A Heuristic Approach to Schedule Reoptimization in the Context of Interactive Optimization

David Meignan
Institut für Informatik
Universität Osnabrück
Osnabrück, Deutschland
dmeignan@uni-osnabrueck.de

ABSTRACT

Optimization models used in planning and scheduling systems are not exempt from inaccuracies. These optimization systems often require an expert to assess solutions and to adjust them before taking decisions. However, adjusting a solution computed by an optimization procedure is difficult, especially because of the cascading effect. A small modification in a candidate solution may require to modify a large part of the solution. This obstacle to the adjustment of a solution can be overcome by interactive reoptimization. In this paper we analyze the impact of the cascading effect on a shift-scheduling problem and propose an efficient heuristic approach for reoptimizing solutions. The proposed approach is a local-search metaheuristic that has been adapted to the reoptimization. This approach is evaluated on a set of problem instances on which additional preferences are generated to simulate desired adjustments of a decision maker. Experimental results indicate that, even with a small perturbation, the cascading effect is manifest and cannot be efficiently tackled by applying recovery actions. Moreover, results show that the proposed reoptimization method provides significant cost gains within a short time while keeping a level of simplicity and modularity adequate for an implementation in a decision support system.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods, Scheduling*

General Terms

Design, Algorithms, Human factors

Keywords

Reoptimization, interactive optimization, shift scheduling, heuristic

1. INTRODUCTION

Optimization models used for decision support rarely consider all aspects of the real decision maker's problem [3, 20]. Several practical considerations explain this gap between an implemented optimization model and the real problem for which the decision maker has to take a decision. First, the optimization problem may be difficult to model, and tools and available resources to design complex models are often limited. For example in the domain of staff scheduling and rostering, optimization problems generally contain multiple objectives that are specific to the application area [9]. Some aspects such as stochastic demand or fairness between employees are difficult to model in a form appropriate for an optimization procedure. In addition to these difficulties inherent to the optimization problem, the knowledge required to model the problem may not be easily accessible. The transcription of expert knowledge into an optimization model may introduce some inaccuracies. Furthermore, it is common that secondary decision criteria are only identified when the real problem needs to be solved (i.e. at the time of the decision process and not during the initial design of the decision support tool). Consequently, in the context of decision aid, an optimization model is likely to present some simplifications and omissions. Interactive optimization may alleviate these inaccuracies and guarantee an efficient use of the optimization system.

In interactive optimization, the optimization system is adjusted through interactions with the decision maker in order to meet his preferences. The feedback of the decision maker on intermediate or previous optimization results can significantly modify the final solution or the performance of the optimization process. There are essentially two types of interactive optimization methods depending on the objective of the interaction. The first type of method uses interaction in order to improve the performance of the optimization process. This form of interaction appears, for instance, in human-guided search approaches [13], as well as interactive parameter tuning [12]. In the second type of interactive method, the objective is to "enrich" the optimization model. The decision maker interacts with the optimization system in order to obtain solutions that are more relevant to the real problem. This type of interaction is considered in interactive methods for multiobjective optimization [18] and interactive evolutionary algorithms [22]. In these two approaches, interactive enrichment concerns respectively the trade-offs between objectives and the fitness function. The interactive reoptimization approach investigated in this study is another form of interactive enrichment.

Reoptimization is the optimization of a problem instance that has been slightly perturbed and for which a solution has been previously found. The reoptimization problem is characterized, first, by the presence of an initial solution that has been obtained by solving the unperturbed problem instance. This initial solution can be used to improve the performance of the reoptimization procedure. Second, in addition to the initial objectives of the optimization problem, it is generally required to minimize the distance between the re-optimized solution and the initial solution. Finally, the requirements in terms of computation time for the reoptimization are usually stronger than for the initial optimization.

Reoptimization is of practical interest in various circumstances [1, 25, 21], and in particular, for tackling problems in dynamic environments. In the latter case, the problem instance changes over time and the current solution has to evolve with the updated problem instance. For example, on a staff scheduling problem, the absence of a staff member can be considered as a dynamic perturbation. In this case, a new schedule has to be found by reoptimizing the initial one. Reoptimization can also be useful in the context of optimization-based decision support systems for adjusting a solution interactively [19]. Unlike "dynamic optimization", a perturbation in interactive reoptimization is a modification made by the decision maker and not a real change in the environment. As explained previously, some inaccuracies in the problem model may necessitate an adjustment of the solution by the decision maker in order to reflect the real decision problem. Interactive re-optimization is a method that aims at solving several issues of a manual adjustment of a solution after the optimization process. Manually modifying a solution has major drawbacks if it is not assisted by an optimization procedure. First, it may be difficult for the user to apprehend all constraints and objectives of the optimization model when a solution is manually edited. In addition, due to the complexity of considered combinatorial optimization problems, it is generally difficult or impossible to reflect the modification to the whole solution. The fact that a local modification in a solution implies to adjust other parts of the solution is referred to as the cascading or propagation effect [19]. Interactive reoptimization aims at overcoming these obstacles. In this approach, solutions modified by the user are reoptimized. The reoptimization procedure globally optimizes a solution to take into account local modifications applied by the user.

