FORMULATIONS FOR THE MINIMUM LATENCY PROBLEM: AN EXPERIMENTAL EVALUATION

Francisco Angel-Bello Acosta[1]

Department of Industrial and Systems Engineering
Tec de Monterrey, campus Monterrey, Mexico
E-mail: *fangel@itesm.mx*

Ada Alvarez Socarrás

Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León, Mexico
E-mail: *adalvarez@mail.uanl.mx*

Irma García

Centre for Research on Applied Mathematics
Universidad Autónoma de Coahuila, Mexico
E-mail: *irma.igarcia@gmail.com*

June 2011

[1] Corresponding author

Abstract

In network problems, the latency is associated with the metric 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 travelling salesman problem, highlighting the similarities and differences between them and evaluating the convenience of including a set of constraints that help to reduce the computation time to reach the optimal solution. We have carried out extensive computational experimentation on asymmetrical instances, since they provide better the characteristics of the deliveryman and repairman problems.

Keywords: minimum latency, repairman problem

1. Introduction

In network problems, the latency is associated with the metric 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 lengths of the paths to all vertices.

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

In the deliveryman and repairman problem, 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 total waiting time of clients. 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 in a single machine.

In single machine scheduling context, 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 total flow time or the total completion time of jobs. In fact, the minimum latency problem arises from this kind of problem (Conway et al. 1967, Rinnooy Kan 1976, Picard and Queyranne 1978).

Besides these straight applications of the MLP, other applications can be found in disk-head scheduling (Blum et al. 1994), in the routing of automated guided vehicles in a flexible manufacturing system (Simchi-Levi and Berman 1991), and in the process of information search in computer networks (Ausiello et al. 2000, Koutsoupias 1996).

It has been established that MLP is NP-hard for general metrics (Sahni and Gonzalez 1976) and when the metric space is induced by a tree (Sitters 2002). In spite of its similarities with the Travelling Salesman Problem (TSP), the MLP is much harder to solve or approximate from a computational point of view than the TSP (Goemans and Kleinberg 1998).

Research has been conducted to develop approximations for the MLP (Blum et al. 1994, Arora and Karakostas 1993, Goemans and Kleinberg 1998, Archer and Williamson 2003, Archer et al. 2004), and the current best approximation ratio for undirected graph is 3.59 (Chaudhuri et al. 2003).

Some authors have proposed exact optimization algorithms to solve variants of the MLP. Lucena (1990) developed an enumerative procedure based on a nonlinear integer programming formulation, using a Lagrangean relaxation to obtain lower bounds and solved instances up to 30 nodes. Fischetti et al. (1993) 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. (2000) obtained exact solutions up to 23 nodes by using dynamic programming with pruning.

Méndez-Díaz et al. (2008) 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 their counterparts in Fischetti et al. (1993) and Eijl (1995).

Sarubbi et al. (2008) implemented, for the minimum latency problem, the model developed by Picard and Queyranne (1978) 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 each 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 and then triangularizing 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 the 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 and they are only used 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. (2008), Gouveia and Voss (1995) and the implementation done by Sarubbi et al. (2008) of Picard and Queyranne formulation (1978). 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 the work with computational experimentation on asymmetrical instances since they provide better the characteristics of the deliveryman and repairman problems. 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 $V=\{0,1,2,\cdots,n\}$ 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 $l_{[i]}=l_{[i-1]}+c_{[i-1][i]}$, 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 $L=\sum_{i=1}^{n} l_{[i]}$. The objective of the minimum latency problem is to find a permutation of nodes that minimizes the total latency.

Indeed, using the formula $l_{[i]}=l_{[i-1]}+c_{[i-1][i]}$ is possible to obtain the latency for each position *i* as follows

$$l_{[1]}=l_{[0]}+c_{[0][1]}=c_{[0][1]}$$

$$l_{[2]}=l_{[1]}+c_{[1][2]}=c_{[0][1]}+c_{[1][2]}$$

$$l_{[3]}=l_{[2]}+c_{[2][3]}=c_{[0][1]}+c_{[1][2]}+c_{[2][3]}$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$l_{[n-1]}=l_{[n-2]}+c_{[n-2][n-1]}=c_{[0][1]}+c_{[1][2]}+c_{[2][3]}+\cdots+c_{[n-2][n-1]}$$

$$l_{[n]} = l_{[n-1]} + c_{[n-1][n]} = c_{[0][1]} + c_{[1][2]} + c_{[2][3]} + \cdots + c_{[n-1][n]}$$

Adding all the latencies we obtain the total latency:

$$L = \sum_{i=1}^{n} l_{[i]} = nc_{[0][1]} + (n-1)c_{[1][2]} + (n-2)c_{[2][3]} + \cdots + 2c_{[n-2][n-1]} + c_{[n-1][n]} = \sum_{i=1}^{n} (n-i+1)c_{[i-1][i]}$$

From this last expression for the total latency, we can see that the minimum latency problem 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 that occupies in the tour.

