An evolutionary and constructive approach to a crew scheduling problem in underground passenger transport

Rafael Elizondo[1], Victor Parada[1], Lorena Pradenas[2], Christian Artigues[3,4]

[1]Department of Informatic Engineering, University of Santiago of Chile, 3659 Ecuador Av. Santiago, Chile
[2]Department of Industrial Engineering, University of Concepción, Edmundo Larenas S/N, Concepción, Chile
[3]CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[4]Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France

Abstract

Operation management of underground passenger transport systems is associated with combinatorial optimization problems (known as crew and train scheduling and rostering) which belong to the np-hard class of problems. Therefore, their resolution in real situations is generally addressed using heuristic methods. This paper considers the duty generation problem, which consists of identifying an optimal trips set that the conductors should complete in a labor day. With regard to the operational and labor conditions, the trains should be driven with the lowest number of conductors and a minimized total idle time between trips. The problem is modeled and solved using a constructive hybrid approach, which has the advantage of visualizing a solution construction similar to the approach typically used by operators who manually solve the problem. This approach takes advantage of the benefits offered by evolutionary methods, which hardly store a candidate solutions population in each stage, controlling in this way the combinatorial explosion of possible solutions. The results that we obtained for problems with similar characteristics to those that are performed manually in the Santiago Metro System were compared with two alternative approaches based on tabu search and a greedy method. The hybrid method

produced similar results to those generated by the tabu search, and both found better results than the greedy method.

Keywords: Graph search methods, Crew Scheduling Problem.

INTRODUCTION

The conductor's activities' planning for passenger vehicles presents a complex decision-making scenario. Operations managers face the tough task of finding a feasible solution from the large number of possible combinations that should be considered. This type of operations is also related to combinatorial optimization and np-hardness, and consequently the search for efficient methods providing continuous resolution continues to be a challenge. Once efficient algorithmic methods are found, computational tools can be built which will allow managers to make rapid decisions with flexibility and efficacy. Additionally, they will be able to simulate new demand scenarios, new facilities, and changes in labor conditions.

Crew scheduling problems emerge in the management of cargo and passenger transport, such as airlines, buses, and trains. These kind of problems typically have been approached by reduction to three main steps (Caprara, Monaci and Toth, 2001): train movement programming, duty generation, and conductor service assignment (rostering). A complete operational programming is generated when these three steps are solved.

The first step consists of timetable generation, which specifies train movement between the start and finish locations for each trip. This definition should consider the demand for trips that exist between the different network origins and destinations, the facilities available, and an adequate balance between service quality and operating cost. The result will be the generation of a set of trips that should be performed during the day. Each trip is defined by the train number, start time and location, and the finish time and location.

Once the timetable is generated, the second step is to generate conductor duties, which are the feasible subsets of trips that can be performed by a single conductor. In the

third step, each generated duty is associated to a conductor, considering homogenous rotation of the conductors to the duties in a longer programming period as well as additional activities such as: free days, vacations, weekends, training periods, etc. The final result is weekly or monthly programming.

Since the seventies, the crew scheduling problem has been widely studied for the case of airlines. However, in the case of surface passenger transport, the majority of the studies have been concentrated in the last few years (Ernst, Jiang, Krishnamoorthy, Owens and Sier, 2004). Commonly, the study of these problems considers the three previously described steps separately. Meanwhile, attempts have been made to address the problem using a single model and consequently, simultaneously solving it using a single method. In this line of work, Freling, Wagelmans & Paixao (1999) and Freling et al. (2003) analyzed the benefits of the integrated versus the independent perspective for each problem. Although their proposal is applicable to real problems, leading towards a decision-making support system (Freling, Lentink and Wagelmans, 2004), the partial benefit that is obtained appears dependent on the flexibility to change the vehicles on duty. The authors have used the embedded column generation method in a branch-and-bound method to generate exact solutions (Friberg and Haase, 1999). However, the combinatorial degree of these problems requires the use of approximation in order to solve real large-size problems (Haase, Desaulniers and Desrosiers, 2001). The partial improvements of the integrated perspective versus independent perspective has also been verified by other authors: Gaffi & Nonato (1999) studied the problem that emerged in public transport using a heuristic method based on lagrangean relaxation.

Several computational tools for the crew scheduling problem in large train systems have been developed. The Italian (Caprara, Fischetti, Toth, Vigo and Guida, 1997; 1999)

and Australian (Ernst, Jiang, Krishnamoorthy, Nott and Sier, 2001) railway systems have been addressed using binary programming. In the first case, the model also considered vehicle assignment using branch-and-cut as a solution method. In the second one, large size problems have been addressed incorporating rostering for each depot existing in the network. Heuristic column generation-based procedures have been proposed to solve the crew pairing problem in the large German railway system(Bengtsson, Galia, Gustafson, Hjorring and Kohl, 2004) The Portuguese and Dutch train systems have been studied with an interactive perspective based on artificial intelligence techniques (Morgado and Martins, 1992; 1998). The user can propose a solution or modify a generated solution following a set of heuristic rules. To model the transport systems of Singapore and Hong Kong, an integration between graphic problem resolution and heuristic methods was used (Chew, Pang, Liu, Ou and Teo, 2001).

The problem has also been studied for mass transit and buses. For the case of buses, Lourenco et al. l (2001) and Dias, de Sousa, & Cunha (2002) developed decision support systems based in genetic and tabu search algorithms.

The real situation presented by subterranean metro train transport has unique characteristics that make it different from urban surface transport systems and from long-distance cargo-and-passenger train systems. For example, in the subterranean transport of passengers, the exchanges of conductors is established in a way that they can only be carried out in some predefined stations and adequately implemented to receive the conductors that should wait there to carry out their next work or their meal break. In general, in the surface transport of passengers, the change of operators can be accomplished at any point on the route, thus in the transport of passengers by bus lines, it is possible to assign a unit operator-duty-bus which includes in addition to the trips, the reception and

delivery of the vehicle at the parking depot, as also for the suspensions of service for the resupply of fuel. This type of assignment is not possible with subterranean passenger trains. From an algorithmic point of view, the underground passenger transport problem has been little explored. In a particular case, Cavique et al. (1999) propose the use of a heuristic search to minimize the number of necessary duties for a determined planning period in the Lisbon underground. They describe a constructive algorithm that, with less computer time, obtains slightly worse solutions than those generated manually for real-size problems; however, the two most elaborated methods based on tabu search find better results in terms of solution quality but require a higher computer time.

