IRSTI 06.58.45

M. Zhuniskhanov[1], R. Suliyev[2]
[1,2]Suleyman Demirel University, Kaskelen, Kazakhstan

OPTIMIZATION ALGORITHMS OVERVIEW FOR COURSE SCHEDULING

Abstract. Today we have a lot of optimization algorithms that can be used for course scheduling, but the main question is which of them fulfills our requirements better and what are the advantages of one from the others. Since course scheduling is an NP-problem, in this article we will look at a few of algorithms that are used to solve different problems and have proved themselves only on the best side and compare their main advantages. When comparing, we will rely not only on the accuracy of the results, but also on the speed of solving complex problems, and on solving complex conditions.

Keywords: algorithm, score, schedule, course, constraint.

***

Аңдатпа. Бүгінде бізде көптеген алгоритмдерді курстар кестесін оңтайландыру үшін пайдалануға болады, бірақ басты мәселе, қайсысы біздің талаптарға жақсы жауап береді және қайсысы басқасынан артықшылықтары бар. Курс жоспарлау NP-проблема болып табылатындықтан, осы мақалада біз бірнеше алгоритмдер қарастырамыз, олар әр түрлі міндеттерді шешу үшін пайдаланылады және өздерін тек жақсы жағынан көрсетті, сонымен қатар олардың негізгі артықшылықтарын салыстырамыз. Салыстыру кезінде, біз нәтижелерінің дәлдігіне ғана емес, сондай-ақ күрделі міндеттерді шешу жылдамдығы мен күрделі жағдайды шешу көрсеткішіне қараймыз.

Түйін сөздер: алгоритм, бағалау кестесі, курс, шектеу.

***

Аннотация. Сегодня у нас есть много алгоритмов оптимизации, которые можно использовать для составления расписания курсов, но главный вопрос в том, какой из них лучше отвечает нашим требованиям и каковы преимущества одного над другими. Поскольку планирование курса является NP-проблемой, в этой статье мы рассмотрим несколько алгоритмов, которые используются для решения различных задач и зарекомендовали себя только с лучшей стороны, и сравним их основные преимущества. При сравнении мы будем полагаться не только на точность результатов, но и на скорость решения сложных задач, а также на решение сложных условий.

Ключевые слова: алгоритм, оценка, график, курс, ограничение.

Introduction

University course scheduling is assigning set of time periods and rooms to a course by scheduling lectures of this course by applying a variety of hard and soft constraints [1]. Constraints are a set of rules for creating a course schedule; they are divided into two types, hard and soft constraints. Our task is to comply with Hard constraints and, if possible, reduce the mismatch to soft constraints. Optimization algorithms each help to solve this problem.

Analysis

To compare the algorithms we choose the concept of score. The more the result matches hard and soft constraint, the higher the score the algorithm will score. Below we will familiarize ourselves with the four types of well-known algorithms and their principles of operation.

A. 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 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.

The Hill Climbing pseudo-code is:

```
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 1. Hill Climbing pseudocode

B. 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 [2]. It works like Hill Climbing, the difference is that it holds a tabu list to avoid local maximums. During movement it is not possible to use the objects that are on the list, since they have already been used. A list can be anything that relates to movement. The list is often customizable.

```
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 2. Tabu Search pseudocode

C. Simulated annealing

Simulated annealing algorithm works in such manner: A step to be a winning step, it's move shouldn't decrease the score. Some case move can decrease the score if it passes a random check. The chance of passing this check decreases relative to the size of score decrement and time the phase passes. The main advantage of this algorithm is that it gives a chance to be selected actions that do not improve the general condition.

```
s = s0
    For k = 0 through kmax (exclusive):
        T ← temperature( (k+1)/kmax )
        Pick a random neighbour, snew ← neighbour(s)
        If P(E(s), E(snew), T) ≥ random(0, 1):
            s ← snew
    Output: the final state
```

Figure 3. Simulated annealing pseudocode

D. Late acceptance

Known also as Late Acceptance Hill Climbing, accept the move which do not decrease the score, or it leads to score that is at least the late score. 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 comparison, 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 [3].

```
Produce an initial solution s
Calculate initial cost function C(s)
Set the initial number of steps I=0
For all k in ( 0.. L-1 ) C_hat[k]=C(s)
Do until a stopping condition
 Construct a candidate solution s*
 Calculate its cost function C(s*)
 v = I mod L
 If C(s*) <= C_hat[v]
 Then accept s*
 Insert cost value into the list C_hat[v] = C(s)
 Increment a number of steps I=I+1
End do
```

Figure 4. Late Acceptannce pseudocode

Dataset

The main goal is to schedule courses of a university, so our dataset formed from course definitions, lecturers, rooms and groups.

