Politecnico di Torino

Stochastic Optimization

Assignment

Maino, Giacomo - 338682
Molinari, Mattia - 337194
Perlo, Alessandro - 337131

A.A. 2024/2025

Abstract

This report provides a comprehensive overview of methodologies and decisions taken during the development of a metaheuristic algorithm to solve the Integrated Healthcare Timetabling Competition 2024. The proposal is based on the Tabu Search algorithm, which has been opportunely adapted to the problem at hand.

1 Problem Overview

Integrated healthcare scheduling involves the coordination of resources across multiple services within a unified healthcare system. Its primary goal is to streamline and optimize patient flow throughout various departments and facilities within the hospital. The advantages of integrated healthcare optimization are numerous, including an enhanced patient experience, increased operational efficiency, and better utilization of resources across the entire hospital network.

The *Integrated Healthcare Timetabling Problem* (IHTP) [1] transpose the aforementioned concept into a combinatorial optimization problem, bringing together three distinct operational challenges: Patient-to-Room Assignment (PRA), Nurse-to-Patient Assignment (NPA) AP: [NRA?] and Surgical Case Planning (SCP). More precisely, the IHTP requires to establish *(i)* the date of admission for patients, *(ii)* the room where they are admitted and *(iii)* operating theater where they are operated, *(iii)* the room nurses are assigned to during each shift. The problem is subject to several constraints, specific to a sub-problem or arising from their interaction. The constraints are the following:

- *Hard Constraints*, which must be satisfied at all costs:
  - No gender mixing in rooms;
  - Patients can be assigned only to a compatible room;
  - Surgeon cannot work overtime;
  - Operating theaters capacity cannot be exceeded;
  - Mandatory patients need to be assigned during the provided period;
  - Patients cannot be scheduled before their release date;
  - Room capacity cannot be exceeded;
  - Rooms with patients assigned must be covered.
- *Soft Constraints*, which can be violated at the cost of a penalty:
  - Age groups mixing in rooms;
  - Nurses require a minimum skill level to cover a certain patient.
  - Continuity of care for patients from nurses;
  - Nurses have a maximum workload during a shift;
  - Number of open operating theaters per day should be minimized;
  - Number of surgeon transfer between operating theaters should be minimized;
  - Patients should be admitted the earliest possible;
  - Number of unscheduled non-mandatory patients should be minimized.

Given a test instance, all the information required are provided, including pre-admitted occupants, patients, nurses, surgeons, operating theaters, rooms, shifts, scheduling dimension in days and penalty weights for soft constraints.

2 Approach

The proposed approach involves the application of the *Tabu Search* (TS) algorithm, which is it is a local search method that aims to find the global optimum of a combinatorial optimization problem. [2]

The main idea behind TS is to enhance local search performances by relaxing its rules: in particular, worsening moves can be accepted if no improving move is available, which prevent from getting stuck in local optima, and previously-visited solutions are discouraged, which prevent cycles. This last aspect is implemented storing moves in a queue, called *tabu list*, the dimension of which needs to be tuned. In order to mitigate the constraints introduced bu the list, the algorithm also includes the possibility to accept forbidden moves when they met a certain *aspiration criterion*, for example if they lead to a better solution than the current one.

The algorithm is the following:

```
algorithm TabuSearch(S, maxIter, f, neighborhoods, aspirationCriteria):
// INPUT
//     S = the search space
//     maxIter = the maximal number of iterations
//     f = the objective function
//     neighborhoods = the definition of neighborhoods
//     aspirationCriteria = the aspiration criteria
// OUTPUT
//     The best solution found

Choose the initial candidate solution s in S
s* <- s  // Initialize the best-so-far solution
k <- 1

while k <= maxIter:
    // Sample the allowed neighbors of s
    Generate a sample V(s, k) of the allowed solutions in N(s, k)
    // s' in V(s, k) iff (s' not in T) or (a(k, s') = true)

    Set s to the best solution in V(s, k)

    // Update the best-so-far if necessary
    if f(s) < f(s*):
        s* <- s

    Update T

    // Start another iteration
    k <- k + 1

return s*
```

The next paragraphs will discuss the main aspects that need to be addressed in order to adapt TS to IHTP.

