2012 Third International Conference on Intelligent Systems Modelling and Simulation

A Non-Worsening Memetic Algorithm for Constructing Feasible Timetables

Tri A. Budiono
School of Information Technology
Murdoch University
Western Australia
t.budiono@murdoch.edu.au

Kok Wai Wong
School of Information Technology
Murdoch University
Western Australia
k.wong@murdoch.edu.au

Abstract— **The choice of local search method in a Memetic Algorithm (MA) affects the effectiveness of the MA to produce feasible timetables. Since the local search itself depends on the definition of the neighborhood operator, we intend to understand the effect of the employed neighborhood operator to the performance of MA. In this work, we examine two typical neighborhood operators used in solving timetabling problems, that is neighborhood based on Move (reschedule lectures and replace rooms) and Swap (exchange the timeslot and room of two lectures). The results reveal the importance of defining neighborhood operator suitable with the problems. The performance of MA which surpasses the corresponding GA is also confirmed by the results which encourage the use of MAs in solving timetabling problems over GA.**

Keywords- Genetic Algorithms; Memetic Algorithms; Neighborhood Operator; Feasible Timetables

I. INTRODUCTION

Timetabling problem assumes a various forms of problem solving situation including educational timetabling, nurse timetabling, sport timetabling and transport timetabling [1]. One of the most studied timetabling problems is university course timetabling (UCTP) due to its practical importance. In a UCTP, lectures for each course (events) must be assigned into a given number of time slots and rooms such that the constraints are satisfied. The imposed constraints are of hard and soft type when their satisfaction is mandatory and desirable, respectively [2]. A timetable in which all lectures have been assigned a timeslot and a room so that no hard constraint is violated is said to be feasible [3].

Genetic algorithms (GA) are global search techniques that have been used widely to produce feasible timetables [4]. However, with their more explorative nature than exploitation of the search space, the solution quality produced by GA may not be more superior than the one produced by local search techniques [5]. Furthermore, they can take a relatively long time to locate the local optimum in a region of convergence [6]. However, GA is able to perform a multidirectional search using a set of candidate solutions [7], which other techniques cannot do so.

Hart suggested that utilizing a local search method within GA can improve the exploiting ability of the search algorithm without limiting its exploring ability [8]. The combination of (GA) and local searches (LS) are recently known as Memetic Algorithms (MA). These methods are inspired by model of Universal Darwinism concept in which natural selection happens not only on gene as biological unit of evolution, but also on meme as cultural unit of evolution [9]. In diverse literatures, MA is also known Hybrid Genetic Algorithm [10,11], Lamarckian Genetic Algorithm [12], and Baldwinian Genetic Algorithm [13].

In this paper, we investigate a reduction version of a typical university course timetabling problems that reflects aspects of the real timetabling problems of Binus University in Indonesia. A memetic algorithm that is constituted of a standard genetic algorithm and a hill climbing algorithm is used to solve the problems. The objective of this work is to understand the effect of neighborhood operator on the memetic algorithm behaviors and performance. Specifically, we aim to establish the effective number of steps required by the local search algorithm within a memetic algorithm that makes use of the chosen neighborhood operator. In addition, we also intend to measure the relative effectiveness of the MA which makes use of a combination of neighborhood operator to the GA in generating feasible timetables.

II. UNIVERSITY COURSE TIMETABLING PROBLEM

Depending on the algorithm techniques used, the timetabling problem may be specified as assigning a room to time-events slots [14], or assigning event to time slots and resolving the room assignment in a separate process [15]. In this work, the assignment of lectures to a timeslot and a designated room is performed simultaneously.

