A Timetabling Applied Case Solved with Ant Colony Optimization

Broderick Crawford[1,2,3], Ricardo Soto[1,4,5], Franklin Johnson[1,6], and Fernando Paredes[7]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{broderick.crawford,ricardo.soto}@ucv.cl
[2] Universidad Finis Terrae, Santiago, Chile
[3] Universidad San Sebastián, Santiago, Chile
[4] Universidad Autónoma de Chile, Santiago, Chile
[5] Universidad Central de Chile, Santiago, Chile
[6] Universidad de Playa Ancha, Valparíso, Chile
franklin.johnson@upla.cl
[7] Escuela de Ingeniería Industrial, Universidad Diego Portales, Santiago, Chile
fernando.paredes@udp.cl

Abstract. This research present an applied case of the resolution of a timetabling problem called the University course Timetabling problem (UCTP), the resolution technique used is based in Ant Colony Optimization metaheuristic. Ant Colony Optimization is a Swarm Intelligence technique which inspired from the foraging behavior of real ant colonies. We propose a framework to solve the University course Timetabling problem effectively. We show the problem and the resolution design using this framework. First we tested our proposal with some competition instances, and then compare our results with other techniques. The results show that our proposal is feasible and competitive with other techniques. To evaluate this framework in practice way, we build a real instance using the case of the school of Computer Science Engineering of the Pontifical Catholic University of Valparaíso and the Department of Computer Engineering at Playa Ancha University.

Keywords: Ant Colony Optimization, Swarm Intelligence, University Course Timetabling Problem.

1 Introduction

The timetabling problems are commonly faced by many institutions as schools and universities. The basic problem is defined as a set of events that must be assigned to a set of timeslot of a way that all the students can attend to all of their respective events. With the reservation of which hard constraints necessarily must be satisfied, and soft constraints that deteriorate the quality of the generated timetabling. Of course, the difficulty can vary in any particular case of the UCTP. [10]. The problem difficulty depends on many factors and in addition the assignment of rooms makes the problem more difficult. Many techniques have

R. Silhavy et al. (eds.), *Artificial Intelligence Perspectives and Applications*,
Advances in Intelligent Systems and Computing 347, DOI: 10.1007/978-3-319-18476-0_27

been used in the resolution of this problem. We can find evolutionary algorithms, tabu-search, constraint programming and genetic algorithms [2]. We present the resolution using an Ant Colony Optimization (ACO) algorithm through the implementation of Hypercube framework (HC). ACO is a Swarm Intelligence technique which inspired from the foraging behavior of real ant colonies. The artificial ants seek the solutions according to a constructive procedure as described in [9]. This ACO exploits an optimization mechanism for solving discrete optimization problems in various engineering domain [8]. We establish the representation for the problem to be solved with ACO, generating an appropriate construction graph and the respective pheromone matrix associated. In the following sections we present the UCTP problem, and the ACO Metaheuristic. Later we present the experimental results. Finally the conclusions of the work appear.

2 University Course Timetabling Problem

The UCTP is an adaptation of an original timetabling problem presented initially by Paechter in [13,12]. It consists of a set of events E and must be scheduled in a set of timeslots $T = \{t_1, ..., t_k\}$ ($k = 45$, they correspond to 5 days of 9 hours each), a set of rooms R in which the events will have effect, a set of students S who attend the events, and a set of features F required by the events and satisfied by the rooms. Each student attends a number of events and each room has a maximum capacity.

We present below a mathematical formulation of the problem. The simplest formula for this problem can be described as a problem of binary integer programming numbers in which the variable $X_{ij} = 1$ if course i is assigned to the classroom j is equal to 0 otherwise . The time in which a classroom can take in a day is divided into k periods:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \tag{2}$$

$$\sum_{i\in P_k}^{n} x_{ij} \leq 1, \forall i, \forall k \tag{3}$$

$$x_{ij} \in \{0,1\} \forall i \forall j \tag{4}$$

where P_k is the set of all courses offered in the period k, and C_{ij} is the cost of assigning the course i to the classroom j. The first constraint ensures that each course is assigned to one classroom. The second in a given period, in most courses offered i during the period k is assigned to each classroom. Implicitly each course is assigned to an only classroom.

