Designing a Multi-agent Approach System for Distributed Course Timetabling

Joe Henry Obit
Labuan School of Informatics Science
Universiti Malaysia Sabah
Labuan, Malaysia
joehenry@ums.edu.my

Dario Landa-Silva
School of Computer Science
University of Nottingham
United Kingdom
dario.landasilva@nottingham.ac.uk

Djamila Ouelhadj
Department of Mathematics
Logistics and Mathematics Management Group
University of Portsmouth
Djamila.ouelhadj@port.ac.uk

Teong Khan Vun
Labuan School of Informatics Science
Universiti Malaysia Sabah
Labuan, Malaysia
nicholastkv@gmail.com

Rayner Alfred
School of Engineering and Information Technology
Universiti Malaysia Sabah
Kota Kinabalu, Malaysia
ralfred@ums.edu.my

***Abstract*—This paper proposes tackling the difficult course timetabling problem using a multi-agent approach. The proposed design seeks to deal with the problem using a distributed solution environment in which a mediator agent coordinates various timetabling agents that cooperate to improve a common global solution. Initial timetables provided to the multi-agent system are generated using several hybrid heuristics that combine graph colouring heuristics and local search in different ways. The hybrid heuristics are capable of generating feasible timetables for all instances of the two sets of benchmark problems used here. We discuss how these initialisation hybrid heuristics can be incorporated into the proposed multi-agent approach in order to conduct distributed timetabling. This preliminary work serves as a solid basis towards the design of an effective multi-agent distributed timetabling system.**

Keywords-component; course timetabling; multi-agent systems; heuristic local search; distributed timetabling.

I. Introduction

Timetabling is a type of assignment problem and usually each problem has unique characteristics and requirements. In the modern world it is crucial to avoid time and resources wastage, therefore the timetabling of activities requires that resources are in place at the right time and in the correct quantity in order to operate effectively and efficiently. An example of a timetabling problem which must be solved effectively is train timetabling. The construction of a train timetable must take into account where and when the train starts and ends its journey every day. Drivers' availability and preferences (based on seniority) also need to be considered together with the number of hours drivers are able to work per shift every week. The construction of bus timetables is a problem that shares many features with train timetabling. However, there are aspects in bus timetabling that do not normally arise in train timetabling, like dead mileage and lay-overs. In bus timetabling, dead mileage is the travelling distance from the last point of service to the bus depot while lay-overs are breaks given at the end of a trip before operating the reverse route begins. Effective bus and train timetables are crucial for the transport system in any city or country to function efficiently. Hence, the construction of quality timetables in different scenarios like transport is extremely important as it usually affects people daily lives. Examination timetabling is another example of an important timetabling problem. Here, each examination activity must be assigned to the right timeslot avoiding clashes between exams that have students in common. Once the clashing problem has been solved, the quality of the timetable is often improved by spreading the exams as much as possible in order to give students adequate time for revision before sitting the exam, or to have sufficient rest before starting the next exam. This short overview of some different types of timetabling problems highlights the importance of constructing good-quality timetables in different scenarios. Whilst the focus of this paper is on the university course timetabling problem, the same objectives are usually involved in many other forms of timetabling problems, where the aim is to ensure that the resources are allocated at the right time and in the right place, avoiding clashes. One objective commonly found in examination timetabling is to satisfy all people who are directly affected by the timetable, such as students and invigilators. Another common objective in educational timetabling is to control costs. For example, poor quality course timetabling can cause higher costs for educational institutions because students might not be able to attend all of their lessons if clashes exist. Then, students may need to take the course next term, extending their study time. From a lecturer's perspective, they might not be able to teach different courses if they have been timetabled into the same timeslot. Also, lecturers may be unable to deliver their lectures as planned if their courses have been assigned to rooms with unsuitable teaching equipment. In addition, poor quality timetables can also result in students having to attend more than two consecutive courses without a break, having a detrimental effect on student concentration.