2.1 Solution Representation

The current status of the problem is represented by three main data structures:

- *Patient Admission and Scheduling (PAS)*: boolean array of size $n_{\text{days}} \times n_{\text{rooms}} \times n_{\text{patients}}$. For each patient, it indicates which room they are assigned to during all the days of recovery.
- *Surgical Case Planning (SCP)*: boolean array of size $n_{\text{days}} \times n_{\text{patients}} \times n_{\text{surgeons}} \times n_{\text{operating theaters}}$. For each patient, it indicates the day of admission (i.e. surgery) along with the predetermined surgeon and the chosen operating theater.
- *Nurse-to-Room Assignment (NRA)*: boolean array of size $n_{\text{shifts}} \times n_{\text{rooms}} \times n_{\text{nurses}}$. For each nurse, it indicates which room they are assigned to during each shift.

In order to lower computational time of certain soft constraints, two other structures are used: *Workload* and *Skill level*, both arrays of size $n_{\text{shifts}} \times n_{\text{rooms}} \times n_{\text{patients}}$, storing the workload and skill level required by a patient assigned to a room during a certain shift.

2.2 Starting Point

As already mentioned, a test instance starts with some patients already admitted to the hospital, called occupants. AP: [We refer to unstable states of the problem as those where hard constraints are not satisfied.] This configuration clearly is an unstable state of the problem, since no hard constraint is guaranteed to be satisfied. With these premises, two approaches can be considered in order to generate a starting point: create a stable configuration from scratch using advanced and exhaustive combinatorial techniques, which is a very complex task both in terms of time and space complexity and would partly defy the purpose of this work; otherwise, accept the unstable configuration as it is and let TS find a stable solution by opportunely tuning its behavior. The

second approach is the one that has been chosen, since it is more efficient and allows to fully exploit the potential of the algorithm. AP: [Basically, we are enforcing all hard constraints when generating the neighborhood, but constraints "Madatory patients need to be assigned during the provided period" and "Rooms with patient assigned must be covered". Eventually, these hard constraints will hopefully be satisfied by the algorithm due to the choices that will be discussed in the next paragraphs.] The steps taken to direct the algorithm in a direction of stability are presented in the next paragraphs.

2.3 Neighborhood Definition

Given the nature of the problem and the constraints involved, the type of moves performed by the algorithm are the following:

- Patient
  - Scheduling: for a patient, admit them to a room on a day and establish the operating theater.
  - Unscheduling: for a patient, remove them complexly from the schedule.
- Nurse
  - Assignment: for a nurse, assign them to a room during a shift.
  - Unassignment: for a nurse, remove them from an assigned room during a shift.

The neighborhood is defined as the set of all possible moves that can be performed on the current solution, meaning that for each of the aforementioned types of moves all the possible combinations of patients, rooms, days, operating theaters, nurses and shifts are considered. Obviously, moves effectively considered are those that do not violate hard constraints and do not destabilize the solution. Moreover, given that the current status of the problem is not guaranteed to be stable, the neighborhood needs to be ulteriorly filtered if necessary: more precisely, when there are still non-admitted mandatory patients, these have priority over the others and scheduling regarding optional patients is ignored.

Since TS operates comparing the current moves with the ones stored in the tabu list, it is also necessary to define a way to compare two moves. For patient moves, the comparison is performed by checking the type and all the parameters involved (i.e. patient, room, day, operating theater), meaning that only the same exact action is considered tabu. For nurse moves, on the other hand, equality is asymmetric and comparison follows these rules: when considering an assignment, it can only be equal to an assignment with the same parameters (i.e. nurse, room, shift), so an exact copy; when considering an unassignment, it can be equal to both an assignment or unassignment with equal nurse and room, independently of the shift. The reason behind this choice is that the nurses assignment is a vital part of the schedule, since room coverage needs to be granted in order to admit patients: the strategy is to force the algorithm in a direction where nurses are more likely to be assigned and avoids the situation where nurses are assigned and unassigned multiple times consecutively just because the shift changes.

