A simple and effective metaheuristic for the Minimum Latency Problem

Marcos Melo Silva[a,*], Anand Subramanian[a,b], Thibaut Vidal[c,d], Luiz Satoru Ochi[a]

[a] *Universidade Federal Fluminense, Instituto de Computação, Rua Passo da Pátria 156, Bloco E - 3º andar, São Domingos, Niterói-RJ, 24210-240, Brazil*

[b] *Universidade Federal da Paraíba, Departamento de Engenharia de Produção, Centro de Tecnologia, Campus I - Bloco G, Cidade Universitária, João Pessoa-PB, 58051-970, Brazil*

[c] *CIRRELT, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal H3C 3J7, Canada*

[d] *Institut Charles Delaunay, Université de Technologie de Troyes, 10010 Troyes, France*

Abstract

The Minimum Latency Problem (MLP) is a variant of the Traveling Salesman Problem which aims to minimize the sum of arrival times at vertices. The problem arises in a number of practical applications such as logistics for relief supply, scheduling and data retrieval in computer networks. This paper introduces a simple metaheuristic for the MLP, based on a greedy randomized approach for solution construction and iterated variable neighborhood descent with random neighborhood ordering for solution improvement. Extensive computational experiments on nine sets of benchmark instances involving up to 1000 customers demonstrate the good performance of the method, which yields solutions of higher quality in less computational time when compared to the current best approaches from the literature. Optimal solutions, known for problems with up to 50 customers, are also systemati-

*Corresponding author: Tel. +55 21 2629-5665; Fax +55 21 2629-5666.

Email addresses: `mmsilva@ic.uff.br` (Marcos Melo Silva), `anand@ct.ufpb.br` (Anand Subramanian), `thibaut.vidal@cirrelt.ca` (Thibaut Vidal), `satoru@ic.uff.br` (Luiz Satoru Ochi)

Preprint submitted to European Journal of Operational Research *March 12, 2012*

cally obtained in a fraction of seconds.

Keywords: Metaheuristics, Minimum Latency Problem, GRASP, Iterated Local Search

1. Introduction

The Minimum Latency Problem (MLP) can be defined as follows. Let $G = (V, A)$ be a complete directed graph, where $V = \{0, \dots, n\}$ is the set of vertices and $A = \{(i, j) : i, j \in V,\ i \neq j\}$ is the set of arcs with associated travel time t_{ij}. Vertex 0 stands the depot while other vertices represent customers. The MLP aims at finding a Hamiltonian circuit that minimizes $\sum_{i=0}^{n} l(i)$, $l(i)$ representing the latency of a vertex $i \in V$, that is, the total travel time to reach i. We also consider the additional constraint that the circuit must start and end at vertex 0 and also that $l(0) = 0$. The MLP is a variant of the well-known Traveling Salesman Problem (TSP) and it is known in the literature under various other names: *Traveling Repairman Problem* (Tsitsiklis, 1992), *Delivery Man Problem* (Fischetti et al., 1993), *Cumulative Traveling Salesman Problem* (Bianco et al., 1993) and *School Bus Driver Problem* (Chaudhuri et al., 2003). The constraint on the tour origin at 0 is sometimes not considered in early articles, but this variant can be tackled in our context by adding a dummy depot such that $t_{0j} = t_{j0} = 0$ for $j \in V$.

The MLP was proven $\mathcal{NP}$-hard for general metric spaces (Sahni and Gonzalez, 1976) and also when the subjacent structure is an edge-weighted tree (Sitters, 2002). For structures such as paths, unweighted trees and trees with diameter at most 3, polynomial-time algorithms based on dynamic programming have been proposed (Blum et al., 1994; García et al., 2002; Wu et al., 2004). Although the MLP seems to be a simple TSP variant, one can verify that the former has properties that are not present in the latter. One of them is that small local modifications in the configuration of the input points can lead to highly nonlocal changes in the structure of an optimal solution (Blum et al., 1994; Goemans and Kleinberg, 1998). Another feature of the

MLP is the nonlocal character of the objective function, as an additional arc inserted in the beginning of the circuit affects the latency of all remaining vertices (Arora and Karakostas, 2003).

Real-life applications of the MLP often arise from distribution systems, where some quality criterion regarding the customer satisfaction must be focused. The MLP considers waiting times (latency) of a service system from the customer's point of view, i.e., while in the MLP the objective is to minimize the average waiting time of each customer, in the TSP the objective is to minimize the total time required to visit all customers. In view of this, one can say that the MLP is customer oriented, while the TSP is server oriented (Archer and Williamson, 2003). Therefore, the MLP can be employed in the modeling of different types of service systems. Important practical applications can be found in home delivery services (Méndez-Díaz et al., 2008), logistics for emergency relief (Campbell et al., 2008) and data retrieval in computer networks (Ezzine et al., 2010). However, although the MLP appears in several important settings, this problem has not received sufficient attention in the literature so far. In particular, few efficient heuristics have been designed to tackle problems of realistic size. Moreover, current exact methods are not capable of consistently solving instances with more than 100 customers.

The contributions of this work are twofold. The first one is to present a simple and effective metaheuristic for the MLP, which combines components of Greedy Randomized Adaptive Search Procedure (GRASP) (Feo and Resende, 1995), Iterated Local Search (ILS) (Lourenço et al., 2003) and Variable Neighborhood Descent with Random neighborhood ordering (RVND) (Mladenović and Hansen, 1997; Subramanian et al., 2010). The second contribution is a simple move evaluation procedure that requires $\mathcal{O}(1)$ amortized elementary operations. Such procedure can be applied to any neighborhood structure based on a bounded number of arc exchanges and thus to all classical neighborhoods used in the MLP literature. The proposed solution approach is easy to implement and relies on very few parameters. Extensive computational experiments on benchmark instances involving up to 1000 customers underline the remarkable performance of this method, both in terms of solution quality and computational efficiency. Known optimal solutions for problems with up to 107 customers are also systematically obtained in a few seconds.

