

THE FINAL REPORT

Dynamic Patient admission scheduling problem

Done by Xavier Vacarel and Rohitangsu Das

C Contents :

1. Description of the problem.
2. Motivations for solving the problem.
3. Our Goal.
4. The Algorithm
5. Work Already done.
6. Dataset we got from the internet.
7. DataSet we created.
8. New things added
9. To the future
10. References

1. Description of the problem.

The dynamic patient admission scheduling problem is about assigning people arriving to a hospital to the right bed. Indeed, it is a very complex problem to manage the patient in a hospital, they all have different diseases, they need different treatment, some of them need emergency care, others have to stay many days, etc... But the hospital has only limited resources, such as rooms, beds, medical equipment, doctors, nurses. So, this problem is about finding the best way of assigning the best resources to the different patients arriving.

2. Motivations for resolving this problem

The motivations are very clear and important, it's very hard to manage a hospital and all its resources, but it's necessary because a lot of lives are engaged in this process. Indeed, sometimes a bad organization in a hospital can provoke indirectly the death of a lot of people in long term. Moreover, it is a very complex problem that has to be resolved very quickly, and it could be impossible to do it for thousands of patients without using a good algorithm. We can also think about big natural disasters like the 2004 tsunami, where a lot of hospitals were destroyed, and the number of patients with big surgery needs increased by millions.

In that kind of disaster management is the most crucial element to save thousands of lives, and having this powerful algorithm implemented can change everything. The reason why we want to work on this problem for our project is the fact that actual algorithms are not really adapted. Lots of algorithms were invented, but most of them are slow whereas solving time is an important criterion. Furthermore, a big problem is the fact that they take in consideration only the patients admitted the previous day. Nevertheless, emergency cases that have to be solved in few hours are frequent, especially for natural disaster, so that is a big issue. Another problem is the fact that sometimes there is a big imbalance for the affected resources. For example, with some algorithm it could happen that one doctor has to work 90 hours a week, while another one has to work only 10 hours. The last problem is the fact that some algorithm are not considering all the parameters, for example sometime it's impossible for the patient to choose the day when he want to go to the hospital.

3) Our goal

The goal of our project is to compare the different algorithms that have been implemented, and then try to make a new one that can answer to the problems exposed previously. In fact, we will see what are the advantages and weakness of the different algorithms and try to see what is impacting them. Like this, we will try to make our own algorithm, able to deal with the different issues like the performance, the ability of considering the patient arriving right now, or a better repartition of the use of the resources.

4. Work Done

1. . A work was done by Hans et al. ,wherein he proposed an algorithm which not only assigns elective patients to operating systems,optimizes the operating theatre utilization and minimizes the total time taken by an operation(Though perfection is quite vital as it is a question of life and dead.). Local and constructive heuristics are applied to solve theproblem
2. The work was done is scheduling patients who needs to be treated by a psychotherapist.We maintain certain data structures such as list of psychotherapists,list of patients over a weekend.The algorithm proposed by Oguluta ,selects a patient from the list,and schedules him/her to a day of the week, keeping in mind the priority and duration involved for treatment.They are scheduled in such a manner that the workload of the physiotherapists is equally balanced. Patients' preferences concerning the day of the week for treatment are not taken into consideration. Ogulata et al. solve the problem using themathematical programming tools GAMS and MPL.
3. Another algorithm proposed by Marinagi,looks to maximise the examination time of a patient and the maximum utilization of the hospital's resources.The problem is solved by a combination of agents ,a hierarchical planner which supports the decomposition of complex tests into smaller parts and a scheduler. Subject to the different actions that need to be executed, which is the result obtained by the planner, the scheduler tries to assign the actions to appropriate timeslots
4. Harper and Shahani [14] describe a simulation model in which bed occupancy and patients' refusals can be calculated, taking into account different what-if scenarios..

4.The Algorithm.

1. We obtained 13 sets of data from the site "<http://allserv.kahosl.be/~peter/pas/>".
2. Our first step was to produce for a particular input data as many output files as possible that satisfied the hard constraints.
3. We did this using a Validator code (a Java byte code), we got the code from the site "<http://allserv.kahosl.be/~peter/pas/>".
4. If the output is satisfied, then the same code computes the violation on the soft constraints.
5. We collect all the violation costs of these validator into sets of size 8, we apply the Tabu Search Algorithm on these data sets, and compute the most optimal solution.

An Insight into the Tabu Search Algorithm.

1. Take a initial solution.
2. Maintain a Tabu List
3. Look at its neighbors and keep adding solution to set known as candidate list, if the elements of that soln isn't present in the tabu list.
4. Choose optimal solution from the candidate list.
5. Compare this solution from previous solution, if better make it the current best solution.
6. Finally, add the elements of this algorithm from the new solution and add it to the Tabu List.
7. 6. Keep repeating this step until a user defined exit condition is encountered.

