

MATHEURISTICS METHODS: AN APPROXIMATION FOR THE STOP NUMBER MINIMIZATION PROBLEM IN AUTONOMOUS VEHICLES- VIPA

Williane Cristina Ribeiro

Universidade Federal de Ouro Preto
Rua 36, 115 - Loanda, João Monlevade - MG
willianecr@gmail.com

Estefânia de Sá Moura

Universidade Federal de Ouro Preto
Rua 36, 115 - Loanda, João Monlevade - MG
estefania.moura@aluno.ufop.edu.br

Rafael Colares Borges de Oliveira

Université Clermont Auvergne
Rue de la Chebarde, 63178 Aubière, França
rafael.colares@gmail.com

Hervé Kerivin

Université Clermont Auvergne
Rue de la Chebarde, 63178 Aubière, França
herve.kerivin@uca.fr

ABSTRACT

Integer Linear Programming are known for their difficulty to be solved, generally requiring exponential time. In this article, two matheuristics methods are applied to the Unit Stop Number Problem (USNP), a dial-a-ride problem arising from the deployment of autonomous electric vehicles. Such methods are developed with the objective of improving the resolution time of USNP by combining exact methods with heuristic approaches. For this, the matheuristics methods Randomized-Rounding and Relax-and-Fix were studied, adapted and implemented in C++ programming language. Since such methods are mainly based on the decomposition of the general problem into smaller simpler ones, we explore different strategies of decomposition. The computational results obtained allowed us to analyze the impact of each one of these strategies.

KEYWORDS. matheuristics, Unit Stop Number Problem, combinatorial optimization

1. Introduction

Alongside the intense economic and technological development from the past few years, environmental problems have also arisen and represent today a major concern. Among them, we highlight the emissions of gases, such as carbon dioxide (CO_2) released by the use of fuels such as oil, coal and natural gas. According to agency Brasil, 25% of global greenhouse gas emissions are caused by transport, with light vehicles accounting for 45% of the volume emitted (see [Brasil, 2018]). In order to deal with the situation, the Council of the European Union has laid down increasingly stringent rules over the emission of pollutants by cars. In the European Union, all produced vehicles must comply with the imposed emission rates. According to the Council of the European Union, gas emissions from passenger cars are expected to be 15% lower in 2025 and to reach a reduction of 35% in 2030 compared to the limits valid in 2021. In addition, the council also sets out proposals to encourage manufacturing zero or low-emission vehicles (see [Europeu, 2018]).

These factors have stimulated the research for cleaner and more sustainable technologies, such as electric vehicle transportation. European companies Ligier and Easymile have developed electric vehicles called VIPA (a French abbreviation for Autonomous Public Vehicle). These cars are designed to run without a driver in a semi-open or closed space, such as industries, hospitals and universities.

The VIPA can be employed in different ways. In the article of [Pimenta et al., 2017], a fleet of VIPA vehicles travels through a circuit containing a set of predetermined stations, in which customers demand for a ride between two stations. Such demands can be ordered using either a smartphone application or via a physical terminal. In this way, the vehicles travel along the circuit always in the same direction and stop at stations upon demand. It is important to consider that an available vehicle has enough fuel to make an entire lap of the circuit. The stations however are not part of the circuit but attached to it as illustrated in Figure 1. Hence, to perform the required stops the vehicles need to perform a detour, which results in additional costs for the service. Therefore, to reduce costs and optimize the service quality reducing the number of stops seems crucial. *The Stop Number Problem* (SNP) consists of allocating a given set of customer demands to identical vehicles in order to minimize the number of total number. Of course, the vehicle's capacity should always be satisfied, that is, the number of customers simultaneously present in a vehicle is bounded from above by its capacity. When all customer demands are required to be satisfied within a single tour (*i.e.*, no vehicle may perform a complete circuit lap before picking up a customer) and every customer demands for a single seat in a vehicle the problem is called *Unit Stop Number Problem* (USNP). This paper focuses on such variant, even if the algorithms proposed can be easily extended to the more general case (*i.e.*, SNP).

