GRASP Algorithms for the Robust Railway Network Design Problem

Bosco García-Archilla, Antonio J. Lozano , Juan A. Mesa, Federico Perea

March 16, 2011

Abstract

This paper analyzes the solvability of a railway network design problem and its robust version. These problems are modeled as integer linear programming problems with binary variables, and their solutions provide topological railway networks maximizing the trip coverage in the presence of a competing mode, both assuming that the network works fine and that links can fail, respectively. Since these problems are computationally intractable for realistic sizes, GRASP heuristics are proposed for finding good feasible solutions. The results obtained in a computational experience indicate that our GRASP algorithms are suitable for railway network design problems.

Keywords: Transportation Robustness Heuristics

1 Introduction

Increasing mobility, longer trips due to house spreading, and traffic congestion in entrances to cities, are some of the reasons why many metropolitan areas are planning, constructing or extending railway systems. The investment in the construction or extension of a Railway Network (RN) (which consists of selecting a set of stations and the links joining them) is justified by the savings in travel times and energy, and by the considerable reduction of pollution that trains generate with respect to buses or cars. Due to the high construction cost of a RN, it is important to pay attention to issues affecting effectiveness and robustness when planning new lines. Since railway networks do not always work as expected, due to uncertainty in the input data, climatological events, or even intentional attacks, the RN to be designed should be not only effective but also "robust". Borrowing from [6] robustness can be defined as *the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.* In our context, a RN is said to be robust when it maintains its functionality as well as possible when one of its links fails.

In this paper we introduce an Integer Linear Programming (ILP) model for the railway network design (RND) problem. Similar models have previously been posed in Laporte et al. [7] and Laporte et al. [8], who design a set of railway lines to optimize a certain utility function, and deal with the robustness of the resulting network. Marín and García-Ródenas [9] go one step further and consider a model with a Logit function for the trip coverage, instead of the previous all-or-nothing models. Since capacity constraints were not considered in these papers, the resulting lines could not be completely defined because frequencies could not be determined. Therefore a subsequent line planning step was needed. In Marín et al. [11] an attempt to integrate both network design and line planning steps is proposed.

The main difference of the previously cited models with respect to the one presented in this paper is that they were thought to design a set of lines, unlike the one in this paper which only designs a topological network, i.e., stations and links joining the stations. The reason why we choose this simplification is that the models presented in [7] and [8] were far too complex. They could hardly handle toy instances made of 10 possible stations. Besides, as mentioned before, the lines designed were not completely defined and the line planning phase was needed anyway. With this simplified model, in which several sets of variables and constraints have been reduced or eliminated with respect to those in [8] and [7], instances constituted by up to 18 possible stations and all $18 \cdot 17 = 306$ origin-destination (O/D) pairs have been solved to optimality in minutes. Besides this simplification, our new ILP model makes use of, at most, 2-index variables, as opposed to the 4-index formulation of previously cited models.

Thus, the model introduced in this paper only aims at designing the infrastructure network, that is, the set of links and stations to be built, whereas the design of lines (together with their frequencies) is to be done in the subsequent line planning phase. Line Planning Problems (LPP) consists of determining the number of lines, their origins, itineraries and destinations as well as the frequency of each one. Lines are selected from a line pool or as paths of a infrastructure (or underlying) network and the objectives can be cost-oriented (e.g. see [5]) and customers-oriented (e.g. see [15]).

Our RND problem can be schematically described from a set of feasible networks R, and a utility function $K : R \to \mathbb{R}$, which assigns to each feasible network a measure to be optimized such as total travel time, trip coverage, etc. Therefore, when the objective is to be maximized (like trip coverage) our problem reduces to

$$\max_{r \in R} K(r).$$

In previous papers issues such as computational complexity or robustness of the RND problem were addressed, but efficient algorithms were not proposed. Although some attempts have been devoted for designing exact algorithms to solve similar problems, see for instance Marín and Jaramillo [10] who apply Benders decomposition techniques, there is no exact algorithm that can deal with real instances of our RND problem with the technology available. Therefore, heuristics are needed in order to provide solutions in a reasonable amount of time. As an example, the Route Generation Algorithm of Baaj and Mahmassani [1] is used by Mauttone and Urquhart [12] in a certain transportation network design problem. Other heuristics for similar problems are due to Nesmachnow et al. [14], who propose evolutionary algorithms for certain communication network design problems.

In this paper we propose Greedy Randomized Adaptive Search Procedure (GRASP) algorithms for our RND problem. GRASP algorithms have been widely used for solving large scale optimization problems since the pioneering work by Feo and Resende [4]. An example within the network design field is Cancela et al. [2], who propose a GRASP algorithm for a telecommunication network design problem. In other fields, [13] successfully applied them to Multi-Target Multi-Sensor Tracking problems and more recently, [3] proposed a GRASP algorithm in order to obtain lower bounds for the manufacturing cell formation problem.

As a second step, in this paper we also address the robustness property of the RND problem. We assume that any of the links can fail, and that the utility of the network is adversely affected by such link failures. Let $K(r,e)$ be the utility function of network $r \in R$ when link $e \in r$ fails. Any link can fail but no more than one link can fail at the same time. If the probability that link e fails is known and equal to γ_e, the construction of a robust network is a probabilistic problem, named probabilistic railway network design (PRND) problem, and can be solved by optimizing the expected utility under all possible scenarios (no failure scenario and single link failure scenarios). This problem was originally introduced in [8] and reduces to

$$\max_{r\in R}\{(1-\sum_{e\in r}\gamma_e)K(r)+\sum_{e\in r}\gamma_e K(r,e)\}\,. \qquad \text{(PRND)} \tag{1}$$

Despite the substantial computational time reduction thanks to the models we propose in this paper, both RND and PRND problems are impossible to handle for realistic instances.

The rest of the paper is structured as follows. Section 2 introduces the Integer Linear Programming (ILP) model that yields an optimal infrastructure network, and the GRASP to be used is presented. Also the extension to the PRND model and the corresponding GRASP algorithm are introduced. The results of some experiments are commented in Section 3. The paper ends with some conclusions and a list of references.

2 A Railway Network Design problem

We begin this section by describing our railway network design problem in the presence of a competing mode, which maximizes ridership assuming the whole network works as planned.

In our RND problem it is assumed that the mobility patterns in a metropolitan area are known. This implies that the number of potential passengers traveling from each origin to each destination is given. It is also assumed that the locations of potential stations are known. In addition, there already exists a different mode of transportation (for example a bus network) competing with the railway to be built. To decide which mode each demand is allocated to, the travel times for both systems are compared in an all-or-nothing way: a potential passenger will use the RN if and only if he/she finds the RN faster than the competing transportation mode. The aim of the model is to design a network (i.e. to decide at which nodes stations are to be located and which edges must be constructed) covering as many passengers as possible, that is, a RN such that the number of passengers for which using the RN is faster than the alternative competing mode is maximized. Since the resources are limited there is also a budget constraint on the construction costs. The model uses the following data and notation:

1. A set $N = \{1, 2, \dots, n\}$ of potential sites for stations is given.

2. A set $E = \{e_1, ..., e_m\} \subseteq N \times N$ of feasible (bidirectional) edges linking the elements in N is known. Therefore, we consider the undirected graph $\mathcal{G} = (N, E)$, from which edges are to be selected to form railway lines. For convenience, we sometimes use the set of arcs of E, denoted by $A = \{a_1, ..., a_{2m}\} \subseteq N \times N$, and the corresponding directed graph (N, A). For notation purposes we denote by a^s, a^t, e^s and e^t the origin and end nodes of arc $a \in A$ and edge $e \in E$, respectively. For the sake of notation, for any $a \in A$ and $e \in E$ we define $a' = (a^t, a^s)$ and $e' = (e^t, e^s)$, respectively. The following relations follow from these definitions:

   (a) $E = \{a \in A : a^s < a^t\}$,

   (b) if $e \in E$ then $e' \notin E$,

   (c) $A = E \cup \{e' : e \in E\}$.