In the situation above, as in many other problems in combinatorial optimization, constructive methods have been used principally to generate initial solutions, and in a second stage the search is performed with another method. The advantage of these methods lies in the speed of solution generation, which is possible because they perform an important pruning in the space of possible combinations as they construct the solution.

A constructive algorithm for a combinatorial optimization problem can be viewed as a graph search process in problem-solving (Russell and Norvig, 2002). In such an area, a graph representing all possible problem states (nodes) is used to systematically explore the search space. Several methods have been developed to find a goal when starting from an initial node (Pearl, 1984), looking to reduce the number of visited nodes by pruning part of the search space. If the domain that contains all the possible solutions of a combinatorial optimization problem is represented by a "space state graph" (Pearl, 1984), then the "search space methods" may be used to navigate the domain of the solutions of the optimization problem. Additionally, metaheuristic methods have been widely used in the last few years to solve many combinatorial optimization problems.

In this paper, we combine the two frameworks mentioned above generating a hybrid algorithm that gradually builds and stores only part of the total possible combinations in order to solve the crew scheduling problem (CSP) that emerges during the second step of the passenger transport operational programming. In short, trips are gradually combined to build duties, and simultaneously duties are combined to produce a complete crew schedule.

In Section 1 the problem is presented and in Section 2, three different approaches to solve the problem are described, while in Section 3 the results attained for real-size problems associated with the Santiago Metro are presented. Finally, the main conclusions and comments of the study are presented.

1. DESCRIPTION OF THE PROBLEM

The CSP considers the following characteristics in this study:

- Relief facilities: Some of the stations in the network have the installations required by conductors who are waiting for their next trip, resting, or having a meal break. Typically, these are the last stations on each line; however, there can be intermediate stations with relief facilities.
- Maximum conduction time. Each duty must respect the limiting value for the total amount of time that each conductor can drive a train each day excluding breaks.
- Continuous conduction time. Each duty must consider a maximum value of continuous conduction time, i.e. the maximum amount of time a conductor can drive without stepping down from the train.

- Meal breaks. Each duty should contain a predefined time interval for meals. When considering all the duties, the meal breaks should be homogenously distributed in the duties.
- Work shifts. In order to cover the service time of a work day, several work shifts are required. These shifts are covered by a set of full-time and part-time conductors. Full-time conductors should perform their duties until a certain time, after which the activities should be performed by part-time conductors. In this way, a set of duties should be generated for each group of workers, considering each shift's start time as problem data.
- Conductors as passengers. A conductor can begin a trip in a relief station different from where his/her last trip ended. Therefore, the conductor can travel as a passenger in a train in order to arrive to another station before his/her next trip begins.
- Routes. The transport network contains several different lines; however, trips of distinct lines cannot be assigned to a single duty.
- Conduction interruptions. An interruption is the minimum interval of time that should exist between two consecutive trips in a duty. At the end of each trip and in certain relief stations, a conductor can make certain maneuvers in order to conduct the train to the first station of his/her next trip. A set of distinct interruptions are allowed during one day. For example, the interruption for the interval of 7:00-8:00 a.m. can be two minutes, while the interruption for the time interval 13:00-14:00 can be one minute. If the time interval that occurs between two trips of a same duty surpasses a given interruption, then the extra

time of the interruption is considered as part of the rest or idle time. If not, it is considered as continuous conduction time.

Then, the problem consists in finding a feasible solution that generates the lowest number of duties with least total idle time.

Let $N = \{\tau_1, \tau_3,\ldots,\tau_n\}$ be a set of n trips that need to be completed during the programming period. For each trip k, the vector (s_k, e_k, o_k, d_k), denotes the start time, the finish time, the trip origin and trip destination. Figure 1 visually describes the problem; in 1a), the trips are organized around the x axis, which indicates the time of each programming day. A feasible duty d_k consists in a set of trips that can be made by a conductor satisfying all the problem constraints presented in the previous section. A solution x for the problem is identified by its set of duties $N(x)$. A solution is feasible if it is composed by feasible duties (Figure 1b). A feasible duty contains trips, rest periods, a meal break, interruptions, and idle time. The total duty time is obtained by adding all the time periods of the duty.

Insert Figure 1 about here

2. MODEL AND SOLUTION WITH EVOLUTIONARY AND CONSTRUCTIVE METHOD

State space representations, originally defined to represent problem solving in artificial intelligence, are useful tools to interpret search processes in combinatorial spaces such as those that typically emerge in combinatorial optimization. To represent a state space, a graph characterizes all the possible transitions (arcs) between the different problem states (nodes). With adequate use of a search method, a path between the initial given state

and a pre-defined goal state can be identified. Therefore, the problem solution is made up of the sequence of steps to reach the goal-state from the initial state, characterizing a path or a subgraph on the graph. To perform this process, several algorithms have been generated based in depth-first and breadth-first searches while using problem information to guide the process (Pearl, 1984).

The heuristic resolution of a combinatorial optimization problem can be tackled with two types of methods: constructive and solution space. In evolutive methods, a solution is built step by step from a set of initial elements. During this process, partial solutions are added each time, producing more complex structures. In the second case, the search takes place directly in a space containing all the possible solutions. From an initial solution, a random or deterministic exploration strategy allows a feasible space exploration. The search process ends when the optimal or a good solution is found. Both constructive and solution space methods can be represented as a search graph similar to those that emerge in the problem-solving area. The main difficulty that both search methods in problem-solving and the combinatorial optimization methods have in solving np-hard problems is the large amount of computer time required to find a good solution. To avoid this difficulty, the natural, implicit pruning scheme embedded in an evolutionary method may be considered to reduce the number of generated structures that need to be stored during the evolutionary construction (Chaudhry and Luo, 2005). Thus, gradually building several solutions and storing them in populations allow us to propose an evolutionary, constructive algorithm to solve combinatorial optimization problems. In Parada et al. (2002), we defined the corresponding operators and numerically tested several of these ideas by solving a cutting stock problem.

