Memetic Algorithm Behavior on Timetabling Infeasibility

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— **Memetic Algorithm is one of the metaheuristic techniques commonly used to solve timetabling problems due to its explorative as well as exploitative properties to produce feasible timetables. To design a memetic algorithm effectively, we need to characterize its behavior on a range of timetabling problems so that its performance can be fine tuned accordingly. In this work, we present our analysis on the memetic algorithm behavior on university course timetabling problems to produce feasible timetables. Results show the specification of Memetic Algorithm operators affect the range of problem size in which the algorithm behaves properly.**

Keywords- Hard Constraints; Infeasibility; Memetic Algorithm; Timetabling

I. INTRODUCTION

The timetabling problems represent highly constrained combinatorial optimization problems arising in various forms that differ mainly in the kind of event to be scheduled, such as educational timetabling [1], nurse scheduling [2], sport timetabling [3] and transportation timetabling [4]. Burke, Kingston and de Werra [5] gave a definition of general timetabling as a problem with four parameters: given T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constraints, the timetabling problem is to allocate times and resources to the meetings such that the constraints are satisfied. Two types of constraint, hard and soft, are normally imposed on timetabling. While hard constraints are mandatory to be satisfied, it is not necessary that the soft constraints are fulfilled. Thus, a solution is said to be feasible if it meets all the hard constraints, and a feasible solution is acceptable when most or all of the soft constraint are not violated [6].

This paper presented a study on 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 (MA) that is constituted of a standard genetic algorithm and a hill climbing algorithm is used to solve the timetabling infeasibility problems.

The principle objective of this work is to understand the underlying characteristics that may govern the relationships between timetabling problems under examinations and the MA behavior and performance. Specifically, an attempt to find out a boundary for the timetabling problem size in order that the MA behaves properly will be conducted. In addition, a simple property that may be used to measure the problem 'complexity' relative to the MA specification will also be pursued. By MA specification, it means the solution representation, fitness function, genetic operators, move operators and local search strategy that are actually implemented in the MA.

II. UNIVERSITY COURSE TIMETABLING

The complexity of university course timetabling problems has driven research in algorithm approaches, methods and techniques for solving these problems [7]. Recalling that timetabling is an NP-hard problem [8, 9], much effort has been devoted to seek for solution techniques that falls into metaheuristics, such as evolutionary techniques, simulated annealing, tabu search, genetic algorithms and other hybrid techniques.

University course timetabling consists of constructing a timetable for a set of lectures in a given time periods and the number of rooms such that a set of constraints are satisfied. Depending on the techniques used, the timetabling process may be specified as assigning a room to time-events slots [10], or assigning event to time slots and resolving the room assignment in a separate process [11]. To reflect the definition of timetabling problem better, in this work, the assignment of lectures to a time slot and a designated room is conducted at the same time.

A. The Problems

For a given number of rooms and timeslots, a timetable containing a predefined number of lectures must be constructed such that all the hard constraints are satisfied.

B. The Constraints

In this work, we only attempt to solve infeasibility in timetabling due to the imposed hard constraints. Thus, we reduce the timetabling optimization to timetabling search problems. An infeasible timetable contains violations to hard constraints, so it is not usable. The MA eliminates these violations and produces feasible timetables.

1. h_1: two lectures must not take place at the same time in the same room (no room conflict)

2. h_2: lectures must happen in a room having enough available seats (number of seat requirement)
3. h_3: lectures must occupy a room suitable for it in terms of facilities (facility requirement)
4. h_4: lectures having professors in common cannot take place at the same time (no professor's time conflict)
5. h_5: lectures having students in common cannot occur at the same time (no student's time conflict)

