StuCoSReC

Proceedings of the 2019 6th Student Computer Science Research Conference

University of Primorska Press

StuCoSReC
Proceedings of the 2019 6th Student
Computer Science Research Conference

Edited by
Iztok Fister Jr., Andrej Brodnik, Matjaž Krnc
and Iztok Fister

Published by
University of Primorska Press
Titov trg 4, SI-6000 Koper

Editor-in-Chief
Jonatan Vinkler
Managing Editor
Alen Ježovnik

Koper, 2019

ISBN 978-961-7055-82-5 (pdf)
www.hippocampus.si/ISBN/978-961-7055-82-5.pdf
ISBN 978-961-7055-83-2 (html)
www.hippocampus.si/ISBN/978-961-7055-83-2/index.html
DOI: https://doi.org/10.26493/978-961-7055-82-5

Kataložni zapis o publikaciji (CIP) pripravili
v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID=302029568
ISBN 978-961-7055-82-5 (pdf)
ISBN 978-961-7055-83-2 (html)

Preface

Computer science is now among the most popular study programmes worldwide. We live in a digital age where most industries rely on data and software programmes. From transport infrastructure to public health systems, banking and communications, computer science is everywhere. Technology has made the world better, faster, and more connected. However, it is easy to miss an important component of this exciting success story.

Such development was made possible thanks to the brilliant minds of IT graduates, who took their passion for technology and used it to create ground breaking gadgets and computer programmes. Here in Slovenia, the three public universities share these values and invest heavily in their computer science students. These efforts facilitate collaboration among our departments, resulting in joint events such as this StuCoSRec student conference.

We are proud that, over the past five years, these Student Computer Science Research Conferences have grown in several ways. In this 6th installment, we received 24 full-paper submissions and one abstract submission. Among these, 21 papers were accepted to these proceedings, and 22 talks are scheduled to be presented during the conference, in three parallel sessions. The continued internationalization of our departments is also reflected with the authors of ten full-paper submissions originating from outside of Slovenia.

The conference is dedicated to graduate and undergraduate students of computer science and is therefore free of charge. We gratefully acknowledge the support of the Faculty of Mathematics, Natural Sciences and Information Technologies (University of Primorska).

Matjaž Krnc

Contents

Preface II

Papers

LSTM Network for Stock Trading
• Dušan Fister and Timotej Jagrič 5–8

Defining computational thinking framework for introductory programming in higher education
• Boštjan Bubnič 9–12

Passive Floating Probe
• Michele Perrone, Urban Knupleš, Mitja Žalik, Vid Keršič and Tadej Šinko 13–17

Efficient Collision Detection for Path Planning for Industrial Robots
• László Zahorán and András Kovács 19–22

Sensitivity analysis for p-median problems
• Ágnes Vida and Boglárka G.-Tóth 23–26

A two-stage heuristic for the university course timetabling problem
• Máté Pintér and Balázs Dávid 27–30

Detection of different shapes and materials by glasses for blind and visually impaired
• Urban Košale, Pia Žnidaršič and Kristjan Stopar 31–34

Comparison of clustering optimization for classification with PSO algorithms
• Klemen Berkovič, Uroš Mlakar, Borko Bošković, Iztok Fister and Janez Brest 35–42

Hierarchical Routing Algorithm for Industrial Mobile Robots by Signal Temporal Logic Specifications
• Balázs Csutak, Tamás Péni and Gábor Szederkényi 43–47

Decolorization of Digital Pathology Images: A Comparative Study
• Krishna Gopal Dhal, Swarnajit Ray, Arunita Das, Iztok Fister Jr. and Sanjoy Das 49–52

Solving multi-depot vehicle routing problem with particle swarm optimization
• Matic Pintarič and Sašo Karakatič 53–56

Recognizing the subject exposure from the EEG signals with artificial neural networks
• Sašo Pavlič and Sašo Karakatič 57–60

Transfer Learning Tuning Utilizing Grey Wolf Optimizer for Identification of Brain Hemorrhage from Head CT Images
• Grega Vrbančič, Milan Zorman and Vili Podgorelec 61–66

System for remote configuration and over the air updates in restricted environments
• Marko Zabreznik and Jernej Kranjec 67–70

