versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

A MULTI-OBJECTIVE ANT COLONY OPTIMIZATION METHOD APPLIED TO SWITCH ENGINE SCHEDULING IN RAILROAD YARDS

Jodelson A. Sabino*
Dep. de Engenharia Industrial / PUC-Rio
Rio de Janeiro – RJ, Brazil
IRIDIA-CoDE / Université Libre de Bruxelles
Brussels – Belgium
jodelson@pobox.com

José Eugênio Leal
Dep. de Engenharia Industrial / PUC-Rio
Rio de Janeiro – RJ, Brazil
jel@puc-rio.br

Thomas Stützle
Mauro Birattari
IRIDIA-CoDE / Université Libre de Bruxelles
Brussels – Belgium
stuetzle@ulb.ac.be, mbiro@ulb.ac.be

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 02/2008; aceito em 11/2009 após 1 revisão
Received February 2008; accepted November 2009 after one revision

Abstract

This paper proposes an ant colony optimization algorithm to assist railroad yard operational planning staff in their daily tasks. The proposed algorithm tries to minimize a multi-objective function that considers both fixed and variable transportation costs involved in moving railroad cars within the railroad yard area. This is accomplished by searching the best switch engine schedule for a given time horizon. As the algorithm was designed for real life application, the solution must be delivered in a predefined processing time and it must be in accordance with railroad yard operational policies. A railroad yard operations simulator was built to produce artificial instances in order to tune the parameters of the algorithm. The project is being developed together with industrial professionals from the Tubarão Railroad Terminal, which is the largest railroad yard in Latin America.

Keywords: ACO; ant colony optimization; railroad yard operational planning; switch engine scheduling.

Resumo

Este trabalho propõe um algoritmo de otimização com colônia de formigas para auxiliar a equipe de planejamento de operações de pátios ferroviários em suas tarefas diárias. O algoritmo proposto é baseado em uma função multi-objetivo que busca a redução dos custos fixo e variável de movimentação de vagões no pátio. Isto é feito através da busca da melhor programação para as locomotivas de manobra, considerando um dado horizonte de planejamento. Como o algoritmo foi desenvolvido para aplicação na vida real, a solução deve ser entregue em um tempo de processamento definido previamente e deve obedecer as políticas operacionais do pátio. Foi desenvolvido um simulador de operações de pátio que gera instâncias artificiais utilizadas para ajuste dos parâmetros do algoritmo. O projeto está sendo desenvolvido em conjunto com profissionais envolvidos na operação do Terminal Ferroviário de Tubarão, o qual é o maior pátio de manobras da América Latina.

Palavras-chave: ACO; otimização com colônia de formigas; planejamento operacional de pátios ferroviários; planejamento de operações de locomotivas de manobra.

1. Introduction

Railroad yards are very important components of a railroad network, as they are the points of origin and destination of all the shipments and freight movements. The management of a railroad yard involves a large number of variables, which makes it a complex task. Once in a yard, inbound trains are disassembled, unloaded and inspected. Next, the cars go to maintenance and cleaning. They are put in a waiting queue to be sorted, loaded and reassembled into blocks that form new outbound trains. The delays associated with these activities constitute a large portion of the overall transit time for rail freight. Because of this, yard activities constitute an important part of freight transportation operations.

The assignment and scheduling of yard locomotives (also called switch engines) to move around railroad cars within the railroad yard area is performed by operation controllers who make their decisions based on intuition, personal experience and the available information on the yard situation. Of course, this procedure does not guarantee an optimal use of the resources involved, and therefore arises the interest in developing computational tools to support human controllers. Our goal is to minimize the costs involved in the switch operations of a typical railroad yard. The costs are a function of the size of the switch engine fleet used as well as of the movements this fleet performs to execute the switch orders. The primary goal of cost reduction indirectly leads to the following collateral benefits:

a. Higher efficiency in resource utilization and higher throughput of switch orders;
b. Quicker response and fault reduction in decision making;
c. Robustness, specially concerning the ability of keeping good railroad yard performance under different circumstances and traffic conditions.

Literature on optimization in railroad yards is scarce. Apart from two previous related works, which are described in section 3, we are not aware of any study on the optimization of the switch engine assignment and scheduling. In this article, we study the application of ant colony optimization (ACO) algorithms for tackling this problem.

ACO algorithms generate candidate solutions for an optimization problem by a construction mechanism, where the choice of the solution component to be added at each construction step is done probabilistically biased by artificial pheromone trails and heuristic information on the problem under consideration (Dorigo & Stützle, 2004). In ACO algorithms, each (artificial) ant starts from an initially empty solution and adds solution components to its current partial solution until a complete candidate solution is obtained.

In this paper we tackle a real-life problem: The switch engine scheduling problem (SESP). The SESP is solved by means of a multi-colony implementation of the ACO technique designed to be used in a decision support tool to help the railroad yard operational planners in their daily routine. The algorithm proposed explores the idea of interactions between multiple ant colonies in solving a multi-objective optimization problem, as presented in Reimann (2002).

The remainder of this article is structured as follows: In section 2, we present basic concepts on railroad yard operational planning and define the SESP. In section 3, we make a brief overview on research and methods to solve the proposed problem. Section 4 presents the ACO algorithm proposed to solve the SESP, including the data structure, the pheromone update formulas and the action choice rules. Experimental results are the focus of section 5, which also presents the railroad yard simulator program, which was build to generate realistic instances. Finally, section 6 concludes this article.

2. Problem statement

This section describes the SESP and models it formally.

2.1 Concepts and terminology

In a rail system, the most important elements are the *cars*, in which the freight or the passengers are transported, the *locomotive* that gives moving power to railroad cars, the *tracks*, through which locomotives and cars move and the *crews* who operate the railroad system. A set of cars that share the same yard operational characteristics are put together in groups that we call *yard blocks*. They are pulled by one or more locomotives. This qualified name was given to differentiate these from the traditional concept of blocks as used in the railroad trip context. A set of locomotives and blocks attached together is called a *train*.

A rail system may be represented by a network where the nodes are stations with a special mission. Those stations may be simple pick-up and delivery stations, crew change points, engine refueling or junction points in the network. Some of these are the yards, which have an important function on the rail transportation: They may be *terminal yards*, with the main function of sending or receiving cars, or *classification yards*, with the main function of rearranging the order of cars in trains or moving cars from a train to another. In addition, some yards combine all or some of the following important functions: disassembling of incoming trains, inspection of cars and engines, cleaning of cars, loading or unloading of cars, train makeup and storage of empty cars. A yard is divided in sets of rail lines, or *tracks*, to attend each of the operational functions. Physically, a yard is built of a complex set of short tracks linked by crossings, turnouts and other railroad devices, which allows the change from one track to another. According to Bektas *et al.* (2007), one of the fundamental efforts in increasing railroad profitability and competitiveness is focused on efficient rail yard management by processing the cars in the yard as rapidly as possible to provide for an efficient utilization of the assets and the timeliness of connections.

Locomotives may work on trips transporting trains between yards or may operate within a railroad yard's bounds, moving around the cars. In the former case, the locomotive is said to be a railway locomotive. In the later case, it is called a *switch engine*. A switch engine may travel alone or it may have a set of cars attached to it. In the former case, it is said to be in *light running*; in the later case, the set formed by the switch engine plus the cars coupled to it is called a *convoy*.

A *switch order* is a transportation request to move a yard block set of cars from an origin track to a destination track within the yard. The act of performing a switch order corresponds to the following sequence of operations that involves a set of cars, operation and planning staff and a switch engine: *light running* from the switch engine initial position to the switch order pickup location; *coupling*, i.e., the operation of attaching a yard block to the switch engine; transportation of the yard block from its origin to the requested destination; and *uncoupling*, i.e., the last stage of the switch order when the cars are disconnected from the switch engine, so the switch engine is set free for being scheduled to another switch order.

It is important to clearly differentiate the naming convention for railroad yards in opposition to the corresponding railway terms. In the context of a railroad, a train is a set of blocks that travel in a railway pulled or pushed by locomotives to perform trips. In a railroad yard, a convoy is a yard block that travels in tracks pulled or pushed by switch engines to perform switch orders.

Personnel in charge for the operations of a railroad yard can be one of train crew, routing crew, maintenance crew, administrative staff or staff for other operations. The most important personnel for the scope of our work are the ones responsible for the yard operations. They are the *engine drivers*, who are the people in charge for driving the switch engines, the *break men*, who mainly couple and uncouple cars and finally the *dispatcher* who monitors and co-ordinates the movement of switch engines and convoys over tracks in a yard.

At a given instant t within the planning horizon, a switch engine e must be in one, and only one, of the following situations:

- In *transportation mode*, i.e., moving a yard block from its pickup to its delivery track;
- In *light running mode*, i.e., moving alone, without any yard block attached to it. This happens mostly when the switch engine moves to the pickup track of the following switch order to be performed;
- In *standing mode*, i.e., idle, parked at one track, with its engine turned on;
- At the pickup track and ready to work but *waiting for the pickup time* window to begin;
- At a certain track, *waiting for the following track* in its path to be freed by another convoy that is currently occupying that track;
- *Idle* at its last delivery track because there is no further switch order assigned to it;
- In *off mode*, i.e., the switch engine will not have any switch order at all assigned to it for the entire duration of the current time horizon, so it is idle at its initial position and it will remain all the time with its engine turned off.

