Applied Mathematical Modelling 37 (2013) 2257–2266

Contents lists available at SciVerse ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Two improved formulations for the minimum latency problem

Francisco Angel-Bello [a], Ada Alvarez [b], Irma García [c,*]

[a] Tecnológico de Monterrey, Campus Monterrey, Department of Industrial and Systems Engineering, Ave. Eugenio Garza Sada 2501, Monterrey, Nuevo León CP 64849, Mexico
[b] Universidad Autónoma de Nuevo León, Graduated Program in Systems Engineering, Av. Universidad s/n. San Nicolás, Nuevo León CP 66451, Mexico
[c] Universidad Autónoma de Coahuila, Research Center on Applied Mathematics, Edif. "S" Campo Redondo. Saltillo, Coahuila CP 25280, Mexico

ARTICLE INFO

Article history:
Received 13 September 2011
Received in revised form 5 May 2012
Accepted 18 May 2012
Available online 28 May 2012

Keywords:
Directed minimum latency problem
Repairman problem
Deliveryman problem
Integer formulations
Computational evaluation

ABSTRACT

In network problems, latency is associated with the metric used to evaluate the length of the path from a root vertex to each vertex in the network. In this work we are dealing with two applications or variations of the minimum latency problem known as the repairman problem and the deliveryman problem. We have developed two integer formulations for the minimum latency problem and compared them with other two formulations from the literature for the time-dependent traveling salesman problem. The present work highlights the similarities and differences between the different formulations. In addition, we discuss the convenience of including a set of constraints in order to reduce the computation time needed to reach the optimal solution. We have carried out extensive computational experimentation on asymmetrical instances, since they provide the characteristics of the deliveryman and repairman problems in a better way.

1. Introduction

In network problems, the latency is associated with the metric used to evaluate the length of the path from a root vertex to each vertex in the network. In the minimum latency problem (MLP), the goal is to find a Hamiltonian path starting from a root node that minimizes the sum of path lengths to all vertices.

In the context of distribution and scheduling there is a group of problems that can be treated as a MLP: the deliveryman problem, the repairman problem, the total completion time problem and the total flow time problem.

In the deliveryman and repairman problems, the locations of clients and agents as well as the service times at clients are known. In both cases, the agent has to visit clients to perform some service and the objective is to find a route that minimizes the clients total waiting time. The basic difference between these two problems is that delivery service times are smaller than repair service times. These problems can be seen as "customer-centric" routing problems because the objective function minimizes the total waiting time of all customers rather than the total travel time of the vehicle.

In machine scheduling context, the problem is related to minimize the total completion time or total flow time of jobs that will be processed on a single machine.

In single machine scheduling environment, when sequence-dependent setup times are considered, the problems related to minimize the total completion time or the total flow time of jobs that will be processed can be formulated in a similar way. In these cases, travel times t_{ij} correspond to the required setup time of the machine to process the job j right after job i. The service times correspond to the processing times of jobs and the objective is to find a permutation of jobs that minimizes the

* Corresponding author. Tel./fax: +52 844 4101242.
E-mail addresses: fangel@itesm.mx (F. Angel-Bello), ada.alvarezs@uanl.mx (A. Alvarez), idgarcia@cima.uadec.mx, irma.igarcia@gmail.com (I. García).

0307-904X/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.apm.2012.05.026

total flow time or the total completion time of jobs. In fact, the minimum latency problem arises from this kind of problem [1–3].

Besides these straight applications of the MLP, other applications can be found in disk-head scheduling [4], in the routing of automated guided vehicles in a flexible manufacturing system [5], and in the process of information search in computer networks [6].

It has been established that MLP is NP-hard for general metrics [7] and when the metric space is induced by a tree [8]. In spite of its similarities with the traveling salesman problem (TSP), the MLP is much harder to solve or approximate, from a computational point of view, than the TSP [9].

Research has been conducted to develop approximations for the MLP [10–12,4,9] and the current best approximation ratio for undirected graph is 3.59 [13]. In the directed case, Nagarajan and Ravi [14] propose an approximating algorithm with a guarantee of $O(n^{1/2+\varepsilon})$, for any constant $\varepsilon > 0$. This result is overcome in the work of Friggstad et al. [15], where the authors proposed an $O(\log n)$-approximation to the directed latency problem.

Some authors have proposed exact optimization algorithms to solve variants of the MLP. Lucena [16] developed an enumerative procedure based on a nonlinear integer programming formulation using a Lagrangian relaxation to obtain lower bounds. In this work instances up to 30 nodes were solved. Fischetti et al. [17] proposed an integer programming formulation and developed a branch-and-bound algorithm using a matroidal structure of the problem to obtain lower bounds. They solved instances up to 60 nodes using this procedure. Wu et al. [18] obtained exact solutions up to 23 nodes by using dynamic programming with pruning.