III. MEMETIC ALGORITHMS

Evolutionary algorithm is one of popular choice for solving timetabling problem [7]. However, it is now well-established that pure evolutionary algorithms are not well suited to fine tune search in complex optimization problems and that hybridization with other techniques can greatly improve the efficiency of search [12, 13]. Memetic Algorithms (MAs) belongs to this hybridization technique, in which it combines Evolutionary algorithm (EAs) framework with problem-specific local search heuristics [14, 15]. 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 [16]. The concept of biological evolution is captured in EAs by codifying the solutions to a given problem in so-called chromosomes. The evolution of chromosomes due to the action of cross-over, mutation and selection are simulated through computer code. In addition, the concept of meme is taken to represent a learning or development strategy (e.g. local refinement, perturbation, constructive heuristics or exact methods, etc) that are employed to improve individuals. Consequently, MAs exhibits the plasticity of individuals that a strictly genetic model fails to capture [17]. Even though MAs has become de facto standard name for this class of algorithms [18-20], the vast literature also refers these algorithms as Hybrid Genetic Algorithm [21-23], Genetic Local Search [24], Lamarckian Genetic Algorithm [25], and Baldwinian Genetic Algorithm [26].

An MA which makes use of a genetic algorithm and hill climbing algorithm has been implemented. The genetic algorithm serves as an outer frame for exploring the search space, within which a local search based on a hill climbing algorithm will exploit the best candidate solutions.

A. Solution Representation

The solution is represented by a direct representation due to [27] that allows a simultaneous assignment of lectures to timeslots and rooms during execution of the algorithm. It consists of a vector of timeslots and room (time-space) containing lectures for each hour. 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.

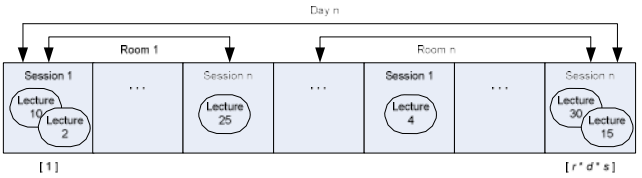

Figure 1. Timetable Solution Representation

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).

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

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

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

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.

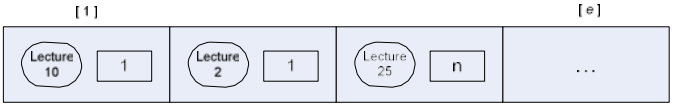

Figure 2. Chromosome Representation

B. 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.

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

For the purpose of measuring how far a timetable from being feasible, we also define a fitness distance as in (5) below. In (4) and (5), E is the number of lectures; H is the number of hard constraints; h is the hard constraint.

$$fd = 1 - \frac{\sum_{c=1}^{E}\sum_{i=1}^{H} h_{i,c}}{ExH} \quad (5)$$

C. Genetic Operator

A standard genetic operator [28] where one offspring solution is produced from two parents at each generation is used in the memetic algorithm.

1. Selection: a pair of solution is selected randomly from the population.
2. Crossover: 2-points cross over operator is used in the memetic algorithm, in which it splits both parents chromosome in parts of random size. It then alternately copies parts from these parents to produce a new child chromosome.
3. Mutation: mutation takes a lecture randomly and moves it to another randomly chosen slot. The mutation size defined in the chromosome's parameters will determine the number of lectures to be moved in a single mutation operation.

D. Neighborhood Functions

Only one basic move is considered, that is moving a lecture to an empty timeslot or room. Recalling that there are five hard constraints imposed, two kinds of move are implemented, namely MoveTime and MoveRoom. The first move (MoveTime) reschedules a lecture from one session to another session (leaving the room unchanged). This kind of move attempts to remedy the infeasibilities due to teacher's time conflict and student's time conflict. The second move (MoveRoom) replaces the room of a lecture of a given course, without changing the time. This move fixes infeasibilities related to room conflict, room capacity and room facility requirements.

E. The Memetic Algorithm

A memetic algorithm that makes use of the aforementioned representations, genetic operators and a local search utilizing the above neighborhood functions has been implemented. The algorithm is outlined below.

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

IV. Experimental Results

Four experiments have been performed by varying the values of each dimension that specify the problem size, which are the number of day, session and room. The results reveal the underlying characteristics that may govern the relationships between timetabling problems under examinations and the MA behavior and performance.

