CIRRELT
1971 – 2021

CIRRELT-2022-09

Learning Tabu Search Algorithms: A Scheduling Application

Nazgol Niroumandrad
Nadia Lahrichi
Andrea Lodi

April 2022

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Learning Tabu Search Algorithms: A Scheduling Application

Nazgol Niroumandrad*, Nadia Lahrichi, Andrea Lodi

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematics and Industrial Engineering, Polytechnique Montréal

Abstract. Metaheuristics are widely recognized as efficient approaches for many combinatorial problems. Studies to improve the performance of metaheuristics have increasingly relied on the use of various methods either combining different metaheuristics or methods originating outside of the metaheuristic field. This paper presents a learning algorithm to improve tabu search by reducing its search space and the evaluation effort. We study the performance of a learning tabu search algorithm using classification methods in an attempt to select moves through the search space more wisely. The experimental results demonstrate the benefit of using a learning mechanism under deterministic and stochastic conditions.

Keywords: Learning tabu search, learning metaheuristics, combinatorial problems, logistic regression, decision trees

---

* Corresponding author: nazgol.niroumandrad@polymtl.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2022

1. Introduction

In recent years, many studies have presented algorithms that combine various methods to improve the performance of metaheuristics, some originating from other optimization algorithms and others from outside the field of metaheuristic. These approaches are referred to as hybrid metaheuristics. Choosing an adequate combination of methods and algorithmic concepts can be a critical factor in achieving high-quality performance when solving hard optimization problems. Developing an effective hybrid approach is generally difficult, and the literature shows that it is nontrivial to generalize a particular hybrid algorithm. An algorithm may work well for some problems but perform poorly for others.

Metaheuristics explore the search space by an iterative process, where they generate a large amount of dynamic data. Some of the information is collected and used. For example, tabu search uses frequencies and tabu information in its search process. However, we argue that metaheuristics do not fully exploit the explicit knowledge discovered during the process to guide the space wisely and make the search process more efficient. The main idea of this paper is to demonstrate that metaheuristics, and more specifically tabu search, can behave more intelligently and efficiently by collecting and using the data generated during the search process. We believe that more advanced techniques such as machine learning (ML) can be utilized to extract valuable knowledge. This knowledge will guide and enhance search performance by facilitating a better exploration of the search space and eventually finding solutions that are unattainable without these learning algorithms, ultimately yielding better solutions for complex problems.

Machine learning techniques can help improve an algorithm in two ways [1]. During the search process in metaheuristics, learning can be used to build approximations and to replace some heavy computations. The learning methods can also explore the space of decision variables and improve any insufficient understanding of some algorithmic decisions. The goal is therefore to extract knowledge from the behavior of the best-performing variables.

Since tabu search (TS) has been successful in solving many hard optimization problems, we chose to study the effects of learning methods on the performance of the tabu search algorithm. We are proposing a novel learning tabu search algorithm using classification methods in the context of a physician scheduling problem. To the best of our knowledge, there is no comprehensive study on integrating ML techniques into TS to explore the search space. Most studies on learning metaheuristics evolve in the context of clustering or intensification and diversification strategies. For example, [2] studied a clustering method that depends on frequency-based memory applied to high-quality solutions. Instead, [3] studied a diversification-based learning method.

Tabu search has also been hybridized with numerous methods, most commonly with evolutionary algorithms. The hybridization can either be an evolutionary method that employs tabu search to generate

improved solutions, or a tabu search method that benefits by evolutionary search as diversification or intensification strategies. References [4]–[6] are examples of this evolutionary/tabu search hybridization.

Although there are a number of studies on improving the performance of TS, the literature lacks a comprehensive study on how ML techniques can be integrated into TS to enhance its search procedure. This paper provides an intensive study on the use of ML techniques in the design of TS, which does not rely on a simple diversification/intensification strategy to improve the search procedure. We believe this paper is beneficial for both academic and industry experts engaged in solving hard combinatorial problems. Our proposed method is used to study a scheduling problem and results are compared with those of [7]. Since the original tabu search presented in [7] proved to be very efficient and optimized under deterministic conditions, this paper focuses on improving, by learning, the performance of the TS in a stochastic environment.

The rest of the paper is organized as follows. In Section 2, we present the algorithm and review the related studies. We present the baselines for the classification methods in Section 3 and explain the proposed method in details. We describe the instance generation and present the results in Section 4 and Section 5 provides concluding remarks.

2. Algorithm definition and related literature

This section provides a brief introduction to Tabu Search. This is followed by a literature review.

2.1. Algorithm definition - Tabu search

In a nutshell, tabu search [8] is an iterative procedure that starts from an initial solution x_0 (possibly infeasible). From each current solution x ($x = x_0$ at the beginning of the procedure), it moves to a neighbor solution x'. The neighborhood is defined by all solutions that can be reached from $x \in X$, where X is the solution space, after applying a specific move. Let us denote this neighborhood by $N(x)$. The next solution x' is the best non-tabu solution in the neighborhood $N(x)$ (an exception is made if a move is tabu but it improves upon the current best solution x^*, i.e., through the so-called aspiration criteria). The function $f(x)$ is defined to evaluate each solution. To prevent cycling, a tabu list (T_{list}) containing attributes of recently visited solutions (or attributes of moves) is maintained. The associated solutions cannot be revisited for a specific number of iterations. Various strategies can be applied to search the neighborhood solutions. The moves and strategies to search the solution space are the main ingredients of tabu search methods. Hence, the adequate combination and sequence of them have a great impact on the quality of the results. Tabu search is used in multiple applications and is adapted to handle uncertainty. This is commonly done by considering several scenarios, typically generated using historical data or a probability

distribution. Instead of moving to the best solution in the neighborhood (deterministic environment), the best solution in average (stochastic environment) is preferred.

A naïve approach to choose a solution x' is evaluating and analyzing all possibilities, which is be computationally expensive. Instead, we can characterize the neighborhood using experiments. In particular, the guiding principle of our investigation is that using a learning method can help us to find good solutions faster. ML techniques allow us to extract knowledge from good solutions and use it to generate even better solutions. This knowledge can be in the form of a set of rules or patterns [9].

2.2. Literature

The study of learning within metaheuristics has not been given the attention it deserves. Conversely, leveraging machine learning techniques to solve combinatorial problems received a lot of attention recently. More precisely, [10] and [11] demonstrated that employing machine learning during the search process can improve the performance of heuristic algorithms. Studies that employ machine learning to enhance heuristics have pursued the following objectives:

- Algorithm selection and analysis [12],
- Learning generative models of solutions [13],
- Learning evaluation functions [14],
- Understanding the search space [15], [16].

For instance, [14] described algorithms that improve the search performance by learning an evaluation function that predicts the outcome of a local search algorithm from features of states visited during the search. Other studies on learning evaluation functions are presented in [17], [18] and [19]. Along the same subject, [20] is another example of a recent study on learning metaheuristics. This study proposed a learning variable neighborhood search (LVNS) that identifies quality features simultaneously. This information is then used to guide the search towards promising areas of the solution space. The LVNS learning mechanism relies on a set of trails, where the algorithm measures the quality of the solutions.

Machine learning was also employed to prune the search space of large-scale optimization problems by developing pre-processing techniques [21], [22]. Some other studies used ML-based methods to directly predict a high-quality solution [23], [24]. Moreover, [25] and [26] provided a review of studies where metaheuristic algorithms benefited from machine learning and the potential future work.

Building upon these previous studies, we propose a learning tabu search algorithm enhanced with classification methods to guide the search through the solution space of hard combinatorial problems. We observe that the emphasis on adaptive memory within tabu search represents the nature of its learning

mechanism. However, most previous works focused on clustering [2] or intensification/diversification [3] strategies. In metaheuristics, intensification and diversification strategies play important roles in the quality of solutions. In a recent study, [27] presented a relaxation-adaptive memory programming algorithm on a resource-constrained project scheduling problem. In that approach, primal-dual relationships help to effectively explore the interplay between intensification, diversification, and learning (IDL). The algorithm is designed to integrate the current most effective Lagrangian-based heuristic with a simple tabu search. The authors aimed to present a study on the IDL relationship when dual information is added to the search. In [28], the authors presented a learning tabu search algorithm for a truck allocation problem in which they considered a trail system (in the ant colony optimization, a trail system is inspired from the pheromone trails of ants to mark a path) for the combination of important characteristics. In this study, a diversification mechanism is introduced to help visit new solution space regions. The authors proposed diversifying the search by performing "good" moves that were not often performed in the previous cycles.

There are also numerous studies found in the literature on improving the performance of tabu search. In particular, [29] emphasized selecting particular attributes of solutions and determining conditions that help to find the prohibited moves, in order to produce high-quality solutions. Following that work, [30] and [31] employed a balance among more commonly used attributes. The presented computational experiments showed that considering these attributes can significantly outperform all other methods. These outcomes underpin researchers' ongoing strategy of identifying attributes that lead to more effective methods. However, to the best of our knowledge, our study in the favor of learning the characteristics of the search space during the tabu search algorithm's search procedure is a novel contribution to the literature. We aim to use classification models to fill this gap. Our contribution centers on how ML helps to learn the best neighborhoods to build or change a solution during the search process.