Méndez-Díaz et al. [19] proposed an integer programming formulation for the traveling deliveryman problem taking advantage of its connections with the linear ordering problem. They presented a cutting plane algorithm that uses valid inequalities associated with the proposed formulation. They compared the lower bounds obtained by their linear programming relaxation with the corresponding counterparts in Fischetti et al. [17] and Eijl [20].

Sarubbi et al. [21] implemented, for the minimum latency problem, the model developed by Picard and Queyranne [3] for the time-dependent traveling salesman problem and solved asymmetric instances up to 80 nodes using professional solver Cplex 9.1.3 and Newton's Barrier Method to compute the linear programming lower bounds. The problem was formulated as a shortest path problem with side constraints on a multi-level graph, where nodes represent the potential position of each customer in the tour.

In general, in minimum latency problems, the cost, distance or time matrices can be both symmetric and asymmetric, but in the above mentioned applications, if the service or processing times are different for each customer or job, the time matrices are always asymmetrical.

In the revised literature we have found that, in general, computational experiments are performed on instances in which symmetric time matrices were generated from coordinates in the plane or on instances in which asymmetric and symmetric time matrices were randomly generated using the uniform distribution followed by triangularization through shortest paths.

When symmetric matrices are used, it is considered that the service or processing times are the same for each client or job. In randomly generated asymmetric instances, it can be assumed that different service or processing times are being considered for each client or job. However, when elements of the matrix are forced to satisfy the triangle inequality the range of the data (largest value–smallest value) is drastically reduced obtaining instances with very little variability. In general, this last type of instances is easier to solve. They will be used in the present work to show that they have lower complexity than instances generated from coordinates.

In this paper we are focusing on two applications or variations of the minimum latency problem: the repairman problem and the deliveryman problem. We propose two linear integer formulations for the minimum latency problem with asymmetric costs and compare them with formulations with the best performance found in the scientific literature, that is, Méndez-Díaz et al. [19], Gouveia and Voss [22] and the implementation achieved by Sarubbi et al. [21] of Picard and Queyranne formulation [3]. We investigate the similarities and differences between these formulations and evaluate the convenience of including a set of constraints that help to reduce the computation time to reach the optimal solution. We complement our work with computational experimentation on asymmetrical instances since they provide the characteristics of the deliveryman and repairman problems in a better way. Finally, we investigate sets of valid inequalities which improve the value of the linear relaxation of the proposed models.

2. Problem formulation

Let $G = (V, A)$ be a complete directed graph where $V = \{0, 1, 2, \ldots, n\}$ is the node set and A is the arc set. Node 0 corresponds to root node. Let $C = (c_{ij})$ a matrix of nonnegative weights (costs) associated with arcs $(i, j) \in A$.

For a given permutation (Hamiltonian path) of nodes $P = 0[1][2]\cdots[n]$ of V, with node 0 in the first position, the latency $l_{[i]}$ of the node in position i is defined as the length of the path from the root node 0 to node $[i]$ and can be calculated as $l_{[i]} = \sum_{j=1}^{i} c_{[j-1][j]}$, where $[i]$ is the index of node in position i in the path, $[0] = 0$ and $l_{[0]} = l_0 = 0$. The total latency for a given permutation P is the sum of the latencies of all the nodes and can be calculated as

$$L = \sum_{i=1}^{n} l_{[i]} = \sum_{i=1}^{n} (n - i + 1) c_{[i-1][i]}.$$

The objective of the minimum latency problem is to find a permutation of nodes that minimizes the total latency.

From this last expression it can be seen that MLP is a particular case of the time dependent traveling salesman problem, where the contribution of each city to the objective function depends on the position it occupies in the tour.In the delivery-man and repairman problems the values c_{ij} are made up of the travel times t_{ij} and the service times s_i and are calculated in the following way:

$$c_{ij} = \begin{cases} t_{0j}, & \text{for } i = 0;\ j = 1,2,\ldots,n; \\ s_i + t_{ij}, & \text{for } i = 1,2,\ldots,n;\ j = 0,1,\ldots,n;\ j \neq i \end{cases}$$

Note that if clients i and j have different service times, then $c_{ij} \neq c_{ji}$.

3. Integer linear formulations for the directed minimum latency problem

In order to provide the mathematical models, we will use the multi-level network shown in Fig. 1, where n copies of nodes $\{1, 2, \ldots, n\}$ are depicted. As node 0 should occupy the first position in all permutations, this position will be referred to as position 0 and corresponds to level 0 on the graphic. Other positions (levels in the graphic) will be referred to as positions $1, 2, \ldots, n$. This graphical representation of the latency problem is similar to the Picard and Queyranne representation [3] for time-dependent traveling salesman problems.