Covering problems and Influence maximization
• Gyöngyvér Vass and Boglárka G.-Tóth 71–74

Strong deep learning baseline for single lead ECG processing
• Csaba Botos, Tamás Hakkel, Márton Áron Goda, István Z. Reguly and András Horváth 75–83

Primerjava osnovnega algoritma po vzoru obnašanja netopirjev in njegove hibridne različice HBA
• Žan Grajfoner and Lucija Brezočnik 85–90

Nadgradnja algoritma FLORS za besednovrstno označevanje slovenskih besedil
• Domen Kavran, Robi Novak, Jan Banko, Rok Potočnik, Luka Pečnik and Borko Bošković 91–99

Analiza igralnih strategij v iterativni zaporniški dilemi
• Klemen Kac and Bor Praznik 101–106

Napovedovanje nogometnega zmagovalca z rekurentno nevronsko mrežo LSTM
• Nejc Planer and Mladen Borovič 107–110

Izboljšanje zaznave sovražnega in zlonamernega govora s pomočjo slovarja besed
• Sašo Kolac, Aljaž Soderžnik, Simon Slemenšek and Borko Bošković 111–114

Investigating patterns using cellular automata
• László Tóth 115

A two-stage heuristic for the university course timetabling problem

Máté Pintér
University of Szeged, Institute of Informatics
Árpád tér 2
Szeged, Hungary, 6720
pmate955@gmail.com

Balázs Dávid[*]
InnoRenew CoE
Livade 6
Izola, Slovenia, 6310
balazs.david@innorenew.eu
University of Primorska, FAMNIT
Glagoljaška ulica 8
Koper, Slovenia, 6000
balazs.david@famnit.upr.si

ABSTRACT

This paper presents a two-stage solution method for the curriculum-based university course timetabling problem. First, an initial solution is created using a recursive search algorithm, then its quality is improved with a local search heuristic. This method utilizes a tabu list of forbidden transformations, as well as a destructive step at the end of each iteration. The algorithm is tested on datasets of the 2007 International Timetabling Competition.

Keywords
University course timetabling; Local search; Heuristic

1. INTRODUCTION

Creating the timetable of employees is a crucial task for any organization, and educational institutions are no exceptions. While timetabling problems can sometimes be extremely hard, they are still solved manually in many cases. This process can take a long time, and the efficiency of the resulting timetables is not guaranteed.

There are several different types of timetabling problems that have to be solved in academia, mainly connected to the students or the employees. This paper will cover such a specific problem, namely university course timetabling.

The first section of this paper will introduce the field of course timetabling, and present the curriculum-based university course timetabling problem in detail. After this, we present a two-stage heuristic for the solution of the problem, which is then tested on datasets of the 2007 International Timetabling Competition. These instances are based on real world timetables of the University of Udine, Italy.

[*]Supervisor

2. COURSE TIMETABLING

Course timetabling is an optimization problem, where resources (courses, rooms, teachers) are allocated to create a timetable that satisfies given constraints. The problem schedules courses and teachers into available time-slots, while trying to avoid the different types of conflicts that can arise. Many variations exist for the course timetabling problem, but the most important ones are the following:

- *Curriculum-based Course Timetabling*: courses belong to one of several curricula, and courses of the same curriculum cannot be scheduled in overlapping time-slots.
- *Post Enrollment based Course Timetabling*: the problem also considers the students enrolled to the courses, and introduces several constraints connected to their attendance.
- *Examination Timetabling*: similar to the general timetabling problem, but considers exam committees instead of teachers, and also deals with the availability of the students for their required exams.

This paper will give a solution algorithm for the curriculum-based course timetabling problem. The following sections will define the problem, present its necessary resources and constraints, and give a short literature overview of the field.

2.1 Problem definition

The curriculum-based university course timetabling problem schedules a set C of courses over a horizon of d days to create a timetable. Each day is divided into s time-slots, and a course has to be assigned to one or more consecutive slots (depending on its length). A room has to be selected where the course will take place, and a teacher is also assigned as the instructor. Courses can belong to different curricula, introducing additional constraints to the assignment. Courses of the same curriculum usually cannot overlap, or should be scheduled close to each other in time. The constraints of the problem belong into two different groups.

