An Adaptive Large Neighborhood Search Method to Plan Patient's Journey in Healthcare

Olivier Gerard, Corinne Lucet, Laure Brisoux Devendeville, Sylvain Darras

An Adaptive Large Neighborhood Search Method to Plan Patient's Journey in Healthcare

Gérard Olivier[1,2], Lucet Corinne[2], Brisoux Devendeville Laure[2], and Darras Sylvain[1]

[1] MIS Laboratory (EA 4290), University Picardie Jules Verne, France
{olivier.gerard, corinne.lucet, laure.devendeville}@u-picardie.fr
[2] Evolucare Technologies, France
{o.gerard, s.darras}@evolucare.com

Abstract. In this paper an adaptation of the Adaptive Large Neighborhood Search (ALNS) to a patient's care planning problem is proposed. We formalize it as an RCPSP problem that consists of assigning a start date and medical resources to a set of medical appointments. Different intensification and diversification movements for the ALNS are presented. We test this approach on real-life problems and compare the results of ALNS to a version without the adaptive layer, called (¬A)LNS. We also compare our results with the ones obtained with a 0-1 linear programming model. On small instances, ALNS obtains results close to optimality, with an average difference of 1.39 of solution quality. ALNS outperforms (¬A)LNS with a gain of up to 18.34% for some scenarios.

Keywords: ALNS · planning · healthcare · optimization

1 Introduction

Improving the health care system is one of the biggest challenges many countries will have to face over the coming years. The complexity of scheduling problems in the healthcare domain is an issue that is increasingly being highlighted by healthcare facilities. This kind of problem belongs to the Resource Constrained Project Scheduling Problem (RCPSP) family that is NP-Hard [7, 2]. The RCPSP problem consists of finding the best assignment of resources and start times to a set of activities. In healthcare it usually involves finding a starting date and medical resources (medical staff, rooms and medical equipment) for an appointment with a patient.

Nowadays, planning the patient's care is mostly done by hand, a difficult and time-consuming task due to the number of appointments and resources to consider, that can be challenged by various kinds of unexpected events. Scheduling problems have been the subject of many studies for decades in various fields [1, 3], and they are of increasing interest in the healthcare domain [6, 12]. The structure of considered problems differs according to the institutions, their size and the number of resources taken into account. This article is focused on scenarios designed with various planners from different health care facilities in France who

deal with real-life problems every day. These scenarios focus on the planning of patient's care in different kinds of institutions. In this work we propose an Adaptive Large Neighborhood Search (ALNS) metaheuristic able to deal with large instances derived from these various real-world healthcare scenarios.

This article is structured as follows. In section 2 we describe our problem. In section 3 we present our ALNS algorithm, with the different movements used. In section 4 we give further details on the scenarios and the corresponding results obtained by the ALNS metaheuristic. We compare the results of this method with the results obtained by the 0-1 linear programming model presented in [5]. In section 5, we conclude with some remarks and perspectives.

2 Problem definition

Our problem can be stated as follows. The horizon H is decomposed into timeslots. We have a finite set of resources R and each resource $r \in R$ is characterized by a set of properties Π_r that determines which roles a resource will be able to hold in an appointment. Availability of each resource is known. A is a set of appointments to be planned, such that each appointment $a \in A$ is characterized by its duration $duration_a$, a feasibility interval of time $[ES_a, LS_a]$, $qtreq_a^\pi$ the amount of resources with property π required by a. $Essential_a$ and $Emergency_a$ are two coefficients used to respectively quantify the importance and the urgency of an appointment a. Both are used by planners to define priorities on some appointments and to specify which ones should be set as soon as possible within their feasibility interval, therefore they both occur as penalties in the objective function. $PreAssigned_a$ is a set of couples $(resource, property)$ pre-assigned to a. To each appointment a could be associated a set of appointments $pred_a$ that must be planned before a.

We define the triplet (a, t_a, R_a), where a is an appointment, t_a the starting date for a and R_a the set of resources assigned to a. A valid solution Sol is a set of triplets that respect the hard time and resource constraints. We denote A_{Sol} the set of scheduled appointments and $A_{\overline{Sol}}$ the set of unscheduled appointments. The quality of a solution is evaluated by the objective function f that is the sum of weights $Essential_a$ of unplanned appointments $a \in A_{\overline{Sol}}$ plus the sum of delay impacts of planned urgent appointments. The aim is to find a valid solution while minimizing objective function f defined in equation 1.

$$f(Sol) = \sum_{a \in A_{\overline{Sol}}} Essential_a + \sum_{a \in A_{Sol}} \frac{t_a - ES_a}{LS_a - ES_a} \times Emergency_a \qquad (1)$$

3 Adaptive Large Neighborhood Search

Adaptive Large Neighborhood Search is based on the Large Neighborhood Search framework defined in [10]. ALNS was first introduced in [9] and was applied on various problems, such as scheduling problems [4, 8]. Different movements are

iteratively applied to a solution in order to explore its neighborhood. Basically, a solution is partially destroyed then reconstructed using destruction and construction heuristics. In a scheduling problem, it usually consists of removing a number of appointments from a solution and then trying to reinsert them.

