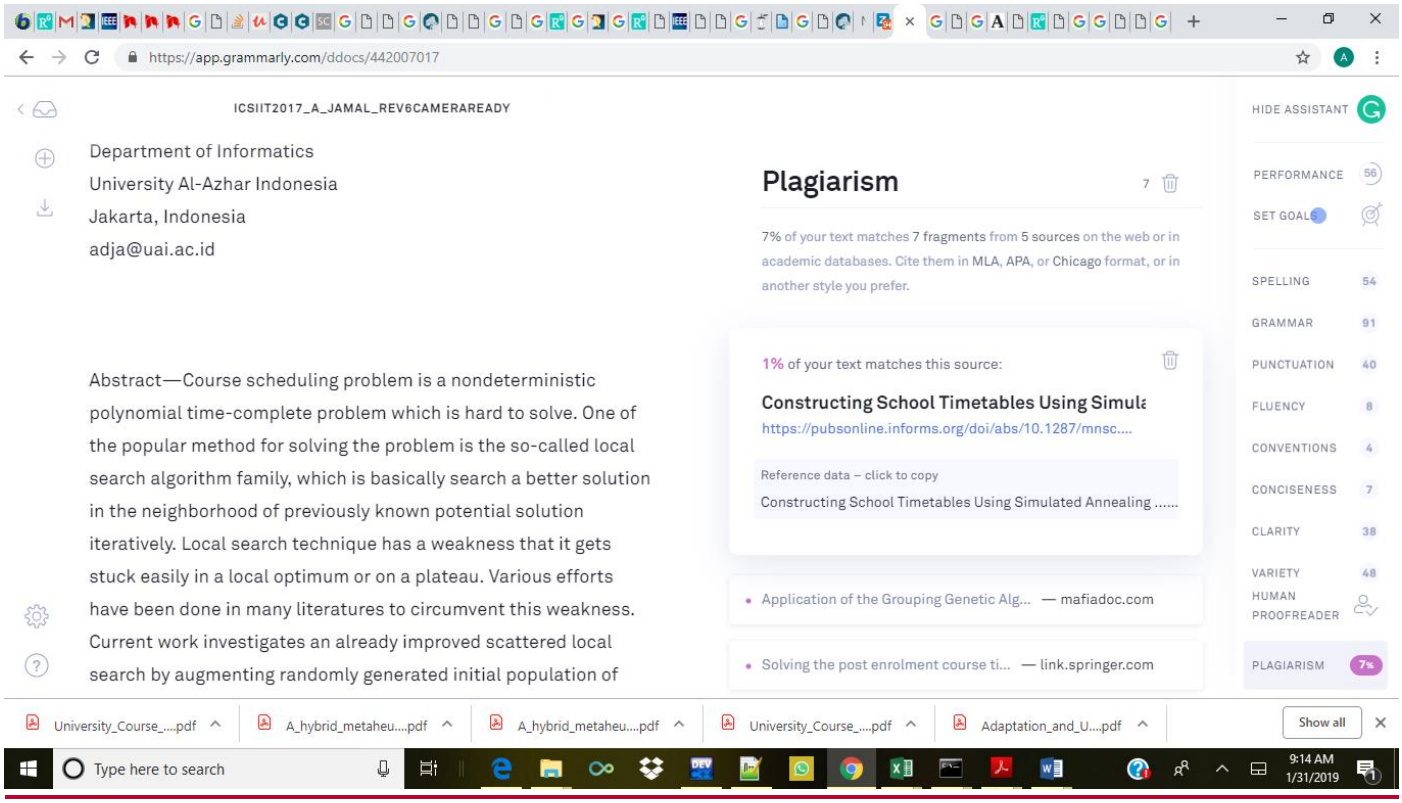

Multiple Scattered Local Search for Course Scheduling Problem

A. Jamal
Department of Informatics
University Al-Azhar Indonesia
Jakarta, Indonesia
adja@uai.ac.id

***Abstract*—Course scheduling problem is a nondeterministic polynomial time-complete problem which is hard to solve. One of the popular method for solving the problem is the so-called local search algorithm family, which is basically search a better solution in the neighborhood of previously known potential solution iteratively. Local search technique has a weakness that it gets stuck easily in a local optimum or on a plateau. Various efforts have been done in many literatures to circumvent this weakness. Current work investigates an already improved scattered local search by augmenting randomly generated initial population of schedules in the local search algorithm. This algorithm will be compared to an originally population based genetic algorithm which is enhanced by letting local search technique on each individual schedule in every generation. The result has shown that the population based local search performs better than the hybrid genetic algorithm.**

