Noname manuscript No.
(will be inserted by the editor)

A biobjective study of the minimum latency problem

N.A. Arellano-Arriaga · J. Molina · S.E. Schaeffer · A.M. Álvarez-Socarrás · I.A. Martínez-Salazar

Received: date / Accepted: date

Abstract We study a bi-objective problem which aims to minimize distance and latency of a single-vehicle route designed to serve a set of client requests. This route satisfies the Hamiltonian cycle definition and we aim to simultaneously minimize the traveled distance of the vehicle as well as the total waiting time of the clients along the route. This problem is considered relevant in contexts where both client service and company profit are priorities. We call it the Minimum Latency-Distance Problem (MLDP) and present a mixed-integer mathematical formulation, a computational analysis, and the introduction of two metaheuristic methods created for this particular problem: one based on a classic multi-objective algorithm as well as a novel genetic algorithm which involves an intelligent selection of the neighborhoods in its local search procedure to approximate the Pareto fronts.

Keywords Combinatorial optimization · Genetic algorithms · Metaheuristics · Multiple-objective programming · Multi-objective optimization · Latency · Distance

N. A. Arellano-Arriaga
PISIS, FIME, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Mexico
Programa de Doctorado de Economía y Empresa, Universidad de Málaga, Malaga, Spain
E-mail: nancyarellanoarg@uanl.edu.mx, nancy.arellano@uma.es

J. Molina
Departamento de Economía Aplicada (Matemáticas), Universidad de Málaga, Malaga, Spain
E-mail: julian.molina@uma.es

S. E. Schaeffer · A. M. Álvarez-Socarrás · I. A. Martínez-Salazar
PISIS, FIME, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Mexico
E-mail: elisa.schaeffer@uanl.edu.mx, iris.martinezsalaz@uanl.edu.mx, ada.alvarezs@uanl.mx

1 Introduction

Multi-objective optimization refers to decision making based on several objectives and the identification of a suitable trade-off among them (Ehrgott 2005; Sengupta et al 2016). In this paper we formulate a bi-objective problem that commonly arises in the context of logistic activities of companies that offer delivery or maintenance services by taking into consideration two objective functions: an economic objective to preserve the profit of the company, and a customer-centered objective to improve the quality of the service by minimizing the total waiting time of each client. This bi-objective problem combines the Minimum Latency Problem (MLP) (Blum et al 1994; Chaudhuri et al 2003; García et al 2002; Lucena 1990; Nagarajan and Ravi 2008) and the Traveling Salesman Problem (TSP) (Applegate et al 2007; Gutin and Punnen 2006; Hoffman et al 2013; Laporte 1992); the former aims to minimize the total waiting time of a set of clients in a route meanwhile the latter aims to minimize the total distance traveled by the vehicle.

Our bi-objective formulation assigns the same importance to the traveled distance and to the waiting time, hence taking into account distance-dependent quantities such as fuel cost simultaneously with quality-of-service aspects such as the waiting time of the clients. Our main contributions are the development of a mixed-integer model to represent this bi-objective problem, an analysis of the computational complexity as well as the design, development and comparison of two metaheuristics methods, both based on the classic NSGA-II algorithm (Deb et al 2000).

The rest of the paper is organized as follows: in Section 2 we present a literature review. Section 3 introduces the assumptions made to formulate this problem as well as the mathematical formulation for it. In Section 4, the computational analysis of this problem is presented. Section 5 presents the metaheuristic algorithms proposed and Section 6 shows the comparison among them. Finally, in Section 7, the conclusions and potential future lines of work are described.

2 Literature review

The TSP is a classic combinatorial problem studied in Operations Research (Applegate et al 2007; Gutin and Punnen 2006; Laporte 1992), which stands for the simplicity with which it is stated and the contrasting complexity it actually has (Applegate et al 2007). In literature, there are several formulations designed to minimize the distance of a single-route vehicle (Laporte 1992; Miller et al 1960; Orman and Williams 2007), generalizations for m-vehicles (M-TSP) (Bektas 2006; Noon and Bean 1993), and numerous approximation algorithms (Chvátal et al 2010; Dorigo and Gambardella 1997; Lin and Kernighan 1973).

In recent years, the interest in the MLP has grown in the Operations Research area (Blum et al 1994; Chaudhuri et al 2003; García et al 2002; Lucena 1990), leading to the development of several formulations to study the min-

imization of the latency objective (Angel-Bello et al 2013; Gouveia and Voß 1995; Méndez-Díaz et al 2008; Picard and Queyranne 1978; Sarubbi et al 2008). This objective has been specially studied in humanitarian logistic research (Ferrer et al 2016; Kovács and Spens 2009, 2007; Stephenson 2017; Tomasini et al 2009; Vargas et al 2017) in recent years. Nonetheless, as by definition, *latency* refers to the delay between the moment a request for an action is made and the moment that action is executed, the study of the minimization of the waiting time can also be found in Medicine (Darcis et al 2017; Littner et al 2005; Paniagua-Soto et al 2013; Squires et al 1975; Sternberg et al 1978) and Telecommunications (Crowcroft et al 2015; Grout et al 2007; Gummadi et al 2002; Joo Ghee et al 2009; Kao et al 2017; Lichtsteiner et al 2008; Lu et al 2007; Schurgers et al 2002).

Our proposed bi-objective approach considers a single-vehicle route and aims to minimize simultaneously the distance and the latency of that route. This combination of objectives has been seen in several medical (Bair et al 2003; Fischer et al 1987; Salami et al 2003) and network applications (Borah et al 2017; Depuy et al 2001; He et al 2015; Kim and Na 2017; Madhyastha et al 2006; Patterson 2004; Sarddar et al 2010, 2011) with the goal of measuring a specific growth that depends on the distance. We propose considering both objectives in a routing context with the purpose of turning a classic routing problem into a client-centered one. The motivation of such an approach is to provide an equilibrium between the two objectives in the decision-making process: the client and the company. This approach aims to the improvement of several aspects a classic approach might not consider such as the attempt of incorporating a quality of the service factor to the routing decision, so as to improve the competitive position of the company as well as the clients satisfaction (Gary L. Clark and Rink 1992; Oliver 1987). We call this problem the Minimum Latency-Distance Problem and, hereinafter, it will be referred to as MLDP.

3 Minimum Latency-Distance Problem

To define the MLDP, let us consider a company that provides a service to a set of clients through a single agent. This agent departs from a known depot and returns to it at the end of the day, visiting each of the clients exactly once. The main goal of this single-vehicle route is the minimization of the total traveled distance of the agent as well as the minimization of the total waiting time of the clients waiting to be served. To obtain a formulation for the MLDP, we assume that the duration of each service, the exact distance between the depot and the clients, as well as the distance between all the clients are known with certainty. The direct proportionality of travel time to distance traveled as well as the sufficient capacity of the agent to attend all the clients in a single route are assumed as well. Note that because of the assumption of linearity between travel time and travel distance, we refer to the two indistinctly in the parameter definition.

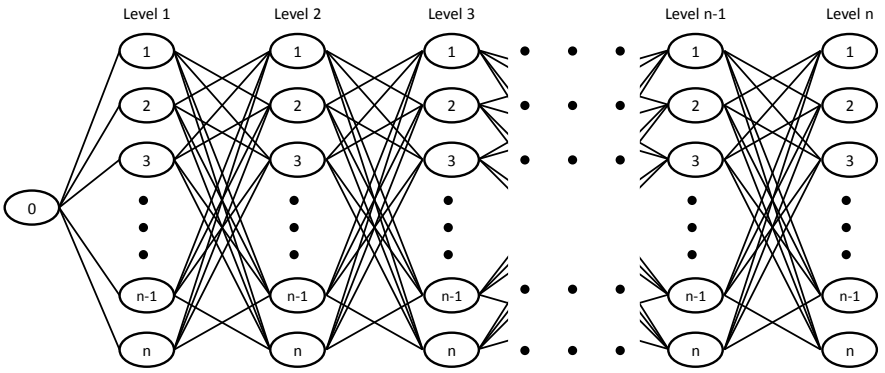


Fig. 1: Multi-level Network used to formulate the MLDP.