A. Problem Definition

Carter and Laporte [16] viewed the UCTP as a multidimensional assignment problem, in which students and teachers are assigned to courses, course sections, or lectures and then the lectures (individual meetings between students and teachers) are assigned to timeslots and classrooms. The real world UCTP aims at producing timetables that satisfies all the hard constraints and minimize the number of violation to soft constraints. In this work, however, we only concern with the construction of feasible timetables, that is for a given number of rooms and timeslots, a timetable containing a predefined number

978-0-7695-4668-1/12 $26.00 © 2012 IEEE
DOI 10.1109/ISMS.2012.38

of lectures must be constructed such that all the hard constraints are satisfied. The following hard constraints are imposed:

1. H1: two lectures must not take place at the same time in the same room (no room conflict)
2. H2: lectures must happen in a room having enough available seats (number of seat requirement)
3. H3: lectures must occupy a room suitable for it in terms of facilities (facility requirement)
4. H4: lectures having professors in common cannot take place at the same time (no professor's time conflict)
5. H5: lectures having students in common cannot occur at the same time (no student's time conflict)

B. Solution Representation

The solution is represented by a direct representation due to [17] that allows a simultaneous assignment of lectures to timeslots and rooms. It consists of a vector of timeslots and room (time-space) containing lectures for each hour. This data structure allows multiple lectures to occupy the same time-space slot during execution of the algorithm. The size of the vector is determined by the number of room (R), number of day (D) and number of session per day (S). Since the algorithm is implemented in Java, all input data are encapsulated in objects, for example a lecture of course is wrapped in an object of Lecture. This timetable solution representation is depicted in the following Fig 1.

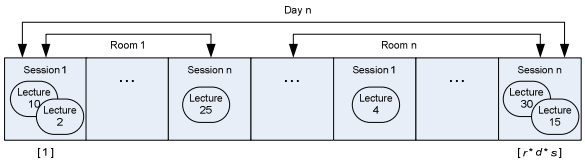

Figure 1. Timetable Solution Representation

The search space is the family of vector T containing lectures, such that $T_i = j$ (with $1 \leq j \leq E$; E is the number of lectures) means that the lecture j occupies the room (1), occurs on the day (2) and during the session (3).

$$room = \frac{(i\%(s*r))}{s} \tag{1}$$

$$day = i/(s*r) \tag{2}$$

$$session = (i\%(s*r))\%s \tag{3}$$

To make the above representation suitable for the genetic operations such as crossover and mutation, a chromosome vector is created from the corresponding timetable by storing a pair of lectures and its position in the timetable vector. Fig. 2 shows this chromosome representation.

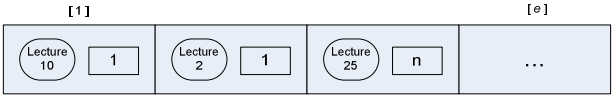

Figure 2. Chromosome Representation

C. Fitness Function and Fitness Distance

Equation 4 shows the fitness function (cost function) that is defined as the score of a timetable with regard to its satisfaction to all the imposed hard constraints. If a lecture satisfies a hard constraint, its score is incremented for that constraint. The total score of a timetable is the sum of points of all lectures in the timetable. This definition of fitness function means that a feasible timetable is achieved when its fitness value equal to one.

$$f = \frac{\sum_{c=1}^{E}\sum_{i=1}^{H} h_{i,c}}{ExH} \tag{4}$$

E is the number of lectures, H is the number of hard constraints

III. THE ALGORITHMS

Since the objective is to understand the effect of neighborhood operators on the memetic algorithm behaviors and performance, we will use both the GA and MA in constructing feasible timetables. The GA and MA share the same genetic operators and the local search used in the MA attempts to search a feasible timetabling in a region defined by the chosen neighborhood operators.

A. Genetic Algorithms

The basic framework of GA used in this paper is a Simple Genetic Algorithm (SGA) [18]. The GA works from an initial population of solutions that are randomly generated (i.e., lectures are randomly assigned to timeslots and rooms). For each generation, it performs three genetic operations. First, it randomly selects n pairs of parents from the current population and produces n new chromosomes by performing a crossover operation on the pair of parents. It then randomly selects n chromosomes from the current population and replaces them with new ones. The algorithm does not select chromosomes for replacement if it is among the best chromosomes in the

population. It repeats these two operations until the best chromosome reaches a fitness value equal to one (meaning that all lectures in the timetable meet the constraints). A 2-points cross over operator is used in the GA, in which it splits both parents chromosome in parts of random size with a probability P_c. It then alternately copies parts from these parents to produce a new child chromosome. Following the crossover operation, the produced child undergoes a mutation operation. This mutation takes a lecture randomly and moves it to another randomly chosen time-space slot with a probability P_m.