5 Hard and Soft Constraint.

There are some constraints we would like to satisfy at any cost so that one could have feasible solution, these are known as Hard Constraints and if this feasible solution satisfies many Soft Constraints, then the method is better.

Examples:

1. No two patients should be allowed the same room.
2. The stay of the patients should be contiguous.
3. Male and female patients should be assigned different rooms.
4. The medical treatment of a patient P_i may require that he/she is assigned to a room R_j with special equipment. These room properties are mandatory for the treatment.

Examples of soft constraints.

1. The age policy should be satisfied.
2. The room patients should be taken into work.
3. Unnecessary transfers should not be made.
4. Gender policy should not be violated.

6. Dataset(We got from the internet)

ARTICLE BENCHMARK DATA SET

Rooms: 150

Roomproperties: 2

Beds: 447

Departments: 6

Specialisms: 6

Patients: 660

Planning horizon: 14

SPECIALISMS:

1 Specialism1

2 Specialism2

3 Specialism3

4 Specialism4

5 Specialism5

6 Specialism6

DEPARTMENTS:

1 Department1 0 0 | 1 1 2 2 2 3

2 Department2 0 0 | 1 2 2 3 2 4

3 Department3 0 0 | 1 3 2 4 2 5

4 Department4 0 0 | 1 4 2 5 2 6

5 Department5 0 0 | 1 5 2 6 2 1

6 Department6 0 0 | 1 6 2 1 2 2

ROOMPROPERTIES:

1 telemetry

2 oxygen

ROOMS:

1 11 | 1 | 1 | D | 1 1 1 2 3 3 | 1 1

2 12 | 1 | 1 | D | 1 1 1 2 1 3 | 1 1

3 13 | 1 | 1 | D | 1 1 1 2 1 3 | 0 1

4 14 | 2 | 1 | D | 1 1 2 2 1 3 | 0 0

5 15 | 2 | 1 | D | 1 1 1 2 1 3 | 0 1

6 16 | 2 | 1 | D | 1 1 1 2 1 3 | 0 1

BEDS:

1 1

2 2

3 3

4 4

5 4

6 5

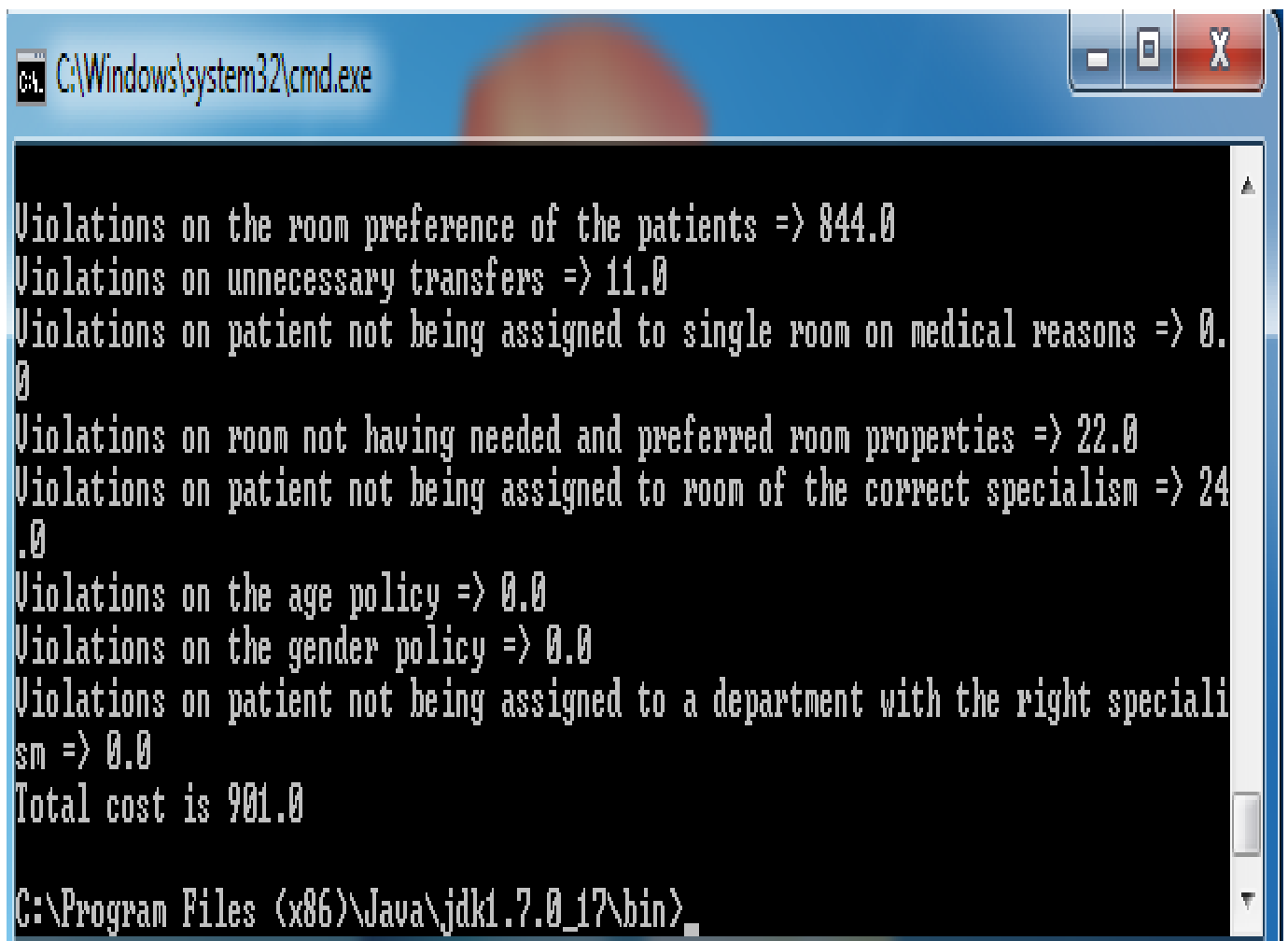
7.Dataset(We created)