As a matter of fact, a switch engine in off mode can not be considered as a part of the yard's switch engines fleet for the duration of the time horizon. If this situation extends to quite a long time it cannot be considered anymore for the computation of the fixed costs of the yard (e.g., it can be taken out from the yard and assigned to another task or it can be sold or its rental can be finished).

2.2 The switch engine scheduling problem

The problem addressed here can be summarized as follows: Given the information about the railroad yard layout, the switch engines available in it and a detailed specification of all pending planned switch orders, the goal is to determine a switch engine schedule, i.e., a one to one assignment of switch engines to switch orders and the ordering of the switch orders assigned to each switch engine such that none of the railroad yard operational constraints are violated and the costs are minimized.

The most important operational constraints to be considered are the following:

- Each switch order may have associated time windows limiting the earliest and latest time for both the pickup and the delivery of the yard block to be moved.
- Not every switch engine can execute any switch order. The total weight of the yard block to be moved has to be lower than the maximum traction effort the switch engine can supply.
- Some switch orders in the plan may have others as a prerequisite. This comes from the fact that sometimes the yard block to be moved is located behind other cars and therefore the nearest set must be moved first.
- All the planned switch orders must be performed (or, at least started) within the planning horizon.

Besides the ones above, the following constraints were assumed to simplify the problem modeling and solution, even if they are not strictly present in the real-world case:

- A switch engine cannot pickup an additional yard block before delivering the yard block it is transporting. This feature is known in the literature as *full truckload* and it is the most common mode of transportation in industrial applications (Jin & Muriel, 2006).
- The yard block must be transported directly from its pickup to its delivery location, i.e., it is not permitted to interrupt a request in course, attend another request, and continue to attend the interrupted request (maybe with a different switch engine), even if this strategy could eventually lead to an overall cost reduction.

There is at least one switch engine in the fleet capable of pushing or pulling the heaviest yard block to be moved. This constraint was set to avoid the need for more than one switch engine to be assigned to a switch order. We assume that if this happens in the real world case, it would be always possible to properly split the heavy convoy to avoid that.

2.3 Basic definitions

A more formal definition of the SESP requires the following elements:

Time: We deal with a fixed planning horizon of h time units that starts at time h_s and ends at time h_f so that $h = h_f - h_s$.

Switch engines: E is the set of switch engines that are available in the yard for executing switch orders during the planning horizon $[h_s, h_f]$. Each switch engine e has its associated power p_e and the corresponding maximum weight q_e that it can push or pull. A switch engine e has also a logical origin track e^+, where it is initially located, and a logical destination track e^-, where it is located at time h_f.

Request graph: Let $G = (V, A)$ be a directed graph, where the node set V embraces all logical tracks in the yard that are a pickup or delivery track of a switch order or the initial or ending position of a switch engine. The nodes v of this graph are connected by the arcs in A. We assume that there is (at least) one feasible path in the yard connecting any pairs of nodes i and j in such a way that a switch engine can go along a path starting at node i and ending at node j. Note that instead of physical locations, which may be identical, nodes in the request graph represent logical locations which are all distinct, therefore, we make distinction between the terms *logical track* and *physical track*: physical tracks are the tracks from the yard and logical tracks are the distinct tracks that appear in the request graph.

Yard layout: Let $G' = (V', A')$ be a node weighted mixed graph, where the node set V' describes all the physical tracks in the yard. The nodes of this graph are connected by the arcs so that the connections between adjacent tracks are represented by corresponding arcs linking adjacent nodes. The direction of these links indicates the operational possibility of moving a switch engine between adjacent nodes in the corresponding direction. The weight of each arc represents the distance traveled by a switch engine to go from middle to middle of two adjacent tracks. Figure shows two adjacent tracks in a yard named *track i* and *track j*, the length of each track (ω_i and ω_j respectively) and the distance ω_{ij} between them, computed by the formula $\omega_{ij} = (\omega_i + \omega_j)/2$. The reader is invited to note that according to the definitions of G and G' it naturally follows that $V \subseteq V'$, thus every node in G has a corresponding node in the graph G' that describes the layout of the yard.

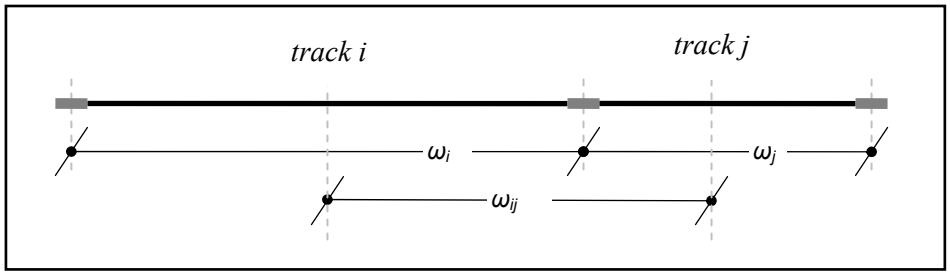

Figure 1 – Two adjacent tracks and the distance between them.

Switch orders: *R* is a given set of switch orders that must be scheduled for execution during the interval *[h_s,h_f]*. For each switch order *r* in *R*, there is one associated set of cars with a total weight w_r that must be picked up from track r^+ and delivered to track r^-, both located within the yard. The pickup time t^{r^+} must be within a given time interval $[t_s^{r^+}, t_f^{r^+}]$ and the delivery time t^{r^-} must be within a given time interval $[t_s^{r^-}, t_f^{r^-}]$.

For each physical track that occurs more than once in the set of switch orders *R*, the request graph contains as many distinguishable copies of this physical track as there are occurrences of the physical track in the set R. Hence, the request graph has *2|R|* nodes, where *|R|* is the number of switch orders, that is, for each switch order there are two nodes in the graph, one for the origin and one for the destination.

Space to time relationship: A switch engine moves in the yard at a speed s_e that mainly varies according to

- its engine power,
- the weight it is pushing or pulling (this weight may be the switch engine's weight itself when it is light running),
- the physical, operational and security constraints of the specific track on which it is moving (e.g., a security policy may impose speed limits for some specific tracks, meteorological conditions like snow or rain could also limit speed or the ground slope may influence speed).

Pickup and delivery path: This is, by definition, an ordered set

$$O_e = \{e^+, v_1, v_2, ..., v_n, e^-\} \subseteq V \quad (1)$$

representing a simple directed path in *G* for a switch engine $e \in E$, satisfying the following:

$$\text{If } v_i = r^+ \text{ in the path } O_e, \text{ then } v_{i+1} = r^-, \text{ i.e., a switch order is directly performed.} \quad (2)$$

$$\{r^+, r^-\} \subset O_e \text{ or } \{r^+, r^-\} \cap O_e = \varnothing, \ \forall r \in R \quad (3)$$

$$w_r \leq q_e \forall r \in R \mid r^+ \in O_e \quad (4)$$

$$\text{If } v_i = r^+ \text{ or } v_i = r^-, \ v_i \in O_e, \text{ then } T_{v_i} \leq t_f^{v_i} \text{ and } T_{v_i} = \max\{t_s^{v_i}, T_{v_{i-1}} + t^{O_e}_{(v_{i-1})(v_i)}\} \quad (5)$$

$$T_{e^+} \geq h_s \text{ and } T_{e^-} \leq h_f \quad (6)$$

where T_i is the instant at which the switch engine $e \in E$ arrives at node $i \in V$ when following its path O_e and $t^{O_e}_{(v_{i-1})(v_i)}$ is the total time needed to go along the fastest path connecting node v_{i-1} to node v_i.

Condition (2) is the full truckload constraint and condition (3) enforces the pickup and delivery pairing. Condition (4) is the problem capacity constraint, Condition (5) indicates that the switch engine may arrive at the pickup or delivery track before the corresponding time window begins; however, if this happens, the switch engine has to wait until the start time of the time window. Finally, Condition (6) specifies that all the switch orders must be executed within the planning horizon *h*.

Figure 2 illustrates a pickup and delivery path O_e. The solid line segments refer to switch engine's *e* travel in transportation mode and the dotted line segments corresponds to its travel in light running mode.

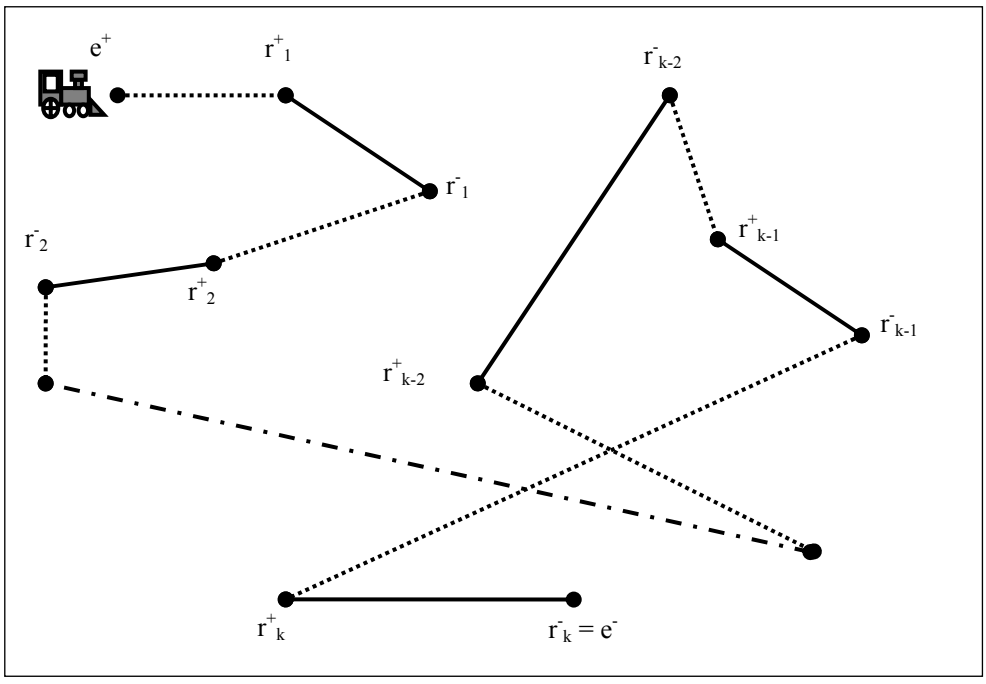

Figure 2 – Pickup and delivery path O_e covered by switch engine *e*.