3. Learning tabu search algorithm

Before getting into the details of our proposed learning TS (L-TS), we first give a brief explanation of, and the baselines for, our chosen classification methods, namely logistic regression (LR) and decision trees (DT). Next, we present our L-TS followed by important characteristics of a case study that we chose in order to evaluate L-TS performance.

3.1. The basics of LR and DT

In this paper, we study the impact of deploying logistic regression as a multinomial classification method, and the decision trees learning method, to guide the tabu search. Here, we give a basic overview of these methods, and define some terms used throughout the paper.

The logistic regression model is one of the most important models that can be applied to analyze categorical data, and the multinomial logistic regression model is a simple extension of the binomial one. It uses predictive analysis to explain the relationship between one nominal dependent variable and one or more independent variables.

In a binomial logistic regression model, let's assume n denote the number of predictors for a binary response Y by the features matrix $X = (x_{ij})_{n\times d}$, where each column represents a feature and each row represents an observation. The numerical value of x_{ij} denotes the measurement of a specific feature j ($j = 1, ..., d$) in a specific observation i ($i = 1, ..., n$). Given a training dataset $(x_i, y_i)_{i=1}^{n}$, where $x_i = (x_{i1}, x_{i2}, ..., x_{id})$ represents the vector feature values of the i^{th} observation, and $y_i \in \{0, 1\}$ for $i = 1, ..., n$, where $y_i = 1$ if observation i is in class 1 and 0 if i is in class 2. In general, the aim is to classify the new observation and identify the relevant features with high classification accuracy. Assume $p(x_i)$ represents the probability for observation i when $y_i = 1$, $p(x_i) = Pr(y_i = 1|x_i)$. Logistic regression determines this probability by learning, from a training set, a vector of weights, w, and a bias term [32]. More precisely, each weight w_j is a real number, and is associated with feature j of observation i. The weight w_j represents how important that input feature is to the classification decision. The bias term b, also called the intercept, is another real number that is added to the weighted inputs. Moreover, z in equation (1) expresses the weighted sum of the evidence for the class, which can also be represented by the dot product notation in (2):

$$z_i = \sum_{j=1}^{d} w_j x_{ij} + b \tag{1}$$

$$z_i = w \cdot x_i + b \tag{2}$$

To create a probability, we pass z through the sigmoid (logistic) function, $\sigma(z)$,

$$y_i = \sigma(z_i) = \frac{1}{1 + e^{-z_i}} \tag{3}$$

Now, we have an algorithm that, given an observation x_i, can compute the probability $P(y_i = 1|x_i)$. Hence, we can classify observation x_i based on:

$$\hat{y}_i = \begin{cases} 1, & \text{if } P(y_i = 1|x_i) > \alpha \\ 0, & \text{otherwise.} \end{cases}$$

where α is a parameter (e.g., 0.5) and the vector w is learned by iteratively minimizing the classification error over the observations in the training set.

Now, we turn to the decision tree, a supervised machine learning technique that develops a tree of decision paths from training data [33]. A decision tree is a predictive model and a mapping from observations

of an event (training data) to conclusions about its target value (label). Decision trees classify data using a set of hierarchical decisions based on features. In the tree structure, leaves represent classifications (also referred to as labels), non-leaf nodes are features, and branches represent combinations of features that lead to the classifications.

Decision trees are a widely-used predictive model for many reasons, such as

- They are easy to interpret and explain to non-technical audiences,
- They require minimal effort for data preparation,
- They can handle all kind of data, including numerical and categorical data, and
- The training data do not require any assumptions of linearity and nonlinear relationships between parameters do not affect tree performances.

As for logistic regression, the tree is constructed by iteratively minimizing a loss function corresponding to misclassification in the training set [34].

Using a predictive method significantly reduces the number of evaluations in the tabu search, and both logistic regression and decision trees are powerful classification methods, each having its own advantages.

3.2. Learning tabu search

We employ the learning procedure in two phases of training and application. The training phase is divided into two stages itself, T_1 and T_2. The first stage, T_1, starts with the original TS to collect data related to the structure of the search space. After collecting enough data, T_2 begins training the logistic regression model for a specific number of iterations. These two stages are presented in Figure 1. I_{st} represents the iteration that phase T_2 (training) starts and I_{end} represents the end of training. After collecting enough data and training the model, the application phase will start in which an action will be taken based on the prediction of the last trained model and after validating the elements of a tabu list (T_{list}). We provide more details below.

The learning methods are very sensitive to the input information. Thus, extracting the features that have the greatest impact on the solution space is the first and most important step. These features represent important characteristics of the solutions space and moves. Table 1 presents these features in different categories.

Let x be the current solution, x^* the best known solution, and x' the next solution. Each solution is represented as a matrix of integers. In the category of cost improvement, we are first considering the improvement in cost which is presented as $\Delta_x = f(x') - f(x)$. The first feature is the value of Δ_x and the second reflects if the solution is improved ($\Delta_x > 0$ in the context of maximizing). In the tabu category,

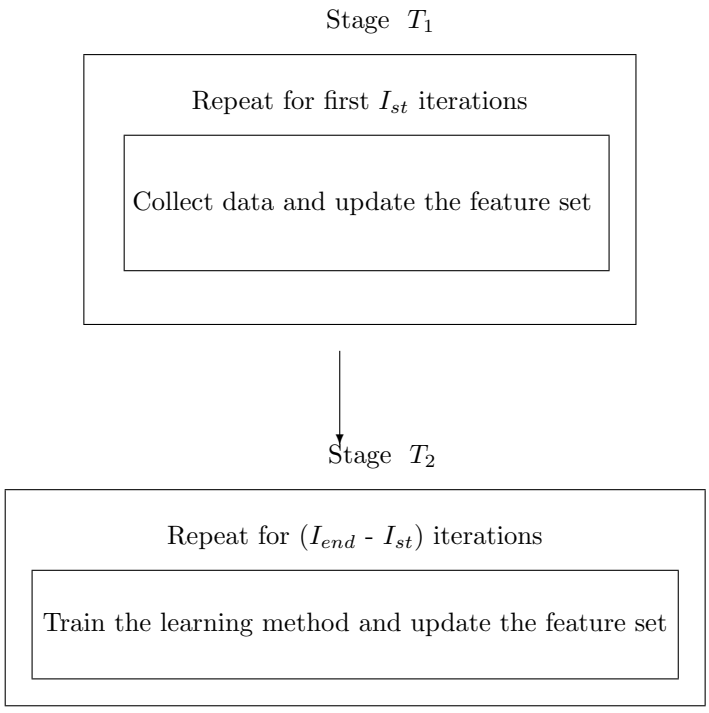


Figure 1: Training phases

we have a binary variable that determines if the accepted move belongs to the T_{list} and whether it has already been visited along with the tabu value for the move, meaning when it will be free and can be considered again. The frequency of the move is also considered, which indicates how many times the move has been visited so far. The next category represents the characteristics of solution. First, the solution x itself. A binary matrix (D^*) denotes the difference between the obtained solution and the last best known solution. A 1 indicates if the corresponding entry is equal, 0 otherwise. Also, a binary matrix (D') denotes the difference between the obtained solution and the previous solution, with the same meaning for 1's and 0's. The final category is related to the move characteristics, i.e., the attractiveness of the move and a trail matrix of the moves. These features were inspired by the recent work of [20]. Here, a trail system influences the decisions made by the ants in the Ant Algorithms and the notion of a move attractiveness shows that moves with high attractiveness values have more chance to be performed. The same list of features is used for the decision trees model.

As tabu search can be used in a stochastic environment, the set of input features in the stochastic

	Input		Output
Category	Features	Format	Label
Cost improvement	Δ_x	$\mathbb{R}$	Observed Output
	$\Delta_x > 0$	$\mathbb{B}$	
Tabu	$x_i \in T_{list}$	$\mathbb{B}$	
	$\lvert t_i \rvert = T_{list}(x)$	$\mathbb{Z}$	
	$Freq_{\_I^D}$	$\mathbb{Z}$	
Solution	x	$M_{\mathbb{Z}}$	
	D^*	$M_{\mathbb{B}}$	
	D'	$M_{\mathbb{B}}$	
Move	Attractiveness	$\mathbb{Q}$	
	Trail of the moves	$M_{\mathbb{Q}}$	

Table 1: Input/Output features for the training model

learning algorithm is modified by considering the average of $f(x)$ over all scenarios at each iteration. In other words, we need to find the move that is the best in average over all the scenarios. Thus, our objective is to find the solution $x' \in N_v(x)$ that minimizes the average solution over all scenarios $\left(\frac{\sum_w f_w(x'_w)}{|W|}\right)$.

The L-TS model differs from the original TS mostly in the search space. We seek to reduce the number of evaluations in the search process. Thus, we are predicting the subset of promising moves, and evaluate only these neighbors instead of evaluating every possible neighbors of $N(x)$.

3.3. Case study: Physician Scheduling

We studied the impact of our proposed learning mechanism on a physician scheduling problem that was previously studied in [7]. We will introduce the main characteristics of the problem, however, we invite those interested to learn more about the problem to review [7] in its entirety. The goal is to find a weekly cyclic schedule for physicians in a radiotherapy department and to assign the arriving patients to the best possible available specialist for their cancer type. Figure 2 shows a schematic diagram of the typical stages in the pre-treatment phase in a radiotherapy center (for patients with different types of cancer, some stages may vary).