B. Neighborhood Operators

A local search requires a neighborhood for searching a feasible timetable by applying a simple local change (move) to the current timetable. Two basic moves are considered, that is moving a lecture to an empty timeslot or room (Move) and swapping timeslots and rooms of two lectures (Swap). Since there are five hard constraints imposed, we devised three neighborhood operators that implement those basic moves, i.e. MoveTime, MoveRoom and SwapEvent.

The MoveTime reschedules a lecture from one session to another session (leaving the room unchanged). This kind of move attempt to remedy the infeasibilities due to teacher's time conflict (H4) and student's time conflict (H5). The MoveRoom replaces the room of a lecture of a given course, without changing the time. This move fixes infeasibilities related to room conflict (H1), room capacity (H2) and room facility requirements (H3). Before performing a move, MoveTime and MoveRoom look for an empty slot and commit the move only when they do not introduce a new infeasibility. While MoveTime and MoveRoom attempt to fix infeasibilities due to time related and room related constraints, the SwapEvent tries to reduce the infeasibilities due to both constraints at once. It randomly takes two lectures and swaps these lectures as long as it improves the fitness of the candidate solution. The three moves in a timetable matrix are illustrated in the Fig 1 below.

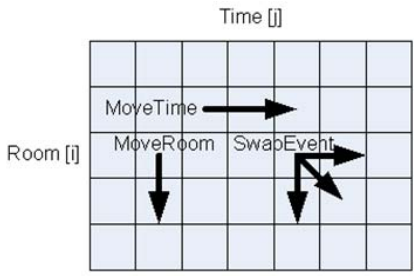

Figure 3. MoveTime, MoveRoom and SwapEvent moves in a timetable matrix

Given the above neighborhood structures, a first improvement hill climbing algorithm is used to search for a feasible timetable in the vicinity of the current solution. This local search makes only moves that always improve the cost function, thus it never performs worsening moves.

C. The Memetic Algorithm

A memetic algorithm that makes use of the representations, genetic operators and a local search utilizing the above neighborhood operators has been implemented. Since this MA uses a local search that never perform worsening moves, we called our algorithm as a non-worsening MA and it is outlined in the Algorithm 1 below.

Algorithm 1 Memetic Algorithm

1: **input**: A timetabling problem instance
2: Initialize population
3: Store best chromosomes
4: Check the fitness of the best chromosome
5: If the best chromosome's *fitness = 1*, a feasible solution is produced
6: Take *n* pairs of chromosomes from the population
7: Perform crossover operation on these pairs
8: Perform mutation operation on these *n* produced offspring chromosomes
9: Perform local search on these *n* produced offspring chromosomes
10: Put these *n* offspring chromosome back to the population
11: Go to step 3 for proceeding to the next generation
12: **output**: A feasible timetable

The local search in step 9 of the Algorithm 1 is a first improvement hill climbing algorithm that makes use of either the Move operator alone or Move and Swap operators altogether.

IV. Experimental Results and Analysis

We coded all the algorithm in Java and run the algorithms on an AMD Turion X2, 2.8 GHz 3 GB RAM PC. A reduction version of a typical university course timetabling problems that reflects aspects of the real timetabling problems of Binus University was used to study the behavior of the algorithms. Table I below shows the data of this UCTP instances.