In the deliveryman 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:

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

In the total completion time problem, as well as in the total flow time problem, on a single machine with sequence-dependent setup times, the values c_{ij} are calculated as

$$c_{ij} = \begin{cases} t_{ij} + s_j, \text{for } i = 0,1,\cdots,n;\ j = 1,2,\cdots,n;\ j \neq i \\ t_{i0}, \text{for } i = 1,2,\cdots,n;\ j = 0; \end{cases}$$

Note that, if different clients (jobs) have different service (processing) times, the weight matrix is asymmetrical.

3. Linear integer formulations for minimum latency problem with asymmetric costs.

The formulations proposed here are inspired in the analysis presented by Gouveia and Voss (1995) for several formulations for the Time-Dependent Traveling Salesman Problem.

In order to provide the mathematical models, we will use the multi-level network shown in Figure 1, where n copies of nodes $\{1,2,\cdots,n\}$ are depicted. As node 0 should occupy the first position in all permutations, this position will be referred as position 0 and corresponds to level 0 on the graphic. Other positions (levels in the graphic) will be referred as positions $1, 2, \cdots, n$.

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

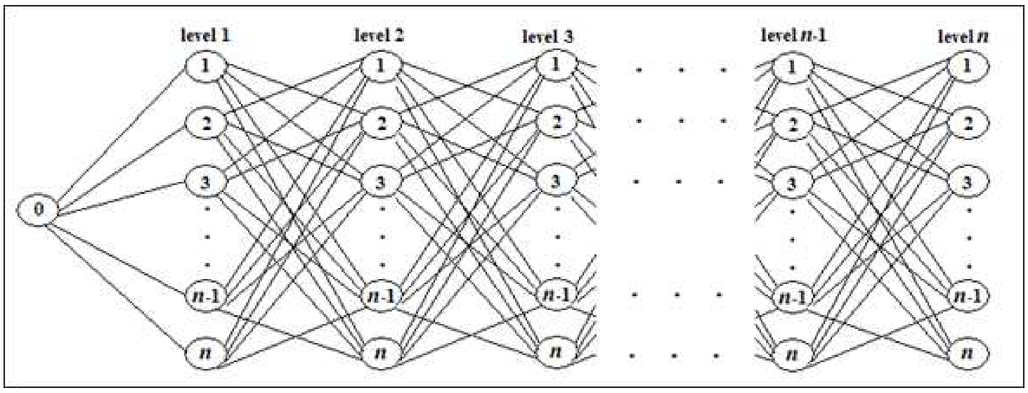

Figure 1. Graphical representation of a multilevel network

The contribution to the objective function of each used arc in a feasible solution depends on the levels that are 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 $x_i^{(1)}, (i = 1,2,\cdots,n)$ are used to calculate the contribution to total latency of any arc between positions 0 and 1. Variables $y_{ij}^{(k)}, (i = 1,2,\cdots,n; j = 1,2,\cdots,n; j \neq i; k = 1,2,\cdots,n-1)$ are used for arcs linking other consecutive positions.

Model A:

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

$$s.t.\begin{cases} \sum_{k=1}^{n} x_i^{(k)} = 1 \quad (i = 1,2,\cdots,n) & (2)\\ \sum_{i=1}^{n} x_i^{(k)} = 1 \quad (k = 1,2,\cdots,n) & (3)\\ \sum_{\substack{j=1\\ j\neq i}}^{n} y_{ij}^{(k)} = x_i^{(k)} \quad (i = 1,2,\cdots,n;\ k = 1,2,\cdots,n-1) & (4)\\ \sum_{\substack{j=1\\ j\neq i}}^{n} y_{ji}^{(k)} = x_i^{(k+1)} \quad (i = 1,2,\cdots,n;\ k = 1,2,\cdots,n-1) & (5)\\ x_i^{(k)} \in \{0,1\} \quad (i = 1,2,\cdots,n;\ k = 1,2,\cdots,n) & (6)\\ y_{ij}^{(k)} \geq 0 \quad (i = 1,\cdots,n;\ j = 1,\cdots,n;\ j \neq i; k = 1,2,\cdots,n-1) & (7) \end{cases}$$