In each stage, our method updates a population of partial constructions $A(t)$ to generate $A(t+1)$ as occurs in the evolutionary methods (Figure 2). With equation (1) a set $A'(t)$ is obtained, formed of the best individuals m which constitute $A(t)$, denoted by c_j, where $j=1,2,\ldots, m$. The partial constructions obtained are submitted to three crossover operations. The first follows the constructive approach, allowing the elements of $A'(t)$ to combine with the original elements, which in this case correspond to the original trips represented in Figure 2 as set E. In the second, the combinations between the actual constructions are allowed; while in the third, following the evolutionary approach, the individuals of $A'(t)$ are combined with the best structures that are found during the search process, i.e., with the elite set $B(t)$. Subsequently, the mutation is performed (modules M in Figure 2), generating a set of partial constructions (C). Following the roulette method, the best solutions from C are selected to finally form the population $A(t+1)$. $B(t)$ is updated by deterministically selecting individuals from C (Algorithm 1).

$$A'(t) = \{c_j; f(c_j) = min\, f(c_i); \forall\ i \in A(t);\ j = 1,2,\ldots, m\} \quad (1)$$

The fitness of the partial constructions is evaluated using the evaluation function (2), where $g(node)$ is a measure of the cost to generate the construction *node* and $h(node)$ is an estimate of the cost that needs to be paid in the search constructive process until the solution of minimum f is obtained. The manner in which $g(node)$ and $h(node)$are calculated will be discussed in detail in this paper.

$$f(node) = g(node) + h(node) \quad (2)$$

In Algorithm 1, the pruning produced by the evolutionary components of the algorithm does not guarantee beforehand an optimal solution. However, when the random

components of the algorithm are removed, and without considering a limit for population and set size, the procedure becomes an intelligent problem-solving search algorithm like the best first search algorithm described in Pearl (1984). On the other hand, once a population $A(t)$ has been obtained with its individuals being solutions of the problem, that is to say solutions that contain all of the trips, the algorithm transforms into a genetic algorithm.

Algorithm 1: Evolutive Method

Input: Set of initial trips, parameters and restrictions
Output: Duties.
Begin
  $E \leftarrow$ Set of initial trips of the problem $\{p_1, p_2, \ldots, p_n\}$;
  Evaluate elements of E according to $f(e_i) = g(e_i) + h(e_i) \ \forall e_i \in E$;
  $A(0) \leftarrow$ Probabilistic Selection of elements of E according to $f(e_i)$;
  $B(0) \leftarrow \varnothing$; $t = 0$;
  Repeat
    $t = t+1$; $C \leftarrow \varnothing$;
    $A'(t) \leftarrow$ Selection from $A(t)$ according to (1);
    $C \leftarrow$ crossover and mutation between elements of
      $A'(t)$ and $A(t)$, $A'(t)$ and $B(t)$, $A'(t)$ and E;
    Apply mutation to elements of C;
    Evaluate $f(c_i) = g(c_i) + h(c_i) \ \forall c_i \in C$;
    Update $B(t)$;
    $A(t+1) \leftarrow$ probabilistic selection of best individuals of C;
  Until a termination criterion is satisfied;
End.

Algorithm 1 requires the definition of some parameters such as: the number of elements of $A'(t)$ which is determined as a fraction (f_s) of the number of elements in $A(t)$ and denoted as n_a; the size of $B(t)$ denoted as n_b; and the mutation probability p_m.

Insert Figure 2 about here

Representation of the crew scheduling problem requires the definition of a construction and an operator in order to generate a new construction based on a pair of

given constructions. A construction is defined by the set of duties, where a duty is defined by a combination of trips.

Let's consider a partial construction $x = \{d_1,\ldots,d_k\}$ defined as a set of duties. Two constructions x_a and x_b are combined to generate two new constructions x'_a and x'_b (Figure 3). Initially x'_a is obtained copying the duties of x_a. Then, all feasible trip insertions from x_b into x'_a are performed. When no other feasible insertion is possible, new empty duties are sequentially generated in x'_a in order to fit all the remaining trips of x_b into x_a. The process is complete when all the trips of x_a and x_b are contained in the duties of x'_a. A similar stage sequence gives x'_b. The feasible insertion of a trip into a duty takes into account all the problem constraints, with the exception that all of the trips are being considered.

In figure 3 , an example of the crossover operator considering 3 duties with 9 different trips in each one is represented. In this example, trips 10 and 15 are inserted in the third duty of x'_a, while trip 12 is inserted in the second duty of x'_a. Two other new duties for x'_a are generated with the remaining trips. Similarly, the second child x'_b is obtained.

Insert Figure 3 about here

The mutation of a new trip is performed as well with a feasible insertion of a trip not yet contained in the construction, but following a mutation probability p_m. A random number between 0 and 1 is generated for each construction, and if this number is smaller than p_m, then a feasible insertion of a new trip proceeds.

The constructive and evolutionary procedure continues until all of trips are included in the set of duties. Note that different work shifts can be easily considered by first separating the set of trips according to each trip's start time and then independently executing the algorithm.

The evolutionary process is conducted by the fitness function $f(x)$, as defined in (2). Both terms should fundamentally minimize idle time and the number of duties, noting that while the lower the idle time between each pair of trips, the higher the available time to fit another trip in the duty. On the other hand, a lower number of duties does not necessarily mean lower idle time. The objective is to minimize the number of duties so that each duty is as complete as possible with respect to the conductor's work day. In short, the duties must have the greatest number of possible trips, and consequently the smallest amount of idle time between each pair of trips. Therefore, idle time between tasks should also be minimized. Optimizing only the number of duties favors the generation of duties with high intermediate idle times, creating conductor discomfort. However, the exclusive minimization of idle time can generate an extreme case: lots of duties but some with only a few trips.