In general, timetabling is a process of allocating, subject to constraints, activities in time and space, in such a way as to satisfy some objectives [15]. The construction of a course timetable to ensure all activities are in place accordingly, is a common problem for institutions of higher learning. Many factors contribute for the difficulty of course timetabling. For example, the requirements to satisfy hard (must be met) and soft (desirable to meet) constraints. Also, in some scenarios the timetabling process is distributed as each department in every faculty is responsible for generating their own timetable and not all information can be shared among departments. This makes constructing the overall timetable a huge challenge. Hence, while constructing one department's timetable, negotiation with other departments is needed in order to synchronize the allocation of shared resources. Failure to coordinate the sharing of timetabling resources will affect the quality of the overall timetable.

Burke and Newall [5] noted that advanced algorithms such as evolutionary methods might not be suitable to solve complex timetable problems especially when dealing with large instances. As a result, they proposed the decomposition method to break large instances into small sub-problems that algorithms are able to handle and then possibly to find an optimal solution. The process of decomposition to tackle timetabling problems has also been studied by Carter [12] who proposed to split large instances into problems small enough to be solved by local search algorithms.

A distributed multi-agent system is a network of agents that work together to solve problems that are beyond their individual capabilities [13]. Multi-agent systems have been applied to tackle problems in various application domains such as e-commerce and scheduling, producing good results. However, few researches have applied multi-agents to tackle educational timetabling problems. Some of these works are briefly discussed next.

Kapalsky et al. [6] tackled a real-world Distributed Timetabling Problem (DisTTP) using a multi-agent system paradigm. Each agent in their model has a different set of requirements to guide them in their search for the optimal solution. In order to coordinate their timetables, all agents in the distributed environment communicate and negotiate to avoid conflicts in the allocation of shared resources. Another example is the work by Di Gaspero et al. [10] who proposed an electronic marketplace called RoomSlot Market place (RSMP). In their approach, agents negotiate the room price assuming that room-slots can be sold and bought. The agents buy room-slots in the RSMP environment to improve their own timetable objective function value.

In this paper we propose the design of a multi-agent system to decompose large instances of the course timetabling problem so that the smaller problems can then be handled by different agents in the distributed environment. In our design, each department is represented by one agent called a Timetable Agent (TA). Each TA is responsible for solving different parts of a problem. A mediator agent (MA) coordinates all the TAs work to achieve the global solution.

In Section II, we describe the proposed design of the distributed timetabling system. We discuss how the ideas of global scheduling, local scheduling and negotiation among agents can be used to conduct distributed course timetabling. In Section III, we describe preliminary experiments in the first stage towards the multi-agent system while Section IV presents preliminary results. Lastly, Section V outlines conclusions and future work.

II. THE DISTRIBUTED TIMETABLING SYSTEM

We consider the course timetabling problem by Socha et al. [9] which includes hard (must be satisfied) and soft (desirable to satisfy) constraints. The hard constraints are: i) no student can be assigned to more that one course in the same timeslot, ii) a room should satisfy the features required by the course assigned to it, iii) the number of students attending the course should be less than or equal to the capacity of room, iv) no more than one course is allowed to be assigned to the same timeslot in each room. The soft constraints are: i) students should not have a single course on a day, ii) students should not have to attend more than two consecutive courses on a day, iii) students should not be asked to attend a course in the last timeslot of the day.

The proposed multi-agent system design is composed of two types of agents:

a) **Mediator Agent** (MA). This agent acts as an intermediary among the timetabling agents. Its main task is to coordinate the local timetable construction and avoid conflicts between timetable agents.

b) **Timetabling Agents** (TAs). The main task of each TA is to search for local improvements to the timetable from their own perspective. During the construction of the local timetable, each agent has to consider the constraints imposed by the mediator agent on the *Shared Courses* as well as their local constraints. Once there is an agreement on the shared courses, the TA's task is to improve the local solution. Each TA can have different objective function to guide the search for local improvements to the timetable.

The proposed distributed multi-agent course timetabling model consists of three main parts:

i. Global scheduling

ii. Local scheduling
iii. Negotiation among agents

A. *Global Scheduling*

In this stage the mediator agent generates the timetable for the courses shared by the different departments. The mediator agent and the timetabling agents cooperate in a distributed search process and could reach an agreement for changes to the shared course timeslots in order to improve the quality of the timetable.

