.0	1.1	2751.5	0.1
Nug30d	TS	[2750*,2750*]	2830	80	0.0	0.1		

are similar after some adjustments in their objective functions. Note that in all of its five runs, algorithm TS was able to optimally solve each of these examples in less than one second.

The limitations of GAMS/CPLEX for solving the MIP models are more evident in examples Eil51a and Eil51b, with $n = 51$ nodes. None of the models were able to solve these examples within the runtime limit of 1000 seconds (and not even in one hour). The best feasible solutions among the models were found with TFr and 3PQ, but still with high optimality gaps (Table 5). Note that for some cases the collected prizes are lower than the travel costs, resulting in negative total profits.

Table 5 – Results of examples Eil51a and Eil51b (*Proven optimal solution).

Example	Solution approach	Total profit	Collected prizes	Travel costs	Gap (%)	Time (sec)	LR (%)	LRTime (sec)
Eil51a	TF	–94	510	604	> 100	1000	457.7	0.4
Eil51a	TFr	22	510	488	> 100	1000	126.2	1.3
Eil51a	KB-DFJ	–213	510	723	> 100	1000	133.0	3.9
Eil51a	FGG	–198	510	708	> 100	1000	127.2	1.1
Eil51a	3PQ	34	510	476	> 100	1000	126.2	1.2
Eil51a	TS	[84*,84*]	510	426	0.0	4.2		
Eil51b	TFr	39	510	471	> 100	1000	127.4	1.4
Eil51b	3PQ	66	521	455	58.7	1000	127.4	1.3
Eil51b	TS	[85,85]	521	436	23.2	9.8		

Differently from the MIP models, TS is able to find the best solutions for these examples in a few seconds. The TS solution of example Eil51a with profit 84 is also optimal as all prizes of this example are equal (total of 510) and the route of this solution corresponds to the minimum cost route of the *TSP*, with value 426. In this solution, node 6 is visited at the 37th order of the route, and this same solution is found if we change $p_{1,6}$ from 10 to any integer value ranging from 11 to 20; however, when $p_{1,6} = 21$ (as prescribed in example Eil51b), it more profitable to serve node 6 at the first order of the route.

Similarly to the results observed in Table 4 for examples Nug30c and Nug30d, as we increase the p_{ki} values relatively to the c_{ij} values, examples Eil51 become easier to solve with the models. Table 6 compares the model performances when solving examples Eil51c, Eil51d and Eil51e with randomly generated p_{ki} values in the intervals [10,20], [20,100] and [100,1000], respectively (the sorted values were then rounded to the nearest integer). Note that the p_{ki} values of Eil51e are relatively large if compared to the ones of Eil51d, which are relatively large if compared to the ones of Eil51c, which are large if compared to Eil51a and Eil51b, while the c_{ij} values of all these examples are the same. As expected, the results of Table 6 reinforce that for the models, Eil51e is easier to solve than Eil51d, which is easier to solve than Eil51c, and so forth. Note that differently from the MIP models, again TS is able to find the best solutions for all examples in a few dozens of seconds.

Regarding the performance of TS, Table 7 summarizes additional information (columns Best TS and Mean TS) along with the best solution provided by all exact approaches for each example (column Best model). Column z presents the best solution value of all models and columns z_b, $Time_b$, $\bar{z}$ and σ_z correspond to the best solution value, the shortest runtime to the best solution value, the average solution value and the standard deviation of the solution values of the five TS runs, respectively. Column $\overline{Gain}$ is the average improvement from the starting solutions in the five TS runs. We observe that the average improvement can be larger than the mean best total profit, for example, for Nug12a in the first line of the table, $\overline{Gain} = 13.2$ and $\bar{z} = 12$ because the mean starting solution value is negative (-1.2). Finally, column $\overline{Time}$ is the average runtime

Table 6 – Results of examples Eil51c, Eil51d and Eil51e (*Proven optimal solution).