Algorithm 1: ALNS

```
1  begin
2      Sol ← greedy() ;
3      Sol_best ← Sol ;
4      ∀m ∈ M, w_m^0 ← 1/|M| ;
5      while stopping criterion are not met do
6          for segment s do
7              m ← RouletteWheel(M, w^s) ;
8              Sol' ← m(Sol) ;
9              π_m is updated according to f(Sol') ;
10             Sol' ← diversification(Sol') ;
11             Sol ← Sol' ;
12             if f(Sol) < f(Sol_best) then
13                 Sol_best ← Sol ;
14             end
15         end
16         update weights w_m^{s+1} for each movement m ;
17     end
18     return Sol_best ;
19 end
```

The general algorithm of our ALNS based on [9] is given in Algorithm 1. An initial solution is computed using a greedy algorithm described in Section 3.1. All movements m in the set of movements M are equally weighted at the beginning of the algorithm. As suggested in [9], the search is divided into blocks of consecutive iterations, called *segments*. For each iteration in a segment s, a movement m is chosen with a roulette wheel selection according to its weight w_m^s and is applied to the current solution Sol. During the course of a segment, a score π_m is associated with each movement m. π_m represents the total reward of the movement, relative to the results of its use. At the end of each segment s, weights w_m^s are updated according to scores π_m. If the search stagnates, diversification movements such as *Swap* or *Left Shift* can be applied to the solution. In an extended period without improvement, a *reset* can also be performed on the current solution Sol. This process is iterated until stop conditions are met. The various movements that ALNS uses for diversification are described below.

3.1 Greedy Algorithm

To build the initial solution, a simple greedy algorithm is used. The order that the greedy algorithm uses to schedule each unscheduled appointment is random. For an appointment a the greedy searches for the first timeslot t_a on which a set of resources R_a that exactly matches its resource requirement is available. This search starts either at ES_a or at a random timeslot $t \in [ES_a, LS_a]$.

3.2 Weight adjustment

As stated above, the search is divided into segments. The score π_m of all movements m is set to zero at the start of each segment. At each iteration within a segment, the score π_m of selected movement m is increased by adding a reward σ according to the following conditions:

$$\sigma = \begin{cases} \sigma_1, & \text{if the produced solution is a new best overall solution} \\ \sigma_2, & \text{if the produced solution is better than the current one} \\ \sigma_3, & \text{if the produced solution is worse than the current one} \end{cases} \quad (2)$$

At the end of each segment s, weight w_m^{s+1} is updated using formula 3:

$$w_m^{s+1} = (1 - r)\ w_m^s\ +\ r\ (\frac{\pi_m}{\theta_m}) \quad (3)$$

where θ_m is the number of times movement m has been used during segment s. The reaction factor r controls the strength of the adjustment from one segment s to the following segment $s\ +\ 1$. If $r = 0$, there is no change and if $r = 1$, the weights for the new segment $s + 1$ only depends on the performance during the previous segment. Movements are selected using a roulette wheel selection method. The number of iterations of a *segment*, the different scores σ_1, σ_2 and σ_3 and the reaction factor r are parameters of the ALNS algorithm.

3.3 Intensification movements

We propose different intensification movements. These movements are assessed by the ALNS and used according to their performance during execution.

Random destruction and greedy reconstruction This movement removes a random number of appointments from the current solution. Then it tries to plan all unscheduled appointments using the greedy algorithm described above. Time complexity of this movement: $O(|A_{\overline{Sol}}| \times |H| \times |R|)$.

Random destruction and optimal reconstruction This movement also removes a random number of appointments and uses an optimal reconstruction to try to plan as many unscheduled appointments as possible. The order in which the appointments will be considered is obtained by a roulette wheel selection, its probabilities being calculated from the importance coefficients $Essential_a$ of unscheduled appointments a. For each a of them and for each timeslot $t \in [ES_a, LS_a]$, a heuristic searches for the best possible set of resources R_a. This choice is based on a stress score α_r for each resource $r \in R_a$. α_r increases each time a movement fails to assign r to an appointment. Conversely, it decreases each time resource r is successfully assigned to an appointment. The most requested resources will fail more often, and therefore have a higher stress score. This allows the ALNS to keep track of the most stressed resources. The heuristic favors for an appointment a the sets of resources R_a with the lowest cumulated stress score. t_a is chosen according to the stress score of R_a and to $Emergency_a$: if appointment a is urgent, earlier timeslots close to ES_a are preferred. Appointments that cannot be rescheduled remain in $A_{\overline{Sol}}$. Time complexity of this movement: $O(|A_{\overline{Sol}}| \times |H| \times |R|)$.