Shift scheduling is an application for which reoptimization has a strong potential. The objective of a shift scheduling problem is to optimize the assignments of employees to shifts for a given planning period. A shift is a time interval in a day (e.g. early 7:00-15:00, late 15:00-22:00) over which employees have to be scheduled [5, 9]. A demand in terms of required employees is defined for each shift and day. A solution to the problem is a set of daily assignments of employees to shifts such as the required number of employees by shift and by day is satisfied. In addition, a solution must satisfy a set of legal constraints and meet as far as possible employees' preferences. In the literature, the reoptimization variant of the shift scheduling problem is mainly studied in the context of dynamic environments, i.e. for adjusting a schedule when an unexpected event has invalidated the initial planning. For example, an employee initially scheduled may be unavailable and a new planning has to be determined to satisfy the demand. A review of staff rerostering models and methods in the context of dynamic environments is presented in [17].

In contrast to existing studies, we present in this paper an approach for dealing with the reoptimization problem in the context of interactive optimization. The key aspects of the proposed method are the simplicity, modularity, and efficiency of the reoptimization method. Simplicity and modularity are essential for an integration in a decision support tool [5]. The method must be easy to implement, and the number of parameters has to be limited. In addition, modularity is essential to take into account the peculiarities of the application domain. Furthermore, the requirements in terms of computational time and solution's quality are critical in the context of interactive optimization. The reoptimization should take only few seconds to recover from a perturbation introduced by the decision maker. To achieve these objectives, we propose an Iterated Local-Search (ILS) [14] which is a conceptually simple metaheuristic with very few parameters. In addition, the proposed implementation makes use of high-level parallelism and delta-evaluation of the constraints to improve the performance of the metaheuristic. The reoptimization method is compared to recovery procedures in order to stress the importance of the cascading effect and to evaluate the proposed method.

The remainder of the paper is organized as follows. In Section 2, the shift scheduling problem and its reoptimization variant are presented. The proposed reoptimization method is detailed in Section 3. In Section 4, the method is evaluated. Conclusions are drawn in Section 5.

2. PROBLEM DEFINITION

2.1 Shift scheduling problem

The reoptimization problem considered in this paper is based on a shift scheduling problem that has been proposed for the International Nurse Rostering Competition held in 2010 (Inrc2010) [11]. A formal definition of the Inrc2010 problem model is given in [15]. The problem contains two types of constraints, namely hard and soft constraints. The hard constraints must be satisfied; otherwise, the solution is unfeasible and may lose its meaning. The soft constraints represent the objectives of the optimization problem, and the number of violations of these constraints has to be minimized.

Two hard constraints are defined:

H_1 *Shift demand coverage*: For each day, the number of employees assigned to a shift must be equal to the demand for the shift. Under-staffing and over-staffing is not allowed.

H_2 *Single shift assignment per day*: At most one shift can be assigned to an employee per day. A shift represents a complete working day.

We differentiate two types of soft constraints. The first set of constraints is related to work regulation and corresponds to the primary objectives, noted R. The second set, noted S, includes constraints that are more flexible and related to individual comfort. The problem model contains twelve soft constraints divided as follows:

R_1 *Maximum number of assignments*: The total number of assignments of an employee within the planning horizon should not exceed a given maximum value.

R_2 *Minimum number of assignments*: The total number of assignments of an employee within the planning horizon should not be less than a given minimum value.

R_3 *Maximum number of consecutive working days*: The number of consecutive working days of an employee should not exceed a given maximum value.

R_4 *Skill requirements*: The employee assigned to a shift should have the required skills for the shift (e.g. head or manager).

R_5 *Unwanted consecutive working shifts*: Some shifts should not be worked consecutively. Unwanted shift patterns consist of two or three shifts, e.g. *[Late; Early]*.

R_6 *Day-off and shift-off requests*: Assignments should satisfy day-off and shift-off requests of employees.

S_1 *Complete working weekends*: Weekends should be either entirely worked or completely free.

S_2 *Identical shifts during weekends*: During weekends completely worked, an employee should have the same shift assigned.

S_3 *Minimum number of consecutive working days*: The number of consecutive working days of an employee should not be less than a given minimum value.

S_4 *Maximum number of consecutive free days*: The number of consecutive day-offs of an employee should not exceed a given maximum value.

S_5 *Minimum number of consecutive free days*: The number of consecutive day-offs of an employee should not be less than a given minimum value.

S_6 *No free day before working weekends*: The day before a completely worked weekend should not be a day-off.

Except for R_4 and R_6, the parameters of these constraints are specified by type of contract and apply to a subset of the employees. For example, the maximum number of assignments may be set to 20 (on a four-weeks planning horizon) for full-time contracts, and limited to 10 for part-time contracts. For the constraint R_4 and R_6, skills, day-offs requests, and shift-offs requests are specified per employee.