1 397 397
2 369 369 369 369 369 369 369 369 369
3 221 221 221 221 221 221
4 155 155 155 155 155 155
5 265 265 265 265 265 265
6 168 168 168 168 168 168
7 12 12 12 12 12 12
8 210 210 210 210 210 210
9 1
10 142 142 142 142 142 142
11 239 239 239 239 239 239 239 239 239 239 239 239 239
12 2 2 2 2 2
13 324 324 324 324 324 324 324 324 324 324
14 274
15 351 351 351 351 351
16 76 76 76 76 76
17 58 58 58 58 58
18 174 174 174 174 174
19 246
20 321 321
21 88
22 244
23 164 164 164 164 164
24 367 367 367 367 367 367 367 367 367 367 367 367 367
25 42 42 42 42 42 42
26 111 111 111 111 111 111
27 223 223 223 223 223 223
28 220 220
29 394 394 394 394
30 294 294
31 330 330 330 330 330 330 330 330 330 330 330 330 330
32 14 14 14 14 14
33 237 237 237 237 237 237
34 343 343
35 429 429 429 429 429
36 415 415 415 415 415
37 179 179 179 179 179
38 178 178 178 178 178 178
39 169 169 169 169 169
40 214 214
41 283 283 283 283 283 283 283 283 283 283 283 283 283
42 331 331 331 331 331 331 331 331 331 331 331 331 331
43 249 249 249 249 249 249 249 249
44 442 442 442 442 442
45 301 301
46 380 380 380 380 380

3)The Results we got

So , after running the Validator code on the input and output file, we get the violations for the soft constraints if and only if the hard constraints are satisfied in the first place.

So for the above input and output files we got the following outputs:



```
C:\Windows\system32\cmd.exe
Violations on the room preference of the patients => 844.0
Violations on unnecessary transfers => 11.0
Violations on patient not being assigned to single room on medical reasons => 0.0
Violations on room not having needed and preferred room properties => 22.0
Violations on patient not being assigned to room of the correct specialism => 24.0
Violations on the age policy => 0.0
Violations on the gender policy => 0.0
Violations on patient not being assigned to a department with the right specialism => 0.0
Total cost is 901.0
C:\Program Files (x86)\Java\jdk1.7.0_17\bin>
```

3)Our implementation

1. Create a program that uses the Tabu Search Algorithm that optimizes the list of Costs.
2. Find an efficient way to write codes that would automatically produce input and output results.This would be to make our job easier.

So,the Tabu search code is written in the matlab,when give this the inputs :-

```
2 841 11 0 3 22 0 0 0
3 845 11 0 22 24 0 5 0
4 845 11 0 22 28 0 0 0
5 845 11 0 22 24 0 5 0
6 846 11 0 22 26 0 5 0
7 845 11 0 12 26 0 0 0
8 844 11 0 14 26 0 0 0
9 844 11 0 21 27 0 5 0
0 844 11 0 16 24 0 5 0
1 832 11 5 4 48 2 3 2
2 820 11 5 32 24 0 5 0
3 829 11 5 22 12 0 5 0
4 829 11 5 22 12 0 5 0
5 834 11 0 22 13 0 5 0
6 846 11 0 13 14 0 5 0
7 897 11 0 22 15 0 5 0
8 823 11 0 34 16 0 5 0
9 834 11 0 22 17 0 5 0
0 723 11 0 22 14 0 5 0
1 822 11 0 22 15 0 5 0
2 822 11 0 22 16 0 5 0
```

The size of the data set was small,70 was the cardinality ,The code gave the 55th as the best solution.The output was actually 723 11 0 21 14 5 5 0.(It represents costs to all the violations.

