University Course Scheduling using the Evolutionary Algorithm[*)]

A. Jamal

Informatics Engineering Department, Faculty of Science and Technology,
University of Al Azhar Indonesia, Jl.Sisingamangaraja, Jakarta, 12110
Tel.62-21-7244456, fax. 62-21-7244767, email : adja@uai.ac.id

Abstract - **Course scheduling problem is hard and time-consuming to solve which is commonly faced by academic administrator at least two times every year. This problem can be solved using search and optimization technique with many constraints. This problem has been well studied in the past, and still becomes favorite subject for researchers. We will briefly discuss the convergence difficulty in our initial work on this subject using a modified hill-climbing search technique[8]. In this paper, an evolutionary algorithm is applied to solve the course scheduling problem and studying mutation techniques involved in the algorithm.**

Keywords - *Course Scheduling, Optimization, Evolutionary Algorithms Genetic Algorithms*

I. INTRODUCTION

Building a course schedule is one of the main challenge at university that must be faced by academic administrator every semester. This problem is considered as an NP-complete problem [2,3] which is quite difficult and time-consuming to solve. This problem has been the subject of extensive research effort due to its complexity and wide application such as school timetables [1], exam scheduling [3] and course scheduling [2,5,8,9,11]. The importances of seeking for good technique to solve this kind of problem are realized by some educational institutions by organizing International Timetabling Competition[1].

The university course scheduling problem is the task of assigning the academic events (such as lectures, tutorials etc) to room and time slots in such a way that taking consideration a predefined set of constraints. Every university may have different constraints. However these constraints usually can be classified as two types, namely hard and soft constraints. Hard constraint must not be violated to construct a valid or feasible schedule, while soft constraints are desired but not absolutely to be fulfilled.

A number of algorithms have been used to solve the course scheduling problem. The graph coloring heuristic techniques whereby course are assigned to rooms and time-slots one by one in particular order are the earliest approaches which are very efficient in small scheduling problem. The second approaches are the local search algorithm family that basically perform search in neighborhood of a known solution state rather than exploring possible solution in wider search space. The most popular local search method are simulated annealing method [1] and tabu search method [5]. The third type is the evolutionary algorithms and genetic algorithms which is based on Darwinian evolutionary theory [3,9,11].

In the current work we use evolutionary algorithm to solve the course scheduling problem, focusing more deeply in reproduction mechanism. This paper is organized as follows: section two described the course scheduling problem based on constraints and its representation model. Section three discuses the algorithms and its applicability to the course scheduling problem. Section four discusses the experimental results and compares it to the previous work [8]. In the last section we present a conclusion and recommendation for future work.

[*)] This paper had been presented at 2nd ICSIIT 2010 International Conference on Soft Computing, Intelligent System and Information System - UNIKA Petra

[1] The International Timetabling Competition 2002- http://www.idsia.ch.
The International Timetabling Competition 2008 sponsored by PATAT and WATT (http://www.cs.qub.ac.uk/itc2007)

II. COURSE SCHEDULING PROBLEM

2.1 Problem definition

Despite of different constraints, definition of university course scheduling problem can vary depend on whether it is based on post student enrollment where a set of student attending each event are defined [2,9,11] or based on curriculum for each faculty where scheduling takes place prior enrollment [10,11]. Since our university (i.e. University Al-Azhar Indonesia) conforms to the curriculum based scheduling, our work stick with it.

Hence the university course scheduling problem is defined as follows: There is a set of room which has a seat capacity and contains specific feature (i.e. laboratories), a set of course which is based on curriculum for each program and may have multiple section (i.e. credit unit where one course section occupy one time-slot), a set of lecturer which has been assigned to specific course(s) and has a certain unavailable time-slot. A set of events (i.e. classes), to be scheduled in a certain number of time-slots or time-period and a room, is defined as combination of a specific course with assigned lecturer attended by certain number of student from a particular group (i.e. a student group come from a specific program and same grade).
A feasible schedule is one in which all the classes have been assigned to a time-slot and a room whereby the following hard constraints are satisfied:

1. rooms must not be double booked for classes at any feasible time-slot (classroom clashed)
2. the room capacity must not be exceeded by the number of attending student
3. the room has a feature (i.e. laboratory) required by the classification
4. a lecturer can not teach more than one one class at the same time
5. a lecturer can not teach any class in time-slot which is unavailable for him/her
6. a student group from the same program and same grade can not attend more than one class at the same time
7. a class with multiple section must be assigned in the same room contiguously.