Example	Solution approach	Total profit	Collected prizes	Travel costs	Gap (%)	Time (sec)	LR (%)	LRTime (sec)
Eil51c	TFr	412	903	491	17.6	1000	492.4	1.7
Eil51c	3PQ	371	897	526	30.7	1000	492.4	1.4
Eil51c	TS	[424,430]	890	460	12.8	41.4		
Eil51d	TFr	4045*	4769	724	0.0	836	4108.8	1.6
Eil51d	3PQ	4045*	4769	724	0.5	1000	4108.8	1.6
Eil51d	TS	[4043,4045*]	4769	724	0.0	41.1		
Eil51e	TFr	47971*	49251	1280	0.0	2.6	48006.8	0.6
Eil51e	3PQ	47971*	49251	1280	0.0	14.4	48006.8	1.6
Eil51e	TS	[47949,47971*]	49251	1280	0.0	66.3		

to the best solution of the five runs. Note in column Best TS that in addition to finding proven optimal solutions in 8 examples, for instances Eil51a and Eil51b, the large percent increase presented by TS in the models best solution (column $\frac{z_b - z}{z}$) reinforces the observed difficulties of the previous exact approaches when prizes become lower and less dispersed.

Table 7 – Performance comparison and additional information (*Proven optimal solution).

Example	Best model		Best TS			Mean TS			
	z	Time (sec)	z_b	$Time_b$ (sec)	$\frac{z_b - z}{z}$ (%)	$\bar{z}$	σ_z	$\overline{Gain}$	$\overline{Time}$ (sec)
Nug12a	12*	0.1	12*	0.0	0.0	12.0	0.0	13.2	0.0
Nug12b	14*	0.1	14*	0.0	0.0	14.0	0.0	17.6	0.0
Nug30a	30*	3.6	30*	0.1	0.0	30.0	0.0	77.8	0.1
Nug30b	35*	5.5	35*	0.1	0.0	35.0	0.0	80.2	0.1
Nug30c	232*	5.7	232*	0.1	0.0	232.0	0.0	159.0	0.1
Nug30d	2750*	0.4	2750*	0.1	0.0	2750.0	0.0	1175.6	0.1
Eil51a	34	1000	84*	0.1	147.1	84.0	0.0	1230.2	4.2
Eil51b	66	1000	85	1.0	28.8	85.0	0.0	1197.6	9.8
Eil51c	412	1000	430	17	4.4	428.8	2.4	1298.4	41.4
Eil51d	4045*	836	4045*	11	0.0	4043.8	1.0	2549.8	41.1
Eil51e	47971*	2.6	47971*	46	0.0	47957.2	9.1	21497.2	66.3

Note also that the simplicity of the tour construction heuristic (step 2 in Section 4) often results in low quality starting solutions, therefore the high average improvement, short average times to the best solution, high means and low standard deviations of the five run incumbents (columns $\overline{Gain}$, $\overline{Time}$, $\bar{z}$ and σ_z, respectively) produced by the improvement phase (step 4 in Section 4) reveal that TS is little sensitive to the starting solutions for this set of instances. On the other hand, $\bar{z}$, $\overline{Time}$ and σ_z get worse as the prizes increase and become more dispersed. That is, differently

from the exact approaches, TS tends to have a better performance (higher quality best solutions, shorter average time to the best solution and lower standard deviation among the best solutions) when prizes are homogeneous or exhibit low variation.

We also analyze the performance of plain local search by applying 2-opt moves to the same five starting solutions. Even though improvements are substantial (reaching 14075.0 for instance eil51e), 2-opt solutions show total profit reductions between 12.7% and 40.8% when compared to TS. The experiments also show an average standard deviation of 103.7, which is considerably higher than what is found with TS. These results suggest that the solution landscapes are characterized by local optima of widely varying quality, as one might have expected. As the goodness of solutions obtained by local search is highly dependent on the starting solution, a method capable to cross the barriers between local optima (such as TS), appears as a much better alternative.

