Published in Naval research logistics vol. 62, no. 1(2015) which should be cited to refer to this work.

A Learning Tabu Search for a Truck Allocation Problem with Linear and Nonlinear Cost Components

David Schindl[1] Nicolas Zufferey[2]

Abstract

The two-level problem studied in this paper consists of optimizing the refueling costs of a fleet of locomotives over a railway network. The goal consists of determining: (1) the number of refueling trucks contracted for each yard (truck assignment problem denoted TAP), and (2) the refueling plan of each locomotive (fuel distribution problem denoted FDP). As the FDP can be solved efficiently with existing methods, the focus is put on the TAP only. In a first version of the problem (denoted (P1)), various linear costs (e.g., fuel, fixed cost associated with each refueling, weekly operating costs of trucks) have to be minimized while satisfying a set of constraints (e.g., limited capacities of the locomotives and the trucks). In contrast with the existing literature on this problem, two types of nonlinear cost components will also be considered, based on the following ideas: (1) if several trucks from the same fuel supplier are contracted for the same yard, the supplier is likely to propose discounted prices for that yard (problem (P2)); (2) if a train stops too often on its route, a penalty is incurred, which represents the dissatisfaction of the clients (problem (P3)). Even if exact methods based on a MILP formulation are available for (P1), they are not appropriate anymore to tackle (P2) and (P3). Various methods are proposed for the TAP: a descent local search, a tabu search, and a learning tabu search (*LTS*). The latter is a new type of local search algorithm. It involves a learning process relying on a trail system, and it can be applied to any combinatorial optimization problem. Results are reported and discussed for a large set of instances (for (P1), (P2) and (P3)), and show the good performance of *LTS*.

Keywords: Truck allocation problem, Metaheuristics, Tabu Search.

1 Introduction and presentation of the problem

The considered problem, which was initially motivated by a problem described in the *Railway Applications Section of INFORMS*, consists of optimizing the refueling costs of a fleet of locomotives over a railway network [32]. It is assumed that there is only one source of fuel: refueling trucks, located at yards. Usually, such trucks are contracted on a yearly or quarterly basis. The problem has two important components:

[1]Geneva School of Business Administration (HEG), Geneva, Switzerland, david.schindl@hesge.ch
[2]**Corresponding author,** *Geneva School of Economics and Management*, GSEM - University of Geneva, Uni-Mail, 1211 Geneva 4, Switzerland, n.zufferey@unige.ch

choose the number of trucks contracted at each yard, and determine the refueling plan of each locomotive (i.e. the quantity of fuel that must be dispensed into each locomotive at every yard). Such components are respectively called the *truck assignment problem* (TAP) and the *fuel distribution problem* (FDP). The constraints are the following: the capacity of the tank of each locomotive is limited, as well as the maximum amount of fuel a truck can provide the same day (capacity constraints); it is forbidden to run out of fuel (fuel constraint). The encountered costs are the weekly operating cost of each refueling truck, the fuel price per gallon associated with each yard (which can vary from yard to yard), and the fixed cost associated with each refueling. A solution satisfying all the above constraints is called *feasible*. The problem consists of finding a feasible solution minimizing the sum of the costs.

Three versions of the problem are studied. They are denoted (P1), (P2) and (P3). In (P1), all the costs are linear and there is a maximum number of times (which is two in the considered instances) a train can stop to be refueled (stop constraint). (P2) is an extension of (P1) where discounted prices can be obtained from fuel suppliers. More precisely, if several trucks from the same supplier are contracted for a same yard, the weekly operating costs of the trucks are reduced in a nonlinear fashion, depending on the number of contracted trucks. (P3) is derived from (P1) by relaxing the stop constraint. Instead, if a train stops too often on its route from the origin station to the destination station, a penalty is incurred, which represents the dissatisfaction of the clients. Such a penalty depends on the number of refueling stops and is nonlinear. Therefore, problem (P3) has a link with customer satisfaction, which is a crucial performance indicator for the railway industry. As mentioned in [11], many organizations monitor their actual performance by measuring service quality (e.g., [16, 17, 30, 38]). As exposed in [31], the refueling operation represents a significant share of the operational costs in the transportation industry and in particular in railway transportation. The indirect costs caused by the delay of trains when their locomotives are being refueled or waiting to be refueled are obviously part of the refueling cost function. In [31], other possible extensions of problem (P1) are also presented, which may be applicable for various practical situations.

Given a solution of the TAP (i.e. a number of trucks for each yard), the FDP can be very efficiently tackled with a so-called *FDP algorithm*, and this was presented in [32]. Therefore, this paper will only focus on the TAP. Some information is however given here to have some idea about the *FDP algorithm*, which consists of three components: a flow algorithm, a repairing procedure and a descent local search. The flow algorithm is an exact method based on a graph model in which a compatible flow with minimum cost is searched. If the fuel constraint is violated, the associated TAP solution is not considered further (as at least one refueling truck is necessarily missing). If the fuel constraint is satisfied but the stop constraint is violated, the repairing procedure is triggered in order to reduce the number of stops without running out of fuel. If it fails, the associated FDP solution is again not considered any further. If a feasible solution is found, then it is improved with a descent local search, where a move consists of adding or removing a refueling opportunity.

This work is an extension of [32], where only problem (P1) is considered, and for which only the single instance provided in [15] is tackled. The main contributions of this paper are the following: (1) two nonlinear extensions of problem (P1) are studied, namely problems (P2) and (P3); (2) a more complete literature review is exposed; (3) a new and general metaheuristic, called the learning tabu search, is proposed in a way it can be adapted to any combinatorial optimization problem; (4) we report on a comprehensive numerical study:

57 instances for (P1), 15 instances for (P2), and 15 instances for (P3).

The paper is organized as follows. Based on [15], an extended literature review is presented in Section 2. In Section 3, some well-known local search techniques are first briefly described (namely the descent algorithm and tabu search). Then, a new type of local search is proposed within a tabu search framework (namely learning tabu search), involving a learning process relying on a trail system. Solution methods for the TAP are designed in Section 4, namely a descent local search, a tabu search, and a learning tabu search. The obtained results are discussed in Section 5. The paper ends up with a conclusion in Section 6.

2 Illustration of the considered problem and literature review

In this section, problem (P1) is illustrated on a small example (subsection 2.1). Then, important surveys on railroad planning and operations are discussed (subsection 2.2). Finally, the literature review focuses on papers dealing with refueling aspects (subsection 2.3).

2.1 Presentation of a small instance of problem (P1)

Based on [15], a small instance is now provided to better capture problem (P1). Consider a simple railroad network made of four yards: y_1, y_2, y_3 and y_4. There are tracks connecting directly y_1 with y_2, y_2 with y_3, and y_3 with y_4. Two trains (t_1 and t_2) run in this railroad every day of the week. Assume that for t_1, the sequence (or the route) in which it stops at the yards is $y_1 \rightarrow y_2 \rightarrow y_3 \rightarrow y_4$. In such a case, y_1 is the *origin* station, y_4 is the *destination* station, and y_2 and y_3 are *intermediate* stations. The sequence in which t_2 stops at the yards is $y_4 \rightarrow y_2 \rightarrow y_1$. The distance (in miles) between yards is symmetrical: $d(y_1, y_2) = 106$, $d(y_2, y_3) = 146$, $d(y_2, y_4) = 162$, $d(y_3, y_4) = 16$. There are two locomotives available for powering the trains: l_1 and l_2. Each locomotive consumes 3.5 gallons of fuel per mile and a locomotive tank holds up to $4,500$ gallons of fuel. Therefore, on a full tank, a locomotive can run for 1285.71 miles. Further, assume that a truck can fuel up to $25,000$ gallons per day. The weekly operating cost per truck is $4,000$\$ and fuel cost per gallon at each yard is: 3.25\$ for y_1, 3.05\$ for y_2, 3.15\$ for y_3 and 3.15\$ for y_4. The fixed refueling cost is 250\$. The optimal decision on the number of contracted trucks consists of only contracting a single truck at yard y_2. The refueling plan for each locomotive train assignment cycle is described in Table 1. In this table, the column "Yard" marks the sequence of yards where the corresponding locomotive can be refueled. Column "Day" denotes the day in the planning horizon in which this event takes place. In the column "Station", "Or." denotes "Origin" and "Int." denotes "Intermediate". Finally column "Stop No." (for "Stop Number") is used to describe the sequential order of potential refueling stops for a locomotive. Observe that neither l_1 nor l_2 are refueled for the first sequence of the train assignment cycle (first row on the table) and this is still a feasible solution because both locomotives still have fuel remaining from the previous occurrence of the cycle. The cost of this solution over the planning horizon (2 weeks) is $90,105.5$\$. Fuel costs are $80,105.5$\$, truck costs are $8,000$\$ and the stop costs are $2,000$\$.