A number of soft constraints that should be minimized could introduced such as a lecturer should be assigned in his/her preference time-slot and room or parallel classes (i.e. the same courses taught by different lecturer) should be scheduled at the same time. However, in the present work we will not take into account soft constraints. The main objective is first to find a feasible course schedule, and then this feasible schedule to be optimized with respect to soft constraints in the next phase.

2.2 Scheduling Model

Deducted from previous published works, there are also two different approaches were used to represent the scheduling model. In the first approach, the scheduling model is represented by a two dimensional matrix where each row corresponds to room, each column to a time-slot, and then the matrix element contains a particular event or blank. This approach is usually used for post-enrollment scheduling problems [2,3,9,11]. In the second approach, a scheduling model is represented directly by a triple of <event, room, time-slot>. Note that an event in the second approach is a combination of course and initially assigned teacher. The second approach is found in the prior-enrollment scheduling publications [10,11].

Though these two different approaches corresponds respectively to two different scheduling problems, but none of the previous authors explained the correlation between the model and the problem, even Pawel Myszkowski [11] who considered both problem variations and used both approached respectively,

The presented work invoke a quite different approach. Both kind of approaches (i.e. matrix and an array of tuple) are used for the sake of ease evolutionary mutation and fitness evaluation. Furthermore instead of using a two-dimensional matrix we explode it to three-dimensional matrix by remodeling a one-dimensional column which represents time-slot into a two-dimensional matrix where each row represent day and each column as hour. In contrary to some previously published work [3,9], where the length of time slot can vary, here the size of matrix is predefined by the problem. Hence, the first hard constraint is completely satisfied (i.e. only one class is scheduled in each at any feasible time-slot)

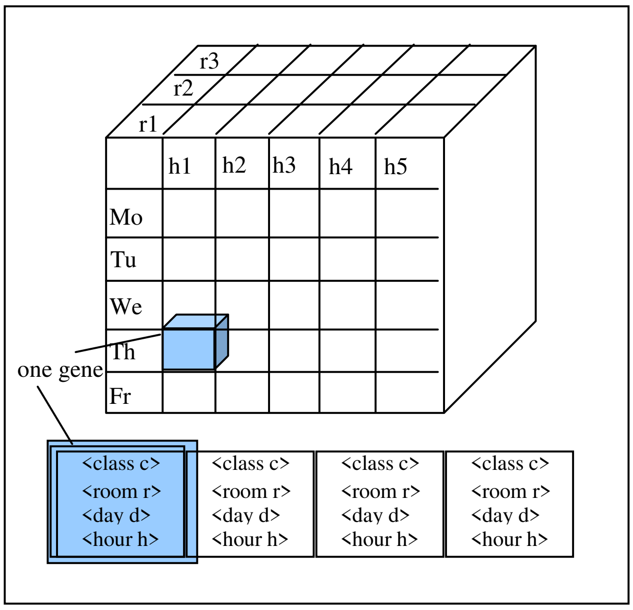
r3
r2
r1
h1
h2
h3
h4
h5
Mo
Tu
We
Th
Fr
one gene
<class c> <room r> <day d> <hour h>
<class c> <room r> <day d> <hour h>
<class c> <room r> <day d> <hour h>
<class c> <room r> <day d> <hour h>

Figure 1. Matrix based - and tuple based schedule model

III. EVOLUTIONARY ALGORITHM

Evolutionary algorithms are population based meta-heuristic optimization algorithm inspired by biological evolution that used mechanisms like mutation, crossover, natural selection and survival of the fittest in order to refine a set of solution candidates iteratively. The first version of the evolutionary algorithm introduced by Rechenberg (1965) where he began with a parent and a mutated one, whichever is the fittest became a new parent [7], hence only one species involves in every generation. In our previous work, we used the same strategy with exception that the number of mutant version is larger [8]. Although the convergence rate to the solution is found to be very fast in the previous work, but this kind of local search is often stuck in a local optimum. In the current work, we will incorporate population for each generation in order to get the necessary diversity in the solution candidates.

The matrix based and array of tuple based scheduling models defined in the previous section represent an individual chromosome where an element of matrix or a single tuple represents a gene as shown in figure 1. A kind of permutation encoding is used for this purpose where each matrix element or each tuple representing a gene will contain an event or class index. Noted that a course taught in more than an hour will be presented in multiple section or multiple genes of same indexes. The permutation encoding in matrix form reveals a straight forward decoding, or strictly speaking no decoding is necessary.