2.2 Reoptimization problem

In the context of interactive optimization, the reoptimization problem appears when an initial solution is modified by the decision maker. These modifications aim at introducing aspects that are not modeled. For instance, the decision maker may want to reflect some parental constraints, or anticipate potential absences. The studied interactive scheduling process is summarized in Figure 1. First, an initial solution is obtained on the basis of the shift scheduling problem described in the previous section. Then, the decision maker specifies some preferences on assignments. Here, the schedule is not directly modified; instead, preferences on assignments are defined. An assignment preference is a shift that is preferred for a specific employee and day. Introducing these preferences result in a perturbed solution. A reoptimization procedure is applied on the perturbed solution to take into account the assignment preferences in addition to the initial constraints. The process of preference definition followed by the reoptimization can be iterated until a satisfactory schedule is obtained.

In comparison to the initial shift scheduling problem, the reoptimization problem contains two additional soft constraints. The first one corresponds to the modifications requested by the decision maker. The second additional soft constraint minimizes the distance between the reoptimized solution and the initial solution. For interactive reoptimization, these two constraints form the preferential model of the decision process, whereas the initial shift scheduling problem is the substantive model [24, 18]. In the following, the assignment preference constraint and the reoptimization distance constraint are described using the notation of [15]. In this notation, a candidate schedule is represented by the integer decision variables $x_{e,d} \in \mathcal{S}, \forall e \in \mathcal{E}, \forall d \in \mathcal{D}$, with $\mathcal{S}$ the set of shifts, $\mathcal{E}$ the set of employees, and $\mathcal{D}$ the set of day in the planning horizon.

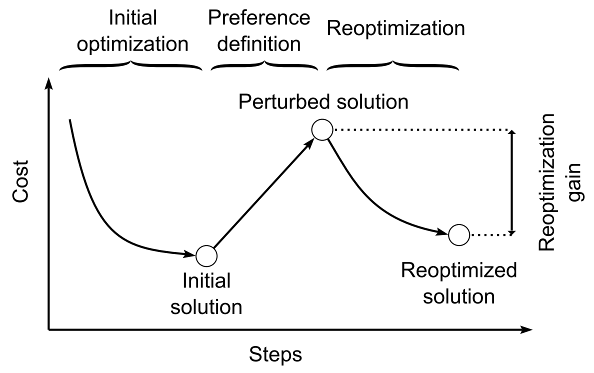

Figure 1: Principle of the interactive reoptimization process.

2.2.1 Assignment preference constraint

An assignment preference is a tuple $\langle e, d, p\rangle \in \mathcal{P}$ indicating that, for the employee $e \in \mathcal{E}$ and day $d \in \mathcal{D}$, the shift $p \in \mathcal{S}$ is the preferred assignment. The assignment preference constraint, noted P, penalizes each assignment $x_{e,d}$ for which a preference is defined and not satisfied. The cost of the constraint is defined as follows:

$$f_P = \sum_{\langle e,d,p\rangle \in \mathcal{P}} \chi(x_{e,d} \neq p)$$

where χ is the function that maps a true proposition to the value 1 and a false one to 0.

2.2.2 Reoptimization distance constraint

In the proposed reoptimization model, a soft constraint has been introduced to minimize the distance between the initial solution and the reoptimized one. A similar constraint can be found in reoptimization models for dynamic environment, however the motivation of such a constraint for interactive reoptimization is different to that for dynamic optimization. The aim of the distance constraint in the case of dynamic optimization is to minimize the number of reassignments of employee according to the current planning [16]. In the context of interactive reoptimization, a schedule's modification is not a real reassignment of employees because the schedule has not yet been decided. However, a distance constraint is required to maintain coherence in the interactive process. If possible, the reoptimized solution should be similar to the initial solution because the preferences of the decision maker have been expressed on the basis of the initial solution. With a completely different reoptimized solution the preferences expressed on the initial solution may lose their meaning. This distance constraint favors the convergence toward a satisfactory solution for the decision maker.

Given an initial solution $x^0_{e,d}$, the cost of the distance constraint D is:

$$f_D = \sum_{e\in\mathcal{E}}\sum_{d\in\mathcal{D}} \chi(x_{e,d} \neq x^0_{e,d})$$

2.2.3 Global objective

Introducing two constraints in the initial problem raises the question of the relative importance of the soft constraints. In the INRC2010 problem model a weight is associated to each soft constraint and possibly varies according to the employee's contract. The objective is to minimize the weighted sum of the violation of constraints. For the proposed shift scheduling problem and reoptimization problem, a lexicographic ordering of the constraints is considered to introduce the two additional constraints. The resulting objective function combines weights and lexicographic order, i.e. some soft constraints have the same priority.