B. *Local Scheduling*

The TAs generate their own initial solution by using a constructive heuristic (see Section III). In this stage, the TAs should avoid the violation of hard constraints but not worrying about the soft constraints. This means that the timetable generated in this stage is feasible for the relevant department but without yet taking into account the shared courses or soft constraints. After the initialization, the agents send their initial solution to the MA. Then, negotiation between the MA and TAs will take place to build the shared course timetable. After the shared course conflicts have been solved to a certain level of quality, the TAs apply a search approach like great evolutionary search [2] or great deluge local search [8] to improve the quality of the local solution.

C. *Negotiation Among Agents*

The MA waits until all TAs send their initial solutions. Upon receiving the local solutions, the MA checks the shared course timeslots. If there are too many conflicts, the MA sends a message to the relevant TA requesting for alternative shared course timeslots. The negotiation process continues until no more conflicts remain or until the violation of soft constraint cannot be improved further. During the negotiation process, the MA takes into account the soft constraints to control the allocation of shared courses. The soft constraints can be violated in a certain acceptable percentage or trajectory. If the TAs ignore the soft constraints it will affect the feasibility of their timetable. In order to satisfy the soft constraints, the MA sets a target for each TA with respect to the shared courses. For example, the local timetable for Computer Science department in shared courses with the Business and Economics department is feasible if Computer Science can register 7 out of each 10 students from Business and Economics and vice versa. Also in this stage, agents make requests and announcements to hopefully reach an agreement to make sure their timetable is consistent, feasible and of good quality. Fig. 1 shows the proposed distributed timetabling system design.

III. Hybrid Heuristic to construct timetable

This section gives the description of several effective hybrid algorithms that can be incorporated in the first stage towards the multi-agent system. For this, we employ the hybrid heuristics developed in our previous work [8] to become the timetabling agents within the multi-agent system. Previous experiments and results are also discussed here but not in the context of the distributed timetabling approach.

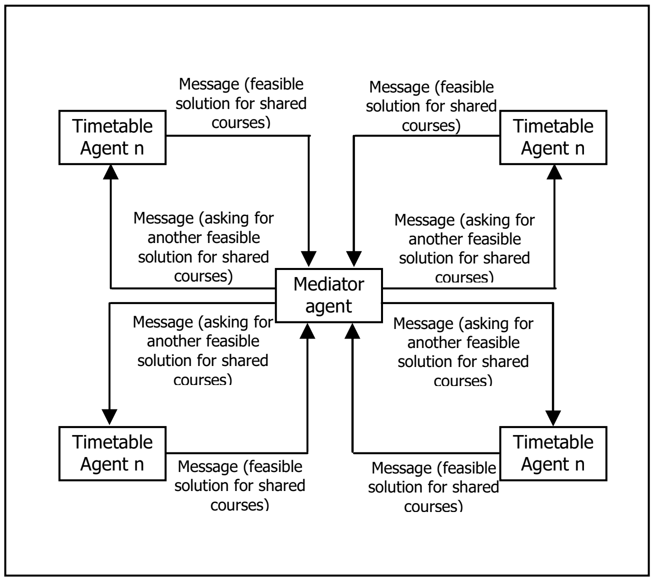
Timetable Agent n
Message (feasible solution for shared courses)
Message (feasible solution for shared courses)
Timetable Agent n
Message (asking for another feasible solution for shared courses)
Message (asking for another feasible solution for shared courses)
Mediator agent
Message (asking for another feasible solution for shared courses)
Message (asking for another feasible solution for shared courses)
Timetable Agent n
Message (feasible solution for shared courses)
Message (feasible solution for shared courses)
Timetable Agent n

Figure 1: Multi-agent distributed timetabling system design

We adopted the heuristic proposed by Chiarandini et al. [10] and added the Largest Degree (LD) Heuristic as described later. Largest Degree (LD) refers to the event with the largest number of conflicting events. In course timetabling problem, the conflicting events refer to events that have at least one student registered in common. This modification of the proposed heuristic was necessary to enable to generate feasible solutions especially for the large problem instance. This hybrid initialization heuristic works as follows.