Using a permutation like encoding has another issue in the crossover mechanism, namely, we have to check whether genes in the half chromosome from one parent are also found in the other half chromosome from the other parent. This extra work requires a large computation effort. Therefore, a crossover-like mechanism within one chromosome is applied, namely swapping a sub-set of scheduled classes. However since this mechanism involves only one parent, precisely speaking it is a mutation mechanism.

Initial population is generated by randomly mutating one very first chromosome as many as number of population. The first chromosome is constructed in such way that all events are scheduled in the schedule matrix and take into account that classes with multiple section are scheduled on the same room and contiguously. Hence it fulfill the first and the seventh hard constraint in the previous section. These two hard constraints are not only used in constructing the first chromosome but also considered in the mutation mechanism, thus we name them fixed constraints. The algorithm used for constructing the first chromosome is based on greedy algorithm by course hours.

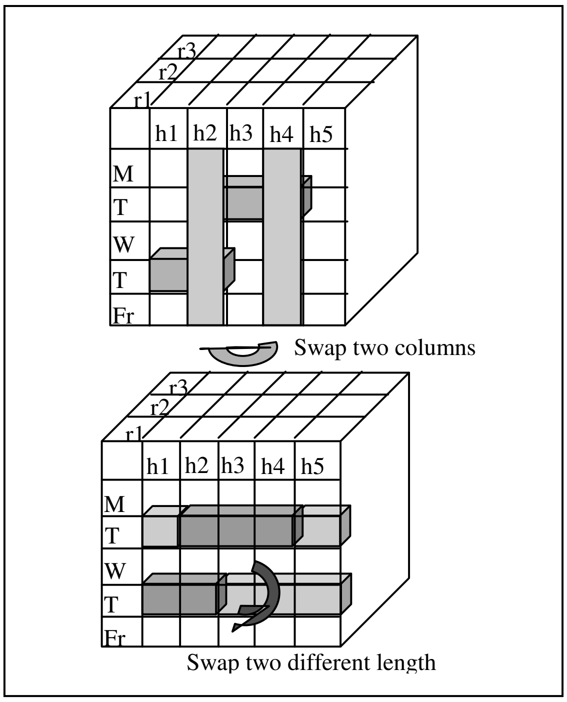

Figure 2 Prohibited mutations

Consequences of considering fixed constraints in the mutation scheme, swapping column wise (changing hour) is prohibited because it would destroy the contiguity of multiple section (multiple hour) classes. Swapping two classes with different number of sections (hours) is sometimes impossible in case insufficient time-slot for longer class in the new room and day to be placed. These two prohibited mutations are shown in figure 2. As in our previous work [8], only two mutation operators are used, namely swapping two arbitrary events (classes) and swapping all events that scheduled on two arbitrary days and rooms.

The fitness of each chromosome depends on the amount of violated hard constraints. A value of 100 is added to the fitness if all event completely comply to one type of hard constraint. This maximum fitness value decreases proportionally with the amount of constraint violation. Every constraint will be evaluate separately for its fitness. Since there are 5 hard constraints left in our problem, the maximum total fitness value to be sought is 500.

All chromosome in the population are ranked according to its fitness. A group of elite chromosome, i.e. the most fittest will be kept for the next generation. The number of this elite group is only a small fraction of the population size. The other survival will come from further evolutionary mechanism, namely selection of parents based on roulette wheel techniques, and then applying both mutating operators with certain probabilities on the selected parents. Replacement of the least fittest chromosomes by elite group finalizes a creation of new generation, and this process is repeated until maximum fitness (valid solution) is found.

IV. EXPERIMENTAL ANALYSIS

For experimental analysis purpose we take a relative small test set of 75 classes (179 course hours) to be scheduled in 4 rooms (192 hour available time-slots). Using local search algorithms (i.e. modified hill-climbing search) from our previous work [8], this test set yields about 40% convergence failure. However, it need only less than 50 iterations or generation when it successfully converged as depicted in figure 3. The reason of high rate of convergence failure is lack of diversity and relatively narrow exploring area search in the used local search techniques which causes the process reaches local maximum quickly.

