University Course Timetabling with Probability Collectives

Brian Autry, Kevin Squire

Naval Postgraduate School
{bmautry, kmsquire}@nps.edu

Abstract. This paper presents a novel approach to the Post Enrollment Course Timetabling track of the 2nd International Timetabling Competition, sponsored by the Practice or Theory and Automated Timetabling (PATAT) and Working Group on Automated Timetabling (WATT). The approach is based on Probability Collective (PC) theory, an agent based approach to optimization. The basics of PC theory are discussed, including the basic algorithm and modifications necessary for course timetabling. Although we did not find a feasible schedule for any of the competition problem instances, our results suggest that Probability Collectives should be further explored for use in timetabling.

1 Introduction

Timetabling is the process of scheduling courses or events in a fixed period of time while satisfying various constraints such as classroom size and teacher availability. While timetabling problems exist in many different fields, the focus here is on timetabling problems in academic institutions, concerning course timetabling and examination timetabling. An overview of timetabling problems and some of the basic approaches used to solve them are discussed in (de Werra 1985) and (Schaerf 1999).

Curriculum based course timetabling is the problem faced by most universities. When developing the course schedule, the main constraints ensure that teachers are not scheduled to teach two or more courses at the same time and that no two courses are scheduled in the same room at the same time. Once the schedule is developed, students select courses based on availability of a seat in that course.

Some universities, such as the Naval Postgraduate School, use a post enrollment course timetabling system to create a course schedule based on student demand as well as teacher and room availability. This problem adds the additional constraint of satisfying student course needs, i.e., the schedule must ensure that no two students are scheduled for two or more courses at the same time.

The research described in this paper was motivated by the current course scheduling problem at the Naval Postgraduate School (NPS). Currently there are approximately 2000 students enrolled at NPS and nearly 500 courses offered each quarter. Scheduling is done manually by two people and the process takes up to 8 weeks to complete the schedule for one quarter. Clearly a better system is needed.

Coincidentally, the 2nd International Timetabling Competition was announced as we were beginning to study this problem. The competition offered a simplified post enrollment timetabling track useful for testing our ideas. This paper describes the methodology and results of our submission to that competition.

The rest of this paper is organized as follows. Section 2 describes the rules and structure of the timetabling competition and Section 3 describes previous methods for solving similar post enrollment timetabling problems. In Section 4 we review various agent-based approaches to timetabling, and provide an overview of Probability Collective (PC) theory, the basis of our approach, in Section 5. Section 6 describes our application of PC theory to post enrollment timetabling of university courses. Section 7 details our results, and Section 8 concludes and describes future work.

2 Timetabling Competition

The 2nd International Timetabling Competition sponsored by Practice and Theory of Automated Timetabling (PATAT) and the Working Group on Automated Timetabling (WATT) was held in 2007 and consisted of three tracks: examination timetabling, post enrollment based course timetabling, and curriculum based course timetabling (Lewis et al. 2007). The post enrollment and curriculum based course timetabling tracks are both subsets of the course timetabling problem. The post enrollment course timetabling track closely models the system used at NPS and was used as a simplified example to test and evaluate the effectiveness of PC theory in solving this type of problem.

The post enrollment based timetabling problem consists of scheduling a set of n events (courses) into 45 timeslots (5 days, 9 hours per day). A set of r rooms exists each with a set of f room-features. A set s of students who attend a varying combination of events is provided. Each of the n events has a set of available timeslots. A set of requirements is also provided that determine which events should occur before other events.

The goal is to schedule each event n into one of the r rooms and one of the timeslots while satisfying the following hard constraints:

- No student should be scheduled for two events at the same time.
- The room assigned to an event should be large enough for all of the students assigned to that event and should satisfy all of the room-features required by that event.
- Only one event is scheduled into each room in any timeslot.
- Events should only be scheduled in available timeslots.
- Events should be scheduled in the proper order as specified by any precedence requirements

In addition to the hard constraints, several soft constraints were also specified:

- Students should not be scheduled for an event occurring at the end of the day.
- Students should not have to attend three or more events in a row.
- Students should not be required to attend only one event on a particular day.