Two types of idle time were identified: internal idle time (t_{in}) consists of time between trips and external idle time (t_{ex}) corresponds to the interval between the end time of the duty's last trip and that work day's end time. Thus, total idle time of a duty is given by $t_i = t_{in} + t_{ex}$. Therefore the total idle time $t(x)$, of a construction x, is given by:

$$t(x) = \sum t_i \tag{3}$$

The generation cost $g(x)$ of a construction x is related to idle time $t(x)$ and the number of duties $N(x)$ belonging to it according to (4). Parameter α produces unit homogenization and weighting of both objectives. The algorithm's goal should be to contain all the trips grouped into the smallest number of duties and with the least amount of idle time possible. According to the construction operator's structure, the goal can be constructed from any construction x.

$$g(x) = t(x) + \alpha\, N(x) \tag{4}$$

Let $t'(x)$ be an estimate of the idle time still to be considered to reach the goal node. Let $N'(x)$ be an estimate of the number of duties in the goal construction. Thus, $h(x)$ is defined according to (5), which represents an approximation of the total cost of the goal construction reachable from x.

$$h(x) = t'(x) + \alpha\, N'(x) \tag{5}$$

Where:

T_{MD}: maximum conduction time for a conductor,

T_j: Duration of trip j,

T: Duration a conductor's work day.

Thus, $N'(x)$ is obtained considering the number of trips still to be assigned to any duty of the construction x as an estimate of the number of trips that can be contained in duty (4, 5).

$$N'(x) = \frac{n - N(x)}{T_{MD}/\overline{T}} \tag{6}$$

$$\overline{T} = \sum_{j=1}^{n} T_j / n \tag{7}$$

In the same way, $t'(x)$ is obtained from $N'(x)$ and the minimum idle time that can be obtained in a duty, i.e. the difference between the length of a conductor's work day and the time effectively used in conduction (8).

$$t'(x) = N'(x)(T - \left\lfloor \frac{T_{MD}}{\overline{T}} \right\rfloor \overline{T}) \tag{8}$$

In an evolutionary algorithm, the complexity is determined multiplying the number of elemental operations required by the algorithm in each stage by the number of generations, which will depend on the solution precision required. However, the constructive nature of algorithm 1 provides prior information on the limit for the number of

generations. Indeed, in the worst case, each new construction contains at least one trip more than were contained in the preceding constructions. In other words, the algorithm evolves constructing more complete solutions each time, until generating a population of complete constructions, which requires at most n generations, which requires at most n^2 elementary operations. Consequently a "good" heuristics solution requires around n^3 elementary operations.

3. SOLUTION USING OTHER TWO METHODS

3.1 Tabu Search Method (TSM).

The tabu search method (TSM) is a metaheuristic method that has been widely used in combinatorial optimization. The technique consists of a set of tools that facilitate the search for the optimal solution to the problem. In each stage, the neighborhood of the current solution is visited to search for a solution that lets the process continue advancing. The history of the path followed is considered to be an important source of information to proceed in the most effective and economic direction.

A detailed description of this methodology is found in Glover & Laguna (1997). To use this technique in a given problem, the following elements need to be identified: the structure that represents a problem solution, the rule that locates a neighbor solution in the actual solution, the way to measure the objective to be optimized, and an initial solution.

For the CSP, we represent a solution in the manner presented in section 2, with the modification that all the trips are contained in each solution x visited. Its evaluation is performed according to (4). To represent the neighborhood, an exchange operation needs to be defined. Let two duties be d_i and d_j such as in (9) and (10) with number of trips u and v, respectively.

$$d_i = \{\tau_{i1}, \tau_{i2},..., \tau_{il}, \tau_{i,l+1},..., \tau_{iu}\} \quad (9)$$
$$d_j = \{\tau_{j1}, \tau_{j2},..., \tau_{jm}, \tau_{j,m+1},..., \tau_{jv}\} \quad (10)$$

An exchange operation is defined between two duties d_i and d_j as the feasible exchange between some of the trips. To this end, the trips τ_{il} of duty d_i and the trips τ_{jm} of duty d_j are identified, generating two new duties (11) and (12), d_i' and d_j'. The exchange is identified by: $m(d_i, d_j, \tau_{il}, \tau_{jm})$.

$$d_i' = \{\tau_{i1}, \tau_{i2},..., \tau_{il}, \tau_{j,m+1},..., \tau_{jv}\} \quad (11)$$
$$d_j' = \{\tau_{j1}, \tau_{j2},..., \tau_{jm}, \tau_{i,l+1},..., \tau_{iu}\} \quad (12)$$

To obtain a solution x', neighbor of the current solution, we apply an exchange operation to a pair of duties of x. Thus, considering all the pairs of duties for a solution, a maximum of $n^3(n\text{-}1)$ neighbor solutions can be generated. We consider the *best_neighbor*(x) procedure that determines the best solution x' reviewing all the possible exchanges. The last q exchanges made $m_1, m_2,\ldots,m_q$ are stored in a tabu list in order to avoid the return of already visited solutions. Therefore, *best_neighbor*(x) considers feasible exchanges that do not imply the return to already visited solutions. A tabu move is considered applicable, except when it leads effectively to a solution that has yet to be visited. The tabu list is *fifo* (*first in, first out*), and consequently as the search process advances, the moves that entered first are the first ones eliminated. The tabu status of a move is always removed when the value of the new generated solution is better than the best values found up to that iteration in the search.

To generate an initial solution x^0 from an ordered list of n trips, n duties are constructed, where each one contains one of the n trips. Let t_{si} and t_{ei} be the start and finish time for each duty. Then $t_{si} = s_i$ and $t_{ei} = e_i$, $i = 1,\ldots,n$. We define a feasible concatenation operation as the incorporation into a duty of the only trip that contains another duty. In this

way, if $d_i = \{\tau_{i1}, ...,\tau_{iu}\}$ and $d_j = \{\tau_{j1}\}$, then $d_i \ \Theta \ d_j$ generates $d_i = \{\tau_{i1}, ...,\tau_{iu}, \tau_{j1}\}$. In algorithm 2, x^0 is generated, concatenating as many duties as possible. A concatenation is considered feasible when it satisfies problem constraints.

Algorithm 2: Concatenation.