A. Experiment 1: Variable Room

This experiment is to identify the effect of varying number of room on the algorithm behavior. Logically, the more room available, the easier for the algorithm to achieve feasibility, which is confirmed by the results depicted in the Fig. 3 below.

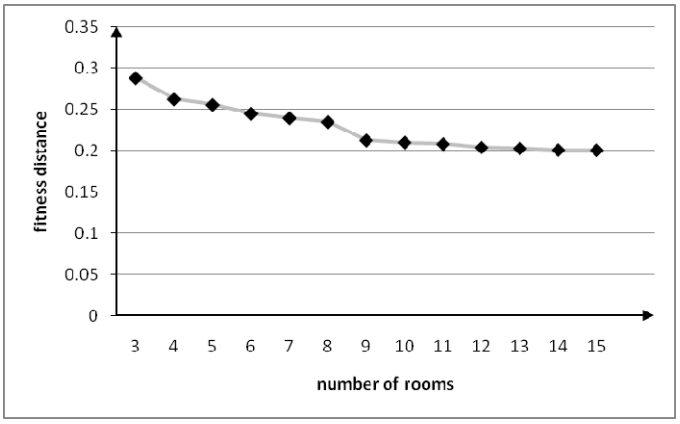

Figure 3. Fitness Distance on Variable Room

B. Experiment 2: Variable Day

This experiment is to identify the effect of varying number of day on the algorithm behavior. As it was expected, the more day available, the easier for the algorithm to achieve feasibility. Fig. 4 shows the results of this experiment.

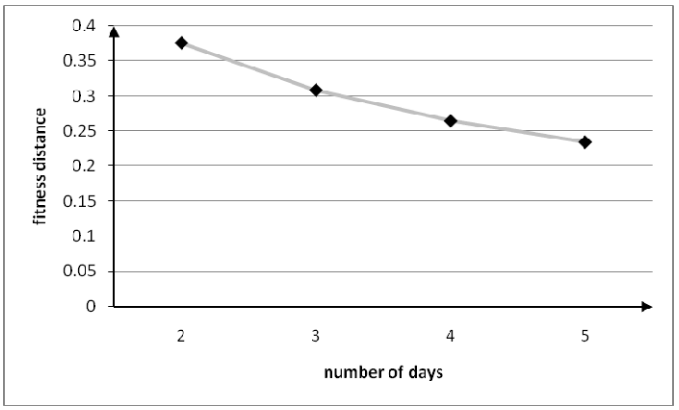

Figure 4. Fitness Distance on Variable Day

C. Experiment 3: Variable Session

This experiment is to identify the effect of varying number of session on the algorithm behavior. Since both the number session and day make up the time dimension of the problem size, similar result is brought in this experiment, that is the

more session available, the easier for the algorithm to achieve feasibility as it shown in the following Fig. 5.

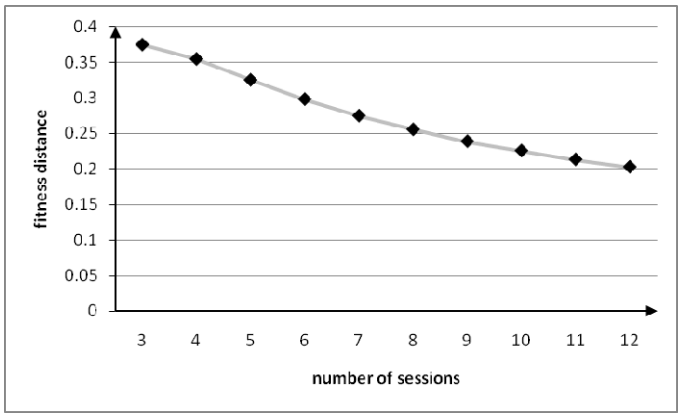

Figure 5. Fitness Distance on Variable Session

D. Experiment 4: Number of Iteration

This experiment is to identify the effect of problem size, characterized by its density, to the behavior and performance of the algorithm. We defined a density as a property of timetabling problem that indicates the proportion of total time needed for the lectures to be scheduled and the size of time-space (room) slots.