No hard constraints can be violated or the solution is rejected. Since for some problem instances it may not be possible to schedule each event and maintain all of the hard constraints, certain events may need to be left out to ensure all hard constraints are satisfied. A timetable which does not have any hard constraint violations but leaves out some events is considered valid. A feasible timetable is one in which there are no occurrences of any hard constraint violations and all events are scheduled.

Solutions are evaluated by first ensuring that they are valid. Next, Distance to Feasibility is calculated by summing up the number of students in each unscheduled event. Finally a Soft Cost is calculated by summing the total number of occurrences of soft constraint violations listed above. The solution with the lowest Distance to Feasibility is winner. If two valid solutions have the same Distance to Feasibility, the solution with the lowest Soft Cost is judged the winner.

3 Post Enrollment Timetabling Solutions

Several widely ranging methods have been used in the past to solve the post enrollment course timetabling problem, most prominently in the 1st International Timetabling Competition held in 2002 (Metaheuristics Network 2002). Among the most successful entries in that competition were a multi-phase approach based on graph coloring and simulated annealing (Kostuch 2005), methods based on tabu search (Cordeau et al. 2003; Di Gaspero et al. 2004), and a variant of the "Great Deluge" algorithm (Burke et al. 2003). In (Chiarandini et al. 2006), the authors developed a hybrid metaheuristic algorithm (a mixture of construction heuristics, variable neighborhood descent, and simulated annealing) which outperformed the winner of the original competition.

More recently, for the 2nd International Timetabling Competition, (Müller 2008b) applied a constraint-based framework from the Constraint Solver Library (Müller 2008a) to multiple timetabling problems, including the post enrollment track. The solver works through a series of algorithms based on local search techniques, and placed fifth in the post-enrollment competition track (and first in the other two tracks). Details of the work of other participants was not available at the time of this writing, but may appear in the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling or similar locations.

4 Agent-based Timetabling

Our approach to timetabling uses Probability Collectives (PC) theory, an agent-based approach to optimization. We describe PC theory in the next section. A number of other agent-based approaches to timetabling have been proposed in the past. In (De Causmaecker et al. 2002), the authors describe in an introductory article how agent-based systems might help deal with the distributed aspects of timetabling. (Kaplansky and Meisels 2004) and (Di Gaspero et al. 2004) follow that up with (different) techniques for using MultiAgent Systems (MAS) to solve university timetabling

problems. In (Kaplansky and Meisels 2004), a timetable is constructed through negotiations among multiple Scheduling Agents and Room Agents, where individual Scheduling Agents solve local problems using Constraint Satisfaction Problems (CSP) techniques. (Di Gaspero et al. 2004) describe a system based on monetary trading among agents, a common agent-based paradigm for solving optimization problems. In their system, each department has three agents: a Solver to generate a timetable for the department, a Negotiator to negotiate with other departments, and a Manager which manages information necessary for the other agents. In later work, (Yang et al. 2006) describe a setup similar to (Kaplansky and Meisels 2004), where Course Agents negotiate with one another through Signboard Agents (along with Publisher Agents and Interface Agents), which interact with each other through *ad hoc* negotiation rules to find a feasible schedule.

Other than the fact that these systems do not deal with post-enrollment scheduling, our PC Theory-based system differs from them in the complexity of the agents and in the way that the agents interact (or in the case of our system, do not interact). As with the systems proposed by Kaplansky and Meisels, and Yang et al, we have one agent per course. In our system, each agent maintains a utility for each feasible timeslot-room, and a probability distribution over the same set of feasible timeslot-rooms. For each iteration, each agent independently chooses a timeslot-room by sampling from its probability distribution, then updates its estimated utility for that room in the context of the choices of all other agents (courses). After a set number of iterations (an epoch), it updates its probability distribution slightly to favor classes which have higher utility.

More details will be given below, but from this description, it is obvious that 1) there is no direct negotiation among agents—all "communication" occurs through the utility calculation; and 2) each agent is relatively simple, in that it does not have to do much more than sample from a probability distribution and update its utilities and probability distributions.