Note that each permutation can be considered as a Hamiltonian path that starts at the root node and visits the nodes in the order indicated by the permutation. In the graphical representation this means that only one node should be selected on each level and it should be different from those nodes selected on other levels.

The contribution to the objective function from each used arc in a feasible solution depends on the levels connected by this arc in the network. If the arc (i,j) connects level k with level $k+1$, then its contribution to the objective function is $(n-k)c_{ij}$.

These facts entail to define the following decision variables for the first formulation

$$x_i^{(k)} = \begin{cases} 1, & \text{if node } i \text{ is in the position } k \text{ in a permutation} \\ 0, & \text{otherwise} \end{cases}$$

In order to evaluate the total latency for any feasible solution the following variables are introduced

$$y_{ij}^{(k)} = \begin{cases} 1, & \text{if node } i \text{ is in the position } k \text{ and node } j \text{ in position } k+1 \text{ in a permutation} \\ 0 & \text{otherwise} \end{cases}$$

Note that it is equivalent to define $y_{ij}^{(k)} = 1$, if arc (i,j) is used to link node i in position k with node j in position $k+1$.

Variables $x_i^{(1)}$, $(i = 1,2,\ldots,n)$, are used to calculate the contribution to total latency of any arc between positions 0 and 1, while variables $y_{ij}^{(k)}$ $(i = 1,2,\ldots,n;\ j = 1,2,\ldots,n;\ j \neq i;\ k = 1,2,\ldots,n-1)$ are used for arcs linking other consecutive positions.

Model A:

$$\min z = n\sum_{i=1}^{n} c_{0i}x_i^{(1)} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}(n-k)c_{ij}y_{ij}^{(k)} \tag{1}$$

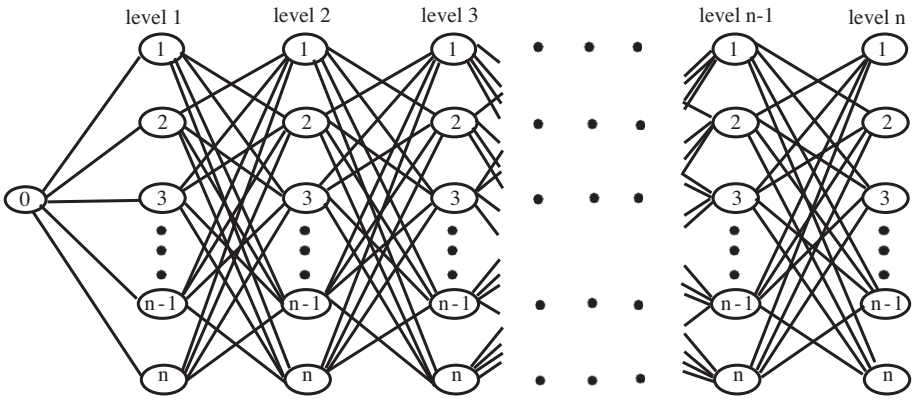

Fig. 1. Graphical representation of the latency problem by a multilevel network.

Subject to:

$$\sum_{k=1}^{n} x_i^{(k)} = 1 \quad (i = 1, 2, \ldots, n), \tag{2}$$

$$\sum_{i=1}^{n} x_i^{(k)} = 1 \quad (k = 1, 2, \ldots, n), \tag{3}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} y_{ij}^{(k)} = x_i^{(k)} \quad (i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, n-1), \tag{4}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} y_{ji}^{(k)} = x_i^{(k+1)} \quad (i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, n-1), \tag{5}$$

$$x_i^{(k)} \in \{0, 1\} \quad (i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, n), \tag{6}$$

$$y_{ij}^{(k)} \geqslant 0 \quad (i = 1, \ldots, n;\ j = 1, \ldots, n;\ j \neq i;\ k = 1, 2, \ldots, n-1). \tag{7}$$

Constraints (2) guarantee that each node occupies a single position in any feasible solution, while constraints (3) guarantee that each position is occupied by a single node. Constraints (4) ensure that only one arc leaves from position k, indeed it does it from the node taking that position. Constraints (5) impose that at position $k + 1$ only one arc can arrive at a time, in fact it is the node that occupies that position. Finally, constraints (6) and (7) establish that $x_i^{(k)}$ are binary and $y_{ij}^{(k)}$ are non-negative real variables, respectively. The previous formulation consists of n^2 binary variables, $n^3 - 2n^2 + 2n$ real variables and $2n^2$ constraints.

This formulation differs from formulation NO2 given by Gouveia and Voss [22] since they do not consider constraints (3) and force variables $y_{ij}^{(k)}$ to be binary variables.