Difficulty based destruction and optimal reconstruction To each appointment a is associated a difficulty score δ_a. Whenever a movement fails to schedule a, its difficulty score δ_a increases. Conversely, each time an appointment a is scheduled, its difficulty score δ_a decreases. This allows the ALNS to keep track of the most difficult appointments to schedule. Among all scheduled appointments, this movement uses a roulette wheel selection based on their difficulty score to select one appointment to remove from A_{Sol}, the other are randomly chosen in A_{Sol}. Then it tries to plan all unscheduled appointments ordered by their decreasing difficulty score (the most difficult being scheduled first). Time complexity of this movement: $O(|A_{\overline{Sol}}| \times |H| \times |R|)$.

Targeted destruction This movement extracts one unscheduled appointment a according to its difficulty score δ_a. Next it tries to find the best timeslot t_a to plan a by removing some scheduled appointments in order to satisfy all resources needed by a. First we randomly select a set $T \subseteq [ES_a, LS_a]$ of timeslots where a could be scheduled by relaxing resource constraints. For each timeslot $t \in T$ we compute all combinations C_t^i of appointments whose removal frees up resources that allow a to be planned. Next we select a timeslot t with a combination C_t^* by probabilistic rules based on *Essential* and *Emergency* factors of contributive appointments. All appointments in C_t^* are unplanned and a is planned on t. Finally, the greedy algorithm is used to place as many unscheduled appointments as possible. Time complexity of this movement: $O(\max(A, H) \times |R| \times |T|)$.

3.4 Diversification movements

Two diversification movements *Swap* and *Left Shift* are used during the execution of the ALNS. They allow the current solution to be perturbed, lowering the odds

that the search remains trapped in a local minimum. They are applied to the current solution when a preset number of iterations without improvement is reached.

Swap The *Swap* randomly chooses two triplets (a_1, t_{a_1}, R_{a_1}) and (a_2, t_{a_2}, R_{a_2}) of Sol and exchanges their start date t_{a_1} and t_{a_2}, if the availability of resources allows it. Otherwise the exchange is aborted. Time complexity of this movement: $O((duration_{a_1} + duration_{a_2}) \times |R|)$.

Left Shift This movement tries to shift the appointments of a subset of randomly chosen resources so that their respective schedules are as compact as possible. The start date t_a of each affected appointment a is brought as close as possible to its ES_a date. Time complexity of this movement: $O(|A| \times |H| \times |R|)$.

3.5 Restart

If the search stagnates for too long, the solution will undergo a complete restart. All appointments $a \in A_{Sol}$ are unscheduled, then the greedy algorithm described above is used. However the greedy will consider appointments ordered by their decreasing difficulty score δ. The most difficult appointments will be processed first. This way we expect to schedule difficult appointments in priority. Time complexity of this movement: $O(|A| \times |H| \times |R|)$.

4 Experimentations & Results

4.1 Experimentations

We generated 80 instances from four scenarios presented in [5] to test our ALNS algorithm. Instances are generated by varying some parameters: size of the instance, essential and emergency coefficients, precedence relationship and resource availabilities. As suggested in [11], we compare the results of ALNS to a version without the adaptive layer, called (¬A)LNS to assess the effect of the adaptive layer. Both ALNS and (¬A)LNS were implemented in C# and tests were run on an Intel i5-8350U processor. We also compare ALNS results to the optimal solutions reached by the linear programming model 0-1 presented in [5] and implemented under CPLEX 12.8.0.0.

We set the parameters of ALNS as follows. The maximum number of iterations in a *segment* was set to 100. The rewards σ_1, σ_2 and σ_3 were set respectively to 75, 20, and 0. For the ALNS, the reaction factor r is set to 0.08. For the (¬A)LNS, r is set to 0. The number of destroyed appointments mentioned in section 3.3 is randomly picked between 5 and 15. For the ALNS and (¬A)LNS the computation time was limited to two minutes when for CPLEX it was limited to two hours.

