Performance Comparison of Sequential and Cooperative Integer Programming Search Methodologies in Solving Curriculum-Based University Course Timetabling Problems (CB-UCT)

Mansour Hassani Abdalla, Joe Henry Obit, Rayner Alfred and Jetol Bolongkikit

Knowledge Technology Research Unit, Universiti Malaysia Sabah,
88400 Kota Kinabalu, Malaysia
mansourabdalla22@gmail.com, joehenry@ums.edu.my,
ralfred@ums.edu.my, jetol@ums.edu.my

Abstract. The current study presents Integer Programming (IP) search methodology approaches for solving Curriculum-Based University Course Timetabling problem (CB-UCT) on real-life problem instances. The problem is applied in University Malaysia Sabah, Labuan International Campus Labuan (UMSLIC). This research involves implementing pure 0-1 IP and further incorporates IP into a distributed Multi-Agent System (MAS) in which a central agent coordinates various cooperative IP agents by sharing the best part of the solutions and direct the IP agents towards more promising search space and hence improve a common global list of the solutions. The objectives are to find applicable solutions and compare the performance of sequential and cooperative IP search methodology implementations for solving real-life CB-UCT in UMSLIC. The results demonstrate both sequential and parallel implementation search methodologies are able to generate and improve the solutions impressively, however, the results clearly show that cooperative search that combines the strength of integer programming outperforms the performance of a standalone counterpart in UMSLIC instances.

Keywords: Timetabling, Integer Programming, Multi-Agent System.

1 Introduction

Timetabling is one of the problems on which so many researches have been done over the years and CB-UCT is an NP-hard and also highly constrained combinatorial problems. This is because specific circumstances give origin to several problems, with an abundance of varying features (constraints) [1]. Timetabling problems are very hard to solve because of these arising problems (varying constraints in every semester). Replicating previous timetable and manually trying to fix the new problem does not solve the problem. In fact, it becomes a burden to academic departments who are involved in timetable generation in every semester.

Timetabling involves two categories of constraints, hard and soft constraints. Hard constraints are mandatory to be satisfied for the timetable to be considered feasible all hard constraints must be fully satisfied, however, soft constraints are not only desirable but the more soft constraints are solved the higher the quality of the solution. In particular soft constraints are used to measure the quality of the timetables. Each institution has their own constraints, some constraints may be considered hard in some institution while some constraints are considered soft in other institution. In addition, the constraints vary from time to time. Additionally, according to [3] modularity is other features which contribute to the hardness of the problems i.e. students are allowed to choose the course from other departments or even from another faculty. Hence to solve all these problems an effective research and search methodology is highly required in this particular domain. In fact, there are so many different techniques proposed in literature such as cooperative search inspired by particle swarm optimisation [2], parallel meta-heuristics [16], parallel local search [2], Parallel Constraint Programming [5] and many more. In recent years scholars acknowledge parallel search as a natural effective solution approach to timetabling problems [8]. However, the questions are what is the best parallel strategy? Can these strategies improve the performance of stand-alone algorithms (i.e. IP, heuristics, and meta-heuristics e.c.t)? Is the proposed parallel IP able to improve the solutions as compared to standalone IP? In order to provide some understanding into these questions, we propose cooperative IP search methodologies to provide some insight into these questions.

In this research, we aim to investigate the performance of sequential and parallel IP for the CB-UCT in UMSLIC. In particular, the standalone sequential IP and parallel IP in which different improving IP agents are running concurrently in different simulated Multi-agent systems are implemented and tested on the real word problem instances. As [2] proposed cooperative search inspired by particle swarm optimisation, this research is inspired by three important objects. Firstly the availability of high performance computer which makes feasible for us to implement IP model which is proven in literature that it requires high performance machine [3], secondly the multi-processor computers and the rise of MAS which motivate parallel processing [4], and finally even though recently there a lot of research devoted to parallel search nonetheless there is little which have taken the advantage of the strength of IP into cooperative search methodology.

Hence the current work contributes to the body of knowledge hereby by proposing both sequential and cooperative integer programming for solving CB-UCT on UMSLIC instances.

2 Related Work

Curriculum-based University Course Timetabling problem (CB_UCT) is very important to research due to their direct importance and relevance in real-life situations [10]. According to [2, 3, 11] requirements differ from one institution to another for any given semester and in fact, according to [11] it is very difficult to produce the general methodology to solve all the problems in every institution. As highlighted by [12] "the problem becomes more complex if the events vary in duration, and each event must occupy only one room for the entirety of this duration". And so in UMSLIC the problems become complex since the duration of each course are not same as some courses take duration of two hours i.e. main courses, some take three hours i.e. language courses.