For the shift scheduling problem solved in the initial optimization step, work regulation constraints R_i have a higher priority than soft constraints S_i. The weights are maintained within these two set of constraints. The objective of the initial optimization step is:

$$minimize_{LEX}\left(\sum f_{R_i}, \sum f_{S_i}\right) \qquad (1)$$

For the reoptimization step, the assignment preference constraint P has priority over soft constraints S_i but not over work regulation constraints R_i. The distance constraint has the lowest priority, as the role of the constraint is only related to the decision process. The objective of the reoptimization step is:

$$minimize_{LEX}\left(\sum f_{R_i}, f_P, \sum f_{S_i}, f_D\right) \qquad (2)$$

In short, the work regulation constraints are the most important objectives, followed by the assignment preferences defined by the decision maker. The soft constraints related to individual "comfort" of employees come third. The distance constraint is the less important objective.

3. OPTIMIZATION PROCEDURES

3.1 Iterated local search

An Iterated Local Search (ILS) [14] is proposed for both optimization and reoptimization phases. This metaheuristic has the advantages of being simple and having few parameters. The ILS procedure used for the shift scheduling problem and the reoptimization of schedules is described in Algorithm 1.

ILS repeats two steps until a stopping criterion is met. Here, the stopping criterion is a time limit T (Algorithm 1, line 5). Starting from an initial solution that is provided or generated, the first step perturbs the solution to escape from local-optima (line 6). The second step improves the resulting solution by local search (line 7). An acceptance criterion is then applied to decide whether the newly improved solution is retained for the next iteration or not (lines 8-9). The implemented criterion only accepts better solutions. The next iteration thus always starts with the best solution found so far.

For the reoptimization, the procedure starts with the perturbed schedule. The perturbations come from the introduction of assignment preferences by the decision maker.

Algorithm 1: ILS procedure for shift scheduling and schedule reoptimization

```
   Data: Time limit $T$, Initial solution $s^0$, Perturbation
         strength $\alpha$
   Result: Best found solution $s^*$
   // Generation/improvement of the initial solution
1  $s \leftarrow s^0$
2  if $s = \emptyset$ then $s \leftarrow generateSolution()$
3  $s \leftarrow variableNeighborhoodDescent(s)$
4  $s^* \leftarrow s$
5  while time limit $T$ not reached do
       // Perturbation
6      $s \leftarrow randomPerturbation(s, \alpha)$
       // Local-search
7      $s \leftarrow variableNeighborhoodDescent(s)$
       // Acceptance criterion
8      if $cost(s) < cost(s^*)$ then $s^* \leftarrow s$
9      $s \leftarrow s^*$
10 end
```

This perturbed solution is an advantage for the search procedure if we assume that few assignment preferences have been added. For the initial optimization phase preceding the definition of assignment preferences, no solution is provided to the ILS procedure. In this case, an initial solution is generated using a greedy generation procedure.

The local-search step of the ILS procedure is a variable neighborhood descent [10] based on block-swap neighborhoods [7]. A block-swap is the exchange, between two employees, of blocks of consecutive shift-assignments on the same range of days. The variable neighborhood descent procedure uses block-swap neighborhoods with a block size varying between 1 and 7. The local-search procedure is described in Algorithm 2. The exploration starts with a block size of 1 for the neighborhood. If no solution in the neighborhood has a better cost than the current solution, then the block size is increased. When a better solution is found in the current neighborhood, the exploration continues from the neighborhood of this new solution, and the block size is reset to 1. The procedure ends when no improving solution is found in the 7-swap neighborhood. In this way, the solution is improved until a local-optimum (w.r.t. all 7 neighborhoods) is found.

The perturbation step of the ILS applies random block-rotations between three employees. The parameter α determines the strength of the perturbation. It corresponds to

Algorithm 2: Variable neighborhood descent procedure

```
   Data: Solution to improve $s$
   Result: Local minimum $s$
1  $k \leftarrow 1$
2  while $k \leq 7$ do
3      $\mathcal{N} \leftarrow SwapNeighborhood(k, s)$
4      while $hasRemainingNeighbor(\mathcal{N})$ do
5          if $currentNeighborCost(\mathcal{N}) < cost(s)$ then
6              $s \leftarrow currentNeighborSolution(\mathcal{N})$
7              $k \leftarrow 1$
8              $\mathcal{N} \leftarrow SwapNeighborhood(k, s)$
9          else
10             $\mathcal{N} \leftarrow nextNeighbor(\mathcal{N})$
11         end
12     end
13     $k \leftarrow k + 1$
14 end
```

the approximate ratio of assignments on which a rotation is applied.

3.2 Implementation details

The time restriction for reoptimizing a schedule is important for the interactive process. Reoptimization should last a few seconds. For computational experiments, the time limit is fixed to two seconds of computation on a standard computer. In order to meet this requirement and ensure good reoptimized solutions, two design features are considered for the implementation of the ILS procedure. First, high-level parallelism is used. Second, constraints are evaluated separately for computing the cost of solutions.

3.2.1 High-level parallelism

High-level parallelism is obtained from concurrent search processes [8]. The parallel version of ILS runs multiple concurrent ILS procedures that exchange asynchronously their best found solutions. This parallelism aims solely to speed up computation by using multi-thread strategy. Although the usage of multiple search-strategies or different parameter-settings could be beneficial, the choice of using the same ILS procedure has been motivated by the simplicity of the approach.