This problem combines two different objectives that are not calculated with the same metric (Blum et al 1994). Angel-Bello et al (2013) state that it is inconvenient to employ a formulation designed for the minimization of the distance of a route to handle an objective of minimizing the total latency along the route. We hence part from a model developed for the objective of latency, into which we incorporate the distance objective. Several formulations for the MLP exist in the literature (Gouveia and Voß 1995; Méndez-Díaz et al 2008; Picard and Queyranne 1978; Sarubbi et al 2008), the best-performing models being those of Angel-Bello et al (2013); we base our model on their reported "Model A".

Let us consider a vertex set $V = \{v_0, v_1, v_2, \ldots, v_n\}$, where v_0 represents the depot and the remaining vertices represent n clients. We define a matrix $\mathbf{T}$ that contains for each pair (v_i, v_j) a weight $t_{i,j} \geq 0$, where the diagonal element $t_{i,i}$ represents the *service time* of client i. This service time refers to the time the agent takes to fulfil the service to the client. For $i \neq j$, $t_{i,j}$ represents the *travel time* from vertex v_i to vertex v_j, this is to say $t_{i,j}$ represents the distance from client i to client j by considering the distance and travel time linearity assumption discussed before. The values $t_{i,j}$ and $t_{j,i}$ may or may not be equal — symmetry is not assumed in our formulation. We set $d_{0,0}$ for the depot to complete the matrix. From $\mathbf{T}$, a *cost matrix* $\mathbf{C}$ is computed as follows:

$$c_{i,j} = t_{i,i} + t_{i,j}. \tag{1}$$

This formulation is based on a multi-level network (see Figure 1) with $n+1$ levels, inspired by the proposals of Picard and Queyranne (1978) for a time-dependent TSP. In each level $\mathcal{L}_i, i > 0$, all vertices v_i are represented. As the solutions to MLDP are permutations of V with the depot in the first position, like the solutions of the TSP and MLP, every solution can be represented in such a multi-level network by selecting one vertex from each level. We represent

these selections with boolean decision variables $x_{i,k}$,

$$x_{i,k} = \begin{cases} 1, & \text{if } v_i \text{ is selected on } \mathcal{L}_k, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where both i and k take values from $1, \ldots, n$. Note that the depot v_0 forms $\mathcal{L}_0$, meaning no selection is made there as every feasible MLDP tour initiates at the depot. This multilevel network is helpful in representing a tour but with the exception of the returning arc traveled by the agent which, by definition of TSP, is considered in the calculations of the total traveled distance for the route. For ensuring that only one vertex is selected per level, we require

$$\forall k : \sum_{i=1}^{n} x_{i,k} = 1, \tag{3}$$

and for making sure that each vertex is selected exactly once, we write

$$\forall i : \sum_{k=1}^{n} x_{i,k} = 1. \tag{4}$$

We denote the resulting $n \times n$ matrix of decision variables by $\mathbf{X}$. Additionally, we write as an auxiliary notation

$$f_{i,j}(k) = x_{i,k} x_{j,k+1}, \tag{5}$$

meaning that $f_{i,j}(k) = 1$ if and only if v_i is selected on $\mathcal{L}_i$ and v_j is selected on $\mathcal{L}_{i+1}$, and zero otherwise[1]. With the above notation, the travel distance from the depot to the first client is defined as

$$\mathcal{T}_0 = \sum_{i=1}^{n} c_{0,i} x_{i,1}, \tag{6}$$

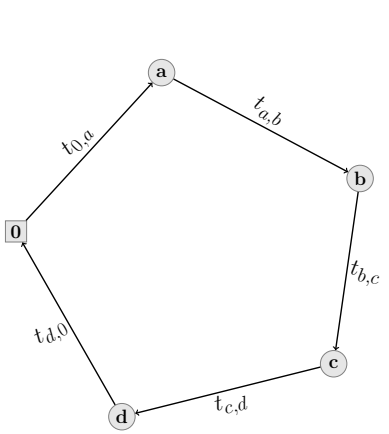
the travel distance from the last client to the depot is defined as

$$\mathcal{T}_n = \sum_{i=1}^{n} c_{i,0} x_{i,n}, \tag{7}$$

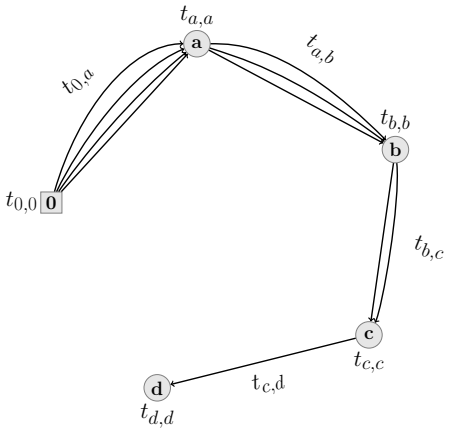
and hence the *total travel distance* from the depot back to the depot, after visiting all the clients, is

$$\mathcal{T} = \mathcal{T}_0 + \mathcal{T}_n + \sum_{k=1}^{n-1} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} f_{i,j}(k). \tag{8}$$

[1] This is a nonlinear combination, but the model can be adapted for linear solvers by defining $\mathcal{O}(n^3)$ additional decision variables $y_{i,j,k}$ to capture the values $f_{i,j}(k)$ and $\mathcal{O}(n^2)$ restrictions to require that $\forall k$, firstly, that $\forall i$ the sum of $y_{i,j,k}$ over all $i \neq j$ equals $x_{i,k}$, and secondly, that $\forall i$ the sum of $y_{i,j,k}$ over all $i \neq j$ equals $x_{i,k+1}$. As $\mathbf{X}$ is binary, it will not be necessary to explicitly require these $y_{i,j,k}$ to take on binary values, as the restrictions will enforce that. This linear version can be found in a preliminary work by Arellano-Arriaga et al (2017).



(a) Distance objective D, (Equation (8))
$D = t_{0,a} + t_{a,b} + t_{b,c} + t_{c,d} + t_{d,0}$



(b) Latency objective L, (Equation (10))
$L = \ell_0 + \ell_a + \ell_b + \ell_c + \ell_d$,
$\ell_i = t_{i,i} + t_{i,j}, \ell_0 = 0$
$L = 0 + (t_{a,a} + t_{0,a}) + ((t_{a,a} + t_{0,a}) + t_{b,b} + t_{a,b}) + \ldots + ((((t_{a,a} + t_{0,a}) + (t_{b,b} + t_{a,b})) + (t_{c,c} + t_{b,c}) + (t_{d,d} + t_{c,d}))$
$L = 4(t_{a,a} + t_{0,a}) + 3(t_{b,b} + t_{a,b}) + 2(t_{c,c} + t_{b,c}) + (t_{d,d} + t_{c,d})$

Fig. 2: This figure depicts a route in terms of each objective involved in this problem; in both sides, the depot is denoted as 0 and each clients is represented by a letter. Identical routes are depicted in both sides. Note that in Figure 2a, the distance objective (D_{route}, Equation (8)) is represented as the total sum of all travel times involved in the route since the agent leaves the depot until he returns to it. In Figure 2b, the sum of the total latency (L_{route}, Equation (10)) of the clients in the route is shown. By definition, this objective accumulates the time the agent spent in every previous client to the actual node he is in, this is denoted by the several arrows in the graph. Note that the return of the agent to the depot is not considered part of the total latency of the route.

Similarly, as each of all the n clients will have to wait for the agent to reach the first client, we write

$$\mathcal{W}_0 = n \sum_{i=1}^{n} c_{0,i} x_{i,1}, \tag{9}$$

and then express the *latency*, total waiting time, over the set of clients (as each client has to wait through the services of all the previous clients and the intermediate travel times) as

$$\mathcal{W} = \mathcal{W}_0 + \sum_{k=1}^{n-1} \sum_{i=1}^{n} \sum_{j=1}^{n} (n-k) c_{i,j} f_{i,j}(k). \tag{10}$$

Note that the depot is not waiting for a service and that the time it takes to return from the last client to the depot is not considered part of the total latency of the clients, although it is included in the calculation of the total travel distance in Equation (8). Figure 2 exemplifies the calculation of the two objectives.