The dash-dotted line segment represents a generic sequence of paired moves first in light running mode and then in transportation mode. The switch engine *e* starts the path from its initial track e^+; then it goes in light running mode up to the track where it has to pick up the load for its first switch order r_1^+; next it is coupled to the set of cars specified in the switch order r_1 and takes them up to the delivery track r_1^-; then it travels in light running mode to the second switch order's pickup track r_2^+ and so on until switch engine *e* performs its last delivery at track r_k^-. This last delivery track is also named e^-, as it is the track where the switch engine *e* ends up its service and where it is located at the end of the planning horizon.

By definition, every node in O_e is a track in the yard, so it has also a corresponding node in graph G'. We use this intuitive function $f : G \rightarrow G'$ to introduce the following definition: Given two adjacent nodes v_j and v_{j+1} in O_e, we take the shortest path P' in G' from v_j to v_{j+1} to define the distance traveled using the following formula:

$$l(v_i, v_j) = \sum_{k,k+1 \in P'} \omega_{k,k+1} \tag{7}$$

where $\omega_{k,k+1}$ is the distance traveled between a pair of adjacent nodes in the path P', i.e., between two connected tracks. Hence, we have that the total distance traveled in a pickup and delivery path O_e is given by:

$$\xi(O_e) = \sum_{v_z, v_{z+1} \in O_e} l(v_z, v_{z+1}) \tag{8}$$

2.4 Problem formulation

We call the set

$$\Omega = \{O_e\}_{e \in E} \tag{9}$$

of pickup and delivery paths such that:

$$\bigcup_{e \in E} O_e = V \tag{10}$$

a *switch engine scheduling problem* solution. We state that solving the *switch engine scheduling problem (SESP)* consists in finding the set Ω^* that minimizes the objective function:

$$C(\Omega) = \frac{c_1}{|E|}|\Omega| + \frac{c_2}{d} \sum_{O_e \in \Omega} \xi(O_e) \tag{11}$$

where $|\Omega|$ is the number of pickup and delivery paths in Ω, c_1 and c_2 are constants defined by the operational planning staff, $|E|$ is the fleet size, i.e., the number of switch engines available in the yard ($|E| \geq |\Omega|$) and d is the maximum distance that can be traveled during the time horizon by an average switch engine at an average speed.

The values of $|E|$ and d were introduced in the original objective function formula proposed in Reimann (2002) in order to normalize the values computed for the fixed and variable costs respectively. The values c_1 and c_2 express the relative importance of the fixed cost with respect to the variable cost involved in the yard operations, thus providing guidelines on which strategy to prioritize when comparing solution qualities: The savings in time spent and distance travelled or the fleet size reduction. These values can be computed from the relationship between the variable cost per time unit and the fixed cost per vehicle per time unit and additionally include the tactical business goals in its computation like increasing or decreasing the switch engine fleet size.

Formulas (9) and (10) describe a railroad yard operational plan for a given time horizon and Formula (11) expresses the objectives of reducing the total solution cost, which is the sum of the fixed and variable costs associated to the set Ω^*. Moreover, the minimization of the expression in Formula (11) implicitly enforces one specific path step route in G' (the one with minimal cost) to be chosen for each path step for each pickup and delivery path in Ω^*.

3. Overview on related works

This section presents the most relevant works related to our research. The railroad yard planning problem presented above is modeled as a multiple pickup and delivery problem with time windows and capacity constraints. It is then solved with a customized multiple ant colony optimization algorithm.

3.1 Railroad yard panning

Many references on railroad planning, like Kraft (1998), Winter (1999), Crainic (2002), Cordeau & Laporte (2003) and Sabino (2004), classify the railroad planning in three levels: strategic, tactic and operational, which correspond to the long term, medium term and short term planning, respectively. The first level is a long term planning related to resource acquisition, the second level concerns to resource allocation for some months and the third level is a short term planning which considers the daily resource utilization. This research deals with the determination of the switch engine schedule in the daily life of a railroad yard routine, which is therefore related to the *operational planning* level of a railroad transportation system. More specifically, at the operational planning level three main activities are performed in a railroad yard: Collecting main tasks, identifying all the necessary switches and defining the switch engine schedule. This research deals with the definition of switch engine schedules: In order to achieve better productivity and system cost effectiveness, the operational tasks are to be done in a combination of shorter time and lower costs. Railroad tracks, cars and switch engines are the resources of the system to be addressed here and productivity is the main goal as it highly influences customer satisfaction and business goals.

The work in Daganzo (1983) emphasizes two important components of the car cycle time: the time the car is actually in motion, which corresponds to 14% of the car cycle time, and the time the car is in a railroad terminal, which corresponds to 62% of the car cycle time. This suggests that a lot of effort should have been dedicated to railroad yard operations optimization. Nevertheless, very few works are available on this problem. According to Lübbecke (2001), the pioneers' work in Charnes & Miller (1956), has been the only one available on railroad terminal scheduling problems for 45 years. The following published work in that field was then Lübbecke (2001). This reference chain supports the evidence that the effort in this research represents something quite new.

The majority of the studies on railroad operational planning focus on railroad freight moving, trip locomotive scheduling, block to train assignment and train scheduling. This is probably because most of the railroad yards in operation nowadays were built some decades ago, when there was plenty of funding available not only to build lots of tracks in the yards but also to make many switch engines available without the need to worry about resource constraints. Anyway there are several papers on railroad operation and tactical planning like

Ahuja *et al.* (2002), Ahuja & Jha (2004) and Ahuja *et al.* (2004). Although these and other research papers like Crainic *et al.* (1984), Crainic & Roy (1988), Crainic & Laporte (1997), Gualda & Murgel (2000) and Crainic (2002) do not focus on the yard routine planning but on tactical and operational concerns of the entire railroad system planning, they gave the background on the terminology, the planning particularities and the approach presented in this work to solve the SESP.

The work in Kraft (1998), which presents a shipment scheduling and operational control method, and the work in Daganzo (1983), which proposes a procedure for improving the efficiency of classification yard operations, provided much background information on railroad yard operations planning, including the structured approach on hierarchical decision and the foundations for the switch engine travel model.

The modeling design to solve the SESP is, in part, based on the work of Pachl (2002). It contains a comprehensive description of the railroad yard elements, signaling principles and traffic control technologies.

3.2 Pickup and delivery problem

According to the model proposed in this research and considering the classical modeling presented in Dumas, Desrosiers & Soumis (1991) and many other works in applied optimization, the pickup and delivery problem with time windows can be considered the problem class that is closest to the SESP. In particular, the SESP can be seen as a pickup and delivery problem with time windows and capacity constraints (PDPTWC) with full truckloads. For a recent and comprehensive survey on pickup and delivery problems we refer the reader to Parragh *et al.* (2007) which covers all problem types where goods are transported between pickup and delivery locations, referred to as Vehicle Routing Problems with Pickups and Deliveries (VRPPD) and presents single and multi vehicle versions of the mathematical problem formulations with the corresponding exact, heuristic, and metaheuristic solution methods discussion.

There is a distinct range of practical PDP problems where, opposite to the SESP, clients require simultaneous pickup and delivery service. This happens, for example, in the soft drink industry, where empty bottles must be returned, reverse logistics problem, when manufacturers want to control the whole lifecycle of their products and in the delivery to grocery stores, where reusable pallets/containers are used for the transportation of merchandise. The reader may refer to Montané & Galvão (2006), Bianchessi & Righini (2007), Zachariadis *et al.* (2007), Subramanian & Cabral (2008) and Wassan *et al.* (2008) as recent references to this other range of PDP problems.

The PDPTWC is a special case of the general pickup and delivery problem, which is in turn a special case of the vehicle routing problem (VRP). A typical VRP occurs when distributing goods to customers. The solution of a VRP calls for the determination of a set of routes, each performed by a single vehicle that starts and ends at its own depot, in a way that all operational constraints are satisfied and the global transportation costs are minimized. For an overview we refer to Toth & Vigo (2002). The VRP turns to a pickup and delivery problem (PDP) if the routes must include both the collection and the delivery of goods and the goods collected from the pickup customers must be carried to the corresponding delivery customer by the same vehicle.