5 Probability Collectives

Probability Collective (PC) theory (also known as Product Distribution (PD) theory) is a relatively new agent-based approach to solving optimization problems, drawing ideas from evolutionary game theory and statistical physics. It has been successfully used in areas including flight control (Bieniawski, Stefan 2005), airline flight scheduling (Antoine et al. 2004), and cooperative sensing (Waldock and Nicholson 2007). Good introductory references describing PC theory include (Bieniawski, S. R. and Wolpert 2004) and (Bieniawski, S. et al. 2004). (Wolpert 2004) gives an in-depth description of the theory. (Macready and Wolpert in press) contains examples of PC theory applied to large k-SAT (100 variable, k=3) constraint satisfaction problems and unconstrained minimization of NK functions (N=300, K=2), which are (perhaps) slightly smaller than but comparable to many timetabling problems.

PC theory is a global optimization technique, and can most easily be compared with genetic algorithms and simulated annealing. In contrast with genetic algorithms, which update populations of solutions, PC theory maintains and updates a

parameterized distribution *p* over the space of solutions (Huang et al. 2005). This distribution *p* is a product distribution, and is iteratively updated so that it peaks around "good" solutions. As with simulated annealing, the updates to *p* are controlled partially by a temperature setting, which provides a trade-off between exploration of the solution space (high temperature) and exploitation of learned knowledge (low temperature).

PC theory distributes an optimization problem among agents representing variables in the system. Each agent maintains a predefined set of possible actions, along with a utility function defined over these actions. The goal of the agent is to optimize the utility function. An agent independently chooses an action, evaluates the utility of its choice in the context of every other agent's current action, and updates the estimated utility of its action. A global utility may also be calculated and used to update the action utility. Each agent also maintains a probability distribution defined over its possible actions. (Each agent's distribution is a component of the global product distribution *p* over the set of solutions, mentioned above). These probabilities are also updated based on the utility calculation and the current "temperature" of the system. Subsequent action choices are made by sampling this distribution. The system evolves until it reaches an equilibrium state where no improvements can be made by altering agent actions. PC theory assumes that each agent is bounded rational and independent and will make choices based only on its own probability distribution, with no direct communication among agents.

One benefit of using PC theory is that since each agent chooses actions independently, the problem can easily be parallelized. Despite this obvious extension, we have not yet taken advantage of this optimization.

In the next two sections, the optimization approach and algorithm from (Bieniawski, S. et al. 2004) are summarized.

5.1 Optimization Approach

Assuming that each agent is bounded rational and operates in an environment with global utility *G*, the system equilibrium will be the optimizer of *G* subject to any constraints imposed. This equilibrium can be found by minimizing the Lagrangian for each agent as a function of the probability distribution associated with the agents' possible actions. The Lagrangian $\mathcal{L}_i\left(q_i\right)$ is given by

$$\mathcal{L}_i\left(q_i\right) = \mathrm{E}\left[G(x_i, x_{(i)})\right] - TS\left(q_i\right)$$

where *G* is the system objective which depends on the agent *i*'s action x_i and the actions of all other agents, $x_{(i)}$. The probability distribution of agent *i* is represented by q_i. *S* is the entropy of this distribution and is given by:

$$S\left(q_i\right) = -\sum_{x_j} q_i\left(x_j\right) \ln q_i\left(x_j\right)$$

T is the temperature of the system and determines the amount of exploration the agent engages in. Each agent attempts to minimize the Lagrangian function $\mathcal{L}_i\left(q_i\right)$, subject to

$$\sum_{x_i} q_i\left(x_i\right) = 1, q_i\left(x_i\right) \geq 0, \forall x_i$$

This ensures that the sum of the probabilities in the probability distribution sum to 1 and that there are no negative probabilities.

When temperature *T* is high, much weight is given to the entropy component of the equation, which minimizes the Lagrangian by encouraging a uniform distribution and therefore encourages more exploration of the space by the agent. As the temperature decreases, exploration becomes less important and the agent begins to exploit action choices which are "better" (lower cost/higher utility) than others.