In terms of the above definitions, MLDP can be stated as finding an assignment to $\mathbf{X}$ that minimizes both Equations (8) and (10) and satisfies both Equations (3) and (4). This non-linear formulation consists of $\mathcal{O}(n^2)$ decision variables and $\mathcal{O}(n)$ constraints.

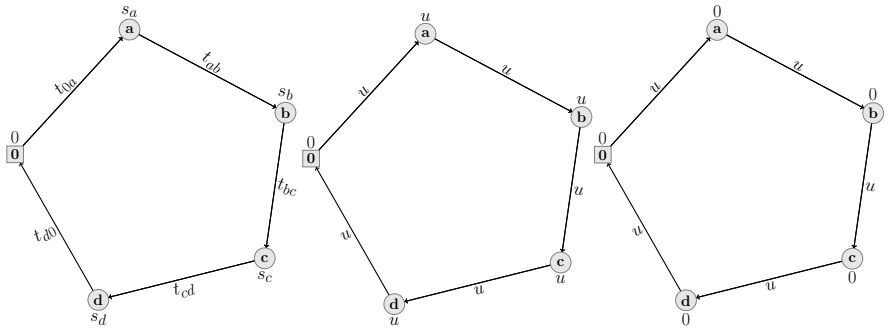
4 Computational Analysis

In this section, we proceed to study the computational complexity of the MLDP. We begin with the description of the required definitions, along with the description of several instances of this problem considered to argue that the problem is NP-hard.

The goal of an optimization problem O_σ is to find the best solution according to a set of constraints that describes the environment of the problem. Each optimization problem is associated to a decision problem D_σ which differs from O_σ in the sense that the objective is treated as a constraint. Given a bound B for the constraint representing the objective of O_σ, the answer to the decision problem is whether or not an assignment exists that satisfies both the original constraints and the new constraint with bound B. Hence the decision problem D_σ has only two outcomes: yes or no. If the decision problem D_σ is proven NP-complete, then the optimization problem O_σ is proven NP-hard (Garey and Johnson 1990; Papadimitriou 1994).

A *complexity class* is defined by several parameters such as non-determinism and restrictions on the amount of computational resources available (namely time and space) (Papadimitriou 1994). NP stands for *non-deterministic polynomial time* meaning that the class is defined by bounding the execution time polynomially under non-deterministic computation: given an input for a problem D_σ, an "oracle" can guess a correct solution in polynomial time. To prove the inclusion of any problem D_σ in the class NP it suffices to demonstrate that any given solution of O_σ can be verified as an actual solution of O_σ in polynomial time. Both of the single-objective optimization problems that are merged into MLDP are NP-hard (Afrati et al 1986; Papadimitriou 1994). In this section we demonstrate that also MLDP itself is NP-hard.

This demonstration is subject to the same assumptions as the model presented in the previous section: we consider one single agent with no capacity restrictions that visits all of the clients in a single tour. This requires efficiently reducing a known NP-complete problem O_σ to our problem (Garey and Johnson 1990). We first define the decision problem D_{MLDP} associated to MLDP, then show that MLDP belongs to NP, and finally establish that an efficient reduction from a known NP-complete problem to our problem exists.



(a) s_i, t_{ij}, arbitrary values (b) s_i, t_{ij}, constant values (c) zero service times, identical constant travel times

Fig. 3: Illustration of some particular instances of the MLDP, which were studied in the complexity proof. In Figure 3a, the general case is shown: in this case, service and travel times take all different arbitrary values. In Figure 3b, a simpler case is shown: all service and travel times are equal to a constant u. Lastly, in Figure 3c, the simplest case is shown: all service times are zero but all travel times are equal to a constant u. This constant can take an arbitrary non-negative value, as long as in the last two cases, all travel times are equal.

By definition, the MLDP consists in finding a route which visits all clients, leaves from a established depot and returns to it, while minimizing the total latency of all the clients and the total traveled distance of an uncapacitated vehicle. To prove the complexity of MLDP, several cases were revised (see Figure 3). We study a general case, which refers to a more real-life situation for the agent, this case considers arbitrary non-zero traveling and service times, (see Figure 3a). We also consider some special cases of this problem (see Figures 3b, 3c) to ensure that in simpler and more arbitrary cases the complexity holds. In this particular study we present the general case with non-zero and arbitrary values for both travel and service times (see Figure 3a). In a previous study, Nucamendi-Guillén et al (2016) discuss latency objective and propose Equation (11), where t_{ij} refers to travel time among clients i and j, as well as s_i and s_j refers to their corresponding service times.

$$\frac{1}{2}\sum_{k=1}^{n-1}(n-k)\sum_{i=0}^{n}\sum_{j=0}^{n}t_{ij}+(s_i+s_j). \tag{11}$$

Using Equation (11), we redefine non-zero travel and service times for computing the bounds used to define the decision problem D_{MLDP} associated to MLDP.

The corresponding decision problem is the following: given the costs (Equation (1)), an upper bound $\mathcal{T} \leq \Theta$ to Equation (8), and an upper bound $\mathcal{W} \leq \Omega$ to Equation (10), is there an assignment $\mathbf{X}$ that satisfies all the original constraints as well as the upper bounds set on the objectives?

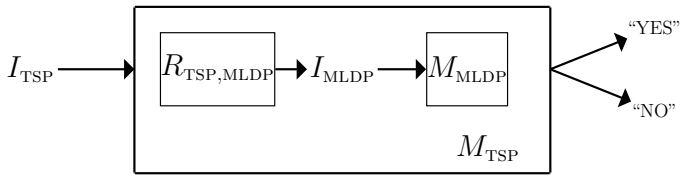


Fig. 4: Reduction diagram from TSP to MLDP (both as decision problems). The main box in the figure depicts an algorithm to solve the TSP problem that takes as input an instance of TSP and determines whether the answer is "yes" or "no" to this instance. To prove the complexity of MLDP, a reduction algorithm must transform a TSP entry into a MLDP one. This MLDP input is then solved and receives a yes or no answer. As a whole, the process takes a TSP instance, reduces it into an MLDP instance, solves the reduced instance with an algorithm for MLDP, and responds correctly "yes" or "no" to the original instance.

We first establish that the feasibility of a given $\mathbf{X}$ can be verified in polynomial-time: each $\mathbf{X}$ captures a permutation that starts at the depot and the permutation can be efficiently recovered from the assignment matrix $\mathbf{X}$ as a visit sequence $v^{(0)}, v^{(1)}, \ldots, v^{(n)}$ where $v^{(0)}$ is always the depot and the visits from v^1 to $v^{(n)}$ correspond to the clients. We need three accumulator variables, one to measure time along the route, one to count the total travel time, and the third to count the total latency. We also need an array of n binary variables, one per each client, to verify that each one is properly visited. We proceed in the visiting order, denoting the source by i and the destination by j adding the value of $t_{i,j}$ to both the time accumulator and the travel-time accumulator, then adding the service time $t_{i,i}$ to the time accumulator, and then the current value of the time accumulator to the latency accumulator, making the client j as visited. If at the end of the permutation, all n binary variables are one, the travel-time accumulator respects its upper bound Θ, and the latency accumulator respects its upper bound Ω, the solution is feasible. As the verification is a polynomial procedure, $D_{\text{MLDP}} \in \mathsf{NP}$.

To establish NP-completeness, we reduce the traveling salesman problem (TSP) in its decision version to D_{MLDP}, as illustrated in Figure 4. In the decision version of TSP, the input is a cost matrix $\mathbf{C}$ for the travel costs between n clients (with the diagonal elements being zero) together with an upper bound to the total cost of the tour, $\mathcal{C}$, and the question is whether a permutation over the set of clients exists that produces a tour with the sum of costs of the segments not exceeding $\mathcal{C}$.