Keywords—Course Scheduling, Optimization, local search Algorithms, Genetic Algorithms

I. INTRODUCTION

Constructing a course schedule is one of the main challenge within a university that must be performed in every academic

period. This problem is considered as a nondeterministic polynomial time-complete problem [1,2] which is quite difficult and time-consuming to solve. This is a multi-dimensional and multi-constrained combinatorial optimization problem, that has been being a subject of extensive research efforts due to its complexity and wide application such as school timetables [3], exam scheduling [2] and course scheduling [1,4,5,6,7,8].

Course scheduling process involves placements of many events in available time and space resources where initially specified constraints must be satisfied. In this study, an event could be a teaching class or a laboratory class where a combination of lecturers or instructors, student-groups and a course was designated beforehand. This type of course scheduling is called a curriculum based course scheduling or timetabling wherein a student-group is associated to the curriculums [9]. Scheduling problem could also include selections of courses that should be taught by which lecturer. Another type of scheduling involves every individual student who has enrolled for their courses. These last two types of course scheduling are not considered in the present study.

There are two sets of predefined constraints which must be deliberated by the university course scheduling problem. The first set of constraints is named hard constraints which must not be violated to construct a valid or feasible schedule. The second set is soft constraints which are desired but not necessarily to be fulfilled. Hence, finding the valid or feasible schedule is merely a search problem rather than an optimization problem. However, fulfilling the soft constraints is a real optimization process in this study.

Course scheduling problem has been solved by wide range type of algorithms. The most traditional one is a so-called graph coloring heuristic method whereby course are assigned to rooms and time-slots one by one in particular order. The second type falls into the local search algorithm family [8] whereby searching is carried out in neighborhood of a known state rather than exploring search space extensively. This type of algorithm is quite popular among the researcher who recommended enhanced variant of local search algorithm such as simulated annealing method [3,5], tabu search method [4] and hill-climbing search method [6].

The last type of algorithm is based on population mechanism which starts with many different solutions and explores possible solution in wider search space. The most popular population based algorithms used to solve course scheduling are evolutionary algorithms [10], genetic algorithm [2,7,9], and harmony algorithm [1].

Due to two different criteria in the course scheduling problem, i.e. hard constraints that governs the search problem and soft constraints that constitutes the objective function of the optimization problem, various approaches can be found in the published research works. Combining these two set constraints with weight factors is the most frequently used approach, wherein the weight factor of soft constraint is just small fraction of hard constraint weight factor. Another popular approach is a staging method wherein a fulfilling of hard constraints in the first stage followed by an optimization process to gain as many as possible soft constraint satisfactions [7]. Some of the researchers used hybrid algorithms to tackle this staging method [9,11].

In the previous study, we had incorporated evolutionary algorithm (EA) [12] and the improved scattered hill-climbing search method (SHC) [6] into three stage method [11]. We had also compared these two algorithm separately, and proved that SHC is faster than EA in case SHC could find the feasible schedule [13]. Though, the probability of success in searching the feasible solution, i.e. the hard constraint free schedule is not more than 30% for SHC. This is caused by the nature of local search of SHC which is good in exploiting the possible solution in the neighborhood of a solution state. In contrary, the EA is good in exploring possible solution by generating new successor state by mutation and cross-over two parent states. Hence the probability of success in searching the feasible schedule is much better, but it needs much more iteration, i.e. number of generation because the nature of probabilistic process in EA.

Knowing the above mentioned phenomenon, a number of SHC method from randomly generated initial states will be conducted to obtain higher probability of success in finding feasible schedules. In case of EA, the exploitation capability will be embedded in each member of population before the evolution process started in every new generation.