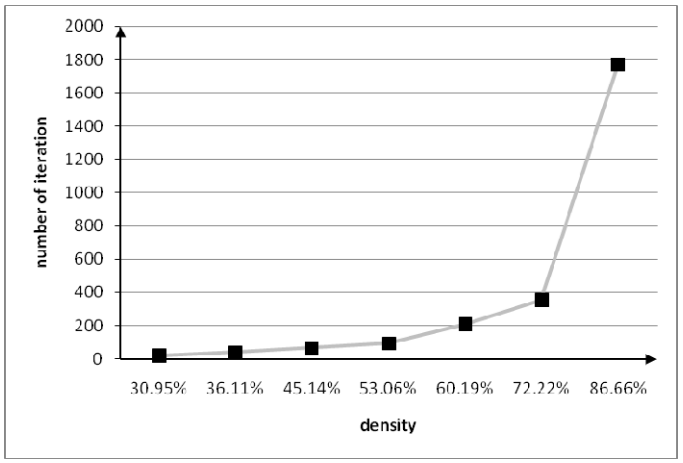

Figure 6. Number of Iteration on Problem Density

Fig. 3, 4 and 5 confirm that the implemented MA can be used to solve the specified timetabling problems. It presents results which agree to the norms of such problems justifies the correct implementation of the memetic algorithm. Fig. 6 shows some interesting results as it apparently establishes an upper bound of the density, beyond which the algorithm will not behave properly.

It turned out that density is not the only property to measure the limit of the algorithm performance relative to the problems at hand. Each dimension (time and room) that make up the total dimension (time-room) also affects the relationships between the problems and algorithm. There is a lower bound of the problem size beyond which the algorithm will behave unpredictably. We found that if either the time dimension (number of day x number of session) is below 27 or the space dimension is less than 4, the algorithm will not converge.

Relative to our current specification of the memetic algorithm, it may also be useful to characterize the problems using density property.

TABLE I. PROBLEM DENSITY AND CONVERGENCE

Case	Density	Convergence
Very Sparse	18.00%	YES
Sparse	36.00%	YES
Dense	72.00%	YES
Very Dense	86.00%	YES
Extra Dense	96.00%	NO

The above table shows that the implemented MA is not yet able to solve high density problems. Since the current neighborhood function performs a move by looking at the empty slots, in a highly dense problem, the current move operators very likely fail to find such empty slots. In this situation, the more useful type of move is swapping lectures in a different time-space slots.

V. CONCLUSIONS AND FUTURE WORKS

A memetic algorithm behavior and performance over a range of timetabling infeasibility problems has been presented, which will be useful for designing the algorithm to suit the problems better. The results not only reveal the limit of the current algorithm specification, but also show that the timetabling problems exhibit a property relative to the algorithm itself. This is useful to measure their relative 'complexity'. This density property can be used to characterize the upper bound for the algorithm specification to behave properly. Interestingly, the lower bound are set by time (number of session and day) as well as room dimension that make up the timetabling problem size under examination.

These results confirmed that the current move operators are not sufficient for solving infeasibility over a wide range of timetabling problems. Instead, it encourages the introduction of another type of move, which is swapping lectures in different timeslot and room. This swap move, along with some heuristics to perform a swap, is the subject of our memetic algorithm specification in the next phase of the research.

REFERENCES

[1] D. de Werra, "An Introduction to timetabling," *European Journal of Operational Research*, vol. 9, pp. 151-162, 1985.

[2] E. K. Burke, P. D. Causmaecker, G. V. Berghe, and H. V. Landeghem, "The state of the art of nurse rostering," *Journal of Scheduling*, vol. 7, pp. 441- 499, 2004.

[3] K. Easton, G. Nemhauser, and M. Trick, "Sports scheduling," in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J. Leung, Ed. Florida: CRC Press, 2004, pp. 52.1-52.16.