In order to transform the input for the decision version of the TSP into an input for D_{MLDP}, we will use the elements of $\mathbf{C}$ as $\mathcal{T}$, set $\Theta = \mathcal{C}$, and only need to compute efficiently an adequate value for the latency bound Ω. We do this in terms of the worst-case costs, computing from the cost matrix $\mathbf{C}$ the largest cost per each row (that is, as if the agent at each client chose the most

expensive segment to continue the tour) in $\mathcal{O}(n^2)$ time, and then sort these worst-case segments from largest to smallest in $\mathcal{O}(n \log n)$ time to construct an upper bound to the worst-case latency (the service times in TSP are the diagonal elements of the cost matrix that are all zero): the worst-case total latency is the sum of the cumulative sums of the sorted list of costs and we use this as Ω. Hence, D_{MLDP} responds "yes" to the transformed input if and only if the decision version of TSP would respond "yes" for the original input.

By reducing TSP to MLDP, we show that MLDP is at least as difficult as TSP. Thus, D_{MLDP} is NP-complete and therefore MLDP is NP-hard.

5 Solution methods

In the previous section we establish that MLDP is an NP-hard problem and therefore exact methods are only expected to solve efficiently very small instances, making heuristic approaches a necessity. In this section, we discuss the design and implementation of the procedures we implement to obtain the Pareto fronts for a set of instances. We describe our implementations for obtaining the exact MLDP Pareto fronts of these instances, as well as the description of two heuristic approaches we construct to obtain the approximations of these Pareto fronts.

5.1 Exact method

MLDP, as stated before, is a problem which deals with the minimization of two objective functions. A well-known classic technique for *exact* solution of multiple objective problems is the ϵ-constraint method (Ehrgott 2005). Nonetheless the ϵ-constraint has some aspects to improve. Mavrotas (2009) developed a technique called Augmecon which guarantees Pareto optimality in the pay-off table of the obtained solution, as well as in the generation process, parting from the idea of the classic ϵ-constraint to constraint one objective to the rest of them. Augmecon is a capable algorithm that works for multiobjective problems with two or more objective functions.

In particular, Mavrotas and Florios (2013) developed an improvement of Augmecon called *Augmecon2*. This procedure is an update of a previous work by Mavrotas (2009), and while it keeps the same idea of the ϵ-constraint, by solving the mono-objective version constrained to the rest of the objectives, it exploits the information from the slack variables in every iteration leading to a significant reduction on computation time as well as the avoidance of many redundant iterations.

In our particular implementation of this algorithm, we keep the distance objective of Equation (8) as the main objective and solve the problem subject to the original constraints along with the latency objective of Equation (10); in our case, this produces a single-objective problem which minimizes the distance of the route constrained to its total latency. The order of the objectives could

be reversed, but according to Angel-Bello et al (2013); Blum et al (1994), as well as the complexity analysis of the latency objective, the alternative would be more time consuming. As a normal ϵ-constraint, Augmecon2 takes a step to make a grid and find the Pareto points. In our case, we define this step by $\delta = 1/2n$, where n denotes the size of the instance. Note that this method is able to find optimal solutions, optimal in terms of efficiency, but it may not find non-supported points in the Pareto front, in other words, all the points in the true Pareto front.

In order to use Augmecon2, which is a linear solver, we linealize our formulation with additional variables and constraints as mentioned in Section 3. As both of the single-objective problems that MLDP may reduce to in this manner are also NP-complete (Afrati et al 1986; Papadimitriou 1994), the asymptotic complexity of computing a Pareto front is exponential in the size of the instance, which in this case is measured in terms of the number of clients. In case the latency objective of Equation (10) is the one being fixed and the problem is solved subject to the original constraints along with the distance objective of Equation (8), an NP-hard problem is expected as well. Therefore, a metaheuristic approach is justified and desirable; next sections describe two memetic algorithms to solve this biobjective problem. In Section 6, we present experimental results for all of the proposed solution strategies.

5.2 Metaheuristic method I. Strategic Memetic Search Algorithm

As an efficient but heuristic alternative, we design and implement a *memetic algorithm* to approximate the Pareto fronts for larger instances.

Algorithm 1 Basic memetic algorithm. *Based on the algorithm proposed by Deb et al (2000)*

```
t := 0
Generate an initial population P_0 of size n_p
Generate an offsprings population, Q_0 of size n_p from P_0
R_0 := P_0 ⋃ Q_0
repeat
    Divide R_t in non-dominating fronts F_i
    P_{t+1} := 0 and i := i + 1
    while |P_{t+1}|+|F_i| < n_p do
        i:=i+1
        P_{t+1} := P_{t+1} ⋃ F_i
        Mutation
        i:=i+1
    end while
    Sort each front by crowding distance and preserve the best n_p
    Generate an offspring population Q_{t+1} of size n_p from P_{t+1}
    R_{t+1} := P_{t+1} ⋃ Q_{t+1}
    t:=t+1
until reaching a stopping condition
```

We take as framework the NSGA-II metaheuristic (see Algorithm 1) by Deb et al (2000), which is an elitist multi-objective evolutionary algorithm operating on a population of solutions. In the following subsections each step and the adaptations made in the design of this strategic memetic search algorithm, hereinafter will be called as SMSA, to solve the MLDP are explained in detail.

5.2.1 Generation of initial population P_0

In our SMSA implementation, the generation of the initial population of size n_p (step two) consists of 60% of random solutions and 40% of solutions obtained using the cheapest insertion heuristic (Rosenkrantz et al 1974). From the latter, half of them are obtained using a distance metric and half using a latency metric.

5.2.2 Local search

Our algorithm is a memetic algorithm, that is, our mutation process is based on a local search procedure. We define two neighborhoods by two different 2-opt (Croes 1958) movements, each of which removes two arcs from the actual solution and reconnects it in a different way to obtain a new solution; the first neighborhood (defined as $i = 0$ in Algorithm 2) is guided by the distance objective and the other is guided by the latency objective (defined as $i = 1$ in Algorithm 2), meaning we include a distance-improvement neighborhood and a latency-improvement neighborhood.

The main part of the mutation process is the ability of the SMSA to move from one neighborhood to another in order to add quality to the approximated front. Despite the movement chosen is a basic 2-opt, the stop criterion is set by a counter which sets to zero once the local search cannot find a better solution and it switches to the other non-explored neighborhood (see Algorithm 2).

Algorithm 2 Strategic memetic search mutation process algorithm

```
1: u := 1
2: Select i ∈ {0, 1} randomly and select neighborhood N_i accordingly
3: while u ≠ 0 do
4:     Explore current neighborhood N_i
5:     if Exploration yielded no improvement then
6:         u := 0
7:     else
8:         u := u + 1
9:     end if
10: end while
```

5.2.3 Generation offsprings population Q_0

In our implementation, the selection of each parent is made through a binary tournament. After selecting both parents, we apply a combination method

based on the OX crossover introduced by Davis (1985), described in detail by Oliver et al (1987). Prins (2004) modified this method and presented it in such a way that the combination of the parents brings up to eight different offsprings. We apply the modified OX crossover previously described in Prins (2004), up to eight different offsprings; we apply this modified crossover repeatedly until a population of the desired size is obtained.

5.2.4 Division of R_0 in non-dominating fronts

In our implementation of Algorithm 1, we consider the ranking process described by Deb et al (2000). The core idea of this ranking is based on the knowledge that to identify a dominating solution i, it is iteratively compared to the other solutions to identify which ones it dominates, recording for each solution the set of solutions that dominate it as well as the set of solutions that it dominates, thus making it faster to rank the solutions.

These two sets consequently allows us to rank each solution in a faster way. This procedure corresponds to step six of Algorithm 1, and has an asymptotic complexity $\mathcal{O}(Mn^3)$, where M refers to the number of objectives involved and n refers to the size of the population (Deb et al 2000).

5.3 Metaheuristic method II. Evolutionary algorithm with Intelligent Local Search

Due to the complexity of MLDP, we also propose an evolutionary algorithm to approximate the true Pareto fronts and in this section we describe it.