From a theoretical point of view, USNP was proved to be NP-hard by [Baïou et al., 2018], which means that a polynomial-time algorithm is unlikely (to say the least) to solve USNP to optimality (see [Garey e Johnson, 2002]). From a practical point of view, the problem was addressed through an exact approach by [Baïou et al., 2018] via an *Integer Linear Program* (ILP) formulation (which will be further detailed in Section 3). In [Pimenta et al., 2017] the general case SNP is heuristically addressed via a GRASP based algorithm. In this article, we present solution methods that stand in between these two approaches. For that, two methods are developed: (i) Relax and Fix which is a matheuristic (a heuristic method solved in an interpolated way by an exact method);

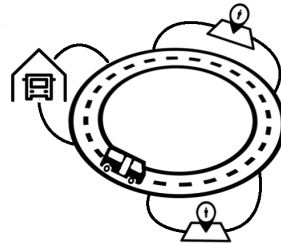


Figure 1: VIPA circuit scheme

and (ii) Randomized Rounding, another matheuristic that is incorporated into a branch-and-bound framework to accelerate the convergence of dual bounds when optimizing the ILP proposed in [Baïou et al., 2018].

The paper is organized as follows. Section 2 presents a small theoretical basis on heuristic and matheuristics methods. Section 3 presents the mathematical modeling of the USNP and the Relax-and-Fix and Randomized-Rounding methods developed in this work. The Section 4 presents and analyzes the computational results obtained using the two matheuristics methods. Finally, in the Section 5 are presented the final considerations and suggestions for future work.

2. Heuristics and Matheuristics Method

As opposed to exact methods, heuristics do not necessarily find an optimal solution to the problem, but rather a feasible solution that is hopefully a good approximation of the optimal one. [Blesa et al., 2009] defines these algorithms as methods of exploration that search to solve a problem without using an exact computational implementation. [Bueno, 2009] also describes heuristics as exploratory algorithms that search a feasible solution (a solution that respects all constraints), by successive approximations directed to an optimal point. According to [Blesa et al., 2009], the main advantages of using a heuristic are economy, speed and ease of use.

In between exact methods and heuristics, the matheuristics emerged. Matheuristics are defined by [Blesa et al., 2009] as conventional heuristic methods with the interpolation of classical mathematical programming techniques and heuristics. It should be noted that even if matheuristics take advantage of exact methods in its construction, they remain heuristic methods and hence no guarantee on the solution optimality is ensured. Nonetheless, heuristics are constantly used as a way to accelerate the resolution of optimization problems.

Indeed, the performance of an ILP is closely related to how fast the solver can converge its primal and dual bounds to the same value and hence can prove the optimality of a given solution. In such case, heuristic methods can be used to speed up the convergence of the primal bounds and hence of the exact method itself. In this paper we adapt two famous matheuristics principles – Randomized Rounding and Relax-and-Fix – and apply them to USNP. For its simplicity, Randomized Rounding is then integrated into an exact approach.

The Randomized Rounding consists of solving the linear relaxation of a given ILP and use information from the fractional optimal solution to build a feasible solution. More precisely, when the variables of an ILP are binary, they assume values between 0 and 1 in the linear relaxation, and

can be seen as the probabilities of assuming their full values in Randomized Rounding [Benedito et al., 2018].

In the Relax-and-Fix method the integer variables of a given ILP are partitioned into two disjunctive subsets. While one of these sets is chosen to maintain the integrality constraints on the variables it contains, the variables in the other sets are relaxed to continuous variables with the objective of significantly reducing the problem's complexity. This problem is then solved and variables having achieved an integer value are fixed until the end of the whole method. A new subset of integer variables having assumed fractional values is then chosen to recover integrality requirements and the procedure is repeated until all variables assume integer values. For this, a predefined criterion for partitioning the variables is required. The size of subsets also affect the number of iterations of the program [Ferreira et al., 2008].

3. Mathematical model and Matheuristics methods

3.1. Unit Stop Number Problem

USNP can be formally described as follows. Let $V = \{1, \dots, n\}$ denote the set of stations sequentially ordered as they appear in the circuit and $E = \{e_1, \dots, e_m\}$ denote the set of unit load customer demands where each demand $e \in E$ is specified by an origin station $o_e \in V$ and a destination station $d_e \in V$ with $o_e < d_e$. Also, let K denote the set of available identical vehicles each with capacity $C \in \mathbb{Z}_+$. Finally, let $\Delta_E(v) = \{e \in E : o_e \leq v \text{ and } d_e \geq v + 1\}$, that is, the set of demands that cross or starts at station v . We say that demand $e \in E$ intersects station $v \in V$ if $e \in \Delta_E(v)$. A feasible solution to USNP is a partition of E into $|K|$ subsets $\{E_1, \dots, E_{|K|}\}$ such that $|\Delta_{E_i}(v)| \leq C$ for any $i \in K$ and any $v \in V$. Vehicle i stops in every station spanned by E_i , that is, $V(E_i)$ and hence the cost of a solution is $\sum_{i=1}^{|K|} |V(E_i)|$. The goal of USNP is to find a feasible solution of minimum cost.