[4] R. Kwan, "Bus and train driver scheduling," in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J. Leung, Ed. Florida: CRC Press, 2004, pp. 51.1-51.18.

[5] E. K. Burke, J. H. Kingston, and D. d. Werra, “Applications to timetabling,” in *The Handbook of Graph Theory*, J. Gross and J. Yellen, Eds. Florida: Chapman Hall/CRC Press, 2004, pp. 445-474.

[6] C. D. Stefano and A. G. B. Tettamanzi, “An Evolutionary Algorithm for Solving the School Time-Tabling Problem,” in *Proceeding of Applications of Evolutionary Computing EvoWorkshop 2001*, E. J. Boers, Ed. Berlin-Heidelberg: Springer Verlag, 2001.

[7] A. Schaerf, “A survey of automated timetabling,” *Artificial Intelligence Review*, vol. 13, pp. 87-127, 1999.

[8] S. Even, A. Itai, and A. Shamir, “On the complexity of timetable and multicommodity flow problems,” *SIAM Journal of Computation*, vol. 5, pp. 691-703, 1976.

[9] T. B. Cooper and J. H. Kingston, “The complexity of timetable construction problems,” in *Proceedings of Practice and Theory of Automated Timetabling (PATAT-95)*, E. K. Burke and P. Ross, Eds. Berlin-Heidelberg: Springer-Verlag, 1996, pp. 283-295.

[10] L. Di Gaspero and 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

[11] O.Rossi-Doria, C.Blum, J. Knowles, M. Samples, K. Socha, B. Paechter, “A Local Search for Timetabling Problem,” in Proceedings of the 4th Internatiomnal Conference on Practice and Theory of Automated Timetabling 2002, Gent, Belgium, August 21-23.

[12] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[13] D. Goldberg and S. Voessner, “Optimizing global-local search hybrids,” in *Proceeding of GECCO-99 (Genetic and Evolutionary Computation Conference)*, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakaiela, and R. Smith, Eds.: Morgan Kaufmann, 1999, pp. 220-228.

[14] P. Moscato, “On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms,” Concurrent Computation Program Caltech, California 1989.

[15] Y.S. Ong, M.H. Lim, N. Zhu, K.W. Wong, “Classification of Adaptive Memetic Algorithms: A Comparative Study,” *IEEE Transactions of Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 1, February 2006, pp. 141-152.

[16] R. Dawkins, *The Selfish Gene*. Oxford: Oxford University Press, 1987.

[17] N. Krasnogor and J. Smith, “Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues,” *IEEE Transactions on Evolutionary Computation*, vol. A, 2005.

[18] P. Moscato, “Memetic algorithms: A short introduction,” in *New Ideas in Optimization*, D. Corne, F. Glover, and M. Dorigo, Eds. New York: McGraw- Hill, 1999, pp. 219-234.

[19] J. W. Hart, N. Krasnogor, and J. Smith, *Recent Advances in Memetic Algorithms*. Berlin-Heidelberg: Springer, 2004.

[20] J. Smith, W. Hart, and N. Krasnogor, “Special Issue on Memetic Algorithms,” *Evolutionary Computation*, vol. 12, pp. 273-353, 2004.

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

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

[23] M. Vazquez and L. Whitley, “A hybrid genetic algorithm for the quadratic assignment problem,” in *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds.: Morgan Kaufmann, 2000, pp. 135-142.

[24] P. Merz, “Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies,” in *Department of Electrical Engineering and Computer Science*. vol. PhD thesis: University of Siegen, 2000.

[25] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and 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.

[26] K. Ku and M. Mak, “Empirical analysis of the factors that affect the Baldwin Effect,” in *The Proceeding of PPSN-V: Parallel Problem Solving From Nature*, 1998, pp. 481-490.

[27] M., Jankovic, “Making a Class Schedule Using a Genetic Algorithm,” Available: http://www.codeproject.com/KB/recipes/GaClassSchedule.aspx. [Accessed: April May 3, 2011], (2008)

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