3. Every feasible arc $a \in A$ has an associated length d_a which can typically be approximated by its Euclidean distance. However, forbidden regions between two stations such as high mountains, rivers, natural parks, etc. may not allow the railway to take the straight line between two points, and therefore distances may increase as a result. To avoid this problem, d_a can also be interpreted as the time needed to traverse arc $a \in A$. Note that, in general, $d_a \neq d_{a'}$.

4. For each node $i \in N$ and each edge $e \in E$ there exists an associated cost of constructing the corresponding infrastructures: c_i is the cost of building a station at node i, c_e is the cost of building link e. A bound $C_{\max}$ on the available budget is also given.

5. The ordered O/D pair set, also referred to as the set of demands, is $W = \{w_1, ..., w_\nu\} \subset N \times N$. For notation purposes, w^s and w^t denote the origin node and destination node of O/D pair w, and $w' = (w^t, w^s)$ for each $w \in W$. The mobility pattern is given by a matrix $G = (g_w)_{w \in W}$, where g_w denotes the number of passengers going from w^s to w^t, $\forall\ w = (w^s, w^t) \in W$. The time needed to go from $w^s \in N$ to $w^t \in N$ by the complementary mode is u_w^{COM}.

A typical example for our experiments is the following.

Example 2.1 *Consider 9 possible stations and the set*

$$\begin{aligned} E = \; & \{(1,2),(1,5),(2,3),(2,4),(2,5),(3,5),(4,5), \\ & (4,7),(4,8),(5,6),(5,7),(5,9),(6,8),(6,9),(8,9)\}, \end{aligned}$$

as depicted in Figure 1. The construction costs associated to the stations are shown by the

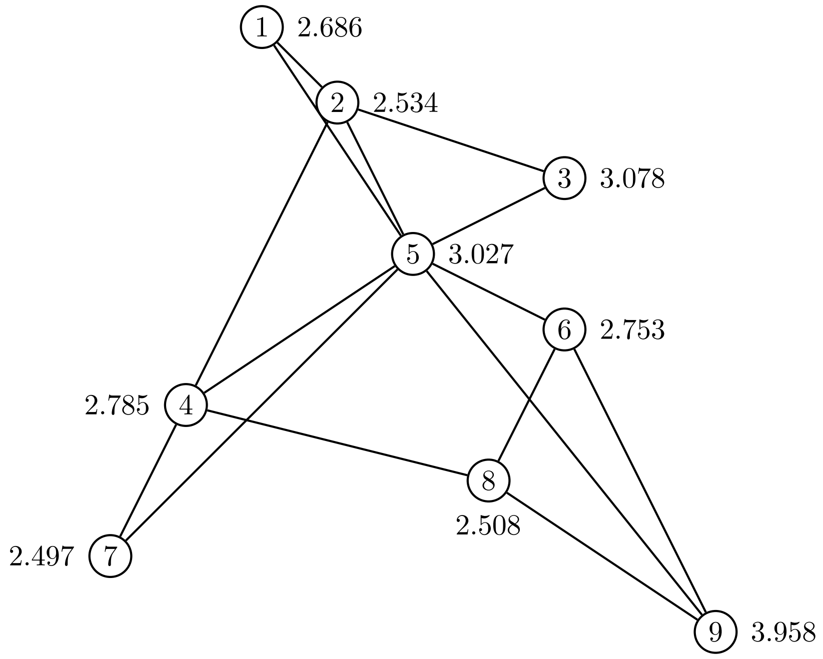

Figure 1: Test Network.

corresponding node. The maximum budget was set to $C_{\max} = 25.394$.

The mobility patterns (number of trips between potential stations) are shown in matrix G.

$$G = \begin{pmatrix} - & 35 & 6 & 7 & 41 & 18 & 22 & 11 & 8 \\ 38 & - & 40 & 40 & 26 & 46 & 7 & 45 & 37 \\ 31 & 23 & - & 45 & 42 & 23 & 14 & 23 & 16 \\ 23 & 12 & 47 & - & 48 & 36 & 11 & 35 & 49 \\ 44 & 33 & 31 & 30 & - & 22 & 32 & 45 & 9 \\ 37 & 36 & 32 & 21 & 37 & - & 17 & 45 & 18 \\ 32 & 26 & 27 & 6 & 21 & 15 & 0 & 9 & 40 \\ 37 & 13 & 5 & 13 & 37 & 38 & 47 & - & 32 \\ 13 & 28 & 46 & 40 & 29 & 21 & 40 & 10 & - \end{pmatrix}.$$

For this example, the travel times between nodes by the complementary mode (u_w^{COM}) were set proportional to their Euclidean distances, whereas the travel times to traverse the railway links were supposed to be half of the corresponding times by the complementary mode, that is, $d_a = u_{a^s a^t}^{COM}/2 \ \forall \ a = (a^s, a^t) \in A$ (note that (a^s, a^t) can also be considered as an O/D pair). This resulted in

$$u_w^{COM} = \begin{pmatrix} 0 & 1.506 & 2.544 & 3.226 & 2.722 & 3.902 & 5.278 & 5.002 & 6.482 \\ 1.506 & 0 & 1.04 & 3.364 & 1.602 & 2.748 & 5.484 & 4.23 & 5.386 \\ 2.544 & 1.04 & 0 & 3.866 & 1.42 & 2.264 & 5.902 & 4.006 & 4.806 \\ 3.226 & 3.364 & 3.866 & 0 & 2.652 & 3.06 & 2.128 & 2.764 & 4.7 \\ 2.722 & 1.602 & 1.42 & 2.652 & 0 & 1.186 & 4.572 & 2.666 & 3.812 \\ 3.902 & 2.748 & 2.264 & 3.06 & 1.186 & 0 & 4.586 & 1.838 & 2.638 \\ 5.278 & 5.484 & 5.902 & 2.128 & 4.572 & 4.586 & 0 & 3.396 & 5.212 \\ 5.002 & 4.23 & 4.006 & 2.764 & 2.666 & 1.838 & 3.396 & 0 & 1.956 \\ 6.482 & 5.386 & 4.806 & 4.7 & 3.812 & 2.638 & 5.212 & 1.956 & 0 \end{pmatrix}$$

and

$$\begin{aligned} \{ \ & d_{1,2} = 0.753, \ d_{1,5} = 1.361, \ d_{2,3} = 0.520, \ d_{2,4} = 1.682, \\ & d_{2,5} = 0.801, \ d_{3,5} = 0.710, \ d_{4,5} = 1.326, \ d_{4,7} = 1.064, \\ & d_{4,8} = 1.382, \ d_{5,6} = 0.593, \ d_{5,7} = 2.286, \ d_{5,9} = 1.906, \\ & d_{6,8} = 0.919, \ d_{6,9} = 1.319, \ d_{8,9} = 0.978\}, \end{aligned}$$

with $d_a = d_{a'} \ \forall \ a \in A$. Although in this example $u_w^{COM} = u_{w'}^{COM} \ \forall \ w \in W : w^s < w^t$, this is not needed in our formulation.