Most literature proposes purely heuristic, meta-heuristics [13, 14] and hyper-heuristics solution methods [12]. However, in recent years, integer programming (IP) methods have been the subject of increased attention [12] because of the availability of powerful computers, proven success strength of the IP, and ability to solve large instance in a small amount of time. In addition, cooperative search (Multi-Agent Systems) appears to attract scholars from both artificial intelligence and operational research community [6]. This is because in multi-agent systems-based approaches, intensification and diversification are possible to achieved through agents' communication and cooperation [2], negotiation of agents to remove the constraints of the event and ability to resources sharing with each other [6], and finally the tendency of guiding algorithms towards more promising search space [3, 4, 8].

However, it is worth noting that, approaches based on operational research do not have good efficiency in solving scheduling problems [6]. Somewhat, they do have easier implementation since they are mostly analyzed by software integrated with efficient and heuristic algorithms [6]. In recent years significant advancement of meta-heuristics in solving university timetabling problems and other complex combinatorial optimization problems have been achieved. These advancements have led to the successful deployment of meta-heuristics in the wide range of combinatorial problems. However, one major drawback of this family of techniques is the lack of robustness on a wide variety of problem instances [15]. Also, the computation times associated with the exploration of the solution space may be very large [8]. Moreover, [16] also emphasized the fact that, the performance of meta-heuristics often depends on the particular problem setting and data.

3 Problem Statement

In every semester, academic institutions are facing difficulties in constructing course timetable. The task is to allocate the set of courses offered by the university to a given set of time periods and available classrooms in such a way no curriculum, lecturer or classroom is used more than once. Essentially the problem in UMSLIC involves assigning a set of 35 timeslots (seven days, with five fixed timeslot per day) according to UMSLIC teaching guidelines.

Each lecturer teaching several courses in each semester and each course has at least one lecture of minimum two hours per week. In addition, UMSLIC's administration has a guideline for the compulsory, elective, center for promotion of knowledge and language learning (PPIB), and center for co-curriculum and student development (PKPP) courses to be enrolled by the students in each of the semesters throughout the students' university days Our approach will also fulfill university teaching guideline where there are some general preferences such as some courses particularly program and faculty courses cannot be scheduled on weekends and must be scheduled on the first or third timeslots of the weekdays. In addition, some courses such as PKPP courses cannot take place on weekdays. In addition, some course such as PPIB courses must be scheduled on second, fourth, or fifth timeslot. Hence, this research concentrates on real-life CB-UTT. In fact, in CB-UTT there are five variables identified namely periods, courses, lecturers, rooms, and curricula. The objective is to assign a period and a room to all lectures of each course according to the hard and soft constraints based on UMSLIC teaching guidelines.

4 Sequential IP

The proposed sequential IP is formulated to solve the problem in two stages; in the first stage, the model solves hard constraints. In the second stage, the model tries to deal with soft constraints and maintain the feasibility of the solution. The objective is to generate feasible timetable solution that is able to satisfy all the people affected by the timetable [7].

In particulars, in the first stage, the IP formulation tackle the hard constraints while in the second stage the timetable is being improved by minimizing the soft constraints as much as possible. Firstly the proposed search IP starts with an empty timetable. Since at the start all the problem instances are already pushed to the hash set (list) from the source pre-processed text file and all the information are now in places, the search IP generates room, day, period, and course at random. Then check for feasibility i.e. the room capacity can accommodate the number of students registered in that particular course, the period if is already occupied on that particular day, and if the course is already scheduled. If all the conditions are feasible, then course will be inserted into timetable and the room is registered as already used at that period in a given day. The course is registered as already scheduled, the timeslot is registered as already used and the number of unscheduled course is decremented. The process repeats until all the courses are feasibly scheduled.

First stage algorithm

```
01:  while all courses is not scheduled do
02:      Select randomly $d \in D$, $r \in R$, $p \in P$, $c \in C$ from the problem instance
03:  if $c \in C$ feasibly can be scheduled i.e. no hard constraint violation then
04:      Insert into timetable and update the number of course scheduled
06:      iter++;
07:  else
08:      Remove $c \in C$ from timetable and update number of course scheduled
09:  end if
10:  end while
```

In the second stage, the simple local search is introduced. The local search gradually tries to improve the quality of the solution generated in the first stage with the knowledge of maintaining the feasibility of the solution. The search is basically based on swapping of events as explained hereby. In this stage, there is two moves, the first move the course is selected randomly and places it into feasible timeslot and room which are also selected at random. In the second move the event is selected at random and insert into empty timeslot. If the course is inserted and the cost factor is improved or even similar to previous cost value the timetable is updated however if the course is inserted but the cost factor is not improved the timetable is not updated. The process keeps repeating until the stopping condition is met which is 300 seconds in this case. The time given is only five minutes; however, the larger the problem instance the higher the time is required [7] and the better the solution.