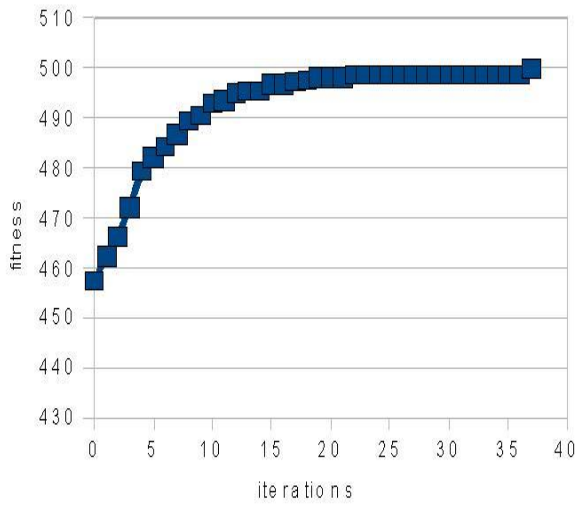

Figure 3. Fitness variation as function of generations produced by modified Hill-climbing search [8]

Incorporating a group of individuals or a population in the present work induces some parameters to be determined. Those parameters are number of population Np, probabilistic rates for mutation of swapping two classes Pc, and probabilistic rates for mutation of swapping two scheduled days/rooms Pd, and percentage of elitism Pe. The first three parameters influence the diversity force of the evolutionary algorithms while the percentage of elitism and together with selection techniques effect the force of pushing quality [4]. Effect of population size is very clear, namely larger size yields more diversity. Percentage of elitism is more or less also quite predictable, namely too much elitism causes less diversity. In this study we took Np=200 and relative small elitism Pe=20%.

The other two parameters, regarding mutations probabilistic, which can result in good convergence rates should be further studied to be chosen.

Hence the objective of present experiment are studying mutation parameters. Fixing the probabilistic rate of swapping two scheduled days/rooms Pd=60%, effect of the probabilistic rate of swapping two classes Pc is studied. We run the program a couple times until either the maximum fitness was found or the limit number of generation was reached for a certain value of PC, and then repeat the process by varying the Pc. The following table presents the statistic results acquired from twelve runs for each parameters value of Pc. The values shown in the table are the number of generation needed before maximum fitness value was reached for four basic statistic parameters (I.e average, standard deviation, min and max) and the percentage of failure attempts from the first twelve runs.

Table 1. Effect of probabilistic rate of swapping two classes on number of generations

Pc	**1**	**0.8**	**0.6**	**0.4**	**0.2**
Average	295	438	645	574	710
Std Dev	191	469	498	301	292
Min	156	179	240	270	352
Max	871	1771	1620	1318	1176
Failure	0	0	0	0	50.00%

Fixing the probabilistic rate of swapping two classes Pc=60% we vary the Pd as shown in the following table

Table 2. Effect of probabilistic rate of swapping two scheduled day on number of generations

Pd	**0.6**	**0.4**	**0.2**	**0.1**	**0**
Average	645	267	186	344	551
Std Dev	498	138	37	312	355
Min	240	144	135	144	245
Max	1620	630	256	1081	1190
Failure	0	0	0	0	30.00%

The results has shown that deviation and average generation significantly effected by these two parameters Pd and Pc. Mutation by swapping two classes (Pc) is required more than mutation by swapping two scheduled day. Furthermore, Pc less than 20% results in convergence failure rate above 50% , while Pd =0 (no mutation by swapping two scheduled day) still yields successful convergence about 70% from twelve attempts. This happened because the swapping two classes is the diversity force for exploitation in detail of successor solution state, while swapping two scheduled day for exploration in wider solution space around the successor. Our test has shown the Pd=20% and Pc=60% give the best result so far.

Experimental results have also shown that the fitness value approaches quickly to the maximum value after about the average number of generations as depicted in figure 4. Thereafter the fitness curve flattens up until certain generation increases again to maximum values. This phenomenon is not found in another run for the same case (i.e. the same test set and same evolutionary parameters) where maximum fitness value is found at about average number of generations as pictured in figure 5. In this case, the fitness values is steadily increasing.

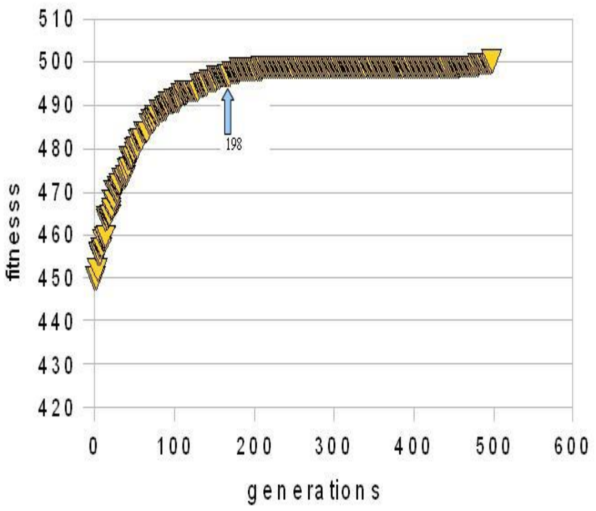

Figure 4. Fitness variation as function of generations produced by successful run after 498 generations, for Pc=1.0, Pd=0.2 where average number of generation is 198.