Algorithm 3 Evolutionary algorithm with Intelligent Local Search algorithm

```
g:=0
Generate initial population P_g of size n_p := n^2
repeat
    Evaluate fitness of P_g
    Group P_g into non-dominating fronts
    Place the best n_e := ⌈n_p/2⌉ solutions into the set Elite (P_e)
    Define n_c := ⌈n_e/2⌉ of couples of solutions labelled as parents
    for each couple in the Elite set do
        Apply crossover
    end for
    g:=g+1
    P_g := P_g ⋃ P_e
    for each solution in P_g do
        NumNeig := 0
        while NumNeig ≤ 7 do
            Apply the intelligent neigborhood selection until NEvalWS consecutive iterations produce no local improvement
            NumNeig := NumNeig + 1
        end while
    end for
until reaching a stopping condition
```

We name this algorithm as Evolutionary algorithm with Intelligent Local Search and hereinafter will be refer to as EiLS. It is based on the Pareto ranking scheme with crowding distance proposed by Deb et al (2000) in their NSGA-II procedure.

EiLS takes the idea of this classic Pareto ranking scheme used in NSGA-II, but with specific crossover and mutation operators. On one hand, the computationally heavy crossover (combining routes is non-trivial) is substituted by a random shake procedure (described in detail in Section 5.3.4), and on the other hand, mutation is replaced by an intelligent local search over various neighborhoods (described in Section 5.3.5). A general overview of EiLS is given in Algorithm 3.

5.3.1 Solution representation and initial solutions generation

Solutions are represented as a permutation of the nodes, this is to say, a vector $P = (P_1, P_2, \ldots, P_n)$ where P_i represents the ith visit along the route. All initial solutions are created generating random permutations of the n nodes. The depot is not permuted but instead placed as thee first node in all orderings, hence producing only feasible solutions. This corresponds to step two in Algorithm 3.

5.3.2 Fitness assignment

We use the same fitness assignment that Deb et al (2000) propose for NSGA-II. The main idea of this ranking is described in detail in Section 5.2.4 for SMSA. This procedure corresponds to step five of Algorithm 3, and has an asymptotic complexity $\mathcal{O}(Mn^3)$, where M refers to the number of objectives involved and n refers to the size of the population (Deb et al 2000).

5.3.3 Selection operator

In step six of Algorithm 3, we define a n_e-member elite set P_e containing the individuals with the best fitness values. In step seven, we select n_c couples of parents among the elite set using binary tournament selection. We set $n_p = n^2$, $n_e = \lceil n_p/2 \rceil$, and $n_c = \lceil n_e/2 \rceil$.

5.3.4 Crossover operator

Each couple produces two new individuals with the following procedure. First, one of the parents is randomly selected (p_1) copied to the first offspring (o_1). Once copied, we apply the following perturbation: the positions of two non-overlapping blocks of size m are exchanged in the route; the value of m is an integer selected uniformly at random in $[2, 0.3n]$.

Table 1 shows an example for obtaining an offspring from a 10-client solution with $m = 3$. Notice the first two rows denote p_1 as parent one and o_1 as the first offspring generated. At this point of the crossover, o_1 is simply

p_1 copied. The following two rows depict the perturbation made from p_1 to o_1. This procedure is then repeated with the second parent; note that the two individuals are processed independently. This corresponds to step nine of Algorithm 3.

Table 1: Crossover operator example in a 10-client instance. D and P_i depicts the position of the depot and of each vertex in the solution.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
p_1	0	2	3	4	1	5	6	8	10	9	7
o_1	0	2	3	4	1	5	6	8	10	9	7
p_1	0	**2**	**3**	**4**	1	5	6	**8**	**10**	**9**	7
o_1	0	**8**	**10**	**9**	1	5	6	**2**	**3**	**4**	7

The elite population and the new offsprings are combined, as described in step twelve of Algorithm 3.

5.3.5 Mutation operator

Instead of a traditional mutation, each individual of the current population undergoes an intelligent local search. For diverse exploration of neighboring solutions in terms of both of the objectives, we locally optimize a compromise function with random weights as proposed by Molina et al (2007). The purpose of a compromise function is to aggregate the weights of the original objectives to diversify the resulting Pareto front by presenting a balance among the different objectives. The construction of a compromise function is described in Algorithm 4.

Algorithm 4 Construction of an objective for local search

1: Pick $t \in \{\text{True, False}\}$ uniformly at random
2: **if** $t = \text{True}$ **then**
3: Select uniformly at random one of the objective functions as such
4: **else**
5: Generate a random linear combination of the objective functions
6: **end if**

Once the objective function to optimize is selected, several neighborhoods $\mathcal{N}_k$ for $k = 1 \dots k_{\max}$ are explored; in total we use twelve neighborhoods, which are described in Section 5.3.6. As proposed by Molina et al (–), the order in which the $\mathcal{N}_k$ neighborhoods are applied depends on their performance. The performance is measured in terms of the quantity of the improvement to the local solution using the objective constructed by Algorithm 4. Each $\mathcal{N}_k$ neighborhood is explored until a maximum number of consecutive iterations with no improvement is reached (we use NEvalWS $= 10n$). After this first

iteration, the performance of the neighborhoods Success_k is updated according to the improvements obtained (see Algorithm 5 for its calculation).

Algorithm 5 Success indicator per explored neighborhood

1: LocalSuccess_k := the number of times the current solution was improved using $\mathcal{N}_k$ in any local exploration
2: Calls_k := the number of times that $\mathcal{N}_k$ has been called so far
3: $\text{Success}_k := \frac{\text{LocalSuccess}_k}{\text{Calls}_k}$

Algorithm 6 Intelligent Neighborhood Selection in EiLS algorithm.

1: **Update:** Success_k values
2: NSelected := 0
3: **while** NSelected = 0 **do**
4: a random number $neig \in \{1, k_{\max}\}$
5: a random number $prob \in [0, 1]$
6: **if** $prob \leq \text{Success}_{neig}$ **then**
7: NSelected := $neig$
8: **end if**
9: **end while**

From this moment on, neighborhoods are selected in decresing order starting by the highest-success performing one. With this design, each $\mathcal{N}_k$ iterates until it is unable to improve its current solution for NEvalWS iterations instead of iterating for a fixed number of evaluations despite the size of the instance. Algorithm 6 shows how each neighborhood has probability to be chosen and a local search is performed until a maximum number of consecutive iterations with no improvement is reached (we use $10n$). After this, the performance of the neighborhoods is updated with respect to the improvements obtained. This is repeated sequentially, using the best solution of the preceding round as the starting solution of the following round, seven times for each individual of the current population. The final scheme of the Mutation Operator for the proposed EiLS algorithm is shown in Algorithm 7.

Algorithm 7 Mutation Operator for the EiLS method.

1: Choose a objetive function to optimize using Algorithm 4
2: NumNeig := 0
3: **while** NumNeig ≤ 7 **do**
4: Choose a Neighborhood structure $\mathcal{N}_k$ using Algorithm 6
5: Apply neighborhood exploration using $\mathcal{N}_k$ till NEvalWS consecutive iterations produce no local improvement
6: NumNeig := NumNeig + 1
7: **end while**

This mutation operator corresponds to step fourteen in Algorithm 3.

5.3.6 Neighborhood structures

In this section we describe the twelve different Neighborhood structures $\mathcal{N}_k$ we propose for the Mutation Operator of our EiLS algorithm, previously described in Section 5.3.5.

- $\mathcal{N}_1$: Two clients are randomly chosen and their positions are exchanged. This neighborhood mutates a solution s_1 by exchanging the positions of two nodes, selected uniformly at random.
- $\mathcal{N}_2$: Three clients chosen at random and their positions are exchanged by moving the earliest one to the position of the middle client, the middle client to the position of the last client, and the last client to the position of the first client. Table 2 shows an example of a 15-client solution with a $\mathcal{N}_2$ move.