The construction cost of each possible link is supposed to be equal to their Euclidean length. A maximum budget according to the construction costs was chosen so that not all possible links can be constructed (note that if we have a large enough budget the solution is trivial:

build the complete network). In this example $C_{\max} = 25.394$.

The variables of our RND model are:

- $x_a = 1$ if arc $a \in A$ is included in the RN, zero otherwise. We sometimes make use of x_e if we refer to an edge instead of an arc (note that $E \subset A$).
- $y_i = 1$ if station $i \in N$ is included in the RN, zero otherwise.
- $f_a^w = 1$ if O/D pair w is assigned to the RN and use arc $a \in A$, zero otherwise.
- $r_w = 1$, if O/D pair w is allocated to the RN, that is, if its fastest route in the RN takes less time than the alternative mode u_w^{COM}; zero otherwise.
- z_{nf} denotes the demand captured by the network. The subindex nf stands for *no* $failure$, and is necessary to distinguish between this variable and those that will be used when modeling the PRND problem.

The objective is to maximize the RN trip coverage (in the absence of failures), i.e., maximize

$$z_{nf} = \sum_{w \in W} g_w r_w. \tag{2}$$

The constraints of our model are grouped according to their aims. Note that for some of them we make use of the standard *big M* technique in linear programming, where M is a sufficiently large real number.

- Budget constraints

$$\sum_{e \in E} c_e\, x_e + \sum_{i \in N} c_i y_i \leq C_{\max}. \tag{3}$$

- Alignment location constraints

$$x_e \leq y_i,\ e \in E,\ i \in \{e^s, e^t\} \tag{4}$$

$$x_e = x_{e'},\ e \in E. \tag{5}$$

- Routing demand conservation constraints

$$\sum_{a \in A: w^s = a^t} f_a^w = 0,\ w \in W, \tag{6}$$

$$\sum_{a \in A: w^s = a^s} f_a^w = r_w,\ w \in W, \tag{7}$$

$$\sum_{a \in A: w^t = a^t} f_a^w = r_w,\ w \in W, \tag{8}$$

$$\sum_{a \in A : w^t = a^s} f_a^w = 0,\ w \in W, \tag{9}$$

$$\sum_{a \in A : k = a^t} f_a^w - \sum_{a \in A : k = a^s} f_a^w \ = \ 0,\ \forall\ k \notin \{w^s, w^t\},\ w \in W. \tag{10}$$

- Location-Allocation constraints

$$f_a^w + r_w - 1 \ \leq \ x_a,\ a \in A,\ w \in W. \tag{11}$$

- Splitting demand constraints

$$\varepsilon + u_w - u_w^{COM} \ - \ M\,(1 - r_w) \ \leq 0,\ w \in W, \tag{12}$$

  where $u_w \ = \ \sum_{a \in A} d_a f_a^w + u_w^{COM}(1 - r_w)$ and $\varepsilon > 0$ is a small tolerance. Note that u_w is not a variable in our model, and is here defined for simplicity in writing constraints (12) only.

- Binary constraints

$$x_a,\ y_i,\ f_a^w,\ r_w \in \{0, 1\}. \tag{13}$$

Constraint (3) states that construction costs cannot exceed the budget, $C_{\max}$. Constraints (4) ensure that an edge is included in the RN only if its endnodes are also selected. Constraints (5) allow the constructed edges to be used in both directions. Constraints (6) to (10) are flow conservation constraints for variables f. Constraints (11) make $x_a = 1$ if a demand pair assigned to the RN uses arc a. Note that, without those constraints, variables x_a would not be forced to be 1 if a passenger is to use the corresponding link in the railway network, and nor would variables y_i. Constraints (12) force variables r_w to be zero if the travel time of the demand pair w equals u_w^{COM} (note the importance of parameter ε to break ties). Note that variable u_w, the travel time of passengers going from w^s to w^t, is

$$u_w = \begin{cases} \sum_{a \in A} d_a f_a^w, & \text{if } r_w = 1 \quad \text{(the time via the RN)} \\ \\ u_w^{COM}, & \text{if } r_w = 0 \quad \text{(the time via the complementary mode)} \end{cases}$$

Note that, in case the travel time by the railway is less than the travel time by the complementary mode, we automatically have that $r_w = 1$, because of the objective function (2). Also note that, if the objective function varies, we may have to complement constraints (12) by adding a constraint that makes $r_w = 1$ if the travel times via the RN are shorter than the complementary mode u^{COM}, like:

$$-u_w + u_w^{COM} - Mr_w \leq 0, \ w \in W.$$

For computational purposes, we have added the following cut to our model:

$$f_a^w \leq r_w, \ \forall \ w \in W, \ a \in A.$$

This cut showed a significant improvement in the computational times.

The total number of variables and constraints of this ILP problem is $O(m\nu)$ and $O(n\nu + m\nu)$, respectively. This model can be used to optimize other utility functions, such as total travel time or construction costs, by slightly changing it.

Example 2.2 *The solution to the problem in Example 2.1 is shown in Figure 2. This network maximizes the trip coverage, that is, maximizes the number of passengers that find the RN more attractive than the complementary mode in terms of travel time. In this example* $z_{nf} = 1249$. *Note that nodes 7 and 9 remain isolated. This is due to the budget constraint.*

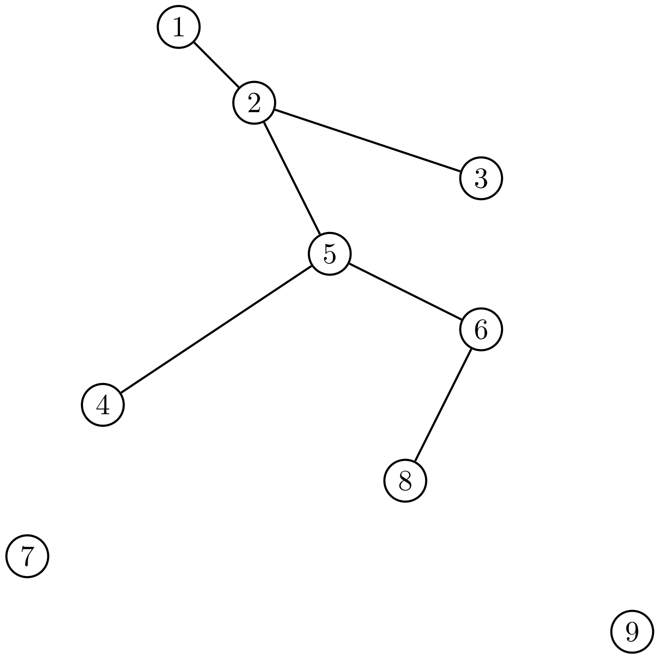

Figure 2: Test Network.