Next we describe the ILP formulation due to [Pimenta et al., 2017] that will be exploited in the development of our matheuristics.

$$\min \sum_{v \in V} \sum_{i \in K} y_v^i \quad (1)$$

$$\text{s.t.:} \quad \sum_{i \in K} x_e^i = 1 \quad \forall e \in E, \quad (2)$$

$$\sum_{j \in \Delta_E(v)} x_e^i \leq C \quad \forall v \in V, i \in K, \quad (3)$$

$$x_e^i \leq y_v^i \quad \forall i \in K, e \in E, v \in \{o_e, d_e\}, \quad (4)$$

$$x_e^i \in \{0, 1\} \quad \forall e \in E, i \in K, \quad (5)$$

$$y_v^i \in \{0, 1\} \quad \forall v \in V, i \in K \quad (6)$$

The model uses two sets of binary variables. The variable x_e^i indicates if demand e is allocated to the vehicle i , that is, $x_e^i = 1$ if $e \in E_i$, otherwise $x_e^i = 0$. The variable y_v^i indicates if vehicle i stops or not at station v , that is, $y_v^i = 1$ if $v \in V(E_i)$, otherwise $y_v^i = 0$.

The objective function (1) of the problem, seeks to minimize the total number of stops. Constraints (2) determine that each demand must be allocated to exactly one vehicle. The capacity

constraints (3) ensures that the vehicle's capacity is respected. The stop constraints (4) imposes that a vehicle must stop at the origin and destination stations of a demand assigned to it. Finally, the constraints (5) and (6) determine the domains of the variables.

3.2. Randomized Rounding

Based on the ILP presented above, we next discuss the proposed Randomized Rounding method. According to [Srinivasan, 1999], Randomized Rounding is an approximation algorithm based on probabilistic calculations that became a standard technique in solving NP-Hard problems. Randomized Rounding manipulates a fractional solution by selecting a variable to be rounded up (or down) using probability techniques. With respect to USNP, since the values of variables y can be easily deduced from variables x by applying

$$y_v^i = \max_{e \in E: v \in \{o_e, d_e\}} x_e^i \quad \forall v \in V, i \in K,$$

one may only focus on making variables x integer.

The Randomized Rounding implementation is divided into two steps. The first step consists of, given a solution of the ILP linear relaxation, allocate a demand $e \in E$ to a vehicle $i \in K$ with a probability given by the value of the associated variable x_e^i in the relaxed solution (see Algorithm 1). Of course, such assignment can only be accepted if the vehicle under analysis has enough space (with respect to its capacity) for accepting the given demand.

Algorithm 1 Randomized Rounding First Step

Require: A solution $\bar{x}$ of the ILP linear relaxation

Ensure: A partial solution x of the ILP

```

1:  $x_e^i = 0 \quad \forall e \in E, i \in K;$ 
2: for  $e \in E$  do
3:   probability  $\leftarrow 0;$ 
4:    $r \leftarrow$  random number between 0 and 1;
5:   for  $i \in K$  do
6:     probability  $\leftarrow$  probability +  $\bar{x}_e^i;$ 
7:     if probability  $> r$  then
8:       if vehicle  $i$  has enough space for accepting demand  $e$  then
9:          $x_e^i \leftarrow 1;$ 
10:      end if
11:      break;
12:    end if
13:  end for
14: end for

```

Since capacity must be respected, it may happen that some demands are not allocated to any vehicle by the end of Algorithm 1. Therefore, the second step of Randomized Rounding creates a penalty strategy for choosing the best vehicle for the rejected demands. Such penalty strategy consists of counting, for each remaining demand, how many additional stops would be required in the given partial solution if the demand is assigned to each vehicle. In case of a tie

between vehicles, the residual capacity is considered. That is, the demand is assigned to the vehicle with highest residual capacity among the ones with smallest penalty. The residual capacity of a vehicle is defined as the sum of free places over each station of the circuit. Algorithm 2 describes this procedure.