TABLE I. Timetabling Problem Instance

Input Data	Value
Number of Lectures	112
Number of Students	900
Number of Student Groups	58
Number of Rooms	28
Number of Sessions per day	13
Number of Day per week	5

We performed three experiments in this study. The first experiment is for establishing the effective number of steps required by the local search that makes use of the neighborhood defined by MoveTime and MoveRoom. The second experiment examines the effectiveness of the local search that utilizes the neighborhood defined by SwapEvent. The third experiment compares the effectiveness the defined neighborhood operators in computing feasible timetables. We set the parameters for both the GA and MA as follows: the population size was set to 100, the

crossover probability P_c was set to 0.8 and the mutation probability P_m was set to 0.3. For each experiment, we performed 100 runs.

A. Experiment 1: Number of Step for Local Search

In the first experiment, we found out that the average number of steps required by the local search that makes use of the neighborhood defined by MoveTime and MoveRoom is 12 steps. It means, if the local search is set to have more than 12 steps, the improvement of the candidate solution fitness is not significant, as depicted in the Fig. 4.

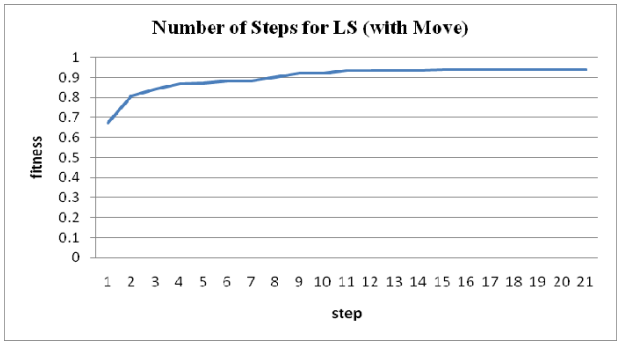

Figure 4. The Number of Steps Required by LS utilizing Move Neighborhood

B. Experiment 2: Local Search Employing SwapEvent Operator

In the second experiment, we learnt that the lesser number of trials (swap attempts), the higher is the percentage of successful swap. However the lesser number of trials also implies the lesser percentage of the fitness improvement. It means that there is a trade-off between the number of swap attempts (trials) and the expected percentage of the fitness improvement, since the higher the number of trials will make the local search expensive. Fig. 5 summarizes these findings.

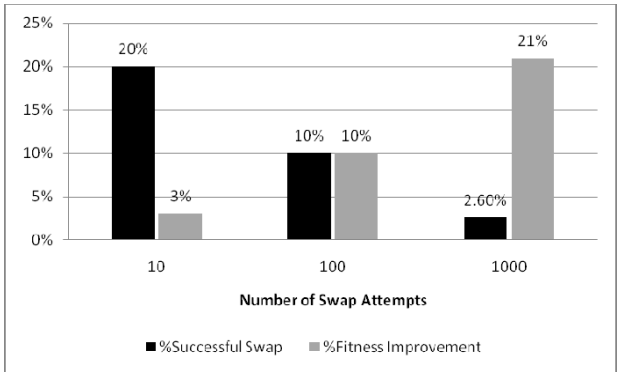

Figure 5. Effectiveness of Local Search with SwapEvent Neighborhood

From the results of this experiment, we found that 100 swap attempt (trials) should give the best trade-off between the percentage of successful swaps and the percentage of fitness improvement. It means, in each step of the local search, SwapEvent operator will generate a 100 random pairs of lectures and attempt to swap them if it can improve the fitness.

C. Experiment 3: Neighborhood Operator Effectiveness Session

In the third experiment, we compare the effectiveness the defined neighborhood operators in computing feasible timetable. Since the neighborhood operator is used by the local search within the MA, and the MA itself comprise a GA and the local search, we can indicate the effect of each neighborhood operator by the relative performance of GA and MA.

Each algorithm was run for 100 times and the average results of running time and number of iteration to achieve feasibility is compared. Fig. 6 shows a sample iteration progress of GA to achieve feasibility. Fig. 7 and Fig. 8 depict the same data for MA using Move neighborhood operator and Move plus Swap operators, respectively. These figures show the importance of choosing neighborhood operators used by the local search, that in turn utilized by the MA.

