Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

Maxat Zhuniskhanov

Analysis of Course Schedule Generation Tool and Its Integration to University Automation System

Presented in Partial Fulfillment for the
Degree of Master of Science in Computing Systems and Software
(degree code: 6M070400)
Department of Computer Sciences
Faculty of Engineering and Natural Sciences

Supervisor: **Rasim Suliyev**

Kaskelen, 2020

Abstract

This research examines the university course schedule generation problem. It begins with the definition of the inconveniences of the old system. Then determine the importance of reducing the resource usage while integrating with the university automation system. This discussion integrates with an overview of the problem itself both from practical and academic perspectives. Then course schedule generation algorithms discussed with the ways of implementing it. This is followed by benchmark and resource usage comparison.

Аңдатпа

Бұл зерттеу университеттік курс кестесін генерациялау проблемасын қарайды. Ескі жүйенің қолайсыздықтарын анықтаудан басталады.Содан кейін университеттің автоматтандыру жүйесімен интеграция кезінде ресурстарды пайдалануды қысқарту маңыздылығын айқындайды. Бұл талқылау ең өзекті мәселелерді шолуды практикалық жағынан, сондай-ақ академиялық көзқарас жағынан ықпалдасады. Содан кейін курс кестесінің генерациялау алгоритмдер және оны жүзеге асыру әдістері талқыланады.Содан кейін және ресурстарды пайдалану салыстырылады.

Аннотация

Это исследование рассматривает проблему генерации расписания университетских курсов. Начинается с определения неудобств старой системы. Затем определяет важность сокращения использования ресурсов при интеграции с системой автоматизации университета.Это обсуждение интегрируется с обзором самой проблемы как с практической, так и с академической точек зрения. Затем обсуждаются алгоритмы генерации расписания курсов и способы его реализации. Затем следует сравнение и использование ресурсов.

Acknowledgements

Thanks to my thesis supervisor for constant support and useful discussion. Thanks to the external reviewer for very useful feedback that helped to significantly improve the current work.

Contents

1. Chapter 1

1.1 Introduction

In Modern highly developed technology world, technology is developing rapidly. More and more new, fast and advanced technologies appear, old solutions do become not actual. The solutions for generating course schedule, it also requires revision. Especially if the system ceases to meet the requirements, it is necessary to consider new technologies, algorithms and experiences in this area.

This work examines university course scheduling problem. Course schedule generation is an allocation of resources; therefore, in many universities responsibility for constructing the course schedule generation is distributed among the academic units. In the most general terms, scheduling can be described as the constrained allocation, of resources to objects, being placed in space-time in such a way as to minimize the total cost of a set of the resources used [9]. Therefore, in the first it is required to determine the optimal algorithm for achieving a result close to the ideal. Secondly, to study the experience of implementing integration between the university's automated system and the system for generating schedules, without unnecessary intermediate systems or additional user actions. Also using several tests to check the results and through the user survey determine their level of satisfaction.

This chapter begins by defining the purpose and reason for writing this work. It describes the importance and difficulty of correctly generating course schedules. Description of solutions is also added in general terms.

2. Chapter 2

2.1 Aim and objectives of research

Reason for research is to define suitable tool for course schedule generation and minimize resource usage. The results of this study determine the importance of generating the right most appropriate course schedule, for the functioning of the university and the satisfaction of participants. Including the answer to the question, which tool for scheduling courses is suitable for integration with the university automation system. Large-scale courses schedule such as university courses may need many days of work spent by qualified personal in order to produce high quality courses schedule with near to optimal constraint satisfaction and optimization of courses schedule's objectives at the same time. For optimization program used combination of algorithms like Tabu Search, Simulated Annealing and Late Acceptance. Started with a quick configuration that involves little or no configuration and optimization code. Used the Bench marker to improve the values of those size parameters. Combining multiple algorithms together, the execution order would be like this:

- First Fit Decreasing
- Hill Climbing
- Late Acceptance
- Tabu Search

3. Chapter 3

3.1 Background of Literature Review

Scheduling has been studied for many years by different scientists and different algorithms have been invented for their implementation. Timetabling a course schedule is one of the special cases of scheduling.

The Timetable is a list that is given mainly by table view, which reflects a series of organized events, in particular the planned place and time of these events. Since the time when educational institutions have become an organized environment, all school activities are established by a timetable. Therefore, they devoted time, energy and human capital to satisfying almost all the necessary restrictions of the participants of the event [8].

The Course generation issue is a regular planning the task that takes place regularly at least once a year at almost in each educational organization. The issue formed from the task of the planning of classes, understudies, rooms and educators at a settled number of time-slots, subject to a limited number of imperatives. A powerful timetable is vital for the fulfillment of instructive prerequisites and the effective use of human and space assets, which make it an improvement issue. Generally, the issue is comprehended physically by trial and hit technique, where a valid solution isn't ensured. Regardless of whether a valid solution is discovered, it is probably going to miss far better solutions. These vulnerabilities have spurred for the logical investigation of the issue, and to build up a mechanized solution method for it. The issue is being contemplated for last over four decades, however, a general solution technique system for it is yet to be figured [4].

The timetabling issue is one of the most difficult issue areas effectively demonstrated to NP (Non deterministic polynomial)-complete and it is deserving of note that as instructive foundations are tested to develop in number and intricacy, their

assets and occasions are getting to be plainly harder to plan [2].

All timetabling issues are well known, but it has various ways of implementation. It is a NP hard problem, which is exceptionally hard to explain by customary techniques and the measure of calculation required to discover ideal solution increments exponentially with problem size.

[1] Gave a review of the diverse strategies utilized as part of the timetable generation. Requirement fulfillment methods were worried as an essential expansion to the instruments that are utilized as a part of solving the timetabling problem.

Slowly theoretical framework called Constraint Satisfaction Problems (CSP) has appeared [7].

As far as the CSP, a timetabling issue can be figured by representing to a timeslot and a room of a lesson as factors of the CSP, accessible timeslots and rooms as estimations of the CSP, though limitations are the different connections between lessons. Consequently, the CSP demonstrate for timetabling issues can be planned by choosing factors, qualities and imperatives.

Constraints divided into two types, hard and soft and they have their own weights. Soft constraints can be neglected if it prevents hard constraint satisfaction.

4. Chapter 4

4.1 Methods and Materials

Case studies and lab experiments will be the main research strategy in this paper. All secondary data will be taken from existing researches on course schedule generation tools, from architecture till usage directions.

There are so many tools available for course schedule generation however, there are some limitations of course schedule generation tools and techniques. Some limitations will create a large impact on data analysis. The limitations identified to analyze, pre-process, process and execute course schedule generation are basically shortcoming associated with the processing and mining techniques. Some shortcoming and limitations can be summarized as follows:

- The present course schedule generation tools have input restrictions.
- The available tools have no user interface or interface is not user friendly.
- Other tools outputs need to be converted to required data.
- There are many tools those are based on virtual machines which work on another machine or operating systems.

The analytical and processing techniques can be improved by implementing user friendly interface and preparation of data in the required formats.

Before starting practical-experimental part of my research, the first step will be qualitatively reading all existing papers (around 10-15) on course schedule generation tools. This will help to understand the current trend, also adopt technology and techniques for using these tools. It is also important to study the difficulties and ways to solve them proposed by researchers. Another type of study is a survey on the shortcomings and advantages of the previous system.

This avoids the repetition of shortcomings, and ignoring the advantages of the new system.

Afterwards, quantitative implementation with test data sets will be implemented on course schedule generation tools. These results will be collected and analyzed. Data sets will be obtained from university automated systems and will be converted to the required format. To test different constraints, various examples will be devised. After each successful step, new restrictions will be added. And so on until all the restrictions are added. Due to the fact that it is not always possible to comply with all the rules, the most unimportant ones may not be fulfilled.

5. Chapter 5

5.1 Optimization Algorithms

Algorithms used for this research are commonly known, but the Huge amount of optimization algorithms exist currently. Due to the inability to use all of these optimization algorithms, several of them have been chosen in accordance with the criteria of speed, accuracy and compatibility. This section describes their characteristics and methods of their use. We will describe the algorithms used in the order of use to improve the result. For these purposes, the generation of the order of the courses is conditionally divided into two stages, the first using data filling algorithms, and the second algorithms to improve the result in order to meet the maximum requirements.

5.1.1 First Fit Decreasing

First Fit Decreasing is a construction heuristic algorithm, which is used to effectively fill empty places. This algorithm works in this manner: Items firstly sorted in decreasing order of their size. Then items placed in container into which it fits. This will help to put down complex elements in the first place, because it is highly likely that they will be difficult to distribute in the remaining places. This algorithm requires the implementation of a comparison of the complexity of the elements.

5.1.2 Hill Climbing

Hill Climbing is a simple local search which tries to find moves with highest score. By trying all selected moves and taking best move. And from this action, he further cycles through all the options to find another best action. When finding

```
sort elements in decreasing order using sort
for objects i to n do
    for placeholders j to m do
        if  object i fits in placeholder j then
            put object i in placeholder j.
            break the loop and put the next object.
        end if
    end for
    if object i did not fit in any available placeholder then
        create new placeholder and put object i.
    end if
end for
```

Figure 5.1: First Fit Decreasing pseudo code

several best actions, he randomly selects one of them as the best. The dignity of choosing the best action may seem like a good solution, but the choice can lead to a local maximum. This happens when further actions can worsen the result. If even one of them is selected, then the search trace may again lead to the previously selected result.

```
current = start
Loop do
    list = neighbours(current)
    nextEval = -infinite;
    nest = nill;
    for all a in list
        if eval(a) > nextEval
            next = a;
            nextEval = EVAL(a);
    if nextEval <= EVAL(current)
        return current;  // Returning current, cause no best found
    current = next;
```

Figure 5.2: Hill Climbing pseudo code

5.1.3 Late Acceptance

This algorithm, also known as Late Acceptance Hill Climbing, uses several movements in one step. Late Acceptance searches for movements that do not reduce the point or leads to a result that is a late result which is the winning result a few steps back. By Edmund K. Burke and Yuri Bykov stated this algorithm at each iteration, a current solution is used to determine the acceptance of a new candidate. In other words, a candidate solution is compared with a current one and accepted when its cost function is not worse. Our idea is to delay this com-

parison, namely: to compare the candidate solution with a solution, which was "current" several steps before. Here, each current solution still takes on the role of an acceptance benchmark, but it will be used at later steps [5].

```
Produce an initial solution s
    Calculate initial cost function C(s)
    Specify Lfa
    For all k{0...Lfa-1} fk:=C(s)
    First iteration I=0;
    Do until a chosen stopping condition
        Construct a candidate solution s*
        Calculate its cost function C(s*)
        v := I mod Lfa If C(s*)fv or C(s*)C(s)
        Then accept the candidate (s:=s*)
        Else reject the candidate (s:=s)
        Insert the current cost into the fitness array fv:=C(s)
        Increment the iteration number I:=I+1
```

Figure 5.3: Late Acceptance pseudo code

5.1.4 Tabu Search

As described Fred Glover, Tabu search is a strategy for solving combinatorial optimization problems whose applications range from graph theory and matroid settings to general pure and mixed integer programming problem [6]. Tabu Search keeps a list so as not to get stuck in a local maximum. This sheet contains a list of recently used objects which cannot be used at the moment. All actions that interact with the object in the list are not accepted. Objects can mean anything about an action, the action itself, a decision, etc.

```
sBest ← s0
    bestCandidate ← s0
    tabuList ← []
    tabuList.push(s0)
    while (not stoppingCondition())
        sNeighborhood ← getNeighbors(bestCandidate)
        bestCandidate ← sNeighborhood[0]
        for (sCandidate in sNeighborhood)
            if ( (not tabuList.contains(sCandidate)) and (fitness(sCandidate) > fitness(bestCandidate)) )
                bestCandidate ← sCandidate
            end
        end
        if (fitness(bestCandidate) > fitness(sBest))
            sBest ← bestCandidate
        end
        tabuList.push(bestCandidate)
        if (tabuList.size > maxTabuSize)
            tabuList.removeFirst()
        end
   end
return sBest
```

Figure 5.4: Tabu Search pseudo code

6. Chapter 6

6.1 Analysing old system

One of the reasons for writing this work is the obsolescence and inconvenience of working with the old system. The input data in it is accepted in only one format, account for administrative workers tarting to rework in the desired format. Accordingly, the output is provided in one format. There is also no possibility of adding a restriction in any view. Inside it, only the work written inside by the algorithm is imprisoned.