Second stage algorithm

```
01:  Stop condition: is set 300 seconds
02:  Best solution = initial solution
03:      while stop condition is not met do
04:  Select two events (d∈D, r∈R, p∈P,  c∈C) AND (d'∈D',r'∈R', p'∈P',  c'∈C')
     randomly from the feasible solution  and swaps  them: new solution S*, OR
     Select Event randomly from feasible timetable and insert in to empty slot: new
     solution S*
06:      if cost Function(S*) < cost Function (best solution
07:  Best solution = S*
08:  iter++;
09:   else
10:  Best solution = Best solution
11:   end if
12:   end while
```

5 Cooperative IP

Figure 1 presents the proposed agent-based IP searches framework. In this research, a decentralized agent-based framework, which consist of given number of agents (*n*) is proposed. Basically this framework is a generic communication protocol for IP search methodology to share solutions among each other. Each IP is an autonomous agent with its own representation of the search environment. All IP agents at the beginning share the same complete feasible solution and then starts with their own search towards more promising search space. Moreover, the communication or ability of the agent to exchange the solutions with one another via the central agents prevent individual agent from trapping on the local optima [8]. Essentially all agents in the distributed environment communicate asynchronously via the central agent. Additionally, it is worth mentioning that, the initial feasible solution is generated by the central agents as well. In clarity this framework will involve asynchronous cooperative communication as follow.

5.1 Central Agent (CA)

The central agent is responsible to generate the initial feasible solution as well as to coordinates the communication process of all other agents involved in the proposed framework. The central agent acts as intermediate agent among the IP agents where it passes the feasible solution and other parameters to the IP agents asynchronously on top of FIPA-ACL communication protocol to improve the solutions. On top of that, the central agent receives the improved solutions from the IP agents and compares the objective function cost value of the received solution with the global solutions on the list, if the improved solution's objective is better than any of the solutions on the global solutions then the worse in the list is replaced. Else the received solution is discarded and the central agent randomly select other solution from the list of the global solutions and send back to that particular agent This procedure continues until the stopping condition is met.

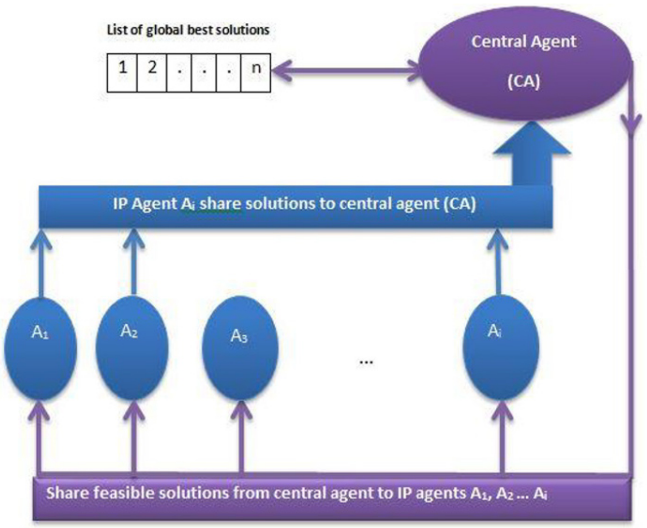
List of global best solutions
1 2 . . . n
Central Agent (CA)
IP Agent Ai share solutions to central agent (CA)
A1
A2
A3
...
Ai
Share feasible solutions from central agent to IP agents A1, A2 ... Ai

Fig. 1. Proposed Agent-based IP Search methodology Framework

5.2 IP Agents (A_i)

All other agents' start from the complete solution received randomly from central agent and iteratively perform search to improve the solution autonomously. In this case the agents have to maintain the feasibility of the solution i.e. do not violent hard constraint. After certain number of iterations according to the rules stated (after every

30 seconds) the agent passes the solution back to the central agent and request new solution from the central agent. The central agent accepts the solution if only the solution is better to the existing global solutions in the list of the solutions (i.e. the solution objective cost is less than the existing global solutions in the list) else the solution is discarded. If the solution is accepted then the solution with higher objective cost function i.e. worse in the list will be replaced. The reason an improving agent's exchange solution is to make sure the agents are not stuck on local optima, moreover scholars highlighted in the literature that, by exchanging the solution the possibility of the agents (algorithms) changing the position towards more promising space is increased [4, 8, 9].