A feasible timetable is one in which all the events have been assigned to a timeslot and a room so that the following hard constraints **H[1-3]** and soft constraints **S[1-3]** are satisfied.

- **H1:** No student attends more than one event at the same time.
- **H2:** The rooms must be sufficiently great for all students who attend a class and to satisfy all the features required by the event.
- **H3:** Only one event per each room at any timeslot.

In addition, all possible generated timetables are penalized for the number of soft constraint violated. these constrint appear next:

- **S1:** A student has a class in the last slot of the day.
- **S2:** A student has more than two classes in a row.
- **S3:** A student has exactly one class on a day.

Feasible solutions are always considered to be superior to infeasible solutions, independently of the numbers of soft constraint violations (SCV). One feasible solution is better than another, if it minimizes the SCV.

3 Framework for UCTP

According to the constraints presented in the previous section and the characteristics of the problem, we can now consider the option to design an effective scheme for the UCTP. We have to decide how to transform the assignment problem (to assign events to timeslots) into an optimal path problem which the ants can solve [3] and then optimize the problem.

We propose the following: an instance of the problem is received as input, then it assigned events to a timeslot, later a matching algorithm [14,11] is used for makes the assignation from rooms to each one of events associated to timeslot. In this point a solution is complete, but a low quality one. Then a local search algorithm [4] is applied that improves the quality of the solution and gives as final optimized result

3.1 Using Max-Min Ant System

Ant colony optimization is a metaheuristic algorithm based on a graph representation in which a colony of artificial ants cooperate in finding good solutions to discrete optimization problems [9]. The ants travel through the construction graph starting from an initial point and selecting the nodes which travel according to a probability function that is given by the pheromone and heuristic information of the problems.

We choose the Max-Min Ant System (MMAS) algorithm to solve the UCTP. The Max-Min Ant System is one of the best performing ACO algorithms [16]. MMAS can easily be extended by adding local search algorithms.

Contruction Graph: One of the main elements of the ACO metaheuristic is to model the problem on a construction graph [7,5], that way a trajectory

through the graph which represents a problem solution. In this formulation of the UCTP is required to assign each one of $|E|$ events to $|T|$ timeslots. Where direct representations of the construction graph is given by $E \times T$; given this graph we can then establish that the ants travel throughout a list of events choosing timeslot for each event. The ants follow one list of events. For each event, the ants decide timeslot t, each event is a single time in a timeslot, thus in each step an ant chooses any possible transition as showed in figure 1.

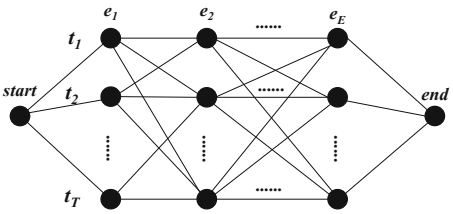

Fig. 1. Construction graph: For each event e, the ant chooses a timeslot t

Now we present the probabilistic function. This function adapting to MMAS according to HC and allows the ants travel through the construction graph selecting a path. We use the probability function, defined in [7]. This function directly depends on the pheromone information τ, and the importance is determinate for the parameter α, and the heuristic information η is determinate for β, for the possible path for a k ant.

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta}}{\sum_{l=0}^{T} [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta}}, j \in \{1, ..., T\} \tag{5}$$

The pheromone matrix represents the pheromones in the path where the ants travels, and indicates the absolute position where the events must be placed. With this representation the pheromone matrix the pheromone does not depend on the partial assignments.

Heuristic Information and Pheromone Update: We define as heuristic information a function that calculates a weighted sum of a set of the soft and hard constrains in each assignation. This function has a high computational cost [14]. In the hypercube framework the pheromone trails are forced to stay in the interval $[0, 1]$. We represent computationally the evaporation of pheromone and in addition the amount of pheromone in the best ant path through the graph, once is completed a tour. The pheromone update rule for MMAS to UCTP is as follows:

$$\tau_{ij} = \rho\tau_{ij} + (1 - \rho)\Delta\tau^{kbest} \tag{6}$$

where ρ is a rate of evaporation $\rho \in\ ]0, 1]$. And $\Delta\tau$ it is associated with quality of the current solution of ant $kbest$. We can use an updating pheromone strategy considering the quality of timetabling solution:

$$\Delta\tau^{kbest} = \frac{1}{fitnessfunction(kbest)} \tag{7}$$

where $fitnessfunction$ determines the quality of the solution of $kbest$ ant, according to the SCV.

