We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS

BOOK CITATION INDEX

INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Chapter

Global Optimization Using Local Search Approach for Course Scheduling Problem

Ade Jamal

Abstract

Course scheduling problem is a combinatorial optimization problem which is defined over a finite discrete problem whose candidate solution structure is expressed as a finite sequence of course events scheduled in available time and space resources. This problem is considered as non-deterministic polynomial complete problem which is hard to solve. Many solution methods have been studied in the past for solving the course scheduling problem, namely from the most traditional approach such as graph coloring technique; the local search family such as hill-climbing search, taboo search, and simulated annealing technique; and various population-based metaheuristic methods such as evolutionary algorithm, genetic algorithm, and swarm optimization. This article will discuss these various probabilistic optimization methods in order to gain the global optimal solution. Furthermore, inclusion of a local search in the population-based algorithm to improve the global solution will be explained rigorously.

Keywords: course scheduling, optimization, local search, genetic algorithm, particle swarm optimization, combinatorial optimization problem, probabilistic optimization algorithm

1. Introduction

Scheduling is the process of assigning a set of given tasks to resources by some means. Among the resources, time resource usually plays a central role in scheduling process; hence this process is often called timetabling. Besides the time resource, there are other resources involved in the scheduling process such as space or room, machine or tools, and human resources. The resources are usually subject to constraints that make scheduling problems interesting for researchers in finding an optimal solution or in developing a method for solving it. Course scheduling problem attracts researchers from the field of operation research and artificial intelligence [1–9].

This manuscript will focus on the problem of university course scheduling which has several variants such as school timetabling [10] and examination scheduling [11–13]. The variation of course scheduling problem is merely due to different constraints on the resources involved in the scheduling processes. Despite of these variations, they can be considered as the same family of course scheduling problem. In the scheduling problem, courses or exams have to be assigned into time

and space resources by considering some constraints. University course scheduling problem can be divided into two categories, post-enrolment scheduling [1–4] and prior-enrolment scheduling [5–9]. In the prior-enrolment-based course scheduling, students are not taken into account as an individual person but as a group of study curriculum and student grade; hence it is also named a curriculum-based scheduling [5, 6]. In the post-enrolment-based course scheduling, students and faculty members or lecturers are considered as individual person and not as specified parameter on courses.

University course scheduling problem is simple to understand, yet complex enough to admit solution at varying level of difficulty in the implementation. Several studies of university course scheduling are conducted using operation research, human computer/machine interface, and artificial intelligence. The main issues in the solving method of university course scheduling problem are quality of the schedule solution, namely, the optimal solution, and time spent to produce the schedule solution, i.e., algorithm efficiency.

The most traditional technique for solving the course scheduling problem is the graph coloring technique [14, 15]. Graph coloring algorithm comes from a classical problem in graph theory which implies the problem to color the nodes of an undirected graph such that no two adjacent nodes share the same color. The course scheduling problem is modeled by letting the nodes and edges represent the courses and the common students, respectively. Dandashi and Al-Mouhamed [15] have given a good historical review on the graph coloring technique since 1967 up to recently.

The second group of solution method is the family of local search methods. This is a heuristic-based search method. It finds the best solution among a number of candidate solutions by applying local change from the last found solution. This is a very fast algorithm, but it has a downside that it can easily be stuck at a local optimum. This local optimum issue can be cured by many local search variants such as the taboo search [13], simulated annealing search [2, 10, 12], and improved hill-climbing search [1, 4, 9, 11, 16].

The third group of solution method is the population-based optimization methods that gain more attention by researchers in seeking new methods that are inspired by the nature phenomena such as genetic algorithm [7, 8, 12], evolutionary algorithm [3, 17, 18], ant colony algorithm [19], bee colony algorithm [20, 21], firefly algorithm [22], and particle swam optimization [23, 24].

Evolutionary algorithm was originally not a population-based approach as introduced by Rechenberg in 1965 where only one species is mutated and only one species, i.e., the fittest one, survived in every evolution generation. Mutation is the only reproduction mechanism necessary in the evolutionary algorithm. Crossover mechanism is another reproduction mechanism inspired by biological evolution theory. Crossover mechanism is a simplification model of genetically offspring from mating process of a parent pair. The work of John Holland in the early 1970s included both genetic operators, and since then a so-called genetic algorithm became popular that belongs to the evolutionary strategy family. In genetic algorithm, each individual in a population forms a candidate solution. The candidate solution is evolved by mutation and crossover mechanism in every generation. Through fitness selection scheme, they move toward a better generation.