Input: trip list, parameters, and constraints.
Output: x^0.
Begin
$L \leftarrow \{\tau_1, \tau_2, ...,\tau_n\}$ in ascending order of s_k;
$d_i = \{\tau_i\} \ \forall \ i = 1,\ldots,n$; $x = (d_i) \ \forall \ i$;
$t_{si} = s_i$; $\forall \ i = 1,\ldots,n$;
$t_{ei} = e_i$; $\forall \ i = 1,\ldots,n$; $i = 1$; $l = i+1$;
Repeat
Repeat
Determine k; $t_{sk} = \min\{(\ t_{sj} - t_{ei}), j = l, n\}$
$\Delta_i = t_{sk} - t_{ei}$;
If $\Delta_i \geq h$ then
$d_i = d_i \ \Theta \ d_k$; Eliminate d_k;
$t_{ei} = e_k$; $l = k + 1$;
Else $l = l + 1$;
Until $l = n$;
$i = i + 1$;
Until $i = n$;
$x^0 \leftarrow$ reordered list of duties.
End.

In the TS, the intensification strategy is used to review promising regions that had already been partially visited by periodically updating a list of solutions considered to be promising. These are solutions from which a continuous drop of the objective function can be obtained during a set number of iterations (5 in this case). The list of promising solutions acts with a FIFO order and have a pre-defined size. The function Intensify (L, x_f) found a new solution x_f from which the normal search continued. The intensification is activated after T_i iterations have occurred without the modifying best solution obtained until then.

Diversification is a strategy that permits a tabu search to visit a previously unconsidered region. With this objective, a new solution is configured based on the actual

solution, increasing the number of duties. To do this, a set that contains the duties with the largest number of trips is selected. Each of these duties is separated in the middle point, generating two new duties. This increase is performed after T_d iterations have occurred without finding a better solution. Algorithm 3 describes the resulting complete tabu search procedure.

Algorithm 3: Tabu Search Method.

Input: Trip list, parameters and constraints.
Output: Duties.
Begin
Generate x^0;
Define T_i and T_d ; $t_1 = 0$; $t_{lch} = 0$;
$x_c = x_{best} = x^0$;
For *Iter* = 0 to *Maxiter* do
$x' = best\_neighbor(x_c)$;
$x_c = x'$;
If $f(x') < f(x_{best})$ then
$x_{best} = x'$; $t_{lch} = Iter$;
Else
If $Iter - t_{lch} > T_d$ then
Diversify(x_c, x_f);
$x_c = x_f$;
Si $f(x_c) < f(x_{best})$ then
$x_{best} = x_c$; $t_{lch} = Iter$;
Else
If $Iter - t_{lch} > T_i$ then
Intensify(L, x_f); $x_c = x_f$;
If $f(x_f) < f(x_{best})$ then
$x_{best} = x_f$; $t_{lch} = Iter$;
End.

3.2 Greedy Method (GM)

Consider a tree enumerating all the feasible combinations of trips to generate a duty. The root node does not have included trips, and the next node level has duties that contain a pair of assigned trips. The intermediate nodes contain duties without all the trips having been assigned. The leaf nodes of this tree correspond to solutions of the problem in which each trip is assigned to some duty. Therefore, a constructive algorithm for the CSP can be obtained by identifying a way to find a path from the origin node to a leaf node. Algorithm

2 finds a path on this tree aiming to accumulate minimal idle time. The next node in a path is obtained incorporating a new trip to a duty in the earliest time possible.

The algorithm constructs sequentially a set of duties through feasible insertions of the trips. Initially, the trips are ordered in ascendant order according to the start time. In any stage of the algorithm, a duty is constructed by reviewing one by one the remaining trips and inserting them in a feasible manner until no new insertion is possible. Figure 4 presents an example of the construction strategy. In the upper part, the trips are presented ordered by start time, while the lower part presents the three duties obtained from those trips. The first duty is built with trips 1, 3, 5 and 8, leaving the trips 2, 4, 6, 7 and 9, which will be distributed to the other two duties in the next two stages.

Insert Figure 4 about here

Meal break assignment in each duty is performed in a randomly weighted manner so that the assignment has the greater probability to occur where the amount of the duty's conduction time is greatest.

In algorithm 4, the function *next_in_L* obtains the first element of the ordered list L and *insertion*(τ_l, $d(m)$) returns *true* when it is feasible to insert τ_l in the duty $d(m)$.

The complexity of this algorithm is determined by the 2 cycles. When both are executed n times, then the complexity is $O(n^2)$

Algorithm 4: Greedy Method

Input: Trip list, parameters and constraints.
Output: Duties.
Begin
  $L \leftarrow \{\tau_1, \tau_2, ..., \tau_n\}$ in ascendant order of s_k;
  $m = 0$;
  Repeat
    $m = m + 1$; $l = 0$;

Repeat
$l = l + 1$;
τ_l = *next_in_L*;
If *insertion*(τ_l , *d*(*m*)) then
$d(m) \leftarrow \tau_l$;
$nt(m) \leftarrow nt(m) + 1$;
$L \leftarrow L \backslash \tau_l$;
Until $l = n$;
Until $L = \phi$;
End.

4. RESULTS

The algorithms were executed in a computer with a 950 Mhz Athlon processor and 256 Mb RAM. Each instance was executed 5 times for both EA and GRA and the results correspond to the average values. The ten problems considered in this study correspond to a real situation that contains trips to be assigned in a work day. For example the problem Pr_1 has 733 trips, 3 relief stations and 58 trains; Pr_2 has 475 trips with two relief stations and 18 trains; Pr_3 has 437 trips, 2 relief stations and 22 trains. Two work shifts are considered, the first from 6:00 AM to 2:30 PM, the second from 2: 00 PM to 10:00 PM. Partial time conductors can be assigned at any moment during the day.

A parameter calibration experiment was performed for algorithm 1 using the problems Pr_2 and Pr_3. An approximate range for each parameter was identified from preliminary executions of the algorithm. Subsequently, the best set of parameters was defined within those ranges. The decision was made based on the best performance of the ones measured in terms of *f*(*node*). Thus, the established parameters are $n_a = 10$, $n_b = 5$, $f_s = 0.2$ and $p_m = 0.02$.

Similarly, a parameter-tuning experiment was performed for TSM. Using a set of preliminary executions to solve Pr_1, a range of best performance was determined for T_i, T_d

and λ. Fixing two parameters, the third was tuned, beginning with t_1 and finishing with λ. As a result of this process, $T_i = 250$, $T_d = 350$ and λ = 17 were obtained.