The remainder of the paper is organized as follows. Section 2 presents some related works. Section 3 describes the proposed algorithm. Section 4 contains the computational results. Finally, Section 5 presents the concluding remarks of this work.

2. Related Works

Several exact and non-exact approaches were proposed to solve the MLP in the literature. However, as shown in the following, exact algorithms are still limited to small problem sizes, while few efficient heuristic procedures have been designed.

Lucena (1990) proposed an early exact enumerative algorithm, relying on a non-linear integer formulation in which lower bounds are derived using a Lagrangian relaxation. Bianco et al. (1993) put forward two exact algorithms that incorporate lower bounds obtained via a Lagrangian relaxation. Fischetti et al. (1993) proposed an enumerative algorithm that makes use of lower bounds obtained from a linear integer programming formulation. Van Eijl (1995) and Méndez-Díaz et al. (2008) developed Mixed Integer Programming (MIP) formulations. The latter also introduced various families of valid inequalities which were evaluated using a branch-and-cut algorithm. Ezzine et al. (2010) proposed two new integer programming formulations and their linear relaxations were evaluated by means of computational experiments. Bigras et al. (2008) suggested a number of integer programming formulations as well as a branch-and-bound algorithm. Abeledo et al. (2010a,b) developed a branch-cut-and-price approach over an extended formulation as well as several families of facet defining inequalities. To our knowledge, their solution method was the only one capable of solving instances with up to 107 customers.

Some approximation algorithms are also known for the MLP (Blum et al., 1994; Goemans and Kleinberg, 1998; Arora and Karakostas, 2003; Ausiello et al., 2000; Archer and Williamson, 2003; Chaudhuri et al., 2003; Fakcharoenphol et al., 2007; Nagarajan and Ravi, 2008; Archer and Blasiak, 2010). The first one was suggested by Blum et al. (1994) with an approximation factor of 144. For general metric spaces, the smallest approximation factor of 3.59 was found by Chaudhuri et al. (2003). For the case where an edge-weighted tree is considered, the smallest approximation factor, of 3.03, was obtained by Archer and Blasiak (2010).

Up to this date, few metaheuristics are available for the MLP. Salehipour et al. (2011) put forward a heuristic algorithm based on GRASP, Variable Neighborhood Descent (VND) and Variable Neighborhood Search (VNS) with a shaking procedure based on random relocations and neighborhood restrictions to spatially close customers. A Tabu Search (TS) was also developed by Dewilde et al. (2010) for the MLP with profits. Other approaches were designed primarily for the cumulative vehicle routing problem (CCVRP), but can also tackle the MLP as a special case with a single vehicle. The memetic algorithm of Ngueveu et al. (2010), especially, introduces new move evaluation procedures in $\mathcal{O}(1)$ operations for some particular neighborhood structures and produces high quality solutions on the previously mentioned MLP benchmark. Finally, Ribeiro and Laporte (2012) developed an adaptive large neighborhood search, which performed well in solving the CCVRP, but it was not tested on the MLP.

It is worth noting that the MLP and the TSP appear as special cases of a more general problem known as the Time-Dependent TSP (TDTSP). In this problem, the cost associated with the travel path between two vertices depends not only on their localization in the metric space but also on the position in which they appear in the circuit. The objective is to minimize the total cost of visiting all nodes (Abeledo et al., 2010a,b; Blum et al., 1994; Lucena, 1990). The TDTSP and the MLP can be also seen as single-machine scheduling problems with sequence-dependent processing times (Bigras et al., 2008; Gouveia and Voss, 1995; Picard and Queyranne, 1978) and under a "flowtime" performance measure (Tsitsiklis, 1992). Finally, other MLP variants can be also found in the literature. The variant with time windows was studied by Heilporn et al. (2010), Tsitsiklis (1992) and Van Eijl (1995). The case with asymmetric costs was examined by Nagarajan and Ravi (2008) and the variant with multiple servers was tackled by Fakcharoenphol et al. (2007).

3. Proposed Algorithm

The simple and efficient metaheuristic proposed here and called GILS-RVND brings together the components of GRASP, ILS and RVND. The pseudocode of the developed approach is presented in Alg. 1. The method performs I_{Max} iterations (lines 3-21), where in each of which an initial solution is generated using a greedy randomized procedure. The level of greediness is controlled by a parameter α, which is randomly chosen among the values of a given set R. Each initial solution is then improved by means of a RVND procedure combined with a perturbation mechanism in the spirit of ILS (lines 8-15), which is run until I_{ILS} consecutive perturbations without improvements are performed. It is important to mention that the perturbation is always performed on the best current solution s' of a given iteration (acceptance criterion). Finally, the heuristic returns the best solution s^* among all iterations.

3.1. Constructive Procedure

The constructive procedure, used to generate initial solutions, is described in Alg. 2. Firstly, a partial solution s is initialized with a vertex associated

Algorithm 1: GILS-RVND

Procedure GILS-RVND(I_{Max}, I_{ILS}, R)
$f^* \leftarrow \infty$;
for $i \leftarrow 1, \ldots, I_{Max}$ **do**
    $\alpha \leftarrow$ random value $\in R$;
    $s \leftarrow$ Construction(α);
    $s' \leftarrow s$;
    $iterILS \leftarrow 0$;
    while $iterILS < I_{ILS}$ **do**
        $s \leftarrow$ RVND(s);
        if $f(s) < f(s')$ **then**
            $s' \leftarrow s$;
            $iterILS \leftarrow 0$;
        endif
        $s \leftarrow$ Perturb(s');
        $iterILS \leftarrow iterILS + 1$;
    endwhile
    if $f(s') < f^*$ **then**
        $s^* \leftarrow s'$;
        $f^* \leftarrow f(s')$;
    endif
endfor
return s^*;
end GILS-RVND

to the depot (line 2), whereas a Candidate List (CL) is initialized with the remaining vertices (lines 3 and 4). In the main loop (lines 6-13), CL is sorted in ascending order according to the nearest neighbor criterion with respect to the last vertex added to s (line 7). A Restricted Candidate List (RCL) (line 8) is then built by considering only the α% best candidates of CL. Next, a customer is chosen at random from RCL and added to s (lines 9 and 10). When the set of the α% best candidates is of size less than one or when $\alpha = 0$, the algorithm chooses the best candidate. The constructive procedure terminates when all customers are added to s.