To get rid of such shortcomings, also give flexibility in use, we will give a new architecture below.

6.2 New architecture

This chapter suggests creating and developing a new information system architecture for use and managing the course schedule at the university. The proposed work tries to integrate the proposed combined algorithm with the real task of administration and management of the formation of the course schedule. As previously written, there were many works where they tried to create a schedule

Functions	Old System	New System
Scheduling	Yes	Yes
Dynamic rules	No	Yes
Dynamic stop condition	No	Yes
Directly loads input data	No	Yes
Dynamic input data format	No	Yes
Directly loads output data	No	Yes

Table 6.1: Supported functions of Old and New System

formation system, but they were suitable mainly for a specific education center. Basically, we will consider the algorithm that will be used for our purpose, as well as focus our attention on the data that will be used for the administration of course scheduling at our university.

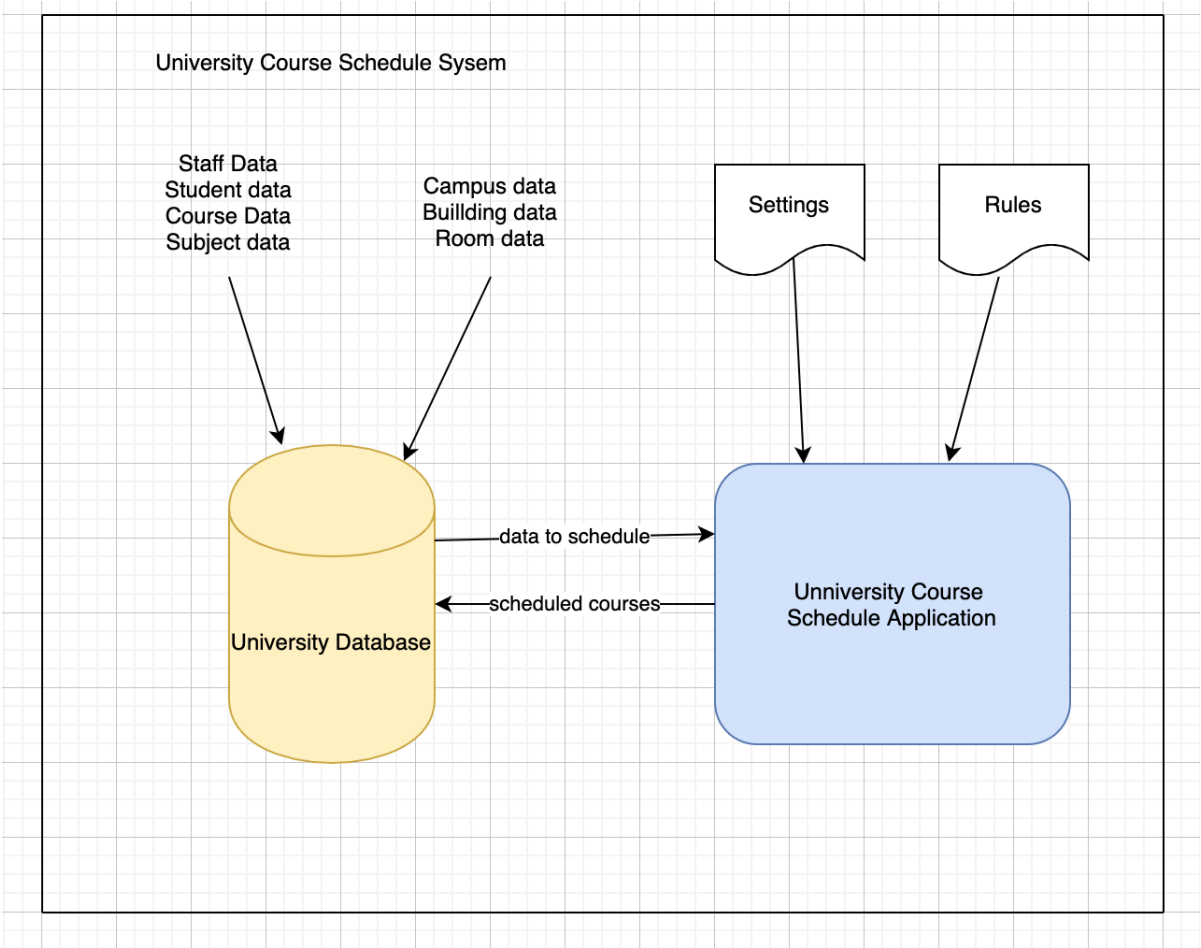

Figure 6.1: Holistic Architecture of University Course Schedule

The figure above shows an overview of the proposed information system. It shows the input and output data of the system. Input information is the foundation of the program, they include data on subjects, teachers, students, buildings and rooms. Their set form a course and therefore are very important for generating a course schedule. Other input to the program is the program settings and rules for creating a course schedule. Setting up the program determines how the program should work, under what conditions it should stop, and which algorithms to use. And the rules determine what restrictions the program has in formulating a course schedule.

6.2.1 System and Design

The choice of program for generating the course schedule was determined from the previously reviewed literature. Recently, almost all NP problems have been resolved through the definitions of hard and soft constraint reasoning. For these purposes, we have come up with the open source Optaplanner. development system. It was also made on the basis of the modern programming language java, just the ease of settings and the availability of autonomous storage of the generation rules identified it as an ideal candidate for our work. Java modern programming language has a number of frameworks that have libraries built-in for connecting to any database for reading and writing data, and can be developed any kind of rests for integration with web interfaces. These web interfaces will interact with end users who will manage the course scheduling program.

After we will discuss the development of a program that will be used to generate the course schedule. We begin the development of the program by disassembling the data storage form and modifying it to generate and use the course schedule.

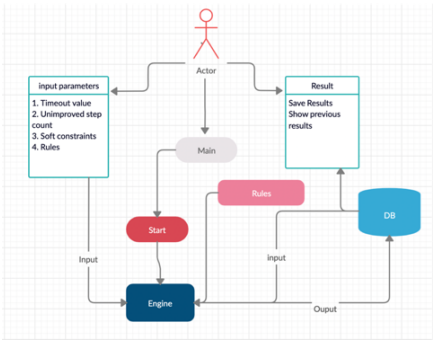

Figure 6.2: Use-case diagram of Course Schedule Generation

7. Chapter 7

7.1 Dataset

During the analysis of the database provided by the university for the course schedule system, the main entities were identified. These entities were presented as key entities. The list of entities looks like this

- Subjects
- Teachers
- Cabinets
- Students

The tables below will give a visual description of each entity and the relationship between other objects

7.2 Database Schema

During system design, the logical data model will be transformed into a physical data model from a chosen database management system called a Database

Entity	Description	Related Entities	Description
Departments	It stores all departments of university	Rooms	All available rooms within this department
		Subjects	All available subjects

Table 7.1: Data Structure – Departments Table

Entity	Description	Related Entities	Description
Rooms	It stores all available rooms in each Department, has attributes seat size, type	Course	Course which can be held in this room

Table 7.2: Data Structure – Rooms Table

Entity	Description	Related Entities	Description
Subjects	It stores data for all the subjects offered by a department. Contains attributes type(theory, practice, laboratory) and hours for each type	Teachers	It stores information about staff who conducts this subject
		Students	It stores information about students who preferred this subject

Table 7.3: Data Structure – Subjects Table

Entity	Description	Related Entities	Description
Staff	It stores information about staff who affiliated to teach subjects	Subjects	It stores data for all the subjects related to staff

Table 7.4: Data Structure – Staff Table

Entity	Description	Related Entities	Description
Course	It stores information about course which is planning entity	Subjects	It stores data for all the subjects related to course
		Subjects	It stores data for all the subjects related to course
		Room	It stores data for all the rooms related to course
		Timeslot	It stores data for all the day and time related to course

Table 7.5: Data Structure – Staff Table

Schema. This model will reflect the technical capabilities and limitations of that database technology, as well as the performance tuning requirements suggested by the database administrator. Database Schema is the physical model or blueprint for a database. It represents the technical implementation of the logical data model. Database Schema specifies details based on the capabilities, terminology and constraints of the chosen database management system. The database schema is a little different compared to the ER Diagram. As the ER Diagrams has the basic entity and does not provide any further details with respect to the various attributes and the key fields within the table or entity. The table below mentions all of the major tables and their related tables with the common identifier column, which is a primary key in the major table and a foreign key in the related table. Using the normalisation method derives these tables and the respective relationships are achieved at the third-normal form level.

The main entity in Figure above is Course, which has relation to entities Lecturer, Rooms, Students, Subjects and Timeslots.

To test and compare the data provided by University. It contains detailed information for usage in course scheduling: 1. teaching staff consisting of 441 unique educators 2. There are also 115 audiences which are divided into lecture halls, laboratories, and also regular cabinets. This auditory has information about the capacity. 3.2376 subjects in which the number of lectures, practices and labo-

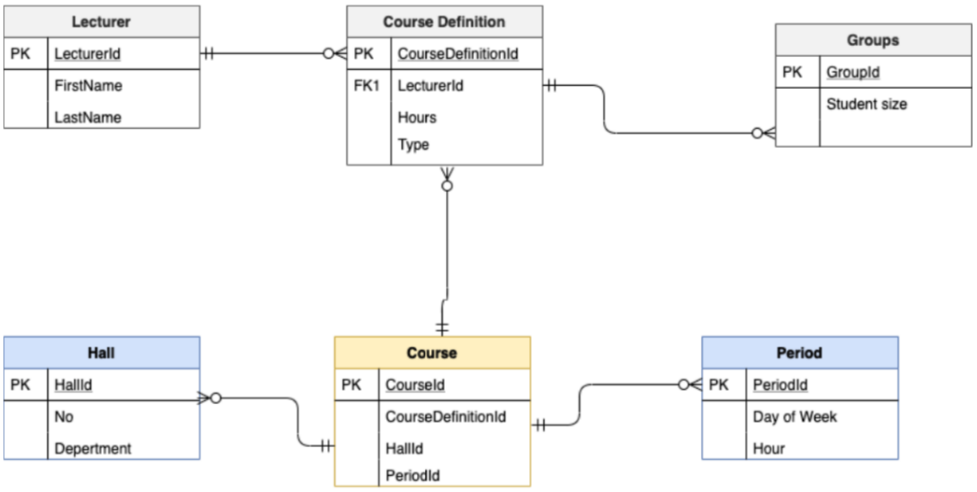

Figure 7.1: Course table showing relation entities

ratory works, and also their total number are prescribed. 4. 2821 sections where information about the relationship of the subject and the teacher is registered.