A. *Largest Degree, Local Search and Tabu Search (IH1)*

Step One - Largest Degree Heuristic. In each iteration, the unassigned event with the largest number of conflicts (other events with students in common) is assigned to a timeslot selected at random without respecting the conflict between the events. After all events have been assigned into a timeslot, the algorithm then applies the maximum matching algorithm for bipartite graphs (see Chiarandini et al. [10]) to assign each event to a room. At the end of this procedure, there is no guarantee that the timetable is feasible. Step two and three as explained below then executed iteratively until a feasible solution is found.

Step Two - Local Search. In this step, the algorithm employs two neighborhood moves. Move one (M1) selects one event at random and assigns it to a feasible pair timeslot-room also chosen at random. Move two (M2) selects two events at random and swaps their timeslots and rooms while ensuring feasibility is maintained. Therefore we use these neighborhood moves M1 and M2 to improve the timetable generated in step one. A move is only accepted if it improves the satisfaction of hard constraints (as the moves always seek feasibility). This step terminates if after ten iterations no move has produced a better (closer to feasibility) solution. The algorithm terminates this step after ten iterations as we do not want to run it too long as it will extend the running time in finding the feasible solution. Moreover, the local search is meant to disturb the solution before the tabu search is executed.

Step Three - Tabu Search. We apply tabu search using only move M1 (select one event at random and assigns it to a feasible pair of timeslot-room also chosen at random). However, here move M1 is a little different than in step two, as the algorithm selects an event to move only if it violates hard constraints. Usually, at this stage, the violation of hard constraints is very low. Then, the algorithm only targets events that violate hard constraints instead of randomly rescheduling all events with the hope of selecting the appropriate timeslot for the right events. The tabu list contains events that were assigned less than *tl* iterations before calculated as $tl = rand(10) + \delta \times n_c$, where $rand(10)$ is a uniform random number between 0 and 10 inclusive, n_c is the number of events involved in hard constraint violations in the current timetable, and $\delta = 0.6$. In order to mitigate the power of tabu search, aspiration criterion is applied to accept when the best known assignment is found. This step terminates if after a fixed number of iterations no move has produced a better solution.

Algorithm 1: Initial Heuristic 1 (IH1)

```
Input: set of events in the poolOfUnscheduled events list E;
Sort the events in E by using Largest Degree (LD) heuristic;
while ( poolOfUnscheduled events list E is not empty ) do
   Select any timeslot t at random;
   Assign event e from E with largest degree (LD) first into t (tie
   break at random);
end
S = current solution;
loop = 0;
while (S is not feasible ) do
  if (loop < 10) then
    if ( coinflip()) then
      S* = M1(S); // apply M1 to S
    end
    else
      S* = M2(S); // apply M2 to S
    end
    if ( f(S*) · f(s)) then
      S Ã S* //accept new solution;
    end
 end
 else
    EHC = set of events that violate hard constraints;
    e = randomly selected member of EHC;
    S* = M2b(S, e); //Perform one iteration tabu search with move
M2b using e;
    if ( f(S*) < f(S) then
      S Ã S*; //accept new solution
    end
    if (loop == tsmax + 10 ) then
      loop = 0;
    end
  end
  loop++;
end
Output: S feasible solution (timetable)
```

B. Saturation Degree, Local Search and Tabu Search (IH2)

This algorithm starts by choosing a random event from the pool of unscheduled events and then it calculates the event's Saturation Degree (SD), which is the number of available resources (timeslots and rooms) to timetable that event without conflicts in the current partial solution. If there is still at least one available resource, assign a timeslot at random to the event and then apply maximum matching algorithm to assign a room. In the case of no resources left for the selected event, the algorithm selects any timeslot at random. Then, it moves all the events from that timeslot into the pool of rescheduled events and assigns the selected event into the now empty timeslot. Events in the pool of rescheduled events need to be rescheduled in any available timeslot, as long as there is available resource for the event. If there is no available resource, the algorithm removes the event to the pool of unscheduled events. In IH1 the assignment of events in step one is done without checking conflicts. Whereas, in IH2, the algorithm needs to check conflicts between the unassigned events first and then select a timeslot. If there are no conflicts, the unassigned event leaves the pool of unscheduled events. The process ends when all events from the pool of unassigned events and rescheduled events become empty. From the experiments we observed that when the whole process ends, the violation of hard constraints is usually very low. Therefore, to ensure feasibility, we then implement local search and tabu search.