In most radiotherapy centers, the physician schedule is task based. Each day is divided into one or multiple periods, and each period is dedicated to a single task [35], [36]. In practice, all of the tasks are scheduled each week, and some tasks are repeated. The goal is to minimize the duration of the pre-treatment phase for patients. This is defined as the time from the patient's arrival day to the day the final task is finished before the treatment starts. This is usually one week for curative patients, while for palliative patients, it should be three days at most. Scheduling may also take into account physician preferences.

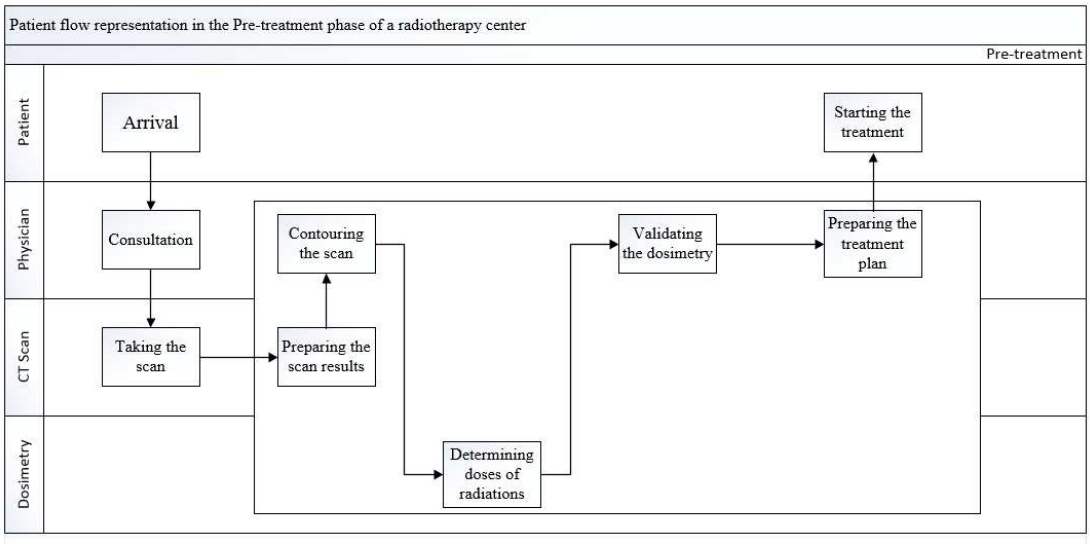


Figure 2: Stages of pre-treatment phase in a radiotherapy center.

The decision variables are:

$$p_{ti}^{d} = \begin{cases} 1, & \text{if physician } i \text{ performs task } t \text{ on day } d \\ 0, & \text{otherwise} \end{cases}$$

$$q_{ij} = \begin{cases} 1, & \text{if patient } j \text{ is assigned to physician } i \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ijt}^{d} = \begin{cases} 1, & \text{if patient } j \text{ undergoes task } t \text{ on day } d \text{ when assigned to physician } i \\ 0, & \text{otherwise} \end{cases}$$

The objective is twofold: we maximize the physician preferences and minimize the pre-treatment duration. Each physician has a preference for each day for each task (s_{ti}^{d}), and the pre-treatment duration is defined as the time from the arrival day a_j to the day the final task $|T^{o_j}|$ is performed.

$$\text{maximize} \quad \sum_{i \in I} \left(\sum_{t \in T} \sum_{d \in D} S_{ti}^{d} p_{ti}^{d} - \sum_{j \in J} \sum_{d \in D} d z_{ij|T^{o_j}|}^{d} - a_j \right) \tag{4}$$

Like most large combinatorial problems, the exact methods cannot always solve the optimization problem in a reasonable amount of time. Thus, we solved this problem with a tabu search metaheuristic. Sn exact mathematical model for this problem was presented in [7]. historical data or a probability distribution. This strategy is common when uncertainty must be considered.

To summarize the tabu search algorithm for our problem, the search space is explored with three types of moves that are closely related to the decision variables (which task should be assigned to a physician on which day, and which patient should be assigned to which physician). Namely,

- *Move 1*: We seek the best sequence of task assignments by swapping the assignments of a physician's current schedule from one day to another in the planning horizon.
- *Move 2*: We introduce more flexibility in a schedule by changing the repeated task. In the simple form of the problem, where the planning horizon is five day, and we have four main tasks, each physician repeats just one task. This task could be replaced by any other. In the case of two time slots, the neighborhood to explore is larger since it is likely that all the tasks are repeated. Five tasks can be allocated to each of the ten time slots, and we evaluate all these options.
- *Move 3*: The third move focuses on the second decision variable, i.e., the assignment of patients to physicians. We explore all possible re-assignments.

Having described the learning algorithm in Section 3.2, we will now demonstrate its application in the physician scheduling problem. Preliminary experiments show that applying the learning algorithm on *Move 1* has the greatest impact on the performance of our original TS. In the L-TS method, we consider a subset of neighbors to evaluate, instead of evaluating all possibilities at each iteration. We are predicting the physician to which *Move 1* should be applied. Since data is the essential component of any learning algorithm, we collect behavioral data in solution space (stage T_1) before we can employ the learning methods (stage T_2 of training phase along with the application phase) within the TS. The L-TS algorithm starts with the original TS to collect the necessary data for I_{st}^D iterations, as demonstrated in Figure 1. We train the learning method for a specific number of iterations (stage T_2), evaluate the result of the learning algorithm and update the set of input features to encourage the method to search more promising regions. In the application phase, we use the last trained model at iteration $I_{end}^D - 1$ to identify (by prediction) promising moves to build $N'(x) \subset N(x)$ (Note that superscript D stands for deterministic and S will be used for the stochastic environment). *Move 2* and *Move 3* are used to diversify the search and the total procedure ends when the stopping criterion ($Stop_{max}$) is reached. Our criterion is 1h of CPU time. Details of the parameter initialization step are documented in Section 4. Algorithm 1 presents the pseudocode of our learning tabu search algorithm.

Algorithm 1 Learning tabu search algorithm

Generate initial solution x^0**:**

a) Assign each task (at least once) to physicians randomly; and,

b) for each patient: assign to the physician who performs task A on the arrival day if the quota is not reached, and to any other available physician otherwise;

Calculate initial cost $f(x^0)$ using function 4 defined in Section 3.3. Create an empty T_{list};
x, $x^* \leftarrow x^0$, $f(x) = f(x^*) = f(x^0)$
it, it, counter* $\leftarrow$ *0*
while stopping criterion is not reached **do**
  while counter $< I_v^D$ **do**
    if $it < I_{st}^D$ **then**
      for $\forall i \in I$ **do**
        for $\forall d \in 1, ..., 10n - 1$ **do**
          for $\forall d' \in d + 1, ..., 10n$ **do**
            Swap task on d and task on d'
          end for
        end for
      end for
    end if
    if $I_{st}^D < it < I_{end}^D$ **then**
      Employ the learning algorithm $\rightarrow$ **Output:** $\exists i \in I$
      for $\forall d \in 1, ..., 10n - 1$ **do**
        for $\forall d' \in d + 1, ..., 10n$ **do**
          Swap task on d and task on d'
        end for
      end for
    end if
    x $\leftarrow x'$ and $f(x) = f(x')$.
    if $f(x') > f(x^*)$ **then**
      $x^* \leftarrow x$, $f(x^*) = f(x)$ and $it^* = it$.
    end if
    Update T_{list}
    it $\leftarrow$ it + 1, counter $\leftarrow$ counter + 1
  end while
  counter $\leftarrow 0$
  Diversification
  while counter $< I_2^D$ **do**
    for neighborhood $v = \{2, 3\}$ **do**
      move v
      x $\leftarrow x'$ and $f(x) = f(x')$.
      if $f(x') > f(x^*)$ **then**
        $x^* \leftarrow x$, $f(x^*) = f(x)$ and $it^* = it$.
      end if
      Update T_{list}
      it $\leftarrow$ it + 1, counter $\leftarrow$ counter + 1
    end for
  end while
  $v = 1$, counter $\leftarrow 0$
end while

4. Experiments

In this section, we answer some important questions: Is learning possible during the TS procedure? Which method of learning (online vs offline; these concepts will be explained later in this section) has the greatest impact? Also, how should we choose the parameters and features of each method?

4.1. Experimental Setup

We compare the results with a benchmark previously published in [7] for both deterministic and stochastic environments in which the performance was compared with CPLEX. We refer to this previous work as original TS in the remainder of the paper. We use 21 generated pseudo-real instances where we vary the number of new patients arriving each week and the number of available physicians. Number of physicians varies from 6 to 10 and number of patients from 7 to 60, ranging from small instances to real-world applications. Each instance is labeled $pr-(\#$ of physicians, $\#$ of patients). In the deterministic case, we select one scenario to obtain a typical schedule, and in the stochastic situation, we consider a subset of W for different scenarios. We refer the readers to [7] for more details on instance generation.

4.2. Experimental Results

We report and analyze results in deterministic and stochastic cases based on 1) performance, 2) efficiency, 3) scalability and 4) sensitivity of the algorithm.