Table 2: $\mathcal{N}_2$ neighborhood example in a 15-client instance.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
s_1	0	**11**	8	12	15	7	**6**	5	13	1	2	14	3	4	**10**	9
m_{s_1}	0	**10**	8	12	15	7	**11**	5	13	1	2	14	3	4	**6**	9

- $\mathcal{N}_3$: One client is selected at random and its position is exchanged with another client that is selected in such a way that the second client must be at most closeness $= 0.5n$ steps away from the first client. Table 3 shows an example of a 10-client solution with a $\mathcal{N}_3$ move.

Table 3: $\mathcal{N}_3$ neighborhood example in a 10-client instance. In this example, the selected_node $= 3$ and the closeness of the clients is defined by closeness $= 0.5 \cdot 3 = 1.5$.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
s_1	0	2	3	**4**	*1*	5	6	8	10	9	7
m_{s_1}	0	2	3	*1*	**4**	5	6	8	10	9	7

- $\mathcal{N}_4$: A client is randomly selected, and then two other clients are selected with the restraint that their positions in the permutation have to be closer to the first selected client by a parameter closeness $= 0.5n$. The position of these three clients are exchanged using the exchange scheme in $\mathcal{N}_2$. Table 4 shows an example of a 15-client solution with a $\mathcal{N}_4$ move.
- $\mathcal{N}_5$: A client is randomly selected and its position is exchanged with the previous client in the permutation.
- $\mathcal{N}_6$: Same structure as $\mathcal{N}_3$, but with a different value of the parameter closeness, namely closeness $= 0.8n$.

Table 4: $\mathcal{N}_4$ neighborhood example in a 15-client instance. selected_node = 8 and the closeness = $0.5 \cdot 8 = 4$.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
s_1	0	11	8	12	15	7	6	5	**13**	1	**2**	14	**3**	4	10	9
m_{s_1}	0	11	8	12	15	7	6	5	**3**	1	**13**	14	**2**	4	10	9

– $\mathcal{N}_7$: Same structure as $\mathcal{N}_4$, but with a different value of the parameter closeness, namely closeness = $0.8n$.
– $\mathcal{N}_8$: The client with the highest cost is selected, for clarity this node will be referred to as V_s, where the cost of the $i-th$ client in the permutation is $(n-i) \cdot d_{i,i-1}$, being $d_{i,i-1}$ the distance to the previous client in the permutation. Once V_s is selected, another client is randomly selected and their position is exchanged. Table 5 illustrates a 10-client example of a $\mathcal{N}_8$ move, row s_i denotes the current solution to modify according to $\mathcal{N}_8$. Row $n-i$ and row $d_{i,i-1}$, together denotes the cost per arc and row m_{s_i} shows the interchange between the highest cost client and a random one.

Table 5: $\mathcal{N}_8$ neighborhood example in a 10-client instance where $V_s = 1$.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
s_1	0	2	3	4	1	5	6	8	10	9	7
$n-i$	10	9	8	7	6	5	4	3	10	2	1
$d_{i,i-1}$	15	8	12	4	7	9	5	4	10	9	
s_1	0	**2**	3	4	1	5	*6*	8	10	9	7
m_{s_1}	0	*6*	3	4	1	5	**2**	8	10	9	7

– $\mathcal{N}_9$: A client is randomly selected in the last half of the positions in the permutation, and its position is exchanged with a client randomly selected. Table 6 shows a 10-client example of a $\mathcal{N}_9$ move.

Table 6: $\mathcal{N}_9$ neighborhood example in a 10-client instance.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
s_1	0	2	3	4	*1*	5	6	8	10	**9**	7
m_{s_1}	0	2	3	4	**9**	5	6	8	10	*1*	7

– $\mathcal{N}_{10}$: A client is randomly selected in the last half of the positions in the permutation, let p be its position. Another client is selected in the last $n-p$ positions, this is, a posterior client is randomly selected, and their positions are exchanged. Table 7 shows a 10-client example of a $\mathcal{N}_{10}$ move.

Table 7: $\mathcal{N}_{10}$ neighborhood example in a 10-client instance. In this example $p = 7$ and $n - p = 3$.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
s_1	0	2	3	4	1	5	6	**8**	10	9	7
m_{s_1}	0	2	3	4	1	5	6	7	10	9	**8**

– $\mathcal{N}_{11}$: Two blocks of clients of size m with empty intersection are randomly selected, and their position is exchange in the permutation. The size of the blocks, m, is randomly selected in $[2, 0.3n]$. Table 8 shows a 15-client example of a $\mathcal{N}_{11}$ move.

Table 8: $\mathcal{N}_1$ neighborhood example in a 15-client instance, where $m = 3$.

	D	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
s_1	0	11	**8**	**12**	**15**	7	6	5	13	1	***2***	***14***	***3***	4	10	9
m_{s_1}	0	11	***2***	***14***	***3***	7	6	5	13	1	**8**	**12**	**15**	4	10	9

– $\mathcal{N}_{12}$: Two positions, p and p^*, are randomly selected. The client in the position p is moved to position p^*.

Note that neighborhoods $\mathcal{N}_6$ and $\mathcal{N}_7$ are the same as $\mathcal{N}_3$ and $\mathcal{N}_4$, but using the parameter closeness $= 0.8n$ instead of closeness $= 0.5n$. Experimental tuning of this parameter showed no specific value to be clearly superior throughout all the problems. So, instead of fixing it, we duplicate the neighborhood structure with these two different values of the parameter and let the intelligent selection system choose the most suitable one, based on its performance.

6 Computational experimentation

We study the performance of the three methods: our implementation of the Augmecon2, our SMSA implementation, as well as the implementation of the EiLS algorithm.

We experiment on instances inspired by Angel-Bello et al (2013), with adjustments to create client locations and service times more similar to real-world settings. We first create c cities with uniform random coordinates in $(X, Y) \in \mathbb{R}$, $X, Y \sim \text{Unif}(0, 1)$, and place n clients, selecting for each client a city, randomly[2] and then computing the coordinates of the client as the coordinate of its city with normally distributed offsets $\Delta X, \Delta Y \sim \text{Norm}(\mu, \sigma)$, where μ and σ are parameters. We compute the travel distances $t_{i,j}$ for $i \neq j$ as

[2] Each city is selected with a probability proportional to the current number of clients attached to it in order to create a population pattern similar to real-world cities.

Euclidean distances between the clients with an exponential delay $\delta \sim \text{Exp}(\lambda_d)$ and the service times $t_{i,i} \sim \text{Exp}(\lambda_t)$ (forcing the service time at the depot to zero, while locating the depot as if it were another client). We experiment with $n \in \{10, 16, 20, 32, 40, 64, 80, 128, 160\}$ clients with $c = \lfloor \log_s n \rfloor$ cities, $\mu = 1/c$, $\sigma = 1/\sqrt{n}$, $\lambda_d = 1/5\sqrt{n}$, and $\lambda_s = 1/\sqrt{n}$, creating 35 instances of each size. Note that by multiplying all the service times by a constant, the relative balance between service and travel times can be adjusted without regenerating the instance.

Augmecon2 was solved using the MILP solver ILOG CPLEX C++ Concert Technology. Both heuristic procedures are coded in C++ and every instance was solved in a Xenon® Intel® CPU E3-1245 v3 @ 3.40GHz, with 16 GB of RAM with each of these methods. All results are in terms of average values for each size, over sets of 35 instances. For comparison purposes a six and a half hour time limit, per instance, was fixed for Augmecon2. The same instance sets are solved by all three solution approaches. CPU Times Elapsed can be seen in Table 9.

To compare the performance of the two proposed metaheuristics against the exact Pareto fronts obtained with the Augmecon2, we consider five metrics: the number of points on the front, the size of the space covered (SCC) (Zitzler and Thiele 1999), the k-distance (labeled as k-D in the comparative table) (Zitzler et al 2002), the distance between the exact Pareto front and the approximate Pareto front (M_1^*) (Zitzler 1999) and the coverage of the frontiers (Zitzler and Thiele 1999). The averages values per instance of this metrics can be found in Table 10, where the superiority of the EiLS procedure is shown. This algorithm is able to use its intelligent neighborhood selection to improve the density of the approximated Pareto fronts while keeping good quality in the solutions. The coverage of the fronts is presented in Table 11, where the superiority of the EiLS algorithm is evident. Note that because Augmecon2 is able to find optimal solutions but it may not find non-supported points in the Pareto front, leading to less points and less variability, we only report EiLS and SMSA coverages. The superiority of the EiLS is able to obtain more points in the fronts, as well as a better spread of them in a shorter time than the two other algorithms.