As a consequence of all these choices, we expect the optimization to be more stable for hard constraint satisfaction, specifically mandatory patients admission and room coverage, but less effective in minimizing soft constraints focusing on nurses.

2.4 Tabu List Size

The tabu list size is a crucial parameter of the algorithm, depending both on the problem dimension and the neighborhood definition we established. A small tabu list size can lead to cycling and getting stuck to a local space, while a large one can slow down the algorithm and make it less effective.

In our case, TS evaluates moves and performances based on a the penalty of the soft constraints, moreover it has been adapted to consider the hard constraints as well. As a consequence, the two previous edge cases translates as follows:

- Small tabu list: the algorithm tends to reaccept previous moves and performs a small set of actions, more precisely nurses assignment and unassignment. In other words, TS focus more on penalty optimization and get stuck in a local space where the best action at each step is to assign or unassign a nurse, which slightly improves the penalty or keeps it constant. This is not desirable, since the hard constraints are not guaranteed to be satisfied.
- Large tabu list: the algorithm tends to avoid previous moves and explores a larger space, performing patient scheduling. However, patient unschedule actions remain tabu for a long time preventing TS to free up a room, which means the current nurse assigned to it is stuck and cannot be exchanged with a better nurse in terms of soft constraints. This is not desirable as well, since the penalty is not minimized.

These cases are the extremes of the spectrum, but evidence well how the tabu list size is the key parameter in order to opportunely balance the algorithm and avoid having hard constraints violated, nor them prevailing over the soft ones. AP: [We remark that the correct tabu list size is a crucial problem-specific parameter, i.e. it depends on the number of rooms, patients, nurses, shifts, etc.]

2.5 Aspiration Criterion

In this case, the aspiration criterion is used to accept forbidden moves when they lead to a better solution than the best overall one found so far.

3 Results

The algorithm was tested with some provided instances, whose parameters are shown in Table 1 along with the chose size for the tabu list. In all the instances, the algorithm was able to reach a solution where no hard constraint was violated, with penalty values shown in Table 2.

Instance	Days	Shifts	Occupants	Patients	Operating Theaters	Rooms	Nurses	Tabu List Size
toy	7	3	2	7	1	3	11	30
1	21	3	7	42	2	5	13	150
2	14	3	5	37	4	6	17	200
3	14	3	10	45	2	6	14	200
4	14	3	7	54	3	8	19	250
6	14	3	8	110	3	9	20	300

Table 1: Test instances parameters and tabu list size.

Test Instance	Starting penalty	Tabu Penalty	Proposed Penalty
toy	2100	269	292
1	6350	3976	3177
2	12950	2443	1583
3	15758	11624	10184
4	13501	6524	2332
6	55500	25366	17048

Table 2: Penalty reached by the Tabu Search algorithm on the test instances, compared to the penalties of the starting configuration and the competition proposed solution.

4 Discussion

The proposed approach performances are overall satisfactory, with the algorithm being able to find a solution guaranteeing no hard constraint violation in all the tested instances. The penalty values reached by TS are sometimes competitive with the precomputed solutions, but are generally higher, especially for the larger problems.

Focusing on the first instances, we notice that they achieve great results in term of patient scheduling, with an high percentage of penalty due to non minimization of nurses soft constraints like continuity of care. As mentioned before, this happens when the tabu list size is too large: however, in this cases, tests performed with smaller tabu list size did not lead to better results, but to a higher penalty value or even violation of hard constraints. One possible solution to this issue could be to implement the possibility for nurse substitution, without the need to free up the room and remove the assigned nurse beforehand: this would allow the algorithm to maintain stability and optimize nurse assignment in a more effective way. It is worth mentioning, however, that this would require a more complex neighborhood definition and a more sophisticated tabu list management, since the algorithm would need to consider the possibility of swapping nurses between rooms and shifts. As a matter of fact, a bad management of this feature could lead to the algorithm cycling between nurses interchange, significantly worsening the performances. For this reasons, an implementation of this kind would surely translate in a more complex and slower algorithm.