7.3 Rules

For the program to produce the desired result, you need some kind of rule by which it must form the result. For this, our system uses Drools - rule engine. Drools is a Business Rules Management System (BRMS), it gives autonomy when writing and making decisions as part of our system. It has the ability to define two types of constraint, hard and soft constraints. In our program to get the results, we needed to determine the types of restrictions: Hard constraints are constraints that cannot be physically fulfilled therefore it is necessary to prevent their occurrence. An example of writing hard constraints in our program. picture Some examples of hard constraints are given below:

- In the same room can not be two different lectures: Two different lessons with different teachers and with different classes should be held in different classrooms.

- One educator cannot teach two different course at the same time: A physical teacher cannot conduct two different lessons at the same time, so we do this option in hard constraints.

```
11
12  rule "roomOccupancy"
13      when
14          Lesson($leftId : id, date != null, $weekDay : date.weekDay, $hour : date.hour, room != null, $room : room)
15          not Lesson(date.weekDay == $weekDay, date.hour == $hour, room == $room, id < $leftId)
16          Lesson(date.weekDay == $weekDay, date.hour == $hour, room == $room, id > $leftId, $rightId : id)
17      then
18          scoreHolder.addHardConstraintMatch(kcontext, -1);
19  end
20
21  rule "lessonsWithSameTeacherInSameTime"
22      when
23          Lesson($id: id, $roomId : room!.id, $teacherId: course!.teacher!.id)
24          Lesson(id > $id, room!=null, room.id == $roomId, course.teacher.id == $teacherId)
25      then
26          scoreHolder.addHardConstraintMatch(kcontext, -2);
27  end
28
29  rule "conflictingLessonsDifferentCourseInSamePeriod"
30      when
31          $courseConflict : CourseConflict($leftCourse : leftCourse, $rightCourse : rightCourse)
32          $leftLesson : Lesson(course == $leftCourse, $weekDay: date!.weekDay, $hour: date!.hour, date != null)
33          $rightLesson : Lesson(course == $rightCourse, date!.weekDay == $weekDay, date!.hour == $hour, this != $leftLesson)
34      then
35          scoreHolder.addHardConstraintMatch(kcontext, - $courseConflict.getConflictCount());
36  end
```

Figure 7.2: Hard Constraints Drools example

- Same student should not have overlapped lessons: The student should be able to participate in all classes of his chosen course.

In weak constraints, rules are usually written that make the distribution of courses convenient for its participants, but they should not break hard constraints.

- Lessons must be distributed evenly throughout the week: If possible every working day of week should have evenly distributed size of lectures on a particular room
- The educator may not want to teach at some time: A teacher may have other activity which is not related to educational process.
- The lecture and practice of one course should be held in same day: To practice learned material from lecture in shortest time

7.4 Configuration

```
// RoomStability: All lessons of a course should be given in the same room.
// Each distinct room used for the lessons of a course, but the first, counts as 1 point of penalty.
rule "roomStability"
    when
        $course : Course()
        accumulate(
            $room : Room()
            and exists Lesson(course == $course, room == $room);
            $roomCount : count($room);
            $roomCount > 1
        )
    then
        scoreHolder.addSoftConstraintMatch(kcontext, -($roomCount.intValue() - 1));
end

// Lessons belonging to a room should be adjacent to each other (i.e., in consecutive hours).
rule "roomLessonCompactness"
    when
        $room : Room()
        Lesson($weekDay : date!.weekDay, $hour : date!.hour, date != null, room == $room)
        not Lesson(date!=null, date.weekDay == $weekDay, date!.hour == ($hour - 1))
        not Lesson(date!=null, date.weekDay == $weekDay, date!.hour == ($hour + 1))
    then
        scoreHolder.addSoftConstraintMatch(kcontext, -2);
end
```

Figure 7.3: Soft Constraints Drools example

8. Chapter 8

8.1 Combined algorithm

The single algorithms good enough to be used in course schedule generation, but there is always a challenge to improve their results. In this work we choose combining multiple algorithms. the main idea is to use algorithms where they are strong. Hill Climbing is good at first steps, but at end can get stuck at Local Optima. Tabu Search good at exiting Local Optima. Last Acceptance added to for wide searching. Later this configuration tuned to perform better, and it was compared with other algorithms using build in Bench marker.

8.2 Configuration

Configuration in Course Schedule system use configuration of Optaplanner library. It uses XML format configuration.

It consists from blocks,

- Termination
- Construction Heuristics
- Local Search

8.2.1 Termination

Termination block uses to configure under which conditions the system should stop generating course scheduling. In tag *secondsSpentlimit, minutesSpentlimit* and *hoursSpentLimit* we can define timer setting to configure exacly time to stop. It's helpful when we want benchmark the result of different algorithms. Next

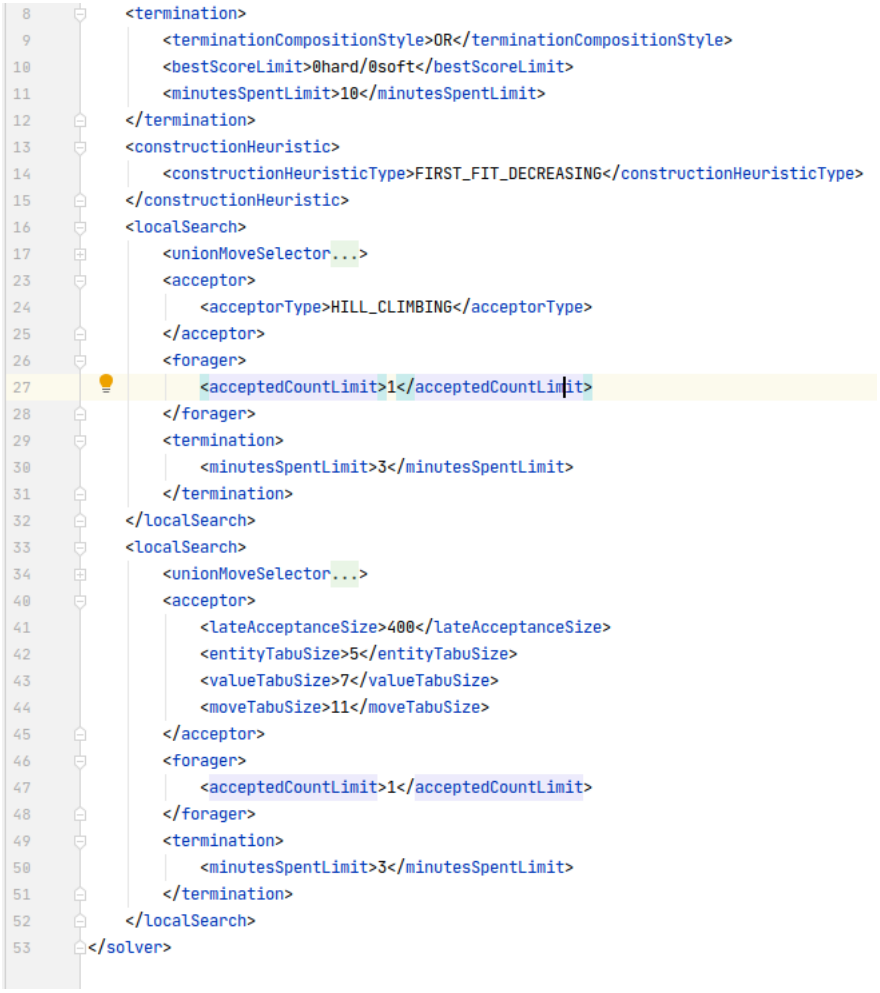

Figure 8.1: Course Schedule Generation System Configuration

option is tag *bestScoreLimit* where you can set value as *0hard/0soft*, to indicate a lower space for the value upon reaching which you can stop the system. There are also options tag *stepCountLimit* which we indicate if you want to stop when a certain number of steps are reached, and *unimprovedStepCountLimit* to indicate the stopping condition the number of steps at which the result does not improve. You can combine several conditions through the *terminationCoompositionStyle* tag. The condition can be combined throught *AND*(и) или через *OR*(или).

8.2.2 Construction Heuristics

Construction Heuristics used to fill Empty courses with random values planning values to make main algorithm to start work easier. Their result isn't feasible, but it finds fast so optimization algorithm can finish it the job. There are a multiple set of construction heuristic algorithms which can be used here. They are

- FIRST FIT

- FIRST FIT. DECREASING
- WEAKEST FIT
- WEAKEST FIT DECREASING
- STRONGEST FIT
- STRONGEST FIT DECREASING

8.2.3 Local Search

In Local Search main metaheuristic algorithm defined. Configuration may have multiple Local Search tags to combine multiple Algorithms. Local Search has three elements: moveSelector, acceptor and forager. More than *moveSelectors* can be defined: we use only *changeMoveSelector* and *swapMoveSelector*. They also can be united by tag *unionMoveSelector*. Accetor checks if move is accepted or not. Forager gathers all accepted moves and selects move which is next step. Usually it select with highest score. The usage of all these elements shown below:

8.3 Benchmark

8.3.1 External dataset

To check how algorithms perform the build in Bench marker used. It's configuration same as Configuration section, except it also has option to configure benchmark types and in one file can be placed multiple configuration of algorithms. Our goal is to compare algorithms on several data set in a series of time limits.

Algorithms to benchmark:

- Hill Climbing
- Tabu Search
- Late acceptance
- Combined

Datasets	Course size
Problem 0	400
Problem 1	600
Problem 2	1000

Table 8.1: Dataset course size

The dataset taken from [3]

The first step is to compare the launch of benchmarker on three different number of courses per 10 minutes. The result obtained is shown below.

Benchmark information

Name	2020-06-08_223253
Aggregation	false
Failure count	0
Starting timestamp	Jun 8, 2020 10:32:53 PM
Warm up time spent	30,000 ms
Parallel benchmark count / available processors	1 / 4
Benchmark time spent	7,237,693 ms
Environment mode	REPRODUCIBLE
Logging level org.optaplanner.core	info
Logging level org.drools.core	info
Solver ranking class	TotalScoreSolverRankingComparator
VM max memory (as in -Xmx but lower)	1,834,483,712 bytes
OptaPlanner version	7.37.0.Final
Java version	Java 1.8.0_111 (Oracle Corporation)
Java VM	Java OpenJDK 64-Bit Server VM 25.111-b14 (Oracle Corporation)
Operating system	Linux amd64 3.13.0-170-generic
Report locale	en_US
Report timezone	Asia/Almaty

Figure 8.2: Benchmark conducted server information

Benchmark executed 4 algorithms in parallel, using Java 1.8.0 in Linux operating system. Maximum allocated memory is 1,834,483,712 bytes (1,8 Gigabytes)

In 8.3 result comparison between algorithms, for Problem 0, the results are same, because in allocated time they solved it. In Problem 1, Combined Algorithm performs better than other. And. the last, for Problem 3 the results of Late Acceptance algorithm and Combined Algorithm are same, they performed better than other two.