II. Solution Method for Course Scheduling Problem

A. Course Scheduling Problem

The curriculum based course scheduling problem consist of the following entities:

- a set of room $\{r_1, r_2, \ldots r_{noR}\}$ which has a seat capacity and specified as a lecture- or an interactive computer classroom,
- a set of time-slot $\{t_1, t_2, \ldots t_{noT}\}$ wherein lecturers has preferences to teach,
- a set of course $\{c_1, c_2, \ldots c_{noC}\}$ which is based on curriculum for each student group and may have multiple section (i.e. credit unit where one course section occupies more than one time-slot),
- a set of lecturer $\{l_1, l_2, \ldots l_{noL}\}$ which has been assigned to specific course(s) and has a certain unavailable time-slot and preference time-slot,
- a set of student group $\{s_1, s_2, \ldots s_{noS}\}$ with associated number of student from a specific program and same grade,
- a set of events (i.e. classes) $\{e_1, e_2, \ldots e_{noE}\}$, to be scheduled in a certain number of time-slots and a room, is defined as combination of a specific course with assigned lecturer attended by certain number of student group.

A course schedule is said to be feasible and valid when the following hard constraints are satisfied:

This article is sponsored by LP2M UAI through International Seminar Grant.

- all events have been assigned to a time-slot and a room (complete schedule),
- no room is occupied by more than one event at the same time (room conflict),
- no lecturer teaches more than one event at the same time (lecturer conflict),
- no student group from the same program and same grade attends more than one event at the same time (student conflict),
- number of attending students in the student group must not exceed the room capacity,
- room has a feature (e.g. laboratory) required by the assigned course,
- no lecturer teaches in time-slot which is unavailable for him/her,
- an event with a multiple section course must be assigned in the same room contiguously (continue event).

There are soft constraints that are preferably fulfilled to get a so-called optimum schedule. Some examples of these soft constraints are as follows:

- a lecturer should be assigned in his/her preference time-slot
- a lecturer should not be assigned in his/her avoiding time-slot
- a classroom should not be half empty, i.e. the number of attending student less than a half of room capacity
- no student should be scheduled to sit more than three events on the same day
- minimize scheduled events in the last time-slot of a day
- morning class-hours are preferred than afternoon class-hours
- minimize unoccupied room between two occupied time-slots

The first three of the above mentioned soft constraints will be taken into account in the presented work.

B. Course Scheduling Model

Course schedules are represented in a variety of models. References [1,2,7,10] uses a two dimensional matrix where rooms and timeslots are represented by row and column and each cell of matrix represents a single event. In the previously published works [6,11,12,13] we extended the dimensional matrix into three dimensional work by remodeling the one dimensional timeslot representing column into a two dimensional matrix where each row corresponds to day and column to hour. Another approach is direct representation, i.e. each gene is represented by a triple of event, room and time-slot $<e_i, r_j, t_k>$ [8, 10].

The matrix model governs that the "room conflict" hard constraint is always fulfilled. However, generating the initial state is quite difficult when all events must be assigned into the cells taken into account multiple section event must be assigned in the same room contiguously. Therefore, the matrix model is not applicable for the present work since it requires randomly generated initial state for SHC. However, instead of using direct representation of triple model, we introduce tuple of two:

$$<e_i, f(r_j, t_k)> \quad (1)$$

where space (room)- and time resources are formulated as a single value function of resources $f(r_j, t_k)$ whose value ranging from 1 to multiplication of *noR* and *noT*, i.e. number of rooms and timeslots, respectively. Generating random initial state of schedule is as easy as generating random number *f* in this range for each event e_i.

C. Scheduling Algorithm

The original hill-climbing search method is a not population based technique. The fore knowledge that this kind of local search method gets stuck frequently in a local optimum as we had shown in [13], lead us to invoke a population of randomly generated schedules performing the scattered hill-climbing search method. This population based scattered hill climbing search, shortly hereafter named a multiple scattered local search, will proceed for certain number of iterations until one or more individual schedules from the population become feasible and valid, i.e. none of hard constraints are violated. While the iteration is still in progress, course schedules that already feasible will further undergo in the second stage, searching for the optimum solution in the neighborhood of the found feasible state.