The procedure described in Algorithm 1 is the version of ILS when only one thread is used. The parallel version is obtained with very little modifications. After the application of the acceptance criterion, the ILS thread updates its best found solution with solutions possibly sent by other threads. If the best found solution has been obtained by the ILS thread during the last iteration, the thread sends its best found solution to other threads.

Using concurrent ILS threads may slightly favor diversification of the search in early iterations. However, no significant difference was found by varying the number of threads from 1 to 4 with the same cumulative computation time.

3.2.2 Constraints evaluation

To enhance further the performance of the ILS procedure, the method for evaluating the constraints has been tuned. In [11], the authors of the INRC2010 benchmark suggest an approach based on "numberings" [6] to evaluate the constraints. This evaluation method uses a set of counters to characterize a solution; then, the values of these counters are used to evaluate multiple constraints. By comparing different evaluation procedures, we came to the same conclusion as in [23], that the tested approach on "numberings" is less efficient than an evaluation method in which constraints are evaluated separately. Consequently, we opt for the latter.

Evaluating the constraints independently has two main advantages. First, it allows delta-evaluation of solutions for the exploration of neighborhoods. Delta-evaluation is a common method for evaluating the neighbors of a solution. During the local search, only the cost differences resulting from the moves are evaluated. Generally, the cost difference between two neighboring solutions is computed faster than a full evaluation of a solution. The second advantage concerns the modularity of the implementation. When constraints have independent evaluation (and delta-evaluation) procedures, it is easier to modify the problem model. The rank and weight of constraints can be easily modified. In addition, the introduction of a new constraint, such as the distance constraint and the preference constraint in the reoptimization model, does not necessitate any change in the initial evaluation procedures. Only a procedure for evaluating the new constraint is required.

4. COMPUTATIONAL EXPERIMENTS

The purpose of the experiments is twofold. The main point is to evaluate the proposed heuristic approach for solving the schedule reoptimization problem. In addition, the experiments aim at providing computational insights into the reoptimization problem and cascading effect.

Three aspects are examined in the computational experiments. In Section 4.1 the proposed heuristic approach is compared to existing results on the INRC2010 benchmark. In Section 4.2 the reoptimization procedure is compared to recovery procedures. The objective is to determine if the proposed global optimization approach is justified and efficient against local recovery procedures. Finally, in Section 4.3 the impact of the distance constraint introduced in the reoptimization problem is analysed.

The INRC2010 dataset[1] has been used for the experiments. This benchmark contains three classes of instances of increasing size called *Sprint*, *Medium* and *Long* instances. Only *Sprint* instances have been used as the number of assignments appears to be reasonable for the context of decision support. The benchmark contains 33 *Sprint* instances. For each instance, 10 employees have to be scheduled on a planning horizon of 28 days.

The problem instances in the INRC2010 benchmark are not reoptimization problems. The initial evaluation of the ILS procedure, in Section 4.1, is based on the original INRC2010 problem model; however, for the experiments presented in Sections 4.2 and 4.3, the benchmark has been adapted to the case of reoptimization.

All experiments were carried out on standard personal computer (Intel Core i5 2.5GHz, 8GB memory). The parameter α of the ILS procedure has been set to 5% for all tests.

4.1 Performance on INRC2010 benchmark

The ILS procedure has been evaluated first on the original INRC2010 problem model. The original INRC2010 problem model contains the constraints presented in Section 2.1, but does not use the reoptimization constraints and objectives introduced in Section 2.2.

In this section we present a comparison between the proposed heuristic and selected approaches from the literature. Three existing approaches have been selected based on the fact that, the average and standard deviation values are available, and the time limit for reported results has been adjusted with the INRC2010 time benchmark. The first method selected for the comparison is a hyper-heuristic presented in [4]. In the latter study, a set of 36 variants of a hyper-heuristic is evaluated. The best performing hyper-heuristic, namely *CF-GD-4* (Choice Function heuristic selection strategy, with a tournament size of 4, and Great Deluge acceptance criterion), has been retained for the comparison. The second method is a harmony search strategy proposed in [2]. The last method is an adaptive neighborhood search method presented in [15].

[1]The INRC2010 dataset is available at
`http://www.kuleuven-kulak.be/nrpcompetition`

The time limit obtained with the Inrc2010 time benchmark for the ILS procedure is 9.5 seconds on *Sprint* instances. For the comparison of ILS with existing approaches, only one thread has been used, and 20 independent runs have been performed on each instance[2].

Figure 2 summarizes the results of pairwise comparisons by instance between ILS and the three selected methods. A t-test with a p-value of 5% has been used to determine if the results per instance are significantly different. A Welch correction is applied to the t-tests for taking into account the differences in sample sizes and variances. Note that the difference in the number of compared instances is due to the availability of the results. In [4], 23 out of 33 instances are considered for the evaluation of the hyper-heuristic, and 30 instances are considered for the adaptive neighborhood search approach in [15].