- **Performance** We evaluated the performance of the learning algorithm by comparing the cost value defined in equation (4).
- **Efficiency** The efficiency of the algorithm was validated by measuring the primal integral [37], comparing the number of calculations performed at each iteration and the time needed for each iteration.
- **Scalability** The scalability of the algorithm was evaluated by testing in small, medium and large-scale instances.
- **Sensitivity** The sensitivity analysis was performed by evaluating the impact of different training features and methods (online/offline) on the performance of the algorithm.

All results were compared and evaluated with respect to the original TS method and a random approach (in which $N'(x) \subset N(x)$ was randomly chosen) to test the performance of the L-TS method. Comparing these three approaches helps us to see the performance of each and confirm the advantage of choosing TS

over the random approach and choosing L-TS over TS. It shows that we are learning during the process and the results are not achieved by chance.

Among the four mentioned criteria used to analyze the results, the first three, namely, performance, efficiency and scalability, are discussed in Sections 4.2.1 and 4.2.3. The fourth criterion, sensitivity, is analyzed in Section 4.2.2.

4.2.1. Experimental performance on the Deterministic case

In the rest, we refer to online learning as our primary method of learning. In this method, the training phase includes two stages, T_1 and T_2. First, we wish to validate our L-TS algorithm and determine the value of the parameters, i.e., the size of the T_{list}, the number of iterations in each neighborhood, the iteration to start the training phase and application phase. The values tested are all related to the size of the instances (i.e., number of patients, number of physicians and number of time blocks). For the deterministic case, we use $I_1^D = 1$, $I_2^D = 2|J| + |I| + |5n|$, $I_3^D = 1$, $I_{st}^D = 3\sqrt{|J| \times |I| \times |D|}$, $I_{end}^D = 2|I_{st}^D|$, $Stop_{max} = 1h$, and we set $\theta^D = 2|J| + |I| + |D|$, while the reason of their choices are discussed in Section 4.2.2.

Table 2: Results for generated instances in the deterministic case

	Tests	GAP - Best (%)			GAP - Avg (%)		
		Random	L-TS-LR	L-TS-DT	Random	L-TS-LR	L-TS-DT
Small	pr-(6,7)	0.5	**-1.0**	**-0.7**	2.3	**-0.7**	0.0
	pr-(6,9)	1.0	0.8	1.3	1.7	**-1.4**	**-0.1**
	pr-(6,11)	2.4	0.3	**-0.8**	2.1	**-0.2**	0.4
	pr-(6,12)	2.9	0.5	1.1	3.1	0.3	0.5
	pr-(8,7)	2.7	**-0.7**	0.7	2.4	**-0.1**	0.7
	pr-(8,9)	2.2	**-0.4**	0.5	2.5	**-0.2**	0.1
	pr-(10,7)	1.5	0.1	0.4	2.3	0.1	0.7
Medium	pr-(6,20)	1.2	0.9	0.6	3.6	1.3	0.5
	pr-(8,11)	3.0	0.6	0.6	2.9	0.4	0.5
	pr-(8,12)	1.5	0.0	**-0.6**	2.3	0.0	0.1
	pr-(8,20)	3.3	0.8	**-0.8**	3.7	**-0.1**	0.8
	pr-(10,9)	2.3	0.3	0.4	2.1	0.1	0.2
	pr-(10,11)	1.7	**-0.3**	**-0.1**	2.2	0.2	0.7
	pr-(10,12)	1.5	0.1	0.6	3.0	0.4	0.5
Large	pr-(6,40)	2.6	**-4.0**	**-3.5**	4.5	**-0.4**	2.1
	pr-(6,60)	9.4	**-0.7**	1.4	5.8	**-3.6**	**-8.7**
	pr-(8,40)	2.3	**-0.5**	0.0	4.4	1.7	1.7
	pr-(8,60)	8.2	1.3	5.2	8.5	3.3	5.3
	pr-(10,20)	3.9	0.2	1.2	4.1	0.4	1.2
	pr-(10,40)	3.4	0.5	0.2	4.4	0.1	2.3
	pr-(10,60)	1.5	0.4	**-1.1**	6.2	2.2	1.5
	Average:	2.81	**-0.03**	0.31	3.53	0.18	0.52

Table 2 compares the cost values of different methods and the gap columns show improvements with

respect to the original TS. In this table, "GAP - Best" compares the best solution obtained from 10 different runs of each method and represents the improvements from the best solution obtained from the original TS. Conversely, "GAP - Avg" represents the average values from ten different runs. A negative value in the GAP columns indicates that the learning stochastic tabu search has improved the solution on average. It can be observed that logistic regression succeeded in slightly improving the cost, by 0.03% on average. We see more improvement in large instances where the algorithm has more flexibility. However, both learning methods, L-TS-LR and L-TS-DT, perform much better than the Random algorithm.

The value of the cost function alone cannot represent the advantage of using each approach. Hence, we measured the primal integral value for all methods to compare the progress of the primal bound's convergence towards the best-known solution over the entire solving time. Figure 3 illustrates this measure for all instances.

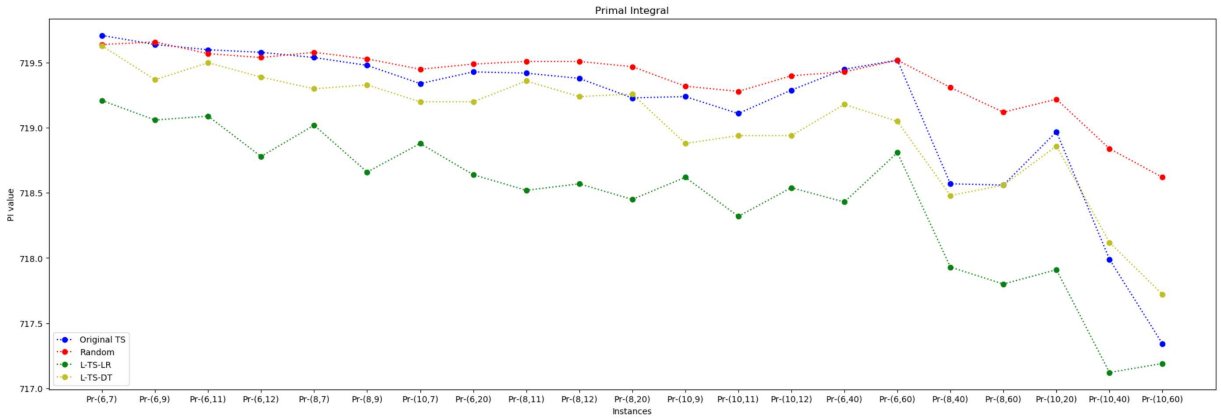


Figure 3: Comparing the convergence speed for different methods in the deterministic case

The idea of the primal integral [37] is that, the smaller the primal integral value is, the better is the expected quality of the solution if we stop the solver at an arbitrary point in time. It can be observed from Figure 3, that the logistic regression method has better primal integral values than the other methods for all instances. Again, this figure shows that both learning methods, presented by green and yellow lines, perform better than the original TS and Random algorithms. Figure 4 compares the number of evaluations at each iteration, the total number of evaluations and the total number of iterations until we reach the stopping criterion for different methods. It clearly shows a decrease in the number of calculations in the learning methods. We can also observe that the random method performs fewer evaluations but requires more iterations (compared with the learning methods) to find the solution, due to its poor performance. This behavior was expected from the random approach since it is evaluating $N'(x)$, a subset of $N(x)$, randomly.

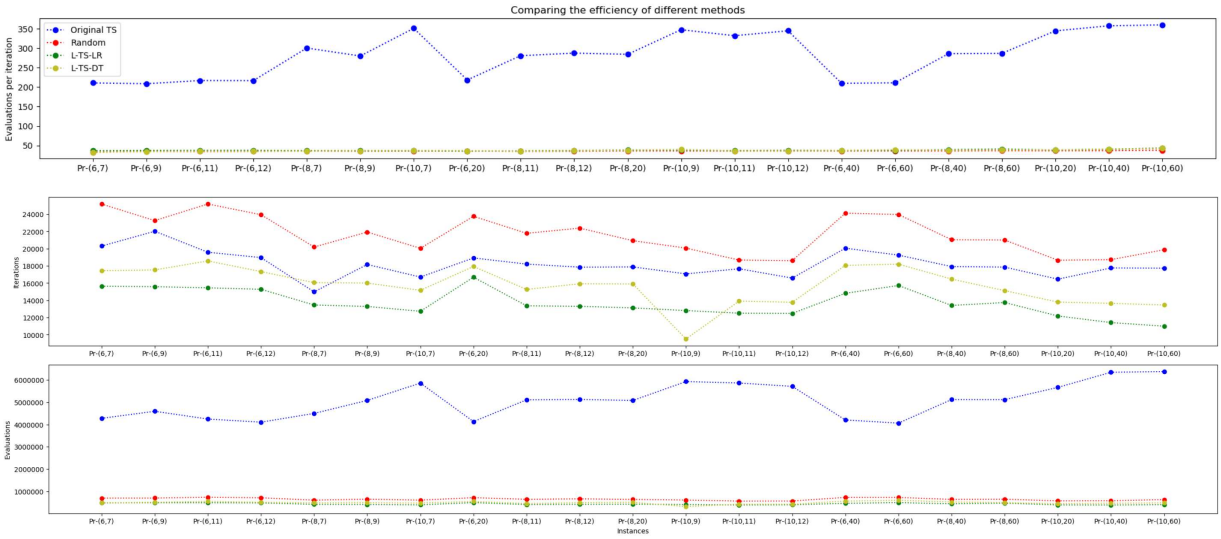
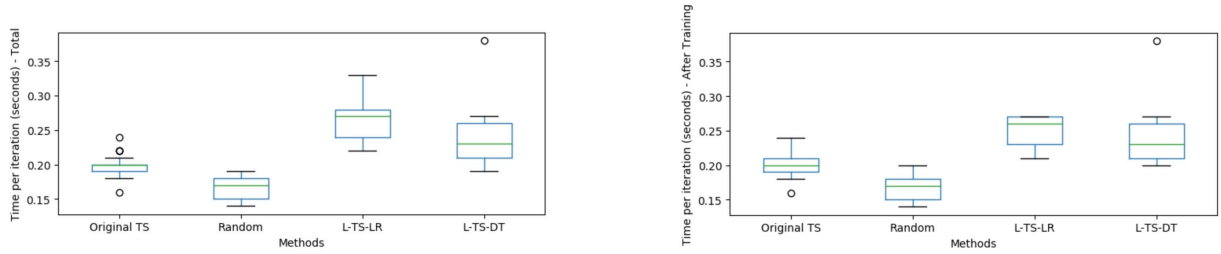