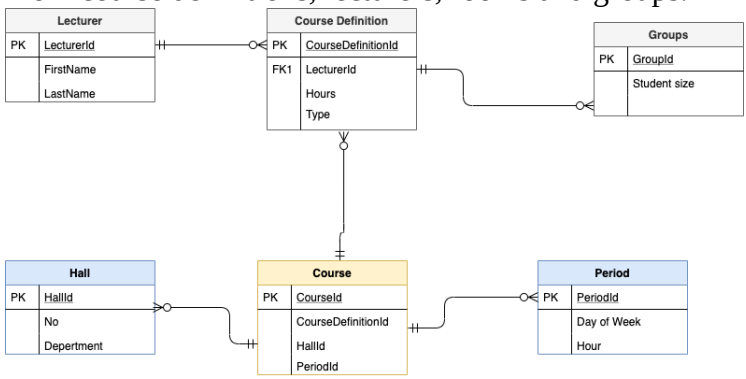

Figure 5. Dataset Relation

Also we will evaluate 3 type of course size to know how it impacts result of algorithm evaluation. And the amount of other dependent data corresponds to its number of courses.

Table 1. *Dataset types*

Абвиатура	Course size
Problem_0	200
Problem_1	400
Problem_2	800

Comparison

For a full check, we will compare the result of the execution of the compared algorithms using the optaplanner opensource program [4]. It integrates easily with other programs and has a very rich and subtle settings. All algorithms

are also supported and a benchmarker is built in to compare results. Below is the result of running benchmarker for 5 minutes.

Table 2. Hard and soft score result. 5-minute timing.

Solver	Total	Average	Standard Deviation	Problem		
				Problem_0	Problem_1	Problem_2
Hill Climbing 0 !	-3hard/-76soft	-1hard/-26soft	1.41/33.73	0hard/0soft 0	0hard/-3soft 0	-3hard/-73soft 0 !
Tabu Search 2 !	-11hard/-117soft	-4hard/-39soft	5.2/38.88	0hard/-3soft 1	0hard/-21soft 1	-11hard/-93soft 2 !
Simulated annealing 3 !	-15hard/-354soft	-5hard/-118soft	6.38/57.28	0hard/-54soft 2	-1hard/-193soft 3 !	-14hard/-107soft 3 !
Late Acceptance 1 !	-4hard/-2027soft	-2hard/-676soft	2.0/749.69	0hard/-69soft 3	0hard/-226soft 2	-4hard/-1732soft 1 !

The Hill Climbing algorithm works well for data such as ours and beyond, so if it is important to get the result very quickly, then the Hill Climbing algorithm is a good choice.

Below figure 6 you can follow the order of growth of compliance with the rules for elapsed time. The Hill Climbing algorithm works well for data such as ours and beyond, so if it is important to get the result very quickly, then the Hill Climbing algorithm is a good choice.

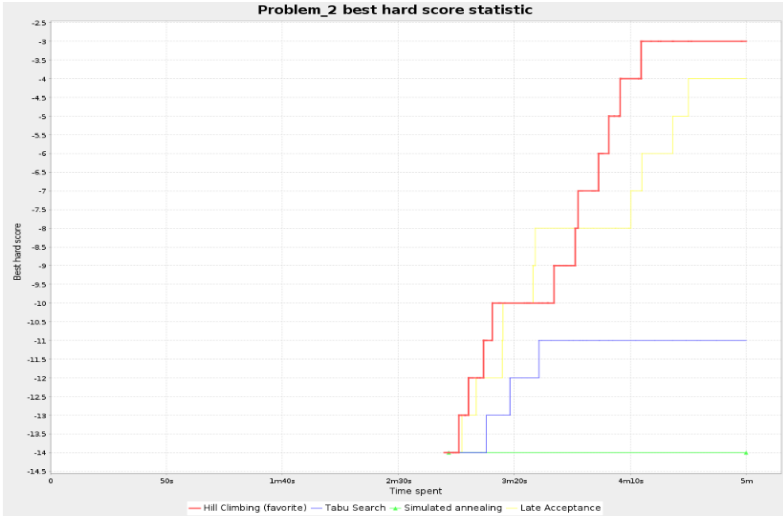

Figure 6. Hard and soft constraint satisfaction vs time consume

Below table 2, you can follow the order of growth of compliance with the rules for elapsed time.