Hard constraints should always be respected, and a timetable is not allowed to violate any of them. The most common hard constraints are the following:

- *Course assignment:* The timetable must contain all courses, and every course has to be assigned to exactly one room, time-slot and teacher.
- *Room availability:* A given room cannot have more than one assigned course at the same time-slot.
- *Teacher availability:* A given teacher cannot be assigned to more than one course at the same time-slot.
- *Room capacity:* A course cannot be scheduled to a room with less capacity than the number of students on the course. Some papers consider this as a soft constraint.
- *Teacher availability:* Teachers can have time-slots when they are not available. Naturally, no course can be assigned to a teacher at a time-slot where they are not available. Some papers consider this as a soft constraint.

Soft constraints can be violated, but they come with a penalty. The typical soft constraints are the following:

- *Compactness:* Isolated courses in a curriculum should be avoided. A course is isolated, if no other course of the same curriculum is scheduled in its previous or next time-slot.
- *Room stability:* Courses of the same curriculum should be assigned to the same room, if possible.
- *Minimal number of days:* Courses of the same curriculum should be spread out over the week: they should be scheduled to a given d amount of days.
- As it was mentioned above, some of the hard constraint (room capacity, unavailability of teachers) can also be considered as a soft constraint.

The objective of the problem is to create a timetable that does not violate any hard constraint, and has a minimum penalty associated to the violations of its soft constraints.

2.2 Literature overview

University course timetabling has been researched intensively in the past decades. The problem itself is NP-complete, which was proven by Even et al. [8], and as a result, many different solution methods were considered for its solution. A good overview of these is given by Bettinelli et al. [4] and Babaei et al. [2]. In the following, we will present some the most important of the approaches. An early mathematical model was given by [1], who consider it as a 3-dimensional assignment problem between courses, time-slots and rooms. A more efficient, two-phase mathematical model is presented by Lach et al. [9], where only courses and time-slots are assigned using a binary variable, and the possible rooms for each time-slot are described by an undirected graph. This approach reduces the model size, and is able to provide solutions for significantly bigger instances.

As mathematical approaches can result in a large model even for relatively small instances, various heuristic methods were also developed for the problem. Different variations of the local search were presented, such as the tabu search of Lü and Hao [10] or the simulated annealing of Bellio et al. [3]. Improved versions of Dueck's Great Deluge algorithm [7] - a genetic algorithm with a philosophy close to local search - were also been published [5]. Hybrid algorithms with multiple stages are also available, like Müller [11] or Shaker et al. [12], both of which are modifications of the Great Deluge.

3. A TWO-STAGE HEURISTIC

In this section, we propose a two-stage heuristic for solving the curriculum-based university course timetabling problem. First, an initial feasible solution is created using a greedy recursive algorithm, then a local search heuristic is utilized to improve its quality.

We apply the following soft constraints from Section 2: Compactness, Room capacity, Room stability, Minimum number of days. The reason for this is that we use the datasets of the Curriculum-based Course Timetabling track of the International Timetabling Competition 2007 (ITC) [6] as evaluation for the method, and this competition also considers the same constraints.

The initial solution is created using a recursive search, which only considers the hard constraints of the problem. The pseudo-code of this can be seen in Algorithm 1.

Algorithm 1 Recursive algorithm for initial solution.

```
Funct recSol($course, node, list$)
 1: if All courses are assigned then
 2:     return TRUE
 3: end if
 4: if No more assignment possibilities then
 5:     return FALSE
 6: end if
 7: if ($course$, $node$) assignment is invalid then
 8:     $node$ := next ($timeslot$,$room$) pair
 9:     return recSol($course, node, list$)
10: end if
11: if $course.teacher$ is available az $node.timeslot$ then
12:     $list \leftarrow$ ($course$,$node$) assignment
13:     $node$ := next ($timeslot$,$room$) pair
14:     $course$ := next free course
15:     return recSol($course, node, list$)
16: else
17:     $node$ := next ($timeslot$,$room$) pair
18:     return recSol($course, node, list$)
19: end if
```

The function requires 3 input data: the course to be considered ($course$), the proposed time-slot and room pair ($node$), and a list of nodes corresponding to the partial solution that is already built ($list$). Initially, $list$ is empty, and $course$ and *node* are randomly chosen. If the assignment of the current course is not possible to this node, then another one is chosen and a recursive call is made. If the course is compatible with the node, and the teacher of the course is also available at the time-slot in question, then the assignment is saved, and the next recursive call will feature a new course. The algorithm terminates if all the courses are assigned, or if there are no more possible nodes to choose from.