	Locomotive l_1				Locomotive l_2			
Stop No.	**Yard**	**Station**	**Day**	**Gallons**	**Yard**	**Station**	**Day**	**Gallons**
1	y_1	Or.	1	0	y_4	Or.	1	0
2	y_2	Int.	1	1,870	y_2	Int.	1	0
3	y_3	Int.	1	0	y_1	Or.	2	0
4	y_4	Or.	2	0	y_2	Int.	2	0
5	y_2	Int.	2	0	y_3	Int.	2	0
6	y_1	Or.	3	0	y_4	Or.	3	0
7	y_2	Int.	3	4,500	y_2	Int.	3	4,500
8	y_3	Int.	3	0	y_1	Or.	4	0
9	y_4	Or.	4	0	y_2	Int.	4	0
10	y_2	Int.	4	0	y_3	Int.	4	0
11	y_1	Or.	5	0	y_4	Or.	5	0
12	y_2	Int.	5	0	y_2	Int.	5	0
13	y_3	Int.	5	0	y_1	Or.	6	0
14	y_4	Or.	6	0	y_2	Int.	6	0
15	y_2	Int.	6	3,010	y_3	Int.	6	0
16	y_1	Or.	7	0	y_4	Or.	7	0
17	y_2	Int.	7	0	y_2	Int.	7	0
18	y_3	Int.	7	0	y_1	Or.	8	0
19	y_4	Or.	8	0	y_2	Int.	8	4,494
20	y_2	Int.	8	0	y_3	Int.	8	0
21	y_1	Or.	9	0	y_4	Or.	9	0
22	y_2	Int.	9	0	y_2	Int.	9	0
23	y_3	Int.	9	0	y_1	Or.	10	0
24	y_4	Or.	10	0	y_2	Int.	10	0
25	y_2	Int.	10	3,752	y_3	Int.	10	0
26	y_1	Or.	11	0	y_4	Or.	11	0
27	y_2	Int.	11	0	y_2	Int.	11	386
28	y_3	Int.	11	0	y_1	Or.	12	0
29	y_4	Or.	12	0	y_2	Int.	12	0
30	y_2	Int.	12	0	y_3	Int.	12	0
31	y_1	Or.	13	0	y_4	Or.	13	0
32	y_2	Int.	13	0	y_2	Int.	13	3,752
33	y_3	Int.	13	0	y_1	Or.	14	0
34	y_4	Or.	14	0	y_2	Int.	14	0
35	y_2	Int.	14	0	y_3	Int.	14	0

Table 1: Fueling plan for each locomotive train assignment cycle in the solution

2.2 General literature review on railroad planning and operations

Railroad planning and operations is a wide field with a lot of money involved. It is exposed in [26] that the railway planning process consists of three levels: (1) strategic (e.g., network planning, line planning); (2) tactical (e.g., timetable generation, railway track allocation, train routing, rolling stock schedules, crew schedules); (3) operational (e.g., real time management). The authors survey papers on the train timetabling, train dispatching, train platforming, and train routing problems, group them by railway network type, and discuss track allocation from a strategic, tactical, and operational level.

In [27], the authors concentrate on organizing, planning and managing the train movement in a network. The three classical management levels for rail planning (i.e. strategic, tactical and operational) are introduced followed by decision support systems for rail traffic control. In addition, they discuss train operating forms, railway traffic control, train dispatching problems, rail yard technical schemes, the performance of terminals, and timetable design. A description of analytical methods, simulation techniques and specific computer packages for analyzing and evaluating the behavior of rail systems and networks is also provided.

In [6], the authors survey the main studies dealing with the train timetabling problem in its *nominal* and *robust* versions. The nominal version of the problem amounts to determining good timetables for a set of

trains (on a railway network or on a single one-way line), satisfying the so-called track capacity constraints, with the aim of optimizing an objective function that can have different meanings according to the requests of the railway company (e.g., one can be asked to schedule the trains according to the timetables preferred by the train operators or to maximize passenger satisfaction). The following two are the main variants of the nominal problem: one is to consider a *cyclic* (or periodic) schedule of the trains that is repeated every given time period (for example every hour), and the other one is to consider a more congested network where only a *non-cyclic* schedule can be performed. In recent years, many papers have been dedicated to the robust version of the problem. In this case, the aim is to determine robust timetables for the trains, i.e. to find a schedule that avoids, in case of disruptions in the railway network, delay propagation as much as possible. The authors present an overview of the main papers on train timetabling, underlining the differences between models and methods that have been developed to tackle the nominal and the robust versions of the problem.

In the general area of railroad planning and operations, a survey of optimization models for railroad operations is presented in [8], particularly in the context of train routing and scheduling. Other interesting surveys on railway optimization can be found in [4, 7, 9, 14]. Various railway features might be discussed (e.g., assignment of seats to passengers [5], connection reliability in railroad yards [19], train pathing and timetabling [24], maintenance requirement [28]). However, as this work focuses on refueling aspects, the literature review in the next subsection concentrates on papers which consider refueling opportunities.

2.3 Literature review covering problems with refueling aspects

In [36], the authors develop robust optimization methods to solve the locomotive routing problem (LRP). More precisely, given a schedule of trains, the *locomotive planning* (or scheduling) *problem* (LPP) is to determine the minimum cost assignment of locomotive types to trains that satisfies a number of business and operational constraints. Once this is done, the railroad has to determine the sequence of trains to which each locomotive is assigned by unit number so that it can be refueled and serviced as necessary. This problem is referred to as the *locomotive routing problem* (LRP). There are two major sets of constraints that need to be satisfied by each locomotive route: (1) locomotive refueling constraints, which require that every unit visit a refueling station at least once for every B_1 miles of travel (e.g. 900 miles), and (2) locomotive servicing constraints, which require that every unit visit a service station at least once for every B_2 miles of travel (e.g. 3000 miles). The output of the LPP is not directly implementable because the LPP does not consider these refueling and servicing constraints. In contrast, the LRP considers these constraints and its output is therefore implementable. Despite its importance in practice, reference [36] is the first attempt to solve this problem.

A locomotive scheduling problem (or locomotive assignment problem) is proposed in [1], which consists of assigning a set of locomotives to trains in a pre-planned train schedule, so as to ensure sufficient power supply. A locomotive routing problem that minimizes the cost for locomotive ownership and assignment over a railroad network is described in [35], where the fuel availability and service constraints are considered in an "aggregation-disaggregation" framework.

For other problems related to fuel cost optimization (airlines and railway), a linear programming model to minimize the total fuel cost for an airline flight schedule, subject to airplane capacity and supplier constraints, is developed in [33]. A multi-period capacitated inventory model is proposed in [39], where the goal consists of determining the refueling schedule of each airplane (along its predetermined route) to minimize the overall fuel costs. In [18], various refueling schedule problems are studied, with the objective to find the optimal travel route that minimizes fuel costs needed to travel from an origin to a destination, or to visit a set of predetermined points. Regarding fuel station selection, [20] and [22] present a greedy algorithm and a mixed integer program to find the optimal location of refueling facilities that maximizes the flow volume covered by the stations without running out of fuel. The model is extended in [21] by adding candidate facilities along network arcs, and further extended in [34] for capacitated fuel stations. A heuristic algorithm for the optimal refueling station locations to maximize the flow that can be refueled with a given number of facilities is developed in [25].