Best solution Criteria. All of our agents are incorporated with integer programming search methodology. Each agent also is capable to compute the final objective function and return it along with the improved solution. The central agent places all the solutions obtained in a sorted list where the solution on top will be the best solution (the solution with minimum objective function value).

In this framework the value of the objective functions is used to determine the quality of the solution. The lower the cost value the better the solution. Hence for the solution which has improved by the IP agents to be considered better to the global existing solutions, the returned improved solution's objective function should be lower to the one of the available in the global solutions objective functions values. Else the solution is discarded.

6 Experimental Setup and Results

In order to evaluate the performance of the proposed agent-based framework compare to stand alone sequential integer programming we have conducted experiments hereby. The experiments have been carried out using real-life instances on UMSLIC semester one 2016/2017 and semester two 2016/2017. Table 1 gives an overview of the instance characteristics.

Table 1. Summary of the dataset from UMSICL academic division

	Semester1 s2016/2017	Semester2 S2016/2017
Number of student	2263	2224
Number of curriculum	65	49
Number of lectures	108	92
Number of courses	134	117
Cumulative number of constraints	4126	2918

The numbers of unavailability (constraints) in each semester greatly differs between the instances. For example, semester one session 2016/2017 there are 4126 sum of all the hard constraints identified while in second-semester session 2016/2017 there are 2918 sum of constraints identified. It should be noted that the constraints mentioned here refers to the total number of hard constraints for all the courses offered in that particular semester. It can be seen that even it is the dataset from the same university but the number of constraints is not the same for particular two semesters. And this is demonstrated why it is very difficult for the university to duplicate the previous timetables since in every semester the constraints are not the same.

The IP agents implemented in the framework to solve CB-UCT are described in section 4. The central agent reads in the problems and generates initial feasible solutions. The central agent then sends the complete feasible solutions to IP agents to improve the solution. When the search is complete (complete n search) the central agent receives the results from the improving agents and insert into the sorted list of size *n*. However, after the list is already with n different solutions, whenever the central agents receive the new solution from the IP agents it will compare its objectives with the existing solutions in the list as explained in section 5 of this paper.

As described in research objectives the tests are designed to compare different groups of IP agents with their Stand-Alone (SA-IP) counterparts. For each scenario, the experiments were conducted over 50 runs for each problem instance and the average objective values were computed. Basically, we conducted 50 runs for each problem instances because we wanted to find the upper and lower bound and hence the overall consistence of the algorithms. The agents conducted only 30 messages to complete each search taking no longer than five minutes to complete whole experiments. The number of conversations 30 is chosen because experimentation shows that the rate of solution improvement is reduced after that number. The 30 conversations last no longer than about five minutes and this is deemed to be a good stopping condition. The results are shown in table 2.

Table 2. Experimental results for the proposed Standalone IP (SA-IP) and Cooperative IP search.

	Number of agents	Semester1 s2016/2017	Semester2 s2016/2017
Initial cost	0-1 IP	368.04	377.29
Final Average cost	SA-IP	326.94	343.69
Final average cost	3	321.20	338.80
Final average cost	6	302.20	318.50
Average improvements (%)	SA-IP	10.99	8.91
Average improvements (%)	3	12.73	10.20
Average improvements (%)	6	17.89	15.58

The improvement from the initial to final cost value for the Standalone IP (SA-IP) is 10.99 and 8.91 for s1 2016/2017 and s2 2016/2017 respectively. Also when three IP agents (Ai) are used is 12.73% and 10.20% for s1 2016/2017 and s2 2016/2017 respectively. On the other hand, the improvement of the solution's cost value when six IP agents (Ai) is used is 17.89% and 15.58% % for s1 2016/2017 and s2 2016/2017 respectively.

The results presented clearly demonstrate that cooperative search outperforms standalone IP in this context. The IP agents improve solution as compared to standalone IP. In addition it worth to note that there is huge possibility that, the solution can be improved even further. Because in fact in this experiments it is not purely parallel as this is just simulation of parallel computing hence the performance might increase more as if each agent run on the self-machines.