As the initial solution is built without considering any soft constraints, it will have a high penalty. A local search method is then used to decrease this penalty by also taking the soft constraints of the problem into consideration. The outline of the algorithm can be seen in Algorithm 2.

Algorithm 2 Local search algorithm.

```
Funct localSearch($tt$, $tabu$, $n$)
 1: $i := 0$
 2: $bestSol := tt$
 3: while $n > 0$ do
 4:     while $foundBetter$ = TRUE do
 5:         $bestCost$ = cost($tt$)
 6:         for Each $a := (course, timeslot, room)$ in $tt$ do
 7:             for Each $b := (timeslot, room)$ in $tt$ do
 8:                 if $(a,b) \in tabu$ then
 9:                     CONTINUE
10:                 end if
11:                 $neighbor := tt$.swap($a$,$b$)
12:                 if $neighbor$ violates hard constraints then
13:                     CONTINUE
14:                 end if
15:                 if cost($neighbor$) $< bestNeighCost$ then
16:                     $bestNeighCost$ := cost($neighbor$)
17:                     $bestNeigh := neighbor$
18:                     $tabutCand := (b,a)$
19:                 end if
20:             end for
21:         end for
22:         if $bestNeighCost < bestCost$ then
23:             $bestCost$ := cost($bestNeigh$)
24:             $bestSol := bestNeigh$
25:             $tabu \leftarrow tabuCand$
26:             $foundBetter$ := TRUE
27:         else
28:             $foundBetter$ := FALSE
29:         end if
30:         if $i > x$ then
31:             $tabu$(0).erase()
32:             $i := 0$
33:         end if
34:         $i := i+1$
35:     end while
36:     destructSearch($bestSol$, $tabu$)
37:     $n := n$-1
38: end while
39: return $bestSol$
```

The input of the algorithm is the *tt* timetable of assignments from the first stage, the empty list *tabu* for forbidden neighborhood transformations, and a parameter n that gives the number of destruction steps. The algorithm considers two different neighborhood transformations:

- *Swap*: The timeslot of a (*course*, *timeslot*,*room*) assignment is swapped with the slot of another assignment.
- *Move*: A (*course*,*timeslot*,*room*) assignment is moved to another (*timeslot*,*room*) position.

The process examines all possible neighborhood moves for a given timetable, and applies the one with the smallest penalty. If the resulting timetable is better than the currently stored best one, then it becomes the new best solution, and the neighborhood move that was used to produce this solution is saved to the *tabu* list of forbidden moves.

If a local optimum is found, the algorithm switches to the *destructSearch* phase. In this phase, a set number of neighborhood moves are applied to the solution, strictly decreasing the quality with each step. After this destruction phase, the local search part of the algorithm is executed again, searching for a new local optimum, while also using the existing *tabu* list to avoid certain transformations. The number of these destruction steps is given by the parameter n in the input.

4. TEST RESULTS

As it was mentioned in the previous Section, the algorithm was tested on the instances of the 2007 International Timetabling Competition. These instances are based on real-life input from the University of Udine. The competition provided three sets of input: Early, Late and Hidden datasets. Due to space limitations, we will only present the first two of these datasets. The most important information about these can be seen in Table 1.

Table 1: Input characteristics

Inst.	Day	Slot	Room	Course	Curr.	Unav.
Early Datasets						
Comp01	5	6	6	30	14	53
Comp02	5	5	16	82	70	513
Comp03	5	5	16	72	68	382
Comp04	5	5	18	79	57	396
Comp05	6	6	9	54	139	771
Comp06	5	5	18	108	70	632
Comp07	5	5	20	131	77	667
Late Datasets						
Comp08	5	5	18	86	61	478
Comp09	5	5	18	76	75	405
Comp10	5	5	18	115	67	694
Comp11	5	9	5	30	13	94
Comp12	6	6	11	88	150	1368
Comp13	5	5	19	82	66	468
Comp14	5	5	17	85	60	486

For each instance, the table gives the number of days, timeslots per day, rooms, courses, curricula and unavailability constraints for the teachers. The following penalty values were used for the soft constraints:

- *Compactness*: Every isolated course is worth 2 ponts.