Algorithm for a one-step single scattered hill-climbing search is shown in Fig. 1.

```
function SCATTERED-HILL-CLIMBING(state)
  inputs:          state: a schedule
  local variables: new, neighbor: a schedule
                   penalty: hard constraint violation
  returns:         new state which has less violation
                   on hard constraint
  penalty ← HC-SC-PENALTY(state) -
  new ← state
  for i ← 1 to neighborhood size do
      neighbor ← MUTATE(state)
      if HC-SC-PENALTY(neighbor) ≤ penalty do
          new ← neighbor
          penalty ← HC-SC-PENALTY(neighbor)
      end if
  end for
  return new
```

Fig. 1. Scattered hill-climbing algortihm

In the present work, this function is invoked for each schedule in the population. Note that the function of HC-SC-penalty is a combination of hard constraint violation ($penalty_{hc}$) reduced by a small part of soft constraint fulfillment ($score_{sc}$)

as given in (2). The soft constraint fulfillment is an optional in this part of algorithm but it turns out that its existence is very significant to improve the effectiveness of the algorithm.

$$HC\text{-}SC\text{-}penalty = weight_{hc} * penalty_{hc} - weight_{sc} * score_{sc} \quad (2)$$

Any schedule that reached the zero violation of hard constraint, will further undergo a local search algorithm as depicted in Fig. 2 to maximize the fulfillment of soft constraints (SC-SCORE). The multiple scattered local search iteration will stop when a specified number of schedules become feasible or maximum iteration is reached.

Performing the scattered local search in a population scheme rather than a single schedule gives more explorative capability in this local search method, of course at cost of extra computation time. To gain a better view whether this extra cost worthy, we will compare the performance of MSLS (multiple scattered local search) with the evolutionary algorithm [12] using the same two tuple model. We add also some exploitation booster by letting each chromosome at every new generation in the evolution process does the same single scattered hill-climbing search from Fig. 1. This addition of hill-climbing search into the standard genetic algorithm is also used in [14] to minimize two objective functions, i.e. soft constraint and robustness.

```
function LOCAL-SEARCH-OPTIMIZATION(state)
  inputs:          state: a feasible schedule
  local variables: new, neighbor: a schedule
                   score: soft constraint fulfillment
  returns:         new state which has more
                   fulfillment on soft constraint
  score ← SC-SCORE(state)
  new ← state
  for i ← 1 to neighborhood size do
      neighbor ← MUTATE(state)
      if SC-SCORE (neighbor) > score and
         HC-PENALTY(neighbor) = 0 do
         new ← neighbor
         score ← SC- SCORE (neighbor)
      end if
  end for
```

Fig. 2. Scattered local search around the feasible schedule algortihm

Comparing with the previous work in [12], the algorithm in the present work can do the real genetic mechanism, namely cross-over. Because of the two tuple model from Eq. 1, the chromosome will form just like a common integer chromosome, as shown in Fig. 3. All events are always taken in the process, satisfy the complete schedule of hard constraint, hence this will not be destroyed by cross-over mechanism as it was in the three dimensional matrix model we used in [12,13]. The present evolutionary algorithm will be called hybrid genetic algorithm (HGA).

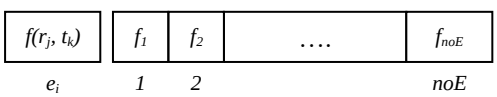

Fig. 3. Course schedule chromosome

where

$$f_i \in \Im^{\text{number of space-time resources}} \quad (3)$$

Tournament of 5 is used for selection of parent chromosome for crossover mechanism. Before parent selection, elitism mechanism is done by using truncated selection scheme. The selection is performed based on violation of hard constraint and fulfillment of soft constraint, combined using weight factor as given in (2).

Elite chromosomes that reach zero hard constraint violation, i.e. feasible schedule, will be further proceeded in the second phase performing local search optimization related to the soft constraints by keeping the zero violation of hard constraint as given in Fig. 2.

III. Results and Discussion

Computation complexity of one iteration for these two different algorithms, i.e. multiple scattered local search (MSLS) and hybrid genetic algorithm (HGA) will be discussed first. The complexity of evolutionary algorithm is more than two times of a single scattered hill-climbing algorithm when the population size of evolutionary algorithm and neighborhood size of hill-climbing are the same [13]. In the present work, the complexity of one iteration for MSLS and HGA is in the same order if the same population– and neighborhood size are used in both algorithms. Hence, the number of iteration is the only extent of computation complexity when we compare MSLS and HGA with the same size of population and neighbors.