In 8.4 we can summary results, where in overall Combined Algorithm won first place. Late Acceptance algorithm closer to it by result.

In Figure 8.5 Combined algorithm show best scability under size of problem set. This shows that it will potentially do better performance when problem size grows.

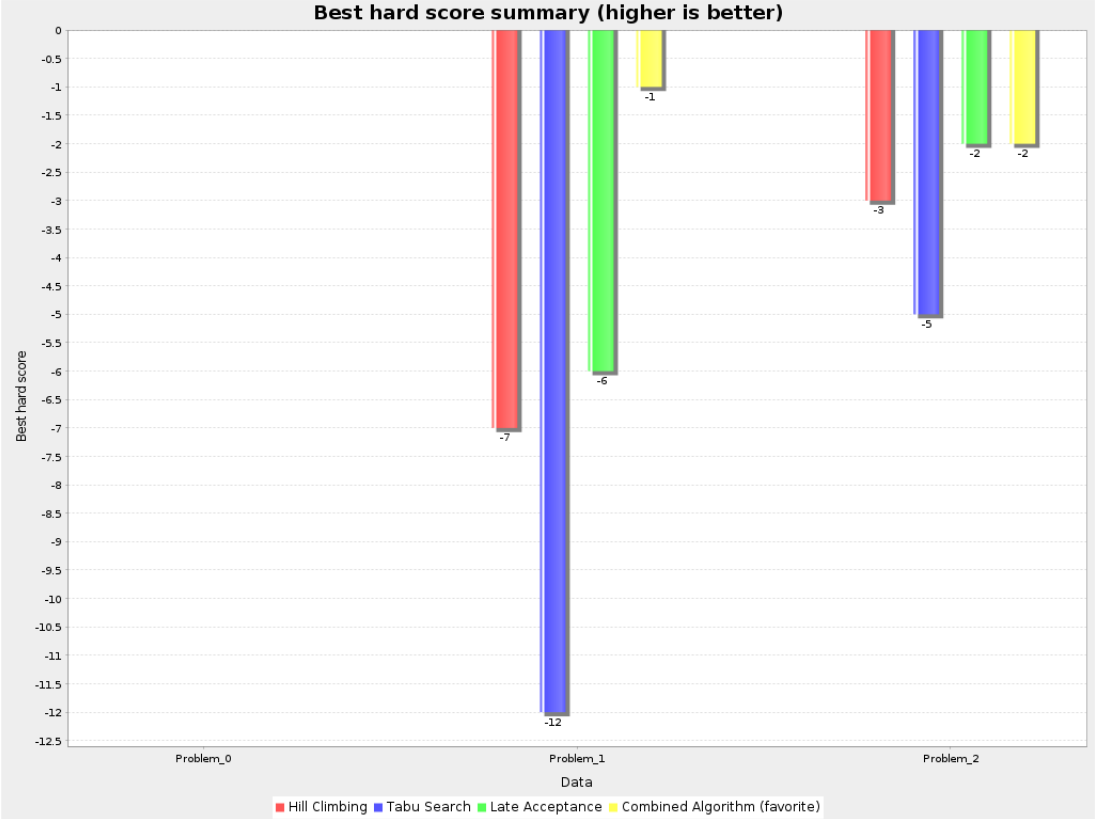

Figure 8.3: Algorithms best score summary

Solver	Total	Average	Standard Deviation	Problem		
				Problem_0	Problem_1	Problem_2
Hill Climbing 2 !	-10hard/-63soft	-4hard/-21soft	2.94/18.24	0hard/-3soft 0	-7hard/-14soft 2 !	-3hard/-46soft 2 !
Tabu Search 3 !	-17hard/-142soft	-6hard/-48soft	4.93/30.69	0hard/-16soft 2	-12hard/-37soft 3 !	-5hard/-89soft 3 !
Late Acceptance 1 !	-8hard/-381soft	-3hard/-127soft	2.52/72.23	0hard/-47soft 3	-6hard/-112soft 1 !	-2hard/-222soft 0 !
Combined Algorithm 0 !	-3hard/-1440soft	-1hard/-480soft	0.82/332.38	0hard/-10soft 1	-1hard/-709soft 0 !	-2hard/-721soft 1 !

Figure 8.4: Algorithms best score table

In figure 8.6 the result difference between results are shown. Combined Algorithm result subtracted from other algorithm results.

In figure 8.7 the percentage difference shown between worst result and others. Here Tabu Search shows worst result among them.

In figure 8.8 bench marker constructed graphic how algorithms performed. as shown they are all finished working in 3 minutes.

In figure 8.9 bench marker constructed graphic how algorithms used memory. The Tabu search algorithm consumed high memory, the Combined algorithm otherwise used low memory.

In figure 8.10 all algorithms didn't get feasible solution, but Combined Algorithm get closer to it than others. This algorithm found this score in approxi-

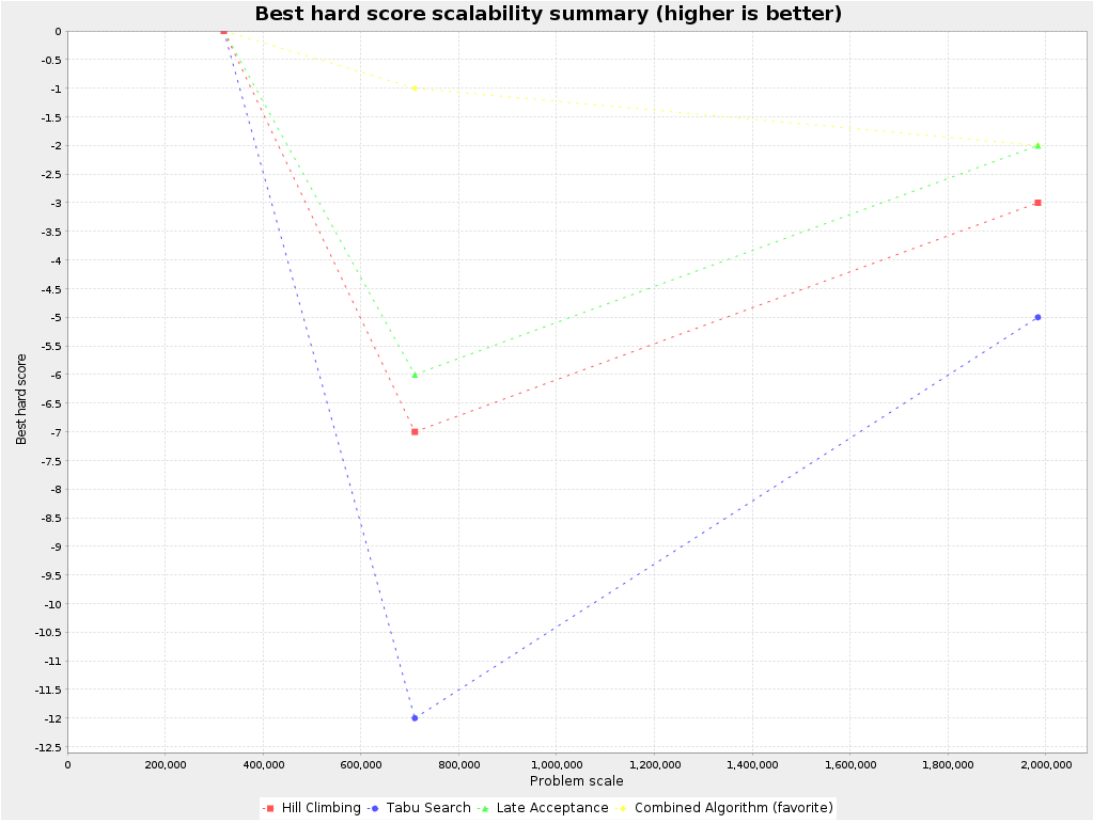

Figure 8.5: Algorithms best hard score scability

mately 6 minutes, but then couldn't improve score.

In figure 8.11 bench marker graphic changed. The memory usage of all algorithms began to grow until the 8th minute, then it began to fall.

In figure 8.12 all algorithms also didn't get feasible solution, but Combined Algorithm and Late Acceptance get closer to it than others. Late Acceptance.

In figure 8.13 bench marker graphic same as for Problem 2. But memory used more than it.

8.3.2 University data set

In university data set courses size is 2376 subject. It size close to Problem 2 data set size. But the form of data a little bit different. A benchmark on university data is very important to improve the outcome of generating a university course schedule. Before generation, you need to clear data from data that does not have a binding. All data must be strongly typed. Exection time for this benchmark is 2 minutes.

In Figure 8.14 the benchmark machines are same as in previous benchmarks.

In Figure 8.15 the combined algorithm performed better than others.

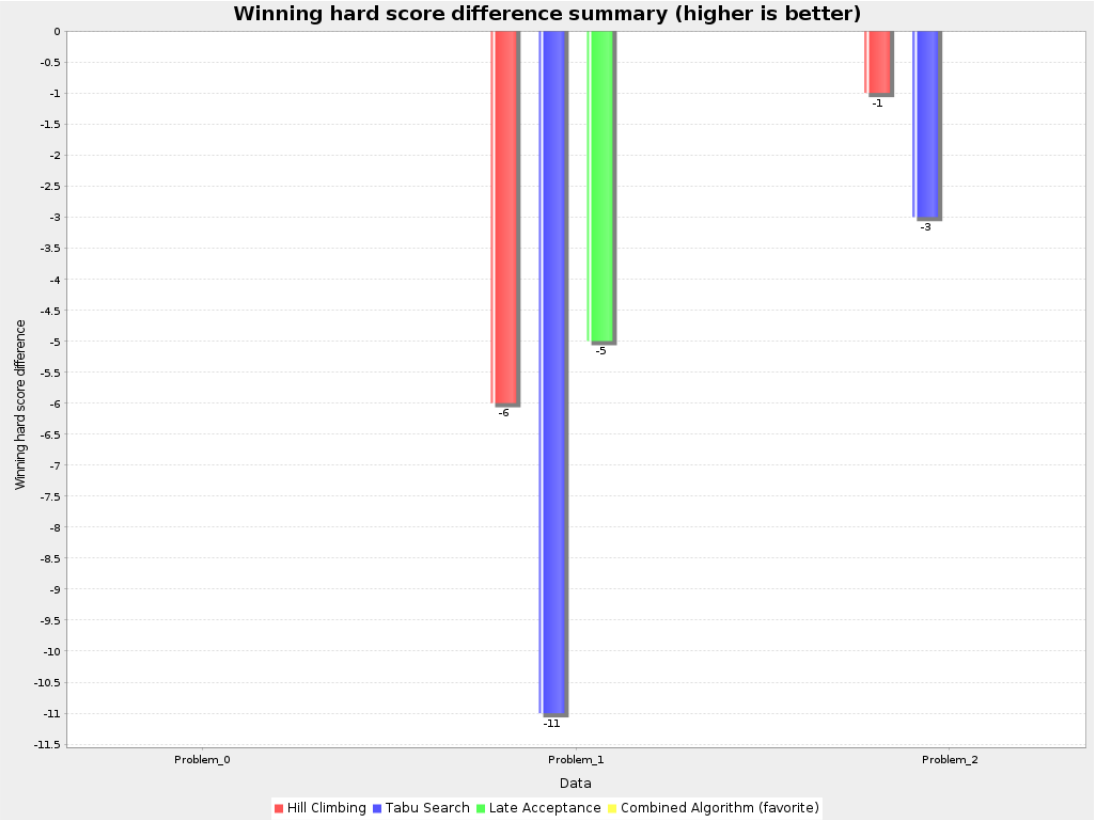

Figure 8.6: Winning hard score summary

In Figure 8.16 show the overall information about benchmark result.