Although the previous example was solved to optimality in around 2 seconds, such a computational time may grow to to 900 seconds when it comes to a 15-node toy example, and it is untractable for realistic sizes (up to 100 possible stations and 200 possible links). In [8] it is proven that the RND is NP-hard. Therefore the search for heuristics to find (good) feasible solutions to real instances in an affordable amount of time applies. In the next section we present the heuristic chosen in this paper: a GRASP algorithm.

2.1 GRASP for RND problems

GRASP is a methaheuristic algorithm applied to large combinatorial problems, which consists of randomly adding elements to the problem's solution set out of the set of $k \in \mathbb{N}$ elements that individually yield the largest improvement in the objective function when added to the previous solution. This procedure is repeated, and each of the (possibly) different obtained solutions form a set of feasible solutions. The final solution chosen by GRASP is the best out of the feasible solution set previously obtained.

It is common to try to improve the solutions generated by GRASP by means of a local search procedure, in which the neighborhood of each such feasible solution is explored.

Our GRASP algorithm is divided into two phases, which are repeated $i_{\max}$ times. At each iteration i, we first construct a feasible solution in the *construction phase*. Such a phase begins with a randomly chosen edge $e_0 \in E$. In the next step, we find the k edges such that when they are individually added to e_0 the trip coverage is maximized and the cost of the resulting network does not exceed $C_{\max}$. Out of those k edges, we randomly pick one, which will be added to e_0. Next we add one more edge to the previously chosen two edges in the same manner. We keep doing this until no more edges can be added without violating the budget constraint. Let FN^i be the network obtained. After this, the *improvement phase* begins. Such a phase, which is a local search algorithm, aims at finding a feasible network in the neighborhood of FN^i with a higher trip coverage. To do this, pick one edge of FN^i, say $\widetilde{e}$, and remove it. Choose the edge not in FN^i such that when it is added to $FN^i - \{\widetilde{e}\}$ the trip coverage of the resulting network is maximized. Keep doing this for all edges in FN^i. Then, pick the resulting network with the highest trip coverage, say FN'. If such trip coverage exceeds that of FN, set $FN^i = FN'$. These two phases are repeated $i_{\max}$ times. A flow chart for this GRASP algorithm is depicted in Figure 3. We now give pseudocodes for both phases, for which the following notation is needed:

Let E be a set of edges, let $N(E)$ be the set of endnodes of the edges in E, let $k \in \mathbb{N}$ be the number of possible edges to be included in each iteration, and let $i_{\max}$ be the maximum number of times that the construction phase and the improvement phase will be repeated. For each t, let FN_t be the set of edges that constitute the feasible network, not necessarily optimal, obtained in the t^{th} iteration of the construction phase. In this iteration t, edge $e_t \in E$ is said to be feasible if $e_t \notin FN_{t-1}$ and the construction cost of network $G_t(N(FN_{t-1} \cup \{e_t\}), FN_{t-1} \cup \{e_t\})$ is not larger than $C_{\max}$. Let FE_t denote the set of

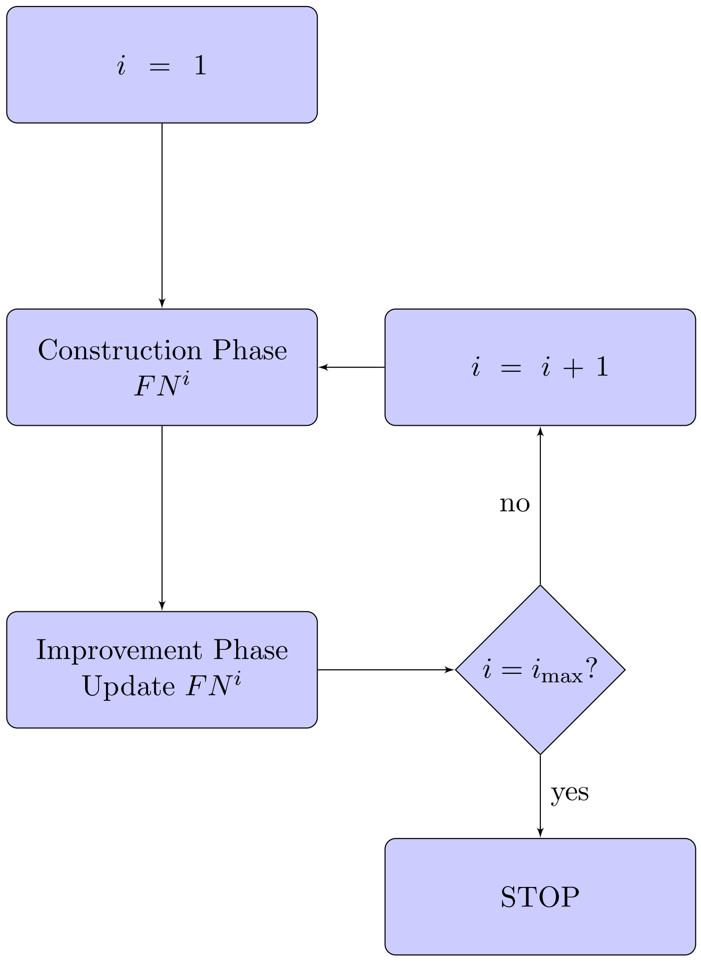

Figure 3: Flow chart for our GRASP algorithm

feasible edges at iteration t.

The construction phase of our GRASP algorithm builds a feasible solution as follows:

CONSTRUCTION PHASE$(E, G, u^{COM}, d, C_{\max}, k)$:

Randomly choose a feasible edge $e_0 \in E$.

1. Initialize $FN_0 = \{e_0\}$, $t = 0$.
2. Set $t = t + 1$ and determine the set FE_t. If $|FE_t| = 0$, go to step 5.
3. If $|FE_t| > k$, determine a set $SE_t^k = \{e_{t_1}, \ldots, e_{t_k}\}$ consisting of the k edges of FE_t that generate the largest trip coverage when they are individually added to FN_{t-1}. Otherwise, that is if $|FE_t| \le k$, set $SE_t^k = FE_t$.
4. Randomly choose one edge $e_t \in SE_t^k$. Set $FN_t = FN_{t-1} \cup \{e_t\}$ and go to 2.