Algorithm 2 Randomized Roundind Second Step

Require: A partial solution x of the ILP

Ensure: An USNP feasible solution

```

1: for  $e \in E : \sum_{i \in K} x_e^i = 0$  do
2:   for  $i \in K$  do
3:     if vehicle  $i$  has enough space for accepting demand  $e$  then
4:        $\text{penalty}_i \leftarrow$  Number of additional stops if demand  $e$  is assigned to vehicle  $i$ ;
5:     else
6:        $\text{penalty}_i \leftarrow \infty$ ;
7:     end if
8:      $\text{residualCapacity}_i \leftarrow \sum_{v \in V} (C - \sum_{e \in \Delta_v} x_e^i)$ ;
9:   end for
10:   $k \leftarrow$  the vehicle with highest residual capacity among the ones with smallest penalty;
11:   $x_e^k \leftarrow 1$ ;
12: end for
```

Computational results obtained when applying Randomized Rounding to solve USNP are presented and analyzed in Section 4.

3.3. Relax and Fix

The Relax-and-Fix method is a matheuristic method based on relaxing the integrality restrictions for some of the integer variables of an ILP in order to make it significantly easier to be solved. [Ferreira et al., 2010] stated that the Relax-and-Fix method is based on the resolution of successive partial linear relaxations. In each iteration, the set of integer variables is partitioned into two subsets and variables in one of the subsets are relaxed to admit continuous values. The problem is then solved by the exact method and all the variables that obtained integer values are fixed and stored. The remaining variables are then redistributed into two subsets and the process is repeated until all variables are fixed. Subsets can be created according to a strategy defined specifically for each problem (see [Pochet e Wolsey, 2006]).

Notice that in an optimal solution from formulation (1)-(6), variables y automatically assume integer values once variables x are integer. For this reason, we may only focus on applying the method over variables x . In this study, variables x are partitioned into three subsets: (i) X^{fix} containing variables that have already been fixed to integer values, (ii) X^{int} containing unfixed variables that are restricted to integer values, and (iii) X^{relax} containing the variables that can admit continuous values. In each iteration, the number of fixed variables increases and the number of relaxed variables decreases.

The set of fixed variables X^{fix} starts empty and for determining which variables should keep the integrality requirements (*i.e.*, variables in X^{int}), we use the notion of demands intersecting

a station. More precisely, at each iteration a station $v' \in V$ is chosen and variables x_e^i – that do not belong to X^{fix} – whose demand intersects station v' are restricted to assume $\{0, 1\}$ values. That is,

$$X^{int} = \{x_e^i : e \in \Delta_E(v'), i \in K\} \setminus X^{fix}.$$

Finally, variables that do not belong neither in X^{fix} nor in X^{int} are put into X^{relax} and hence are allowed to assume continuous values, that is

$$X^{relax} = \{x_e^i : e \in E, i \in K\} \setminus \{X^{fix} \cup X^{int}\}.$$

Once the sets X^{fix} , X^{int} and X^{relax} have been defined, the associated Mixed Integer Linear Program (MILP) is solved. Variables that obtained integer values are then stored into X^{fix} and a new station $v'' \in V$ is chosen for defining the next set of variables to recover integrality. The program ends when all variables have been fixed and stored in X^{fix} . Our proposed Relax-and-Fix method is formally described in Algorithm 3.

Algorithm 3 Relax-and-Fix

Require: An instance of USNP

Ensure: A feasible solution

```

1:  $X^{fix} \leftarrow \emptyset$ 
2:  $X^{int} \leftarrow \emptyset$ 
3:  $X^{relax} \leftarrow \{x_e^i : e \in E, i \in K\}$ 
4: while  $\{X^{int} \cup X^{relax}\} \neq \emptyset$  do
5:   Choose a station  $v' \in V$  to define the next intersection
6:    $X^{int} \leftarrow \{x_e^i : e \in \Delta_E(v'), i \in K\} \setminus X^{fix}$ 
7:    $X^{relax} \leftarrow \{x_e^i : e \in E, i \in K\} \setminus \{X^{fix} \cup X^{int}\}$ 
8:   Solve the associated MILP to obtain a solution  $(\bar{x}, \bar{y})$ 
9:    $X^{fix} \leftarrow \{x_e^i : e \in E, i \in K \text{ and } \bar{x}_e^i \text{ is integer}\}$ 
10: end while

```

Next we present the computational results obtained when applying our Randomized Rounding and Relax-and-Fix methods to solve USNP.