Algorithm 2: Construction

Procedure Construction(α)
$s \cup \{0\}$;
Initialize Candidate List CL;
$CL \leftarrow CL - \{0\}$;
$r \leftarrow 0$;
while $CL \neq \emptyset$ **do**
  Sort CL in ascending order according to their distance with respect to r;
  Update RCL considering only the α% best candidates of CL;
  Choose $c \in RCL$ at random;
  $s \cup \{c\}$;
  $r \leftarrow c$;
  $CL \leftarrow CL - \{r\}$;
endwhile
return *s;*
end Construction

3.2. Improvement Procedure with Efficient Move Evaluations

The local search is performed by a method based on RVND. Let t be the number of neighborhood structures and $N = \{N^1, N^2, N^3, \ldots, N^t\}$ be their corresponding set. Whenever a given neighborhood of the set N fails to improve the current best solution, RVND randomly selects another neighborhood from the same set to continue the search. Preliminary tests revealed that this approach is capable of finding better solutions as compared to those that adopt a deterministic order.

The set N is composed of the following five well-known TSP neighborhood structures, whose associated solutions are explored in an exhaustive fashion with a best improvement strategy.

- **Swap** — $N^{(1)}$ — Two customers of the tour are interchanged.
- **2**-opt — $N^{(2)}$ — Two non-adjacent arcs are removed and another two are inserted in order to build a new feasible tour.

- **Reinsertion** — $N^{(3)}$ — One customer is relocated to another position of the tour.

- **Or**-opt2 — $N^{(4)}$ — Two adjacent customers are reallocated to another position of the tour.

- **Or**-opt3 — $N^{(5)}$ — Three adjacent customers are reallocated to another position of the tour.

Algorithm 3: RVND

Procedure RVND(s)
Initialize the Neighborhood List NL;
Initialize re-optimization data structures on subsequences;
while $NL \neq \emptyset$ **do**
    Choose a neighborhood $N^{(\eta)} \in NL$ at random;
    Find the best neighbor s' of $s \in N^{(\eta)}$;
    if $f(s') < f(s)$ **then**
        $s \leftarrow s'$;
        $f(s) \leftarrow f(s')$;
        Update NL;
        Update re-optimization data structures;
    else
        Remove $N^{(\eta)}$ from the NL;
    endif
endwhile
return s;
end RVND

It is worth emphasizing that neighbor evaluations in the MLP case are less straightforward than for the classic TSP, since the time of service of a large proportion of customers are impacted during moves. A direct cost evaluation procedure involves examining customer visits in the sequence order with a view of computing their latencies and thus the total cost. This method unfortunately leads to $\mathcal{O}(n)$ operations for each move evaluation, resulting in $\mathcal{O}(n^3)$ operations for a full neighborhood search. In the spirit of the feasibility checking approach of Savelsbergh (1985), originally designed for the vehicle routing problem with time windows, Ngueveu et al. (2010) proposed a move evaluation procedure, which requires $\mathcal{O}(1)$ amortized operations. Such procedure is based on the management of global information on partial routes.

We follow this line of thought and propose a very simple move evaluation approach, also requiring $\mathcal{O}(1)$ amortized operations, which can be generally applied to any neighborhood structure based on a bounded number of arc exchanges. The approach relies on a re-optimization framework "by concatenation", originally developed by Kindervater and Savelsbergh (1997) and extended by Vidal et al. (2011) to a wide range of move evaluation settings presenting temporal characteristics. Indeed, any arc exchange-based move involves separating the visit sequence into several subsequences, which are then concatenated together. We thus introduce some "re-optimization data structures" to characterize the cost of subsequences $\sigma = (\sigma_u, \dots, \sigma_v)$, and show how to update them on larger subsequences by induction on the concatenation operator, here defined as $\oplus$. The following data structures are used:

- Duration $T(\sigma)$ to perform the visits sequence σ;
- Cost $C(\sigma)$ to perform the sequence, when starting at time 0;
- Delay cost $W(\sigma)$, related to a one time unit delay in the starting time.

For a sequence with a single vertex $\sigma^0 = i$, the duration $T(\sigma^0)$ is 0 by default since there is no travel time. The constant cost $C(\sigma^0)$ is set to 0, whereas the delay cost $W(\sigma^0)$ is set to 1 when the vertex is a customer, otherwise $W(\sigma^0) = 0$. Proposition 1 then enables to compute these values on larger subsequences by induction on the concatenation operator.