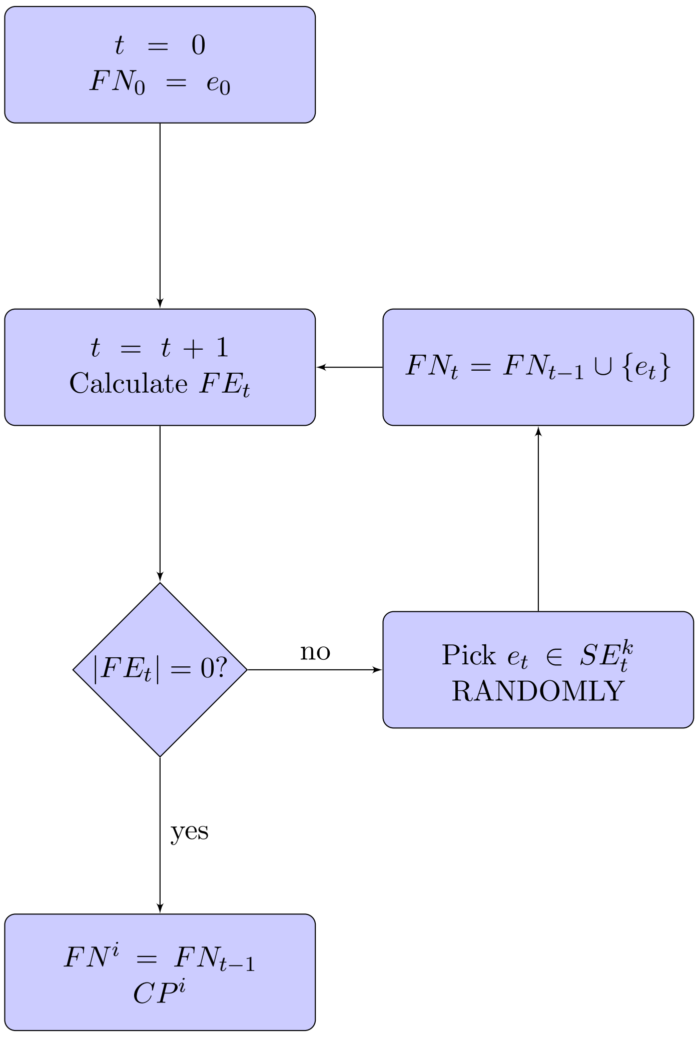

Figure 4: Flow chart for the construction phase

5. $FN^i = FN_{t-2} \cup \{e^*_{t-2}\}$ is the final solution in this iteration, where $\{e^*_{t-2}\}$ is the edge in SE^k_{t-2} that generates the largest trip coverage when it is individually added to FN_{t-2}. Let CP^i be the trip coverage of network FN^i.

END CONSTRUCTION PHASE

We note that, from step 5, this algorithm chooses its final edge in a greedy manner, regardless the value of k, since it would be useless to pick in this final step the second, or third, or forth, ..., or k-th best edge. In Figure 4 a flow chart summarizes the construction phase. After a feasible solution FN^i has been found, an improvement phase is designed to (try to) find a better solution in the neighborhood of FN^i.

```
IMPROVEMENT PHASE($E, G, u^{COM}, d, C_{\max}, FN^i = \{e^i_1, ..., e^i_q\}$)
Define $\overline{FN^i}$ as $E \setminus FN^i$.
For $j$ from 1 to $q$ do:
    1. Let $FN^i(j) = FN^i \setminus \{e^i_j\}$.
    2. Let $\widetilde{e} \in \overline{FN^i}$ the edge that gives the largest improvement in the objective function when
       added to $FN^i(j)$ among all edges in $\overline{FN^i}$ that satisfy the budget constraint. If there is
       no such $\widetilde{e}$, store $FN^i(j)$, set $j = j+1$ and go to 1. Otherwise, set $FN^i(j) = FN^i(j) \cup \{\widetilde{e}\}$
       and go to 2.
End For
Let $j^*$ be such that $FN^i(j^*)$ is not worse than $FN^i(j)$, $\forall\ j \neq j^*$. If $FN^i(j^*)$ improves
$FN^i$, then we set $FN^i = FN^i(j^*)$. Otherwise we remain with $FN^i$. Let $IP^i$ be the trip
coverage of network $FN^i$.
END IMPROVEMENT PHASE
```

In Figure 5 a flow chart summarizes the improvement phase. Both phases are repeated $i_{\max}$ times. Let $i^* = \arg\max_i IP^i$. The final solution is FN^{i^*}. One of the objectives in our computational experiments will be to check whether the improvement phase actually improves the construction phase or not.

2.2 Robust railway network design

In this section we model the probabilistic version of the RND problem, in which link failures can occur (named probabilistic railway network design, PRND). The deterministic RND problem yields a network that optimizes a certain utility function when no failures occur. We now assume that in the event of link disruption, an alternative transportation mode is provided to passengers, e.g., a bus service, which transports passengers of the RN between the stations adjacent to the failing link. This generates extra cost for the network operator and extra travel time for passengers.

In order to write PRND problem as a mixed integer linear programming problem, we need to add the following parameters (plus those defined for the RND problem):

1. α_a is the time needed to go from the railway station to the departure point of the alternative mode at station a^s plus the time needed to go from the arrival point of the alternative mode at station a^t, $\forall\ a \in A$.

2. γ_e is the probability of edge $e \in E$ to fail.