In Figure 8.17 show difference between best and other scores.

In Figure 8.18 show difference between worst and other scores in percentages.

In Figure 8.19 shows the score increasing dynamic for University data set.

In Figure 8.20 shows the speed of calculations for University data set.In end it speeds up.

In Figure 8.21 shows the memory usage of algorithms in University data set.

As the result of the benchmark combined algorithm performed good, so it's a good candidate for use in generating course schedules. Therefore recommended for using it in course schedule generation. Perhaps if someone try other settings then the result can be improved.

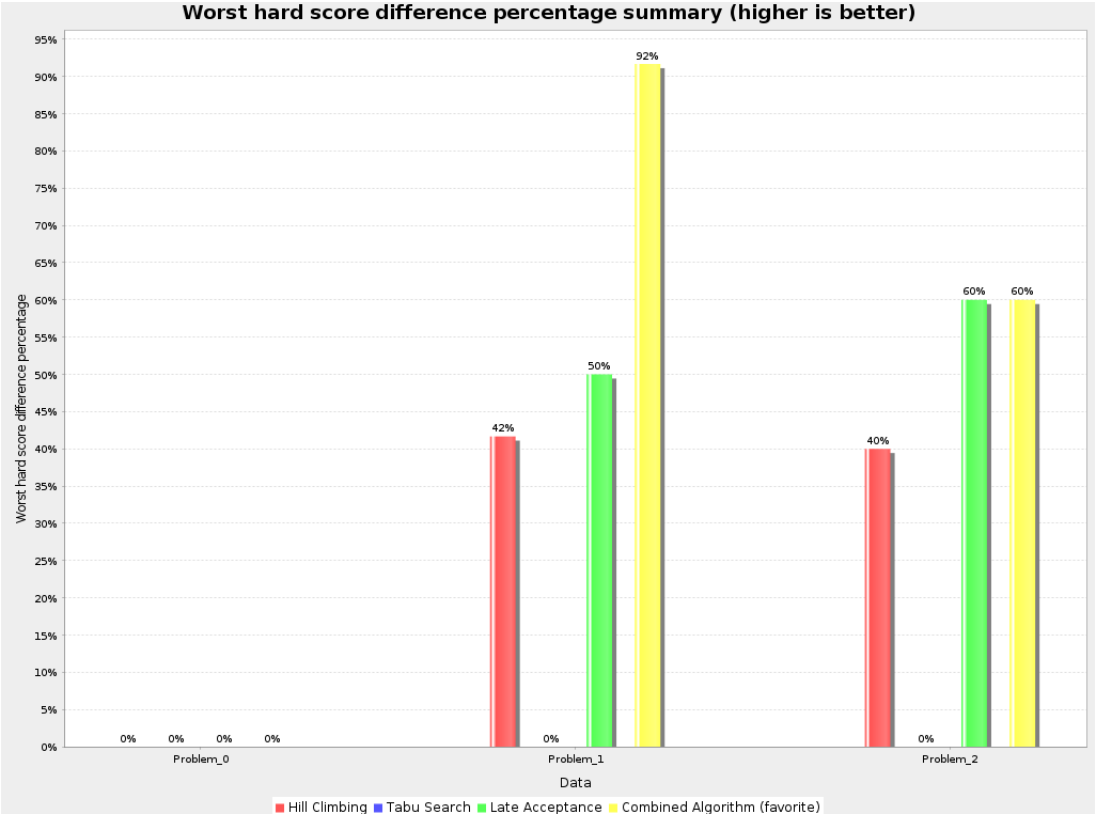

Figure 8.7: Worst score difference

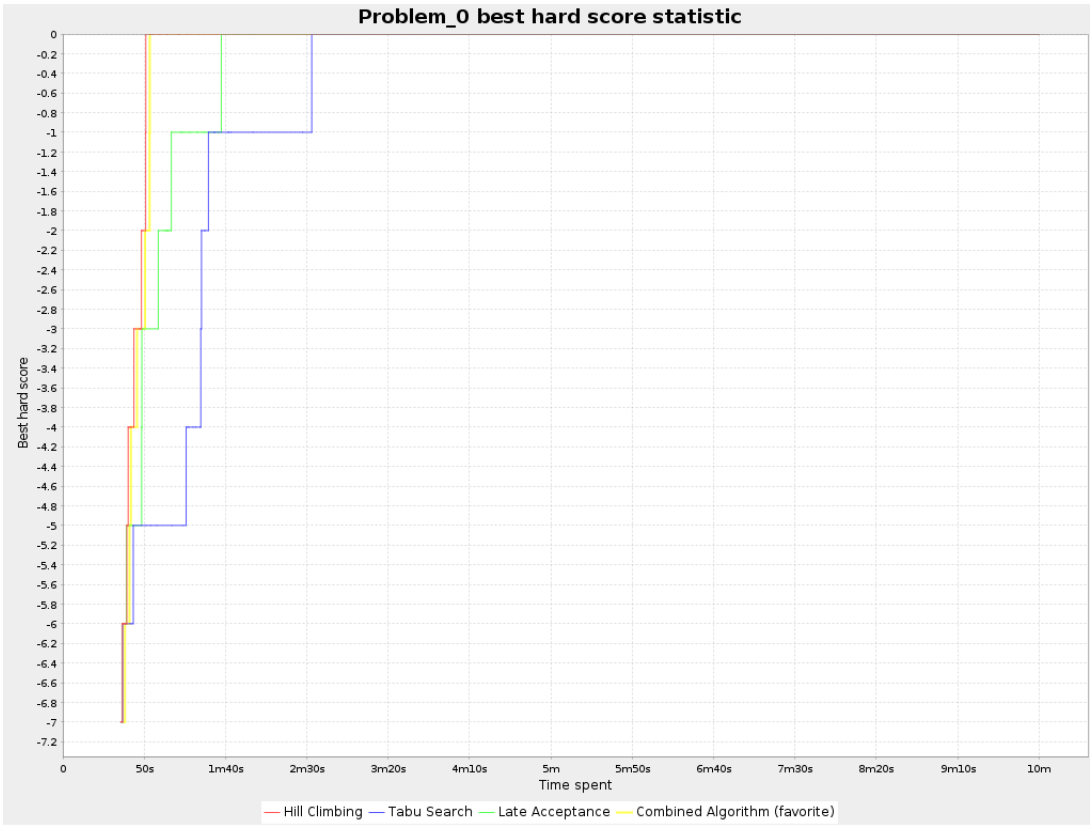

Figure 8.8: Problem 0 best hard score

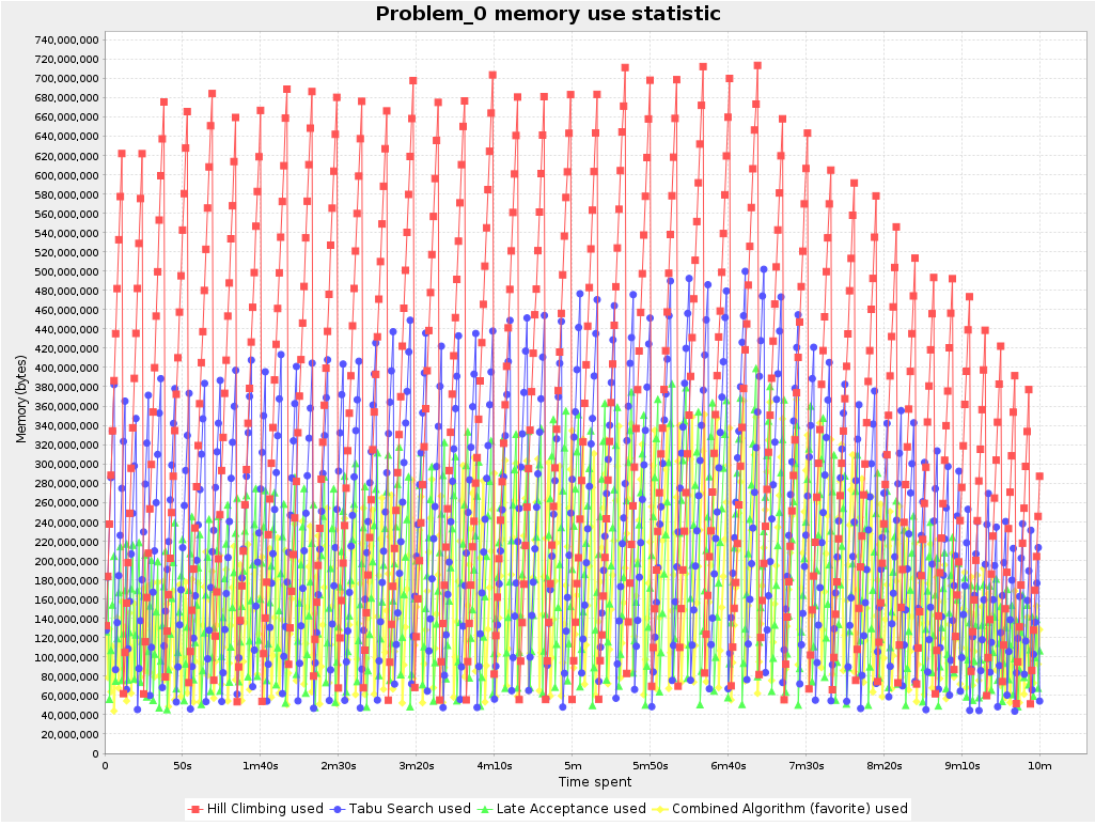

Figure 8.9: Problem 0 worst score difference

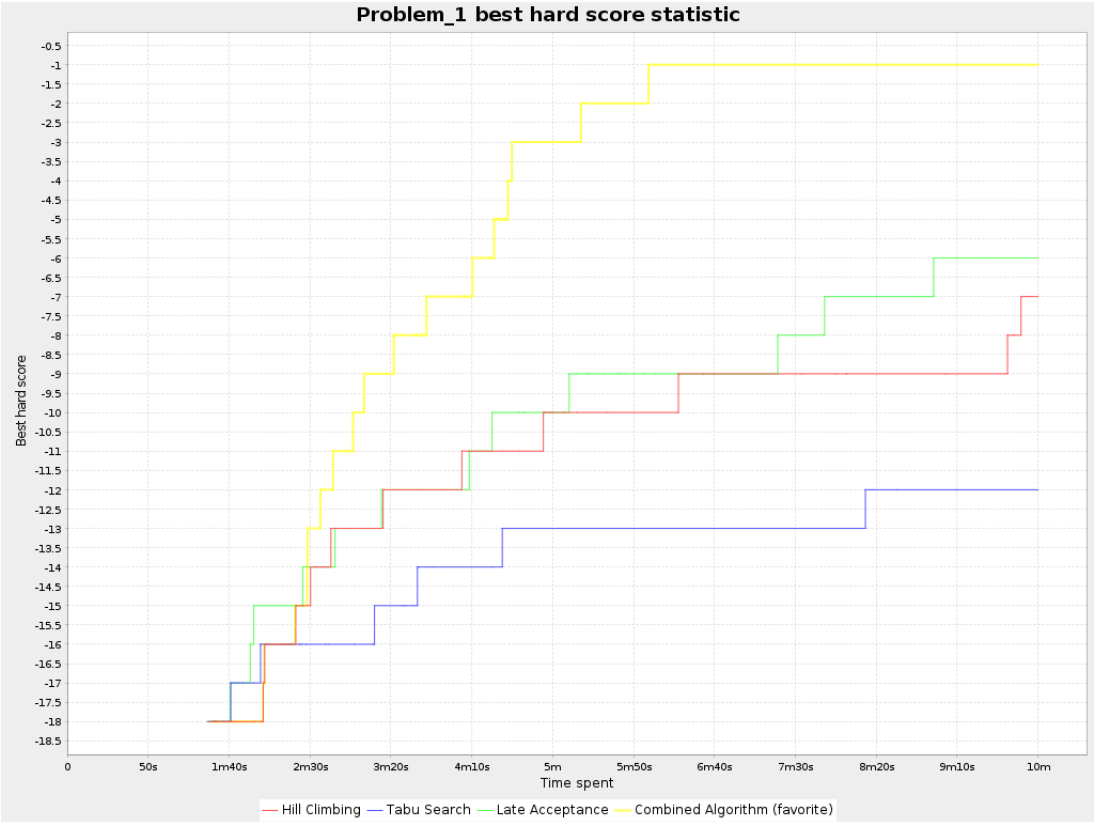

Figure 8.10: Problem 1 best hard score