Figure 5 shows the evolution of the algorithm through the best construction of each generation $A(t)$. The minimum of $f(node)$ corresponds to the largest number of "good" duties generated, i.e. duties containing the maximum number of trips. After such point, $f(node)$ due to the increase difficulty of produce good duties waiting to be assigned. When there are few trips remaining in each construction, it is not possible to generate duties as good as the first ones. The function $h(node)$ has linear behavior since, in calculating the remaining number of duties to generate, only trips that have not yet been assigned to any duty are considered. This number diminishes linearly as the iterative process increases. The staggered behavior of $g(node)$ is due to the gradual generation of new duties that emerge when it is not possible to incorporate a new trip into the already existing duties.

Insert Figure 5 about here

The speed of the greedy algorithm to complete its job can be appreciated in Table 1; the computacional time required does not exceed two seconds and in all cases is less than the time required for TSM as well as EM. In turn, EM is faster than TSM in the ten cases studied.

The bicriterial nature of the problem studied in this paper can be seen to break down the objective function in its two components: idle time and number of duties. In this line of reasoning, it is found that TSM outperforms EM in the instances Pr2, Pr3, Pr7 and Pr9.

The number of duties is directly related to the number of train conductors required daily, while the idle time is related to the degree of concentration of duties and

consequently with the quality of work of the conductors. The manner of considering both elements depends on each company: in this study both criteria have been considered equally and the results obtained in six of the cases studied, EM solves the problem with a fewer number of conductors to fulfull the daily programming.

Insert Table 1 about here

5. CONCLUSIONS

In this paper, we proposed methods to address operation management problems which emerge in underground passenger transport. The first method proposed is based on the gradual solution construction by recurring to evolutionary methods. The generation of partial constructions is represented by an And / Or graph. The graph of the constructions is implicitly generated, following a scheme similar to that used by the evolutive algorithms, selecting in each stage a set that constitutes a population of constructions. Furthermore, the space state representation used as described in this paper is a useful tool to visualize any constructive algorithmic scheme. Under the framework described in this paper, such methods can be analyzed with the purpose of determining an admissible heuristic function that, at the same time, is an exactitude measurement. As a consequence, it is possible to identify theoretically the needed modifications in order to get an exact, instead of a heuristic, method.

The EM and TSM provide better results than the GM in terms of the number of duties. However, in terms of idle time, the TSM provides better results than the other two methods.

It is suggested to address this problem as a multiobjective optimization in future investigations.

Acknowledgments

The authors would like to thank two anonymous referees and an associate editor that helped greatly in improving the paper. The second author was supported by The Millennium Scientific Institute Chile: Complex Engineering Systems. The third author was supported by Project C-13955/18- Fundación Andes; Project 204.097.007-1.0 - UDEC and Project II-0457-FA-FCD-FI-FC-ALFA.

REFERENCES

Bengtsson, L. R., Galia, T., Gustafson, C. , Hjorring, C. and Kohl, N. (2004). Railway crew pairing optimization. Carmen Research and Technology Report, CRTR-0408. Carmen Systems AB.

Caprara, A., Fischetti, M., Toth, P., Vigo, D. and Guida, P. L. (1997). Algorithms for railway crew management. Mathematical Programming; 79; 1-3; 125-141.

Caprara, A., Fischetti, M. and Toth, P. (1999). A heuristic method for the set covering problem. Operations Research; 47; 5; 730-743.

Caprara, A., Monaci, M. and Toth, P. (2001). A global method for crew planning in railway applications. 17-36.

Cavique, I., Rego, C. and Themido, I. (1999). Subgraph ejection chains and tabu search for the crew scheduling problem. Journal of the Operational Research Society; 50; 6; 608-616.

Chaudhry, S. S. and Luo, W. (2005). Application of genetic algorithms in production and operations management: a review. International Journal of Production Research; 43; 19; 4083-4101.

Chew, K. L., Pang, J., Liu, Q. Z., Ou, J. H. and Teo, C. P. (2001). An optimization based approach to the train operator scheduling problem at Singapore MRT. Annals of Operations Research; 108; 1-4; 111-122.

Dias, T. G., de Sousa, J. P. and Cunha, J. F. (2002). Genetic algorithms for the bus driver scheduling problem: a case study. Journal of the Operational Research Society; 53; 3; 324-335.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., Nott, H. and Sier, D. (2001). An integrated optimization model for train crew management. Annals of Operations Research; 108; 1-4; 211-224.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B. and Sier, D. (2004). An annotated bibliography of personnel scheduling and rostering. Annals of Operations Research; 127; 1-4; 21-144.

Freling, R., Wagelmans, A. P. M. and Paixao, J. M. P. (1999). An overview of models and techniques for integrating vehicle and crew scheduling. Computer-Aided Transit Scheduling, Proceedings; 471; 441-460.

Freling, R., Huisman, D. and Wagelmans, A. P. M. (2003). Models and algorithms for integration of vehicle and crew scheduling. Journal of Scheduling; 6; 1; 63-85.

Freling, R., Lentink, R. M. and Wagelmans, A. P. M. (2004). A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm. Annals of Operations Research; 127; 1-4; 203-222.

Friberg, C. and Haase, K. (1999). An exact branch and cut algorithm for the vehicle and crew scheduling problem. Computer-Aided Transit Scheduling, Proceedings; 471; 63-80.

Gaffi, A. and Nonato, M. (1999). An integrated approach to ex-urban crew and vehicle scheduling. Computer-Aided Transit Scheduling, Proceedings; 471; 103-128.

Glover, Fred W. and Laguna, Manuel Tabu Search.Springer: Boston; 1997.

Haase, K., Desaulniers, G. and Desrosiers, J. (2001). Simultaneous vehicle and crew scheduling in urban mass transit systems. Transportation Science; 35; 3; 286-303.

Lourenco, H. R., Paixao, J. P. and Portugal, R. (2001). Multiobjective metaheuristics for the bus-driver scheduling problem. Transportation Science; 35; 3; 331-343.