The probabilistic nature of both algorithms makes the behavior of algorithm difficult to analyze. Reference [15, 16] introduced an empirical approach to analyze the behavior of non-deterministic algorithm by constructing run time distribution (RTD) and run length distribution (RLD). In practice, this empirical RTD and RLD are determined by running the respective algorithm for a number times on a given problem instance up to some cut-off time or iteration and then for each successful run, recording the required time or the required number of iteration to find a solution, respectively for RTD or RLD. For this study, we use two small sets of curriculum for experiment as given in Table I.

TABLE I. TWO SMALL SETS OF CURRICULUM DATA

Specification	I	II
Number of events	25	51
Number of instructors	14	23
Number of class rooms	2	4
Number of student groups	4	7
Number of events hour	67	138
Available time slots	80	160

For the first test, we run both algorithms to find a feasible schedule on the same curriculum data I by setting a cut-off iteration of 1000. The computational parameter size, i.e. population- and neighborhood size for both algorithms are set to be equal. The resulting RTD and RLD are given in Fig. 4 and Fig. 5. That RTD has similar shape as RLD proves the time consumed for one iteration by both algorithms with the same computational size parameters are equal. Furthermore, the test results have shown that MSLS yields higher probabilities of success for all range of iterations. This outcome is very contradictive with the previous comparing study between traditional GA and the single SHC wherein the first algorithm is more superior in the probability behavior [12,13].

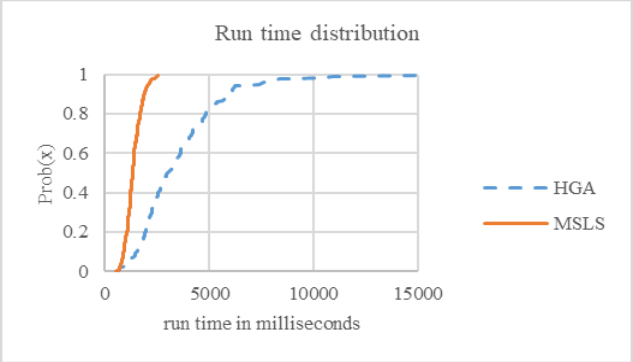

Fig. 4. RTD for MSLS and HGA with the same computational size parameter, i.e. population size=40 and neighborhood size =50

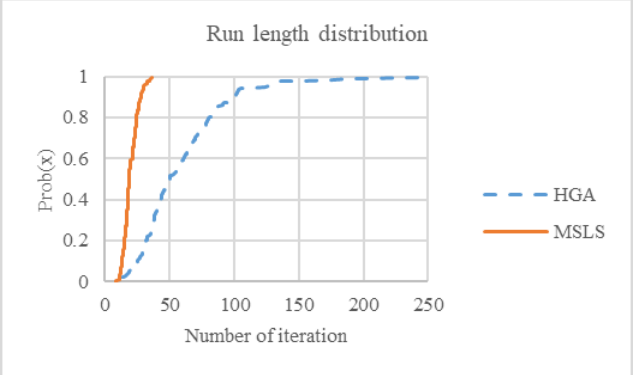

Fig. 5. RLD for MSLS and HGA with the same computational size parameter, i.e. population size=40 and neighborhood size =50

Both computational size parameters give the extent of randomization on the algorithms but for different objectives. While the number of schedule population aims to add more exploration power in a wider search area, the neighborhood size of scattered schedules yields more exploitation random capability in depth.

Effect of these two computational parameter sizes are studied. Higher neighborhood size yields better probability in term of iterations as shown in RLD from Fig. 6. However, in term of run time, this parameter has no effect since this neighborhood size is inversely proportional to run time as depicted in RTD from Fig.7.

Effect of the population sizes are presented in Fig. 8 and Fig. 9. Note that when only one schedule is considered in the population, namely S_c=1, MSLS becomes a scattered local search as in the previous published works [6,13]. The probability of success for the scattered local search here is not more than 50%. Probability in the iteration distribution is getting better when the population size is larger, though in run time distribution probability curve shifts right to slower region of computing run time. In spite of this, it is proven that a probability of 99% success for MSLS needs a sufficient population size.