After a fixed set of iterations (epoch), the probabilities are update using Newton updating, with the update equation

$$q_i\left(x_i\right) \to q_i\left(x_i\right) - \alpha q_i\left(x_i\right) \times \left\{ \frac{\mathrm{E}\left[G \mid x_i\right] - \mathrm{E}\left[G\right]}{T} + S\left(q_i\right) + \ln q_i\left(x_i\right) \right\} \quad (1)$$

where α is a step size determining how much the existing probability is modified by this iteration's results. The probability distribution is then renormalized ensuring that there are no negative probabilities and that the sum all of the probabilities is 1.

To calculate the expected utility for each agent, we use

$$\mathrm{E}\left(g_i \mid x_i = j\right) = \frac{N_{ij}^{(k)}}{D_{ij}^{(k)}} = \frac{\sum_m g_i(x_i = j, x_{(i)}) 1\left(x_i = j\right) + \gamma N_{ij}^{(k-1)}}{\sum_m 1\left(x_i = j\right) + \gamma D_{ij}^{(k-1)}} \quad (2)$$

where $1\left(x_i = j\right)$ equals 1 when $x_i = j$ and 0 otherwise. The agent's private utility is represented by g_i. *D* tracks the number of times an agent *i* chooses a particular choice *j* and *N* tracks the private utility when then choice is made. Data aging is controlled by γ.

Constraints are added to the system by the addition of Lagrange multipliers, λ_j, to the global utility along with constraint functions, $c_j\left(\vec{x}\right)$, as

$$G\left(\vec{x}\right) \to G\left(\vec{x}\right) + \sum_j \lambda_j c_j\left(\vec{x}\right).$$

The update rule for the Lagrange multipliers is

$$\lambda_j \rightarrow \lambda_j + \eta \mathrm{E}\left[c_j\left(\vec{x}\right)\right] \qquad (3)$$

where η is separate step size.

Expected utilities for each agent are computed by Monte-Carlo simulation. This is accomplished by all agents repeatedly identically and independently sampling their distributions to generate moves, and then calculating utilities based on these moves. The private utility calculation should be chosen to ensure low bias and low variance. Low bias ensures that the private utility closely resembles the global utility. Low variance ensures that each agent's contribution to global utility is distinguishable.

5.2 Solution Algorithm

The basic algorithm to solve problems with PC theory is as follows:

1. Initialize the system
    a. Initialize the parameters {T, α, γ}. Set the convergence criteria δ.
    b. Select the number of Monte Carlo Samples.
    c. Initialize the probability collectives
2. While $\delta \geq \left\| \vec{\lambda}^{k} - \vec{\lambda}^{k-1} \right\| + \sum_i \left\| \vec{q}_i^{\,k} - \vec{q}_i^{\,k-1} \right\|$
    a. For each Monte-Carlo sample,
        i. Jointly IID the sample
        ii. Evaluate the objective function
        iii. Compute each agent's private utility
    b. Compute the expected utility for each agent using Eq. (2).
    c. Update the probability distributions using Eq. (1).
    d. Update the Lagrangian multipliers using Eq. (3).
3. Final Evaluation
    a. Determine the highest probability value for each variable
    b. Evaluate the objective function with this set of values.

6 Application of PC Theory to University Course Timetabling

In this section, the details of how the post enrollment course timetabling problem was approached using PC theory are described. In this application, each event is represented by an agent, and each agent's actions are drawn from time-slots/rooms available to that agent, and are determined individually based on the specific requirements of the event.

In our system, an agent's private utility is calculated by counting the number of collisions that exist given a certain time-slot/room choice[1]. Collisions can occur in

[1] Technically, the utility is the negative of the number of collisions, and we wish to maximize this value (minimize the number of collisions). This will be implied in the rest of the paper.

two ways. First, if an event is scheduled in the same timeslot and room as another event, the number collisions are equal to the size of the union of the two sets of students. The second form of a collision is when two events are scheduled in the same timeslot but in different rooms and the intersection of the two sets of students is non-zero. The size of the intersection is the number of collisions in this case.