A PDPTWC is a PDP in which each vehicle has a capacity that must never be exceeded by the total weight of the goods being transported. In addition to the capacity constraint, there is a time window within which the vehicle must serve each pickup and delivery request. Optimization algorithms for the PDPTWC, such as the ones presented in Desrosiers *et al.* (1986) and Parragh *et al.* (2007b), include dynamic programming methods applied to the single-vehicle dial-a-ride problem, and the Dantzig-Wolfe decomposition method with column generation, which can also solve the multiple-vehicle version of the PDPTWC as shown in Lübbecke (2001). Since only modest size instances (up to 55 requests) have been solved optimally by such methods, several types of heuristics have been explored: decomposition methods, construction methods, improvement methods, and incomplete optimization based on mathematical programming methods. In fact, exact resolution is not viable for real size problems due to the excessive computational time required. On the other side, metaheuristics like ant colony optimization have shown to be capable of solving real problems close to optimality in reasonable computational time (Barcus, 2004). This evidence is present in some important surveys on the pickup and delivery problem like Mitrovic-Minic (1998), Mitrovic-Minic & Laporte (2003) and Parragh *et al.* (2007a and 2007b).

Finally, the expression *full truckload* refers to the restriction that a vehicle can only pick up a good if it is empty. In solving any real world case of this class of problems, a common objective is to reduce empty miles, and to improve customer service, like in Jula *et al.* (2003).

3.3 Ant colony optimization

Ant colony optimization (ACO) algorithms generate candidate solutions for an optimization problem by a construction mechanism where the choice of the solution component to be added at each construction step is done probabilistically biased by artificial pheromone trails and heuristic information on the problem under consideration (Dorigo & Stützle, 2004; Dorigo, Birattari & Stützle, 2006).

The work in Dorigo & Stützle (2004) highly influenced the selection of the ACO method to solve the SESP as it presents the ACO algorithms family as a set of efficient methods to solve several classic optimization problems. Nevertheless, the most important references to our research are: 1- Reimann (2002), which presents a multi-colony ACO algorithm to solve a full truckload transportation problem, similar to the problem been tackled in this research; and 2- the working paper by Gambardella *et al.* (2003), which presents two software tools to assist the schedule planner in solving goods distribution VRPs.

A couple of selected works that use ACO to solve the VRP in industrial applications are Pellegrini, Favaretto & Moretti (2007), which proposes two variants of ACO in a multiple colonies framework to solve a the problem of delivering food products to restaurants and retailers and Manfrin (2004), which cites some real life VRP examples.

4. Solution approach

This section presents an overview of the modeling and the solution approach we designed to tackle the SESP. The solution strategy is based on ACO.

The most important reasons for choosing ACO to solve the SESP were:

- The easiness of implementation and maintenance: Besides the straightforward implementation, new problem characteristics and constraints are added with minimal code review;
- good results in previous related researches: ACO has been shown to be an effective approach to solve combinatorial optimization problems like the one proposed in this research;
- the SESP belongs to the class of NP-hard problems and efficient exact algorithms to solve it for realistic problem sizes do not exist so far. Moreover, even for small real world SESP instances, one cannot guarantee to present a solution at any given time using exact approaches.

4.1 The algorithm framework

The basis for the algorithm designed to solve the SESP is the *CompetAnts* algorithm presented in Reimann (2002). There are two main reasons for choosing this procedure. The first is that CompetAnts outperforms the traditional ant system algorithm significantly for the problem tackled by Reimann, which is quite similar to the SESP, especially for the nature of the two conflicting sub goals. The second is that CompetAnts is rather robust and, according to Reimann (2002), it performs better than the other tested ACO implementations for instances with very tight and very loose time windows such as the ones that can be found very often in SESPs.

The following paragraphs provide a brief explanation of the CompetAnts logic. For a detailed explanation of this procedure we refer the reader to Reimann (2002).

There are two ant colonies with different priority rules. One colony, called the *empty move population*, emphasizes the reduction of variable costs related to light running moves from their original position to the pickup location. The other colony, called the *waiting time population*, focuses on maximizing the switch engine utilization by minimizing the waiting time at the pickup location, thus reducing the fixed cost related to the fleet size, that is, the number of switch engines needed.

It depends on the instance characteristics, which of the two colonies finds the best solution. For example, if time windows are tight, then the waiting time population will not have many chances to increase the utilization of the switch engines by reducing the waiting time at the pickup track. On the other hand, if time windows are wide, the waiting time population tends to increase more the utilization of the switch engines. Considering that reducing both waiting time and empty moves will reduce the total cost, the idea is to explore the strength of both colonies. First, the population size is not constant. After each iteration, ants from the colony that had the lowest average cost μ migrate to the other colony, therefore giving more computational power to that colony in the following iteration. Furthermore, some ants from each population utilize not only the pheromone information of their own population but also the one of the other. The ants decision about utilizing the foreign pheromone or not is based on the comparison of the best solution found by each ant colony in the previous iteration. The ants that use pheromone information from the other population are called *spy ants* and the ants that use only their own colony pheromone information are called *native ants*.

The schedule construction is done in the same way for both ant colonies. Starting at $t=0$, switch orders are sequentially assigned to a switch engine until the end of the planning horizon is reached or there is no more feasible assignment. At this point, another switch engine is

brought into use, the time is reset to zero and the switch order assignment is done for this new switch engine. This procedure is repeated until all switch orders are assigned or there is no other vehicle to bring into use. Only in the former case a feasible solution was found.

The CompetAnts procedure is shown in Figure 3. It sets up the two ant colonies, determines the number of ants and the number of spies in each colony, calls the Ant Systems algorithm once for each ant colony and controls the termination.

```
Procedure CompetAnts
   Read parameters;
   Initialize System;
   Execute first iteration
      Call AntSystem for Empty Move Population;
      Call AntSystem for Waiting Time Population;
   For i from 2 to max_iterations
      Adapt population sizes based on the average cost;
      Determine the number of spies based on best solution;
      Execute optimization
         Call AntSystem for Empty Move Population;
         Call AntSystem for WT Population;
   End For
   Output result
```

Figure 3 – The CompetAnts Procedure, as presented in Reimann (2002).

4.2 Action choice rules

There are three possible situations that an ant can face in the construction of its path, depending on the last operation done:

1. The last operation done added a switch order to a switch engine schedule and there is still some switch orders that may be assigned to the same switch engine, and then the next step will be to choose a new switch order;
2. the last operation done added a switch order to a switch engine schedule but there is no further order that may be assigned to the same switch engine, and then the next step will be to choose a new switch engine to bring into use;
3. the last operation introduced a new switch engine, and then the next step will be to choose the first switch order for that switch engine.

The pheromone information on the previous decisions at the three possible situations described above is stored in three different pheromone matrices. For example, if we have p switch orders and q switch engines, we use a (pxp) pheromone matrix to store the pheromone values corresponding to the first case above, that is, τ_{ij} gives the desirability of choosing switch order j directly after switch orders i, a (pxq) matrix in the second case where τ_{ij} gives the desirability of introducing switch engine j directly after switch orders i and a (qxp) matrix in the third case where τ_{ij} gives the desirability of choosing switch order j as the first switch order to be assigned to switch engine i.

The random proportional rule is the same in all cases and it follows the same formulas used in Reimann (2002). A native ant k_n makes its decision based in the following probabilities:

$$p_{ij}^{k_n}(t) = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{h \in N_i}[\tau_{ij}]^{\alpha}[\eta_{ij}(t)]^{\beta}} & \text{if } j \in N_i \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where N_i is the feasible neighborhood of ant k_n when being at node i, α and β are the parameters that determines the relative influence of the pheromone trail and the heuristic information and τ_{ij} is the pheromone information stored at the edge *(i, j)*.

The action choice rule for the native ant k_n is the usual ant system action choice rule. A spy ant k_f makes its decision based on the weighted average of the pheromone information $\tau_{i,j}^{n}$ of its own population and the foreign pheromone information $\tau_{i,j}^{f}$:

$$p_{ij}^{k_f}(t) = \begin{cases} \dfrac{[(\chi \cdot \tau_{ij}^{n}) + ((1-\chi) \cdot \tau_{ij}^{f})]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{h \in N_i}[(\chi \cdot \tau_{ih}^{n}) + ((1-\chi) \cdot \tau_{ih}^{f})]^{\alpha}[\eta_{ih}(t)]^{\beta}} & \text{if } j \in N_i \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where $0 \leq \chi \leq 1$ represents the importance of the own population's pheromone information.

Both the native and the spy ants consider all the operational constraints presented in subsection 2.2 when evaluating the feasibility of their neighborhood states. This subtle fact is an important difference between the original CompetAnts algorithm and the algorithm proposed in this work.

4.3 Priority rules

The priority rules are implemented through the use of distinct visibility formulas for each population. The priority rule for the waiting time population intends to maximize switch engine utilization and is given by the following formula:

$$\eta_{ij}^{WT} = \begin{cases} e^{-4K \cdot [PTWB_i + 2 \cdot EPST_i(t)]} & \text{if } (i,j) \text{ is a feasible edge} \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where $PTWB_i$ stands for the given pickup time window begin and $EPST_i$ stands for the computed early possible start time at the pickup location. $EPST_i$ takes into account the pickup time window and the real switch engine arrival time at the pickup location and uses the one that is later.

The constant K aims to avoid getting all zero values computed for η_{ij}^{WT} if the magnitude of $PTWB_i$ and $EPST_i$ is very large. The value of K used in the experiments is 0.25 times the value of the time horizon. The constants 4 and 2 are the same used in Reimann (2002).