Figure 4: Comparing computing performance for all methods in the deterministic case with online learning



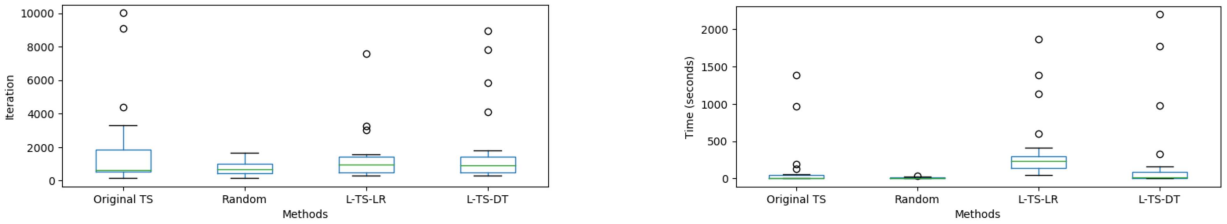
(a) Total

(b) After training

Figure 5: Comparing computing time per iteration for all methods in the deterministic case with online learning

The learning algorithms were employed within the search process, so, the pre-processing procedure, including data preparation and training time, has an impact on the total computing time. To evaluate this impact, we compared in Figure 5 the computational performance (time per iteration) of each method overall (Figure 5a) and for the application phase only (Figure 5b). We see in Figure 5 that logistic regression requires the most training time. This is due to the time needed to manipulate the training data and to train the model in online learning. However, this figure shows that the time spent for each iteration was nearly identical for all methods, with an average difference of around 5 seconds, which is rather insignificant (Figure 5b shows that the average time per iteration for the original TS is 0.2 seconds, while for the L-TS-LR model this time is 0.25 seconds).

We also compared how quickly different methods obtained the best solution by comparing the number of iterations and the time required by each method to obtain the best solution. This is shown in Figure 6. Figure 6a clearly illustrates that the L-TS-LR model finds the best solution faster than original TS.



(a) Number of iterations to reach the best solution

(b) Time (seconds) to reach the best solution

Figure 6: Comparing the speed of obtaining the best solution for all methods in the deterministic case with online learning

All of these illustrations, including the improvement in the objective value presented in Table 4, show the advantage of employing the learning TS idea and in particular the logistic regression model. More precisely, given the similar computing time and number of iterations to reach the best solution, as well as the improvement in the cost value, the logistic regression method demonstrates superior performance.
We performed a large number of experiments to validate our method. In the next section, we present these experiments within different settings, first for parameter settings and second for different testing strategies.

4.2.2. Design of experiments

In this section, we explain how we chose the parameters and features of our learning method. Different sets of parameters were considered in designing our experiments including fixed parameters and parameters based on the size of each instance. We also had different sets of parameters for both L-TS-LR and L-TS-DT models. Table 3 gives a detailed view of the different parameter settings.

Table 3: Design of experiments for parameter settings

Methods	Parameters	Fixed parameter					Dependent on the size of the instance		
TS	I^D_{st}	5	30	60	100	200	$I_{\sqrt{\lvert I\rvert\times\lvert J\rvert\times\lvert T\rvert}}$	$I_{2\times\sqrt{\lvert I\rvert\times\lvert J\rvert\times\lvert T\rvert}}$	$I_{3\times\sqrt{\lvert I\rvert\times\lvert J\rvert\times\lvert T\rvert}}$
	I^D_{end}	30	60	100			I^D_{st}		
LR	Regularization method	L1	L2						
	Training method	Batch	MINI-Batch						
	Learning iterations	1000	500	100	10				
DT	Min sample count						$\lvert I\rvert$	$\lvert I\rvert\times\lvert J\rvert$	$\lvert I\rvert\times\lvert J\rvert\times\lvert T\rvert$
	Max depth	50	10						

As an illustration, in Table 4 we present part of our experiment on varying the sizes of I^D_{st} and I^D_{end} parameters. This table presents "GAP- Best" by varying parameters and compares the best solution obtained from 10 different runs for each parameter and the best solution obtained from the original TS. It shows that increasing the size of the training set (I^D_{end} - I^D_{st}) improves the performance of the learning model. This can be observed by comparing the columns of $I_{60}-I_{160}$, $I_{100}-I_{200}$, $I_{100}-I_{160}$ and $I_{100}-I_{130}$ standing for $I^D_{st}-I^D_{end}$. However, these parameters should be optimized to avoid the risk of over-fitting.

There is an exception in column $I_{30} - I_{130}$. This is due to the early start of the training phase and lack of meaningful data. Hence, we evaluated the method based on different instance sizes of the problem in the last three columns. In our experiments, we observed that the best parameter is set close to the iteration where the algorithm starts descending.

Table 4: Comparing the performance of the learning algorithm by varying the size of training set and training duration

	Tests	GAP - Best (%)							
		Fixed parameter					Dependent on the size of the instance		
		$I_{30} - I_{130}$	$I_{60} - I_{160}$	$I_{100} - I_{200}$	$I_{100} - I_{160}$	$I_{100} - I_{130}$	$I_{\sqrt{\lvert I\rvert\times\lvert J\rvert\times\lvert T\rvert}}$	$I_{2\times\sqrt{\lvert I\rvert\times\lvert J\rvert\times\lvert T\rvert}}$	$I_{3\times\sqrt{\lvert I\rvert\times\lvert J\rvert\times\lvert T\rvert}}$
Small	pr-(6,7)	**-1.2**	0.0	0.0	**-0.2**	0.0	**-0.7**	**-0.7**	**-1.0**
	pr-(6,9)	1.3	0.8	1.0	0.8	0.8	0.8	1.5	0.8
	pr-(6,11)	0.5	**-0.3**	1.1	1.1	1.3	**-0.5**	**-0.3**	0.3
	pr-(6,12)	1.6	1.1	0.8	0.8	0.8	0.3	0.3	0.5
	pr-(8,7)	**-0.4**	0.7	**-0.5**	**-0.5**	**-0.5**	0.0	**-0.5**	**-0.7**
	pr-(8,9)	0.7	0.9	0.4	0.4	0.0	0.0	**-0.4**	**-0.4**
	pr-(10,7)	0.3	0.4	0.0	0.0	0.6	0.3	**-0.3**	0.1
Medium	pr-(6,20)	0.3	1.6	0.6	0.6	0.6	0.0	2.5	0.9
	pr-(8,11)	**-0.4**	**-0.2**	0.2	0.2	0.2	0.6	0.0	0.6
	pr-(8,12)	0.0	**-0.2**	0.0	0.0	0.0	**-0.2**	**-0.6**	0.0
	pr-(8,20)	0.2	**-1.0**	0.6	0.6	0.6	**-0.6**	**-0.2**	0.8
	pr-(10,9)	0.9	1.1	**-0.1**	**-0.1**	**-0.1**	0.3	0.0	0.3
	pr-(10,11)	0.4	0.1	0.0	0.0	0.0	**-0.3**	0.1	**-0.3**
	pr-(10,12)	0.3	0.1	**-0.4**	0.7	0.7	0.4	0.1	0.1
Large	pr-(6,40)	**-5.7**	**-5.7**	**-5.7**	**-6.2**	**-6.6**	0.0	**-1.3**	**-4.0**
	pr-(6,60)	**-14.0**	**-7.0**	**-3.5**	**-3.5**	**-3.5**	**-0.7**	**-2.2**	**-0.7**
	pr-(8,40)	4.8	0.5	0.8	0.8	0.8	**-0.8**	**-1.3**	**-0.5**
	pr-(8,60)	8.3	6.9	5.8	5.8	5.8	2.0	4.9	1.3
	pr-(10,20)	0.9	1.7	1.9	1.2	2.0	0.6	1.2	0.2
	pr-(10,40)	4.4	0.6	0.7	3.1	3.1	1.1	1.1	0.5
	pr-(10,60)	2.8	1.6	0.9	1.9	1.9	**-0.2**	**-0.2**	0.4
	Average:	0.29	0.18	0.21	0.35	0.4	0.1	0.18	**-0.03**

Table 5 presents the results obtained from a subset of different groups for training features in our primary move, v_1, with online learning. It can be observed that considering the set of all features in the first column, performed better than all other subsets and explains why we chose this set of features to train our learning method. In this table each column represents a subset of training features that previously presented in Table 1 and "All" shows the results when we consider all features.