Proposition 1. *Let* $\sigma = (\sigma_u, \ldots, \sigma_v)$ *and* $\sigma' = (\sigma'_w, \ldots, \sigma'_x)$ *be two sub-sequences of visits. The sub-sequence* $\sigma \oplus \sigma' = (\sigma_u, \ldots, \sigma_v, \sigma'_w, \ldots, \sigma'_x)$ *is characterized by the following values:*

$$T(\sigma \oplus \sigma') = T(\sigma) + t_{\sigma_v \sigma'_w} + T(\sigma') \tag{1}$$

$$C(\sigma \oplus \sigma') = C(\sigma) + W(\sigma')(T(\sigma) + t_{\sigma_v \sigma'_w}) + C(\sigma') \tag{2}$$

$$W(\sigma \oplus \sigma') = W(\sigma) + W(\sigma') \tag{3}$$

The proposed neighborhood evaluation procedure relies on Proposition 1 to compute the cost of relevant sub-sequences (and their reversal for 2-opt moves), during a preprocessing phase in $\mathcal{O}(n^2)$ operations; and then to evaluate the cost $C(\sigma)$ of tours issued from moves as the concatenation of several sub-sequences. Classical neighborhoods in the literature correspond to a concatenation of less than five sub-sequences. Hence, when data on sub-sequences has been processed, any move evaluation is performed in a constant number of operations and thus a full neighborhood is performed in $\mathcal{O}(n^2)$ operations. An illustrative example of the re-optimization data structures and the cost evaluation is given in Appendix A.

The pseudocode of the RVND procedure is described in Algorithm 3. Firstly, the Neighborhood List (NL) is initialized with all neighborhood structures, as well as the re-optimization data structures (lines 2-3). In the main loop (lines 4-15), a given neighborhood structure $N^{(\eta)}$ is selected at random from the NL (line 5). Neighbor solutions are evaluated in $\mathcal{O}(1)$ operations, using the re-optimization data structures and the best found solution is stored in s' (line 6). In case of improvement, NL is repopulated with all neighborhood structures and the re-optimization data structures are updated (lines 7-12). Otherwise, $N^{(\eta)}$ is removed from NL (line 13). The procedure terminates when NL becomes empty.

3.3. Perturbation Mechanism

When the local search fails to improve a solution s, a perturbation is applied over the best current solution s' of the corresponding GILS-RVND iteration. The perturbation mechanism, called double-bridge, was originally developed by Martin et al. (1991) for the TSP. It consists in removing and re-inserting four arcs in such a way that a feasible tour is generated (see Fig. 1). This mechanism can also be seen as a permutation of two disjoint segments of a tour.

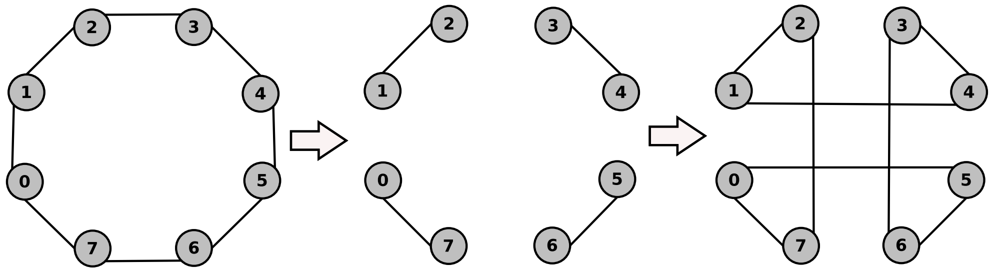

Figure 1: Double-Bridge

4. Computational Results

The algorithm was coded in C++ (g++ 4.4.3) and executed on an Intel® Core$^{\text{TM}}$ i7 2.93 GHz, with 8.0 GB of RAM memory running under GNU/Linux Ubuntu 10.04 (*kernel* 2.6.32-25). Only a single thread was used in the experiments.

Through preliminary tests, we observed that the values $I_{Max} = 10$, $I_{ILS} = \min\{100, n\}$ and $R = \{0.00, 0.01, 0.02, \ldots, 0.25\}$ resulted in a good trade-off between solution quality and run time. This parameter setting has thus been used in the following experiments.

GILS-RVND was tested on 9 sets of benchmark instances. Seven of these sets were generated by Salehipour et al. (2011), where each of them is composed of 20 instances with 10, 20, 50, 100, 200, 500 and 1000 customers, respectively. Another set of 10 instances ranging from 70 to 532 customers was selected from the TSPLIB (Reinelt, 1991) by the same authors. It is important to mention that Salehipour et al. (2011) considered the version where the objective is to find the minimum latency over a Hamiltonian path starting from vertex 0. Finally, we present results on a last subset of instances from the TSPLIB, selected by Abeledo et al. (2010a,b) and composed of 23 test-problems ranging between 42 and 107 customers.

Until this date, optimal solutions were only reported by Salehipour et al. (2011) for instances with up to 20 customers. We had access to the source code of the BCP approach of Abeledo et al. (2010a), and ran their algorithm to find the optimal solutions of all instances with up to 50 customers. Salehipour et al. (2011) and Ngueveu et al. (2010) also did not report detailed results for every instance, but only the average gap between their solutions and those obtained by a greedy nearest neighbor heuristic on each group of instances. Hence, in order to compare our heuristic with the best methods developed by Salehipour et al. (2011) and Ngueveu et al. (2010), we report the average solution quality of GILS-RVND over 10 runs, for each instance, relative to the nearest neighbor heuristic, which is equivalent to our constructive procedure with $\alpha = 0$.

In the tables presented hereafter, **Instance** denotes the instance, **OPT** is the optimal solution, **UB** indicates the upper bound obtained by the nearest neighbor heuristic, **Best Sol.** corresponds to the best solution obtained by GILS-RVND and **Avg. Sol** denotes the average solution of 10 executions found by GILS-RVND. **Avg. Time** is the average time in seconds of 10 executions of the proposed algorithm, while **cTime** represents scaled run times, estimated on a Pentium IV by means of the factors of Dongarra (2011). Finally, **Avg. Gap** is the average gap between the average solution and either the optimal solution (Avg. Gap = (100(Avg. Sol - OPT)/OPT)), when available, or the UB given by the nearest neighbor heuristic (Avg. Gap = (100(Avg. Sol - UB)/UB)). Improved solutions are highlighted in boldface.