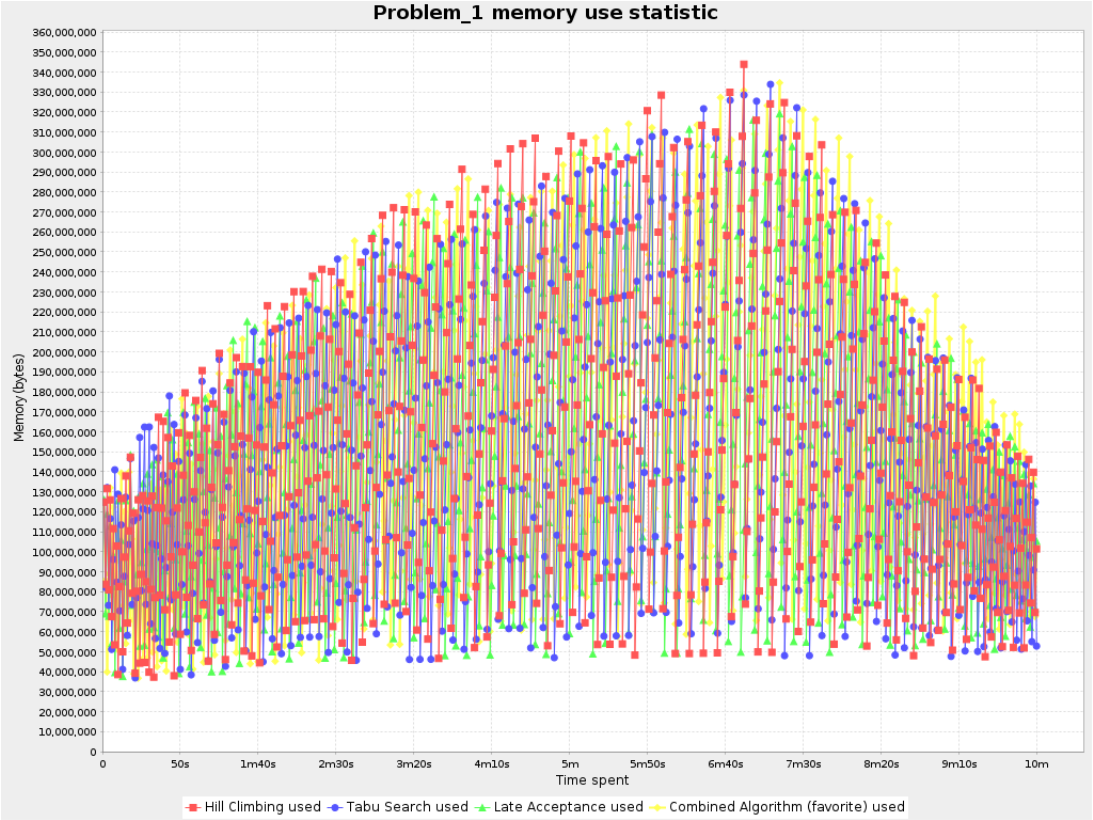

Figure 8.11: Problem 1 worst score difference

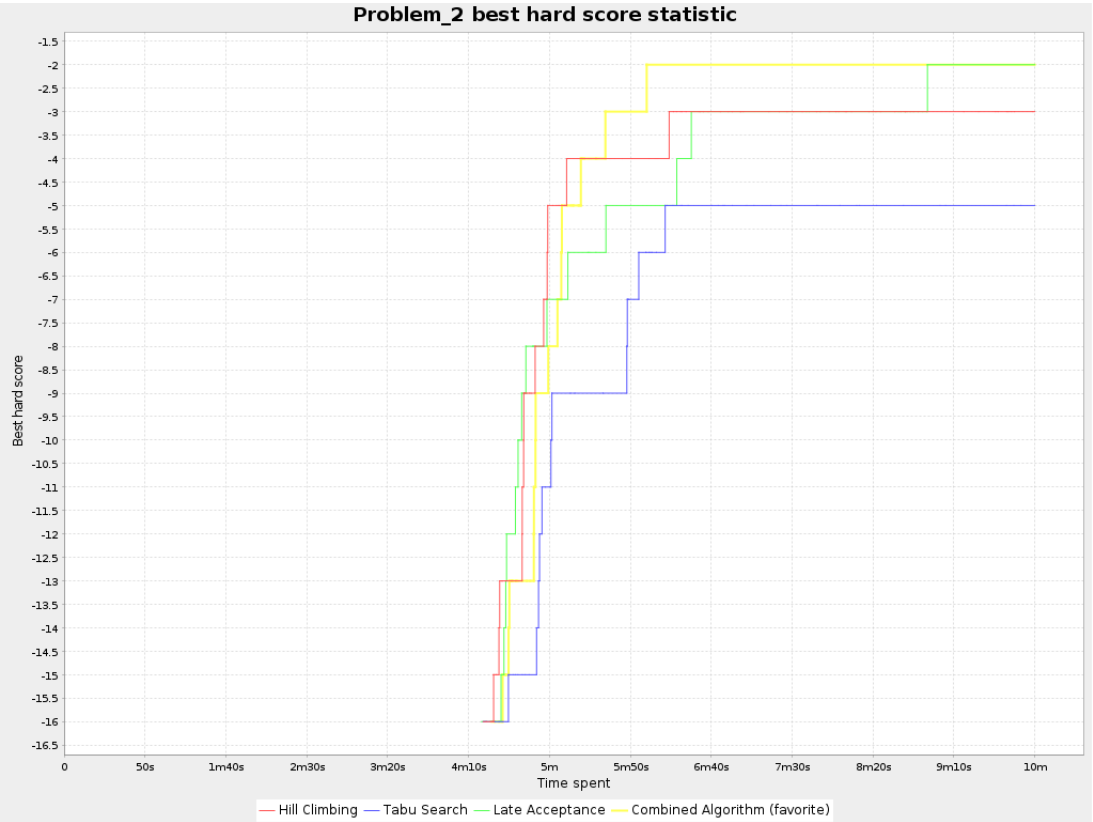

Figure 8.12: Problem 2 best hard score

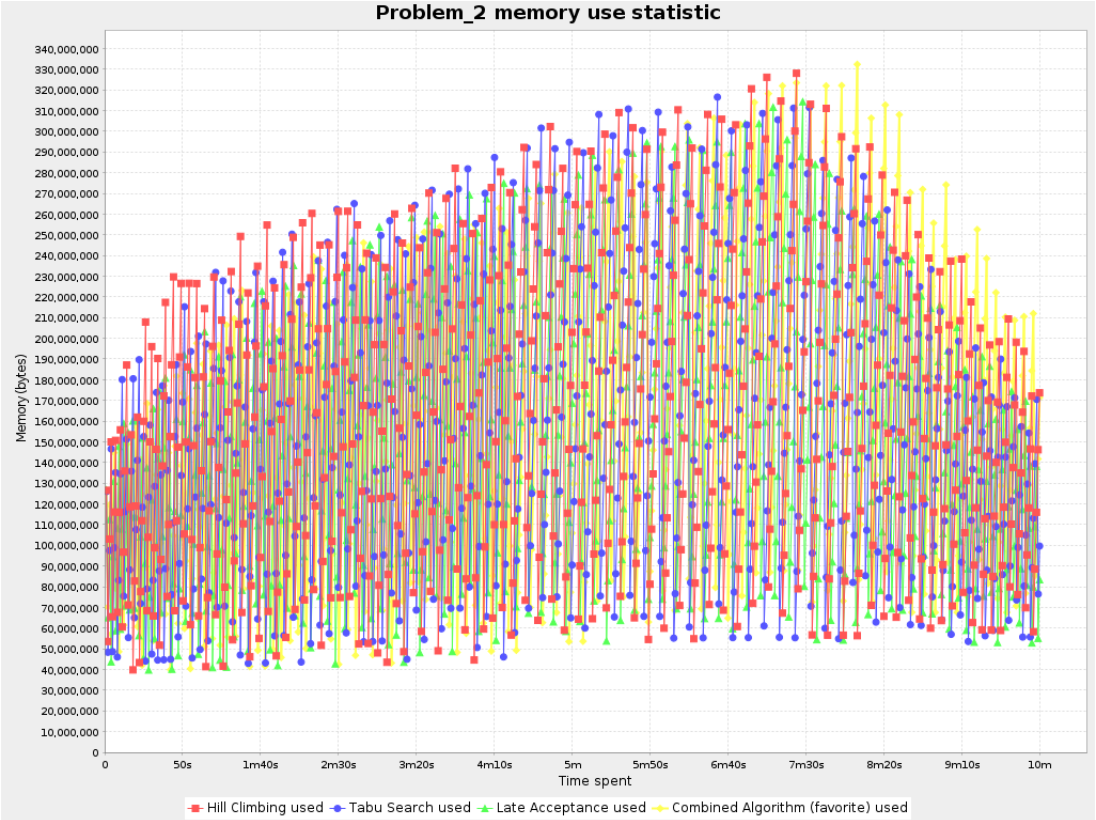

Figure 8.13: Problem 2 worst score difference

Benchmark information

Name	2020-05-15_054208
Aggregation	false
Failure count	0
Starting timestamp	May 15, 2020 5:42:08 AM
Warm up time spent	30,000 ms
Parallel benchmark count / available processors	1 / 4
Benchmark time spent	534,705 ms
Environment mode	REPRODUCIBLE
Logging level org.optaplanner.core	info
Logging level org.drools.core	info
Solver ranking class	TotalScoreSolverRankingComparator
VM max memory (as in -Xmx but lower)	1,834,483,712 bytes
OptaPlanner version	7.37.0.Final
Java version	Java 1.8.0_111 (Oracle Corporation)
Java VM	Java OpenJDK 64-Bit Server VM 25.111-b14 (Oracle Corporation)
Operating system	Linux amd64 3.13.0-170-generic
Report locale	en_US
Report timezone	Asia/Almaty