The results show that a proper definition of neighborhood operators will help the MA to perform significantly faster than the corresponding GA. Even, the employment of local search by using different neighborhood operators gives significantly different performance of the corresponding MA. For example, in Fig. 7 and Fig 8, a sample run of the MA that uses a local search based on Move and Swap neighborhood operators surpasses the MA that only uses a local search based on Move neighborhood operator only. Table II confirms this result, since it gives the average number of iteration and the average runtime to achieve feasibility of both algorithms for 100 runs.

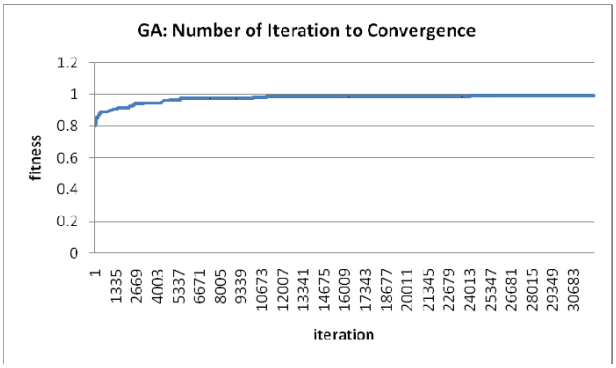

Figure 6. Iteration Progress to Feasibility: GA

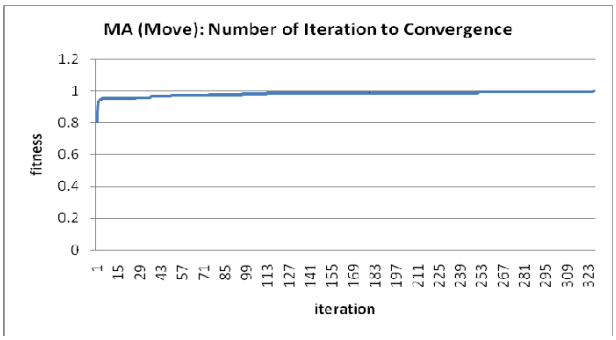

Figure 7. Iteration Progress to Feasibility: MA1

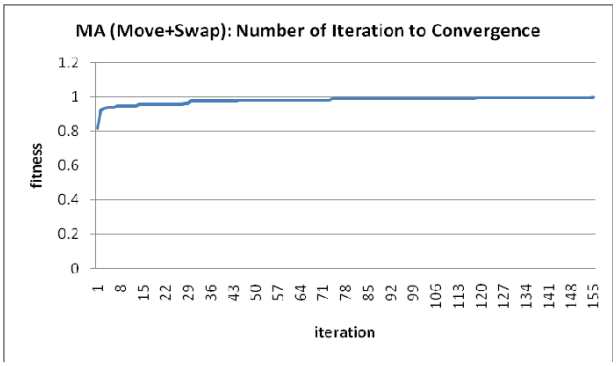

Figure 8. Number of Iteration on Problem Density

Table II shows the effectiveness of MA relative to the GA for the problem under examination. Other than indicating the effect of different neighborhood operators, the results also encourage the use of MA in solving timetabling problems since it performs significantly faster than GA on the same problem

TABLE II. RELATIVE EFFECTIVENESS OF NEIGHBORHOOD OPERATORS

Algorithm	#Iteration	Time (s)	Remarks
GA	18,122	120	Simple GA (SGA)
MA1	311	60	SGA+LS (Move)
MA2	223	78	SGA+LS (Move+Swap)

V. CONCLUSIONS

A study on the effectiveness of neighborhood operators that are used by a hill climbing local search embedded in a memetic has been presented. Two basic neighborhood operators have been investigated, i.e. neighborhood based on Move (reschedule lectures and replace rooms) and Swap (exchange the timeslot and room of two lectures). A first improvement hill climbing local search that always perform non-worsening moves by utilizing these two neighborhood operators was employed in an MA to search for feasible timetables. The results indicate that different neighborhood operator gives different performance of the corresponding MA. It apparently reveals the importance of defining neighborhood operators to suit the problems under examination. The performance of MA which surpasses the corresponding GA is also confirmed by the results which encourage the use of MA in solving timetabling problem over GA.