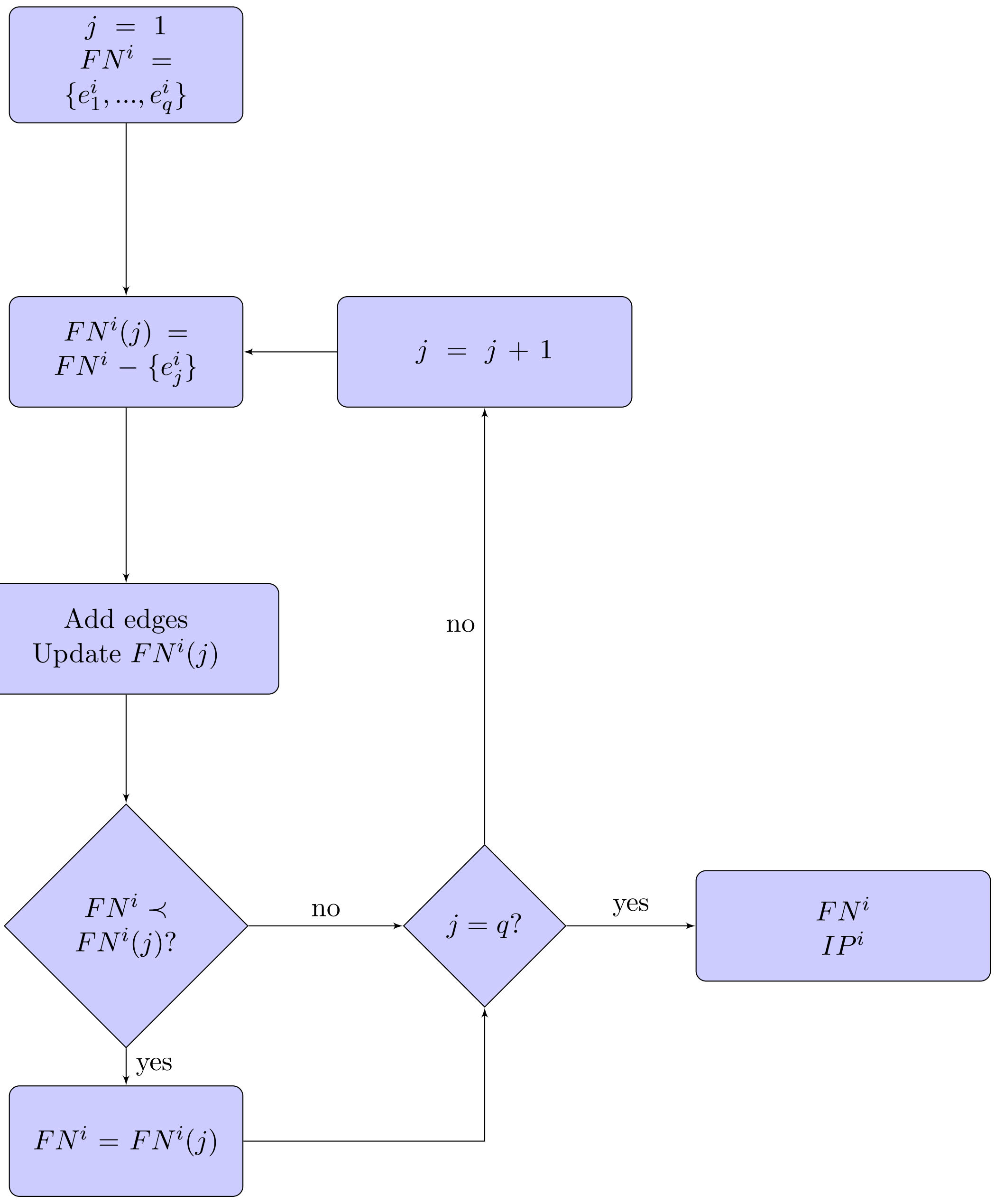

Figure 5: Flow chart for the improvement phase

We also need to add the following variables:

- $\phi_a^w(e) = 1$ if demand $w \in W$ uses arc $a \in A$ when link $e \in E$ fails, and zero otherwise.
- $\rho_w(e) = 1$ if demand w is allocated to the RN when link $e \in E$ fails, and zero otherwise.
- Variable $z_f(e)$ denotes the demand captured by the RN if edge e fails. Note that $z_f(e) = \sum_{w \in W} \rho_w(e) g_w$.

The objective function of the ILP model that describes PRND problem is:

$$(1 - \sum_{e \in E} \gamma_e) z_{nf} + \sum_{e \in E} \gamma_e z_f(e). \tag{14}$$

Note the equivalence between (1) and (14) when link failures are not considered in the latter, that is, when $\gamma_e = 0 \;\forall\; e \in E$. Note as well that, if edge e does not belong to the RN, then $z_f(e) = z_{nf}$. Therefore, (14) is equivalent to

$$(1 - \sum_{e \in RN} \gamma_e) z_{nf} + \sum_{e \in RN} \gamma_e z_f(e),$$

where the notation $e \in RN$ means that edge e is part of the network RN.

In addition to constraints (3) - (12), our PRND model also includes the following constraints:

- Routing demand conservation constraints when edge e fails

$$\sum_{a \in A: a^t = w^s} \phi_a^w(e) = 0,\; w \in W,\; e \in E,$$
$$\sum_{a \in A: a^s = w^s} \phi_a^w(e) = \rho_w(e),\; w \in W,\; e \in E,$$
$$\sum_{a \in A: a^t = w^t} \phi_a^w(e) = \rho_w(e),\; w \in W,\; e \in E,$$
$$\sum_{a \in A: a^s = w^t} \phi_a^w(e) = 0,\; w \in W,\; e \in E,$$
$$\sum_{a \in A: a^t = k} \phi_a^w(e) - \sum_{a \in A: a^s = k} \phi_a^w(e) \;=\; 0,\; k \notin \{w^s, w^t\},\; w \in W,\; e \in E.$$

- Location-Allocation constraints when edge e fails

$$\phi_a^w(e) + \rho_w(e) - 1 \;\leq\; x_a,\; a \in A,\; w \in W,\; e \in E.$$

- Splitting demand constraints when edge e fails

$$\varepsilon + \upsilon_w(e) - u_w^{COM} - M\,(1-\rho_w(e)) \leq 0,\ w \in W, e \in E,$$

  where
$$\begin{aligned}\upsilon_w(e) &= \textstyle\sum_{a\in A\setminus\{(e^s,e^t),(e^t,e^s)\}} d_a\phi_a^w(e) + (u_e^{COM}+\alpha_e)\phi_e^w(e)\\ &+(u_{e'}^{COM}+\alpha_{e'})\phi_{e'}^w(e) + u_w^{COM}(1-\rho_w(e)),\end{aligned}$$

  M is a large enough real number and $\varepsilon > 0$ is a small tolerance.

- Binary constraints

$$\phi_a^w(e),\ \rho_w(e) \in \{0,1\}.$$

These constraints have an analogous interpretation to that of constraints (3) - (12), with the only difference that now we consider that edge $e \in E$ fails, and therefore the necessary time to traverse it changes from d_e to $u_e^{COM}+\alpha_e$ (the necessary time to traverse the edge by the complementary mode plus the necessary time to change from one mode to another), and from $d_{e'}$ to $u_{e'}^{COM}+\alpha_{e'}$. Note that $\upsilon_w(e)$ is

$$\begin{cases}\displaystyle\sum_{a\in A\setminus\{e,e'\}} d_a\phi_a^w(e) + (u_e^{COM}+\alpha_e)\phi_e^w(e) + (u_{e'}^{COM}+\alpha_{e'})\phi_{e'}^w(e) & \text{if } \rho_w(e)=1\\ u_w^{COM} & \text{if } \rho_w(e)=0.\end{cases}$$

Now the number of variables and constraints increases to $O(\nu m^2)$ and $O(m\nu n + m^2\nu)$, respectively. The complexity of the PRND problem (much higher than the already complex RND) makes it necessary for us to propose heuristics for its resolution.

For computational purposes, we have added the same cut as in the RND problem plus

$$\phi_a^w(e) \leq \rho_w(e),\ \forall\ w\in W,\ a\in A,\ e\in E.$$

This cut showed a significant improvement in the computational times.

An adaptation of the previously presented GRASP algorithm to problem PRND is trivial just by changing the objective function in the algorithm presented in Section 2.1.

3 Experiments

In order to test the validity of our GRASP algorithms, we performed some computational experiments. 10 random instances were generated for different values of n, $n \in \{10, 12, 13, 14, 15, 16, 18\}$, having this way 70 different instances. For each value of n, a grid with n cells was built.