The priority rule for the empty move population is:

$$\eta_{ij}^{EM} = \begin{cases} e^{-16\cdot(DIST(i,j))} & \text{if } (i,j) \text{ is a feasible edge} \\ 0 & \text{Otherwise.} \end{cases} \quad (15)$$

This priority rule is influenced solely by the distance to be traveled from the delivery location of the last switch order that was performed to the pickup location of the following switch order, so that the ants have a higher probability to build paths with lower total distance traveled to compose the solution. As the distance traveled from each pickup location to its corresponding delivery location are given data (i.e. not changed according to the problem solution), only the empty moves could be minimized to accomplish this goal; this justifies the name given to the empty move population. The constant -16 is the same as used in Reimann (2002).

The value of η which is the inverse of the distance between two adjacent cities in the original ACO algorithm presented in Dorigo *et al.* (2006), is now computed with formulas (14) and (15), defining new interpretations for the *distance* considered in the context of the SESP.

4.4 Pheromone update rules

Two different pheromone update rules were tested in this paper. The first uses the same schema proposed in Reimann (2002) and the second one follows the rank based ant system rule as proposed in Bullnheimer *et al.* (1999).

4.4.1 CompetAnts rule

After all the ants have completed their solutions, they possibly deposit pheromones on the edges they have traveled for building their solutions. In the pheromone update, first all the pheromone trails are lowered by some fixed factor. The update rule is as follows:

$$\tau = \rho \cdot \tau_{ij} + \sum_{\lambda=1}^{\Lambda} \Delta\tau_{ij}^{\lambda}, \forall (i,j) \in A \text{, and } 0 \le \rho \le 1 \quad (16)$$

Here, ρ is the trail persistence and A is the set of arcs in the request graph G. Only the ants that achieved the best Λ solutions deposit pheromones and pheromone evaporation occurs only once for each edge. The formula for pheromone deposit value is as follows:

$$\Delta\tau_{ij}^{\lambda} = \begin{cases} 1 - \dfrac{\lambda - 1}{\Lambda} & \text{if } 0 \le \lambda \le \Lambda \\ 0 & \text{Otherwise} \end{cases} \quad (17)$$

In this formula, proposed in Reimann (2002), the value of the objective function is not used to compute the value of $\Delta\tau_{ij}^{\lambda}$. The number of ranked ants is a parameter and it was set, as proposed in Bullnheimer *et al.* (1999), to the number of ants divided by 16. For the remaining of this paper, the CompetAnts pheromone update rule will be called CME.

4.4.2 Ant system rank-based rule

Besides the original pheromone update schema proposed in the CompetANTS algorithm, we tried an improvement over ant system called *rank-based* version where the best-so-far ant always deposits the largest amount of pheromone in each iteration. According to Bullnheimer *et al.* (1999) the rank-based ant system performs slightly better than elitist ant system and significantly better than the original ant system.

Before updating the pheromone trails, the ants are sorted by increasing solution cost and the quantity of pheromone an ant deposits is weighted according to the rank *r* of the ant. In each iteration only the *(w-1)* best-ranked and the ant that produced the best-so-far schedule are allowed to deposit pheromone. The best-so-far schedule gives the strongest feedback, with the weight *w*. The *r*-th best ant of the current iteration contributes with the weight given by *max{0,w-r}*. Thus the update of pheromone trails is made based on the following formula:

$$\tau_{ij} = \rho \cdot \tau_{ij} + \sum_{r=1}^{w-1} (w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs} \qquad (18)$$

where $\Delta\tau_{ij}^{r} = 1/C^{r}$, $\Delta\tau_{ij}^{bs} = 1/C^{bs}$, C^{bs} is the cost of the best-so-far schedule and C^{r} is the cost of *r*-th best ant schedule. All the schedule costs are computed using formula (11).

One important difference between the two pheromone update rules proposed here is that the ant system rank-based pheromone update rule (from now on simply called RNK) uses the cost C^{bs}, which is the objective function value corresponding to the best-so-far schedule considering all the iterations done so far. This best-so-far schedule does not necessarily belong to the set of ants of the current iteration, hence this preserves information about the quality of the past solutions, while the CME pheromone update schema uses only the feedback of the current iteration.

5. Experimental Analysis

This section presents the parameter settings and the empirical results. Subsection 5.1 briefly presents the method used to generate instances for testing. Subsection 5.2 presents the experimental setup and subsection 5.3 presents and discusses the experimental results.

5.1 The instance generator

The need for the development of the switch order generator program emerged during the computational test phase of this research program and it deserves some special remarks. As the main objective of this project is to apply the algorithm in real life, we were interested, since the very beginning, in comparing the computer delivered schedule with the one provided by the railroad yard operations planners. In the early stages of the research, as we had neither computer recorded data on the switch orders to be executed nor the scheduled plan passed to the switch order drivers, we observed for some hours the dispatcher's work at his office desk in the railroad yard control room, manually taking note of the input information gathered and output data (that is, the switch orders) delivered to the engine drivers. We then submitted the same input to a prototype version of the solution program and compared the operational cost of the dispatched switch orders done in real life with the cost

of the solution delivered by the prototype program (that is, the solution in the case the list of switch orders were known in advance and so the switch engines would work according to the plan proposed by the prototype program). The results of this simple comparison procedure were quite encouraging as they suggested huge savings. However the sample size, the sampling method and the sampling frame were considered not to be enough to evaluate the degree of efficiency of the newly proposed method.

Improving the real life data collection was a challenging task. The decision process is made manually and passed to the switch engine drivers by voice messages. This makes it very difficult to guarantee data integrity and also to get a reasonable large sample. This led us to the development of a switch order generator program, SOG hereafter. This program produces a set of switch orders to be executed based on the operation simulation of an unloading railroad yard and considering the following parameters that describe the most important physical and operational characteristics of the yard, considering that all the probability distributions are uniform:

GENERAL PARAMETERS

- planning horizon duration;
- average empty car weight;
- average loaded car weight;
- maximum service time;
- probability that an empty car block arrives.

TRAIN ARRIVAL

- arrival cycle time;
- probability that an arriving train contains n blocks (n=1, 2, 3, 4 or 5).

UNLOAD

- average duration of an assisted unload operation;
- average duration of a non-assisted unload operation;
- probability that a given unload operation will need switch engine assistance;
- probability that a loaded block will not be unloaded during the current time horizon.

INSPECTION

- probability that an inspected car needs maintenance;
- probability that an inspected block needs to be cleaned;
- maximum number of splits after inspection.

CLASSIFICATION

- probability that a block in the parking area needs classification switches.

SOG is implemented as a two stage procedure. In the first stage, a block arrival and unloading schedule is produced. The second stage procedure reads the data produced in the previous stage and progressively produces a consistent list of switch orders needed to perform all the operations, following the operational rules defined by the program input parameters. The switch orders generated in the second stage are chronologically added from

train break up to train make up for the given time horizon. All the input data required by SESP are generated for each switch order. The SOG logic was validated by field professionals and then several sets of SOG output were carefully checked against a set of manually gathered real life instances to fine tune the parameters of the generator so that it resembles real life data. The ability to build as many instances as we want with the SOG program was the essence of its usefulness.

5.2 Problem sets and output metrics

The goal of the computational experiments was (i) to test if one of the two implemented pheromone update rules variants prevails; (ii) to check if there exist recommended values for the initial number of ants, the heuristic exponent and the pheromone evaporation factor; (iii) to verify if the ACO recommended values (if any) change with the instance size and (iv) to check if the average elapsed time to produce good solutions were compatible with the performance expected by the practitioners from the railroad yard planning team.

To get a reasonable base input set, the SOG program was used to generate 50 input instances each one with over 200 switch orders to be executed in a 6 hour time horizon.

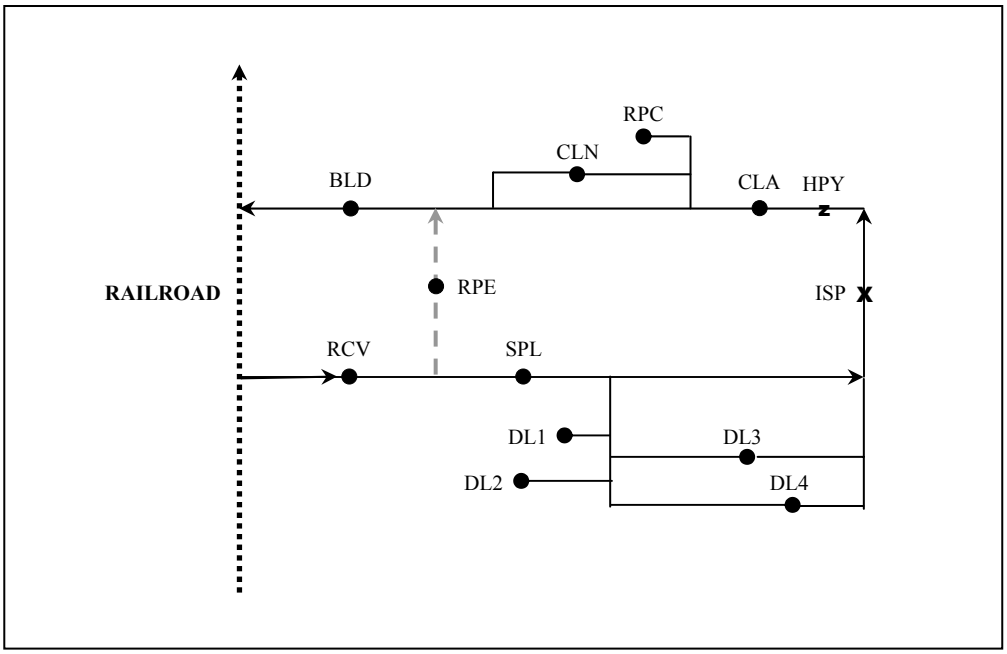

Figure 4 – The RRT1 railroad terminal layout.