3.2 UCTP-MMAS Algorithm

The general structure of the algorithm is presented, in which some modifications added to the ones presented in [14,16]. A new assignation values to τ_{max},τ_{min}, a new pheromone update rules.

Algorithm 1. UCTP-MMAS Algorithm

```
1: Input: Problem Instance I
2: initialize feromone values τ_max and τ_min
3: calculate dependencies between events e_i, i ∈ {1, ..., E}
4: sort Events according dependencies
5: repeat
6:    for a = 1 to m do
7:       construction process of ant a
8:       for i = 1 to |E| do
9:          chooser timeslots t_j according to probabilities p_{i,j} for de event e_i
10:         storing partial route for k_a ant
11:      end for
12:      apply matching algorithm for assign rooms
13:      select the best solution for iteration
14:   end for
15:   applying local serach to best solution according to the fitness
16:   select the best global solution
17:   apply pheromone update for k_best ant
18: until the termination condition is satisfied (iterations or time)
19: Output: An optimized solution for I
```

Only the solution that causes the fewest number of hard constraint violations is selected for improvement by the Local Search. The pheromone matrix is updated only once by each iteration, and the global best (k_{best}) solution is used for the update. The description is illustrated in algorithm 1. This algorithm use it a matching function to associate events with rooms. To optimize the solution uses 1OPT, 2OPT and 3OPT Local Search algorithm [1].

4 Comparisons

The algorithm was implemented and submitted to a series of tests. The behavior of the proposed framework was observed in the resolution of the UCTP.

Instances of the UCTP are structured using a generator[1]. This generator allows generating classes of small, medium instances which reflect varied timetabling problems. In addition it was used a series of 20 instances created for International Timetabling Competition[2], these instances are made with the same generator. The instances with different parameters are presented in the following table.

Table 1. Parameter for the small and medium instances

Parameter	small	medium
Number of events	100	400
Number of rooms	5	10
Number of features	5	5
Features by room	3	3
Usage percentage	70	80
Number of students	80	200
Events Maximum for students	20	20
Maximum students per event	20	20

We firstly studied the best parameters configuration using the small instances. The UCTP-MMAS was tested without local search, making an evaluations with different ants numbers m and with different evaporations factors ρ, the parameters of $\alpha = 1$, number on attempts $= 10$ and a maximum time by attempt $= 90$ seconds for all the tests. The results are in table 2:

Table 2. Evaluation of parameters m and ρ using small1.tim

Parameter	m			ρ		
Values	5	10	20	0.2	0.5	0.8
SCV	17	16	16	15	13	17
Seconds	6.79	7.46	6.06	7.11	8.1	6.79

According to table 2, we observe that the best results are obtained using the parameter $m = 20$, obtaining an evaluation of 16 in 6.06 seconds. And for the case of evaporation factor, the best value is $\rho = 0{,}5$ in 8.1 seconds.

4.1 Comparative Results

Table 3 presents comparative results between the solutions obtained for different instances[3] the UCTP solved with different techniques such as Simulated annealing (SA), Advanced Search (AS) and Simulated Annealing with Local Search (SA-LS). These algorithms are compared according to SCV.

[1] `http://www.dcs.napier.ac.uk/~benp`
[2] `http://www.or.ms.unimelb.edu.au/timetabling`
[3] `http://www.idsia.ch/Files/ttcomp2002/`

Table 3. Number of SCV obtained with International Timetabling Competition instances

Algorithm	com01	com02	com03	com04	com05	com06	com07	com08	com09	com10
SA	45	25	65	115	102	13	44	29	17	61
AS	257	112	266	441	299	209	99	194	175	308
SA-LS	211	128	213	408	312	169	281	214	164	222
UCTP-MMAS	240	133	204	426	406	179	261	204	157	263
Algorithm	com11	com12	com13	com14	com15	com16	com17	com18	com19	com20
SA	44	107	78	52	24	22	86	31	44	7
AS	273	242	364	156	95	171	148	117	414	113
SA-LS	196	282	315	345	185	185	409	153	281	106
UCTP-MMAS	268	212	341	329	172	234	371	124	245	101