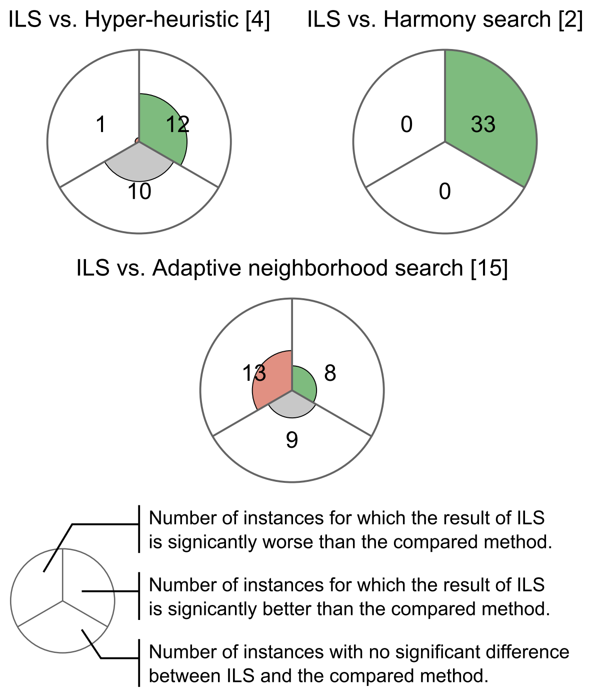

Figure 2: Comparative results of ILS on *Sprint* instances of the Inrc2010 benchmark.

The results reported in Figure 2 provide strong evidence that ILS outperforms the hyper-heuristic and the harmony search approach on *Sprint* instances of the Inrc2010 benchmark. The ILS procedure obtains significantly better results than the hyper-heuristic on 12 instances, and for 1 instance ILS is worse than the hyper-heuristic. The t-tests between ILS and the harmony search indicates that ILS obtains significantly better results than the harmony search on all instances. The third circular chart in Figure 2 indicates that the results of ILS are globally worse than those of the adaptive neighborhood search procedure. However, the difference is less marked than for the two first charts.

This first experiment shows that the ILS procedure provides competitive results. Considering the simplicity of ILS and the modularity of the implementation, these results suggest that the proposed heuristic approach is a good basis for an interactive optimization approach.

[2]Detailed results of ILS for *Sprint* instances are provided as supplementary material.

4.2 Reoptimization versus recovery

4.2.1 Generation of reoptimization problems

As explained previously, instances of the Inrc2010 benchmark have to be adapted to the reoptimization problem. The procedure for obtaining instances of reoptimization problem is the following. First, *Sprint* instances are solved with the ILS procedure. The objective considered for this optimization step is the lexicographic minimization of work regulation constraints and soft constraints as described in Equation 1 (Section 2.2). The obtained solutions are the initial (unperturbed) solutions. Then, random preferences are generated on these initial solutions in order to create the perturbed solutions. The perturbations correspond to 5 or 10 assignment preferences that are randomly generated. Note that the assignment preferences are generated so that they are not satisfied in the initial solutions. The resulting problems are the reoptimization of the perturbed solutions according to the objective presented in Equation 2 (Section 2.2). The latter objective contains, in addition to the work regulation constraints and soft constraints, an assignment preference constraint and a reoptimization distance constraint.

For each *Sprint* instance, 10 solutions to reoptimize are generated with 5 assignment preferences and 10 solutions with 10 assignment preferences. Initial solutions are obtained with 20 independent runs of the ILS procedure. To ensure good initial solutions, the parallel version of the ILS procedure is run on 4 threads with a time limit of 10 seconds.

4.2.2 Comparison between ILS and recovery

Considering the reoptimization problem, ILS is a “global” optimization approach in the sense that the search can potentially cover the whole search space thanks to the perturbation step. In concrete terms for the reoptimization, ILS is potentially able to optimize globally a perturbed solution instead of only applying local improvements to recover from the introduced perturbations. In this second experiment, ILS is compared to “pure” local-search procedures called recovery procedures. The objective is to determine if a global optimization approach is justified for the reoptimization when only small perturbations are introduced, and to evaluate the efficiency of ILS against recovery procedures.

For the reoptimization, the parallel version of the ILS procedure is run on 4 threads and the time limit is set to 2 seconds. The results are compared with those of two recovery procedures. The first recovery procedure is a 1-swap descent procedure. It is a local-descent procedure that improves the perturbed solution using the 1-swap neighborhood structure. Only improving moves within neighborhoods are accepted. The procedure stops when a local-optimum, w.r.t. the 1-swap neighborhood structure, is found. The second recovery procedure is a variable neighborhood descent procedure using block-swap neighborhood structures with block sizes of 1 to 7. This procedure corresponds to the local-search step of the ILS. These two recovery procedures only improve locally the perturbed solutions. In this way, the cost difference between ILS and recovery procedures would reflect the contribution of the global optimization.