While all models were unable to obtain solutions with positive total profit for example KroA100 within the time limit of 1000 seconds, TS found a solution of value 8718 in 97 seconds, which is optimal as all prizes are identical (total of 30000) and the route of this solution corresponds to the minimum cost route of the *TSP*, with value 21282. The last three nodes visited by the travelling salesman in this tour before returning to the depot are nodes 27, 92 and 46, respectively. If the customers of nodes 27 and 92 double their priority prizes for being visited in the last position of the tour (i.e., $p_{99,27} = 600$, $p_{99,92} = 600$ and all remaining prizes are $p_{ki} = 300$), the best route found by TS visits node 92 in the last order (just after visiting node 27 in the penultimate order), obtaining a higher profit ($30300 - 21295 = 9005$) than before ($30000 - 21282 = 8718$), but with a slightly higher travelling cost. If on the other hand, the customer of node 27 doubles the priority prize for being visited in the first order of the tour (i.e., $p_{1,27} = 600$, $p_{99,92} = 600$ and all remaining prizes are $p_{ki} = 300$), the best found solution does not change and node 27 is still visited in the penultimate position. However, if customers of nodes 27 and 92 pay priority prizes of $p_{1,27} = 900$, $p_{99,92} = 900$ for being visited in the first and last order of the tour, respectively, then the best found route follows these orders to collect these higher prizes and obtain higher profits ($31200 - 21767 = 9433$).

These and the former examples illustrate the performance of the models and algorithm TS when solving the *TSPPP*. We also replaced the assignment equalities (2) – (3) by corresponding assignment inequalities in models TF and TFr, but the results obtained for all these examples in most cases did not improve.

6 CONCLUDING REMARKS

A traveling salesman is basically motivated to make a tour in order to sell commodities. Cost minimization is only a consequence of a maximal profit objective. This point of view has been explored in different studies to cope with situations in which the standard approach of transportation cost minimization not necessarily implies the salesman maximal profit objective. This paper studies the particular situation where different priority prizes are paid by the customers depending on the order they are visited in the traveling salesman route, called the Traveling Salesman Problem with Priority Prizes (*TSPPP*). This type of salesman benefit is detached in

the linear parcel of an integer quadratic program that formalizes the problem. To the best of our knowledge, there is no other studies in the literature that have directly dealt with the *TSPPP*. To cope with the node visit order, we explore a representation for the *TSPPP* grounded on the *QAP* in Koopmans and Beckmann [18]. We present different MIP models derived from this *QAP* and other models to deal with the *TSPPP*, as well as an effective tabu search algorithm, capable to solve larger problem instances.

An interesting perspective for future research would be to investigate the use of more effective *TDTSP* formulations to deal with the *TSPPP* [1, 16], as well as to explore the application of *TSPPP* approaches in real-life situations, such as in single machine scheduling problems with priority prizes and in the planning of touristic tours [29]. Another future research would be to compare the present tabu search algorithm with other metaheuristics for this problem, such as variable neighborhood search and evolutionary algorithms. Furthermore, the condition imposing the visit to all nodes in the traveling salesman route may not be pertinent for some applications, thus another interesting line of research would be to extend the present approaches to deal with the prize-collecting version of the *TSPPP*.

ACKNOWLEDGEMENTS

The authors would like to thank the two reviewers for their useful comments and suggestions of revision, and CNPq and FAPESP for the financial support.

REFERENCES

[1] ABELEDO H, FUKASAWA R, PESSOA A & UCHOA E. 2013. The time dependent traveling salesman problem: polyhedra and algorithm. *Math. Prog. Comp.*, **5**: 27–55.

[2] APPLEGATE DL, BIXBY RE, CHVATAL V & COOK WJ. 2006. The Traveling Salesman Problem: A Computational Study. Princeton University Press.

[3] BALAS E. 1989. The prize-collecting traveling salesman problem. *Networks*, **19**(6): 621–638.

[4] BIANCO L, MINGOZZI A & RICCIARDELLI S. 1993. The traveling salesman problem with cumulative costs. *Networks*, **23**(2): 81–91.