An agent's local utility is calculated as the expected number of collisions given its current utility function and probability distribution, and is calculated by summing the products of the probability of choosing each timeslot-room combination and the expected utility associated with that choice. An average of the local utilities is used to estimate actual global utility. According to (Bieniawski, S. et al. 2004), the local utility function should be chosen to have low bias and variance with respect to the global utility. While we believe our choice follows these guidelines, other local utility functions may be more appropriate.

6.1 Initialization

During the system initialization data structures are generated representing the events, rooms, and students using data provided by the International Timetabling Competition. We map students to events, rooms to available features, required features to events, events to available time slots, and evaluate event precedence. Initial parameters, including the rate of cooling (or rate of change of temperature) (ΔT), the number of Monte-Carlo samples (m) per iteration, and initial temperature, are set according to user input. Initial temperature is carefully selected to ensure a good amount of exploration occurs early on. The rate of cooling controls how rapidly the system moves from exploration to trade off exploitation.

Next, each event is initialized. The probability distribution is set to only include rooms of the appropriate size, rooms that have the features required by that event, and timeslots that are available to that event. The distribution is initialized by assigning an equal probability to each available choice. Data structures to track utilities for individual choices and the number of times each choice is made (N and D respectively) are created and initialized to zero.

Instead of setting the step size α and data aging factor γ to a preset value, each event sets its own values based on the number of students assigned to that event. Smaller classes are assigned a lower value for α and γ to allow for more movement for a given iteration while larger classes are given higher values to prevent large jumps between iterations. The theory behind this is based on how particles react in the real world. For a given energy imparted on a system, smaller particles will move faster and farther than larger ones. The end result is that events with larger class sizes tend to be placed earlier while the smaller events are allowed more freedom to move and find an optimum slot.

The final phase of event initialization consists of pre-calculating the number of collisions that can occur between two distinct events, by calculating the student set intersection between every pair of events.

6.2 The Optimization

For every iteration of the optimization, the *m* Monte-Carlo samples for each event (representing room and time slot choices for the next *m* iterations) are first generated. We used stochastic universal sampling (Baker 1987) to generate all *m* samples up front in O(*m*) time. These samples are initially ordered and must be randomly permuted.

Next, we iterate *m* times. For each iteration, each agent computes its private utility (number of student/course collisions) for its chosen room and time slot, and tracks the number of times each particular choice is made. After all iterations, each event updates its *N* and *D* data structures and calculates expected utility using Eq. (2). They also calculate their localized global utility and entropy.

If the change in average localized global utility $\geq \delta$, temperature is decreased by ΔT and next set of Monte-Carlo samples is calculated. If the change in average localized global utility is minimal, then the algorithm moves to the final evaluation phase.

6.3 Final Evaluation

At this point each event should have a probability distribution that reflects the best choice or choices of timeslot-room combinations that allow it to minimize the total number of system collisions. The next step is to assign events to the timeslot-room combination that best suits the event while ensuring that none of the hard constraints are violated. Since it is likely that in a very dense schedule there may still be constraint violations, events are scheduled in descending order of number of students associated with that event.

For each event, the algorithm attempts to schedule the event in the timeslot-room combination with the highest probability. This choice must be compared with already scheduled events to ensure that there are no student collisions, no two events are scheduled in the same room at the same time, and that all events are scheduled in the order established by any precedence requirements. Lower probability choices are tried if the first choice is not successful. If no acceptable timeslot-room combination is found, the event is unscheduled.

7 Results

Sixteen problem instance data sets were provided by the International Timetabling Competition for testing purposes. Table 1 lists the number of events, rooms, possible room features, and number of students associated with each instance.

For the competition, a time limit was imposed based on the number of computer cycles. A benchmark tool was provided on the competition website. The test machine was a virtual machine running Windows XP and allocated 1024 MB of RAM. The base machine was a Macintosh iMac running at 2.8GHz. Based on the benchmark tool, the maximum allowable run time was approximately 600 seconds.

The following parameters were used for the competition: $\Delta T = 0.90$, 500 Monte-Carlo samples per iteration, and an initial temperature factor of 2.0. The determination of α and γ was made based on the actual minimum event student size of 1 and a maximum of 98. The formula to calculate both variables ensured that the values for the largest event were fixed at 0.9 and the variables for the smallest event were fixed at 0.1. The resulting formula is:

$$\alpha = \gamma = 0.00825x + 0.0964$$

where x is the number of students requesting the event.