A discretionary service facility model, which determines the optimal location of facilities to maximize the possible potential customer flow, is presented in [2]. A flow-based set covering model to locate vehicle refueling stations that minimizes the total facility cost, while ensuring each passing vehicle to meet a fuel station before running out of fuel, is described in [37]. In [29], a linear mixed integer mathematical model is presented, which integrates not only fuel station location decisions, but also locomotive refueling schedule decisions. The proposed model helps railroads decide with which fuel stations to contract, and how each locomotive should purchase fuel along its predetermined shipment path, without running out of fuel, while minimizing the sum of fuel purchasing costs, shipment delay costs (due to refueling), and contracting charges.

In [31], a mixed integer program is formulated for problem (P1), and the formulation is enhanced by valid inequalities and domination rules. The authors tackled successfully a large set of linear instances, which will also be considered in this paper.

3 Local search algorithms

This section is dedicated to local search techniques. Two well-known methods are first presented within this context: the descent algorithm and tabu search. Then, in subsection 3.2, a new type of local search is proposed. This local search involves a learning process relying on a trail system within the framework of tabu search.

3.1 Descent algorithm and regular tabu search

Let f be an objective function to minimize. At each step of a *local search*, a *neighbor* solution s' is generated from the current solution s by performing a specific modification on s, called a *move*. All solutions obtained from s by performing a move are called *neighbor solutions* of s. The set of all neighbor solutions of s is denoted by $N(s)$. First, a local search needs an initial solution s_0 as input. Then, the algorithm generates

a sequence of solutions $s_1, s_2, \ldots$ in the search space such that $s_{r+1} \in N(s_r)$. The process is stopped for example when an optimal solution is found (if it is known), or when a time limit is reached. Famous local search algorithms are: the descent method, simulated annealing, variable neighborhood search, and tabu search. The descent method consists of performing the best improving move iteratively until no such move exists, in which case a local optimum has been reached. The *tabu search* uses the following strategy to avoid being trapped in a local optimum. When a move is performed from a current solution s_r to a neighbor solution s_{r+1}, it is forbidden to perform the reverse of that move during *tab* (parameter) iterations. Such forbidden moves are called *tabu* moves. Formally, the solution s_{r+1} is computed as $s_{r+1} = \arg \min_{s \in N'(s_r)} f(s)$, where $N'(s)$ is a subset of $N(s)$ (set of neighbor solutions of s) containing solutions which can be obtained from s by performing a non tabu move. Many variants of this basic tabu search algorithm can be found in [13]. More generally, the reader is referred to [12] for a recent book on metaheuristics, and to [40] for guidelines to efficiently design a metaheuristic.

3.2 Learning tabu search (LTS)

A tabu search with a learning process relying on a *trail* system is now proposed and denoted *LTS*. It is assumed that a solution s of the considered problem can be denoted $s = \{c_1, c_2, \ldots, c_n\}$, where c_i is the i^{th} characteristic (i.e. specific feature) of solution s. At each iteration, in order to generate a neighbor solution s' of the current solution s, a basic move $m(c_i)$ consists of adding to s – or removing from s – a characteristic c_i. The straightforward notation $s' = s + m(c_i)$ can thus be used. Then, it is forbidden (i.e. tabu) to perform the reverse move for *tab* (parameter) iterations. More precisely, if c_i is added to (resp. removed from) s, it is forbidden to remove it from (resp. add it to) s for *tab* iterations.

The trail system relies on the idea that if some combinations of characteristics *often* belong to *good* solutions during the search process, such combinations of characteristics should be favored when generating new solutions. If we consider combinations of two characteristics (for example), the trail $tr(c_i, c_j)$ associated with characteristics c_i and c_j indicates if it is a good idea to have both characteristics c_i and c_j in a solution, according to the observation of the history of the search. The trail $Tr(s, c_i)$ associated with a neighbor solution $s' = s + m(c_i)$ can be defined as $Tr(s, c_i) = \sum_{c_j \in s} tr(c_i, c_j)$. If $m(c_i)$ is an *add* move (i.e. feature c_i will be added to s to get s'), it is interesting to perform it if $Tr(s, c_i)$ is large. In contrast, if $m(c_i)$ is a *drop* move (i.e. feature c_i will be removed from s to get s'), it is interesting to perform it if $Tr(s, c_i)$ is small. Note that this can be generalized if more than two characteristics are considered (for triplets, the notation $tr(c_i, c_j, c_k)$ would for example be used).

Let us define a *cycle* of size I (parameter) as a sequence of I iterations of tabu search. Every I iterations, the best solution $\hat{s}$ of the last cycle is used to update the trail system as follows: $tr(c_i, c_j) = \rho \cdot tr(c_i, c_j) + \Delta tr(c_i, c_j)$, where $\Delta tr(c_i, c_j)$ is proportional to the quality of $\hat{s}$ if both characteristics c_i and c_j appear in $\hat{s}$, and $\Delta tr(c_i, c_j) = 0$ otherwise. The parameter ρ belongs to $[0, 1]$ and simulates trail evaporation. Variants would be to make $\Delta tr(c_i, c_j)$ proportional to: (1) the number of times both characteristics c_i and c_j appear in the solutions visited within the cycle, or (2) the average quality of the solutions of the cycle having both

characteristics c_i and c_j.

At each iteration of *LTS*, we first pick a random set A of non tabu neighbor solutions obtained with add moves, and a random set D of non tabu neighbor solutions obtained with drop moves, such that $|A| = |D|$. Note that the size of A is an important and sensitive parameter to tune. Let A_q be the set containing the q (parameter) solutions of A with the largest trail values, and let D_q be the set containing the q solutions of D with the smallest trail values. The performed move among $A_q \cup D_q$ is then the best one according to the objective function f of the considered problem. This technique is particularly relevant if it is cumbersome to evaluate a solution with objective function f, as f is only used to evaluate a sample of solutions which rank highly according to the trail function Tr (which does not require much computation). In other words, at each iteration of *LTS*, the objective function f (or its associated incremental computation) is only used to evaluate $2 \cdot q$ *promising* neighbor solutions according to Tr, which is faster than to evaluate a *random* sample of $2 \cdot q$ neighbor solutions.

In order to help visiting new regions of the solution space, the following diversification mechanism (denoted *DIV*) is introduced, relying on a different use of the trail system. Conceptually, the trail system is used to favor good moves which were often performed in the previous cycles. In contrast, to diversify the search, we propose here to perform good moves which were not often performed in the previous cycles. More precisely, let $s' = s+m(c_i)$. If $m(c_i)$ is an add (resp. drop) move, then it is diversifying to perform it if $Tr(s, c_i)$ is small (resp. large). Hence, each iteration of *DIV* is performed as above, but the set A_q (resp. D_q) respectively contains the q solutions of A (resp. D) with the smallest (resp. largest) trail values. The mechanism *DIV* relies on two sensitive parameters t_1 and t_2: it is triggered if t_1 (parameter) iterations without improving $s^\star$ (the best encountered solution during the search) have been performed, and it is performed until one of the following conditions is satisfied: (1) $s^\star$ has been improved; (2) a sequence of t_2 (parameter) iterations of *DIV* have been performed. The complete *LTS* method is described in Algorithm 1.

We now discuss how *LTS* compares with standard local searches and more specifically with classical ant algorithms (see [3] and [10] for recent overviews). We point out that within the local search framework, there already exist some mechanisms favoring good moves (resp. attributes) which were frequently performed (resp. encountered) in the past of the search process. Such mechanisms are however very differently managed (e.g., see [12]). The technique proposed in this paper is original because of the joint use of the following two elements: (1) a trail system managed with an evaporation component and a reinforcement component; (2) combinations of characteristics are handled (instead of individually considering each characteristic).