After successful mimicking of the nature phenomena from the evolution theory, once again, Mother Nature has inspired researchers to develop new optimization algorithm based on swarm intelligent theory. Swarm behavior or swarming is a collective behavior exhibited by particularly animals which aggregate together in finding food and moving or migrating in some direction. Ant colony optimization algorithm is one of the first metaheuristic optimizations in this group of

optimization method, initially proposed by Marco Dorigo in 1992. Initially ants wander randomly, and upon finding food they return to their colony while leaving trail called pheromone trails. When other ants find such trail, they are likely to follow the trail and return and reinforce the trails if they eventually find food. The pheromone trail is the main issue of swarm intelligent communication in ant colony, on which the algorithm was developed.

Another algorithm that imitates the intelligent foraging behavior of animal is artificial bee colony optimization algorithm proposed by Karaboga in 2005. There are three groups of bees in bee colony, i.e., employed bee, scouts, and onlookers. Employed bees go to their food source and back to hive and dance. Onlookers watch the dances of the employed bees, and depending on employed bee's waggle dances, food sources are chosen or abandoned. The employed bees whose food sources have been abandoned become a scout and start to seek for a new food source.

The most recent bio-inspired algorithm, as far as the author's knowledge, is the firefly algorithm developed by Xin-She Yang in 2008. It is a heuristic algorithm which is a population-based stochastic method which is derived and motivated by the flashing or mating behavior of fireflies. The position of all fireflies represents a possible set of solutions, and their light intensities represent corresponding fitness values or quality of all solutions.

Particle swarm optimization is a population-based evolutionary computation technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. This algorithm is the simplest model of swarm behavior algorithm. This algorithm shares similarity with genetic algorithm, but it differs mainly due to the absence of genetic operator. A kind of communication between particles in the swarm controls the movement of each particle in searching food. When an animal spots a location that is rich of food, it memorizes the location until better location is found. The movement of each particle is calibrated to its best location so far and the best location from the whole animals in the swarm. The algorithm is also much simpler because it has only few parameters to adjust compared to genetic algorithm. Its simplicity and its generic computation model for broad application make this algorithm more attractive to assess than other new algorithms inspired by animal behavior as reviewed by Poli et al. [23].

The following sections will be structured as follows: first we will discuss the probabilistic optimization methods from three different previously described approaches. Thereafter we will rigorously explain the university course scheduling problem and how we model it appropriately for all the employed solution methods. Discussion of experimental result will be presented in the last section before some concluding remarks are briefly inscribed.

2. Probabilistic optimization method

From the beforehand described optimization approaches for discrete problem such as university course scheduling problem, we can summarize that there are three groups of solution approach, namely, coloring graph, local search approach, and the population-based approach. The population-based approach can be further classified into the population-based evolutionary approach and the population-based social behavior approach.

While the probabilistic characteristic is inherent in the population-based approaches, the local search approach is a single-based solution technique and normally not a probabilistic solution method. In this article, we will bring probabilistic nature into the local search approach by introducing scattered neighborhood [9] and multiple random start local search method [25]. We will discuss thoroughly this

multiple-scattered local search and compare these two population-based methods, namely, genetic algorithm and particle swarm optimization, respectively, from evolutionary approach family and social behavior approach family.

2.1 Multiple-scattered local search (MSLS)

Local search approach has little probabilistic nature. In its original form, namely, the steepest-ascent hill-climbing method, wherein from its current position systematically for every possible single mutation is evaluated to find the highest fitness increase in the neighborhood, this method has no probabilistic aspect at all. The fast local search, called the next ascent hill-climbing [26], wherein a single mutation is evaluated systematically from current position until any increase in the fitness is found, has a little probabilistic nature. Random mutation hill-climbing [26] or scattered local search [9] is a probabilistic variation of local search wherein from its current position, a random mutation mechanism takes place until an increase in fitness is found. Random mutation hill-climbing takes randomly a single mutation, while scattered hill-climbing takes a certain number of mutation randomly, and the highest increase in fitness is chosen. Other local searches, such as taboo search and simulated annealing search, are also improvements and derivatives of the hill-climbing search method. However, they can hardly be classified as hill-climbing search technique since the changing makes them differ too much from the original method.