The constraints (2) guarantee that each node occupies a single position in any feasible solution, while constraints (3) guarantee that each position is occupied for a single node. Constraints (4) ensure that from position *k* can leave just one arc and it is indeed from the node that occupies that position. Constraints (5) impose that at position *k*+1 can arrive just one arc and it is indeed to 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 (1995) in that they do not consider constraints (3) and force variables $y_{ij}^{(k)}$ to be binary variables.

Note that constraints in Model A force variables $y_{ij}^{(k)}$ to take integer values even if they are considered 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), then, from (4) follows that $\sum_{\substack{l=1 \\ l \neq r}}^{n} y_{rl}^{(k)} = 1$ and $\sum_{\substack{l=1 \\ l \neq i}} y_{il}^{(k)} = 0, (i = 1,2,\cdots,n; i \neq r)$. This means that only variables $y_{rj}^{(k)}, (j = 1,2,\cdots,n; j \neq r)$ may take positive values. On the other hand, from (5) follows that $\sum_{\substack{l=1 \\ l \neq t}}^{n} y_{lt}^{(k)} = 1$ and $\sum_{\substack{l=1 \\ l \neq j}} y_{lj}^{(k)} = 0, (j = 1,2,\cdots,n; j \neq t)$, which means that only variables $y_{lt}^{(k)}, (l = 1,2,\cdots,n; l \neq t)$ may be different from zero.

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

Therefore, from $\sum_{\substack{l=1 \\ l \neq r}}^{n} y_{rl}^{(k)} = 1$ 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 $\sum_{\substack{l=1 \\ l \neq t}}^{n} y_{lt}^{(k)} = 1$ we have $y_{rt}^{(k)} = 1$ while the other variables involved 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 $i, j = 1,2,\cdots,n; j \neq i, i \neq r, j \neq t$, for this *k*. Since this is true for any *k*, we have that $y_{ij}^{(k)} \in \{0,1\}, k = 1,2,\cdots,n-1; i, j = 1,2,\cdots,n; j \neq i$.

On the other hand, including constraints (3) allow eliminating the double of the variables and constraints that would be eliminated if they were not included when one variable is set equal to 1 in the process of branching. This has an impact on the time to achieve the optimum solution as will be shown in the computational experiments

The second formulation is derived from the previous one by eliminating variables $x_i^{(k)}$ and considering variables $y_{ij}^{(k)}$ as binary variables. Next we explain how we have accomplished it.

Rewritting constraints (5) as

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

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

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

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:

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

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

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

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

Constraints (9) guarantee that a single arc leaves the first level of the network, while constraints (10) ensure that a single arc arrives to level *n*.

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

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

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

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

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

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

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

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

$$y_{ij}^{(k)} \in \{0,1\},\ (i = 1,2,\cdots,n;\ j = 1,2,\cdots,n;\ j \neq i; k = 1,2,\cdots,n-1) \qquad (13)$$

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

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

Note that constraints (8) - (10) are the constraints that find the shortest path from 0 to *n* over the network in Figure 1 and ensure the continuity of the path. Therefore, only one set of constraints (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):

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

$$s.t.\begin{cases}\sum_{\substack{j=1\\j\neq i}}^{n}(y_{ij}^{(k)}-y_{ji}^{(k-1)})=0, \quad (i=1,2,\cdots,n; k=2,3,\cdots,n-1) & (8)\\ \sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}y_{ij}^{(1)}=1; & (9)\\ \sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}y_{ji}^{(n-1)}=1; & (10)\\ \sum_{\substack{j=1\\j\neq i}}^{n}y_{ij}^{(1)}+\sum_{k=1}^{n-1}\sum_{\substack{j=1\\j\neq i}}^{n}y_{ji}^{(k)}=1, \quad (i=1,2,\cdots,n) & (11)\\ y_{ij}^{(k)}\in\{0,1\}, (i=1,2,\cdots,n; j=1,2,\cdots,n; j\neq i; k=1,2,\cdots,n-1) & (13)\end{cases}$$