These promising results may be furthered demonstrated by using the MA to solve timetabling benchmark problem such as Socha et. al [15] data sets. These effort will establish the advantage of using MAs over other algorithm techniques in solving timetabling problem.

REFERENCES

[1] R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, S.Y. Lee, "A Survey of Search Methodologies and Automated Approaches in Examination Timetabling. " J. Scheduling vol 12 no 1, pp. 55-89, 2009.

[2] C.D. Stefano, A.G.B Tettamanzi, "An Evolutionary Algorithm for Solving the School Time-Tabling Problem." Proc. of Applications of Evolutionary Computing EvoWorkshop 2001, E. J. Boers, Ed. Berlin-Heidelberg: Springer Verlag, 2001.

[3] O. Rossi-Doria, B. Paechter, "A memetic algorithm for university course timetabling." Combinatorial Optimisation, Lancaster University, UK, 2004.

[4] A. Schaerf, "A survey of automated timetabling." Artificial Intelligence Review, vol. 13, pp. 87-127, 1999.

[5] M. Al-Betar, A.T.G. Khader, A.T. Gani, "A harmony search algorithm for university course timetabling," Proc. 7th Int. Conf. Pract.Theory Automated Timetabling, pp. 1–12, 2008.

[6] B. McCollum, "University timetabling: Bridging the gap between research and practice." Proc. 6th Int. Conf.Pract. TheoryAutomated Timetabling, pp. 15–35, 2006.

[7] M. Gen, R. Cheng, "Genetic Algorithms and Engineering Design." New York: Wiley, 1997.

[8] W.E. Hart, "Adaptive Global Optimization with Local Search." vol. PhD thesis San Diego: University of California, 1994.

[9] R. Dawkins, "The Selfish Gene." Oxford: Oxford University Press, 1987.

[10] E.K. Burke, D. Elliman, R. Weare, "A Hybrid genetic algorithm for highly constrained timetabling problems." Proc. of the Sixth International Conference (ICGA95), L. Esheman, Ed.: Morgan Kaufmann, pp. 605-610, 1995

[11] L. Merlot, N. Boland, B. Hughes, P. Stuckey, "A hybrid algorithm for the examination timetabling problem." Proc. The Practice and Theory of Automated Timetabling (PATAT IV). vol. 2740 Berlin: Springer, 2004.

[12] G. M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, A.J. Olson, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function." J Comp Chem, vol. 14, pp. 1639-1662, 1998.

[13] K. Ku, M. Mak, "Empirical analysis of the factors that affect the Baldwin Effec." Proc. of PPSN-V: Parallel Problem Solving From Nature, pp. 481-490, 1998.

[14] L. Di Gaspero, A. Schaerf, "A case-study for EasyLocal++: the course timetabling problem." Technical Report UDMI/13/2001/RR, Dipartimento di Matematica e Informatica - Università di Udine, 2001.

[15] O. Rossi-Doria, C. Blum, J. Knowles, M. Samples, K. Socha, B. Paechter, "A Local Search for Timetabling Problem." Proc. of the 4th Internatiomnal

Conference on Practice and Theory of Automated Timetabling, Gent, Belgium, August 21-23, 2002.

[16] M.W. Carter, G. Laporte, "Recent developments in practical course timetabling." Proc. 2nd Int. Conf.Pract. TheoryAutomated Timetabling, (Lecture Notes in Computer Science), vol. 1408, pp. 3–19, 1998.

[17] M. Jankovic, "Making a Class Schedule Using a Genetic Algorithm." online: http://www.codeproject.com/KB/recipes/GaClassSchedule.aspx. [Accessed: April May 3, 2011].

[18] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning." 1st edition, Addison-Wesley Longman Publishing Co., Inc. Boston, 1989.