Table 3. Hard and soft score result. 20-minute timing.

Solver	Total	Average	Standard Deviation	Problem		
				Problem_0	Problem_1	Problem_2
Hill Climbing 1 !	-2hard/-17soft	-1hard/-6soft	1.0/6.03	0hard/0soft 0	0hard/-3soft 0	-2hard/-14soft 1 !
Tabu Search 3 !	-7hard/-70soft	-3hard/-24soft	3.37/24.86	0hard/-1soft 1	0hard/-11soft 1	-7hard/-58soft 3 !
Simulated annealing 2 !	-3hard/-593soft	-1hard/-198soft	1.41/170.95	0hard/-29soft 2	0hard/-132soft 2	-3hard/-432soft 2 !
Late Acceptance 0	0hard/-780soft	0hard/-260soft	0.0E0/193.51	0hard/-63soft 3	0hard/-194soft 3	0hard/-523soft 0

For the same data, we will run a benchmark with a time of 20 minutes and look at the final result and the implementation of compliance restrictions.

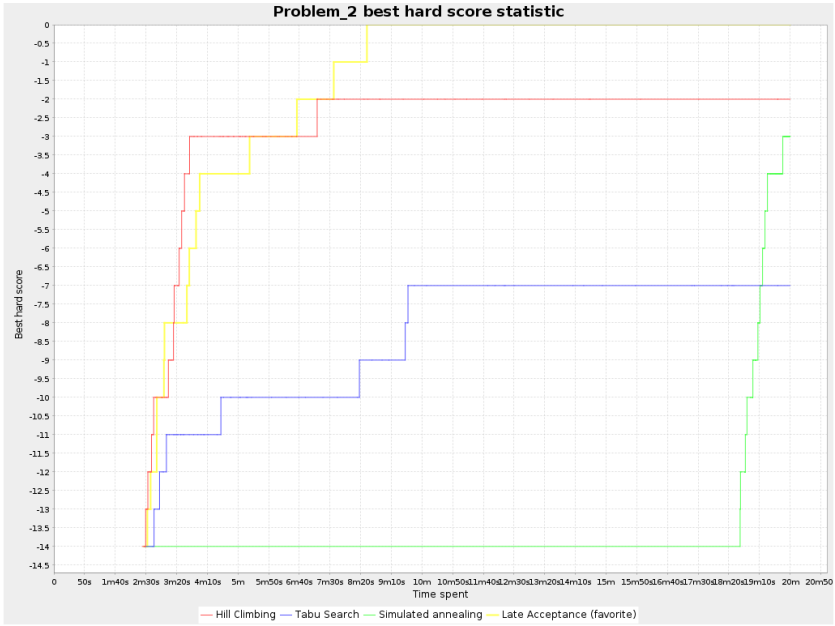

Figure 7. Hard and soft constraint satisfaction vs time consume

Here, according to the results, Late Acceptance showed itself well. In which growth shows stability. For big data, it's similar to ours and when time allows, you need to use this algorithm.

Discussion

The dataset size and evaluation time has impact on the desired result, so you need to analyze it with different values to get the best result. Also, following the graph, you can combine the algorithms, first apply Hill Climbing, since it shows itself well at first, then applying Late acceptance which shows stable growth and gives the best result for long-term work.

Conclusion

As a result, all algorithms are designed to solve the course generation problem, but they do it differently. The four algorithms considered can still be compared with other metaheuristic algorithms to obtain even better results.

When comparing, we compared the results for a short run time where the Hill Climbing algorithm showed better, also checked for a long run time, where another Late Acceptance algorithm showed the highest result.

References

1 Schaerf, A. A survey of automated timetabling. *Articifal Intelligence Review*, 13 (2), (1999): pp. 87-127.

2 Glover, F. Tabu Search-Part I. *ORSA Journal on Computing*, 1 (3), (1989): pp. 135-206.
3 Edmund, K.B., Bykov. Y. Conference: Proceedings of the Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), p.1.
4 What is OptaPlanner? URL: https://www.optaplanner.org/