Differences between this model and the proposed by Picard and Queyranne (in what follows Model PQ) are: we are not considered 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 2*n* binary variables and 2*n* constraints less than Model PQ. We include an additional constraint, namely (10), for ensuring that a single arc arrives to the last level. With this transformation, model B can be viewed 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, valid inequalities were considered. Namely, the following constraints were added to both models:

$$u_i-u_j+n\sum_{k=1}^{n-1}y_{ij}^{(k)}+(n-2)\sum_{k=1}^{n-1}y_{ji}^{(k)}\leq n-1, (i,j=1,2,\cdots,n; j\neq i), \qquad (15)$$

$$\sum_{i=1}^{n}u_i=\tfrac{n(n+1)}{2}, \qquad (16)$$

In addition, the following valid inequality was added to Model A:

$$u_i=\sum_{k=1}^{n}kx_i^{(k)}, (i=1,2,\cdots,n), \qquad (17)$$

Where the variables u_i correspond to the position of node *i* in the permutation

The resulting models will be referred as Model A (Modif.) and Model B (Modif.)

Doing so, linear relaxations were improved and we expected that optimal solutions were reached in a faster way, but as can be seen in 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 triangular asymmetrical instances (denoted as TrATRP).

Instances for the first class (GTRP) were obtained by uniformly randomly generating points with real coordinates in the 100 x 100 square 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:

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

Symmetrical instances GTRP-S0 are commonly used in the scientific literature, S1 are related to the deliveryman problem and GTRO-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.

4.1 Comparison with other models

As we mentioned before, formulations with better performance that have been published are due to Méndez-Díaz et al (2008), Gouveia et al. (1995) and Picard and Queyranne (1978), which have been denoted in the following tables by MZL, NO2 and PQ respectively. In the first column appears the number of nodes and values in each entry are averaged over 25 instances.

Table 1 shows the CPU time expended by Cplex 11.1 for solving models MZL, NO2 and PQ with geometrical instances for TRP. 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 in this time.

	GTRP-S0			GTRP-S1			GTRP-S2		
n	MZL	NO2	PQ	MZL	NO2	PQ	MZL	NO2	PQ
10	1.1432	0.1430	0.1256	1.1169	0.1556	0.1237	1.4607	0.1581	0.1336
15	216.06	2.4368	2.6292	263.71	1.8563	2.1292	200.09	1.6461	2.2888
20	4907.8*	25.504	26.783	6066.4*	35.518	39.181	5859.5*	23.199	27.706

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

Regarding CPU time for reaching the optimal solution, we can observe that formulations NO2 and PQ have a similar behaviour. 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 developed to be coupled with a cutting plane algorithm, where it really shows its potential.

As in this work we focus on comparing the different formulations in relation to the CPU time expended for reaching the optimal solution, in what follows we refer only to formulations NO2 y PQ.

In Table 2 are compared the Model A proposed in this work with NO2.

	GTRP-S0		GTRP-S1		GTRP-S2		Total Average	
N	Model A	NO2	Model A	NO2	Model A	NO2	Model A	NO2
10	0.2306	0.14308	0.23308	0.1556	0.2468	0.15812	0.23682	0.15226
15	2.16	2.43688	2.0688	1.85636	1.51364	1.64612	1.91414	1.97978
20	25.3245	25.5044	32.6279	35.5182	21.1852	23.1995	26.3792	28.0740
25	317.047	356.884	195.378	244.066	134.759	172.533	215.728	257.828
30	1548.86	1812.66	1358.77	1582.83	901.571	1117.32	1269.73	1504.27

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

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

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

GTRP	GTRP-S0		GTRP-S1		GTRP-S2		Total Average	
n	Model B	PQ	Model B	PQ	Model B	PQ	Model B	PQ
10	0.12052	0.12564	0.13624	0.12376	0.13572	0.13364	0.13082	0.12768
15	2.48388	2.62928	2.66132	2.1292	1.90692	2.2888	2.35070	2.34909
20	28.0022	26.7832	42.5917	39.1812	27.9579	27.7063	32.8506	31.2235
25	441.190	457.297	294.431	298.976	206.880	228.187	314.167	328.153

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

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

In Table 4 below we compared the four models for triangular asymmetrical instances (TrATRP).