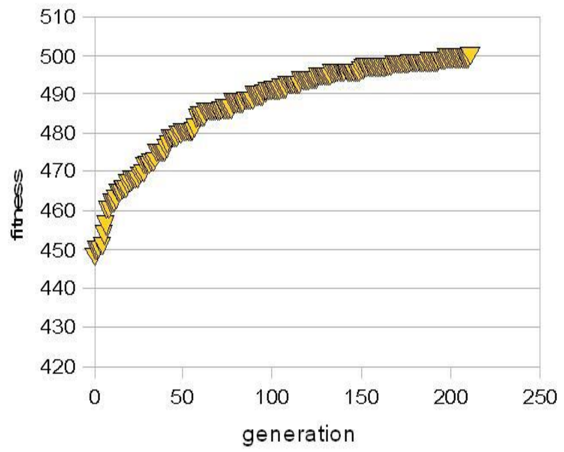

Figure 5. Fitness variation as function of generations produced by successful run after 211 generations, for Pc=1.0, Pd=0.2 where average number of generation is 198.

V. CONCLUSION

We have presented the evolutionary algorithm for solving of university course scheduling problem. This study has shown that the probabilistic rate of two mutation types has significant effect on successful rate of the algorithm. This evolutionary algorithm can find feasible or valid course schedules very good by using small probabilistic rates for mutation of swapping two scheduled days/rooms Pd and large probabilistic rates for mutation of swapping two classes Pc.

The mutation by swapping two scheduled days/rooms which is the force for exploration in wider solution space around the successor can good replace the real crossover mechanisms between two parents.

The mutation by swapping two classes is absolutely needed to ensure convergence. This is the force for exploitation in the vicinity of global optimum solution. Some experimental results have shown that this exploitation force need more time in one case than the others.

For future work, we aim to take into account the soft constraints by introducing second stage local optimization.

REFERENCES

[1] Abramson, D. (1991) Constructing School Timetables using Simulated Annealing: Parallel and Sequential Solutions", Management Science, Vol. 37, No. 1, , January, 1991, 98-113

[2] Al-Betar M.A., Khader A.T. and Gani T.A. (2008) A Harmony Search Algorithm for University Course Timetabling. In: Burke E., Gendreau M. (eds.). The Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, Montréal, Canada, 2008.

[3] Burke, E.K., Elliman, D.G. and Weare, R.F. (1994) A Genetic Algorithm based University Timetabling System, In Proceedings of the 2nd East-West International Conference on Computer Technologies in Education, Sept, 1994, Crimea, Ukraine, 35-40

[4] Eiben, A.E. and Smith, J.E. (2007) Introduction to Evolutionary Computing, Natural Computing Series 2nd Edition, Springer

[5] Elloumi, A., Kamoun, H. and Ferland, J.(2008) A Tabu Search for Course Timetabling Problem at a Tunisian, in Proceeding of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT '08, Edmund K Burke and Michel Gendreau (eds), August 2008

[6] Elmohamed, M.A.S., Fox, G. and Coddington, P.(1998) A Comparison of Annealing techniques for Academic Course Scheduling", DHPC-045, SCSS-777, 1998

[7] Haupt, R.L. And Haupt, S.E.(2004) Practical Genetic Algorithms, Wiley-InterScience 2nd Edition

[8] Jamal, A. (2008) Solving University Course Scheduling Problem using Improved Hill Climbing Approach, In Proceeding of the International Joint Seminar in Engineering, August 2008, Jakarta, Indonesia

[9] Lewis, R. and Paechter, B. (2005) Application of the Grouping Genetic Algorithm to University Course Timetabling, In G. Raidl and J. Gottlieb (eds) Evolutionary Computation in Combinatorial Optimization, Berlin Germany, Springer LNCS 3448, pages 144-153

[10] Moody, D., Kendall, G. and Bar-Noy, A.(2008) Constructing initial neighborhoods to identify critical constraints, In Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT '08, Edmund K Burke and Michel Gendreau (eds), August 2008

[11] Myszkowski, P. and Norbeciak, M. (2003) Evolutionary Algorithms for Timetable Problems. Annales UMCS Informatica AI, 2003, 115-125. DOI= http://www.annales.umc.lublin.pl/