8)The new things we added

We created a php script , that when given as arguments the number of patients ,number of departments ,number of rooms ,the code returns as output a new input file.

Now our main plan was to somehow reduce the overhead of producing too many output files for the Validator code by hand, Xavier worked on it and these is what we have made.

9)In the Future.

These is what we could look to do in the future. There are a lot of algorithms to address this very important problem. For example CF-SA-16-ver2, ConstraintSolver and really a new thing called the Drools Solver. We could compare them .

execution time	instance	CF-SA-16-ver2			ConstraintSolverSA			DroolsSolver		
		min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
short	testdata1	734,00	764,08	15,21	765,40	842,10	43,00	1014,00	1114,82	92,21
	testdata2	1308,20	1378,32	44,48	1312,00	1376,12	39,15	1774,20	1841,56	107,00
	testdata3	898,00	925,04	19,58	902,40	920,14	13,30	1156,20	1238,30	118,69
	testdata4	1432,80	1461,34	22,34	1465,60	1513,12	26,24	1881,00	2001,70	145,91
	testdata5	646,40	656,00	6,21	652,00	661,20	6,42	698,40	718,00	17,61
	testdata6	889,60	914,88	18,28	924,60	953,46	15,87	1094,60	1165,34	69,23
	testdata7	1451,00	1484,56	21,70	1776,80	1840,48	37,98	2336,00	2424,88	107,71
	testdata8	5045,40	5151,28	75,38	5691,40	5870,88	121,90	7519,20	7824,96	348,22
	testdata9	24602,00	25017,98	232,50	55625,40	56922,54	755,82	39762,20	39887,62	160,83
	testdata10	11546,40	11676,84	116,25	12349,00	12909,38	261,44	14918,80	15458,68	636,76
	testdata11	21867,40	22060,46	147,25	20387,60	20968,42	385,89	24334,60	25876,64	1835,32
	testdata12	36779,40	37230,24	302,19	39136,80	39620,62	296,06	44924,60	47258,50	2517,62
	testdata13	11646,80	11782,30	110,74	13596,20	14187,38	362,11	16177,20	16593,42	413,59
execution time	instance	CF-SA-16-ver2			ConstraintSolverSA			DroolsSolver		
		min.	avg.	st. dev.	min.	avg.	st. dev.	min.	avg.	st. dev.
long	testdata1	671,20	696,08	19,23	712,00	752,60	18,43	885,00	910,08	26,78
	testdata2	1210,40	1229,90	12,87	1236,40	1257,16	15,51	1459,60	1512,64	42,20
	testdata3	827,00	848,78	12,43	832,60	851,12	19,68	1046,00	1074,92	18,33
	testdata4	1283,00	1335,22	25,83	1358,40	1410,28	32,66	1727,00	1743,64	17,62
	testdata5	638,40	641,44	2,94	644,00	647,68	3,07	681,60	689,84	5,86
	testdata6	828,80	858,04	22,49	870,80	884,26	12,06	995,80	1015,02	25,47
	testdata7	1331,20	1365,64	27,42	1597,40	1712,60	60,80	2078,60	2162,86	63,96
	testdata8	4682,00	4755,18	51,52	5486,20	5641,78	82,98	7098,60	7217,72	72,64
	testdata9	22221,80	22405,84	106,89	46252,20	48180,10	1092,72	39381,40	39554,82	106,94
	testdata10	9806,60	9908,98	68,95	12033,00	12332,98	213,66	13558,60	13885,12	282,00
	testdata11	16025,60	16408,30	221,82	19770,60	20007,34	250,38	22145,40	22577,04	440,56
	testdata12	28553,40	28808,54	204,68	37783,80	38352,92	404,71	42275,40	42949,50	513,10
	testdata13	10277,60	10404,00	96,14	13383,00	13654,48	190,03	15256,80	15437,74	233,00

Table 1: Overall comparison of solution methods

(The picture is taken from "<http://allserv.kahosl.be/~peter/pas/>")

10)References.

- <http://www.sciencedirect.com/science/article/pii/S0933365712001169>
- <http://satt.diegm.uniud.it/uploads/Papers/CeSc11.pdf>
- <http://satt.diegm.uniud.it/uploads/Papers/CeSc11.pdf>
- <http://allserv.kahosl.be/~peter/pas/>