Table 5: Comparing the effect of different feature sets in learning - without noise

	Tests	GAP - Best (%)							
		All	Δ_x	$\Delta_x > 0$	x	D^*	D'	(x, Δ_x)	(D^*, Δ_x)
Small	pr-(6,7)	0.0	0.5	0.5	0.3	0.3	0.5	0.5	0.3
	pr-(6,9)	1.1	0.0	0.0	0.5	1.1	1.1	0.5	0.3
	pr-(6,11)	**-0.3**	**-0.3**	**-0.3**	0.0	0.0	0.0	0.0	0.0
	pr-(6,12)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	pr-(8,7)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	pr-(8,9)	0.0	1.0	1.0	0.0	0.4	0.4	0.4	0.4
	pr-(10,7)	0.1	0.3	0.3	0.3	0.3	0.3	0.1	0.1
Medium	pr-(6,20)	1.1	1.1	1.1	0.0	1.1	1.1	1.1	1.1
	pr-(8,11)	0.0	0.0	0.0	0.0	**-0.2**	**-0.2**	0.0	0.0
	pr-(8,12)	0.4	0.0	0.0	**-0.2**	**-0.2**	**-0.2**	**-0.2**	**-0.2**
	pr-(8,20)	0.0	**-0.7**	**-0.7**	0.0	0.0	0.0	0.0	0.0
	pr-(10,9)	0.2	0.2	0.2	0.0	0.0	0.0	**-0.3**	**-0.3**
	pr-(10,11)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	pr-(10,12)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Large	pr-(6,40)	0.7	0.7	0.7	0.0	0.7	0.7	0.7	0.7
	pr-(6,60)	0.0	22.0	22.0	9.8	9.8	9.8	9.8	9.8
	pr-(8,40)	**-0.7**	**-1.7**	**-1.7**	**-1.0**	**-1.0**	**-1.0**	**-1.0**	**-1.0**
	pr-(8,60)	**-0.7**	**-0.7**	**-0.7**	**-0.7**	**-0.7**	**-0.7**	**-0.7**	**-0.7**
	pr-(10,20)	0.0	**-0.5**	**-0.5**	0.0	0.0	0.0	0.0	0.0
	pr-(10,40)	0.7	1.3	1.3	0.2	0.2	0.2	0.2	0.2
	pr-(10,60)	**-1.6**	**-1.9**	**-1.9**	**-1.9**	**-1.9**	**-1.9**	**-1.9**	**-1.9**
	Average:	0.05	1.01	1.01	0.34	0.46	0.48	0.44	0.41

In addition to different sets of parameters, we experiment with different learning methods, including online and offline learning, and also different prediction outputs. Table 6 outlines the design of these experiments. What do online and offline learning mean in the context of our proposed L-TS algorithm? As explained before, online learning is our primary method of learning and was explained in detail in Section 3. However, in offline learning, the learning procedure was modified so that the algorithm runs twice, in two separate rounds.

Table 6: Design of experiments for tests

	Training				Prediction	
	Physician	Patient	Initial Solution	Subset of Features	Physician	Move
Offline Learning	✓	✓	✓		✓	
Online Learning	✓			✓	✓	✓

The first round for the offline learning includes the training phase. In this way, we collect the behavioral data from the search space for I^D_{st} iterations, and the parameter of I^D_{end} is set to $Stop_{max} = 1h$. The last

trained model is saved and used in the second round. This round was done only once and the results were saved to reuse in the second round.

The second run functions as the application phase. It starts by collecting data in the first I_{st}^{D} iterations. It then uses the pre-trained model from the first round to predict the output. We save the training time in this part.

We use different strategies to choose between the trained models, to determine how to best use them to predict the output. To choose the appropriate strategy, we study its impact on the performance. For instance, we may need to train the model using all 10 randomly-generated initial solutions in some cases, while in others we can use the trained model from one initial solution. In this way, we are always seeking to minimize the training time. These strategies are based on

- the number of physicians;
- the number of patients and
- the data used to generate the initial solution (different seeds).

In strategies based on the number of physicians, we varied the size of the physicians for instances with the same sizes in number of patients. For example, we used the trained model from instance pr-(6,7) to predict the output in the second phase for instances pr-(8,7) and pr-(10,7). This pattern applies for strategies based on number of patients and generating initial solution as well.

The performance of these experiments on the logistic regression model are shown in Table 7. The "GAP - Best" and "GAP - Avg" columns present the improvements from the best solution obtained from the original TS. It can be observed that learning on one initial solution is more robust than other strategies; however, its performance ($+0.1\%$ on average) is still worse than the online learning method presented in Table 2 (-0.3% on average).

The performance of online versus offline learning was also evaluated based on computing time. Figure 7 compares the time per iteration for different methods, while Figure 8 compares the total number of iterations per instance for each method. It is obvious that the offline method performs much faster than the online method. It still needs some time to treat and categorize the behavioral data for prediction, however, it is almost as fast as the original TS and the difference is negligible.

The original tabu search presented in [7] proved to be very efficient and obtained near-optimal solutions. Thus, there is no room for improvements in some instances in the deterministic case. However, due to the variability of scenarios in the stochastic case, there is more flexibility in the cost function and deploying learning methods shows more potential. The next section presents the experiments in a stochastic environment.

Table 7: Comparing the performance of logistic regression in the deterministic case with offline learning

	Tests	GAP - Best (%)			GAP - Avg (%)		
		Physicians	Patients	Initial Sol.	Physicians	Patients	Initial Sol.
Small	pr-(6,7)	0.0	0.1	**-0.2**	**-0.1**	**-0.2**	**-0.1**
	pr-(6,9)	1.5	1.9	1.6	**-0.9**	**-0.7**	**-1.1**
	pr-(6,11)	1.8	1.9	0.7	0.7	0.6	**-0.5**
	pr-(6,12)	2.1	2.4	1.2	1.0	1.4	0.2
	pr-(8,7)	0.9	1.0	0.6	0.9	0.8	0.5
	pr-(8,9)	0.7	1.1	0.4	0.6	0.7	0.2
	pr-(10,7)	0.1	0.4	0.5	0.2	0.2	0.3
Medium	pr-(6,20)	26.0	1.3	0.8	1.8	1.5	0.9
	pr-(8,11)	0.3	0.2	0.4	0.0	**-0.1**	0.2
	pr-(8,12)	**-0.2**	**-0.1**	**-0.4**	0.0	0.0	**-0.2**
	pr-(8,20)	**-0.8**	**-0.7**	0.4	**-1.1**	**-1.1**	**-0.3**
	pr-(10,9)	1.1	1.2	0.8	0.5	0.6	0.1
	pr-(10,11)	**-0.1**	**-0.2**	0.1	0.3	0.1	0.3
	pr-(10,12)	0.9	0.9	0.7	0.6	0.6	0.4
Large	pr-(6,40)	5.3	4.8	1.5	5.4	4.9	2.1
	pr-(6,60)	2.9	3.3	6.5	**-8.0**	**-8.7**	**-5.7**
	pr-(8,40)	**-0.3**	0.0	1.4	**-0.6**	1.0	0.2
	pr-(8,60)	3.6	3.8	4.4	6.7	5.7	3.4
	pr-(10,20)	1.3	1.2	0.7	0.9	0.9	0.3
	pr-(10,40)	2.0	2.2	1.3	1.6	1.7	0.5
	pr-(10,60)	0.7	0.7	0.1	1.0	0.9	0.8
	Average:	0.7	0.7	0.1	1.0	0.9	0.8

4.2.3. Experimental performance on the Stochastic case

To evaluate the solution obtained from the stochastic tabu search algorithm, we proceed as follows:

- Generate a set A of scenarios (up to 50 different scenarios),
- For each instance (i.e., pr-(6,7) to pr-(10,60)), run the algorithm using 10, 30 or 50 scenarios from set A (using one scenario is equivalent to the deterministic case) for $Stop_{max}$ = 2.5, 5, 7.5 and 10 hours.

We demonstrated in the deterministic case that online learning outperforms offline learning in terms of the solution quality, so we proceed with online learning in the stochastic case. For the stochastic case, the values of the parameters I_1^S, I_3^S and θ^S are the same except that I_2^S is now equal to $|J|$, the number of patients, $I_{st}^S = \sqrt{|J| \times |I| \times |D|}$, and $Stop_{max} = 5h$.

Table 8 contains the results of this experiment with online learning. It can be observed that both learning methods improved the stochastic tabu search globally. We can see the most improvement in cases using 10 scenarios, with -1.67% on average for L-TS-LR and -3.02% for L-TS-DT methods.