C. Largest Degree, Saturation Degree, Local Search and Tabu Search (IH3)

Two well-known graph coloring heuristics are incorporated in this approach, namely, Largest Degree (LD) and Saturation Degree (SD). From the outset, the events in the pool of unscheduled events are sorted based on LD. After that, the algorithm chooses the event with the highest LD and calculates its SD. In this procedure, the initialization heuristic attempts to place all events into timeslots while avoiding conflicts. In order to do that, the heuristic uses the SD criterion and a list of rescheduled events to temporarily place conflicting events. The heuristic tries to do this for a given time, however, once that time has elapsed; all remaining unscheduled events are placed into random timeslots. In the second and third step, the heuristic uses subsequence simple local search and tabu search to achieve feasibility. The local search attempts to improve the solution but it also works as a disturbing operator. The tabu search uses move M1 only and is carried out for a fixed number of iterations.

D. Constraint Relaxation Approach (IH4)

In this final approach, we introduce extra timeslots to place events with zero SD. This initialization method works as follows. First, we sort the events in the pool of unscheduled events using LD. The event with the LD is chosen to be scheduled first. In the case that, there is no available resource for the chosen event (event with zero SD), the event will be distributed randomly into the extra timeslots. The number of extra timeslots needed is determined by the instance size. For example, in this experiments, we added ten extra timeslots for instances that $100 < |E| \leq 200$ and $100 < |S| \leq 200$ respectively, whereas, 15 extra timeslots added when instances having $200 < |E| \leq 400$ and $200 < |S| \leq 400$ respectively. By introducing extra timeslots the algorithm managed to find free-conflict timetables in short computational time and then the search can concentrate on satisfying the soft constraints by moving all events in the extra timeslots into the 45 valid timeslots. Once the algorithm managed to assign all events in the valid timeslots plus the extra timeslots without conflicts, the algorithm then perform great deluge [7] to reduce the number of timeslots down to 45 valid timeslots if necessary.

IV. PRELIMINARY RESULTS AND DISCUSSION

The proposed hybrid heuristic initialisation methods were applied to the Socha et al. [9] instances and also to the ITC 2002 instances [1]. We did not impose time limit as a stopping condition, each algorithm stops when it finds a feasible solution. All methods successfully generate initial solution for small instances in just few seconds. The medium and large Socha et al. instances are more difficult as well as all ITC 2002 instances. However, the proposed methods generated feasible solutions for all instances demonstrating that the hybridisation compensates weakness in one component with strengths in another one in order to produce feasible solutions in reasonable computation times.

Tables 1 and 2 compare the performance of each method on the Socha et al. and the ITC 2002 instances respectively. The first column in each table indicates the problem instance. The next four columns give the best objective function value (soft constraints violation) obtained by each heuristic. The last column in each table indicates the best computation time in seconds and the corresponding heuristic. The results show that none of the heuristics clearly outperforms the others in terms of the objective function value (soft constraints violation) obtained. Each of the four heuristics outperforms the other three in some of the problem instances. With respect to computation time we can see in Table 1 that for the Socha et al. problems, the heuristic that achieved the best objective value was almost never the fastest one (except in problem instance M2). However, for the ITC 2002 problems, we see in Table 2 that in several cases the heuristic producing the best objective value was also the fastest one. As indicated above, the hybrid initialisation heuristic (IH4) that uses extra timeslots to deal with conflicts and then great deluge as the local search to guide the solution to feasibility, is never the fastest approach. However, this heuristic IH4 was capable of producing the best solutions for two of the Socha et al. instances and six of the ITC 2002 instances.