[5] CHRISTOFIDES N. 1979. The Traveling Salesman Problem. Wiley Chichester.

[6] CONWAY R, MAXWELL W & MILLER L. 1967. Theory of Scheduling. Addison-Wesley.

[7] DANTZIG R, FULKERSON R & JOHNSON S. 1954. Solution of a large-scale traveling salesman problem. *Operations Research*, **2**: 393–410.

[8] DANTZIG G & RAMSER J. 1959. The truck dispatching problem. *Operations Research*, **12**: 81–91.

[9] FEILLET D, DEJAX P & GENDREAU M. 2005. Traveling salesman problem with profits. *Transportation Science*, **39**: 188–205.

[10] FISCHETTI M, LAPORTE G & MARTELO S. 1993. The delivery man problem and cumulative methods. *Operations Research*, **6**: 1055–1064.

[11] FOX K, GAVISH B & GRAVES S. 1980. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, **28**: 1018–1021.

[12] FRANCA PM, SOSA NM & PUREZA V. 1999. An adaptive tabu search algorithm for the capacitated clustering problem. *International Transactions in Operational Research*, **6**: 665–678.

[13] GLOVER F & LAGUNA M. 1997. Tabu Search. Kluwer Academic Publishers, Massachusetts.

[14] GOEMANS M & KLEINBERG J. 1998. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, **82**: 114–124.

[15] GOUVEIA L & VOSS S. 1995. A classification of formulations for the (time-dependent) traveling salesman problem. *European Journal of Operational Research*, **83**: 69–82.

[16] GODINHO MT, GOUVEIA L & PESNEAU P. 2014. Natural and extended formulations for the time-dependent traveling salesman problem. *Discrete Applied Mathematics*, **164**: 138–153.

[17] GRUNDEL DA & JEFFCOAT DE. 2004. Formulation and solution of the target visitation problem. Proceedings of the AIAA 1st Intelligent Systems Technical Conference.

[18] KOOPMANS TC & BECKMANN M. 1957. Assignment problems and the location of economic activities. *Econometrica*, **25**: 53–76.

[19] HEFFLEY DR. 1972. The quadratic assignment problem: A note. *Econometrica*, **40**: 1155–1163.

[20] HELD M & KARP RM. 1971. The traveling salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, **1**: 6–25.

[21] LAWLER EL. 1963. The quadratic assignment problem. *Management Science*, **19**: 586–599.

[22] LUCENA A. 1990. Time-dependent traveling salesman problem – the deliveryman case. *Networks*, **20**: 753–763.

[23] MIRANDA G, LUNA HP, MATEUS GR & FERREIRA RPM. 2005. A peformance guarantee heuristic for electronic components placement problems including thermal effects. *Computers and Operations Research*, **32**: 2937–2957.

[24] MILLER CE, TUCKER AW & ZEMLIN RA. 1960. Integer programming formulation of traveling salesman problems. Journal of the Association for Computing Machnery, **7**: 326–329.

[25] PICARD JC & QUEYRANNE M. 1978. The time-dependent traveling salesman problem and its application to the tardiness problem in one machine scheduling. *Operations Research*, **26**: 86–110.

[26] ONCAN T, ALTINEL IK & LAPORTE G. 2009. A comparative analysis of several assymetric traveling salesman problem formulations. *Computers and Operations Research*, **36**: 637–654.

[27] PUREZA V, MORABITO R & REIMANN M. 2012. Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the VRPTW. *European Journal of Operational Research*, **218**: 636–647.

[28] SARUBBI JFM & LUNA HPL. 2007. The multi-commodity traveling salesman problem. INOC – International Network Optimization Conference, April 2007. `http://www.poms.ucl.ac.be/inoc2007/Papers/author.68/paper/paper.68.pdf`.

[29] SILVA AA, MORABITO R & PUREZA V. 2018. Optimization approaches to support the planning and analysis of travel itineraries. *Expert Systems with Applications*, **112**: 321–330.