Note that that the concept of trail system also exists in ant algorithms, but it is managed very differently in *LTS*. In contrast with the ant algorithms: (1) *LTS* is a local search dealing with a single solution, and not a method based on the management of a population of solutions; (2) fewer parameters have to be tuned in *LTS*; (3) *LTS* does not require the use of a normalization process; (4) *LTS* performs each decision quickly and with an aggressive manner; (5) *LTS* does not use jointly (but sequentially) information based on the history of the search (known in the ant community as the *trail* system) and on the short term profit (known in the ant community as the *heuristic information* or the *visibility* or the *greedy force*).

In subsection 4.2, we describe how *LTS* can be successfully adapted to the TAP. Note that it might be well-suited for other combinatorial optimization problems, like the following ones. (1) In the *graph coloring problem*, a characteristic could be the belonging of a vertex to a color class, and the trail system could be a matrix [vertex, vertex] indicating that two vertices should get the same color. (2) In the *vehicle routing problem*, a characteristic could be the assignment of a client to a truck, and the trail system could be a matrix [client, client] indicating that two clients should be served by the same truck. (3) In the *facility location problem*, a characteristic could be the positioning of a facility (e.g., a distribution center, a central warehouse, a plant) on a location, and the trail system could be a matrix [facility, facility] indicating that two facilities should be close to each other.

Algorithm 1 *LTS*: Learning Tabu Search

Initialization

1. construct an initial solution s;
2. set $s^\star = s$ and $f^\star = f(s)$;
3. set $iter = 0$ (iteration counter);
4. set *DIV* as off (i.e. not active);

While a time limit is not reached, **do**:

1. from s, generate a set A (resp. D) of non tabu neighbor solutions obtained with add (resp. drop) moves, such that $|A| = |D|$;
2. if *DIV* is off, identify the set $A_q \subseteq A$ (resp. $D_q \subseteq D$) containing the solutions with the q largest (resp. smallest) trail values;
3. if *DIV* is on, identify the set $A_q \subseteq A$ (resp. $D_q \subseteq D$) containing the solutions with the q smallest (resp. largest) trail values;
4. select the neighbor solution: set $s' = arg \min_{s'' \in A_q \cup D_q} f(s'')$;
5. update the current solution: set $s = s'$;
6. update the tabu status: the reverse move is forbidden for *tab* iterations;
7. update the best encountered solution: if $f(s) < f^\star$, set $s^\star = s$ and $f^\star = f(s)$;
8. set $iter = iter + 1$;
9. update the status of *DIV*:
   (a) switch on *DIV* if p_1 iterations without improving $s^\star$ have been performed;
   (b) switch off *DIV* if it has been performed during p_2 consecutive iterations, or if $s^\star$ has been improved within the current application of *DIV*;
10. if $(iter \mod I) = 0$, update the trail system with the best solution $\hat{s}$ among the last I iterations;

Return $s^\star$

4 Local search algorithms for the TAP

In this section, three solution methods are proposed for the TAP and thus for the whole problem: a descent local search *DLS*, a regular tabu search *TS*, and a learning tabu search *LTS*. In each method, an initial solution is generated by running the *FDP algorithm* with an infinite number of trucks on each yard and all stops considered as open (in order to avoid any bias). In such a case, for each yard y and each day d, the output of the *FDP algorithm* is a quantity $Q_{y,d}$ of fuel (in gallons) that must be dispensed to the locomotives. Let Q_y be the maximum (among the days) quantity of delivered fuel at this yard (i.e. $Q_y = \max_d Q_{y,d}$), and C be the given capacity of a truck (in gallons/day). Then we set the number of trucks on each yard as $\lceil Q_y/C \rceil$. Finally, we keep open only the stops where the corresponding locomotive is refueled.

4.1 Descent algorithm and tabu search

The common features of *DLS* and *TS* are the following. A solution is a pair (T, S), where T and S are vectors of respective sizes equal to the number of yards and the total number of stops (among all locomotives). The j^{th} component of T is the number of contracted trucks at yard y_j, and the i^{th} component of S indicates if the i^{th} stop in S is open or closed. From a current solution (T, S), a move simply consists of adding a contracted truck to a yard (*add move*), or in removing a contracted truck from a yard (*drop move*). When a move is performed, it is evaluated by the use of the *FDP algorithm*. During the evaluation, all the costs are considered: the refueling costs for the used gallons of fuel, the fixed refueling costs and the contracting costs of the trucks. At each iteration, $|A|$ add moves and $|D|$ drop moves are generated. Preliminary tuning experiments led to the following parameter setting: $|A| = |D| = 10$. Note that more refined strategies (e.g., dynamic ways of updating $|A|$ and $|D|$) were also tested to tune these parameters, but since they did not clearly improve our results, we preferred to keep the algorithms as simple as possible. The same remark actually holds for all parameters introduced in this paper. The resulting descent local search *DLS* for the TAP is presented in Algorithm 2.

In contrast with *DLS*, the proposed tabu search *TS* does not stop when it reaches a local optimum, but returns the best encountered solution within a predefined time limit. When an add (resp. drop) move is performed on yard y_j, it is forbidden to consider yard y_j for a drop (resp. add) move for *tab* iterations (parameter tuned to 10).

Algorithm 2 *DLS*: Descent Local Search for the TAP

Construct an initial solution (T, S);

While a local optimum is not reached, **do**:

1. In solution (T, S), randomly choose a set D containing yards for which drop moves are allowed; for any yard $y_j \in D$ and from (T, S), remove a truck from it, open all the stops and apply the *FDP algorithm* to evaluate such a drop candidate move;
2. In solution (T, S), randomly choose a set A of yards; for any yard $y_j \in A$ and from (T, S), add a truck to it, open all the stops and apply the *FDP algorithm* to evaluate such an add candidate move;
3. From (T, S), perform the best move among the $\mid A \cup D \mid$ above candidate moves, and rename the resulting solution as (T, S);

4.2 Learning tabu search

In order to adapt *LTS* to the considered problem, we mainly have to define what is a *characteristic* and to set a *learning* process based on a *trail* system.

A characteristic is simply a truck on a given yard. Note that there is no a priori upper bound on the number of trucks on each yard. Consequently, the set of characteristics may be infinite. However, the number of *distinct* characteristics is finite and equal to the number of yards. Since two trucks on the same yard are perfectly identical and interchangeable, the trail function can naturally be expressed on the finite set made of all pairs of yards. In our implementation, the trail $tr(x, y)$ associated with two yards x and y aims to indicate if it is a good idea (according to the objective function) to have trucks on both yards x and y in the same solution.

All trails are initialized to 0, and at the end of each iteration of *LTS*, they are globally updated with $\hat{s}$ (the best solution of the cycle) as follows (with ρ tuned to 0.9): $tr(x, y) = \rho \cdot tr(x, y) + \Delta tr(x, y)$, where $\Delta tr(x, y)$ is the number of trucks on x and y, computed only if $\hat{s}$ has at least one truck on both x and y (it is 0 otherwise). Let $Tr(s, x) = \sum_{y \in s} tr(x, y)$ be the *trail value* associated to yard x in solution s. If $Tr(s, x)$ is large, it is interesting to select a move which adds a truck to x (because the history of the search seems to indicate that it is a good idea to have trucks on yard x, as well as on the yards which already contain trucks in the current solution s). Otherwise, it is interesting to select a move which removes a truck from x.

We now describe more precisely how to select a move at each iteration. Recall that in *DLS* and in *TS*, the performed move is the best among the ones in the set $\mid A \cup D \mid$, with $|A| = |D| = 10$. In *LTS*, we first randomly choose two sets A and D of size 20. Then, let A_q (resp. D_q) be the subset of A (resp. D) containing the q (parameter fixed to 10) moves with the largest (resp. smallest) trail values. Note that computing the trail value of a move is much quicker to compute than the value of the resulting neighbor solution, as the FDP algorithm is requested for it. The performed move among $A_q \cup D_q$ is the best one according to the objective function of the problem (i.e. the sum of the costs). Therefore, for *DLS*, *TS* and *LTS*, the performed move has the best objective function value among a sample of 20 evaluated solutions.