In our preliminary experiments, we implemented a sequential heuristic (see [5, 9]) but were able to generate feasible timetables only for the small instances of the Socha et al. dataset (in fact, these small instances are considered to be easy). Even after considerably extending the computation time, the sequential heuristic was not able to generate feasible solutions for the medium and large Socha et al. instances or the ITC 2002 datasets.

TABLE I. RESULTS OBTAINED BY HYBRID INITIALISATION HEURISTIC ON 11 SOCHA ET AL. PROBLEM INSTANCES

Problem	IH1	IH2	IH3	IH4	Min Time
S1	173	198	207	200	0.077 (IH2)
S2	211	217	189	208	0.078 (IH2)
S3	176	190	188	209	0.062 (IH2)
S4	250	174	203	192	0.047 (IH1)
S5	229	238	226	217	0.078 (IH2)
M1	817	772	802	774	5.531 (IH3)
M2	793	782	784	802	10.952 (IH2)
M3	795	867	828	817	6.64 (IH3)
M4	735	785	811	795	5.828 (IH2)
M5	773	771	784	769	16.670 (IH1)
L	1340	1345	1686	1670	300.0 (IH1)

TABLE II. RESULTS OBTAINED BY HYBRID INITIALISATION HEURISTIC ON THE 20 ITC 2002 PROBLEM INSTANCES

Problem	IH1	IH2	IH3	IH4	Min Time
Com01	805	786	805	805	1.93 (IH3)
Com02	731	776	731	778	1.36 (IH3)
Com03	760	812	760	777	1.14 (IH2)
Com04	1201	1178	1201	1236	4.46 (IH2)
Com05	1246	1243	1246	1135	2.11 (IH3)
Com06	1206	1219	1206	1133	1.33 (IH3)
Com07	1391	1388	1391	1265	2.10 (IH3)
Com08	1001	968	1001	1006	1.81 (IH2)
Com09	841	859	841	843	1.46 (IH1)
Com10	786	816	786	799	4.64 (IH3)
Com11	852	877	852	839	1.05 (IH1)
Com12	814	831	814	788	2.21 (IH2)
Com13	1008	1010	1008	1009	2.26 (IH1)
Com14	1040	1032	1040	1355	3.71 (IH2)
Com15	1165	1162	1165	1161	1.56 (IH3)
Com16	887	911	887	888	1.09 (IH3)
Com17	1227	1032	1227	1199	1.13 (IH2)
Com18	793	724	793	763	1.29 (IH3)
Com19	1184	1212	1184	1209	3.22 (IH3)
Com20	1137	1161	1137	1205	0.08 (IH3)

Based on the performance shown in these experiments by the four initialization heuristics, we suggest that these algorithms can be incorporated in our proposed multi-agent system to find the feasible solution for each single timetabling agent in the distributed environment. We believe that cooperation among the agents can allow the strengths of one agent's heuristic to compensate for the weaknesses of another. Therefore, the multi-agents can employ the same or different initialization heuristic for producing their own feasible department timetabling solutions. In order to assess the benefit of tackling timetabling in a distributed manner, we intend to compare our multi-agent approach to other effective course timetabling methods that have not been designed with a distributed mindset, like the multi-stage approach by Kostuch [14].

V. CONCLUSION AND FUTURE WORK

Multi-agent approaches are promising as a technique for solving large course timetabling problems in a distributed manner. This technique has been long used in other domains such as e-commerce, scheduling and planning, and has been scientifically proven successful. Since different heuristics have different strengths and weaknesses, we believe that cooperation among the agents can allow the strengths of one agent's heuristic to compensate for the weaknesses of another. We outlined here the design for a multi-gent based cooperative search approach for course timetabling problems composed of a population of heuristic timetabling agents. The heuristic agents perform a local search through different solution spaces starting from their own department timetable. Each timetabling agent can employ different or the same heuristic for constructing their individual solution. We want to explore this new dimension by applying artificial intelligent agents for solving complex dynamic timetabling problems. The size of the problem contributes to the complexity of the timetabling problem particularly when multiple department timetables must be consolidated into one overall timetable. Therefore, this is an interesting challenge for multi-agent systems to apply a decomposition approach to tackle the problem. Having implemented four hybrid heuristics for the individual timetabling agents to create their own timetable, out future work is to develop the mediation stage and test the proposed distributed multi-agent model.