4. Computational Results

In order to test the proposed algorithms, we used the commercial MILP solver ILOG CPLEX 12.10 coupled with ILOG CPLEX Concert Technology that allows an interface with CPLEX libraries using C++ language. A set of 18 USNP instances was generated using different scenarios based on the capacity C , the number of demands m and the density ρ (defined as the number of demands over the number of stations). For each combination of parameters $C \in \{2, 5, 8\}$, $m \in \{30, 40, 50\}$ and $\rho \in \{1.5, 4.5\}$, an instance was randomly generated. The generated instances are referred by the code C_m_p . All tests were performed on a single-threaded machine equipped with Intel Xeon E7 2.50 GHz and 32Gb of RAM running linux.

4.1. Randomized Rounding

Due to its simplicity (*i.e.*, with the appropriate data structures the algorithm runs in linear time), the proposed Randomized Rounding procedure can be implemented within a Branch-and-Bound framework to quickly generate feasible solutions. In this sense, the relaxed solution used in the input comes from the the solution obtained from the linear relaxation of a given node in the Branch-and-Bound enumeration tree. The goal here is allow the solver to improve its upper bounds throughout the optimization and hence speed up its performance. In order to evaluate the method, each generated instance was tested within a time limit of one hour. We compare the performances obtained by CPLEX with its default settings and the ones obtained when applying Randomized Rounding to generate feasible solutions along the optimization process.

In addition, it is reasonable to imagine that if the matheuristic is inputted with similar fractional solutions, its output will consist of similar feasible solutions. Moreover, as one goes deeper into the Branch-and-bound enumeration tree, it becomes more and more unlikely to obtain a better incumbent solution. For these reasons and to avoid wasting time into redundant executions of the matheuristic, Randomized Rounding procedure is not applied at all nodes of the enumeration tree. Instead, it is called systematically only for the first 5 nodes. From then, the method is invoked once on every 10 nodes.

One should also note that the Randomized Rounding procedure described by Algorithm 1 and Algorithm 2 treats each demand sequentially and hence it may have its behaviour affected depending on the order on which demands are presented. In other words, the method may yield different feasible solutions if demands are considered in different sequences. In this work, two sequencing strategies are considered:

1. Demands are ordered by its intrinsic lexicographic order. That is, given two demands $i \in E$ and $j \in E$, we say that $i \prec j$ if $o_i < o_j$ or $(o_i = o_j \text{ and } d_i < d_j)$.
2. Demands are ordered decreasingly by its number of shared stations. That is, given two demands $i \in E$ and $j \in E$, we say that $i \prec j$ if

$$\left| \left\{ e \in E : o_e \in \{o_i, d_i\} \text{ or } d_e \in \{o_i, d_i\} \right\} \right| > \left| \left\{ e \in E : o_e \in \{o_j, d_j\} \text{ or } d_e \in \{o_j, d_j\} \right\} \right|.$$

Table 1 shows the results obtained for each generated instance. Results obtained with default CPLEX are displayed under column CPL. Results obtained with the application of Randomized Rounding within CPLEX framework are displayed under columns RR1 and RR2 for strategies 1 and 2, respectively. Under section Execution time, the total time spent is displayed in seconds. The relative gap percentage obtained by the end of the time limit is provided under section Gap. Section Nodes stands for the number of nodes (in thousands) explored in the enumeration tree and Obj provides the best objective function value obtained.