It should be noted that including constraints (3) allow the elimination of twice the number of variables and constraints that would be eliminated if they were not included when one variable $x_i^{(k)}$ is set equal to 1 in the process of branching. This directly impacts on the time to achieve the optimum solution, as will be shown in the section devoted to computational experiments.

On the other hand, note that constraints in Model A force variables $y_{ij}^{(k)}$ to take integer values even if they are defined as non-negative real variables in constraints (7).

In fact, let suppose that $x_r^{(k)} = 1$ and $x_t^{(k+1)} = 1$. Taking into account the assignment constraints (2) and (3), it follows from (4) that $\sum_{l=1, l\neq r}^{n} y_{rl}^{(k)} = 1$ and $\sum_{l=1, l\neq i}^{n} y_{il}^{(k)} = 0$, ($i = 1, 2, \ldots, n$; $i \neq r$). This means that only variables $y_{rj}^{(k)}$, ($j = 1, 2, \ldots, n$; $j \neq r$) may take positive values. Moreover, it follows from (5) that $\sum_{l=1, l\neq t}^{n} y_{lt}^{(k)} = 1$ and $\sum_{l=1, l\neq j}^{n} y_{lj}^{(k)} = 0$ $(j = 1, 2, \ldots, n;\ j \neq t)$, which means that only variables $y_{lt}^{(k)}$ $(l = 1, 2, \ldots, n;\ l \neq t)$ may take non zero values.

Combining these two results we see that arcs from level k to level $k + 1$ must leave from node r. Furthermore, the only node that may receive arcs in level $k + 1$ is the node t.

Therefore, from $\sum_{l=1, l\neq r}^{n} y_{rl}^{(k)} = 1$ we have $y_{rt}^{(k)} = 1$ while the other variables involved in the summation are zero. Similarly, arcs leaving level k must arrive to node t in level $k + 1$ but only from node r in level k may leave arcs. Consequently, from $\sum_{l=1, l\neq t}^{n} y_{lt}^{(k)} = 1$ we have $y_{rt}^{(k)} = 1$ while the other variables in the summation are zero.

In short, if $x_r^{(k)} = 1$ and $x_t^{(k+1)} = 1$, constraints of Model A guarantee that $y_{rt}^{(k)} = 1$ and $y_{ij}^{(k)} = 0$ for $i, j = 1, 2, \ldots, n$; $j \neq i$, $i \neq r$, $j \neq t$, for this k. Since this is true for any k, we have that $y_{ij}^{(k)} \in \{0, 1\}$, $k = 1, 2, \ldots, n-1$; $i, j = 1, 2, \ldots, n$; $j \neq i$.

Our second formulation (Model B below) has been derived from the previous one by eliminating variables $x_i^{(k)}$ and defining variables $y_{ij}^{(k)}$ as binary variables. Next we explain how this was accomplished.

Rewriting constraints (5) as

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} y_{ji}^{(k-1)} = x_i^{(k)} \quad (i = 1, 2, \ldots, n;\ k = 2, 3, \ldots, n) \tag{5'}$$

it can be noted that the left sides of constraints (4) and (5′) are equal to the same value of $x_i^{(k)}$ for k from 2 to $n - 1$. Therefore, we obtain

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} (y_{ij}^{(k)} - y_{ji}^{(k-1)}) = 0 \quad (i = 1, 2, \ldots, n;\ k = 2, 3, \ldots, n-1). \tag{8}$$

These constraints ensure that the number of arcs going in and out on each node is the same from level 2 to level $n - 1$.

On the other hand, for $k = 1$ and $k = n$, the following expressions are obtained from (4) and (5′) respectively:

$$\sum_{j=1,j\neq i}^{n} y_{ij}^{(1)} = x_i^{(1)}; \quad \sum_{j=1,j\neq i}^{n} y_{ji}^{(n-1)} = x_i^{(n)}. \tag{$*$}$$

Now, taking into account that only one arc can leave from the first level of the network and only one arc can arrive to the last level and adding for i from 1 to n in $(*)$, the following two constraints are obtained:

$$\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} y_{ij}^{(1)} = 1; \tag{9}$$

$$\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} y_{ji}^{(n-1)} = 1; \tag{10}$$

Constraint (9) guarantees that a single arc leaves the first level of the network while constraint (10) ensures that a single arc arrives to level n.

Now, adding constraints (5) for k from 1 to $n-1$ and using (2), we obtain

$$\sum_{k=1}^{n-1}\sum_{j=1,j\neq i}^{n} y_{ji}^{(k)} = 1 - x_i^{(1)}$$

And substituting $x_i^{(1)}$ from $(*)$, we get

$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ij}^{(1)} + \sum_{k=1}^{n-1}\sum_{j=1,j\neq i}^{n} y_{ji}^{(k)} = 1 \quad (i = 1,2,\ldots,n) \tag{11}$$

In a similar way we can start from (4) to get

$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ji}^{(n-1)} + \sum_{k=1}^{n-1}\sum_{j=1,j\neq i}^{n} y_{ij}^{(k)} = 1, \quad (i = 1,2,\ldots,n) \tag{12}$$

Constraints (11) and (12) guarantee that only one node should be selected on each level and it should be different from nodes selected on other levels.