Table 1 compares the solutions obtained by the exact algorithm of Abeledo et al. (2010a) and those found by GILS-RVND for the set of 23 test-problems selected from the TSPLIB by Abeledo et al. (2010a). GILS-RVND found the optimal solutions, when available, in all executions. Moreover, new best solutions were found for the two remaining open instances (rat99 and eil101).

Table 2 compares the best solutions obtained by Salehipour et al. (2011) and those found by GILS-RVND for the set of 10 instances selected from the TSPLIB by Salehipour et al. (2011). It can be observed that GILS-RVND was capable of equaling or improving the best known solutions in all cases. The average gap between the average solutions obtained by GILS-RVND and best solutions reported by Salehipour et al. (2011) was -2.34%. When computing the average, we did not consider the result of the last instance, called "att532", since the large gap between the two methods seemed abnormal.

Table 3 presents the average results of GILS-RVND on the instances of size 10, 20 and 50, generated by Salehipour et al. (2011). These solutions are compared with optimal values, found either by Salehipour et al. (2011) for problems with less than 20 customers, or by Abeledo et al. (2010a) for problems with 50 customers. It is noteworthy that GILS-RVND found optimal solutions during all executions, with an average computational time of 0.01, 0.02 and 0.55 seconds, respectively.

Tables 4, 5, 6 and 7 compare the results of GILS-RVND with the upper bounds obtained using the nearest neighbor heuristic on the remaining instances suggested by Salehipour et al. (2011) with 100, 200, 500 and 1000 customers. As expected, the proposed method largely improves upon the nearest neighbor heuristic, with average gaps of -13.00%, -14.35%, -15.16% and -15.47%, respectively.

Finally, Table 8 presents a summarized comparison between the average gaps, with respect to the nearest neighbor heuristic, found by GILS-RVND, Salehipour et al. (2011) (GRASP+VNS deep version) and Ngueveu et al. (2010) (memetic deep version) for the seven sets of instances of Salehipour et al. (2011). The proposed algorithm led to larger improvements for most

Table 1: Results for TSPLIB instances selected by Abeledo et al. (2010a,b)

Instance	Abeledo *et. al.* OPT or UB	GILS-RVND Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)
dantzig42	12528	12528	12528.00	0.00	0.16
swiss42	22327	22327	22327.00	0.00	0.16
att48	209320	209320	209320.00	0.00	0.32
gr48	102378	102378	102378.00	0.00	0.33
hk48	247926	247926	247926.00	0.00	0.30
eil51	10178	10178	10178.00	0.00	0.49
berlin52	143721	143721	143721.00	0.00	0.46
brazil58	512361	512361	512361.00	0.00	0.78
st70	20557	20557	20557.00	0.00	1.65
eil76	17976	17976	17976.00	0.00	2.64
pr76	3455242	3455242	3455242.00	0.00	2.31
pr76r	345427	345427	345427.00	0.00	2.41
gr96	2097170	2097170	2097170.00	0.00	6.19
rat99	58288*	**57986**	**57986.00**	-0.52	11.27
kroA100	983128	983128	983128.00	0.00	8.59
kroB100	986008	986008	986008.00	0.00	9.21
kroC100	961324	961324	961324.00	0.00	8.17
kroD100	976965	976965	976965.00	0.00	8.46
kroE100	971266	971266	971266.00	0.00	8.31
rd100	340047	340047	340047.00	0.00	8.52
eil101	27519*	**27513**	**27513.00**	-0.02	12.76
lin105	603910	603910	603910.00	0.00	8.42
pr107	2026626	2026626	2026626.00	0.00	10.89

* Optimality is not proven for this instance

Table 2: Results for TSPLIB instances selected by Salehipour et al. (2011)

Instance	Salehipour *et. al.* Best Sol.	GILS-RVND Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)
st70	19553	**19215**	**19215.00**	-1.73	1.51
rat99	56994	**54984**	**54984.00**	-3.53	9.47
kroD100	976830	**949594**	**949594.00**	-2.79	6.90
lin105	585823	585823	585823.00	0.00	6.19
pr107	1983475	**1980767**	**1980767.00**	-0.14	8.13
rat195	213371	**210191**	**210335.90**	-1.42	75.56
pr226	7226554	**7100308**	**7100308.00**	-1.75	59.05
lin318	5876537	**5560679**	**5569819.50**	-5.22	220.59
pr439	18567170	**17688561**	**17734922.00**	-4.48	553.74
att532	18448435	5581240	5597866.80	–	1792.61
Average				-2.34	

Table 3: Results obtained on the small instances generated by Salehipour et al. (2011), involving 10, 20 and 50 customers

	OPT		
Instance	S10	S20	S50
TRP-Sn-R1	1303	3175	12198
TRP-Sn-R2	1517	3248	11621
TRP-Sn-R3	1233	3570	12139
TRP-Sn-R4	1386	2983	13071
TRP-Sn-R5	978	3248	12126
TRP-Sn-R6	1477	3328	12684
TRP-Sn-R7	1163	2809	11176
TRP-Sn-R8	1234	3461	12910
TRP-Sn-R9	1402	3475	13149
TRP-Sn-R10	1388	3359	12892
TRP-Sn-R11	1405	2916	12103
TRP-Sn-R12	1150	3314	10633
TRP-Sn-R13	1531	3412	12115
TRP-Sn-R14	1219	3297	13117
TRP-Sn-R15	1087	2862	11986
TRP-Sn-R16	1264	3433	12138
TRP-Sn-R17	1058	2913	12176
TRP-Sn-R18	1083	3124	13357
TRP-Sn-R19	1394	3299	11430
TRP-Sn-R20	951	2796	11935

instance categories and also required smaller scaled computational time, thus outperforming previous methods.