Features			CPLEX		[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Name	n	m	Value	Time	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
10_01	10	18	1271	20,201	0	0,0	0	0			0	0			0	0	
10_02	10	18	1561	6,187	0	0,0	0	0			0	0			0	0	
10_03	10	16	1609	4,328	0	0,0	0	0			0	0			0	0	
10_04	10	16	1193	7,874	0	0,0	0	0			0	0			0	0	
10_05	10	17	1191	14,390	0	0,0	0	0			0	0			0	0	
10_06	10	17	1322	19,351	0	0,0	0	0			0	0			0	0	
10_07	10	17	1624	5,438	0	0,0	0	0			0	0			0	0	
10_08	10	18	1328	29,342	0	0,0	0	0			0	0			0	0	
10_09	10	15	1254	4,000	0	0,0	0	0			0	0			0	0	
10_10	10	17	1336	23,577	0	0,0	0	0			0	0			0	0	
12_01	12	25	2112	82,502	0	0,0	0	0			0	0			0	0	
12_02	12	25	2026	58,625	0	0,0	0	0			0	0			0	0	
12_03	12	24	1842	18,391	0	0,0	0	0			0	0			0	0	
12_04	12	24	2032	141,454	0	0,0	0	0			0	0			0	0	
12_05	12	22	2095	49,313	0	0,0	0	0			0	0			0	0	
12_06	12	22	1943	12,579	0	0,0	0	0			0	0			0	0	
12_07	12	25	1981	136,909	0	0,0	0	0			0	0			0	0	
12_08	12	23	1892	29,016	0	0,0	0	0			0	0			0	0	
12_09	12	22	2111	24,532	0	0,0	0	0			0	0			0	0	
12_10	12	24	1973	40,985	0	0,0	0	0			0	0			0	0	
13_01	13	25	2423	41,349	0	0,0	0	0			0	0			0	0	
13_02	13	27	2106	104,499	0	0,0	0	0			0	0			0	0	
13_03	13	23	2419	7,172	0	0,0	0	0			0	0			0	0	
13_04	13	28	2250	161,570	0	0,0	0	0			0	0			0	0	
13_05	13	26	2331	58,535	0	0,0	0	0			0	0			0	0	
13_06	13	25	2248	97,580	0	0,0	0	0			0	0			0	0	
13_07	13	25	2199	86,094	0	0,0	0	0			0	0			0	0	
13_08	13	28	2390	95,230	0	0,0	0	0			0	0			0	0	
13_09	13	30	2654	98,967	0	0,0	0	0			0	0			0	0	
13_10	13	24	2322	119,111	0	0,0	0	0			0	0			0	0	
14_01	14	27	2882	120,033	0	0,0	0	0			0	0			0	0	
14_02	14	24	2493	274,369	0	0,0	0	0			0	0			0	0	
14_03	14	24	2417	112,547	0	0,0	0	0			0	0			0	0	
14_04	14	24	2729	388,268	0	0,0	0	0			0	0			0	0	
14_05	14	26	2667	302,376	0	0,0	0	0			0	0			0	0	
14_06	14	24	2603	248,285	0	0,0	0	0			0	0			0	0	
14_07	14	22	2544	13,139	0	0,0	0	0			0	0			0	0	
14_08	14	22	2601	28,123	0	0,0	0	0			0	0			0	0	
14_09	14	21	2527	10,499	0	0,0	0	0			0	0			0	0	
14_10	14	28	2562	463,097	0	0,0	0	0			0	0			0	0	
15_01	15	31	3201	103,909	0	0,0	0	0			0	0			0	0	
15_02	15	31	3155	155,509	0	0,0	0	0			0	0			0	0	
15_03	15	32	3353	419,087	0	0,0	0	0			0	0			0	0	
15_04	15	29	3276	65,576	0	0,0	0	0			0	0			0	0	
15_05	15	30	2559	55,045	0	0,0	0	0			0	0			0	0	
15_06	15	30	3001	259,339	0	0,0	0	0			0	0			0	0	
15_07	15	28	3045	305,198	0	0,0	0	0			0	0			0	0	
15_08	15	31	2887	214,126	0	0,0	0	0			0	0			0	0	
15_09	15	31	2979	338,286	0	0,0	0	0			0	0			0	0	
15_10	15	26	2341	123,846	0	0,0	0	0			0	0			0	0	
16_01	16	31	3010	93,701	0	0,0	0	0			0	0			0	0	
16_02	16	37	3664	868,559	0	0,0	0	0			0	0			0	0	
16_03	16	31	3739	246,428	0	0,0	0	0			0	0			0	0	
16_04	16	33	3466	918,887	0	0,0	0	0			0	0			0	0	
16_05	16	31	3703	365,239	0	0,0	0	0			0	0			0	0	
16_06	16	33	3561	361,553	0	0,0	0	0			0	0			0	0	
16_07	16	34	3437	472,051	0	0,0	0	0			0	0			0	0	
16_08	16	33	3515	165,305	0	0,0	0	0			0	0			0	0	
16_09	16	30	3145	504,001	0	0,0	0	0			0	0			0	0	
16_10	16	35	3260	563,560	0	0,0	0	0			0	0			0	0	
18_01	18	37	4197	893,785	0	0,0	0	0			0	0			0	0	
18_02	18	38	5000	973,639	0	0,0	0	0			0	0			0	0	
18_03	18	37	4217	1732,401	0	0,0	0	0			0	0			0	0	
18_04	18	40	4554	2827,983	0	0,0	0	0			0	0			0	0	

Figure 6: Table of results.

Depending on the value of n, different types of grids were built. Other than $n = 13$, all grids could be built in the $n_1 \times n_2$ form, with $n_1 n_2 = n$. The different types are shown in Table 1. Once the cells were drawn, each potential station i was randomly located in the i^{th} cell. The set E was generated as follows. There is a possible link joining a node i with its neighbor node j with probability 0.8 (the probability of joining not-neighboring nodes was set to zero). We consider that two cells are neighbors if their borders share at least one point. Therefore, given a cell, its neighbors are in the N, NE, E, SE, S, SW, S, NW cells (where N, S, E and W stand for North, South, East and West, respectively), provided that such cells exist. Two nodes are neighbors if the cells they are into are neighbors.

n	10	12	13	14	15	16	18
Type of grid	5×2	6×2	$(6 \times 2) + 1$	7×2	3×5	4×4	3×6

Table 1: Different types of grids for different values of n.

We considered all $n(n-1)$ possible O/D pairs. The time needed to traverse the generated links by the RN (d_a) was set proportional to their Euclidean lengths, and the time by the complementary mode, (u_a^{COM}) was set to twice d_a. The cost for building station i was set randomly in the interval $[2, 4]$, and the cost for building link e was set equal to its Euclidean distance. The number of passengers going from the potential station w^s to the potential station w^t (g_w) was randomly generated in the interval $[5, 50]$. One instance with $n = 9$ of this is Example 2.1. In Table 6, the first and the second column denote the number of nodes and edges of each instance, respectively.

3.1 Solving instances to optimality

Our experiments begun with solving each instance to optimality by using GAMS 23 and CPLEX in an Intel(R) Core(TM)2 CPU computer, with 2.13 GHz and 1.97 GB of RAM memory. The optimal solution to each instance and the necessary time for CPLEX to find and guarantee such a value are shown in columns *Opt value* and *CPLEX time*, respectively, of the table in Figure 6. We checked the correlation between the computational time needed by CPLEX and the number of nodes and edges (individually) in four different models: linear, logarithmic, potential and exponential, by means of Microsoft EXCEL. The correlation coefficient R^2 of each model and each independent variable (nodes or edges) are shown in Table 2. This showed that the variable that best explains the computational times to find an optimal solution by CPLEX is the number of edges, and that such a computational times exponentially depends on the number of edges. More specifically, if y denotes the computational time and x denotes the number of edges, we have that the corresponding tendency curve is $y = 0.191e^{(0.2366x)}$. This means that, for a 50 edge instance the estimated time needed by CPLEX would be 0.3 days, for a 60 edge instance this time would be 3.2 days and

Model	Linear	Logarithmic	Potential	Exponential
R^2, Nodes	0.3995	0.3460	0.7400	0.7534
R^2, Edges	0.3706	0.3090	0.7857	0.7955

Table 2: Values of R^2 for the different models and different factors.

n	$k = 1$	$k = 2$	$k = 3$	$k = 4$
Average improvement	2.5	5.5	8.7	10.9

Table 3: Actual improvement of the improvement phase with respect to the construction phase in RND problems.

for a 80 edge instance this time would grow to 367.2 days. This simple analysis, together with the theoretical proof that this model is NP-hard, supports the need of heuristics for this problem.