- *Roomy capacity*: The capacity of a room can be violated, but every student above the capacity is worth 1 point.
- *Room stability*: If lectures of a course are scheduled into more than one room, then the penalty for each room beyond the first is 1 point.
- *Minimum number of days*: If a course is scheduled on less days than the minimum required number, 5 penalty points are given for missing each day.

Test results of the algorithm are presented in Table 2. The algorithm was executed ten times for each instance, and the rounded average of these solutions is give by the table. The destructive search ran for 30 steps in each iteration.

Table 2: Test results

Instance	Runningtime	Penalty
Early Datasets		
Comp01	32s	51
Comp02	16M26s	328
Comp03	9M16s	314
Comp04	8M25s	348
Comp05	7M32s	423
Comp06	14M54s	503
Comp07	15M46s	592
Late Datasets		
Comp08	10M31s	361
Comp09	12M3s	285
Comp10	16M17s	536
Comp11	43s	37
Comp12	11M4s	525
Comp13	9M42s	331
Comp14	7M32s	434

The table presents the total running time of both stages for each instance, as well as the total penalty of the best achieved solution. It can be seen from the results that while the running times are acceptable, the penalties in some cases are a bit high. This means that there is still room for the improvement of the algorithm.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we examined the curriculum-based university course timetabling problem, and developed a two-stage heuristic algorithm for its solution. This algorithm uses a greedy recursive approach to construct an initial solution, then increases its quality with the help of a local search method. While the local search itself is not fully a tabu search algorithm, it utilizes a tabu list to store forbidden transformations. A destructive step is also executed to escape local optima at the end of each iteration.

The algorithm was tested on the datasets of the 2007 International Timetabling Competition. While feasible solutions were found for all instances, both their running time and quality can be improved. We would like to implement faster approaches for creating the initial solution, as well as implementing a proper tabu search algorithm instead of the current local search.

6. ACKNOWLEDGMENTS

Máté Pintér was supported by "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3- VEKOP-16-2017-0002 (supported by the European Union and co-funded by the European Social Fund). Balázs Dávid acknowledges the European Commission for funding the InnoRenew CoE project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European regional Development Fund), and is thankful for the support of the National Research, Development and Innovation Office - NKFIH Fund No. SNN-117879.

7. REFERENCES

[1] E. A. Akkoyunlu. A linear algorithm for computing the optimum university timetable*. *The Computer Journal*, 16(4):347–350, 1973.
[2] H. Babaei, J. Karimpour, and A. Hadidi. A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43–59, 2015.
[3] R. Bellio, S. Ceschia, L. D. Gaspero, A. Schaerf, and T. Urli. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *ArXiv*, abs/1409.7186, 2014.
[4] A. Bettinelli, V. Cacchiani, R. Roberti, and P. Toth. An overview of curriculum-based course timetabling. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 23(2):313–349, 2015.
[5] E. Burke, Y. Bykov, J. Newall, and S. Petrović. A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research*, 13(2):139–151, 2003.
[6] I. T. Competition. International timetabling competition 2007. http://www.cs.qub.ac.uk/itc2007/index.htm, 2007. Accessed: 2019-07-30.
[7] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86 – 92, 1993.
[8] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science*, pages 184–193, 1975.
[9] G. Lach and M. E. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194(1):255–272, 2012.
[10] Z. Lü and J.-K. Hao. Adaptive tabu search for course timetabling. *European Journal of Operational Research*, 200(1):235–244, 2010.
[11] T. Müller. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, 172(1):429, 2009.
[12] K. Shaker, S. Abdullah, A. Alqudsi, and H. Jalab. Hybridizing meta-heuristics approaches for solving university course timetabling problems. In P. Lingras, M. Wolski, C. Cornelis, S. Mitra, and P. Wasilewski, editors, *Rough Sets and Knowledge Technology*, pages 374–384. Springer Berlin Heidelberg, 2013.

StuCoSReC

University of Primorska Press
www.hippocampus.si
ISBN 978-961-7055-82-5