In general, the hill-climbing procedures are as follows:

1. Evaluate the fitness, i.e., the heuristic objective function, for the current position.

2. Find the candidate for the next position in the neighborhood area by small changing from the current position and then evaluate the fitness for the candidate position.

3. If there was an increase in fitness, then set the candidate for the next position as the current position.

4. Repeat from step 1 until the maximum iteration was reached or other stop criteria are fulfilled.

The difference between various hill-climbing search described in previous paragraph is found only in step 2 of the above procedure. Because of systematically searching for the candidate for the next position, the steepest ascent hill-climbing and the next ascent hill-climbing methods are classified as deterministic search techniques. The random mutation hill-climbing and the scattered hill-climbing are probabilistic search techniques due to random mutations in searching for the next position.

Random restart hill-climbing is another approach to embed the probabilistic nature. In case the hill-climbing search stopped due to its maximum iteration, the whole hill-climbing search above restarted with a completely new initial position which set randomly. The restart algorithm will continue until the optimization criteria are reached. The multiple-scattered local search is actually a set number of scattered hill-climbing searches running with randomly set initial positions.

Let np be the number of initial position and n the number of scattered candidate position in the neighborhood of current position, and then the algorithm of the multiple-scattered local search is as follows:

1. Set up np initial positions randomly, and evaluate their heuristic fitness or objective functions, and save in array of np current positions.

2. Repeat the following steps until maximum iteration or another stop criterion is reached:

   a. For each position in a set of np positions, do:

      i. Find n new positions in the neighbor of current position.

      ii. Evaluate the fitness of each new position, and save the best position whose fitness has the highest value.

      iii. If the best position gives an increase in fitness compared to the current position, set the best position as current position.

2.2 Genetic algorithm

Genetic algorithm is a population-based optimization technique that simplifies survival of the fittest principle from the evolution theory. Any individual chromosome represents a solution state of the problem. Every generation has a population that consists of a fixed number of individual instances. The population of the next generation will inherit their ancestor by crossover of two mating chromosomes, by mutation of single individual chromosome, and by elitism. All of these three evolution mechanisms hold the true survival of the fittest principle.

Elitism in the evolution mechanism means superior chromosomes survive to the next generation without any changes. This elite group which is only a small fraction of the population has the highest fitness in the population. The rest of the population will be selected in pairs through a probabilistic scheme involving their fitness values. Each pair acts as parent that will have two descendants produced by crossover between two parent's chromosomes. Since the population size is constant, every pair of children replaces their parents in every generation. Finally, mutations take place in all children by chance of probability distribution.

The following is the pseudocode of genetic algorithm:

1. Set up np individual chromosomes for the first generation.

2. Repeat the following steps limited by maximum generation number:

   a. Evaluate the fitness values of every chromosome in the population of current generation.

   b. Sort the chromosomes by the fitness values.

   c. Select a small number of ne elite chromosomes, i.e., the highest ranked chromosomes.

   d. Select a pair of parents via a selection mechanism from the rest of the chromosomes.

   e. Based on a crossover probability, crossover scheme takes place between each selected pair, and the two successors will replace the parents.

f. Based on a mutation probability, every single chromosome outside the elite group undergoes a mutation.

Elitism mechanism has a purpose to make sure that the individuals with the highest fitness do not vanish by chance of probabilistic. Pairing the parents aims to find better descendants which differs from both parents but still inherits their characteristics. Hence, crossover mechanism has a function of an exploration for new chromosomes. Single chromosome is slightly changing by mutation which aims to improve the individual in exploitation scheme. Exploration power of crossover combined with exploitation power of mutation and an additional power, a conservatism of elitism, makes genetic algorithm very popular for a long time in the area of artificial intelligence and organization research for optimization problem [17].

2.3 Particle swarm optimization

Foraging behavior of some animals is in a group of numerous individuals in swarm formation. In the swarm behavior, animals such as birds, insects, or fishes move in such mechanism that they do not collide with each other, but they can cover broaden area for finding the foods. This behavior inspired an optimization technique called particle swarm optimization which is rather simple than the previously nature-inspired genetic algorithm. Every position of animals in the swarm represents a solution state of the problem. Every animal or particle moves in directions that are influenced by the best individual position so far and the best position of all individuals in the swarm. The best position means, in the real natural condition, the richest food available, and in the optimization problem means the highest fitness solution state.