Table 1 reports the average number of assignment preferences satisfied with the ILS reoptimization procedure, 1-swap recovery procedure, and 1-7-swap recovery procedure. In average, the reoptimization with ILS satisfied 2.81 and 5.49 preferences respectively for 5 and 10 introduced assignment preferences. The difference between 1-swap and ILS is substantial with an average gain of more than 2 preferences satisfied when 10 preferences are introduced. The difference between 1-7-swap and ILS is less important and necessitate a closer look at the cost of reoptimized solutions.

Table 1: Number of assignment preferences satisfied by recovery procedures and ILS

Introduced preferences	Average number of satisfied preferences		
	1-swap	1-7-swap	ILS
5	1.67	2.22	2.81
10	3.18	4.30	5.49

In Table 2 the cost of reoptimized solutions are compared between ILS and 1-swap, and between ILS and 1-7-swap. Note that it is not possible to report an average difference value due to the lexicographic objective function. (E.g. a solution with 3 unsatisfied preferences and 10 unsatisfied soft constraints, noted $\langle 3, 10\rangle$, is better than a solution with a cost $\langle 3, 12\rangle$, but worse than a solution $\langle 4, 8\rangle$. Here, it does not make sense to average the values of unsatisfied soft constraints.) In Table 2, the advantage of ILS against 1-7-swap is apparent. When 5 preferences are introduced, 63.94% of the solutions reoptimized with ILS are better than solutions obtained with 1-7-swap. Only 36.06% of resulting solutions have the same cost.

Table 2: Comparison of the cost of reoptimized solutions

Pref.	ILS vs. 1-swap		ILS vs. 1-7-swap	
	same cost	better cost	same cost	better cost
5	21.21%	78.79%	36.06%	63.94%
10	6.67%	93.33%	17.27%	82.73%

The reoptimization with ILS naturally outperformed the recovery procedures. However, the results reveal the difficulty of the reoptimization. The introduction of 5 assignment preferences is rather small considering the size of problem instances. In this context, the gain of ILS in Table 2 is substantial. These comparative results support the fact that a global optimization approach is justified to tackle the reoptimization problem, even though the perturbation is small.

4.3 Impact of the distance constraint

In this section we analyse the impact of the distance constraint on the results of ILS. The introduction of a distance constraint in the reoptimization problem raises two questions. First, considering that the constraint is ranked last in the objective function of the reoptimization problem, the distance between the initial solution and the reoptimized solution may not be significantly minimized. The second issue concerns the potential alteration of the local-search process. During the local-search step of the ILS procedure, the distance constraint guides the search towards the initial solution. This tendency may hinder the search to explore solutions that are far from the initial one. Consequently, the constraint may reduce the effectiveness of ILS.

These issues are investigated by comparing the result of ILS on reoptimization problems with and without the distance constraint. The set of reoptimization problems without distance constraint is obtained by removing the corresponding constraint from the problems considered in the previous section. The previous results of ILS are compared to the results obtained by the same procedure on problems without distance constraint.

Table 3 reports the average distance to initial solutions resulting from the reoptimization with and without distance constraint. When the distance constraint is taken into account, the distance to the initial solution is reduced by 43%. In addition, 86% of the solutions obtained with the distance constraint are closer or at equal distance to the initial solutions. These results confirm that the distance constraint significantly reduces the distance between the initial solution and the reoptimized solution, although the constraint is ranked last in the objective function.

Table 3: Average distances to initial solutions obtained with and without distance constraint

Introduced preferences	Average distance to initial solution	
	with constraint	without constraint
5	16.86	29.55
10	27.02	47.28

In order to examine the possible impact of the distance constraint on the cost of reoptimized solutions, a Wilcoxon signed-rank test has been performed. The test compares the costs with and without distance constraint (paired by problem instance). The results of the test, using a significance level of 5%, show no significant difference. This result supports the fact that the distance constraint has no significant impact on the effectiveness of ILS.

5. CONCLUSION

In this paper, we proposed a heuristic approach for the reoptimization of schedules in the context of interactive optimization. The objective was to design a method suitable for a decision support tool which implies to find a good balance between simplicity, modularity and efficiency of the reoptimization method.

The proposed problem model contains two constraints specific to the reoptimization that are added to an initial shift scheduling problem using a lexicographic ordering. This order between constraints is a convenient way to integrate the new constraints without introducing additional parameters such as weight values. The procedure for reoptimizing schedules is an ILS. This metaheuristic is conceptually simple, and the proposed implementation has only one parameter to be set. Two design features are considered to obtain appropriate performances for the interactive reoptimization process. First, high-level parallelism is used. Second, constraints are evaluated separately in order to allow a delta-evaluation of solutions and favor the modularity of the implementation.

A set of computational experiments has been performed to evaluate the effectiveness and adequacy of the proposed heuristic approach for reoptimizing schedules. The ILS procedure has been evaluated first on a shift scheduling problem and compared to three approaches from the literature. The results show that ILS provides competitive results in spite of the simplicity of the procedure. The second experiment compared ILS with two recovery procedures in order to determine if a global optimization approach is justified for the reoptimization of schedules. This comparison reveals that ILS provides significant cost gain, and support the fact that a global optimization approach is required to tackle the reoptimization problem, even though the perturbation is small. The last experiment analysed the impact of the distance constraint on the results of ILS. The results of the latter experiment show that the distance constraint allows to reduce significantly the distance between the initial solution and the reoptimized solution. In addition, the introduction of this constraint within the reoptimization problem model does not have significant impact on the effectiveness of ILS.