REFERENCES

[1] B. Paechter, L. M. Gambardella, and O. Rossi-Doria. International Timetabling Competition 2002. Metaheuristics Network. [Online]. Available: http://www.idsia.ch/Files/ttcomp2002/

[2] D. Landa-Silva and J. Henry Obit, "Evolutionary Non-Linear Great Deluge for University Course Timetabling," Hybrid Artificial Intelligent Systems, E. Corchado, X. Wu, E. Oja, A. Herrero, B. Baruque (eds.), Lecture Notes in Computer Science, Vol. 5572, pp. 269-276, Springer, 2009.

[3] E. Burke, B. A. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A Graph-Based Hyper-heuristic for Educational Timetabling Problems," European Journal of Operational Research, vol. 176, pp. 177–192, 2007.

[4] E. K. Burke and J. P. Newall, "A Multistage Evolutionary Algorithm for the Timetable Problem," IEEE Transactions on Evolutionary Computation. Vol. 3, No. 1, April 1999.

[5] E. Kaplansky, G. Kendall, A. Meisels, and N. Hussin, "Distributed Examination Timetabling," Proceeding of the 5th International Conference on the Practise and Theory of Automated Timetabling (PATAT 2004), pp. 511-516.

[6] J. Henry Obit and D. Landa-Silva, "Computational Study of Nonlinear Great Deluge for University Course Timetabling," Intelligent Systems - From Theory to Practice, Studies in Computational Intelligence, Vol. 299, V. Sgurev, M. Hadjiski, and J. Kacprzyk, Eds. Springer-Verlag, 2010, pp. 309–328.

[7] J. Henry Obit, D. Landa-Silva, M. Sevaux, and D. Ouelhadj, "Non-linear Great Deluge with Reinforcement Learning for University Course Timetabling," Metaheuristics – Intelligent Decision Making, Series Operations Research/Computer Science Interfaces, Springer, November 2011.

[8] D. Landa-Silva and J. Henry Obit, "Comparing Hybrid Constructive Heuristics for University Course Timetabling". *Proceedings of the VII ALIO/EURO Workshop on Applied Combinatorial Optimization*, Porto Portugal, pp. 222-225, May 2011.

[9] K. Socha, J. Knowles, and M. Samples, "A Max-Min Ant System for the University Course Timetabling Problem," Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002), Lecturer Notes in Computer Science 2463, Springer-Verlag, 2002, pp 1-13.

[10] L. D. Gaspero, S. Missaro, and A. Schaerf, "A Multi-agent Architecture for Distributed Course Timetabling," Proceedings of the 5th International Conference on the Practise and Theory of Automated Timetabling (PATAT 2004), pp. 471-474.

[11] M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria, "An Effective Hybrid Algorithm for University Course Timetabling," Journal of Scheduling, vol. 9, pp. 403–432, 2006.

[12] M. W. Carter, "A Decomposition Algorithm for Practical Timetabling Problems," Dept. Industrial Eng, University Toronto, Working Paper 83-06, 1983.

[13] G. O'Hare and N. Jennings, (Eds.), "Foundations of Distributed Artificial Intelligence," Wisely, New York, 1996, John Wiley Sixth-Generation Computer Technology Series. ISBN:0-471-006750.

[14] P. Kostuch, "The University Course Timetabling Problem with a Three-phase Approach," in The Practice and Theory of Automated Timetabling V, LNCS 3616. Springer, 2005, pp. 109–125.

[15] V. Wren, "Scheduling, Timetabling and Rostering – A Special Relationship?" The Practice and Theory of Automated Timetabling, Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT 1995), LNCS 1153, pringer, pp. 46-75, 1996.