Table 1. Description of problem instances.

Instance	Events	Rooms	Features	Students
1	400	10	10	500
2	400	10	10	500
3	200	20	10	1000
4	200	20	10	1000
5	400	20	20	300
6	400	20	20	300
7	200	20	20	500
8	200	20	20	500
9	400	10	20	500
10	400	10	20	500
11	200	10	10	1000
12	200	10	10	1000
13	400	20	10	300
14	400	20	10	300
15	200	10	20	500
16	200	10	20	500

Table 2. Competition results.

Instance	Distance to Feasibility	Soft Cost	Run Time (sec)	Worst Case	Percent Scheduled
1	3076	1798	597	10515	70.7
2	3170	1725	586	10515	70.0
3	1997	3010	145	13383	85.1
4	2040	2711	145	13396	84.8
5	1239	1157	539	6275	80.3
6	971	1322	529	6218	84.4
7	687	1449	133	6733	89.8
8	756	1496	138	6916	89.1
9	2814	2076	593	10714	73.7
10	3035	1685	581	10492	71.1
11	2804	2937	159	13608	79.4
12	2930	3123	155	13607	78.5
13	1424	1175	535	6358	77.6
14	1362	1154	538	6257	78.2
15	808	1362	142	6527	87.1
16	576	1366	145	6819	91.6

Table 2 displays the results of the competition runs. The worst case column indicates the Distance to Feasibility if no events were placed. The program generated valid timetables but was unable to find a feasible solution for any of the problem instances. Each instance is known to have at least one feasible solution though the competition organizers feel that these solutions will most likely not be found in the give time. At the time of writing this paper, the results from the competition were not available and therefore a comparison with the other competition entries is not yet possible.

Figure 1 shows the progression of average localized global utility over time. Three different problem instances of varying size are shown. ("Utility" here is actually a cost and lower average costs are desired.) The initial high temperature allows for greater exploration in the beginning of the optimization which explains the initial rise in average localized global utility followed by a rapid drop to nearly zero as the events find their optimal timeslot-room combination.

Figure 2 and 3 show the progression of the probability collectives through the optimization process for events 5 and 21 of problem instance 3. Student sizes for the events were 65 and 15 respectively. By iteration 37, event 5 narrowed the number of possible timeslot room combinations to one. At the end of the optimization, event 21 had 3 timeslot-room combination choices of about equal quality. This solution indicates that the optimization problem is underconstrained as implemented, leading to a suboptimal solution.

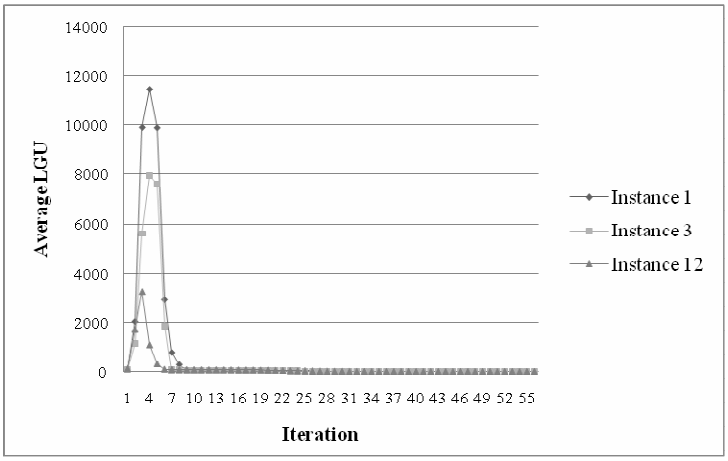

Figure 1. Change in average localized global utility by iteration.