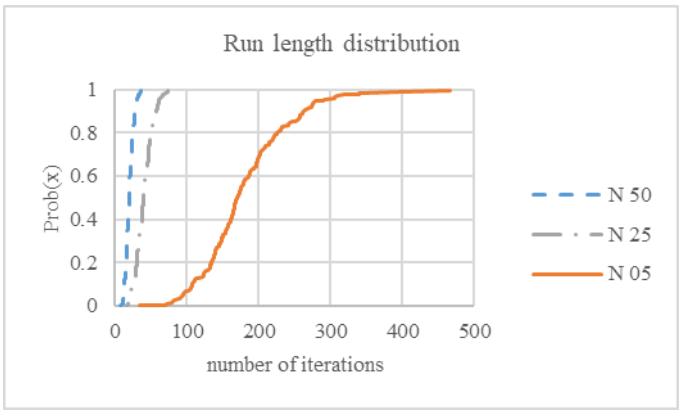

Fig. 6. RLD for MSLS with the same population size=40 and variation of neighborhood size (~~n~~N= 5, 25, 50)

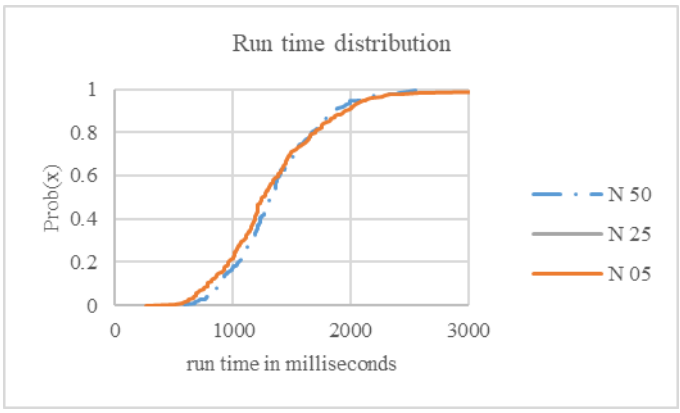

Fig. 7. RTD for MSLS with the same population size=40 and variation of neighborhood size (~~n~~N= 5, 25, 50)

Results given in Fig. 8 and Fig. 9 are obtained by running MSLS on curriculum data II for 200 times up to cut-off 5000 iterations. The problem of curriculum data II is two times bigger than data I. Doubling the problem size requires at least five times number of iterations and eight times longer run time for a same level of probability of success.

Beside the size parameters, i.e. population- and neighborhood size, MSLS has also two weight factors to be set. In searching for feasible schedules, penalty function that includes small contribution of soft constraint satisfaction, i.e. 0.1 SC + 0.9 HC yields better probability curve than 100% HC in the penalty function as shown in Fig. 10. Nevertheless,

excessive contribution of soft constraint satisfaction will shift RTD curve slightly to slower region of computation time.

RTD shown in Fig. 10 is obtained from 200 test runs on curriculum data I up to cut-off 100 iterations or four feasible schedules were found. It is interesting to note here that MSLS is capable to seek a number of valid schedules up to about 20% of population size without too much extra iterations or computation time. RLD from this test is not presented because it has similar shape with RTD since run time required for one iteration is equal in this particular case for every probability curve in Fig. 10.

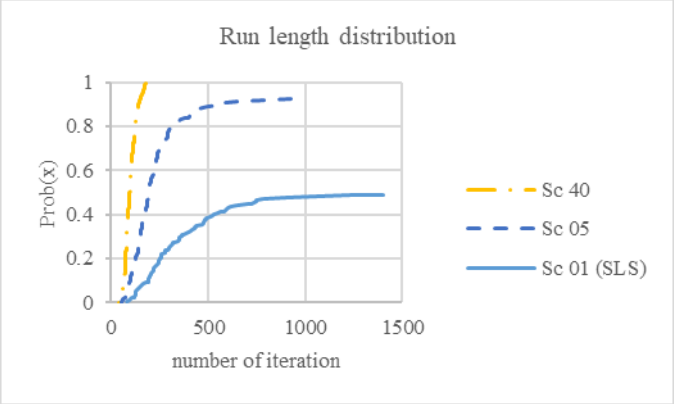

Fig. 8. RLD for MSLS with the same neighborhood size =50 and variation of population size (Sc= 1, 5, 40)