Let np particles be in the swarm and each particle has its own initial positions Xo and velocity Vo; then, the pseudocode of the particle swam optimization is as follows:

1. Evaluate the fitness value of each initial position Xo, and save the individual best position as $XPbest$ and the global best position as $XGbest$.

2. Repeat the following steps until maximum iteration or another stop criterion is reached.

   a. Update the velocity of each particle for the next move as follows:

   $$Vi + 1 = Vi + r_1 * (XPbest - Xi) + r_2 * (XGBest - Xi)$$

   b. Update the new position of each particle as follows:

   $$Xi + 1 = Xi + Vi + 1$$

   c. Evaluate the fitness value of the new position of each particle, and update the individual best position and the global best position if necessary.

According to James Kennedy who proposes particle swarm optimization (1995), the adjustment particle movement toward the individual and global best solution is conceptually similar to the crossover mechanism in genetic algorithm. This movement ensures exploration power of the algorithm. The two types of best positions are in some sense acting as elitism in genetic algorithm which holds the conservatism of the particle swarm optimization algorithm.

2.4 Exploitation power of local search in the global optimization

Genetic algorithm and particle swarm optimization have great power of exploration. The exploration power could almost guarantee finding a global optimum if the number of iteration is large enough or at least finding a sufficient global optimum for a reasonable number of iteration. Lack of exploitation power in particle swarm optimization and slight exploitation power in genetic algorithm make these two algorithms slow in finding of good enough solutions. We will introduce the exploitation power into genetic algorithm and particle swarm optimization with the aid of the local search on the elite groups [17, 18].

2.4.1 Hybrid genetic algorithm (HGA)

In the original genetic algorithm, the elites are not mutated, but could be replaced by new elites in the next generation. Empowering the algorithm using single iteration of scattered local search on the elites, the hybrid genetic algorithm becomes as follows:

1. Set up np individual chromosomes for the first generation.

2. Repeat the following steps limited by maximum generation number:

    a. Evaluate the fitness values of every chromosome in the population of current generation.

    b. Sort the chromosomes by the fitness values.

    c. Select a small number of ne elite chromosomes, i.e., the highest ranked chromosomes.

    d. For each chromosome in ne elite, do:

        i. Randomly develop n new mutated chromosomes.

        ii. Evaluate the fitness of each new mutated chromosome, and save the best chromosome whose fitness has the highest value.

        iii. If the best chromosome gives an increase in fitness compared to the current elite chromosome, set the best chromosome as the new elite chromosome.

    e. Select pair of parents via a selection mechanism from the rest of the chromosomes.

    f. Based on a crossover probability, crossover scheme takes place between each selected pair, and the two successors will replace the parents.

    g. Based on a mutation probability, every single chromosome outside the elite group undergoes a mutation.

2.4.2 Hybrid particle swarm optimization (HPSO)

We will bring the exploitation power in the particle swarm optimization by performing scattered local search on the individual best positions and the global

best position of particles in the swarm. Hence, the hybrid particle swarm optimization algorithm becomes as follows:

1. Evaluate the fitness value of each initial position ***Xo***, and save the individual best position as ***XPbest*** and the global best position as ***XGbest***.

2. Repeat the following steps until maximum iteration or another stop criterion is reached.

   a. Update the velocity of each particle for the next move as follows:

   $$Vi + 1 = Vi + r_1 * (XPbest - Xi) + r_2 * (XGBest - Xi)$$

   b. Update the new position of each particle as follows:

   $$Xi + 1 = Xi + Vi + 1$$

   c. Evaluate the fitness value of the new position of each particle, and update the individual best position and the global best position if necessary.

   d. For each individual best position ***XPbest*** and the global best position ***XGbest***, do:

      i. Find *n* new positions in the neighbor of the best position.

      ii. Evaluate the fitness of each new position, and save the new best position whose fitness has the highest value.

      iii. The new best position will replace the best position under consideration, if it gives increase in fitness.

3. University course scheduling problem and model

3.1 Problem definition

University course scheduling is a process of assigning a set of courses to limited time resources and space resources, namely, classrooms. We consider only curriculum-based or prior-enrollment course scheduling. In this case, each course has been set who will teach and which student group will attend the course. The assigning process must satisfy some hard constraint, for instance, avoiding lecturer conflict. In addition, some soft constraint such as considering preference time of lecturer will be desired to be fulfilled in the schedule.