Thus, a valid formulation for the minimum latency problem can be established using constraints (8)–(12) when variables $y_{ij}^{(k)}$ are restricted to be binary variables.

$$y_{ij}^{(k)} \in \{0,1\} \quad (i = 1,2,\ldots,n;\ j = 1,2,\ldots,n;\ j \neq i;\ k = 1,2,\ldots,n-1). \tag{13}$$

For the objective function it is enough to replace $x_i^{(1)}$ in (1) using $(*)$:

$$z = n\sum_{i=1}^{n} c_{0i}\sum_{j=1,j\neq i}^{n} y_{ij}^{(1)} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} (n-k)c_{ij}y_{ij}^{(k)} \tag{14}$$

Note that constraints (8)–(10) ensure the continuity of the shortest path from 0 to n over the network in Fig. 1. Therefore, only one set of constrains (11) or (12) is sufficient to ensure that the selected node at every level is different from the selected nodes at other levels. For this reason, the set of constraints (12) will not be included in the second formulation.

In short, the second formulation is as follows.

Model B:

$$\min z = n\sum_{i=1}^{n} c_{0i}\sum_{j=1,j\neq i}^{n} y_{ij}^{(1)} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} (n-k)c_{ij}y_{ij}^{(k)} \tag{15}$$

Subject to:

$$\sum_{\substack{j=1\\ j\neq i}}^{n} (y_{ij}^{(k)} - y_{ji}^{(k-1)}) = 0 \quad (i = 1,2,\ldots,n;\ k = 2,3,\ldots,n-1), \tag{16}$$

$$\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ij}^{(1)} = 1, \tag{17}$$

$$\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ji}^{(n-1)} = 1, \tag{18}$$

$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ij}^{(1)} + \sum_{k=1}^{n-1}\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ji}^{(k)} = 1 \quad (i = 1,2,\ldots,n), \tag{19}$$

$$y_{ij}^{(k)} \in \{0,1\} \quad (i = 1,2,\ldots,n;\ j = 1,2,\ldots,n;\ j \neq i;\ k = 1,2,\ldots,n-1). \tag{20}$$

Differences between this model and the one proposed by Picard and Queyranne [3] (in what follows Model PQ) are: First, we are not considering returning arcs in the objective function. Secondly, we do not define variables for the incoming and outgoing arcs of the root node. Doing so, Model B has $2n$ binary variables and $2n$ constraints less than Model PQ. We include an additional constraint, namely (10), in order to ensure that a single arc arrives to the last level. With this transformation, model B can be regarded as a preprocessing of PQ formulation, where $2n$ binary variables and $(2n-1)$ restrictions were eliminated.

Finally, with the purpose of strengthening the proposed models, we define variables u_i, corresponding to the position of node i in the permutation and include the following valid inequalities (15)–(17)

$$u_i - u_j + n\sum_{k=1}^{n-1} y_{ij}^{(k)} + (n-2)\sum_{k=1}^{n-1} y_{ji}^{(k)} \leqslant n-1, \quad (i,j = 1,2,\ldots,n;\ j \neq i) \tag{21}$$

$$\sum_{i=1}^{n} u_i = n(n-1)/2 \tag{22}$$

$$u_i = \sum_{k=1}^{n} k x_i^{(k)}, \quad (i = 1,2,\ldots,n) \tag{23}$$

The set of constraints (15) were inspired by ideas presented by Desrochers and Laporte [23] and were added to both models. Note that when arc (i,j) is used on some level, constraints (15) guarantee that $u_j = u_i + 1$.

Constraint (16) is redundant for the integer models A and B proposed here, but it helps to improve the objective value of linear relaxation.

Finally, constraints (17) were added to Model A to relate variables u_i with the assignment variables $x_i^{(k)}$. Resulting models will be referred as Model A (Modif.) and Model B (Modif.), respectively.

Doing so, linear relaxations were improved and we expected that optimal solutions were reached quicker but, as it can be seen from the computational experiments, this is not true in all cases.

4. Computational experiments

4.1. Instances

Two classes of instances were generated for conducting the experiments: geometrical instances (denoted as GTRP) and triangularized asymmetrical instances (denoted as TrATRP).