The railroad yard layout submitted to SOG is a prototype unload terminal with assigned zones for train reception (RCV) and breakup (SPL), locomotive maintenance (RPE), unload facilities (DL1, DL2, DL3 and DL4), an inspection point (ISP), a hump yard (HPY) preceeding a classification zone (CLA), one cleaning zone (CLN), one car repair zone (RPC) and finally a train makeup zone (BLD). Figure 4 shows the layout of the RRT1 terminal. The

doted line represents the railroad and the continuous lines represent railroad yard tracks. Each black circle in this layout represents a branched structure of at least 6 parallel tracks. The interested reader is referred to Sabino (2004) for further details on the RRT1 terminal.

The operational parameter values used in the experiments are shown in Table 1. The planning horizon was set to 6 hours, which is the actual duration of a work shift as defined in the real world case study. The number of switch orders to be scheduled is the most important parameter regarding the problem size. As in the real world case it is around 10 switch orders per hour, thus adding up 60 switch orders per planning horizon, we considered three different problem scales: 20 to represent a small instance; 60 to represent the real world average case and 100 to represent a heavy loaded yard. The importance of fixed costs was set to a reasonable value considering the tactical goals for some real world cases collected. The number of switch engines available was set quite over the average number in real world. We did so because we were more interested in starting from a scenario where many switch engines were available and find out the number of switch engines needed in the solution than to start with a small set of switch engines and test the algorithm reaction to resource bottlenecks. To set up the switch engines initial locations, we randomly distributed them over the seven areas of the yard. The switch engines power was set randomly selecting some values from the real world case study. This approximately reproduces the real world situation.

Table 1 – Operational parameter settings for the experimental runs.

Parameter	Notation	Value(s)
Planning horizon (in hours)	h	6.0
Number of switch orders to be scheduled	*n_orders*	20, 60 and 100
Importance of fixed cost	c_1/c_2	0.8
Number of switch engines available	*n_vehicles*	40

The ACO parameter values used in formulas 12, 13, 16 and 18 are shown in Table 2. The initial number of ants for both ant colonies was 40 in the original CompetAnts algorithm, so we decided to test three values to investigate the effect of more ants in solution quality. This also served to validate the suggestion that the number of ants should be equal the number of cities in a TSP instance (Dorigo & Stützle, 2004). In fact, the number of cities in the TSP corresponds to double of the number of switch orders in the SESP as each switch order introduces a pickup and a delivery stop in the switch engine's route.

The pheromone exponent, the initial pheromone values and the number of ants that deposit pheromone in the RNK algorithm were set in accordance to the guidelines proposed in Dorigo & Stützle (2004). The population's pheromone importance was set to the same value used in Reimann (2002) which means that both populations have the same importance.

A single run was performed on n=50 different instances (Birattari, 2004). Every combination of operational and ACO parameter settings was submitted once both to RNK and CME program versions. This experimental strategy resulted in a set of 144 runs for each algorithm, producing 144 x 50 x 2 = 14400 results for experimental evaluation.

Table 2 – ACO parameter settings for the experimental runs.

Parameter	Notation	Value(s)
Initial number of ants (both for EM and WT colonies)	m	40, 60, 100 and 200
Pheromone exponent	α	1
Heuristic exponent	β	1, 3, 5 and 10
Pheromone evaporation factor	ρ	0.02, 0.10 and 0.50
Initial pheromone value	τ_0	0.1
Importance of the own population's pheromone	χ	0.5
Number of ants that deposit pheromone (only for RNK)	w	6

The number of iterations set for the CompetAnts algorithm was 30 for every run. After finishing the 30th iteration (this was the only stopping criterion) we gathered the following information:

- The solution value, i.e., the best objective function value found after 30 iterations;
- The CPU time consumed up to the point the best solution was found. This is not necessarily the total CPU time consumed after 30 iterations as the best solution can be found (and it is, in fact, mostly found) before the last iteration;
- The number of iterations up to the point the best solution was found. This value was kept just to make it possible to estimate a recommended number of iterations based on the comparison on the elapsed time, the CPU time consumed, the number of iterations and the expected time to get a good solution.

To summarize the data gathered, we always report the central tendency of the observation instead of the best result observed (Birattari & Dorigo, 2007). We used one way analysis of variance (a.k.a. ANOVA), with a level of significance of 5%, to compare the groups and evaluate whether there is evidence that the means of the populations differ. We never perform cumulative calculations for all the values of *n_orders* because they represent different sizes of operational loads to the yard and, thus, they must be analyzed separately.

Both algorithms were implemented in *ANSI C* and the source code was compiled with *gcc* version 3.4.6. All tests were run under Red Hat Enterprise Linux, CentOS release 4.5 running on an AMD Opteron™ 244 working at 1,75GHz with 1MB L2 cache and 2GB of RAM.

5.3 Experimental results

The first analysis done was a direct comparison between the solution qualities obtained in RNK and CME algorithm implementations. Figure 5 shows the solution value average and standard deviation over all tested values of 20, 60 and 100 switch orders. The actual values are shown on the left and a histogram shows the values side by side. In this overall analysis we can easily notice that there is no significant difference between the solution qualities for any value of *n_orders*. The analysis of variance supports this intuitive hypothesis as it did not produce evidence of statistically significant difference.

		Algorithm	
n_orders	Solution Value	**CME**	**RNK**
20	Avg	0,1254	0,1294
	StdDev	0,0272	0,0315
60	Avg	0,4245	0,4238
	StdDev	0,0362	0,0471
100	Avg	0,7857	0,7709
	StdDev	0,0516	0,0861

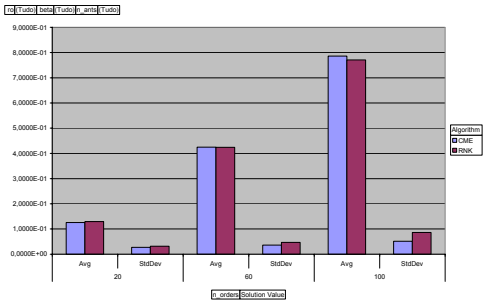

Figure 5 – Comparing solution quality for RNK and CME.

The first conclusion is that if we summarize over all the combinations of parameters tested (that is, without a closer look on the ACO parameters) we cannot say that any of the algorithms outperforms the other regarding the solution quality.

The CPU time consumed, as presented in Table 3, had a relatively high standard deviation in all cases thus making it impossible to detect any statistically significant difference, hence, from the overall results we cannot say that any algorithm version can consistently produce better results than the other, neither regarding the solution quality nor the solution cost.

Table 3 – Comparing CPU time consumed for RNK and CME.

		Algorithm	
n_orders	CPU time	**CME**	**RNK**
20	Average	1,43	1,26
	Standard Deviation	2,92	2,68
60	Average	31,18	24,39
	Standard Deviation	35,70	39,18
100	Average	87,87	75,35
	Standard Deviation	88,49	102,89

The next step was to verify the influence of the ACO parameters on the solution quality. We were particularly interested in obtaining recommended values for the parameters m, β and ρ. The histogram in Figure 7 compares the average solution value for each value of $m \in \{40, 60, 100, 200\}$. The bars are grouped according to the number of switch orders. We noticed that there is no significant difference in the solution quality for *n_orders=20*. For *n_orders=60* and *n_orders=100*, no matter the algorithm, the solution quality slightly increases as the value of m increases. Moreover, the RNK algorithm produces better results than CME for $m = 100$ and $m = 200$. All these statements are supported by the ANOVA statistical test. For example, from Figure 6 the summary and ANOVA essential probability information for $m = 200$ and *n_orders=60* shows that there is a statistically significant difference between the solution values for CME and RNK.

The fact that larger values for m lead to better results comes from the additional number of solutions generated as we increase the value of m.

SUMMARY

Group	*Count*	*Sum*	*Average*	*Variance*
CME	600	248,7811	0,414635	0,001284
RNK	600	243,1033	0,405172	0,001729

ANOVA

Source of variations	*SS*	*df*	*MS*	*F ratio*	*P-value*	*F critical*
Between Groups	0,026864	1	0,026864	**17,83305**	2,59E-05	**3,849232496**
Within groups	1,804707	1198	0,001506			
Total	1,831572	1199				

Figure 6 – ANOVA results for *m =200*, *n_orders=60*.