This will allow to better measure the impact of the trail system on the search. Note that if DIV is open (i.e. the diversification mechanism is activated), A_q (resp. D_q) is the subset of A (resp. D) containing the q moves with the smallest (resp. largest) trail values.

5 Results

In this section, we first present the set of considered (P1) instances in subsection 5.1, and the way we have generated the instances for (P2) and (P3). Then, we discuss the obtained results for (P1), (P2) and (P3) in subsections 5.2, 5.3 and 5.4, respectively. We tested the proposed algorithms on an Intel Quad-core i7 @ 3.4 GHz with 8 GB DDR3 of RAM memory. A common time limit $TT = 60$ minutes is imposed to each proposed method (i.e. DLS, TS and LTS). Note that DLS is restarted from scratch each time a local optimum is found, in order to be able to apply it during TT minutes, and thus to perform a fair comparison with the two other metaheuristics. In addition, all the provided results are average values over 5 runs.

Problem (P1) was initially motivated by an INFORMS contest involving 31 research teams. The approaches of the three best teams are described in [15] and are also discussed in [32]. The winners of the contest (Mor Kaspi and Tal Raviv) formulated the problem as a mixed integer linear program (MILP), for which the original formulation was tightened with nine families of valid inequalities and symmetry breaking constraints. This MILP approach was published in [31]. The authors proposed a set of 56 additional instances, which will be tackled in subsection 5.2. Note that other MILP formulations exist (e.g., [23]).

5.1 Description of the instances

The considered 57 instances are presented in Table 2, and the way they have been generated is detailed in [31]. As exposed in [31], the instance labeled I1 was introduced as a challenge in [15]. It is characterized by a fourteen day cyclic schedule, and the schedule of each locomotive is divided into fourteen runs. The instance consists of 74 yards, 214 locomotives and 5,264 stops. The fixed refueling cost is 250$ for all stops, the truck contracting cost is 8,000$, the tanker truck capacity is 25,000 gallons per day, and the locomotive tank capacity is 4,500 gallons. The new instances proposed in [31] (labeled from I2 to I57) are based on: (1) random networks with a number N of yards; (2) twelve-day cyclic schedules with a number n of stops. Instances I2 to I49 consist of 8 instances for each combination of values for N in $\{75, 100, 120\}$, n in $\{5,000; 10,000; 30,000\}$, while instances I50 to I57 are larger instances with $N = 196$ and $n = 130$. For each (N, n) combination, the 8 instances correspond to each combination of values for: the contracting costs (5,000$ and 7,000$), tanker truck capacities (25,000 and 50,000 gallons), and locomotive tank capacities (3,500 and 5,500 gallons). The fixed refueling cost was set to 250$ for each stop.

The (P2) instances have been generated based on the following idea. If several trucks from the same company are contracted for the same yard, the company is likely to propose discounted prices for that yard. More formally, for the linear instances presented in Table 2, let CC_y be the contracting cost (in \$/(truck·period))

associated with yard y. Each (P2) instance is generated from its corresponding (P1) instance as follows. If m trucks are located at yard y, then the corresponding contracting cost (in \$/(truck·period)) becomes the nonlinear function $CC_y = (0.8)^m \cdot CC_y$ (where $m \geq 2$).

Let k be the number of stops (excluding the origin) of a train. The (P3) instances have been generated based on the following idea. Firstly, the clients would probably not mind if one or two refueling stops are encountered along the route of a train. But from a certain threshold (fixed to two in this paper), a dissatisfaction D will appear, which will augment in a quadratic fashion with k. For this reason, the following penalty function is proposed, where c is a parameter: $D(k) = c \cdot (k-1) \cdot (k-2)$ if $k > 0$, and $D(k) = 0$ if $k = 0$. One can remark that $D(k) = 0$ as well if $k \in \{1, 2\}$. Parameter c was chosen to be equal to the fixed cost associated with a refueling action. Therefore, for a given route, the dissatisfaction penalty will dominate the fixed refueling costs if $k > 2$.

Table 2: Description of the (P1) instances.

Instance	**Number N of yards**	**Number n of stops**	**Tanker Capacity**	**Contracting cost**	**Locomotive capacity**
I1	73	5,250	25,000	4,000	4,500
I2	75	10,000	25,000	5,000	3,500
I3	75	10,000	25,000	5,000	5,500
I4	75	10,000	25,000	7,000	3,500
I5	75	10,000	25,000	7,000	5,500
I6	75	10,000	50,000	5,000	3,500
I7	75	10,000	50,000	5,000	5,500
I8	75	10,000	50,000	7,000	3,500
I9	75	10,000	50,000	7,000	5,500
I10	75	5,000	25,000	5,000	3,500
I11	75	5,000	25,000	5,000	5,500
I12	75	5,000	25,000	7,000	3,500
I13	75	5,000	25,000	7,000	5,500
I14	75	5,000	50,000	5,000	3,500
I15	75	5,000	50,000	5,000	5,500
I16	75	5,000	50,000	7,000	3,500
I17	75	5,000	50,000	7,000	5,500
I18	100	10,000	25,000	5,000	3,500
I19	100	10,000	25,000	5,000	5,500
I20	100	10,000	25,000	7,000	3,500
I21	100	10,000	25,000	7,000	5,500
I22	100	10,000	50,000	5,000	3,500
I23	100	10,000	50,000	5,000	5,500
I24	100	10,000	50,000	7,000	3,500
I25	100	10,000	50,000	7,000	5,500
I26	100	5,000	25,000	5,000	3,500
I27	100	5,000	25,000	5,000	5,500
I28	100	5,000	25,000	7,000	3,500
I29	100	5,000	25,000	7,000	5,500
I30	100	5,000	50,000	5,000	3,500
I31	100	5,000	50,000	5,000	5,500
I32	100	5,000	50,000	7,000	3,500
I33	100	5,000	50,000	7,000	5,500
I34	120	10,000	25,000	5,000	3,500
I35	120	10,000	25,000	5,000	5,500
I36	120	10,000	25,000	7,000	3,500
I37	120	10,000	25,000	7,000	5,500
I38	120	10,000	50,000	5,000	3,500
I39	120	10,000	50,000	5,000	5,500
I40	120	10,000	50,000	7,000	3,500
I41	120	10,000	50,000	7,000	5,500
I42	120	5,000	25,000	5,000	3,500
I43	120	5,000	25,000	5,000	5,500
I44	120	5,000	25,000	7,000	3,500
I45	120	5,000	25,000	7,000	5,500
I46	120	5,000	50,000	5,000	3,500
I47	120	5,000	50,000	5,000	5,500
I48	120	5,000	50,000	7,000	3,500
I49	120	5,000	50,000	7,000	5,500
I50	196	30,000	25,000	5,000	3,500
I51	196	30,000	25,000	5,000	5,500
I52	196	30,000	25,000	7,000	3,500
I53	196	30,000	25,000	7,000	5,500
I54	196	30,000	50,000	5,000	3,500
I55	196	30,000	50,000	5,000	5,500
I56	196	30,000	50,000	7,000	3,500
I57	196	30,000	50,000	7,000	5,500

5.2 Results on the (P1) instances

Firstly, we would like to mention the results obtained by the MILP approach proposed in [31]. Their MILP approach relies on ILOG-CPLEX 12.1 and was tested on a Xeon X3450 2.67 GHz workstation with 16 GB of RAM. The smallest instance I1 (i.e. the one of the competition) was optimally solved within 24 hours. This was achieved by reinforcing their formulation with nine problem-specific families of valid inequalities, which dramatically improved both the lower bound and the quality of the obtained solution. On average, they observed that adding these cuts to the initial formulation permitted to close 99.34% of the remaining optimality gap on the smaller instances (with number of yards $n \in \{75, 100, 120\}$), and 91.10% on the large instances (with $n = 196$). They set the time limit to one hour for the small instances and three hours for the large ones.