Instances for the first class (GTRP) were obtained by randomly generating points with real coordinates from a uniform distribution between 0 and 100 and by taking the Euclidean distances (rounded down to integers) as travel times t_{ij}. Then, for generating service times s_i three cases were considered:

Instances GTRP-S0: $s_i = 0$.
Instances GTRP-S1: s_i were randomly chosen from the interval $[(0,(t_{\max} - t_{\min})/2]$.
Instances GTRP-S2: s_i were randomly chosen from the interval $[(t_{\max} + t_{\min})/2, (3t_{\max} - t_{\min})/2]$ where $t_{\max} = \max\{t_{ij}\}$ and $t_{\min} = \min\{t_{ij}\}$.

The costs c_{ij} associated to arcs (i,j) were calculated as follows:

$$c_{ij} = \begin{cases} t_{0j}, & \text{for } i = 0;\ j = 1,2,\ldots,n; \\ s_i + t_{ij}, & \text{for } i = 1,2,\ldots,n;\ j = 0,1,\ldots,n;\ j \neq i. \end{cases}$$

Symmetrical instances GTRP-S0 are commonly used in scientific literature, GTRP-S1 are related to the deliveryman problem and GTRP-S2 to the repairman problem. Several values of n (number of nodes) were tested. There are 25 instances for each value of n in each group.

For the second class of instances (TrATRP), values c_{ij} were randomly chosen from the interval [1,100] and then, they were forced to satisfy the triangle inequality through the shortest path.

All experiments were conducted on a PC with a 3.00 GHz Intel processor and 3.21 GB of RAM under Windows XP. All formulations were implemented using the concert technology of Cplex 11.1.

4.2. Comparison with previous models

As we mentioned before, formulations with better performance that have been published are due to Méndez-Díaz et al. [19], Gouveia and Voss [22] and Picard and Queyranne [3]. They will be denoted in the following tables by MZL, NO2 and PQ, respectively.

Table 1 shows the CPU time needed by Cplex 11.1 for solving models MZL, NO2 and PQ with geometrical instances for TRP. Values on each entry are averaged over 25 instances. The number of nodes is indicated in the first column.

For each instance in the group the solver was allowed to run for 2 hours. The symbol "$*$" means that the solver was unable to reach the optimal solution for some of the instances within this time.

Regarding CPU time for reaching the optimal solution, we can observe that formulations NO2 and PQ have a similar behavior. However, formulation MZL could not get the optimal solution after 2 hours for some instances with 20 nodes. Nevertheless, we should remark that formulation MZL was originally developed to be coupled with a cutting plane algorithm; in that case it really shows its potential.

Considering that in this work we focus on comparing the different formulations in relation to the needed CPU time for reaching the optimal solution, in what follows we refer only to formulations NO2 and PQ.

Table 2 shows a comparison between NO2 and the Model A proposed in this work.

It can be observed that Model A needs less CPU time than Model NO2 for reaching optimal solution as the size of the instances increases. This confirms the convenience of the inclusion of constraints (3) in Model A.

In Table 3 we show a comparison between Model B, proposed in this work, and PQ model

These models have similar behaviors. Even though Model B required less CPU time for instances with 25 nodes, none of them could get optimal solutions for instances with 30 nodes due to lack of memory.

From above tables we can conclude that Model A is the one that reaches the optimal solution using less computing time. This behavior is more appreciable for large instances. Note that, for instances with 30 nodes, Model A consumes about 234.6 seconds less than NO2, which means that NO2 needs 18.48% of additional computing time compared to Model A. For instances with 25 nodes Model B consumes, in average, 98.4 seconds more than Model A (45.6% of additional computing time), while PQ consumes 112.2 seconds more than Model A, which represents 52.11% of additional CPU time.

In Table 4 we compare the four models for triangularized asymmetrical instances (TrATRP).

Note that for this kind of instances all formulations are capable of obtaining optimal solutions up to 40-nodes instances.

Table 1
CPU time in seconds for Mendez-Diaz, Gouveia and PQ formulations for GTRP instances.

n	GTRP-S0			GTRP-S1			GTRP-S2		
	MZL	NO2	PQ	MZL	NO2	PQ	MZL	NO2	PQ
10	1.14	0.14	0.12	1.11	0.15	0.12	1.46	0.15	0.13
15	216.06	2.43	2.62	263.71	1.85	2.12	200.09	1.64	2.28
20	4907.8*	25.50	26.78	6066.4*	35.52	39.18	5859.5*	23.20	27.71

Table 2
CPU time in seconds for Model A and NO2 for GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2		Total Average	
	A	NO2	A	NO2	A	NO2	A	NO2
10	0.23	0.14	0.23	0.15	0.24	0.15	0.23	0.15
15	2.16	2.43	2.06	1.85	1.513	1.64	1.91	1.97
20	25.32	25.50	32.62	35.51	21.18	23.19	26.37	28.07
25	317.05	356.88	195.38	244.07	134.76	172.53	215.72	257.83
30	1548.94	1812.71	1358.83	1582.80	901.57	1117.32	1269.73	1504.31