For these instances and compared with the other solutions, the UCTP-MMAS presents two characteristics to evaluate. First, it has the capacity to generate feasible solutions for these instances. These instances are difficult because they are from competitions Timetabling. Second, the quality of the generated solutions is very low compared with to Simulated Annealing, which has the best found historical results for these instances, but in comparison with the other instances it does not present great difference. It is not possible to decide if a technique is better than other, since the differences in results can be explained by different external agent.

Table 4 presents the comparison for the small and medium instances for the algorithm for UCTP with HC and local search (UCTP-MMAS) and MMAS pure (MMAS-p).

Table 4. It present the SCV obtained with small and medium instances for UCTP-MMAS and MMAS-p

Algorithm	small1	small2	small3	medium1	medium2
UCTP-MMAS	**0**	**4**	**1**	**138**	**186**
MMAS-p	3	6	3	152	250

We can observe for these instances that the UCTP-MMAS present a superiority in the quality of the generated solutions (smaller SCV). We can say the our proposed improves the quality of the ant algorithm applied. Table 5 presents the comparison with other ACO algorithm such as Ant Colony System algorithm of Krzysztof Socha (ACS) and to algorithm based on Random Restart Local Search (RRLS).

According to the results of UCTP-MMAS performs better than the other algorithms for small and medium instances, improving in all tested instances. only in the medium2 instance was surpassed by ACS.

Table 5. Results obtained with small and medium instances

Algorithm	small1	small2	small3	medium1	medium2
UCTP-MMAS	**0**	**4**	**1**	**138**	186
ACS	1	3	1	195	**184**
RRLS	11	8	11	199	202

4.2 Practical Case

To test this project on a practice way, we implement a the UCTP-MMAS with 2 real cases. We created an instance of the Pontificia Universidad Católica de Valparaíso (PUCV) and specifically for the school of Informatics Engineering. We implemented a tool in C language, to enter the courses, semester, assistants and assistantship, and indicate the times to the week that are dictated and his characteristics. In addition the number of rooms and their characteristics are entered to him. The system generates an instance introducing a factor of correlation between the events, generated an instance with the same format as competition instances. Stored this information, the algorithm is ready to be used. Table 6 presents the characteristics for the PUCV instance.

Table 6. Characteristics of UCV instance

Characteristic	value
Rooms and lab	9
Events	194
Total Attending	600
Features	5
maximum events by student	8
maximum students by event	20-45

Before using the instance it was necessary to correct some parameters of the algorithm implemented, since for the instance of PUCV the number of timeslot that they are used are 40 and not 45 like for other problems of the UCTP. In addition we create an adaptation to the soft constraint.

The instance was executed using a number of ants m= 20, evaporation factor ρ= 0.5. Time to local search 100 seconds, total time by reboots = 900 seconds, number of reboots = 10. The best solution was obtained approximately in 600 seconds with an evaluation of SCV = 0, which implies that the algorithm generated a complete timetable feasible and with the best possible quality.

For the quality of the obtained solution, it can be inferred that the generated instance that simulate the hour load of a semester of the school of computer science engineering had a low degree of correlation between courses of different semesters, thus a high performance solution was obtained.

We also implemented a real instance of Universidad de Playa Ancha (UPLA). This instance has 40 timeslots and their characteristics are presented in table 7:

Table 7. UPLA instance characteristic

Characteristic	value
Rooms and lab	22
Events	112
Total Attending	154
Features	5
maximum events by student	8
maximum students by event	40

The instance was executed using a number of ants m= 15, evaporation factor ρ= 0.01, total time by reboots = 600 seconds, number of reboots = 2. The best solution was obtained approximately to the 270 seconds with an evaluation of SCV = 0, which implies that the algorithm generated a feasible solution. This occurs because the instance is very simple, given the high number of rooms available, the few events to program and the low number of attendants.