Our results on the 57 linear instances are presented in Table 3. For each instance indicated in column 1, the following pieces of information are given. The solution value obtained by the MILP approach is given in column 2. Note that the times needed to get such values are not detailed in [31]. Column 3 gives the percentage gap of the MILP approach according to LB (lower bound returned by CPLEX). The percentage gap of DLS according to MILP is indicated in column 4. The same information is provided for TS and LTS in columns 5 and 6, respectively. Average results are given in the last line, and indicate that after 60 minutes of computation, the three proposed algorithms have an average gap of approximately 2% from the MILP approach. We can remark that LTS is slightly better than DLS and TS: on average, LTS saves 4083\$ (resp. 3850\$) when compared to DLS (resp. TS). We can observe that the MILP approach finds the optimal solution on all but 17 instances, and when the optimal solution is not found, the gap according to LB is very small. This occurs on almost all the large instances, which indicates again that metaheuristics would be appropriate for much larger instances of (P1).

In order to better compare the behavior of the proposed algorithms, Figure 1 is helpful. For DLS, TS and LTS, Figure 1 compares the evolution of the best encountered solution value during 60 minutes (again averaged over 5 runs). On the one hand, we can observe that LTS clearly outperforms the other two methods from the beginning to the end of the hour of computation. This indicates that the learning process (i.e. the trail system) introduced to TS to derive LTS is relevant. On the other hand, we also remark that TS is better than DLS in the first 30 minutes, and then both methods have a comparable behavior.

Table 3: Results on the (P1) instances.

Instance	**MILP**	**MILP vs** LB	***DLS* vs MILP**	***TS* vs MILP**	***LTS* vs MILP**
I1	11399670.88	0.00%	0.32%	0.38%	0.31%
I2	19060186.08	0.09%	1.39%	1.36%	1.37%
I3	18655818.52	0.00%	1.24%	1.23%	1.20%
I4	19135933.94	0.08%	1.79%	1.79%	1.80%
I5	18729818.52	0.00%	1.55%	1.57%	1.55%
I6	19037307.6	0.01%	1.32%	1.30%	1.31%
I7	18627101.52	0.00%	1.14%	1.13%	1.14%
I8	19109525.2	0.01%	1.72%	1.71%	1.71%
I9	18691101.52	0.00%	1.46%	1.45%	1.48%
I10	9569687.48	0.00%	2.26%	2.27%	2.28%
I11	9352917.24	0.00%	1.68%	1.66%	1.66%
I12	9639946.52	0.00%	3.00%	3.02%	3.03%
I13	9405898.56	0.00%	2.20%	2.19%	2.20%
I14	9569436.44	0.00%	2.24%	2.27%	2.26%
I15	9347917.24	0.00%	1.68%	1.68%	1.69%
I16	9638742.52	0.00%	3.00%	3.03%	3.02%
I17	9400731.76	0.00%	2.22%	2.21%	2.22%
I18	18486857.64	0.01%	1.53%	1.57%	1.59%
I19	18115582.24	0.00%	1.40%	1.37%	1.37%
I20	18576870.68	0.02%	1.98%	2.00%	2.04%
I21	18190541.16	0.00%	1.77%	1.73%	1.74%
I22	18481809.8	0.01%	1.49%	1.48%	1.49%
I23	18107425.32	0.00%	1.38%	1.34%	1.33%
I24	18569628.16	0.02%	1.88%	1.90%	1.89%
I25	18178384.24	0.00%	1.70%	1.68%	1.67%
I26	9513404.44	0.00%	2.34%	2.35%	2.38%
I27	9307231.88	0.00%	1.86%	1.85%	1.89%
I28	9587819.24	0.00%	3.13%	3.17%	3.18%
I29	9367219.32	0.00%	2.45%	2.45%	2.48%
I30	9513404.436	0.00%	2.35%	2.36%	2.39%
I31	9307231.88	0.00%	1.88%	1.89%	1.88%
I32	9587819.24	0.00%	3.14%	3.15%	3.18%
I33	9367219.31	0.00%	2.47%	2.48%	2.48%
I34	19118270.68	0.01%	1.75%	1.70%	1.71%
I35	18727561.28	0.00%	1.42%	1.39%	1.39%
I36	19220470.08	0.01%	2.22%	2.17%	2.20%
I37	18809509.08	0.00%	1.84%	1.78%	1.78%
I38	19115473.16	0.00%	1.66%	1.65%	1.64%
I39	18720151.48	0.00%	1.44%	1.42%	1.42%
I40	19216606.56	0.00%	2.17%	2.10%	2.11%
I41	18800151.48	0.00%	1.86%	1.82%	1.82%
I42	9480955.8	0.00%	2.85%	2.92%	2.91%
I43	9283343.119	0.00%	2.24%	2.24%	2.24%
I44	9568439.8	0.00%	3.80%	3.92%	3.90%
I45	9355576.56	0.00%	2.94%	2.96%	2.96%
I46	9480955.8	0.00%	2.83%	2.91%	2.87%
I47	9283343.12	0.00%	2.24%	2.24%	2.24%
I48	9568439.8	0.00%	3.80%	3.85%	3.92%
I49	9355576.56	0.00%	2.94%	2.98%	2.96%
I50	53748956.06	0.28%	1.69%	1.64%	1.50%
I51	52591915.2	0.08%	1.77%	1.71%	1.56%
I52	53979085.06	0.35%	2.15%	2.14%	1.94%
I53	52781608.56	0.09%	2.32%	2.28%	2.04%
I54	53596524.72	0.04%	1.52%	1.50%	1.41%
I55	52495703.16	0.00%	1.40%	1.42%	1.28%
I56	53782299.4	0.05%	1.94%	1.92%	1.76%
I57	52658448.16	0.01%	1.79%	1.80%	1.59%
Average	**19532799.21**	**0.0205263%**	**2.0271817%**	**2.0259959%**	**2.0061365%**

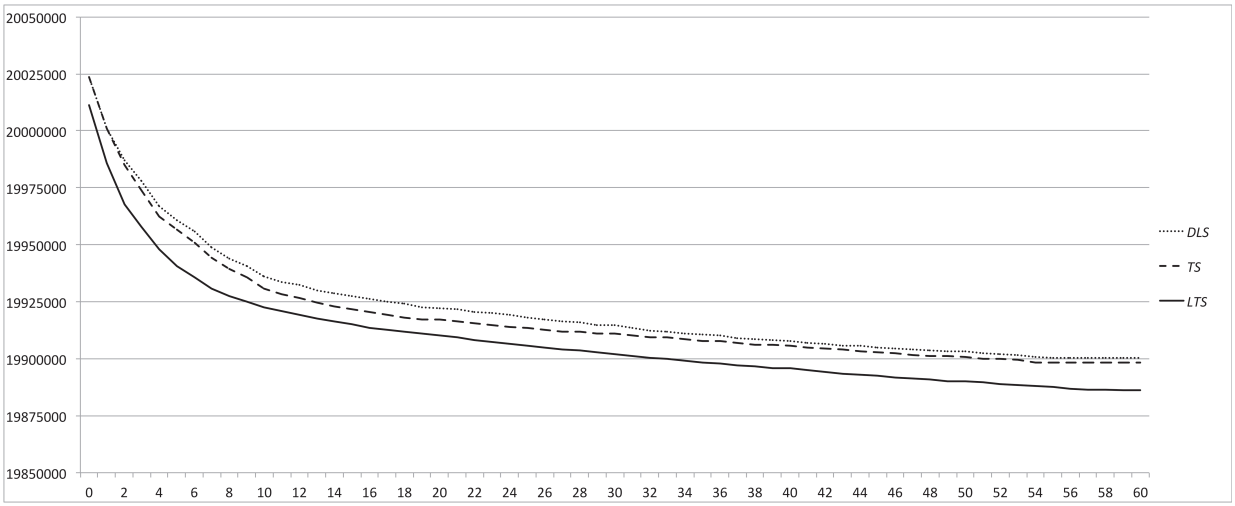

Figure 1: Evolution of *DLS*, *TS* and *LTS* on the (P1) instances

5.3 Results on the (P2) instances

The results on the (P2) instances are presented in Table 4. The considered set of 15 instances is a representative sample generated from the 57 linear instances. For each instance indicated in column 1, the following information is given. Column 2 gives the best value $f^\star$ obtained with the best run of all the compared metaheuristics. The percentage gap of *DLS* according to $f^\star$ is indicated in column 3. The same information is provided for *TS* and *LTS* in columns 4 and 5, respectively. Average results are given in the last line. Again, we can remark the superiority of *LTS* over the other methods.