The main benefits of the agent-based approach adopted for CB-UCT in UMSLIC are the possibilities of intensifying and diversifying the search space, where IP agents (*Ai)* are able to changes solutions with each other in the distributed MAS [2]. This leads the improving agents to easily move towards the most promising search areas of the search space [6]. Basically, by the analysis, the results, the numbers of IP agents used in the framework determine the quality of the solution generated. In this regard, we find that the quality of the solution in this framework proves to increase slightly as the number of IP agents (Ai) is increased.

7 Conclusion

In this research, we have conducted a comprehensive study of sequential IP search methodology approach in solving CB-UCT. In addition, we have focused on methods based on a cooperative search by incorporating sequential IP into agent-based multi-agent systems. In the current study, we have demonstrated on how IP can be integrated into MAS in order to conduct the cooperative search in solving the CB-UCT. To prove this hypothesis, we have justified the capabilities of MAS and how cooperative search can be natural approaches to solve the problem and find higher quality solutions as compared to the standalone sequential counterpart.

The advantages of using MAS in CB-UCT as compared to standalone IP in this context is the ability for the IP agents to share the best part of the solutions and the possibility of the agent moving towards more promising search space.

In general, cooperative outperform standalone IP can be attributed to the fact in standalone IP once the algorithms stuck on local optima it cannot improve solution anymore however in cooperative search once agent stack on local optima agent can changes solutions and through that, the agent is able to escape from local optima.

References

1. Landir S, Maristela O.S., Alysson M.C.: Parallel local search algorithms for high school timetabling problems, European Journal of Operational Research,Volume 265, Issue 1, 2018, Pages 81-98, ISSN 0377-2217,
2. Obit, J. H., Alfred. R., Abdalla, M.H.: A PSO Inspired Asynchronous Cooperative Distributed Hyper-Heuristic for Course Timetabling Problems. Advanced Science Letters, (2017)11016-11022(7)
3. Obit. J. H., Ouelhadj, D., Landa-Silva, D., Vun, T. K.., Alfred, R.: Designing a multi-agent approach system for distributed course timetabling. IEEE Hybrid Intelligent Systems (HIS), 10.1109/HIS(2011)-6122088.

4. Lach, G., & Lübbecke, M. E. (2012).: Curriculum based course timetabling: new solutions to Udine benchmark instances. Annals of Operations Research, 194(1), 255-272
5. Regin JC, Malapert A.: Parallel Constraint Programming. (2018). springerprofessional.de. Retrieved 17 April 2018.
6. Babaei, H., Hadidi, A A.: Review of Distributed Multi-Agent Systems Approach to Solve University Course Timetabling Problem. Advances In Computer Science : An International Journal, 3(5), 19-28. (2014).
7. Lach, G., & Lübbecke, M. E.: Curriculum based course timetabling: new solutions to Udine benchmark instances. Annals of Operations Research, 194(1), 255-272 (2012).
8. Cung, V.-D., Martins, S. L., Ribeiro, C. C., Roucairol, C.: Strategies for the parallel implementation of metaheuristics. In Essays and surveys in metaheuristics (pp. 263-308): Springer (2002).
9. Obit, J. H.: Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems. Ph.D. Thesis, School of Computer Science University of Nottingham (2010)
10. Babaei, H., Karimpour, J., & Hadidi, A. (2015).: A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86, 43-59. doi:10.1016/j.cie.2014.11.010
11. Obit. J.H., Yik. J. K., Alfred. R.: Performance Comparison of Linear and Non-Linear Great Deluge Algo...: Ingenta Connect. (2018). Ingentaconnect.com. Retrieved 17 April 2018, from http://www.ingentaconnect.com/content/asp/asl/2017/00000023/00000011/art00129
12. Antony E.P, Hamish W., Matthias E., David M.R, Integer programming methods for large-scale practical classroom assignment problems. (2015). Computers & Operations
13. Yik , J. K., Obit, J. H., Alfred, R.: Comparison of Simulated Annealing and Great Deluge Algorithms for...: Ingenta Connect. (2018). Ingentaconnect.com. Retrieved 17 April 2018,
14. Norgren, E., Jonasson, J.: Investigating a Genetic Algorithm-Simulated Annealing Hybrid Applied to University Course Timetabling Problem: A Comparative Study Between Simulated Annealing Initialized with Genetic Algorithm, Genetic Algorithm and Simulated Annealing. DIVA. Retrieved 17 April 2018,
15. Di Gaspero, L., McCollum, B., Schaerf, A.: The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3).
16. Crainic, T. G., Toulouse, M.: Parallel strategies for meta-heuristics. In Handbook of metaheuristics (pp. 475-513(2003)): Springer.