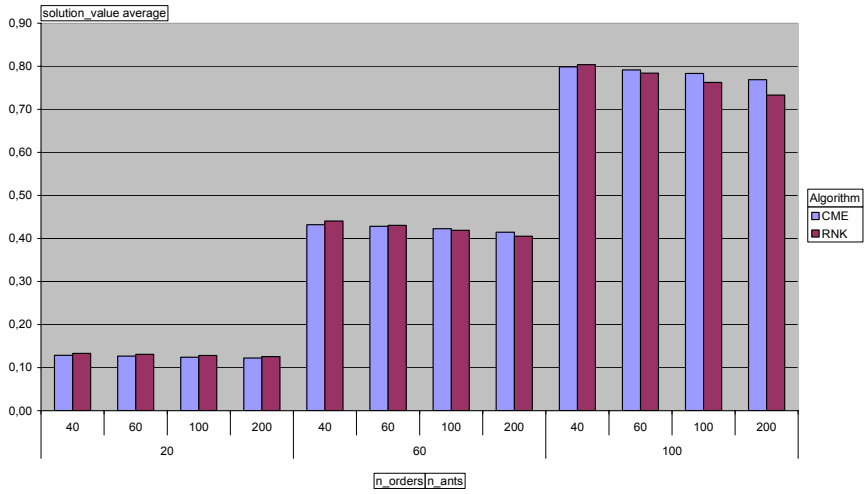

Figure 7 – Comparing RNK with CME solution for different values of *n_orders* and *m*.

Regarding the values of ρ and β, we came to the following conclusions after analyzing the average solution value for combinations of *m*, ρ and β for each possible value of *n_orders*:

- For the RNK algorithm, $\rho = 0.5$ leads to better solution values, especially for the higher values of *m* and no matter the value of β. This is valid, independent of *n_orders* and the best combination is $\rho = 0.5$ and $\beta = 5$;
- For the CME algorithm, $\beta = 5$ and $\rho = 0.5$ is the best combination for *n_orders=60* or *n_orders=100*. For *n_orders=20* the best combination is $\beta = 3$ and $\rho = 0.5$;
- RNK with its best combination $\rho = 0.5$ and $\beta = 5$ outperforms CME with any combination of ACO parameters.

These last statements can be illustrated with the two histograms in Figure 8, both for *m*=100 and *n_orders=60*, where the y-axis in the upper graph presents the average solution values for RNK and the graph on the bottom presents the equivalent values for CME. Each bar represents a different value of ρ. Each group of three bars represents a different value of β, so for each of the four values of β the corresponding results for the three values of ρ are

shown side by side. The bars are positioned with corresponding values of ρ and β sorted from left to right.

After finding the best combination for ρ and β we went back to the initial comparison between the two algorithms. The conclusions can also be illustrated with the graphs in Figure 8: As both graphs use the same scale for the y-axis, the better results obtained for the RNK algorithm, with β = 5 and ρ = 0.5 are visually evident.

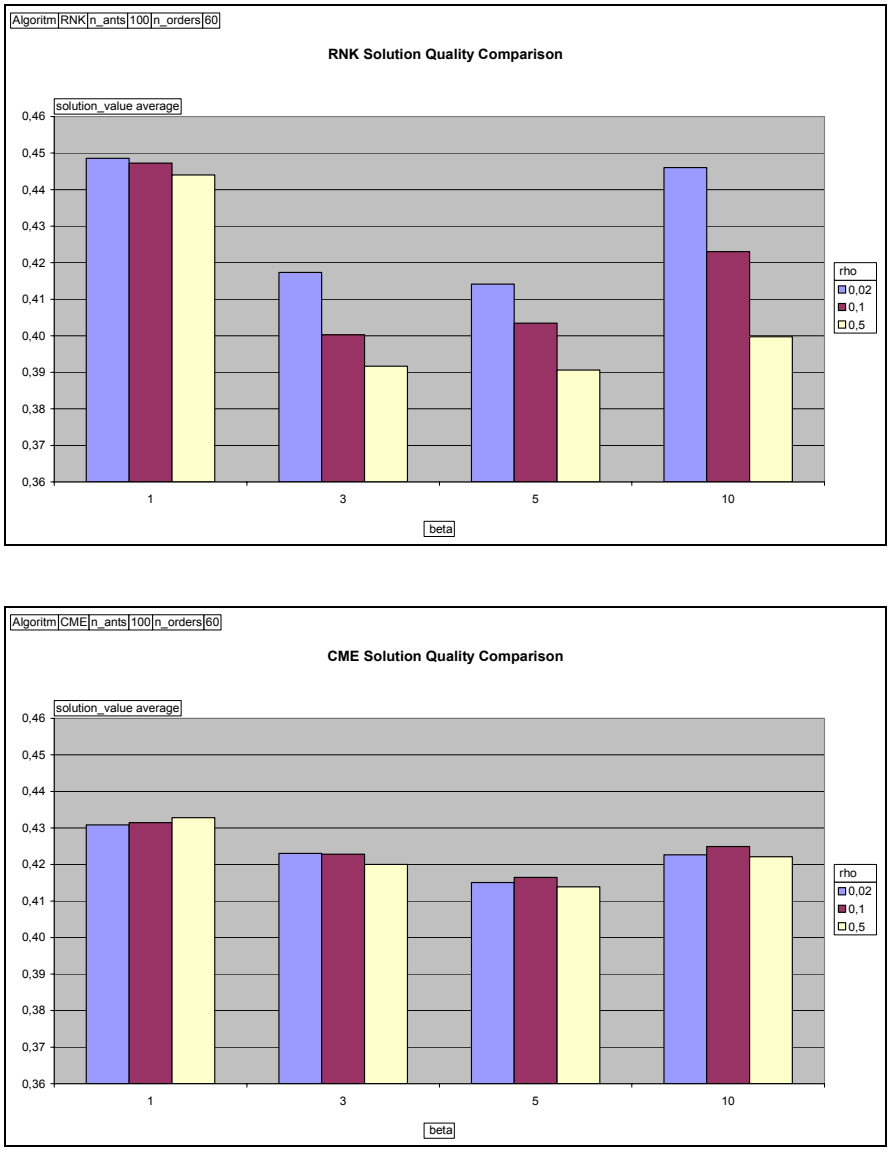

Figure 8 – RNK (up) and CME (down) solution values comparison for ρ and β changing.

A closer look at the internal characteristics of the RNK and the CME algorithms leads to the possible reasons for the better performance of the RNK: Considering that the CME logic does not carry on any information about the performance of the previous iterations to the following ones, RNK exploits more strongly the best-so-far solution and therefore is more directly guided towards good solutions.

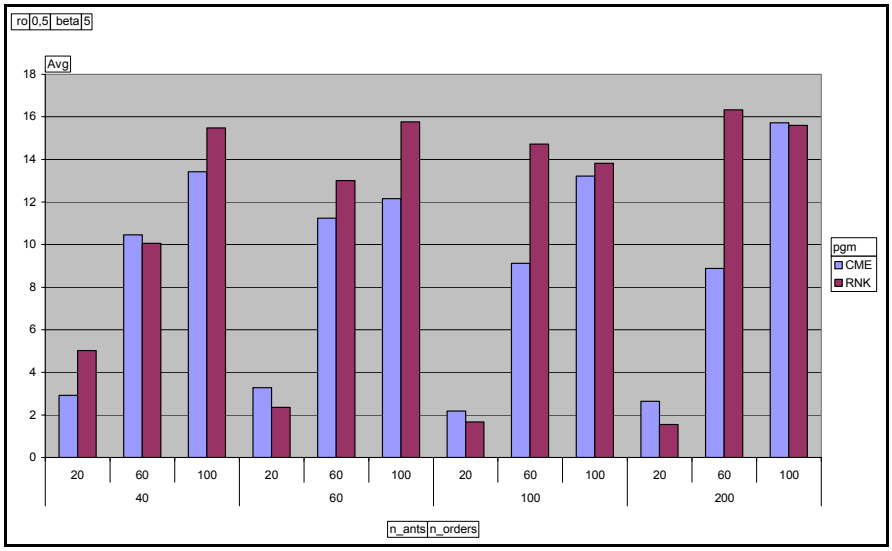

Figure 9 – Average number of iterations for RNK and CME.

Figure 9 shows the average number of iterations at which the best solution was found for different values of m considering $\beta = 5$ and $\rho = 0.5$. The bars corresponding to results with CME and RNK are paired for each value of *n_orders* and then grouped for each value of m. Clearly, the number of iterations needed to reach the solution increases with the value of *n_orders*. As the standard deviation was relatively high in all cases, all the remaining comparisons did not result in any statistically significant difference.

In general, the average number of iterations and, thus, the CPU time needed to get the solution were so disperse that it was impossible to come to any conclusion in the comparison of the solution costs.

To check the possibility of using the proposed algorithm to solve real world instances, we computed the average elapsed time over the 50 runs with the RNK variant using the best combination of parameters and considering *n_ordes*=60, which represents the most realistic instance size. The resulting elapsed time was 82 seconds, with standard deviation 0.5. This fits the expectations of the practitioners, as with this computation time it would be possible to recompute the switch engines schedule more than 100 times during a standard 6 hours planning horizon. The average number of switch engines used in the solutions of these realistic sized instances was 4, the minimum was 3 and the maximum was 7. These values are consistently less than the expected number for a yard with the same workload. These results shows that the implementation of the RNK algorithm to support the yard operations planning could also reduce the number of required switch engines to complete the yard's operations.

6. Conclusions and future work

We developed two variants of the CompetAnts algorithm to tackle the specific problem of switch engine scheduling in a railroad yard. The two implementations differ solely in their pheromone update rules. We developed also a switch order generator program to create sets of artificial input data, which enabled us to perform a significant number of computational experiments to identify the best solution approach and to produce parameter settings guidelines to solve this problem. For conveniently chosen ACO parameters, the RNK algorithm performs better than the CME algorithm thus this would be the choice for the practical problem.

In summary, the computational results obtained so far confirm that the switch engine scheduling problem can effectively be tackled in practice.