Future work will focus on the evaluation of the interactive reoptimization approach with real users and the analysis of usability aspects.

6. ACKNOWLEDGMENTS

This work was supported by the Deutsche Forschungsgemeinschaft (DFG), under grant ME 4045/2-1, for the project “Interactive metaheuristics for optimization-based decision support systems”. The author wishes to thank Sigrid Knust for her suggestions on preparing the manuscript.

7. REFERENCES

[1] G. Ausiello, V. Bonifaci, and B. Escoffier. *Computability in Context: Computation and Logic in the Real World*, chapter Complexity and Approximation in Reoptimization, pages 101–129. Imperial College Press, 2007.

[2] M. A. Awadallah, A. T. Khader, M. A. Al-Betar, and A. L. Bolaji. Nurse rostering using modified harmony search algorithm. In *SEMCCO*. LNCS, 7077:27–37, Springer Berlin Heidelberg, 2011.

[3] J. P. Barthélemy, R. Bisdorff, and G. Coppin. Human centered processes and decision support systems. *European Journal of Operational Research*, 136(2):233–252, 2002.

[4] B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, and G. V. Berghe. One hyper-heuristic approach to two timetabling problems in health care. *Journal of Heuristics*, 18:401–434, 2012.

[5] E. K. Burke, P. D. Causmaecker, G. V. Berghe, and H. V. Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499, 2004.

[6] E. K. Burke, P. D. Causmaecker, S. Petrovic, and G. V. Berghe. Fitness evaluation for nurse scheduling problems. In *CEC*. 2:1139–1146, 2001.

[7] E. K. Burke, T. Curtois, R. Qu, and G. V. Berghe. A time predefined variable depth search for nurse rostering. *INFORMS Journal on Computing*, 25(3):411–419, 2013.

[8] T. Crainic and M. Toulouse. *Handbook in Metaheuristics*, chapter Parallel Strategies for Meta-heuristics, pages 475–513. Springer, 2003.

[9] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[10] P. Hansen and N. Mladenović. *Handbook of Metaheuristics*, chapter Variable Neighborhood Search, pages 145–184. Kluwer Academic, 2003.

[11] S. Haspeslagh, P. D. Causmaecker, A. Schaerf, and M. Stølevik. The first international nurse rostering competition 2010. *Annals of Operations Research*, Online, DOI: 10.1007/s10479-012-1062-0, 2012.

[12] F. Hutter, T. Bartz-Beielstein, H. Hoos, K. Leyton-Brown, and K. Murphy. *Empirical Methods for the Analysis of Optimization Algorithms*, chapter Sequential Model-Based Parameter Optimisation: an Experimental Investigation of Automated and Interactive Approaches, pages 361–411. Springer, 2010.

[13] G. W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher. Human-guided search. *Journal of Heuristics*, 16(3):289–310, 2010.

[14] H. R. Lourenço, O. C. Martin, and T. Stützle. *Handbook of Metaheuristics*, chapter Iterated Local Search, pages 321–353. Kluwer Academic, 2003.

[15] Z. Lü and J.-K. Hao. Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research*, 218:865–876, 2012.

[16] B. Maenhout and M. Vanhoucke. An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research*, 38(10):1400–1411, 2011.

[17] B. Maenhout and M. Vanhoucke. Reconstructing nurse schedules: Computational insights in the problem size parameters. *Omega*, 41(5):903–918, 2013.

[18] K. Miettinen, F. Ruiz, and A. P. Wierzbicki. *Multiobjective Optimization*, chapter Introduction to Multiobjective Optimization: Interactive Approaches, pages 27–58. Springer, 2008.

[19] M. L. Pinedo. *Scheduling Theory, Algorithms, and Systems*, chapter Design and Implementation of Scheduling Systems: Basic Concepts, pages 459–483. Springer, fourth edition, 2012.

[20] B. Roy. Main sources of inaccurate determination, uncertainty and imprecision in decision models. *Mathematical and Computer Modelling*, 12(10–11):1245–1254, 1989.

[21] H. Shachnai, G. Tamir, and T. Tamir. A theory and algorithms for combinatorial reoptimization. In *LATIN*. LNCS, 7256:618–630, Springer Berlin Heidelberg, 2012.

[22] H. Takagi. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[23] C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, and E. Housos. A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, 219(2):425–433, 2012.

[24] J. Wessels and A. P. Wierzbicki. *Model-Based Decision Support Methodology with Environmental Applications*, chapter Model-Based Decision Support, pages 9–28. Springer, 2000.

[25] A. Zych. *Reoptimization of NP-hard problems*. PhD thesis, ETH Zürich, 2012. Nr. 20257.