Formally, the curriculum-based course scheduling problem will be described as follows:

- There is a set of courses $[C_1, C_2, ... C_{nc}]$ where each course is attended by a number of students from specified studies and taught by specified lecturers. A course could be given in more than 1 course hour and probably need computer equipment in the classroom.

- There is a set of lecturers $[L_1, L_2, ... L_{nl}]$ who have been assigned to teach some courses and probably have some unavailable time slots and some preference time slots to teach.

- There is a set of student group [S_1, S_2, ... S_{ns}] which consist of a number of student from the same department and the same grade.

- There is a set of room [R_1, R_2, ... R_{nr}] which has a number of seat capacity and some room equipped with computers.

- There is a set of time slot [T_1, T_2, ... T_{nt}] which can be expressed in daytime and clock time.

A feasible course schedule has to satisfy the following eight hard constraints:

- Complete schedule which means all courses have been assigned into available time and rooms

- Avoiding room conflict which means no room is used by more than one course event at the same time slot

- Avoiding lecturer conflict which means no lecturer teaches more than one course event at the same time slot

- Avoiding student conflict which means no student group from the same department and same grade must attend more than one course event at the same time

- Avoiding excessive attending student number in classroom

- Avoiding lecturers' unavailable time which means no lecturer teaches in his or her unavailable time slot

- Proper equipped classroom which means room has a feature needed by the assigned course

- Continuous course event which means multiple hour courses must be assigned in the same room contiguously

Beside the hard constraints that must be fulfilled unconditionally, course schedule preferably satisfy some soft constraints such as:

- Avoiding lecturer's preventive time which means lecturers do not teach in his or her preventive time slot.

- Lecturer's preference time which means lecturers teach in their preference time

- Full classroom which means the number of students joining the course is more than the half of classroom capacity

A course schedule is defined as optimum if the number of hard constraint violation is zero and the number of soft constraint fulfillment is as many as possible.

3.2 Course scheduling model

Computational model of a course schedule can be represented as a two-dimensional matrix where rooms and time slot denote as row number and column number [3, 5, 7]. Each matrix cell is filled up with course event. Time slots comprise

of day and hour, for instance, if the number of day is 5 and every day consists of 8 course hours, then the number of time slot columns are 40 time slots. A variant of this model has flexible time slot length [7].

A slightly different model from the two-dimensional model is a three dimensional-model wherein the two components of time slot are put into a two-dimensional matrix, namely, hour in row and day in column. The room dimension is placed in the third direction of matrix [9, 17, 18, 27]. As in the two-dimensional matrix model, every matrix cell contains a single hour of course event.

These two matrix-based models have an advantage that the "room conflict" hard constraint is inherently fulfilled. However, it has disadvantage on "complete schedule" hard constraint, namely, ensuring all course events are completely put in the model. Hence the matrix-based model is inappropriate for randomly assigning course events in the matrix. Another problem with the matrix-based model is in crossover evolution mechanism where parent mating could make a course event duplicate and course event missing in their successors [9, 17].

To get around these two problems, we will use a list of 3-tuple which consists of course event, room, and time slot $<C_i, R_j, T_k>$ [6, 25]. This 3-tuple list can be simplified by introducing a space–time function $f(R_j, T_k)$. If we denote all the variable using only their indices in the 3-tuple such that $<C_i, R_j, T_k>$ is written as $<i, j, k>$, then the space–time function can be expressed as

$$f(j, k) = k + (j - 1) * nt \qquad (1)$$

where $j = 1 .. nr$ is room index and $k = 1 .. nt$ is time slot index and maximum value of $f(j,k) = nr*nt$.

Hence the 3-tuple list schedule is shortened as a vector *Sch*:

$$Sch = \{f_1(j, k), f_2(j, k), ..., f_i(j, k), ... f_{nc}(j, k)\} \qquad (2)$$

where index i represents a course which is associated to student groups S_l, lecturers L_m, and course event duration.