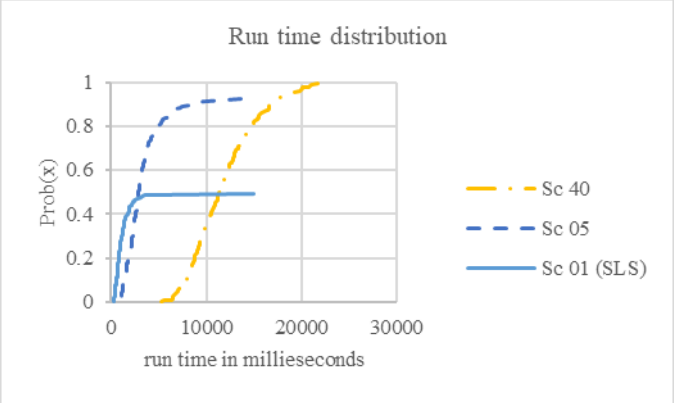

Fig. 9. RTD for MSLS with the same neighborhood size =50 and variation of population size (Sc= 1, 5, 40)

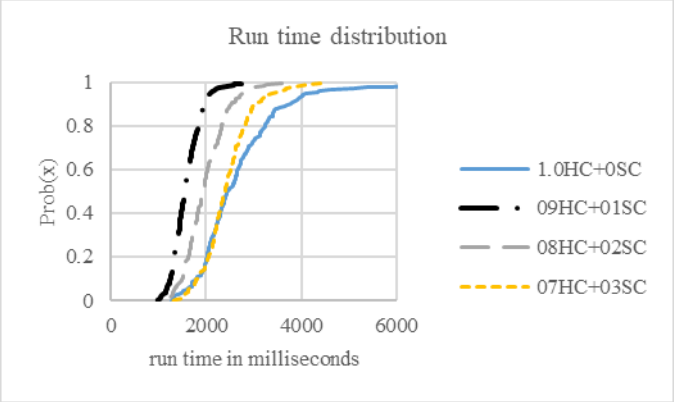

Fig. 10. RTD for MSLS with the same neighborhood size =50 and population size=40, varying weight factor in penalty function in (2)

For the last test, we evaluate effect of neighborhood size on the probabilistic of success for HGA. The test results are given again in Fig. 11 and Fig. 12. If neighborhood size n=0, HGA becomes a traditional GA, and the test result shows that this GA is the worst among HGA with respect to RTD and RLD, even for HGA with very small neighborhood n=2. Similar to the case for MSLS, effect of neighborhood size only significant on RLD, not on RTD with the same reason.

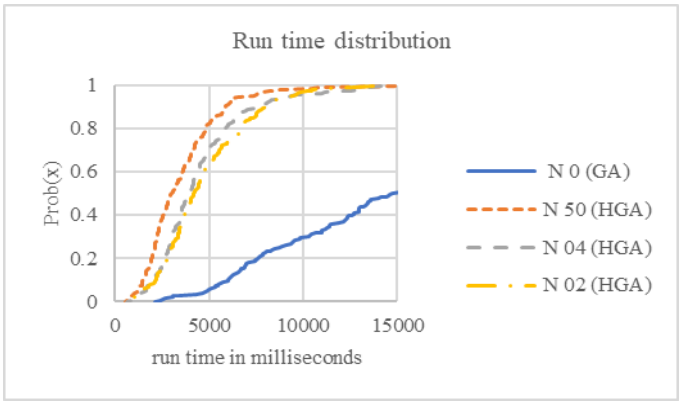

Fig. 11. RLD for HGA with the same population size=40 and variation of neighborhood size (n= 0, 2, 4, 50)

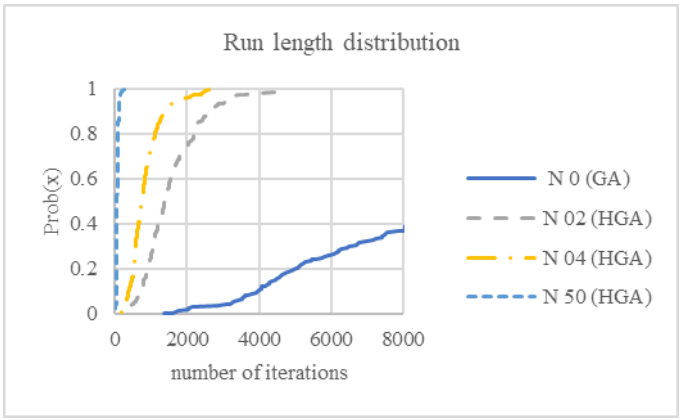

Fig. 12. RTD for HGA with the same population size=40 and variation of neighborhood size (n= 0, 2, 4, 50)