Table 9: Average CPU Times obtained by the three proposed solution methods. These times are shown in averages per each set of instances and measured in seconds.

Instance	Size	Augmecon2	SMSA	EiLS
MLDP_10	10	41.038	2.925	0.230
MLDP_16	16	179.722	6.443	1.778
MLDP_20	20	800.531	7.948	6.880
MLDP_32	32	1537.099	101.345	75.754
MLDP_40	40	22088.572	327.655	223.579

7 Conclusions

In this study, we propose a bi-objective approach which combines two well-known combinatorial problems with the goal to integrate the client as an active part in the decision making process of a routing problem. In this problem, which we call the Minimum Latency-Distance Problem (MLDP), we propose to minimize the travel distance and the total latency of an assumed infinite-capacity route to improve two important aspects of the problem: the economic decisions of the company and the degree the company's clients service, by the minimization of the total waiting time of all the clients in the route. By considering not only the economic aspects of a classic routing problem but also incorporating the social aspect in the decision making process, we aim to improve the general benefit of the company by having a quality service and therefore, a better position in the market.

We present a study of the computational complexity of the MLDP in order to prove this bi-objective problem is an NP-hard problem, and justify the employment of approximated solution methods. We propose a non-linear formulation for this problem, which, by the addition of several linearization variables, can be solved to obtain the exact Pareto fronts. We provide a set of new instances which take into account several adjustments to recreate real-world settings and were able to find the exact Pareto front of all instance sets containing from ten to forty clients. As approximated solution methods are desirable, we propose a comparison between a metaheuristic based on the classic multi-objective NSGA-II algorithm which we call Strategic Memetic Search Algorithm (SMSA) and an intelligent neighborhood selection method which we call Evolutionary algorithm with Intelligent Local Search (EiLS). We found that the EiLS method, guided by an intelligent selection of a more appropriate neighborhood in the local search procedure, is superior than the

Table 10: Comparison among the metrics obtained per algorithm. These values are in average terms per set of instances.

	Augmecon2				SMSA				EiLS			
n	# Pts	SCC	k-D	M_1^*	# Pts	SCC	k-D	M_1^*	# Pts	SCC	k-D	M_1^*
10	4.622	0.912	1.414	0	8.446	0.938	0.213	0.295	35.218	0.939	<0.001	0.016
16	5.727	0.876	0.098	0	9.743	0.892	0.326	0.146	91.422	0.975	<0.001	0.023
20	7.249	0.904	0.057	0	9.893	0.891	0.166	0.292	138.261	0.915	<0.001	0.013
32	9.326	0.978	0.193	0	11.327	0.879	0.085	0.299	201.319	0.881	<0.001	0.055
40	9.785	0.962	0.058	0	11.994	0.816	0.068	0.107	592.368	0.819	<0.001	0.068

Table 11: Average coverage of the approximated fronts obtained by the SMSA and EiLS procedures.

Instance	Size	c(SMSA,EiLS)	c(EiLS,SMSA)
MLDP_10	10	<0.001	0.796
MLDP_16	16	0.062	0.832
MLDP_20	20	0.183	0.857
MLDP_32	32	0.195	0.847
MLDP_40	40	0.237	0.914

SMSA method, in terms of density of the fontier, hypervolume percentage, computational time required and other quality metrics.

In future work, it is desirable to consider real-world conditions to develop a more realistic mathematical formulation, the easiest modification to point out is the introduction of several vehicles to visit the set of clients as well as the introduction of capacity in each one of these vehicles. It would be advisable to consider the stochastic version of this problem as well as the redefinition of the assumptions made for the linearity between traveling time and traveled distance.

Compliance with Ethical Standards

Conflict of Interest: The authors declare that they have no conflict of interest.

Acknowledgements

The first author would like to thank CONACyT, the Mexican National Council for Science and Technology, which supports her studies at UANL under the scholarship number 446316.

References

Afrati F, Cosmadakis S, Papadimitriou CH, Papageorgiou G, Papakostantinou N (1986) The complexity of the travelling repairman problem. RAIRO—Theoretical Informatics and Applications 20(1):79–87

Angel-Bello F, Álvarez A, García I (2013) Two improved formulations for the minimum latency problem. Applied Mathematical Modelling 37(4):2257 – 2266

Angel-Bello F, Martinez-Salazar I, Alvarez A (2013) Minimizing waiting times in a route design problem with multiple use of a single vehicle. In: INOC—International Network Optimization Conference, Elsevier, Tenerife, Spain, pp 269–273

Applegate DL, Bixby RE, Chvatal V, Cook WJ (2007) The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton University Press, Princeton, NJ, USA

Arellano-Arriaga NA, Alvarez-Socarrás AM, Martínez-Salazar IA (2017) A sustainable bi-objective approach for the minimum latency problem. In: Smart Cities—Second International Conference, Smart-CT 2017, June 14-16, 2017, Springer, Málaga, Spain, pp 11–19

Bair W, Cavanaugh JR, Movshon JA (2003) Time course and time-distance relationships for surround suppression in macaque V1 neurons. Journal of Neuroscience 23(20):7690–7701

Bektas T (2006) The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34(3):209 – 219

Blum A, Chalasani P, Coppersmith D, Pulleyblank B, Raghavan P, Sudan M (1994) The minimum latency problem. In: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, ACM, New York, NY, USA, pp 163–171

Borah SJ, Dhurandher SK, Woungang I, Kumar V, Barolli L (2017) A multi-objectives based technique for optimized routing in opportunistic networks. Journal of Ambient Intelligence and Humanized Computing pp 1–12

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., IEEE, Cambridge, MA, USA, USA, pp 36–45

Chvátal V, Cook W, Dantzig GB, Fulkerson DR, Johnson SM (2010) Solution of a large-scale traveling-salesman problem. In: Júnger M, Liebling TM, Naddef D, Nemhauser GL, Pulleyblank WR, Reinelt G, Rinaldi G, Wolsey LA (eds) 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, Springer, Berlin, Heidelberg, pp 7–28

Croes GA (1958) A method for solving traveling-salesman problems. Operations Research 6(6):791–812

Crowcroft J, Segal M, Levin L (2015) Improved structures for data collection in static and mobile wireless sensor networks. Journal of Heuristics 21(2):233–256

Darcis G, Bouchat S, Kula A, Van Driessche B, Delacourt N, Vanhulle C, Avettand-Fenoel V, De Wit S, Rohr O, Rouzioux C, et al (2017) Reactivation capacity by latency-reversing agents ex vivo correlates with the size of the HIV-1 reservoir. Aids 31(2):181–189

Deb K, Pratap A, Agarwal S, Meyarivan T (2000) A fast elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2):182–197

Depuy GW, Savelsbergh MW, Ammons JC, McGinnis LF (2001) An integer programming heuristic for component allocation in printed circuit card assembly systems. Journal of Heuristics 7(4):351–369

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation 1(1):53–66

Ehrgott M (2005) Multicriteria Optimization. Springer, Secaucus, NJ, USA

Ferrer JM, Ortuño MT, Tirado G (2016) A grasp metaheuristic for humanitarian aid distribution. Journal of Heuristics 22(1):55–87

Fischer W, Wictorin K, Björklund A, Williams L, Varon S, Gage F, et al (1987) Amelioration of cholinergic neuron atrophy and spatial memory impairment in aged rats by nerve growth factor. Nature 329(6134):65–68

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

Garey MR, Johnson DS (1990) Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA

Gary L Clark PFK, Rink DR (1992) Consumer complaints: Advice on how companies should respond based on an empirical study. Journal of Consumer Marketing 9(3):5–14

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

Grout V, McGinn J, Davies J (2007) Real-time optimisation of access control lists for efficient internet packet filtering. Journal of Heuristics 13(5):435–454