Figure 8.14: University data benchmark info

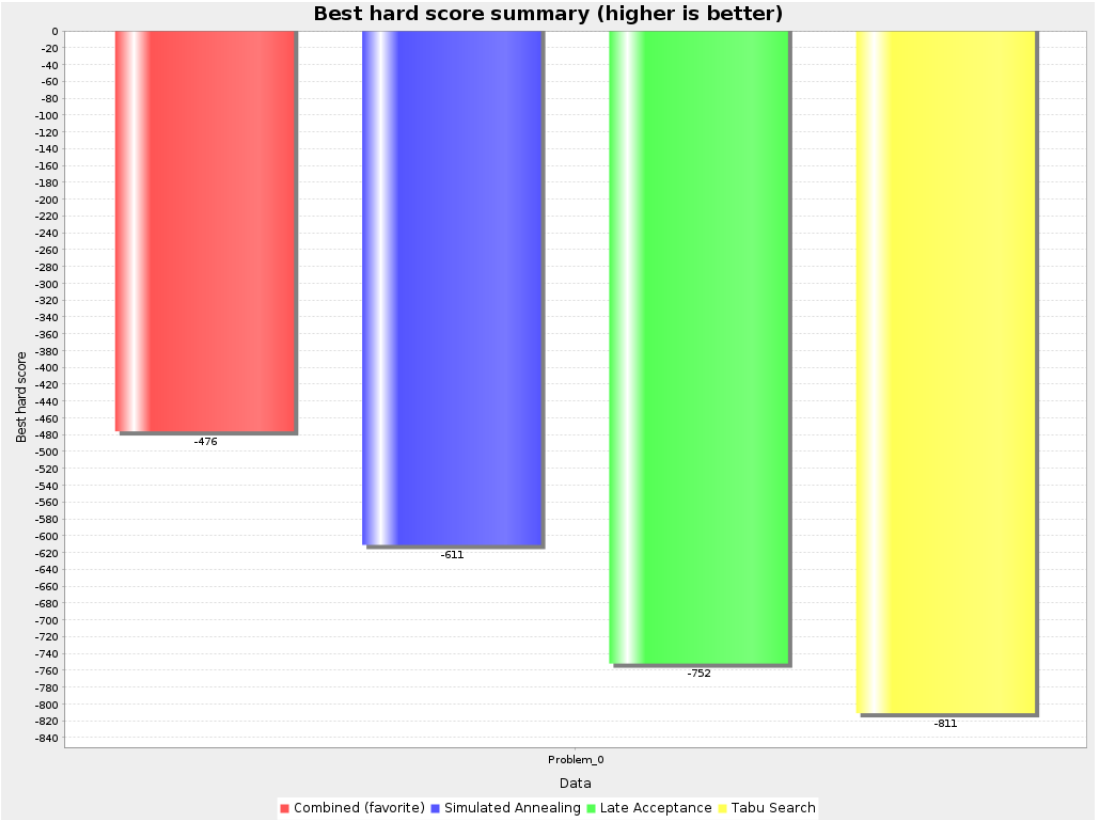

Figure 8.15: University data best hard score summary

Solver	Average	Problem
		Problem_0
Combined 0 !	41.31%/0.00%	41.31%/0.00% 0 !
Simulated Annealing 1 !	24.66%/0.00%	24.66%/0.00% 1 !
Late Acceptance 2 !	7.27%/0.00%	7.27%/0.00% 2 !
Tabu Search 3 !	0.00%/0.00%	0.00%/0.00% 3 !