Table 1. Comparisons of ALNS results with optimal solutions

Instance	ALNS		CPLEX		Δ_f	Gap_f
	$f(Sol_{best})$	$\lvert A_{\overline{Sol}_{best}}\rvert$	$f(Sol_{best})$	$\lvert A_{\overline{Sol}_{best}}\rvert$		
SurgDep 1	0	0	0	0	0	0%
SurgDep 2	0	0	0	0	0	0%
SurgDep 3	1	1	0	0	1	100%
SurgDep 4	3.75	2	1.48	0	2.27	153.17%
SurgDep 5	14.54	3	9.54	0	5	52.4%
SurgDep 6	0	0	0	0	0	0%
SurgDep 7	4.31	2	1.48	0	2.83	196.13%
SurgDep 8	0	0	0	0	0	0%

Table 2. Comparisons of ALNS results with (¬A)LNS

Scenario	$\lvert A\rvert$	$\lvert H\rvert$	ALNS Average Gap_f	ALNS Average Δ_f	Average $\lvert A_{Sol_{best}}\rvert$ / $\lvert A\rvert$
SurgDep	16	52	-1.76%	-0.14	93.06%
	48	52	-12.61%	0.56	90.51%
RehabCenter	96	48	-5.26%	-1.04	99.45%
	288	48	-7.16%	-11.02	99.38%
Admission	136	240	-18.34%	-8.80	94.22%
	408	240	-2.93%	24.41	95.66%
CardioRehab	160	240	-7.52%	-2.61	98.62%
	480	240	-17.92%	25.88	98.78%

4.2 Results

Our different approaches have been applied to all generated instances. Comparisons between the optimal results obtained by CPLEX and those obtained by ALNS are reported in Table 1. For CPLEX and ALNS we give $f(Sol_{best})$ and the number of unplanned appointments $|A_{\overline{Sol}_{best}}|$, the difference Δ_f between CPLEX and ALNS results and the Gap_f from CPLEX ($Gap_f = (ALNS - CPLEX)/CPLEX$). Not surprisingly, CPLEX cannot conclude on large instances, and we can only obtain optimality on small instances (SurgDep scenario, the smallest of our scenarios with 16 appointments to schedule on one day). We note that ALNS obtains results close to optimality, with an average difference of 1.39 of solution quality computed by Equation 1.

We next compare the results obtained by ALNS and (¬A)LNS. Results are reported in Table 2. We give for each scenario the number of appointments $|A|$, the number of timeslots $|H|$, the average Gap_f ($Gap_f = (ALNS - (\neg A)LNS)/(\neg A)LNS$) from (¬A)LNS, the average difference Δ_f between (¬A)LNS and ALNS and the average percentage of scheduled appointments

$|Sol_{best}|$ / $|A|$. We see that ALNS outperforms (¬A)LNS on all scenarios, especially on the largest ones with an improvement of 17.92% on the scenario with the highest number of appointments to schedule. The average gain obtained with ALNS exceeds the average gain reported in [11]. These results suggest that some combinations of movements may be especially efficient depending on the scenario. They are therefore favored by ALNS and thus lead to better solutions.

5 Conclusion & Perspectives

In this paper we presented an adaptation of the ALNS framework to a patient's healthcare planning problem. Such a method, whose execution times are very short and provide solutions quite close to the optimum, is very interesting for applications in the real world. Indeed, the first results of this approach are promising especially on large instances. Further tests should be conducted to study the impact of parameters on the performance of ALNS. We would also investigate how the weights of movements evolve according to the scenarios in order to understand why some of them seem more efficient than others from one scenario to another.

Acknowledgements. This project is supported by LORH project (CIFRE N^o 2018/0425 between Evolucare and MIS Laboratory).

References

1. Anthony, R.: Planning and control systems: a framework for analysis. Division of Research, Graduate School of Business Administration, Harvard University (1965)
2. Baptiste, P., Laborie, P., LePape, C., Nuijten, W.: Constraint-based scheduling and planning. In: Foundations of artificial intelligence, vol. 2, pp. 761–799. Elsevier (2006)
3. Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., Sterna, M., Weglarz, J.: Handbook on Scheduling. Springer (2019)
4. Bueno, E.: Mathematical modeling and optimization approaches for scheduling the regular-season games of the National Hockey League. Ph.D. thesis, École Polytechnique de Montréal (2014)
5. Gérard, O., Devendeville, L., Lucet, C.: Planning problem in Healthcare domain. In: 17th International Workshop on Project Management and Scheduling PMS'20 (2021)
6. Hall, R. (ed.): Handbook of Healthcare System Scheduling. International Series in Operations Research & Management Science, Springer (2012)
7. Johnson, D., Garey, M.: Computers and intractability: A guide to the theory of NP-completeness. WH Freeman (1979)
8. Muller, L.: An Adaptive Large Neighborhood Search Algorithm for the Multi-mode RCPSP. DTU Manag. Eng **3**, 25 (2011)
9. Ropke, S., Pisinger, D.: An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation science **40**(4), 455–472 (2006)

10. Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: 4th International Conference on Principles and Practice of Constraint Programming CP'98. pp. 417–431. Springer (1998)
11. Turkeš, R., Sörensen, K., Hvattum, L.: Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. European Journal of Operational Research (2020)
12. Zhang, X., Ma, S., Chen, S.: Healthcare service configuration based on project scheduling. Advanced Engineering Informatics **43**, 101039 (2020)