A Appendix

A.1 Results

Listing 1: Toy

```
Total violations = 0

COSTS (weight X cost):
RoomAgeMix.....5  (  5 X   1)
RoomSkillLevel.....27  (  1 X  27)
ContinuityOfCare.....53  (  1 X  53)
ExcessiveNurseWorkload.....4  (  1 X   4)
OpenOperatingTheater.....100  ( 50 X   2)
SurgeonTransfer.....0  (  5 X   0)
PatientDelay.....80  ( 10 X   8)
ElectiveUnscheduledPatients.....0  (300 X   0)
Total cost = 269
```

Listing 2: Test 01

```
Total violations = 0

COSTS (weight X cost):
RoomAgeMix.....85  (  5 X  17)
RoomSkillLevel.....122  (  1 X 122)
ContinuityOfCare.....1160  (  5 X 232)
ExcessiveNurseWorkload.....183  (  1 X 183)
OpenOperatingTheater.....360  ( 30 X  12)
SurgeonTransfer.....1  (  1 X   1)
PatientDelay.....715  (  5 X 143)
ElectiveUnscheduledPatients.....1350  (150 X   9)
Total cost = 3976
```

Listing 3: Test 02

```
Total violations = 0

COSTS (weight X cost):
RoomAgeMix.....80  (  5 X  16)
RoomSkillLevel.....136  (  1 X 136)
ContinuityOfCare.....223  (  1 X 223)
ExcessiveNurseWorkload.....459  (  1 X 459)
OpenOperatingTheater.....130  ( 10 X  13)
SurgeonTransfer.....0  ( 10 X   0)
PatientDelay.....715  (  5 X 143)
ElectiveUnscheduledPatients.....700  (350 X   2)
Total cost = 2443
```

Listing 4: Test 03

```
Total violations = 0

COSTS (weight X cost):
RoomAgeMix.....10  (  1 X  10)
RoomSkillLevel.....29  (  1 X  29)
ContinuityOfCare.....630  (  5 X 126)
ExcessiveNurseWorkload.....650  ( 10 X  65)
OpenOperatingTheater.....50  ( 10 X   5)
SurgeonTransfer.....0  (  5 X   0)
PatientDelay.....105  ( 15 X   7)
```

```
ElectiveUnscheduledPatients ..... 10150 (350 X 29)
Total cost = 11624
```

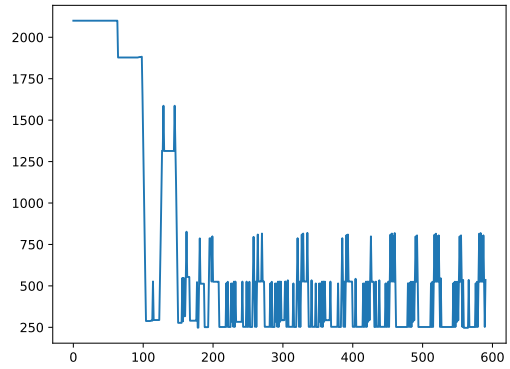
Listing 5: Test 04

```
Total violations = 0

COSTS (weight X cost):
RoomAgeMix ..... 16 ( 1 X 16)
RoomSkillLevel ..... 750 ( 5 X 150)
ContinuityOfCare ..... 326 ( 1 X 326)
ExcessiveNurseWorkload ..... 3465 ( 5 X 693)
OpenOperatingTheater ..... 150 ( 10 X 15)
SurgeonTransfer ..... 7 ( 1 X 7)
PatientDelay ..... 1310 ( 10 X 131)
ElectiveUnscheduledPatients ..... 500 (250 X 2)
Total cost = 6524
```