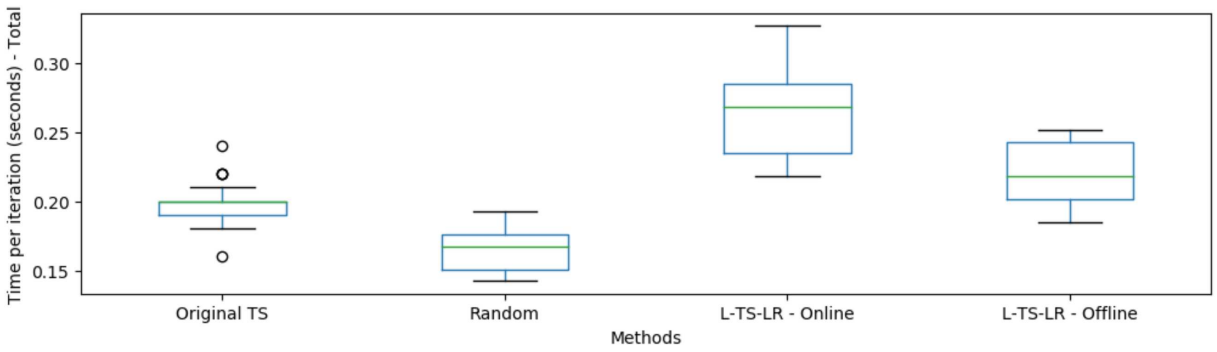


Figure 7: Comparing computing time per iteration for online vs. offline learning in the deterministic case

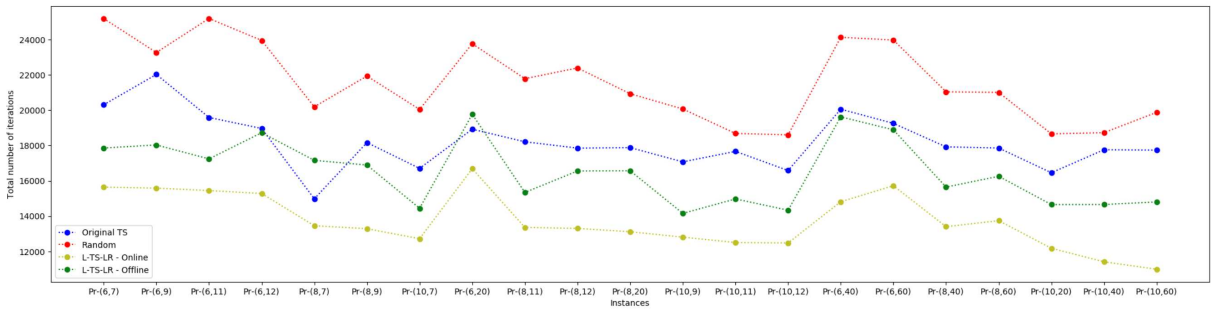
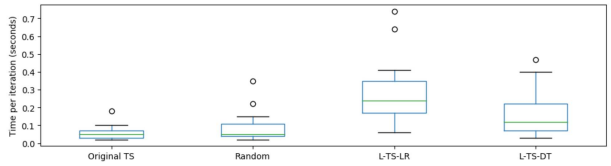


Figure 8: Comparing total number of iterations for online vs. offline learning in the deterministic case

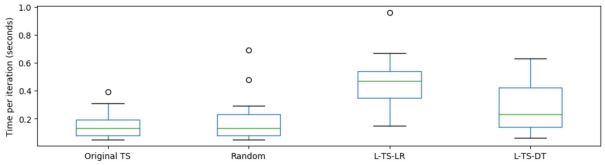
Table 8: Comparing the performance of different methods in the stochastic case

	Tests	GAP - Best (%)									GAP - Avg (%)								
		10 Scenarios			30 Scenarios			50 Scenarios			10 Scenarios			30 Scenarios			50 Scenarios		
		Random	L-TS-LR	L-TS-DT	Random	L-TS-LR	L-TS-DT	Random	L-TS-LR	L-TS-DT	Random	L-TS-LR	L-TS-DT	Random	L-TS-LR	L-TS-DT	Random	L-TS-LR	L-TS-DT
Small	pr-(6,7)	1.3	**-0.3**	**-0.3**	1.7	**-0.7**	**-0.5**	1.0	**-0.7**	**-0.2**	1.1	0.2	**-0.3**	0.8	**-1.4**	**-1.1**	0.9	**-1.1**	**-0.2**
	pr-(6,9)	0.8	0.0	0.0	1.3	0.0	**-0.3**	1.5	0.3	0.3	0.9	**-1.2**	0.0	1.5	**-1.1**	**-0.8**	0.5	**-2.2**	0.3
	pr-(6,11)	2.2	**-0.3**	**-0.6**	2.2	**-0.3**	**-0.3**	1.1	**-1.3**	**-1.1**	1.0	**-1.1**	**-0.6**	**-0.5**	**-1.1**	**-0.9**	2.6	**-2.6**	**-1.1**
	pr-(6,12)	0.8	**-1.4**	**-0.8**	0.8	**-0.6**	**-0.3**	0.8	**-1.6**	**-1.3**	0.3	**-1.3**	**-0.8**	1.2	**-1.1**	**-0.4**	1.6	1.3	**-1.3**
	pr-(8,7)	1.3	**-0.5**	**-0.4**	1.3	**-0.4**	**-0.4**	**-0.4**	**-1.1**	**-0.5**	0.4	**-1.0**	**-0.4**	1.5	**-0.6**	0.0	2.1	**-1.3**	**-0.5**
	pr-(8,9)	1.7	**-0.4**	**-0.4**	1.1	**-0.4**	-0.7	1.4	**-0.4**	**-0.4**	1.6	**-0.6**	**-0.4**	1.6	**-0.5**	0.1	0.7	**-1.4**	**-0.4**
	pr-(10,7)	1.0	**-0.7**	0.0	1.8	0.1	0.3	1.0	0.1	**-0.1**	1.3	**-0.4**	0.0	1.0	**-0.5**	**-0.1**	0.7	**-0.9**	**-0.1**
Medium	pr-(6,20)	0.7	**-1.3**	**-2.0**	0.6	**-1.9**	**-0.6**	2.7	0.6	**-0.3**	1.6	**-1.4**	**-2.0**	1.5	**-4.4**	**-4.0**	0.5	0.6	**-0.3**
	pr-(8,11)	1.0	**-0.6**	**-1.0**	1.2	**-0.4**	**-0.4**	0.9	**-0.6**	0.0	1.3	**-1.3**	**-1.0**	0.5	**-1.1**	**-0.5**	0.3	**-1.6**	0.0
	pr-(8,12)	1.2	**-0.8**	**-0.2**	1.8	**-0.6**	**-1.0**	0.0	**-1.5**	**-2.1**	0.6	**-0.8**	**-0.2**	0.5	**-1.9**	**-0.7**	0.0	**-2.0**	**-2.1**
	pr-(8,20)	2.4	**-0.7**	0.4	0.8	**-1.3**	**-1.0**	1.8	**-0.4**	**-1.0**	**-1.3**	**-1.4**	0.4	0.8	**-3.1**	**-1.1**	1.7	**-2.1**	**-1.0**
	pr-(10,9)	1.5	**-0.1**	**-0.3**	1.7	0.1	0.3	1.0	**-0.4**	**-0.4**	1.5	**-0.4**	**-0.3**	0.9	**-0.8**	**-0.6**	0.3	**-1.6**	**-0.4**
	pr-(10,11)	0.4	**-1.0**	**-0.4**	0.7	**-0.7**	**-0.3**	0.9	0.3	0.3	1.5	**-1.0**	**-0.4**	0.0	**-0.7**	**-0.1**	**-0.4**	**-1.9**	0.3
	pr-(10,12)	1.3	**-0.3**	**-0.3**	1.8	**-0.4**	**-0.4**	0.4	**-0.4**	**-0.4**	1.2	**-0.3**	**-0.3**	0.8	**-1.2**	0.5	0.1	**-2.2**	**-0.4**
Large	pr-(6,40)	5.2	4.6	1.1	4.3	**-1.4**	**-1.0**	**-1.3**	**-3.5**	**-3.1**	2.2	1.9	1.1	0.3	**-1.8**	**-0.3**	4.3	**-0.8**	**-3.1**
	pr-(6,60)	**-10.7**	**-25.0**	**-42.9**	8.1	0.0	0.0	3.8	**-1.5**	**-3.8**	16.3	**-44.7**	**-42.9**	12.4	0.3	**-76.5**	0.1	**-3.2**	**-3.8**
	pr-(8,40)	1.3	**-0.6**	**-6.6**	1.6	**-0.5**	**-1.4**	0.8	0.0	**-1.8**	2.8	**-0.4**	**-6.6**	**-0.3**	**-0.9**	**-0.9**	0.5	**-2.4**	**-1.8**
	pr-(8,60)	0.0	**-3.2**	**-3.7**	0.5	**-1.4**	**-4.5**	1.7	**-0.3**	**-2.0**	3.3	**-4.8**	**-3.7**	3.1	1.6	**-27.7**	0.1	**-3.5**	**-2.0**
	pr-(10,20)	1.1	**-0.7**	**-0.8**	1.9	**-0.2**	**-0.3**	1.5	0.0	**-0.5**	1.2	**-1.0**	**-0.8**	0.2	**-2.3**	**-1.9**	0.2	**-4.6**	**-0.5**
	pr-(10,40)	0.2	**-2.1**	**-2.8**	1.5	0.0	**-0.6**	0.9	**-0.4**	**-0.5**	0.4	**-1.0**	**-2.8**	0.3	**-0.3**	0.0	0.1	**-1.5**	**-0.5**
	pr-(10,60)	2.0	0.3	**-1.7**	2.6	**-0.5**	**-0.3**	0.9	0.4	0.0	1.5	4.3	**-1.7**	0.6	**-1.1**	**-2.7**	2.8	**-1.9**	0.0
	Average:	0.79	**-1.67**	**-3.02**	1.87	**-0.54**	**-0.65**	1.07	**-0.59**	**-0.91**	1.93	**-2.75**	**-3.02**	1.36	**-1.14**	**-5.7**	0.94	**-1.75**	**-0.91**