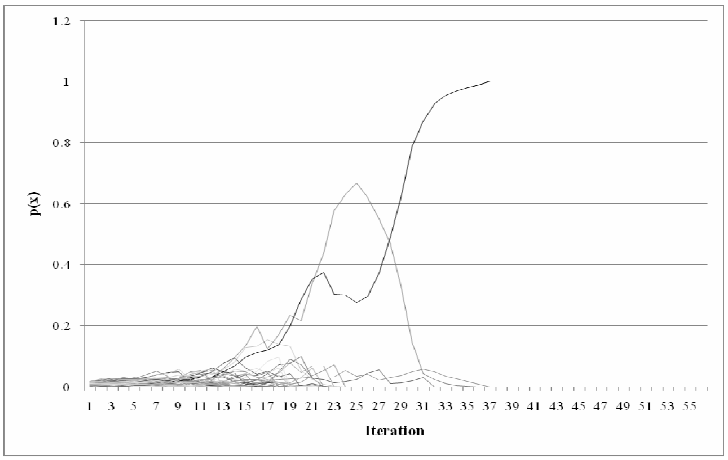

Figure 2. Evolution of the probability collective for event 5 of problem instance 3.

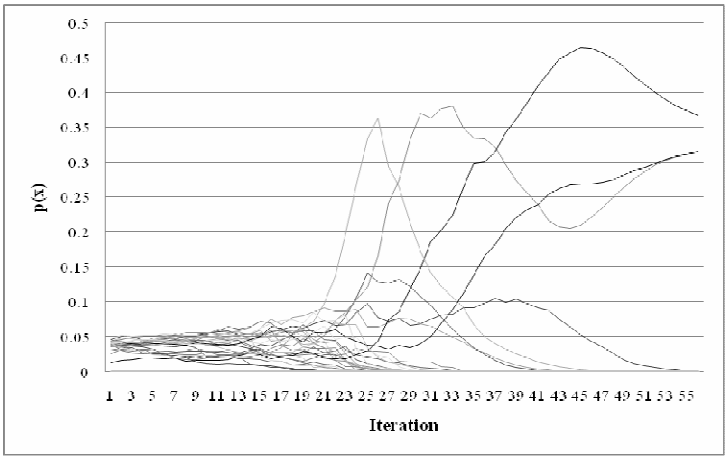

Figure 3. Evolution of the probability collective for event 21 of problem instance 3.

8 Conclusion and Future Work

We have implemented a solution to the post enrollment course timetabling problem based on Probability Collectives theory. The algorithm produced valid timetables for every instance, though 100% student placement was not achieved for any instance. The algorithm was able to successfully schedule between 70.0% and 91.6% of the student event requests.

The biggest issue in our solution is that the timetabling problems as given (and as implemented) are underconstrained, leading to many agents finding high utility values for multiple timeslot/room pairs. This problem may suggest a two- or multi-stage solution where classes are first deconflicted and times and rooms are chosen later, although it is currently unclear how to easily implement this using probability collectives. Additional soft constraints may better push the optimization in a particular direction (and indeed, we did not consider the specified soft constraints in our algorithm). (Macready and Wolpert in press) provide a mechanism to deal with this problem, by simultaneously calculating multiple solutions to the problem. We plan to explore these solutions.

One of the major benefits of PC theory is the inherent parallelism, which we have not yet taken advantage of. Finally, as indicated in the introduction, we plan to apply this algorithm to the NPS scheduling problem, which instigated this research. In doing so, we will have to take into account additional practical constraints, including:

- Multiple sections of the same course. This actually adds flexibility to the timetabling process by allowing the student's course requirement to be fulfilled in different timeslots.
- Multiple professors for the same course. This problem description considers professors linked to individual courses. At NPS, some courses are taught

by multiple professors, particularly when considering labs associated with a course.

- Courses that are taught via online methods. These courses still have students assigned, but do not require a room.
- Room availability. Often rooms are prescheduled for events that are not related to a course.
- Professor preferences. Some professors are only available on certain days of the week and at specific hours. Also, there are some classrooms that professors prefer to not teach in.
- Departmental ownership of rooms. The problem formulation does not address assigning courses to rooms owned by the associated departments.

Most of these issues have an impact on timetable creation, although they should not affect the basic algorithm.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and constructive criticisms. This work was partially funded by the Naval Postgraduate School.

References

Antoine N, Bieniawski S, Kroo I, et al. (2004). Fleet Assignment Using Collective Intelligence. 42nd Aerospace Sciences Meeting (AIAA 2004).