Table 3
CPU times in seconds for Model B and PQ for GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2		Total Average	
	B	PQ	B	PQ	B	PQ	B	PQ
10	0.12	0.12	0.13	0.12	0.13	0.13	0.13	0.12
15	2.48	2.62	2.66	2.12	1.90	2.28	2.35	2.34
20	28.00	26.78	42.59	39.18	27.95	27.70	32.85	31.22
25	441.19	457.29	294.43	298.98	206.88	228.19	314.16	328.15

Table 4
CPU times in seconds for NO2, PQ, Model A and Model B for TrATRP instances.

TrATRP				
n	NO2	PQ	A	B
10	0.08	0.03	0.07	0.03
15	0.23	0.38	0.32	0.39
20	0.78	0.75	0.90	0.61
25	3.96	3.46	4.17	3.19
30	10.29	9.31	13.21	9.36
35	36.26	32.88	45.65	34.70
40	122.40	100.15	186.10	134.80

Table 5
Impact of the valid inequalities on the gap values in the linear relaxation of Model A for GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2	
	A	A (Modif.)	A	A (Modif.)	A	A (Modif.)
10	9.027	3.365	5.938	2.480	2.638	1.061
15	16.260	8.718	8.145	4.137	3.470	1.760
20	19.308	10.998	10.774	6.336	3.907	2.319
25	23.748	14.300	9.968	5.724	4.041	2.486
30	20.553	12.348	9.914	6.308	3.782	2.466

Table 6
Impact of the valid inequalities on the CPU times in the linear relaxation of Model A for GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2	
	A	A (Modif.)	A	A (Modif.)	A	A (Modif.)
10	0.018	0.023	0.021	0.024	0.020	0.025
15	0.056	0.081	0.057	0.085	0.065	0.095
20	0.165	0.269	0.191	0.270	0.199	0.293
25	0.454	0.764	0.505	0.813	0.518	0.845
30	1.053	1.693	1.278	1.784	1.369	1.927

Table 7
Impact of the valid inequalities on the gap values in the linear relaxation of Model B for GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2	
	B	B (Modif.)	B	B (Modif.)	B	B (Modif.)
10	9.027	4.546	5.938	3.364	2.638	1.571
15	16.261	9.919	8.145	5.206	3.470	2.292
20	19.309	12.317	10.775	7.500	3.907	2.741
25	23.749	15.675	9.968	6.709	4.041	2.882
30	20.553	13.728	9.914	7.282	3.782	2.790

Table 8
Impact of the valid inequalities on the CPU times in the linear relaxation of Model B for GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2	
	B	B (Modif.)	B	B (Modif.)	B	B (Modif)
10	0.012	0.016	0.014	0.018	1.571	0.023
15	0.038	0.059	0.046	0.075	2.292	0.095
20	0.140	0.203	0.188	0.279	2.741	0.359
25	0.424	0.637	0.602	0.928	2.882	1.134
30	1.122	1.528	1.514	2.558	2.790	2.973

Table 9
Impact of the valid inequalities on the CPU time expended for obtaining optimal solutions using Model A on GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2	
	A	A (Modif.)	A	A (Modif.)	A	A (Modif.)
10	0.23	0.18	0.23	0.19	0.24	0.19
15	2.16	2.70	2.06	2.21	1.51	2.02
20	25.32	33.39	32.62	48.17	21.18	26.75
25	317.05	395.01	195.38	286.66	134.76	198.69

Table 10
Impact of the valid inequalities on the CPU time expended for obtaining optimal solutions using Model B on GTRP instances.

n	GTRP-S0		GTRP-S1		GTRP-S2	
	B	B(Modif.)	B	B(Modif.)	B	B(Modif.)
10	0.12	0.14	0.13	0.16	0.13	0.18
15	2.48	3.04	2.66	2.73	1.90	2.04
20	28.00	50.94	42.59	95.72	27.95	51.77
25	441.19	547.02	294.43	408.13	206.88	303.67

In Tables 5–8 we evaluate the impact of including the valid inequalities in Model A and B. Tables 5 and 7 display the relative gap (%) between the value of the linear relaxation and optimal integer value while Tables 6 and 8 exhibit the time (in seconds) expended for solving the linear relaxation.

From these tables we can observe that including the valid inequalities in Model A and B improves the linear relaxation in all cases, while the CPU time has only a small increase.

In Tables 9 and 10 we evaluate how the valid inequalities impact on the CPU time needed for obtaining the optimal integer solution.