Table 1: Randomized Rounding Results

Inst.	Execution time (s)			Gap (%)			Nodes			Obj		
	CPL	RR1	RR2	CPL	RR1	RR2	CPL	RR1	RR2	CPL	RR1	RR2
2_30_1.5	93.4	64.2	86.1	0	0	0	7.1	6.1	7.9	35	35	35
2_30_4.5	27.7	57.6	77.6	0	0	0	1.8	8.7	13.4	24	24	24
2_40_1.5	3600	3600	3600	10.8	13.9	14	137.6	94.6	88.3	50	50	50
2_40_4.5	3600	3108	3369	4.2	0	0	169.8	318.9	330.1	35	35	35
2_50_1.5	3600	3600	3600	19.2	11.6	11.4	48.6	42.4	40.7	63	62	62
2_50_4.5	3600	3600	3600	37.3	35.8	37	49.0	35.7	38.8	48	47	49
5_30_1.5	127.6	24.5	24.6	0	0	0	0.9	1.4	1.4	25	25	25
5_30_4.5	407.6	11.1	10.9	0	0	0	138	347	347	14	14	14
5_40_1.5	629.1	751	719.4	0	0	0	18.8	30.9	30.9	37	37	37
5_40_4.5	310.6	804.5	755.2	0	0	0	5.9	22.9	22.9	19	19	19
5_50_1.5	3600	3600	3600	5.5	7.4	7.4	49.6	50.4	51.5	46	46	46
5_50_4.5	3600	3600	3600	23.7	24.6	24.3	35.1	38.3	40.9	30	30	30
8_30_1.5	2.7	3	3.1	0	0	0	0.02	0.01	0.01	24	24	24
8_30_4.5	2.7	4.8	4.6	0	0	0	0.04	0.02	0.02	11	11	11
8_40_1.5	32.3	37.6	36.1	0	0	0	0.6	0.03	0.03	32	32	32
8_40_4.5	8.4	22.9	21.7	0	0	0	0.09	0.17	0.17	14	14	14
8_50_1.5	188.3	223.3	201.9	0	0	0	2.14	1.2	1.2	41	41	41
8_50_4.5	2416	2665	2526	0	0	0	17.2	24.6	24.6	22	22	22

The results obtained show that in some cases the presence of the matheuristic allowed the framework to have some gain of performance. For example the *Execution time* has reduced in the instances 2_30_1.5, 2_40_4.5, 5_30_1.5 and 5_30_4.5, the *Gap* was lower for instances 2_40_4.5, 2_50_1.5 and 2_50_4.5, the size of the search tree (*Nodes*) was smaller for several instances. Finally, for instances 2_50_1.5 and 2_50_4.5 a solution with better cost could be found within time limit. See the results in bold in Table 1.

We can observe from Table 1 that the instances where $C = 2$ were the hardest to be solved by CPLEX. Indeed for $C = 2$ and $m \geq 40$, any of the tested instances could be solved to optimality within time limit. Fortunately, for these same instances the matheuristic revealed itself efficient, allowing to solve one of the unsolved instances to optimality and decreasing the relative gap obtained for the others. For the instances where $C=5$ and $C=8$, we did not obtained significant performance gains, except for a reduction in the execution time for instances 5_30_1.5 and 5_30_4.5.

As the results show in the Table 1, the two strategies present very close results. In terms of speeding up the resolution, *RR1* gives better results for the most difficult instances (*i.e.*, $C = 2$) while *RR2* for the instances where $C \geq 5$, but in this instances, the exact method (*CPL*) gives a better solution than the Randomized Rounding. In larger instances, like instance 2_50_4.5 which is stopped by time limit, *RR1* gets a better objective solution with a smaller *GAP* and also with a smaller number of visits *Nodes* in the search tree. This allows us to conclude that Randomized Rounding is most effective for the instances more difficult (where $C=2$) when the demands are ordered by its intrinsic lexicographic order (*RR1*).

4.2. Relax-and-Fix

In the context of Relax-and-Fix procedures, it is well known that the criterion used for defining which variables are considered integer and which ones are considered continuous can have a great influence on the difficulty of solving the resulting MILP (see [Ferreira et al., 2008]). Therefore, in order to test the performance of our Relax-and-Fix heuristic, two strategies were designed to decide which stations should be first chosen to define X^{int} :

1. The Demands are inserted in increasing order of the demand intersection station. That is, the demands of intersection station 1 are treated before the demands of intersection station 2.
2. The demands are inserted in increasing order of quantity of customers at intersection station. That is the demands in the station with greater quantity of customers are treated before

In order to avoid replicating efforts, only stations $v \in V$ whose set $\Delta_E(v)$ is inclusion-wise maximal are considered on the definition of X^{int} . Such stations are referred to as a *maximal-intersection station*. In other words, a station $v \in V$ is said to be a *maximal-intersection station* if there exists no station $v' \in V$ for which $\Delta_E(v) \subset \Delta_E(v')$.