Morgado, E. M. and Martins, J. P. (1992). Scheduling and Managing Crew in the Portuguese Railways. Expert Systems with Applications; 5; 3-4; 301-321.
Morgado, E. M. and Martins, J. P. (1998). CREWS_NS - Scheduling train crews in the Netherlands. Ai Magazine; 19; 1; 25-38.
Parada, V., Pradenas, L., Solar, M. and Palma, R. (2002). A hybrid algorithm for the non-guillotine cutting problem. Annals of Operations Research; 117; 1-4; 151-163.
Pearl, J. Intelligent Search Strategies for Computer Problem Solving.Addison Wesley Publishing Company; 1984.
Russell, S. J. and Norvig, P. Artificial Intelligence: A Modern Approach.Prentice Hall; 2002.

TABLE 1

Comparison of Results.

		EA			TS			GRA		
Problem	N° of trips	N° of Duties	Idle time [h:m:s]	Run time [m:s]	N° of Duties	Idle time [h:m:s]	Run time [m:s]	N° of Duties	Idle time [h:m:s]	Run time [m:s]
P_1	733	82.8	163:26:54	01:04	86	102:38:05	07:46	87.4	132:6:30	00:02
P_2	437	37.0	71:04:43	00:10	31	49:29:05	01:35	33.6	60:23:37	00:01
P_3	475	44.0	62:55:02	00:17	38	59:13:52	02:08	42.4	58:55:50	00:01
P_4	281	150.0	158:11:45	00:07	174	46:54:11	02:28	153.0	26:49:06	00:01
P_5	733	117.4	197:47:42	01:45	121	101:32:31	40:00	135.0	152:32:43	00:01
P_6	779	92.4	221:04:06	01:22	156	19:52:58	19:09	98.4	79:47:59	00:02
P_7	697	74.6	155:35:07	00:59	74	79:23:53	07:24	98.4	141:29:00	00:01
P_8	900	108.8	216:15:54	02:38	193	38:29:31	30:07	114.6	73:50:59	00:02
P_9	764	132.8	174:32:11	01:42	121	41:48:11	45:00	139.8	121:14:38	00:01
P_{10}	797	95.0	215:19:14	01:24	179	58:27:31	40:00	107.0	66:33:53	00:02