Table 4: Results on the (P2) instances.

Instance	$f^\star$	***DLS***	***TS***	***LTS***
I1	11435319.27	0.00%	7.44%	0.16%
I2	19234059.01	2.35%	0.99%	0.00%
I9	18874080.52	2.20%	0.00%	0.83%
I10	9713546.672	0.00%	1.48%	2.19%
I17	9535836.048	0.58%	0.21%	0.00%
I18	18683351.5	1.42%	0.00%	1.94%
I25	18390821.3	2.06%	0.54%	0.00%
I26	9660556.888	0.25%	0.00%	2.38%
I32	9517126.984	0.00%	1.30%	1.50%
I33	19340993.46	3.68%	0.00%	0.92%
I41	19035034.74	1.33%	0.00%	0.03%
I42	9656106.576	0.00%	5.16%	6.24%
I49	9532777.6	0.00%	2.10%	2.59%
I50	54269407.81	12.24%	12.89%	0.00%
I57	53266695.02	13.09%	11.36%	0.00%
Average		**2.61%**	**2.90%**	**1.25%**

Figure 2 is similar to Figure 1 but is associated with the (P2) instances. The same observations as before can be made: *LTS* outperforms *TS* and *DLS*, *TS* is better than *DLS* during half an hour, *TS* and *DLS* are comparable during the second half hour.

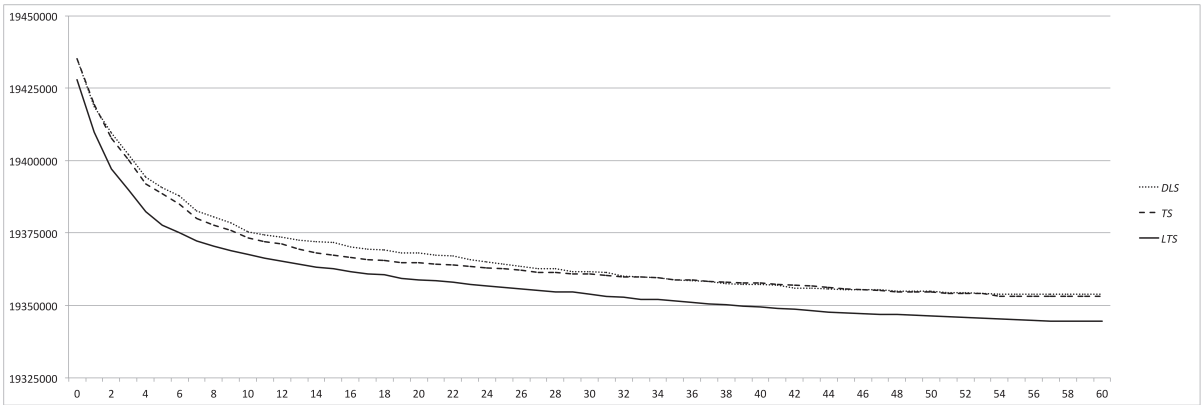

Figure 2: Evolution of *DLS*, *TS* and *LTS* on the (P2) instances

5.4 Results on the (P3) instances

The results on the (P3) instances are presented in Table 5, which has the same format as Table 4. The considered set of 15 instances is again a representative sample generated from the 57 linear instances. Again, we can remark the average superiority of *LTS* over the other methods, particularly on the larger instances.

Table 5: Results on the (P3) instances.

Instance	$f^\star$	***DLS***	***TS***	***LTS***
I1	11446570.05	0.000000000%	0.023256084%	0.036615335%
I2	19324160.48	0.019589943%	0.000000000%	0.028009030%
I9	18975380.52	0.000000000%	0.022408742%	0.025809179%
I10	9792158.584	0.005651379%	0.000000000%	0.035485945%
I17	9621090.32	0.016110440%	0.000000000%	0.038799262%
I18	18782861.56	0.000000000%	0.000323699%	0.007360220%
I25	18512637.3	0.025547522%	0.000000000%	0.003511137%
I26	9741347.08	0.000000000%	0.014050788%	0.029679181%
I32	9628320.68	0.007389201%	0.000000000%	0.010688136%
I33	19453901.54	0.038045386%	0.000000000%	0.002928431%
I41	19167618.12	0.042874727%	0.009196260%	0.000000000%
I42	9760601.12	0.003710427%	0.000000000%	0.022935800%
I49	9661933.68	0.000000000%	0.025584609%	0.031378067%
I50	54488723.66	0.170712540%	0.149455943%	0.000000000%
I57	53505311.55	0.191943385%	0.116490405%	0.000000000%
Average		**0.034772%**	**0.024051%**	**0.018213%**

Figure 3 is similar to Figure 2 but is associated with the (P3) instances. Again, *LTS* outperforms *TS* and

DLS. But in contrast with the previous problems, *DLS* and *TS* are comparable during the first 5 minutes, and then *TS* outperforms *DLS*.

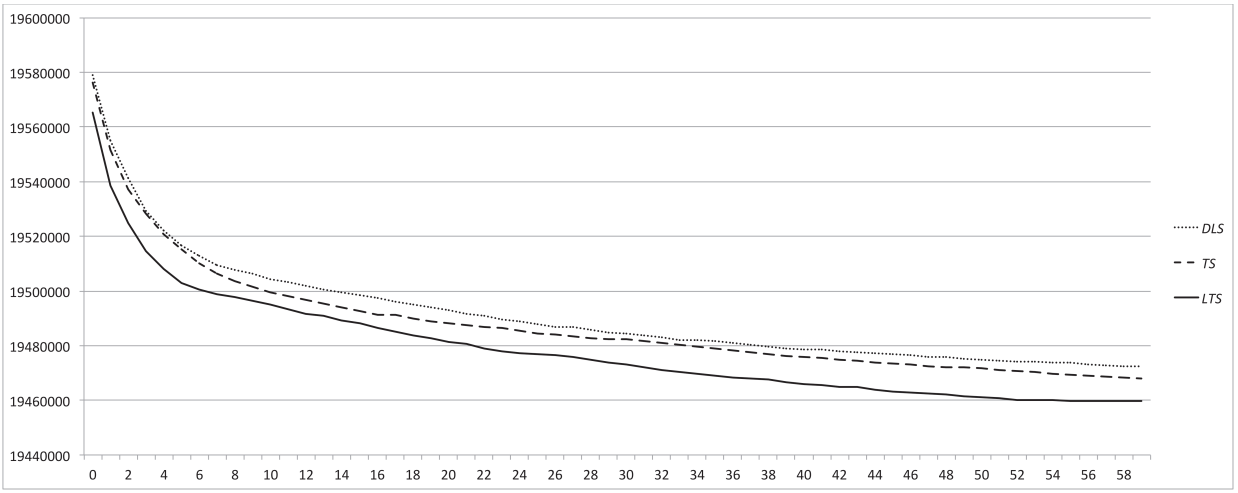

Figure 3: Evolution of *DLS*, *TS* and *LTS* on the (P3) instances

6 Conclusion

A refueling problem in a railway network is considered in this paper. We decompose it in two optimization levels. The top level consists of determining the number of refueling trucks at each yard. It is called the truck assignment problem (TAP). The bottom level of the problem consists of determining the refueling plan of each locomotive. It is called the fuel distribution problem (FDP) and it can be efficiently solved with the *FDP algorithm* proposed in a previous study. The goal consists of minimizing the sum of various types of cost components, while respecting several operational constraints.