Moreover in order to limit the execution time of the matheuristic procedure, a time limit was set to each MILP resolution. In this study we considered time limits of 300, 900, 1800 seconds. When time limit is reached, the optimization is stopped and the best known feasible solutions is kept.

The Table 2 shows the results obtained for the tested instances. The results obtained with default CPLEX are displayed under column CPL and the results obtained with Relax-and-Fix for strategies 1 and 2 are displayed under columns RR1 and RR2, respectively, for each time limit T1, T2 and T3. The cost of the best solution found and the total time spent during optimization (in seconds) are displayed under sections Obj and Execution time.

Table 2: Relax and Fix Results

Inst.	Obj							Execution time (s)								
	CPL	RR1			RR2			CPL	RR1			RR2				
		T1	T2	T3	T1	T2	T3		T1	T2	T3	T1	T2	T3		
2.30.1.5	35	36	36	36	37	37	37	93.4	36.9	37.7	37.7	58.8	57.6	58.1		
2.30.4.5	24	25	25	25	24	24	24	27.7	8.1	8.2	8.3	14.2	14.2	14.3		
2.40.1.5	50	51	51	51	52	52	52	3600	442	439	439	405	405	408		
2.40.4.5	35	36	36	36	37	37	37	3600	113	114	112	127	124	129		
2.50.1.5	63	64	64	64	66	66	64	3600	479	1109	1981	393	1009	1912		
2.50.4.5	48	49	49	48	49	49	49	3600	600	1214	1539	638	1520	2428		
5.30.1.5	25	25	25	25	25	25	25	127	8.1	7.9	8.4	31.9	31.5	32.4		
5.30.4.5	14	15	15	15	15	15	15	408	6.2	6.3	6.4	6.4	5.9	6.0		
5.40.1.5	37	38	38	38	38	38	38	629	68.3	65.3	66.4	63.4	61.9	63.7		
5.40.4.5	19	22	22	22	20	20	20	311	33.8	35.4	34.2	23.4	23.0	23.4		
5.50.1.5	46	48	48	48	47	47	47	3600	271	263	255	220	216	222		
5.50.4.5	30	33	32	32	33	33	33	3600	506	1101	1537	326	329	337		
8.30.1.5	24	25	25	25	25	25	25	2.7	3.2	3.2	3.2	2.7	2.8	2.7		
8.30.4.5	11	12	12	12	12	12	12	2.7	2.8	2.7	2.8	2.4	2.5	2.4		
8.40.1.5	32	33	33	33	33	33	33	32.3	44.0	44.9	43.6	69.7	70.8	71.9		
8.40.4.5	14	14	14	14	14	14	14	8.4	9.7	9.9	9.6	10.6	10.4	10.9		
8.50.1.5	41	41	41	41	41	41	41	188	96.7	92.8	96.1	99.5	102	102		
8.50.4.5	22	24	24	24	23	23	23	2416	202	191	197	77.2	74.9	79.3		

In Table 2 it is possible to observe that in some cases the Relax-and-Fix was not capable of finding a solution that was as good as the one found by the exact method. However, the method remains quite competitive as on average it was capable of providing solutions that were only 3.68% far from the ones obtained with CPLEX. Such small difference is considered acceptable for heuristic methods. The variation of the time limit only directly influenced the value of the objective function

of the hardest instances (*e.g.*, 2_50_1.5, and 2_50_4.5). However, increasing time limit has obviously an important impact on the execution time.

In the same way that the Randomized Rounding, Relax-and-Fix's performance also increases as capacity decreases. Except for the easiest instances (8_30_1.5, 8_30_4.5, 8_40_1.5 and 8_40_4.5), the method does not affect the reduction of the execution time. For all other instances Relax-and-Fix was able to significantly reduce the execution time. On instance 2_40_4.5, such reduction was of 96%.

As we can see, the two strategies used to choose the demands in the construction of Relax-and-Fix, generated very close results. For the instances where $C = 2$, *RR1* has a better gain of in terms of execution time while *RR2* has better results when $C = 5$.