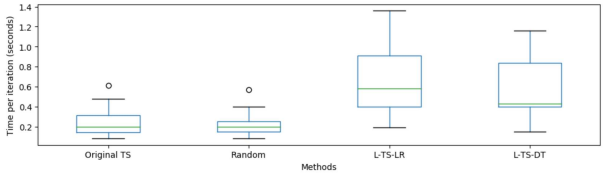
The performance of each learning algorithm in a stochastic environment was also evaluated based on its computing time. Figure 9 compares the time per iteration by method for 10, 30 and 50 scenarios. It is clear that increasing the number of scenarios increases the computing time.



(a) 10 Scenarios



(b) 30 Scenarios



(c) 50 Scenarios

Figure 9: Comparing computing time per iteration for all methods in the stochastic case with online learning

We observe that with 10 scenarios, the average gap improved by 1.67% with logistic regression and 3.02% with decision trees models. Also, with 30 and 50 scenarios, we have (0.54%, 0.65%) and (0.59%, 0.91%) improvements in average gap for L-TS-LR and L-TS-DT models.

5. Conclusion

In this paper, we proposed a learning tabu search method and studied its performance on a physician scheduling problem. The performance of the proposed algorithm was evaluated using the benchmark instances. We presented our experimental design and evaluated the new method in both deterministic and stochastic environments. We showed that our method is very efficient compared with the original tabu search and a random method, especially in its stochastic version. Over 21 instances, the average gap improved by 1.67% with logistic regression and 3% with decision trees in the cases with 10 scenarios. The learning methods obtained best solutions faster than the original TS and random methods over the computing time. Although we studied the application of this method in a scheduling problem, tabu search has already been used to solve pretty much all optimization problems. Thus, the learning tabu search algorithm can be adapted to other applications. Future work could employ learning tabu search to solve other optimization problems by generalizing several ingredients for which we gave special attention to our specific application.

References

[1] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *European Journal of Operational Research*, 2020.

[2] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decision sciences*, vol. 8, no. 1, pp. 156–166, 1977.

[3] F. Glover and J.-K. Hao, "Diversification-based learning in computing and optimization," *Journal of Heuristics*, vol. 25, no. 4-5, pp. 521–537, 2019.

[4] Y. Chen, J.-K. Hao, and F. Glover, "An evolutionary path relinking approach for the quadratic multiple knapsack problem," *Knowledge-Based Systems*, vol. 92, pp. 23–34, 2016.

[5] Y. Jin and J.-K. Hao, "Hybrid evolutionary search for the minimum sum coloring problem of graphs," *Information Sciences*, vol. 352, pp. 15–34, 2016.

[6] X. Lai and J.-K. Hao, "A tabu search based memetic algorithm for the max-mean dispersion problem," *Computers & Operations Research*, vol. 72, pp. 118–127, 2016.

[7] N. Niroumandrad and N. Lahrichi, "A stochastic tabu search algorithm to align physician schedule with patient flow," *Health care management science*, vol. 21, no. 2, pp. 244–258, 2018.

[8] F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[9] F. Arnold, Í. Santana, K. Sörensen, and T. Vidal, "Pils: Exploring high-order neighborhoods by pattern mining and injection," *Pattern Recognition*, p. 107 957, 2021.

[10] R. Battiti and M. Brunato, "The lion way," *Machine Learning plus Intelligent Optimization. LION-lab, University of Trento, Italy*, vol. 94, 2014.

[11] F. Hafiz and A. Abdennour, "Particle swarm algorithm variants for the quadratic assignment problems-a probabilistic learning approach," *Expert Systems with Applications*, vol. 44, pp. 413–431, 2016.

[12] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, "Algorithm runtime prediction: Methods & evaluation," *Artificial Intelligence*, vol. 206, pp. 79–111, 2014.

[13] J. Ceberio, A. Mendiburu, and J. A. Lozano, "The plackett-luce ranking model on permutation-based optimization problems," in *2013 IEEE Congress on Evolutionary Computation*, IEEE, 2013, pp. 494–501.

[14] J. Boyan and A. W. Moore, "Learning evaluation functions to improve optimization by local search," *Journal of Machine Learning Research*, vol. 1, no. Nov, pp. 77–112, 2000.

[15] D. C. Porumbel, J.-K. Hao, and P. Kuntz, "A search space "cartography" for guiding graph coloring heuristics," *Computers & Operations Research*, vol. 37, no. 4, pp. 769–778, 2010.

[16] J.-P. Hamiez and J.-K. Hao, "An analysis of solution properties of the graph coloring problem," in *Metaheuristics: computer decision-making*, Springer, 2003, pp. 325–345.

[17] S. Baluja, A. Barto, K. Boese, J. Boyan, W. Buntine, T. Carson, R. Caruana, D. Cook, S. Davies, T. Dean, *et al.*, "Statistical machine learning for large-scale optimization," 2000.

[18] J. Boyan and A. W. Moore, "Learning evaluation functions for global optimization and boolean satisfiability," in *AAAI/IAAI*, 1998, pp. 3–10.

[19] C. Bongiovanni, M. Kaspi, J.-F. Cordeau, and N. Geroliminis, "A predictive large neighborhood search for the dynamic electric autonomous dial-a-ride problem," Tech. Rep., 2020.

[20] S. Thevenin and N. Zufferey, "Learning variable neighborhood search for a scheduling problem with time windows and rejections," *Discrete Applied Mathematics*, vol. 261, pp. 344–353, 2019.

[21] Y. Sun, X. Li, and A. Ernst, "Using statistical measures and machine learning for graph reduction to solve maximum weight clique problems," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[22] J. Lauri and S. Dutta, "Fine-grained search space classification for hard enumeration variants of subset problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 2314–2321.

[23] B. Abbasi, T. Babaei, Z. Hosseinifard, K. Smith-Miles, and M. Dehghani, "Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management," *Computers & Operations Research*, vol. 119, p. 104 941, 2020.

[24] M. Fischetti and M. Fraccaro, "Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks," *Computers & Operations Research*, vol. 106, pp. 289–297, 2019.

[25] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi, "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art," *European Journal of Operational Research*, 2021.

[26] E.-G. Talbi, "Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics," 2020.

[27] R. C. L. Riley and C. Rego, "Intensification, diversification, and learning via relaxation adaptive memory programming: A case study on resource constrained project scheduling," *Journal of Heuristics*, vol. 25, no. 4-5, pp. 793–807, 2019.

[28] D. Schindl and N. Zufferey, "A learning tabu search for a truck allocation problem with linear and nonlinear cost components," *Naval Research Logistics (NRL)*, vol. 62, no. 1, pp. 32–45, 2015.

[29] F. Glover and M. Laguna, *Tabu search*. John Wiley & Sons, Inc., 1993.

[30] Y. Wang, Q. Wu, and F. Glover, "Effective metaheuristic algorithms for the minimum differential dispersion problem," *European Journal of Operational Research*, vol. 258, no. 3, pp. 829–843, 2017.

[31] Q. Wu, Y. Wang, and F. Glover, *Advanced algorithms for bipartite boolean quadratic programs guided by tabu search, strategic oscillation and path relinking*, 2017.

[32] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009, ISBN: 0131873210.

[33] Y.-J. Hu, T.-H. Ku, R.-H. Jan, K. Wang, Y.-C. Tseng, and S.-F. Yang, "Decision tree-based learning to predict patient controlled analgesia consumption and readjustment," *BMC medical informatics and decision making*, vol. 12, no. 1, pp. 1–15, 2012.

[34] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[35] I. A. Bikker, N. Kortbeek, R. M. van Os, and R. J. Boucherie, "Reducing access times for radiation treatment by aligning the doctor's schemes," *Operations research for health care*, vol. 7, pp. 111–121, 2015.

[36] B. Vieira, E. W. Hans, C. van Vliet-Vroegindeweij, J. van de Kamer, and W. van Harten, "Operations research for resource planning and-use in radiotherapy: A literature review," *BMC medical informatics and decision making*, vol. 16, no. 1, p. 149, 2016.

[37] T. Berthold, "Measuring the impact of primal heuristics," *Operations Research Letters*, vol. 41, no. 6, pp. 611–614, 2013.