5. Concluding Remarks

This paper introduced a new hybrid metaheuristic for the Minimum Latency Problem, which gathers several successful concepts from GRASP, ILS, RVND, along with simple move evaluation procedures in $\mathcal{O}(1)$ time. The latter methodology can be applied to any neighborhood structure based on a bounded number of arc exchanges or visit relocations. The overall approach is simple to describe and to implement. Its effectiveness, in terms of both solution quality and computational time, was assessed by extensive experiments on 173 benchmark instances containing up to 1000 customers. The

Table 4: Results on the 100-customers instances generated by Salehipour et al. (2011)

Instance	UB	GILS-RVND Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)
TRP-S100-R1	35334	32779	32779.00	-7.23	7.05
TRP-S100-R2	38442	33435	33435.00	-13.02	7.51
TRP-S100-R3	37642	32390	32390.00	-13.95	7.07
TRP-S100-R4	37508	34733	34733.00	-7.40	7.27
TRP-S100-R5	37215	32598	32598.00	-12.41	8.87
TRP-S100-R6	40422	34159	34159.00	-15.49	7.82
TRP-S100-R7	37367	33375	33375.00	-10.68	8.74
TRP-S100-R8	38086	31780	31780.00	-16.56	7.08
TRP-S100-R9	36000	34167	34167.00	-5.09	7.47
TRP-S100-R10	37761	31605	31605.00	-16.30	6.78
TRP-S100-R11	37220	34188	34188.00	-8.15	7.75
TRP-S100-R12	34785	32146	32146.00	-7.59	7.20
TRP-S100-R13	37863	32604	32604.00	-13.89	7.78
TRP-S100-R14	36362	32433	32433.00	-10.81	6.29
TRP-S100-R15	39381	32574	32574.00	-17.28	7.13
TRP-S100-R16	39823	33566	33566.00	-15.71	7.97
TRP-S100-R17	41824	34198	34198.00	-18.23	9.23
TRP-S100-R18	39091	31929	31929.00	-18.32	7.07
TRP-S100-R19	39941	33463	33463.00	-16.22	8.08
TRP-S100-R20	39888	33632	33632.00	-15.68	8.73
Average				-13.00	

Table 5: Results on the 200-customers instances generated by Salehipour et al. (2011)

Instance	UB	GILS-RVND Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)
TRP-S200-R1	105044	88787	88794.60	-15.47	73.39
TRP-S200-R2	104073	91977	92013.10	-11.59	68.07
TRP-S200-R3	111644	92568	92631.20	-17.03	67.11
TRP-S200-R4	104956	93174	93192.30	-11.21	72.17
TRP-S200-R5	101912	88737	88841.20	-12.83	70.77
TRP-S200-R6	103751	91589	91601.90	-11.71	70.52
TRP-S200-R7	109810	92754	92763.20	-15.52	72.80
TRP-S200-R8	103830	89048	89049.00	-14.24	75.15
TRP-S200-R9	100946	86326	86326.00	-14.48	65.45
TRP-S200-R10	108061	91552	91596.50	-15.24	74.25
TRP-S200-R11	103297	92655	92700.60	-10.26	73.15
TRP-S200-R12	107715	91457	91504.10	-15.05	76.74
TRP-S200-R13	100505	86155	86181.40	-14.25	72.96
TRP-S200-R14	107543	91882	91929.10	-14.52	70.94
TRP-S200-R15	100196	88912	88912.40	-11.26	70.41
TRP-S200-R16	104462	89311	89364.70	-14.45	77.89
TRP-S200-R17	107216	89089	89118.30	-16.88	71.17
TRP-S200-R18	108148	93619	93676.60	-13.38	77.03
TRP-S200-R19	105716	93369	93401.60	-11.65	71.08
TRP-S200-R20	116676	86292	86292.00	-26.04	70.61
Average				-14.35	

Table 6: Results on the 500-customers instances generated by Salehipour et al. (2011)

Instance	UB	GILS-RVND Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)
TRP-S500-R1	2192688	1841386	1856018.70	-15.35	1738.48
TRP-S500-R2	2176449	1816568	1823196.90	-16.23	1476.13
TRP-S500-R3	2261125	1833044	1839254.20	-18.66	1557.48
TRP-S500-R4	2088773	1809266	1815876.40	-13.06	1597.06
TRP-S500-R5	2216937	1823975	1834031.70	-17.27	1530.94
TRP-S500-R6	2137187	1786620	1790912.40	-16.20	1576.91
TRP-S500-R7	2212936	1847999	1857926.60	-16.04	1584.67
TRP-S500-R8	2132165	1820846	1829257.30	-14.21	1565.01
TRP-S500-R9	2141458	1733819	1737024.90	-18.89	1409.23
TRP-S500-R10	2163387	1762741	1767366.30	-18.31	1621.85
TRP-S500-R11	2288538	1797881	1801467.90	-21.28	1530.98
TRP-S500-R12	2081530	1774452	1783847.10	-14.30	1554.75
TRP-S500-R13	2080370	1873699	1878049.40	-9.73	1598.46
TRP-S500-R14	2051683	1799171	1805732.90	-11.99	1701.90
TRP-S500-R15	2035804	1791145	1797532.90	-11.70	1623.79
TRP-S500-R16	2142426	1810188	1816484.00	-15.21	1583.70
TRP-S500-R17	2117999	1825748	1834443.20	-13.39	1549.80
TRP-S500-R18	2159400	1826263	1833323.70	-15.10	1620.02
TRP-S500-R19	2009335	1779248	1782763.90	-11.28	1602.87
TRP-S500-R20	2155026	1820813	1830483.30	-15.06	1507.96
Average				-15.16	

Table 7: Results on the 1000-customers instances generated by Salehipour et al. (2011)