Baker JE (1987). Reducing bias and inefficiency in the selection algorithm. Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application. Cambridge, Massachusetts, United States, Lawrence Erlbaum Associates, Inc.

Bieniawski S (2005). Distributed Optimization and Flight Control Using Collectives. Department of Aeronautics and Astronautics, Stanford University. PhD**:** 169.

Bieniawski S, Wolpert DH and Kroo I (2004). Discrete, Continuous, and Constrained Optimization Using Collectives. The 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. Albany, NY.

Bieniawski SR and Wolpert DH (2004). Product Distributions for Distributed Optimization. International Conference on Complex Systems. Bar-Yam Y, Perseus Books.

Burke E, Bykov Y, Newell J, et al. (2003). A Time Predefined Approach to University Timetabling. Yugoslav Journal of Operations Research 13(2): 139-151.

Chiarandini M, Birattari M, Socha K, et al. (2006). An Effective Hybrid Algorithm for University Course Timetabling. J. of Scheduling 9(5): 403-432.

Cordeau J-F, Jaumard B and Morales R (2003). Efficient Timetabling Solution with Tabu Search http://www.idsia.ch/Files/ttcomp2002/jaumard.pdf. Accessed 29 June 2008.

De Causmaecker P, Demeester P, Lu Y, et al. (2002). Agent Technology for Timetabling. Fourth International Conference on the Practice and Theory of Automated Timetabling IV. Burke E and De Causmaecker (Eds.) P. Gent, Belgium.

de Werra D (1985). An Introduction to Timetabling. European Journal of Operational Research 19(2): 151-162.

Di Gaspero L, Mizzaro S and Schaerf A (2004). A MultiAgent Architecture for Distributed Course Timetabling. Fifth International Conference on the Practice and Theory of Automated Timetabling. Burke EK and (editors) MT. Pittsburgh (PA), USA.

Huang C-F, Bieniawski S, Wolpert DH, et al. (2005). A comparative study of probability collectives based multi-agent systems and genetic algorithms. Proceedings of the 2005 conference on Genetic and evolutionary computation. Washington DC, USA, ACM.

Kaplansky E and Meisels A (2004). Negotiation Among Scheduling Agents for Distributed Timetabling. Fifth International Conference on the Practice and Theory of Automated Timetabling. Burke EK and (editors) MT. Pittsburgh, PA USA.

Kostuch P (2005). The University Course Timetabling Problem with a Three-Phase Approach. Sixth International Conference on the Practice and Theory of Automated Timetabling. Burke E and Trick M. Pittsburgh, PA USA, Springer. 3616/2005**:** 109-125.

Lewis R, Paechter B, McCollum B, et al. (2007). Post Enrolment based Course Timetabling: A Description of the Problem Model used for Track Two of the Second International Timetabling Competition, Cardiff University, Cardiff Business School, Accounting and Finance Section.

Macready W and Wolpert D (in press). Distributed constrained optimization with semicoordinate transformations. Operations Research.

Metaheuristics Network (2002). International Timetabling Competition. http://www.idsia.ch/Files/ttcomp2002/.

Müller T (2008a). Constraint Solver Library. http://cpsolver.sf.net/. Accessed 29 June 2008.

Müller T (2008b). ITC2007: Solver Description. www.cs.qub.ac.uk/itc2007/winner/bestexamsolutions/itc2007_Muller.pdf. Accessed 29 June 2008.

Schaerf A (1999). A Survey of Automated Timetabling. Artif. Intell. Rev. 13(2): 87-127.

Waldock A and Nicholson D (2007). Cooperative decentralised data fusion using probability collectives. 1st International Workshop on Agent Technology for Sensor Networks.

Wolpert D (2004). Information Theory - The Bridge Connecting Bounded Rational Game Theory and Statistical Physics. In: Complex Engineering Systems. Braha D and (Editors) YB-Y, Perseus Books.

Yang Y, Paranjapea R, Benedicentia L, et al. (2006). A system model for university course timetabling using mobile agents. Multiagent and Grid Systems – An International Journal 2(3): 267-275.