Despite the fact that the linear relaxation has been improved with the inclusion of the valid inequalities, it does not mean that necessarily the optimal integer solution is reached in a quicker way. Note that the CPU time was increased in all cases.

5. Conclusions

From our computational experiments we may conclude that Model A performs better than the other formulations. The inclusion of constraints (3) in this model helped to reduce the computational time to reach the optimal solution while the valid inequalities, although they improved the value of the linear relaxation of the proposed models, they did not reduce the computational time.

Moreover, it is observed that, for the studied models, instances generated from coordinates are harder to solve than those generated at random and then forced to meet the triangle inequality. Despite this, we recommend to use instances generated from coordinates when performing computational experiments related to the addressed problems. These instances capture in a better way the characteristics of these problems.

Acknowledgements

This work was partially sponsored by the Mexican National Council of Science and Technology (Grant 61903) and by the Research Chair in Industrial Engineering of Tecnológico de Monterrey (ITESM Research Fund CAT128). These grants are gratefully acknowledged.

References

[1] R. Conway, W. Maxwell, L. Miller, Theory of Scheduling, Addison-Wesley, 1967.
[2] A. Rinnooy Kan, Machine Scheduling Problems, Martinus Nijhoff, The Hague, 1976.
[3] J. Picard, M. Queyranne, The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling, Oper. Res. 26 (1) (1978) 86–110.
[4] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, in: Proceedings of the 26th ACM Symposium on Theory of Computing (STOC), 1994, pp. 163–171.
[5] D. Simchi-Levi, O. Berman, Minimizing the total flow time of n jobs on a network, IIE Trans. 23 (3) (1991) 236–244.
[6] G. Ausiello, S. Leonardi, A. Marchetti-Spaccamela, On salesmen, repairmen, spiders and other traveling agents, in: Proceedings of the Italian Conference on Algorithms and Complexity, 2000, pp. 1–16.
[7] S. Sahni, T. Gonzalez, P-complete approximation problems, J. Assoc. Comput. Mach. 23 (1976) 555–565.
[8] R. Sitters, The minimum latency problem is NP-hard for weighted trees, in: Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization, 2002, pp. 230–239.
[9] M. Goemans, J. Kleinberg, An improved approximation ratio for the minimum latency problem, Math. Program. 82 (1998) 111–124.

[10] A. Archer, D. P. Williamson, Faster approximation algorithms for the minimum latency problem, in: Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 88–96.
[11] A. Archer, A. Levin, D.P. Williamson, A faster, better approximation algorithm for the minimum latency problem, SIAM J. Comput. 37 (2008) 1472–1498.
[12] S. Arora, G. Karakostas, Approximation schemes for minimum latency problem, in: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, 1993, pp. 688–693.
[13] B. Chaudhuri, S. Godfrey, S. Rao, K. Talwar, Paths, trees, and minimum latency tours, in: Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS), 2003.
[14] V. Nagarajan, R. Ravi, The directed minimum latency problem, in: A. Goel et al. (Eds.), APPROX and RANDOM 2008, LNCS, vol. 5171, Springer-Verlag, 2008, pp. 193–206.
[15] Z. Friggstad, M. Salavatipour, Z. Svitkina, Asymmetric traveling salesman path and directed latency problems. In Proceedings of the 21st ACM–SIAM Symposium on Discrete Algorithms, 2010, pp. 419–428.
[16] A. Lucena, Time-dependent traveling salesman problem–the deliveryman case, Networks 20 (1990) 753–763.
[17] M. Fischetti, G. Laporte, S. Martello, The delivery man problem and cumulative matroids, Oper. Res. 41 (6) (1993) 1055–1064.
[18] BY. Wu, Z.-N. Huang, F.-J. Zhan, Exact algorithms for the minimum latency problem, Inform. Process. Lett. 92 (6) (2004) 303–309.
[19] I. Méndez-Díaz, P. Zabala, A. Lucena, A new formulation for the traveling deliveryman problem, Discrete Appl. Math. 156 (2008) 3223–3237.
[20] C. Eijl, A polyhedral approach to the delivery man problem, Memorandum COSOR 95-19, Eindhoven University of Technology, 1995.
[21] J. Sarubbi, H. Luna, G. Miranda, Minimum latency problem as a shortest path problem with side constraints, in: Book of Extended Abstracts of the XIV Latin Ibero-American Congress on Operations Research (CLAIO 2008), 2008.
[22] L. Gouveia, A. Voss, A classification of formulations for the (time-dependent) travelling salesman problem, Eur. J. Oper. Res. 83 (1995) 69–82.
[23] M. Desrochers, G. Laporte, Improvements and extensions to the Miller–Tucker–Zemlin sub-tour elimination constraints, Oper. Res. Lett. 10 (1991) 27–36.