TrATRP

n	NO2	PQ	Model A	Model B
10	0.08064	0.03628	0.07436	0.03304
15	0.23008	0.38508	0.32368	0.3998
20	0.785	0.75316	0.90824	0.61812
25	3.96252	3.46108	4.1726	3.1962
30	10.29124	9.31996	13.21684	9.36972
35	36.2642	32.8894	45.65376	34.70048
40	122.408	100.159	186.1078	134.8014

Table 4: CPU times in seconds for NO2, PQ, Model A y Model B for TrATRP instances

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

In tables 5 to10 we evaluate the impact of including the valid inequalities in Model A and B. Columns labeled “Gap” displayed the relative gap (%) between the value of the linear relaxation and optimal integer value. Columns labeled “CPUtime” exhibit the time expended for solving the linear problem relaxation.

GTRP-S0				
	Model A		Model A (Modif.)	
n	Gap	CPUtime	Gap	CPUtime
10	9.02734	0.01808	3.36523	0.02316
15	16.2607	0.0568	8.71885	0.0818
20	19.3089	0.16568	10.9981	0.26996
25	23.7487	0.45436	14.3002	0.76432
30	20.5532	1.05316	12.3487	1.69376

Table 5: Impact of the valid inequalities in the linear relaxation of Model A for GTRP-S0 instances

GTRP-S1				
	Model A		Model A (Modif.)	
n	Gap	CPUtime	Gap	CPUtime
10	5.93819	0.02128	2.48083	0.02496
15	8.14567	0.0574	4.13784	0.0856
20	10.7748	0.19188	6.3361	0.27056
25	9.968	0.5052	5.7246	0.8132
30	9.91427	1.27828	6.30896	1.78496

Table 6: Impact of the valid inequalities in the linear relaxation of Model for GTRP-S1 instances

GTRP-S2				
	Model A		Model A (Modif.)	
n	Gap	CPUtime	Gap	CPUtime
10	2.638043	0.02004	1.06114	0.02516
15	3.470068	0.06512	1.76076	0.09496
20	3.907974	0.19996	2.3191	0.29384
25	4.04143	0.51888	2.48656	0.84504
30	3.78227	1.36944	2.46675	1.92744

Table 7: Impact of the valid inequalities in the linear relaxation of Model A for GTRP-S2 instances

GTRP-S0				
	Model B		Model B (Modif.)	
n	Gap	CPU time	Gap	CPU time
10	9.02731	0.01248	4.54695	0.01628
15	16.2607	0.03816	9.91958	0.05948
20	19.3089	0.14056	12.3169	0.20308
25	23.7487	0.42492	15.6752	0.6374
30	20.5532	1.12248	13.7276	1.52872

Table 8: Impact of the valid inequalities in the linear relaxation of Model B for GTRP-S0 instances

GTRP-S1				
	Model B		Model B (Modif.)	
n	Gap	CPU time	Gap	CPU time
10	5.93819	0.01436	3.36414	0.0188
15	8.14567	0.04696	5.20623	0.0756
20	10.7748	0.1888	7.50072	0.2792
25	9.968	0.60252	6.70923	0.92816
30	9.91427	1.51452	7.2822	2.55816

Table 9: Impact of the valid inequalities in the linear relaxation of Model for GTRP-S1 instances

GTRP-S2				
	Model B		Model B (Modif.)	
N	Gap	CPUtime	Gap	CPUtime
10	2.63804	0.01492	1.57176	0.02304
15	3.4701	0.05812	2.29203	0.09568

20	3.90797	0.20684	2.74111	0.35932
25	4.04143	0.65376	2.88252	1.13436
30	3.78227	1.58988	2.79047	2.97316

Table10: Impact of the valid inequalities in the linear relaxation of Model for GTRP-S2 instances

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 little increase.

In tables 11 and 12 below we evaluate how the valid inequalities impact in the CPU time for obtaining the optimal integer solution.