3.2 Solving instances with GRASP

In order to check whether the improvement phase actually improved the construction phase, in previous experiments we also compared the performance of the algorithm in both phases. Table 3 shows the average improvement for each problem size, which was calculated as $100 \times \sum_{i=1}^{i_{\max}} (\frac{IP_i}{CP_i} - 1)$; CP_i and IP_i being the values of after construction phase and after the improvement phase of each iteration, respectively. Note that the improvement is significant. From these results, we concluded that the improvement phase gives significant extra value to the GRASP algorithm.

AHORA VENDRÍAN LOS EXPERIMENTOS CON LAS INSTANCIAS NUEVAS EN FORTRAN!

3.3 GRASP in PRND problems

ESTO HAY QUE ACTUALIZARLO!! We also performed our experiments for PRND problems, where the probability of failure of each edge $(s, t) \in E$ was set proportional to their length, $\gamma_e = d_e/(16 \sum_{e \in E} d_e)$. Due to its extremely high complexity, only the instances for $n \in \{6, 9, 10\}$ were solved to optimality with CPLEX, where our GRASP algorithms obtained the optimal solution in all instances. On the other hand, the Greedy algorithm failed in finding the optimal solution in more than 80% of the instances, and the average objective function value achieved by the Greedy solution was around 90%.

Regarding computational times, see Table 4, the GRASP algorithms found the optimal solutions in a much more reasonable amount of time. Note that times diminish when k passes from 3 to 4. **This may be due to the fact that when $k = 4$ the part of solution space**

n	CPLEX	$k=1$	$k=2$		$k=3$		$k=4$	
	t	t	i^*	t	i^*	t	i^*	t
6	10	0,1	10	0,7	3	0,4	5	0,5
9	8952	1,1	45	53	25	32	36	44
10	29033	1,4	65	36	58	117	36	73
Average	12665	0,9	40	30	29	50	26	39

Table 4: Number of iterations and running times in seconds of each algorithm in PRND problems.

n	k=1	k = 2	k = 3	k = 4
6	3	9	10	19
9	2	5	6	10
10	2	5	7	15
Average	2,7	6,3	7,7	14,7

Table 5: Actual improvement of the improvement phase with respect to the construction phase in PRND problems.

explored is wider than when $k = 3$, and therefore sometimes GRASP finds the optimal solution sooner.

We also compared the performance of the algorithm in its construction phase and in its improvement phase. Table 5 shows the average improvement for each problem size.

The results obtained make us conclude that, for the PRND problem,

1. the Greedy algorithm is not accurate, even for small size instances.
2. our GRASP algorithms for $k \in \{2, 3, 4\}$ seem to be a good alternative to find good feasible solutions in a reasonable amount of time.
3. the improvement phase proposed actually improves the construction phase of the GRASP algorithm by 6% to 14%, on average.

Comparing both problems, note that the average computational times to calculate the optimal solution to PRND problems by CPLEX were 1170 times those needed to calculate the optimal solution to RND problems. In the GRASP algorithms those times also increased, but only by a factor between 13 and 19, see Table 6.

4 Conclusions

AMPLIAR LAS CONCLUSIONES!!! In this paper we have introduced a new model for the railway network design problem, and a model for the same problem assuming that

Algorithm	CPLEX	$k = 1$	$k = 2$	$k = 3$	$k = 4$
RND time	10,82	0,07	30	50	39
PRND time	12665	0,9	1,67	2,67	2,67
Increase factor	1170	13	18	19	15

Table 6: Average computation time for RND and PRND problems, for $n \in \{6, 9, 10\}$

links can fail. Due to their NP-hardness, it is necessary to find approximation algorithms that provide "good" feasible solutions in a reasonable amount of time. The algorithms proposed in this paper are GRASP algorithms, which proved to be a **good and fast alternative for our** railway network design problems. Computational experiments showed that, in a reasonable CPU time, GRASP algorithms find good feasible network designs, in many cases even the optimal one.

acknowledgements

We would like to thank the Spanish *Ministerio de Fomento* under grant PT-2007-003, *Ministerio de Ciencia y Educación* under grant MTM2009-14243 and the *Junta de Andalucía* under grant P09-TEP-5022 for supporting this research. **Special thanks are due to two anonymous referees for their valuable comments.**

References

[1] Baaj, M., Mahmassani, H.: An ai-based approach for transit route system planning and design. Journal of Advanced Transportation **25**(2), 187–210 (1991)

[2] Cancela, H., Robledo, F., Rubino, G.: A grasp algorithm with tree based local search for designing a survivable wide area network backbone. Journal of Computer Science & Technology **4**(1), 52–58 (2004)

[3] Díaz, J.A., Luna, D., Luna, R.: A grasp heuristic for the manufacturing cell formation problem. TOP **In Press** (2011)

[4] Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters **8**, 67–71 (1989)

[5] Goossens, J., van Hoesel, C., Kroon, L.: A branch-and-cut approach for solving railway line-planning problems. Transportation Science **38**, 379–393 (2004)

[6] Institute of Electrical and Electronics Engineers: *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries* (1990)

[7] Laporte, G., Marín, A., Mesa, J., Perea, F.: Designing robust rapid transit networks with alternative routes. Journal of Advanced Transportation **45**, 54–65 (2011)

[8] Laporte, G., Mesa, J., Perea, F.: A game theoretic framework for the robust railway transit network design problem. Transportation Research Part B **44**, 447–459 (2010)

[9] Marín, A., García-Ródenas, R.: Location of infrastructure in urban railway networks. Computers and Operations Research **36**, 1461–1477 (2009)

[10] Marín, A., Jaramillo, P.: Urban rapid transit network design: accelerated Benders decomposition. Annals of Operations Research **169**(1), 35–53 (2009)

[11] Marín, A., Mesa, J.A., Perea, F.: Integrating robust railway network design and line planning under failures. Lecture Notes in Computer Science **5868**, 273–292 (2009)

[12] Mauttone, A., Urquhart, M.E.: A route set construction algorithm for the transit network design problem. Computers & Operations Research **36**, 2440–2449 (2009)

[13] Murphey, R., Pardalos, P., Pitsoulis, L.: A GRASP for the multitarget multisensor tracking problem. In: Networks, Discrete Mathematics and Theoretical Computer Science Series, vol. 40, pp. 277–302 (1998). American Mathematical Society

[14] Nesmachnow, S., Cancela, H., Alba, E.: Evolutionary algorithms applied to reliable communication network design. Engineering Optimization **39**(7), 831–855 (2007)

[15] Schöbel, A., Scholl, S.: Line planning with minimal transfers. In 5th Workshop on Algorithmic Methods and Models for Optimization of Railways, Number 06901 in Dagstuhl Seminar Proceedings, 2006