Figure 8.16: University data best performance table

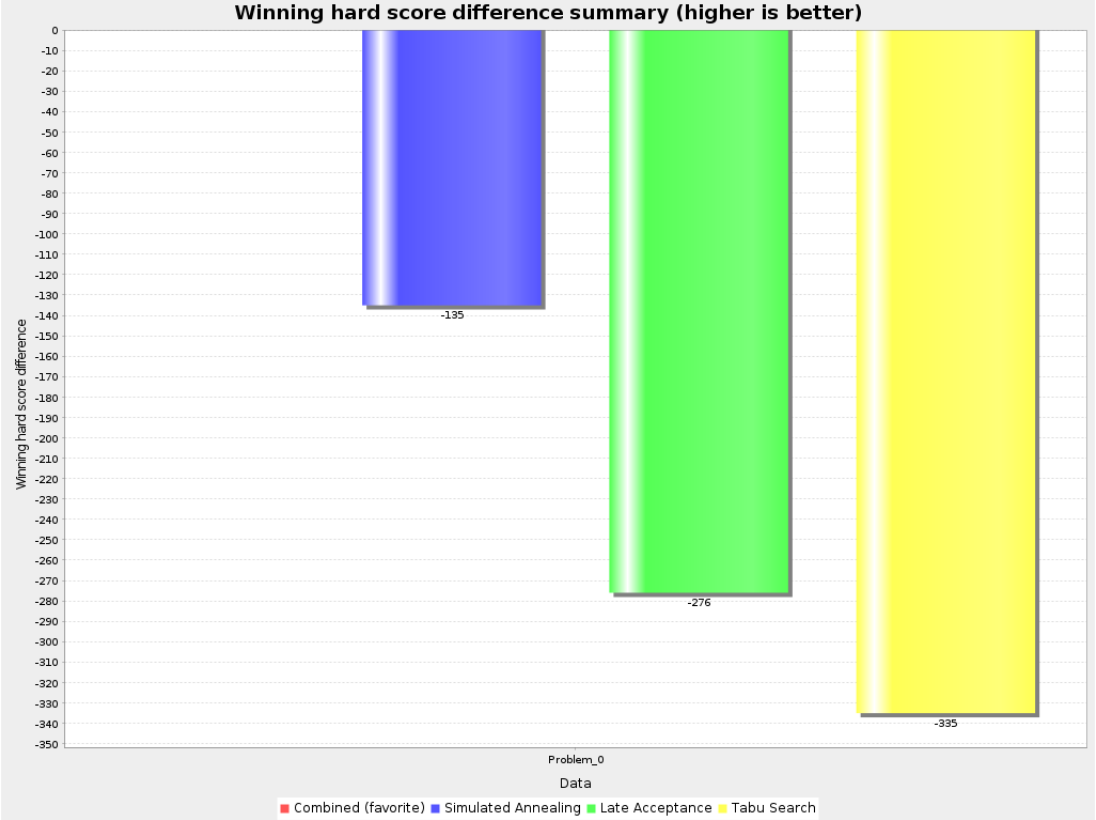

Figure 8.17: University data winning hard score difference summary

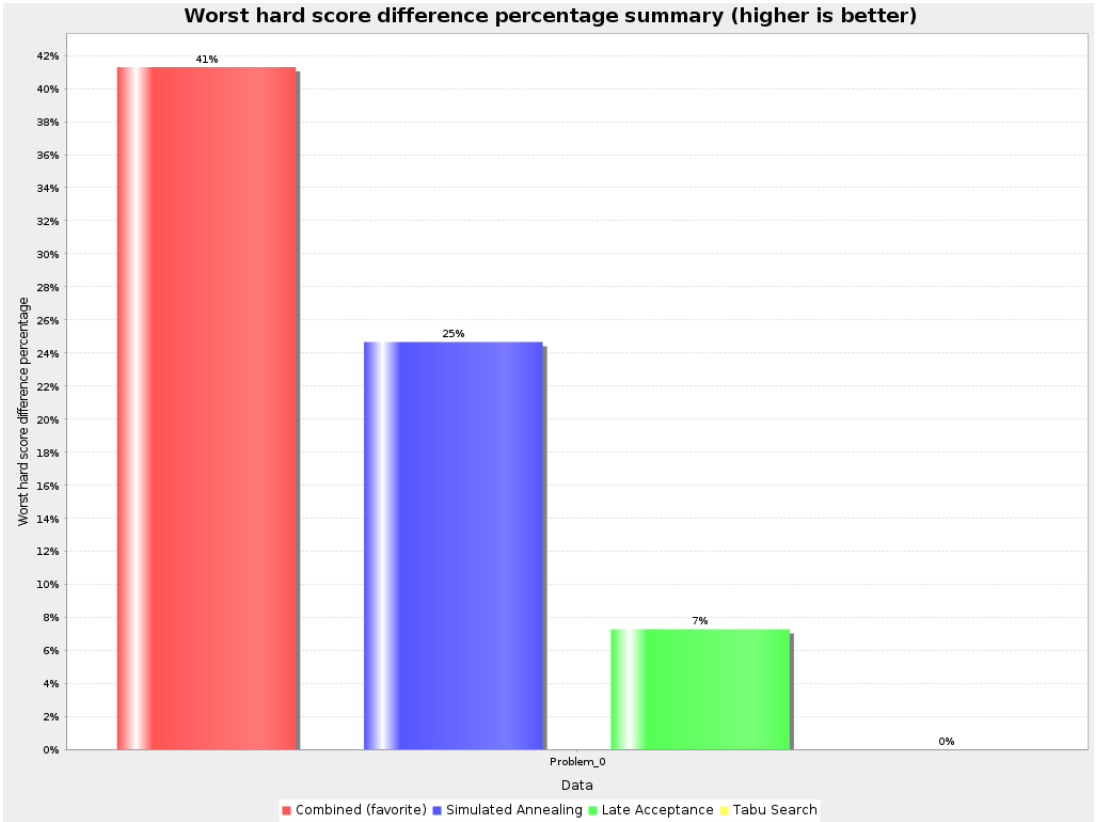

Figure 8.18: University data worst score difference percentage summary

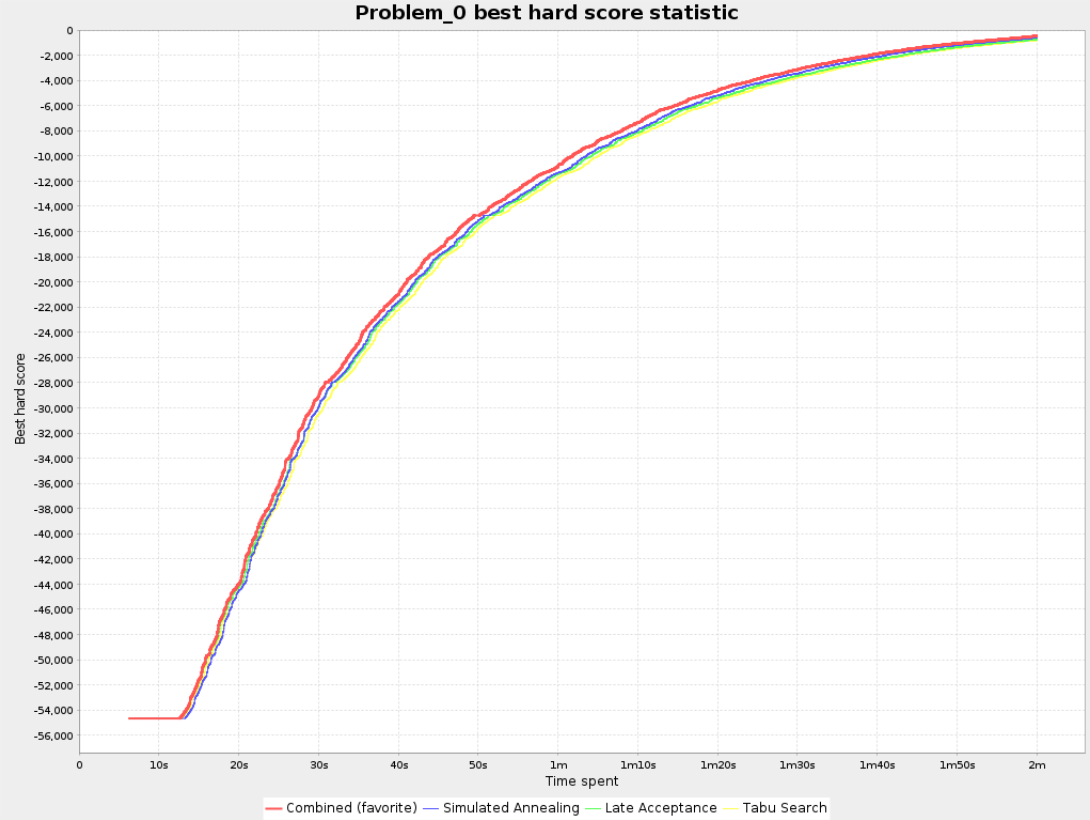

Figure 8.19: University data best hard score statistics

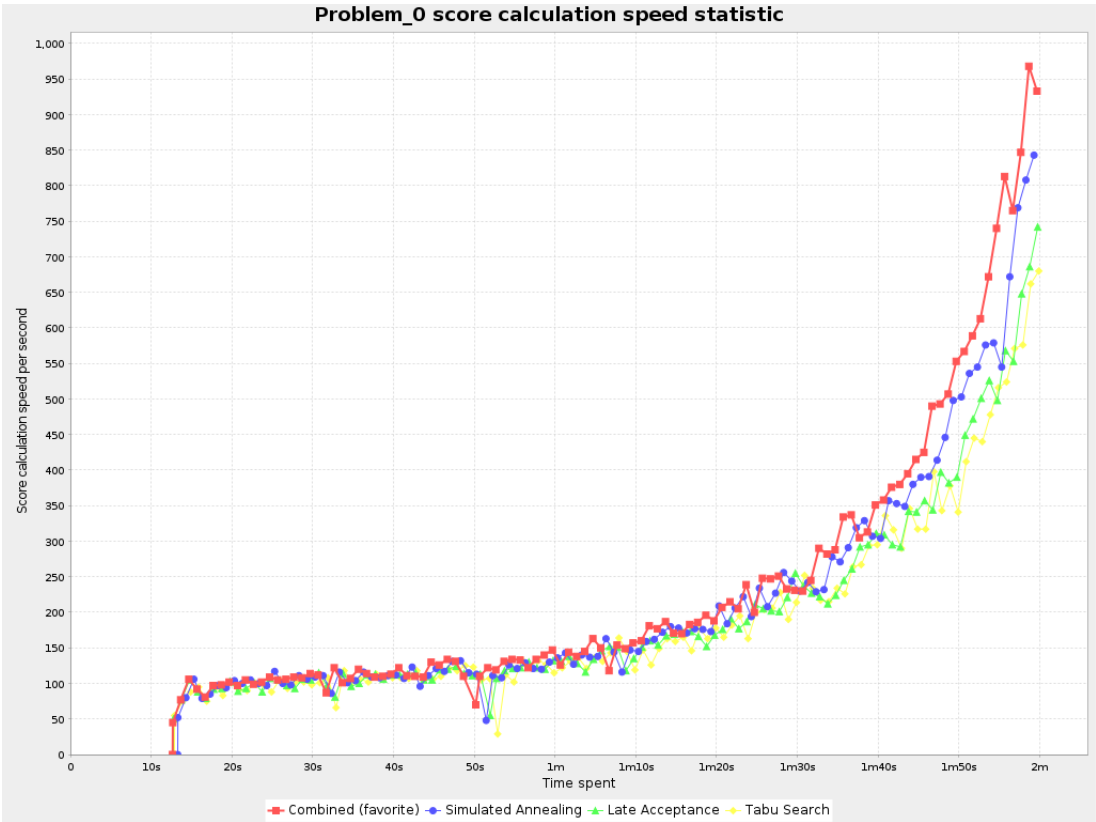

Figure 8.20: University data score calculate speed statistics

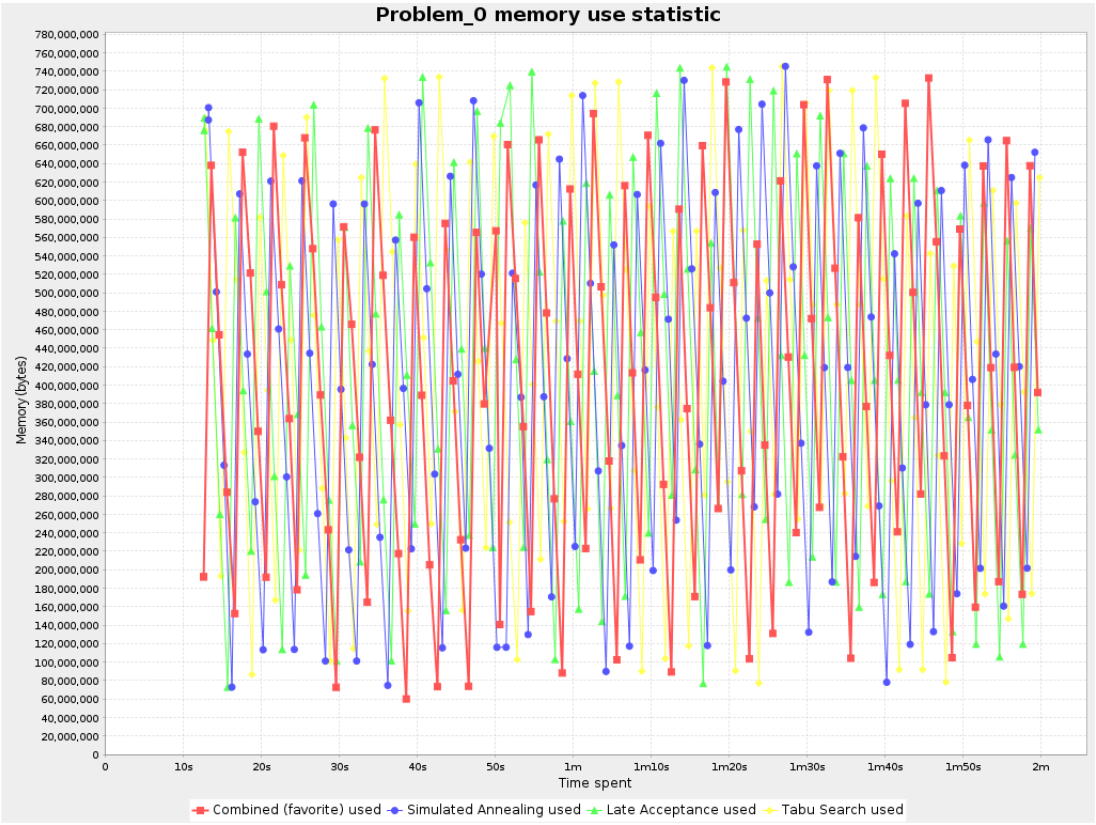

Figure 8.21: University data memory usage statistics

9. Chapter 9

9.1 Result

After Launch, the system produces result which can be used for current academic year. There is always an option to rerun again with different configuration to get better result. It is up to admin users by checking current result satisfaction. Sample produced result which output to user interface.

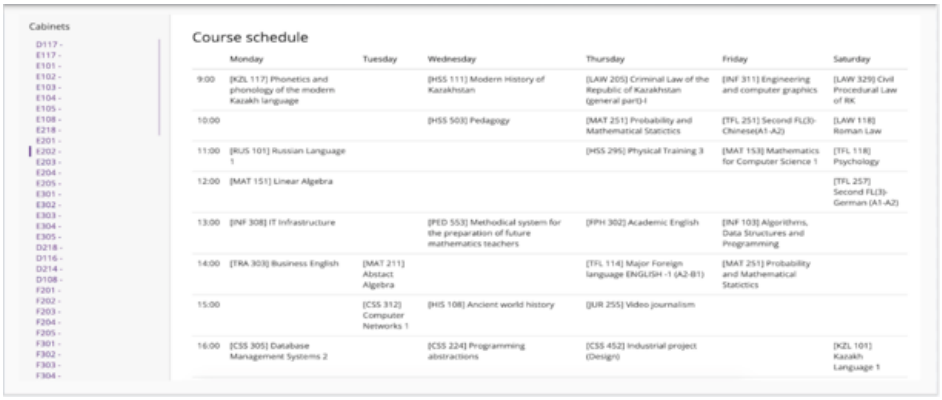

Figure 9.1: Sample Course Schedule Generation program output

9.2 Discussion

The task of generating course schedule depends on the data and the applicable rules, so it needs to find its own approach since the requirements are specific, and it is very difficult to adapt the universal approach to other tasks. Another difficulty is to determine the optimal result for the allocated time, and time is also spent on checking the result and launching it in several stages. Therefore, the requirements must be divided and atomic.

10. Conclusion

This work presents solutions that will work with the University data, therefore deviating in detail from other works. But in the future it is possible to apply for other data that have similar parameters in the amount of data with minor changes. There is always an opportunity to improve the interaction of university data and the generation of course schedules, and I hope that this work will help to prepare good ground in this direction.

References

[1] Schaerf A. *Local Search Techniques for Large High School Timetabling Problems.* IEEE Transactions on Systems, Man and Cybernetics., 1999.

[2] Ossam C. "University Scheduling using Genetic Algorithm." 2009.

[3] *Curriculum Based Course Timetabling.* `http://www.cs.qub.ac.uk/itc2007/curriculmcourse/course_curriculm_index.htm`. Accessed: 2020-05-01.

[4] Deb K. Datta D. and Fonseca C.M. *Multi-objective evolutionary algorithms for resource allocation problems.* Springer-Verlag Press, 2006.

[5] Yuri Bykov Edmund K. Burke. "Proceedings of the Conference on the Practice and Theory of Automated Timetabling". In: PATAT, 2008, p. 1.

[6] Fred Glover. "Tabu Search-Part I." In: *ORSA Journal on Computing, Volume 1, Issue 3: 135-206* (1989), p. 190.

[7] Lhomme O. *Consistency techniques for numeric CSPs.* 13th Intl. Joint Conference on Artificial Intelligence., 1993.

[8] R.J. Willemen. "School timetable construction : algorithms and complexity". English. PhD thesis. Department of Mathematics and Computer Science, 2002. ISBN: 90-386-1011-4. DOI: `10.6100/IR553569`.

[9] Anthony Wren. *Scheduling, Timetabling and Rostering - A Special Relationship?* 1995.