GTRP	S0		S1		S2		Total Average	
n	A	A (Mod)	A	A (Mod)	A	A (Mod)	A	A (Mod)
10	0.2306	0.1887	0.2331	0.1963	0.2468	0.1995	0.2368	0.19485
15	2.16	2.7075	2.0688	2.2151	1.5136	2.0243	1.9142	2.31567
20	25.3246	33.394	32.628	48.174	21.185	26.752	26.379	36.1071
25	317.048	395.01	195.38	286.66	134.76	198.69	215.71	293.457

Table 11: Impact of the valid inequalities in the CPU time expended for obtaining optimal solutions using Model A on GTRP instances

GTRP	S0		S1		S2		Total Average	
n	B	B (Mod)	B	B (Mod)	B	B (Mod)	B	B (Mod)
10	0.12052	0.14884	0.1362	0.16388	0.1357	0.1826	0.1308	0.16511
15	2.48388	3.04568	2.6613	2.73112	1.9069	2.0408	2.3507	2.60587
20	28.0022	50.9491	42.592	95.7272	27.958	51.7712	32.851	66.1492
25	441.19	547.021	294.43	408.129	206.88	303.671	314.17	419.607

Table 12: Impact of the valid inequalities in the CPU time expended for obtaining optimal solutions using Model B on GTRP instances

In spite of the linear relaxation is improved with the inclusion of the valid inequalities, this does not mean that necessarily the optimal integer solution is reached in a faster 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 computation time to reach the optimal solution, while the valid inequalities, although improved the value of the linear relaxation of the proposed models, did not reduce the computational time.

On the other hand, 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 because they capture better the characteristics of the studied problems.

REFERENCES

A. Archer and D. Williamson. *Faster Approximation Algorithms for the Minimum Latency Problem*, Proceedings of the fourteenth annual ACM-SIAM Symposium on Discrete algorithms, Baltimore, 2003.

A. Archer, A. Leviny and D. Williamson. *A Faster, Better Approximation Algorithm for the Minimum Latency Problem*, citeseer.ist.psu.edu/676049.html, 2004.

S. Arora and G. Karakostas. *Approximation Schemes for Minimum Latency Problem*, Proceedings of the thirty-first annual ACM Symposium on Theory of computing, Atlanta, 688-693, 1993.

G. Ausiello, S. Leonardi, A. Marchetti-Spaccamela, *On salesmen, repairmen, spiders and other traveling agents*, in: Proc. of the Italian Conference on Algorithms and Complexity, 1-16, 2000.

A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan and M. Sudan, *The Minimum Latency Problem*, Proceedings of 26th ACM Symp. on Theory Of Computing (STOC), 163-171, 1994.

R. Conway, W. Maxwell, and L. Miller. *Theory of Scheduling*. Addison-Wesley, 1967.

B. Chaudhuri, S. Godfrey, S. Rao and K. Talwar. *Paths, Trees, and Minimum Latency Tours*, Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS), 2003.

M. Desrochers and G. Laporte. *Improvements and extensions to the Miller-Tucker-Zemlin sub-tour elimination constraints,* Operational Research Letters 10, 27-36, 1991.

C. Eijl. *A polyhedral approach to the delivery man problem*, Memorandum COSOR 95-19, Eindhoven University of Technology, 1995

M. Fischetti, G. Laporte, S. Martello, *The delivery man problem and cumulative matroids*, Operations Research 41 (6) 1055-1064, 1993.

M. Goemans and J. Kleinberg. *An improved approximation ratio for the minimum latency problem*, Mathematical Programming 82, 111-124, 1998
L. Gouveia , A.Voss. *A classification of formulations for the (time-dependent) travelling salesman problem*. European Journal of Operational Research 83, 69–82, 1995.

E. Koutsoupias, C.H. Papadimitriou, M. Yannakakis, *Searching a Fixed graph*, ICALP, 280-289, 1996.

A. Lucena. *Time-dependent traveling salesman problem- the deliveryman case*. Networks, vol 20, 753-763, 1990

I. Méndez-Díaz, P. Zabala , A. Lucena *A new formulation for the Traveling Deliveryman Problem* Discrete Applied Mathematics 156, 3223-3237, 2008.

J. Picard, M. Queyranne. *The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling*. Operations Research 26 (1) 86-110, 1978.

A. Rinnooy Kan. *Machine Scheduling Problems*. Martinus Nijhoff, The Hague , 1976.

J. Sarubbi , H. Luna, G. Miranda . *Minimum latency problem as a shortest path problem with side constraints*. In: XIV Latin Ibero-American Congress on Operations Research (CLAIO);2008

D. Simchi-Levi and O. Berman. *Minimizing the total flow time of n jobs on a network*. IIE Transactions, 23(3):236–244, 1991.

R. Sitters. *The minimum latency problem is NP-hard for weighted trees*, in: Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization, 230-239, 2002.

S. Sahni, T. Gonzalez. *P-complete approximation problems*, Journal of the Association for Computing Machinery 23, 555-565, 1976.