5. Conclusions

This article proposes in parallel two matheuristics methods – the Randomized-Rounding and the Relax-and-Fix – in order to solve the so-called Unit Stop Number Problem. Such problem arises from the deployment of specific autonomous electric vehicles named VIPA that are developed by European companies. Matheuristics are known to interpolate heuristic methods with exact ones in order to find feasible solutions that are near-optimal in a reasonable time. Such interpolation may be done in many different ways. On one hand, Randomized-Rounding focus on heuristically manipulating the optimal solution of a linear relaxation to obtain a feasible solution. On the other hand, Relax-and-Fix has its foundations on the decomposition of a given Integer Linear Programming formulation into many simpler Mixed Integer Linear Programs.

Due to its low computational complexity, the Randomized-Rounding method could be integrated into a Branch-and-Cut framework with the intention of providing exact solutions in faster time. Such procedure revealed itself to be effective for the hardest instances tested. Indeed, for such instances the method was able to speed up convergence towards optimality and reduce the branching enumeration tree size (*i.e.*, less nodes were required to be explored).

Relax-and-Fix, on the contrary, relies on the resolution of Mixed Integer Linear Programs that may require exponential time to be solved. The inclusion of such procedure in a Branch-and-Cut framework is therefore discouraged. Our results show however that such MILPs are significantly easier to solve than the original formulation. Such fact allows to obtain feasible solutions in reasonable time. Moreover, our results indicate that the method is quite efficient since the cost of the obtained feasible solutions either are either optimal or fairly near-optimal.

In short, the methods proved to be applicable to the problem under study, and in different ways were capable of improving the resolution performance of USNP. As perspectives for future studies, one idea is to test the performance of the proposed methods on real-world instances, for which we still do not have access since the VIPA is still very recent.

6. Acknowledgements

The authors would like to thank the laboratory of excellence IMobS3, the University of Clermont Auvergne, the Federal University of Ouro Preto and the National Council for the Improvement of Higher Education (CAPES) for the financial and material support.

References

- Baiou, M., Colares, R., and Kerivin, H. (2018). The stop number minimization problem: Complexity and polyhedral analysis. *International Symposium on Combinatorial Optimization*.
- Benedito, M. P. L. et al. (2018). Approximation algorithms for hub location problems: Algoritmos de aproximação para problemas de localização e alocação de terminais. *1*.
- Blesa, M. J., Blum, C., di Gaspero, L., and Roli, A. (2009). Hybrid metaheuristics. *6th international Workshop, HM*, p. 180–181.
- Brasil, A. (2018). Efeito estufa: transporte responde por 25% das emissões globais. URL <https://agenciabrasil.ebc.com.br/geral/noticia/2018-12/efeito-estufa-transporte-responde-por-25-das-emissoes-globais>. Accessed:2020.05.13.
- Bueno, F. (2009). Métodos heurísticos: Teoria e implementações. *IFSC. Araranguá*.
- Europeu, C. (2018). Normas de emissão de co2 para automóveis de passageiros e veículos comerciais ligeiros: Conselho define posição. URL <https://www.consilium.europa.eu/pt/press/press-releases/2018/10/10/co2-emission-standards-for-cars-and-vans-council-agrees-its-position/>. Accessed:2019.03.19.
- Ferreira, D., Morabito, R., and Rangel, S. (2008). Um modelo de otimização inteira mista e heurísticas relax and fix para a programação da produção de fábricas de refrigerantes de pequeno porte. *Production*, 18(1):76–88.
- Ferreira, D., Morabito, R., and Rangel, S. (2010). Relax and fix heuristics to solve one-stage one-machine lot-scheduling models for small-scale soft drink plants. *Science+Business Media*, 37(4):684–691.
- Garey, M. R. and Johnson, D. S. (2002). *Computers and intractability*, volume 29. wh freeman New York.
- Pimenta, V., Quilliot, A., Toussaint, H., and Vigo, D. (2017). Models and algorithms for reliability-oriented dial-a-ride with autonomous electric vehicles. *European Journal of Operational Research*, 257:601–613.
- Pochet, Y. and Wolsey, L. A. (2006). Production planning by mixed integer programming. *Science+Business Media*.
- Srinivasan, A. (1999). Approximation algorithms via randomized rounding: a survey. *Series in Advanced Topics in Mathematics, Polish Scientific Publishers PWN*, p. 9–71.