IV. Conclusions

We have presented a multiple scattered local search algorithm, which is a population based version of randomized hill-climbing algorithm. Compared to the original hill-climbing search, the algorithm incorporates stochastic behavior in two different parts, namely during the improvement step in the neighborhood region and in the constructing of initial population. states. The probabilistic behavior of the algorithm was evaluated using empirical approaches by constructing run time distribution and run length distribution. The test results have shown that the probabilistic of success run for the

presented algorithm is better than the original scattered local search and also much better than the hybrid genetic algorithm for the same problem instances.

Another advantage of the multiple scattered local search which is still to be exploited, is a strong parallelism inherent in the algorithm. Taking advantage of this parallelism by implementing the algorithm in parallel processing is practically necessary to solve the real world course scheduling problem whose problem size much bigger than in the presented work.

REFERENCES

[1] M.A. Al-Betar, A.T. Khader and T.A.Gani, "A harmony search algorithm for university course timetabling," in Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, E. Burke, M. Gendreau (eds.). The Montréal, Canada, 2008.

[2] E.K. Burke, D.G. Elliman and R.F. Weare, "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, pp. 35-40

[3] D. Abramson, "Constructing school timetables using simulated annealing: parallel and sequential solutions", Management Science, Vol. 37, No. 1, ,January, 1991, 98-113

[4] A. Elloumi, H. Kamoun and J. Ferland, "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, E.K. Burke and M. Gendreau (eds), August 2008

[5] M.A.S. Elmohamed, G. Fox, and P. Coddington, "A comparison of annealing techniques for academic course scheduling", DHPC-045, SCSS-777, 1998

[6] A. Jamal, "Solving university course scheduling problem using improved hill climbing approach," in Proceeding of the International Joint Seminar in Engineering, August 2008, Jakarta, Indonesia

[7] R. Lewis and B. Paechter, "Application of the grouping genetic algorithm to university course timetabling," in Evolutionary Computation in Combinatorial Optimization, G. Raidl and J. Gottlieb (eds), Berlin Germany, Springer LNCS 3448, 2005, 144-153

[8] D. Moody, G. Kendall and A. Bar-Noy, "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), Universete de Montreal, August 2008

[9] S. Massoodian, A. Esteki, "A hybrid genetic algorithm for curriculumc based course timetabling," in Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT '08, Edmund K Burke and Michel Gendreau (eds), Universete de Montreal, August 2008

[10] P. Myszkowski and M. Norbeciak, "Evolutionary algorithms for timetable problems," Annales UMCS Informatica AI, 2003, 115-125. DOI= http://www.annales.umc.lublin.pl/

[11] A. Jamal, "A three stages approach of evolutionary algorithm and local-search for solving hard- and soft constrained course scheduling problem," in Proceeding of the 11th Seminar on Intelligent Technology and Its Application, Surabaya, Indonesia, 2010, paper no.118, p.324-328

[12] A. Jamal, "University course scheduling using the evolutionary algorithm," in Proceeding of International Conference on Soft Computing, Intelligent System, and Information System, Bali, Indonesia, 2010, republished in Jurnal Al-Azhar Indonesia Indonesia, Seri Sains dan Teknologi, Vol. I, No. 1 Maret 2011

[13] A. Jamal, "Evaluation of modified scattered hill-climbing method and evolutionary approach for course scheduling problem," in Prooceedings of Seminar Nasional Komputasi (SNAKOM 2012), Bandung

[14] C. Akkan and A. Gulcu, "A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem," in Proceedings of the 11th International Confenference on Practice and Theory of Automated Timetabling (PATAT-2016) – Udine, Italy, August 23–26, 2016, p. 451-456

[15] H. H. Hoos and T. Stutzle, "Stohastic local search: Foundations and applications", Morgan Kaufmann Publishers is an imprint of Elsevier, 2005

[16] H. H. Hoos, "Stochastic local search: Methods, models, applications", PhD Dissertation Technisen Universitat Darmstadt, Germany, 1998