As one of the next steps, we will try to improve the solution quality obtained by the ACO algorithm through the inclusion of a post-optimization local search procedure, the adoption of other ACO algorithms like MACS-VRPTW (Gambardella *et al.*, 1999) or simply choosing other pheromone update schemas such as MAX-MIN ant system (Stützle & Hoos, 2000) or ant colony system (Dorigo & Gambardella, 1997). Another direction to extend the scope of this research is tackling other problems in the railroad yard operation. An example is the switch engine routing problem: how to determine the best route at a certain moment to go from the pickup to the delivery locations such that delays due to the concurrent need for a specific track are avoided. As a larger step, we are also considering to tackle other, more complex problems like the track allocation problem that requires for its solution actually a solution to the switch engine scheduling problem.

References

(1) Ahuja, R.K. & Jha, K.C. (2004). New Approaches for Solving the Block-to-Train Assignment Problem. Working Paper, Department of Industrial & Systems Engineering, University of Florida, Gainesville, FL, USA.

(2) Ahuja, R.K.; Jha, K.C. & Liu, J. (2004). Solving real-life railroad blocking problems. Working Paper, Department of Industrial & Systems Engineering, University of Florida, Gainesville, FL, USA.

(3) Ahuja, R.K.; Liu, J.; Orlon, J.B.; Sharma, D. & Shughart, L.A. (2002). Solving real-life locomotive scheduling problems. *Transportation Science*, **32**, 358-369.

(4) Barcus, L.; Rodríguez, V.M.; Álvarez, M.J.; Robusté, F. (2004). Routing Design for Less-Than-Truckload Motor Carriers Using Ant Colony Techniques. Working Paper, Departamento de Economía de la Empresa, Universidad Carlos III de Madrid.

(5) Bektas, T.; Crainic, T.G. & Morency, V. (2007). Improving performance of rail yards through dynamic reassignments of empty cars. Working Paper, Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT).

(6) Bianchessi, N. & Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, **34**, 578-594.

(7) Birattari, M. (2004). On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs? Technical Report, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

(8) Birattari, M. & Dorigo, M. (2007). How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs? *Optimization Letters*, **1**(3), 309-311.

(9) Bullnheimer, B. & Hartl, R.F. (1999). Strauss C. A new rank-based version of the Ant System: A computational study. *Central European Journal for Operations Research and Economics*, **7**(1), 25-38.

(10) Charnes, A. & Miller, M.H. (1956). A model for the optimal programming of railway freight train movements. *Management Science*, **3**, 74-92.

(11) Cordeau, J.-F. & Laporte, G. (2003). The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *4OR-Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, **1**, 89-101.

(12) Crainic, T.G. (2002). *A survey of Optimization Models for Long-Haul Freight Transportation*. Department of Administration Sciences, University of Quebec at Montreal, Montreal, QC, Canada.

(13) Crainic, T.G. & Laporte, G. (1997). Planning models for freight transportation. *European Journal of Operational Research*, **97**, 409-438.

(14) Crainic, T.G.; Ferland, J.A. & Rousseau, J.M. (1984). A Tactical Planning Model for Rail Freight Transportation. *Transportation Science*, **18**(2), 165-184.

(15) Crainic, T.G. & Roy, J. (1988). OR tools for tactical freight transportation planning. *European Journal of Operational Research*, **33**, 290-297.

(16) Daganzo, C.F.; Dowling, R.G. & Hall, R.W. (1983). Railroad Classification Yard Throughput: The Case of Multistage Triangular Sorting. *Transportation Research*, **17A**(2), 95-106.

(17) Dorigo, M.; Birattari, M. & Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, **1**(4), 28-39.

(18) Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization*. MIT.

(19) Dorigo, M. & Gambardella, L.M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, **1**(1), 53-66.

(20) Dorigo, M. & Stützle, T. (2002). The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. **In**: *Handbook of Metaheuristics* [edited by F. Glover and G. Kochenberger], pages 251-285. Springer, New York, NY.

(21) Dumas, Y.; Desrosiers, J. & Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, **54**, 7-22.

(22) Gambardella, L.M.; Rizzoli, A.E.; Oliverio, F.; Casagrande, N.; Donati, A.V.; Montemanni, R. & Lucibello, E. (2003). Ant Colony Optimization for vehicle routing in advanced logistics systems. **In**: *MAS2003 - The International Workshop on Modeling & Applied Simulation* [edited by A.G. Bruzzone and R. Mosca], pages 3-9.

(23) Gambardella, L.M.; Taillard, E. & Agazzi, G. (1999). MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. **In**: *New Ideas in Optimization* [edited by D. Corne, M. Dorigo, and F. Glover], pages 63-76. McGraw Hill, London, UK.

(24) Gualda, N.D.F. & Murgel, L.M.F.G. (2000). A Model for the Train Formation Problem. **In**: *Third International Meeting for Research in Logistics*, Trois-Rivières, Québec, Canada. CD-ROM.

(25) Jin, Y. & Muriel, A. (2006). Single-Warehouse Multi-Retailer Inventory Systems with Full Truck Load Shipments. Working Paper, Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst, MA.

(26) Jula, H.; Dessouky, M.; Ioannou, P. & Hall, R. (2003). Full-Truck-Load Assignment And Route Planning In Deterministic And Stochastic Environments. **In**: *NSF Design, Service and Manufacturing Grantees and Research Conference Proceedings* [edited by R.G. Reddy], pages 2957-2971. The University of Alabama, Tuscaloosa, AL 35487, USA.

(27) Kraft, E.R. (1998). A Reservations-Based Railway Network Operations Management System. 244p. PhD thesis, Department of Systems Engineering, University of Pennsylvania, Philadelphia, PA, EUA.

(28) Lübbecke, M. (2001). Engine Scheduling by Column Generation. PhD thesis, Braunschweig, University of Technology, Braunschweig, Germany.

(29) Manfrin, M. (2004). Ant Colony Optimization for the Vehicle Routing Problem. Diplome d'Études Approfondies en Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium. September 3.

(30) Maniezzo, V. & Carbonaro, A. (1999). Ant Colony Optimization: An Overview. Technical report, Scienze dell'Informazione, University of Bologna, Bologna, Italy.

(31) Mitrovic-Minic, S. (1998). Pickup and Delivery Problem with Time Windows: A Survey. Burnaby, BC, Canada. Technical report, SFU CMPT TR 1998-12 – School of Computing Science, Simon Fraser University.

(32) Mitrovic-Minic, S. & Laporte, G. (2003). Waiting Strategies for the Dynamic Pickup and Delivery Problem with Time Windows. Technical report, SFU CMPT TR 1998-12 – School of Computing Science, Simon Fraser University.

(33) Montané, F.A.T. & Galvão, R.D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, **33**, 595-619.

(34) Pachl, J. (2002). *Railway Operation and Control*. VTD Rail Publishing.

(35) Parragh, S.N.; Doerner, K.F. & Hartl, R.F. (2007). A survey on pickup and delivery models Part II: Transportation between pickup and delivery locations. *Zeitschrift für Betriebswirtschaft*, **58**(2), 81-117.

(36) Pellegrini, P. & Birattari, M. (2007). Implementation effort and performance: A comparison of custom and out-of-the-box metaheuristics on the vehicle routing problem with stochastic demand. **In**: *Engineering Stochastic Local Search Algorithms, International Workshop, SLS 2007* [edited by T. Stützle, M. Birattari, and H.H. Hoos], pp. 31-45. Springer Verlag, Berlin, Germany.

(37) Pellegrini, P.; Favaretto, F. & Moretti, E. (2007). Multiple Ant Colony Optimization for a Rich Vehicle Routing Problem: a Case Study. *Lecture Notes in Computer Science*, **4693**, Springer Berlin, Heidelberg, pp. 627-634.

(38) Reimann, M. (2002). Ant Based Optimization in Goods Transportation. PhD thesis, University of Vienna, Vienna, Austria.

(39) Sabino, J.A. (2002). Ant Colony Systems Applied to Switch Engine Assignment and Routing in a Railroad Yard. Student Paper Award on Management Science in Railroad Applications from RASIG, INFORMS (Institute for Operations Research and Management Science) Annual Meeting, San Jose, CA, USA.

(40) Sabino, J.A. (2004). Competição Entre Colônias de Formigas Aplicada à Designação de Locomotivas de Manobras em Pátios Ferroviários. Master's thesis – Departamento de Informática, Centro Tecnológico, Universidade Federal do Espírito Santo, Vitória (ES), Brasil.

(41) Stützle, T. & Hoos, H.H. (2000). MAX-MIN Ant System. *Future Generation Computer Systems Journal*, **16**(8), 889-914.

(42) Subramanian, A. & Cabral, L.A.F. (2008). An ILS based heuristic for the vehicle routing problem with simultaneous pickup and delivery and time limit. *Lecture Notes in Computer Science*, **4972**, 135-146.

(43) Toth, P. & Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM, Philadelphia, PA.

(44) Wassan, N.A.; Wassan, A.H. & Nagy, G. (2008). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of combinatorial optimization*, **15**(4), 368-386.

(45) Winter, T. (1999). Online and Real-Time Dispatching Problems. PhD thesis, Department of Mathematical Optimization, Braunschweig University of Technology, Braunschweig, Germany.

(46) Zachariadis, E.E.; Tarantilis, C.D. & Kiranoudis, C.T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with applications*, **36**(2), part 1, 1070-1081.