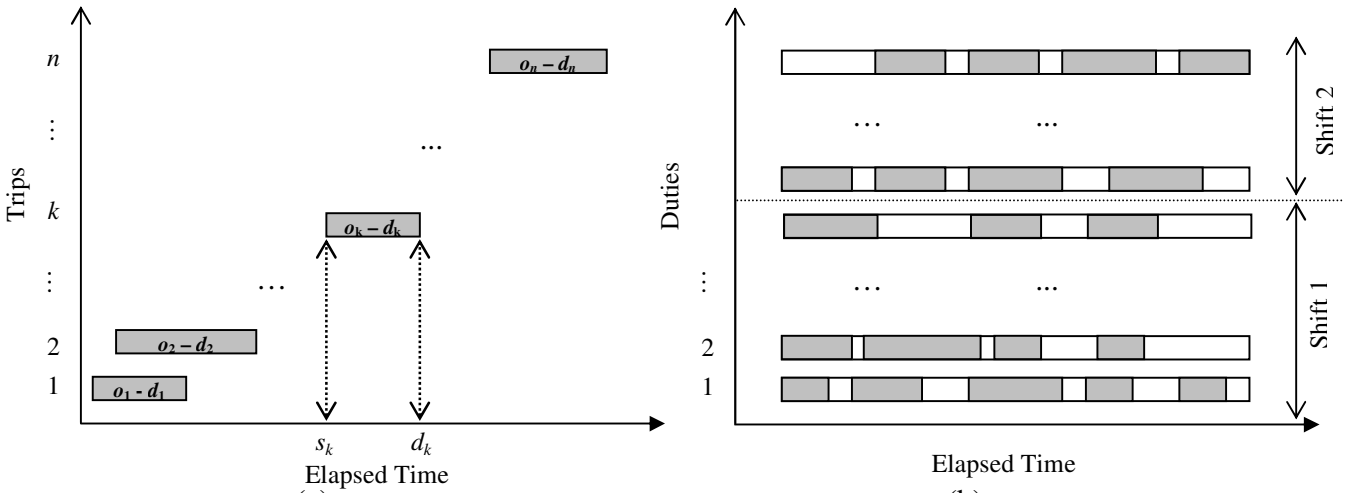

FIGURE 1
Schematic representation of the problem

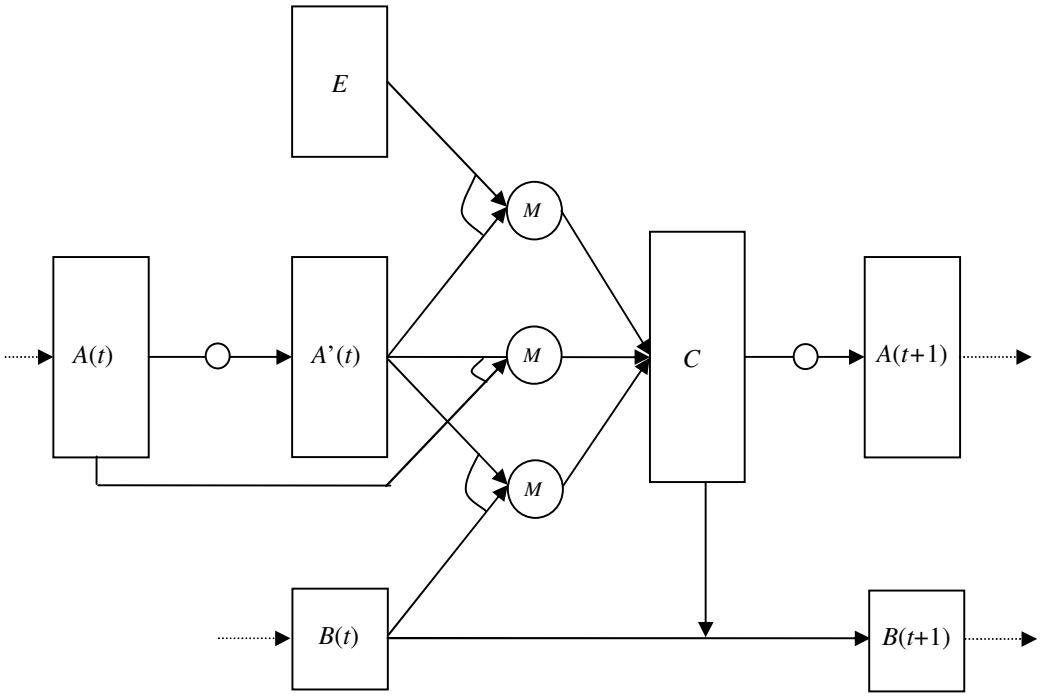

FIGURE 2
Algorithm

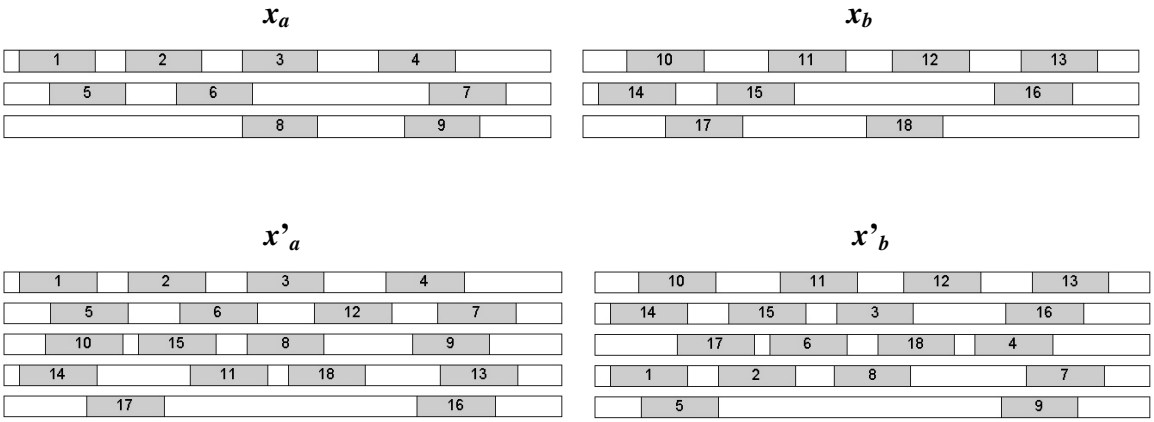

FIGURE 3
An example of the constructive operator

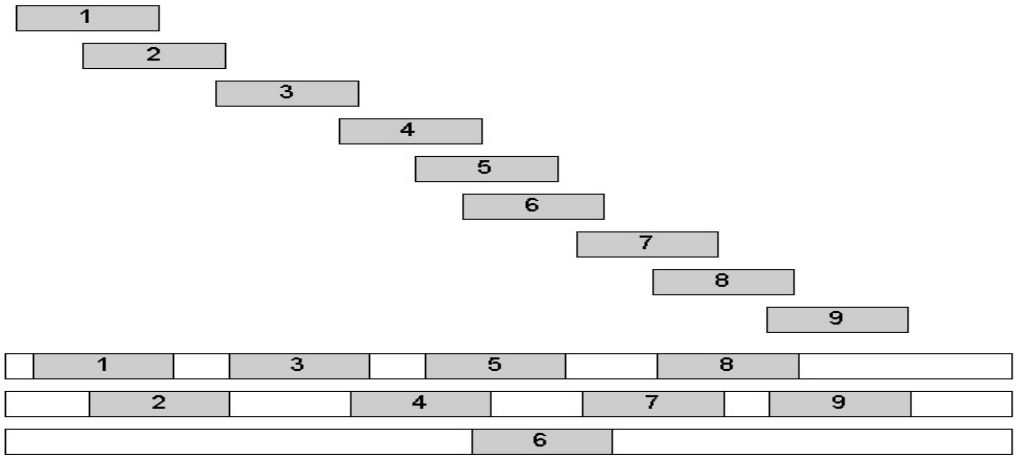

FIGURE 4
Nine originally ordered trips generating three duties

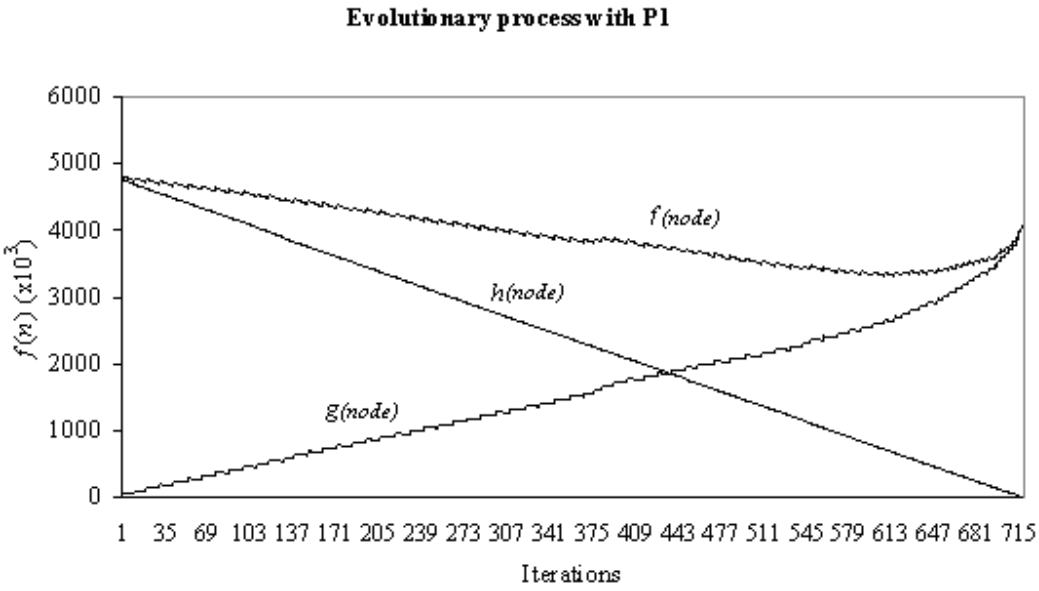

FIGURE 5 (*f*(n*ode*))
Convergence of EM