Three variants of the problem are considered, which are denoted (P1), (P2) and (P3). In (P1), all the costs are linear and there is a maximum number of times a train can stop to be refueled (stop constraint). (P2) is an extension of (P1) where nonlinear discounted prices can be obtained from fuel suppliers when several trucks are contracted for a same yard. (P3) is derived from (P1) by relaxing the stop constraint. Instead, the refueling stops are penalized in the objective function in a nonlinear fashion. Three methods are proposed for the TAP: a descent local search *DLS*, a tabu search *TS*, and a learning tabu search *LTS*. The latter is a new type of local search approach, involving a learning process which relies on a trail system.

We have observed that *LTS* outperforms the other considered solution methods. On the (P1) linear instances, *LTS* has an average gap of 2% from the optimum provided by a refined MILP approach relying on CPLEX 12.1. On the (P2) and (P3) instances – for which the MILP approach cannot be used because nonlinear costs have to be tackled – *LTS* shows promising results. An important advantage of *LTS* is its flexibility: in contrast with a MILP approach, *LTS* can be easily adapted if new costs and constraints, including nonlinear

ones, are added to the problem.

Among future avenues of research, we mention the following. Firstly, dual information from the FDP network flow optimal solution could be used to better guide the TAP local search. Secondly, extensions of the problem may be considered by taking into account other types of costs and constraints, or adding other optimization levels (e.g., the sorting of the trains on the tracks of each yard). Finally, as *LTS* turned out to be the most promising approach, it would be interesting to apply it to other problems.

References

[1] R. K. Ahuja, J. Liu, J. B. Orlin, D. Sharma, and L. A. Shughart. Solving real-life locomotive scheduling problems. *Transportation Science*, 39 (4):503 – 517, 2005.

[2] O. Berman, R. C. Larson, and N. Fouska. Optimal location of discretionary service facilities. *Transportation Science*, 26 (3):201 – 211, 1992.

[3] C. Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373, 2005.

[4] M.R. Bussieck, T. Winter, and U.T. Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming*, 79:415 – 444, 1997.

[5] V. Cacchiani, A. Caprara, and P. Toth. Solving a Real-World Train Unit Assignment Problem. *Mathematical Programming*, 124:207 – 231, 2010.

[6] V. Cacchiani and P. Toth. Nominal and Robust Train Timetabling Problems. *European Journal of Operational Research*, 219 (3):727 – 737, 2012.

[7] A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth. *Transportation, Handbooks in Operations Research and Management Science*, volume 14, chapter Passenger railway optimization, pages 129 – 187. Elsevier, Amsterdam, 2007.

[8] J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science*, 32 (4):380 – 404, 1998.

[9] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. *Network Routing, Handbooks in Operations Research and Management Science*, volume 8, chapter Time constrained routing and scheduling, pages 35 – 139. Elsevier, Amsterdam, 1995.

[10] M. Dorigo, M. Birattari, and T. Stuetzle. Ant colony optimization – artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1 (4):28–39, 2006.

[11] D. Gelders, M. Galetzka, J. P. Verckens, and E. Seydel. Showing results? An analysis of the perceptions of internal and external stakeholders of the public performance communication by the Belgian and Dutch Railways. *Government Information Quarterly*, 25:221 – 238, 2008.

[12] M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, 2010.

[13] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, 1997.

[14] D. Huisman, L.G. Kroon, R.M. Lentink, and M.J.C.M. Vromans. Operations research in passenger railway transportation. *Statistica Neerlandica*, 59:467 – 497, 2005.

[15] INFORMS RAS Competition. http://www.informs.org/community/ras/problem-solving-competition/2010-ras-competition. 2010.

[16] C. D. Ittner and D. F. Larcker. Innovations in performance measurement: Trends and research implications. *Journal of Management Accounting Research*, 20:205 – 238, 1998.

[17] M. D. Johnson and C. Fornell. A framework for comparing customer satisfaction across individuals and product categories. *Journal of Economic Psychology*, 12:267 – 286, 1991.

[18] S. Khuller, A. Malekian, and M. Julian. To fill or not to fill: The gas station problem. *Lecture Notes in Computer Science*, 4698:534 – 545, 2007.

[19] E. R. Kraft. Priority-Based Classification for Improving Connection Reliability in Railroad Yards. *Journal of the Transportation Research Forum*, 56 (1):107 – 119, 2002.

[20] M. Kuby and S. Lim. The flow-refueling location problem for alternative-fuel vehicles. *Socio-Economic Planning Sciences*, 39 (2):125 – 145, 2005.

[21] M. Kuby and S. Lim. Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs. *Networks and Spatial Economics*, 7 (2):129 – 152, 2007.

[22] M. Kuby, S. Lim, and C. Upchurch. Dispersion of nodes added to a network. *Geographical Analysis*, 37 (4):384 – 409, 2005.

[23] V. Prem Kumar and M. Bierlaire. Optimizing Fueling Decisions for Locomotives in Railroad Networks. *Transportation Science*, 2013.

[24] Y. Lee and C.-Y. Chen. A heuristic for the train pathing and timetabling problem. *Transportation Research B*, 43:837 – 851, 2009.

[25] S. Lim and M. Kuby. Heuristic algorithms for siting alternative-fuel stations using the flow-refueling location model. *European Journal of Operational Research*, 204 (1):51 – 61, 2010.

[26] R. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway track allocation: models and methods. *OR spectrum*, 33 (4):843 – 883, 2011.

[27] M. Marinov, I. Sahin, S. Ricci, and G. Vasic-Franklin. Railway operations, time-tabling and control. *Research in Transportation Economics*, 41:59 – 75, 2013.

[28] G. Maroti and L. Kroon. Maintenance routing for train units: The transition model. *Transportation Science*, 39:518 – 525, 2005.

[29] S. M. Nourbakhsh and Y. Ouyang. Optimal fueling strategies for locomotive fleets in railroad networks. *Transportation Research Part B*, 44 (8-9):1104 – 1114, 2010.

[30] D. Otley. Performance management: A framework for management control systems research. *Management Accounting Research*, 10:363 – 382, 1999.

[31] T. Raviv and M. Kaspi. The locomotive fleet fueling problem. *Operations Research Letters*, 40:39 – 45, 2012.

[32] D. Schindl and N. Zufferey. Solution methods for fuel supply of trains. *Information Systems and Operational Research*, 51 (1):22 – 29, 2013.

[33] J. S. Stroup and R. D. Wollmer. A fuel management model for the airline industry. *Operations Research*, 40 (2):229 – 237, 1991.

[34] C. Upchurch, M. Kuby, and S. Lim. A capacitated model for location of alternative-fuel stations. *Geographical Analysis*, 41 (1):127 – 148, 2009.

[35] B. Vaidyanathan, R. K. Ahuja, J. Liu, and L. A. Shughart. Real-life locomotive planning: New formulations and computational results. *Transportation Research Part B*, 42(2):147 – 168, 2008.

[36] B. Vaidyanathan, R. K. Ahuja, and J. B. Orlin. The locomotive routing problem. *Transportation Science*, 42 (4):492 – 507, 2008.

[37] Y. W. Wang and C. C. Lin. Locating road-vehicle refueling stations. *Transportation Research Part E*, 45 (5):821 – 829, 2009.

[38] V. Zeithaml, M. J. Bitner, and D. D. Gremler. *Services marketing. Integrating customer focus across the firm*. New York: McGraw-Hill Companies, 2006.

[39] P. P. Zouein, W. R. Abillama, and E. Tohme. A multiple period capacitated inventory model for airline fuel management: a case study. *Journal of the Operational Research Society*, 53(4):379 – 386, 2002.

[40] N. Zufferey. Metaheuristics: some Principles for an Efficient Design. *Computer Technology and Applications*, 3 (6):446 – 462, 2012.