Instance	UB	GILS-RVND Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)
TRP-S1000-R1	6030081	5107395	5133698.30	-14.87	31894.51
TRP-S1000-R2	6282704	5106161	5127449.40	-18.39	30881.19
TRP-S1000-R3	5874496	5096977	5113302.90	-12.96	30184.15
TRP-S1000-R4	5892443	5118006	5141392.60	-12.75	29951.12
TRP-S1000-R5	6192250	5103894	5122660.70	-17.27	30129.51
TRP-S1000-R6	6056173	5115816	5143087.10	-15.08	28161.57
TRP-S1000-R7	5973701	5021383	5032722.00	-15.75	25945.41
TRP-S1000-R8	6046965	5109325	5132722.60	-15.12	26572.71
TRP-S1000-R9	6159862	5052599	5073245.30	-17.64	26330.40
TRP-S1000-R10	5843354	5078191	5093592.60	-12.83	25676.31
TRP-S1000-R11	6057225	5041913	5066161.50	-16.36	26235.63
TRP-S1000-R12	5996323	5029792	5051235.20	-15.76	27910.11
TRP-S1000-R13	6052162	5102520	5131437.50	-15.21	28475.89
TRP-S1000-R14	5952120	5099433	5118980.60	-14.00	27639.81
TRP-S1000-R15	5934175	5142470	5174493.20	-12.80	27633.07
TRP-S1000-R16	5925180	5073972	5090280.50	-14.09	26653.16
TRP-S1000-R17	6068380	5071485	5084450.40	-16.21	27503.43
TRP-S1000-R18	6169728	5017589	5037094.00	-18.36	28808.09
TRP-S1000-R19	6004893	5076800	5097167.60	-15.12	29637.49
TRP-S1000-R20	6159355	4977262	5002920.60	-18.78	27499.24
Average				-15.47	

Table 8: Comparison among different solution approaches.

n	Salehipour *et. al.* UB (%)	Salehipour *et. al.* Time (s)	Nguevu *et. al.* UB (%)	Nguevu *et. al.* cTime (s)	GILS-RVND UB (%)	GILS-RVND Avg. Time (s)	GILS-RVND cTime (s)
10	-2.44	0.00	-2.43	0.00	-2.44	0.00	0.00
20	-9.86	0.04	-10.11	0.01	-10.28	0.02	0.05
50	-9.74	3.54	-9.36	1.44	-11.01	0.55	1.36
100	-11.66	103.92	-11.95	93.26	-13.00	7.64	18.94
200	-16.21	3995.00	-12.81	938.16	-14.35	72.08	178.72
500	-9.71	10381.36	-13.85	16208.70	-15.16	1576.60	3909.03
1000	–	–	–	–	-15.47	28186.14	69884.87
Average	-9.94	2413.06	-10.09	2873.60	-11.04	276.15	684.68
					-11.67*	4263.29*	10570.42*

* Average considering the instances with 1000 customers.

method systematically obtains the optimal solution in a few seconds on instances with up to 107 customers, where this value is known. Moreover, all best known solutions of the benchmark instances have been either equaled or improved.

Promising avenues of research involve extending the proposed algorithm to other MLP variants such as the MLP with profits and the version with multiple vehicles.

References

Abeledo, H., Fukasawa, R., Pessoa, A., Uchoa, E., December 2010a. The time dependent traveling salesman problem: Polyhedra and algorithm. Tech. Rep. RPEP Vol.10 n.15, Universidade Federal Fluminense, Brasil.

Abeledo, H. G., Fukasawa, R., Pessoa, A. A., Uchoa, E., 2010b. The time dependent traveling salesman problem: Polyhedra and branch-cut-and-price algorithm. In: Proceedings of the 9th International Symposium on Experimental Algorithms, SEA 2010. pp. 202–213.

Archer, A., Blasiak, A., 2010. Improved approximation algorithms for the minimum latency problem via prize-collecting strolls. In: Proceedings of the 21th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 429–447.

Archer, A., Williamson, D. P., 2003. Faster approximation algorithms for the minimum latency problem. In: Proceedings of the 40th Annual ACM-SIAM Symposium on Discrete algorithms. pp. 88–96.

Arora, S., Karakostas, G., May 2003. Approximation schemes for minimum latency problems. SIAM Journal on Computing 32 (5), 1317–1337.

Ausiello, G., Leonardi, S., Marchetti-Spaccamela, A., 2000. On salesmen, repairmen, spiders, and other traveling agents. In: Proceedings of the 4th Italian Conference on Algorithms and Complexity. pp. 1–16.

Bianco, L., Mingozzi, A., Ricciardelli, S., 1993. The traveling salesman problem with cumulative costs. Networks 23 (2), 81–91.

Bigras, L.-P., Gamache, M., Savard, G., 2008. The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. Discrete Optimization 5 (4), 685–699.

Blum, A., Chalasanit, P., Pulleyblankt, B., Raghavan, P., Sudan, M., 1994. The minimum latency problem. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing. pp. 163–171.

Campbell, A., Vandenbussche, D., Hermann, W., 2008. Routing for relief efforts. Transportation Science 42 (2), 127–145.

Chaudhuri, K., Godfrey, B., Rao, S., Talwar, K., 2003. Paths, trees, and minimum latency tours. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003. pp. 36–45.

Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F. C. R., Vansteenwegen, P., 2010. Heuristics for the traveling repairman problem with profits. In:

Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2010. pp. 34–44.

Dongarra, J. J., 2011. Performance of various computers using standard linear equations software. Tech. Rep. CS-89-85, Computer Science Department, University of Tennessee, Knoxville, TN, USA.

Ezzine, I. O., Semet, F., Chabchoub, H., 2010. New formulations for the traveling repairman problem. In: Proceedings of the 8th International Conference of Modeling and Simulation.

Fakcharoenphol, J., Harrelson, C., Rao, S., November 2007. The k-traveling repairmen problem. ACM Transactions on Algorithms 3 (4), 40+.

Feo, T. A., Resende, M. G. C., 1995. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization 6 (2), 109–133.