5 Conclusion

In this research we have presented a formal model in order to apply the Hypercube framework to solve the University course timetabling problem (UCTP) making use of Max-Min Ant System, an efficient model was generated to solve instances of this problem creating good construction graph and a good pheromone matrix. We presented the test result made for the UCTP-MMAS. We observed that the UCTP-MMAS presented good results for instances of small and medium. Although the results were of low quality for the instances of the competition, we emphasize the fact that our approach always generates feasible solutions and for instances of normal difficulty have a good evaluation. We applied our algorithm to solve a real instance to the school of Computer Science of the PUCV and UPLA, for which created a feasible solution, this validates the use of a technique useful in real applications. As future work, we hope to improve the proposed algorithm and develop a suitable interface to apply the algorithm to other real instances and integrate the constraints of teachers in future instances. In addition we will try to integrate our algorithm with Autonomous Search [15,6].

Acknowledgments. Franklin Johnson is supported by Postgraduate Grant PUCV 2014, Broderick Crawford is supported by Grant CONICYT/FONDECYT/REGULAR/1140897, Ricardo Soto is supported by Grant CONICYT/FONDECYT/INICIACION/11130459, And Fernando Paredes is supported by Grant CONICYT/ FONDECYT/REGULAR/1130455

References

1. Abuhamdah, A., Ayob, M., Kendall, G., Sabar, N.: Population based local search for university course timetabling problems. Applied Intelligence, 1–10 (2013)
2. Babaei, H., Karimpour, J., Hadidi, A.: A survey of approaches for university course timetabling problem. Computers and Industrial Engineering (2014) (in press)
3. Blum, C., Dorigo, M.: Hc-aco: the hyper-cube framework for ant colony optimization. In: Proc. MIC 2001-Metaheuristics Int. Conf., pp. 399–403 (2001)
4. Blum, C., Dorigo, M.: The hyper-cube framework for ant colony optimization. Trans. Sys. Man Cyber. Part B 34(2), 1161–1172 (2004)
5. Blum, C., Roli, A., Dorigo, M.: Hc-aco: The hyper-cube framework for ant colony optimization. IEEE Transactions on Systems, Man, and Cybernetics–Part B, 399–403 (2001)
6. Crawford, B., Soto, R., Castro, C., Monfroy, E.: Extensible CP-based autonomous search. In: Stephanidis, C. (ed.) Posters, Part I, HCII 2011. CCIS, vol. 173, pp. 561–565. Springer, Heidelberg (2011)
7. Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: New Ideas in Optimization, pp. 11–32. McGraw-Hill Ltd., UK (1999)
8. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation (1997)
9. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, USA (2004)
10. ten Eikelder, H.M.M., Willemen, R.J.: Some complexity aspects of secondary school timetabling problems. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 18–27. Springer, Heidelberg (2001)
11. Johnson, F., Crawford, B., Palma, W.: Hypercube framework for ACO applied to timetabling. In: Bramer, M. (ed.) Artificial Intelligence in Theory and Practice. IFIP, vol. 217, pp. 237–246. Springer, Boston (2006)
12. Paechter, B.: Course timetabling evonet summer school (2001), `http://evonet.dcs.napier.ac.uk/summerschool2001/problems.html`
13. Paechter, B., Rankin, R.C., Cumming, A., Fogarty, T.C.: Timetabling the classes of an entire university with an evolutionary algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 865–874. Springer, Heidelberg (1998)
14. Socha, K., Knowles, J., Sampels, M.: A $\mathcal{MAX} - \mathcal{MIN}$ ant system for the university course timetabling problem. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) ANTS 2002. LNCS, vol. 2463, pp. 1–13. Springer, Heidelberg (2002)
15. Soto, R., Crawford, B., Monfroy, E., Bustos, V.: Using autonomous search for generating good enumeration strategy blends in constraint programming. In: Murgante, B., Gervasi, O., Misra, S., Nedjah, N., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2012, Part III. LNCS, vol. 7335, pp. 607–617. Springer, Heidelberg (2012)
16. Stützle, T., Hoos, H.H.: Max-min ant system. Future Gener. Comput. Syst. 16(9), 889–914 (2000)