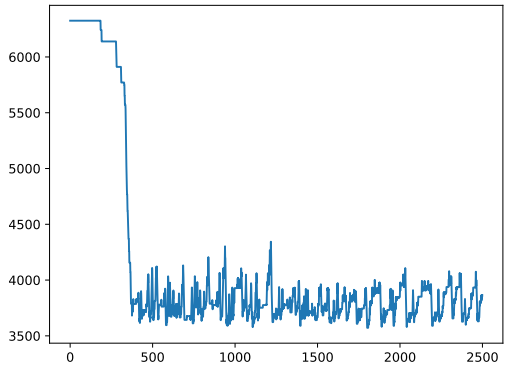
Listing 6: Test 06

```
Total violations = 0

COSTS (weight X cost):
RoomAgeMix ..... 45 ( 5 X 9)
RoomSkillLevel ..... 115 ( 1 X 115)
ContinuityOfCare ..... 2170 ( 5 X 434)
ExcessiveNurseWorkload ..... 896 ( 1 X 896)
OpenOperatingTheater ..... 480 ( 30 X 16)
SurgeonTransfer ..... 50 ( 10 X 5)
PatientDelay ..... 2110 ( 10 X 211)
ElectiveUnscheduledPatients ..... 19500 (500 X 39)
Total cost = 25366
```

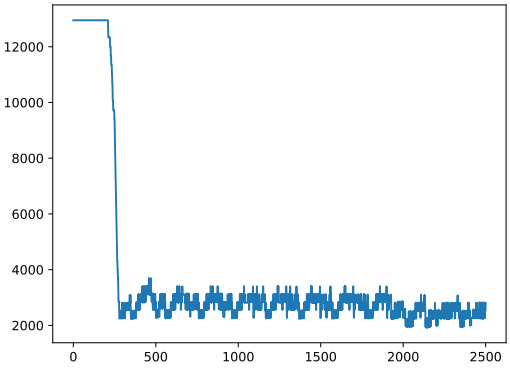
A.2 Plots



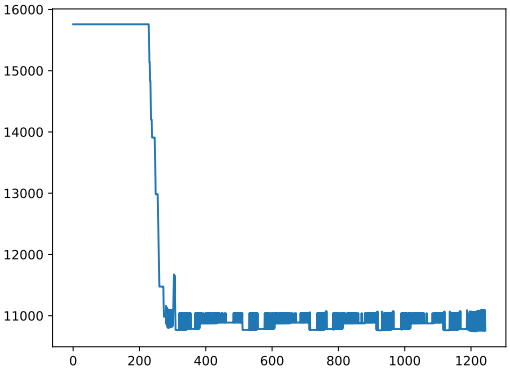
(a) Toy.



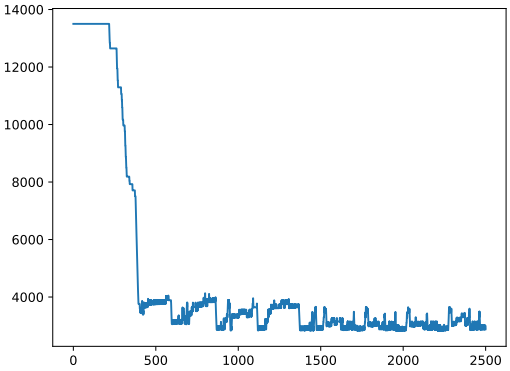
(b) Test 01.



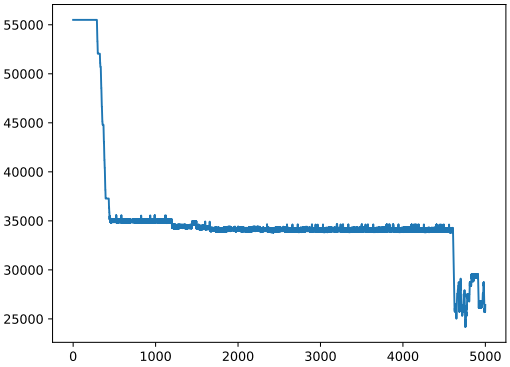
(c) Test 02.



(d) Test 03.



(e) Test 04.



(f) Test 06.

Figure 1: Costs of the solution found by the Tabu Search algorithm on the test instances.

References

[1] URL: `https://ihtc2024.github.io/`.

[2] Fred Glover. "Tabu Search—Part I". In: *ORSA Journal on Computing* 1.190-206 (3 1989). URL: `https://doi.org/10.1287/ijoc.1.3.190`.