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

García, A., Jodrá, P., Tejel, J., 2002. A note on the traveling repairman problem. Networks 40 (1), 27–31.

Goemans, M. X., Kleinberg, J. M., 1998. An improved approximation ratio for the minimum latency problem. Mathematical Programming 82 (1), 111–124.

Gouveia, L., Voss, S., May 1995. A classification of formulations for the (time-dependent) traveling salesman problem. European Journal of Operational Research 83 (1), 69–82.

Heilporn, G., Cordeau, J.-F., Laporte, G., 2010. The delivery man problem with time windows. Discrete Optimization 7 (4), 269–282.

Kindervater, G., Savelsbergh, M., 1997. Vehicle routing: Handling edge exchanges. In: Aarts, E., Lenstra, J. (Eds.), Local Search in Combinatorial Optimization. Wiley, New York, pp. 337–360.

Lourenço, H., Martin, O., Stützle, T., 2003. Iterated Local Search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Kluwer Academic Publishers, pp. 320–353.

Lucena, A., 1990. time-dependent traveling salesman problem - The deliveryman case. Networks 20 (6), 753–763.

Martin, O., Otto, S. W., Felten, E. W., 1991. Large-step Markov chains for the traveling salesman problem. Complex Systems 5 (3), 299–326.

Méndez-Díaz, I., Zabala, P., Lucena, A., 2008. A new formulation for the traveling deliveryman problem. Discrete Applied Mathematics 156 (17), 3223–3237.

Mladenović, N., Hansen, P., November 1997. Variable Neighborhood Search. Computers & Operations Research 24 (11), 1097–1100.

Nagarajan, V., Ravi, R., 2008. The directed minimum latency problem. In: Proceedings of the 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques. pp. 193–206.

Ngueveu, S., Prins, C., Wolfler Calvo, R., 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Computers & Operations Research 37 (11), 1877–1885.

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

Reinelt, G., 1991. TSPLIB–A traveling salesman problem library. INFORMS Journal on Computing 3 (4), 376–384.

Ribeiro, G., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Computers & Operations Research 39 (3), 728–735.

Sahni, S., Gonzalez, T., July 1976. P-complete approximation problems. Journal of The ACM 23 (3), 555–565.

Salehipour, A., Sörensen, K., Goos, P., Bräysy, O., 2011. Efficient GRASP+VND and GRASP+VNS metaheuristics for the traveling repairman problem. 4OR: A Quarterly Journal of Operations Research 9 (2), 189–209.

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

Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S., Farias, R., November 2010. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research 37 (11), 1899–1911.

Tsitsiklis, J. N., 1992. Special cases of traveling salesman and repairman problems with time windows. Networks 22 (3), 263–282.

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

Vidal, T., Crainic, T., Gendreau, M., Prins, C., 2011. A unifying view on timing problems and algorithms. Tech. rep., CIRRELT.

Wu, B. Y., Huang, Z.-N., Zhan, F.-J., December 2004. Exact algorithms for the minimum latency problem. Information Processing Letters 92 (6), 303–309.

Appendix A. Example of efficient move evaluation for the MLP

Figure A.2 presents a numerical example on a small problem with five vertices. The cost of each arc is indicated in italics. An Or-opt2 move is illustrated, involving the relocation of customers 2 and 3 after customer 6. Starting from the sequence $(0, 1, 2, 3, 4, 5, 6, 0)$, this move produces a concatenation of four sub-sequences $(0, 1) \oplus (4, 5, 6) \oplus (2, 3) \oplus (0)$.

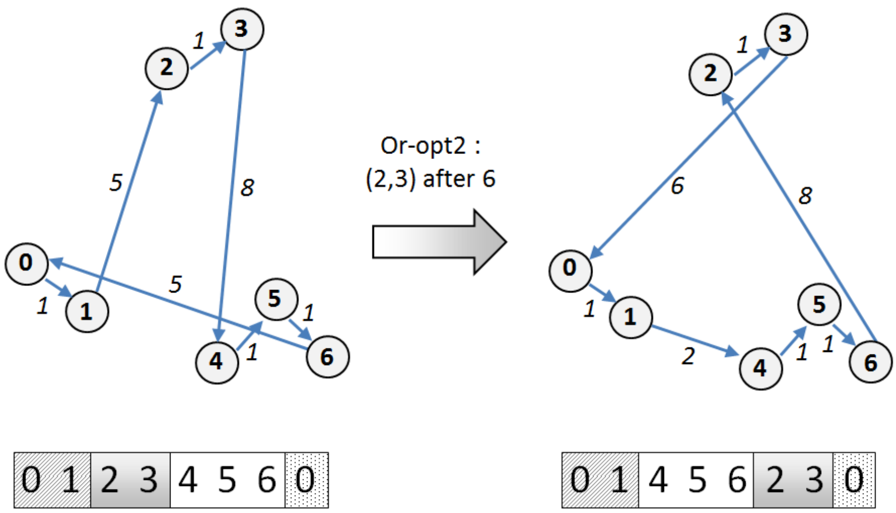

Figure A.2: Or-opt2 move as a concatenation of four sub-sequences

The relevant preprocessed data structures are reported in the top of Table A.9. The cost of the new sequence is then obtained by applying the concatenations equations three times, to compute $C((0, 1) \oplus (4, 5, 6))$ and so on.

Table A.9: Preprocessed data structures and partial results for a sequence evaluation

σ	$T(\sigma)$	$C(\sigma)$	$W(\sigma)$
$(0,1)$	1	1	1
$(4,5,6)$	2	3	3
$(2,3)$	1	1	2
(0)	0	0	0
$(0,1) \oplus (4,5,6)$	5	13	4
$(0,1) \oplus (4,5,6) \oplus (2,3)$	14	40	6
$(0,1) \oplus (4,5,6) \oplus (2,3) \oplus (0)$	14	**40**	6