Gummadi KP, Saroiu S, Gribble SD (2002) King: Estimating latency between arbitrary internet end hosts. In: Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment, ACM, New York, NY, USA, pp 5–18

Gutin G, Punnen A (2006) The Traveling Salesman Problem and Its Variations. Combinatorial Optimization, Springer, USA

He D, Mujica G, Portilla J, Riesgo T (2015) Modelling and planning reliable wireless sensor networks based on multi-objective optimization genetic algorithm with changeable length. Journal of Heuristics 21(2):257–300

Hoffman KL, Padberg M, Rinaldi G (2013) Traveling salesman problem. In: Gass SI, Fu MC (eds) Encyclopedia of Operations Research and Management Science, Springer, Boston, MA, pp 1573–1578

Joo Ghee L, Junaid Q, Chun Tung C, Archan M (2009) Minimum latency broadcasting in multiradio, multichannel, multirate wireless meshes. IEEE Transactions on Mobile Computing 8(11):1510–1523

Kao YH, Krishnamachari B, Ra MR, Bai F (2017) Hermes: Latency optimal task assignment for resource-constrained mobile computing. IEEE Transactions on Mobile Computing 16(11):3056–3069

Kim S, Na JH (2017) Improving latency using codes in mission-critical communication. In: Advanced Communication Technology, 2017 19th International Conference on, IEEE, USA, pp 730–734

Kovács G, Spens K (2009) Identifying challenges in humanitarian logistics. International Journal of Physical Distribution & Logistics Management 39(6):506–528

Kovács G, Spens KM (2007) Humanitarian logistics in disaster relief operations. International Journal of Physical Distribution & Logistics Management 37(2):99–114

Laporte G (1992) The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59(2):231–247

Lichtsteiner P, Posch C, Delbruck T (2008) A 128x128 120 db 15 μs latency asynchronous temporal contrast vision sensor. IEEE journal of solid-state circuits 43(2):566–576

Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21(2):498–516

Littner M, Kushida C, Wise M, Davila D, Morgenthaler T, Lee-Chiong T, Hirshkowitz M, Daniel L, Bailey D, Berry R, Kapen S, Kramer M (2005) Practice parameters for clinical use of the multiple sleep latency test and the maintenance of wakefulness test. Sleep Standards of Practice Committee of the American Academy of Sleep Medicine 28(1):113–121

Lu G, Krishnamachari B, Raghavendra CS (2007) An adaptive energy-efficient and low-latency mac for tree-based data gathering in sensor networks: Research articles. Wireless Communications and Mobile Computing — Advances in Resource-Constrained Device Networking 7(7):863–875

Lucena A (1990) Time-dependent traveling salesman problem–the deliveryman case. Networks 20(6):753–763

Madhyastha HV, Anderson T, Krishnamurthy A, Spring N, Venkataramani A (2006) A structural approach to latency prediction. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, ACM, New York, NY, USA, pp 99–104

Mavrotas G (2009) Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. Applied Mathematics and Computation 213(2):455–465

Mavrotas G, Florios K (2013) An improved version of the augmented -constraint method (augmecon2) for finding the exact pareto set in multi-objective integer programming problems. Applied Mathematics and Computation 219(18):9652 – 9669

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

Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. J ACM 7(4):326–329

Molina J, López-Sánchez A, Hernández-Díaz A, Martínez-Salazar I (–) A multi-start algorithm with intelligent neighborhood selection for solving multi-objective humanitatian vehicle routing problems. Journal of Heuristics -(-):–

Molina J, Laguna M, Marti R, Caballero R (2007) SSPMO: A scatter tabu search procedure for non-linear multiobjective optimization. INFORMS Journal on Computing 19(1):91–100

Nagarajan V, Ravi R (2008) The directed minimum latency problem. In: Goel A, Jansen K, Rolim JDP, Rubinfeld R (eds) Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques: 11th International Workshop, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, Springer, Berlin, Heidelberg, pp 193–206

Noon CE, Bean JC (1993) An efiicient transformation of the generalized traveling salesman problem. INFOR: Information Systems and Operational Research 31(1):39–44

Nucamendi-Guillén S, Martnez-Salazar I, Angel-Bello F, Moreno-Vega JM (2016) A mixed integer formulation and an efficient metaheuristic procedure for the k-travelling repairmen problem. Journal of the Operational Research Society 67(8):1121–1134

Oliver RL (1987) An investigation of the interrelationship between consumer (dis)satisfaction and complaint reports. Advances in Consumer Research 14(1):218–222

Orman A, Williams H (2007) A survey of different integer programming formulations of the travelling salesman problem. In: Kontoghiorghes EJ, Gatu C (eds) Optimisation, Econometric and Financial Analysis, Springer, Berlin, Heidelberg, pp 91–104

Paniagua-Soto J, Ruiz-Garca J, Iznaola-Muoz M, Ruiz-Serrano L (2013) Multiple sleep latency test in patients with suspected narcolepsy. review of 45 cases. Sleep Medicine 14(1):272 – 273, 5th World Congress on Sleep Medicine, Valencia, Spain

Papadimitriou CM (1994) Computational complexity. Addison-Wesley, Massachusetts, USA

Patterson DA (2004) Latency lags bandwith. Communications of the ACM 47(10):71–75

Picard JC, Queyranne M (1978) The Time-dependent Traveling Salesman Problem and its Application to the Tardiness Problem in One-machine Scheduling. Operations Research 26(1):86–110

Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31(12):1985–2002

Rosenkrantz DJ, Stearns RE, Lewis PM (1974) Approximate algorithms for the traveling salesperson problem. In: IEEE Conference Record of 15th Annual Symposium on Switching and Automata Theory, General Electric Corporate Research and Development, Schenectady, N.Y., USA, pp 33–42

Salami M, Itami C, Tsumoto T, Kimura F (2003) Change of conduction velocity by regional myelination yields constant latency irrespective of distance between thalamus and cortex. Proceedings of the National Academy of Sciences 100(10):6174–6179

Sarddar D, Banerjee J, Jana T, Saha SK, Biswas U, Naskar M (2010) Minimization of handoff latency by angular displacement method using gps based map. International Journal of Computer Science Issues 7(3):29–37

Sarddar D, Chatterjee S, Jana R, Babu SS, Khan HN, Biswas U, Naskar M (2011) Minimization of handoff latency by distance measurement method. International Journal of Computer Science Issues 8(2)

Sarubbi J, Luna H, Miranda G (2008) 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, Cartagena, Colombia

Schurgers C, Tsiatsis V, Ganeriwal S, Srivastava M (2002) Optimizing Sensor Networks in the Energy-Latency-Density Design Space. IEEE Transactions on Mobile Computing 1(1):70–80

Sengupta R, Gupta A, Dutta J (2016) Decision Sciences: Theory and Practice. CRC Press, Boca Raton, FL, USA

Squires NK, Squires KC, Hillyard SA (1975) Two varieties of long-latency positive waves evoked by unpredictable auditory stimuli in man. Electroencephalography and Clinical Neurophysiology 38(4):387 – 401

Stephenson MO (2017) The theory and practice of international humanitarian relief coordination. In: Koops JA, Biermann R (eds) Palgrave Handbook of Inter-Organizational Relations in World Politics, Palgrave Macmillan UK, London, pp 485–502

Sternberg S, Monsell S, Knoll RL, Wright CE (1978) The latency and duration of rapid movement sequences: Comparisons of speech and typewriting. In: Stelmach GE (ed) Information Processing in Motor Control and Learning, Academic Press, pp 117 – 152

Tomasini R, Van Wassenhove L, Van Wassenhove L (2009) Humanitarian Logistics. Palgrave Macmillan UK, UK

Vargas L, Jozefowiez N, Ngueveu SU (2017) A dynamic programming operator for tour location problems applied to the covering tour problem. Journal of Heuristics 23(1):53–80

Zitzler E (1999) Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, ETH Zurich, Switzerland

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4):257–271

Zitzler E, Laumanns M, Thiele L (2002) SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou K, et al (eds) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), pp 95–100