The first hard constraint, namely, the complete schedule constraint, is assured by schedule model in Eq. (2). The other seven hard constraints are taken into account in penalty function *HC(Sch)* which is an accumulation of every hard constraint violation in each course C_i. The three soft constraints are taken into account in score function *SC(Sch)* which is an accumulation of every soft constraint fulfillment in every course C_i. Hence the fitness function is formulated as

$$fitness(Sch) = w_{sc} * SC(Sch) - w_{hc} * HC(Sch) \qquad (3)$$

where w_{sc} and w_{hc} are weight factors. The objective of optimization is to maximize this fitness function and make sure a zero hard constraint penalty function *HC(Sch)*.

Using the tuple list scheduling model, the problem size of course scheduling is mainly determined by the number of course *nc*.

3.3 Computational model

In the previous section, we explain how the scheduling model is described including the heuristic fitness function. Here we will describe the computational model of three algorithms, namely, multiple-scattered local search, hybrid genetic algorithm, and hybrid particle swarm optimization. Computational model identifies parameters for the computational size. The objective of defining the

computational model is to ensure a fair comparison among three different types of algorithm, because considering only an equal problem size can yield biased evaluation if the computational model is significantly different.

The global search computational model is governed by population number np. In the case of multiple-scattered local search, this exploration power size determines the number of hill climbers; in the case of hybrid genetic algorithm, it is the amount of chromosomes; and for hybrid particle swarm optimization, it is set by the number of particles in the swarm. The local search algorithm of all three algorithms implements the same scattered local search; hence the size of this exploitation power is determined by the number of probable positions in the neighborhood ne.

The computational model size, i.e., population number np and neighbor number ne, and the problem size, i.e. course number nc, will completely govern the size of three different algorithms. The performance of the success runs for the heuristic approach control by the required number of iterations or generation in the case of genetic algorithm.

4. Experimental results and discussion

4.1 Evaluation model

Because course scheduling is a non-deterministic polynomial complete problem (NP-problem) [5, 28], algorithm evaluation tool for a deterministic polynomial problem (P-problem), such as complexity asymptotic analysis, is not really appropriate to evaluate course scheduling algorithms. An empirical approach was introduced by Hoos [29, 30] to analyze the behavior of non-deterministic polynomial problem for such three algorithms under consideration. Run time distribution (RTD) and run length distribution (RLD) are empirically constructed by running the same algorithm with the same condition for a sufficient number of runs until some stop criteria are reached or up to some cutoff time or maximum iteration. For each run, the required run time and the required number of iteration, for RTD or RLD, respectively, to reach a good solution are recorded.

RTD and RLD will represent cumulative probability distribution function:

$$F(x) = P \mid x \leq X \mid \qquad (4)$$

where X is a random variable which represents required run time in RTD or required run length in RLD.

Specification	Set I	Set II
Number of courses	25	51
Number of rooms	2	4
Number of time slots	80	160
Number of instructors	14	23
Number of student group	4	7
Number of course hours	67	138

Table 1.
Two sets of small course scheduling data [15].

4.2 Experimental data

We will evaluate the three algorithms using two sets of small curriculum-based course scheduling problems as given in **Table 1**.

On each set, evaluation will be performed for 250 runs on each set of scheduling problems until at least one zero $HC(Sch)$ or a specified maximum iteration is reached. Size of computational model of all three algorithms, i.e., population number np and neighborhood number ne, will be varied to grant the performance behavior of these three different algorithms.

4.3 Discussion of results

The effect of two computational size parameters, namely, the population number np and the neighborhood number ne, will be studied. While the population number np gives the exploration force to enhance a global search, the neighborhood number ne provides the exploitation force for the local search.

4.3.1 Exploration for global search

We will firstly investigate influences of parameter np on the success probability for each algorithm separately using the scheduling problem set I. Test results from the hybrid particle swarm optimization are represented in **Figures 1** and **2**, respectively, RLD and RTD. **Figure 1** shows that the higher number of population, the better its run length distribution, namely, the most left distribution function. It is also shown that using cutoff of 1000 iterations, 100% probability of success to find a zero $HC(Sch)$ is obtained for a population number np of at least 40, and for the smaller np, it is slightly less than 100%. However, the variation of population number almost does not affect RTD for the hybrid particle swarm optimization as shown in **Figure 2**.

Figures 3 and **4**, respectively, represent RLD and RTD from the multiple-scattered local search. **Figure 3** shows the same behavior as **Figure 1** for the hybrid particle swarm optimization, namely, increasing number of population shifts RLD function to the left side, the better behavior. Run time distribution using multiple-scattered local search depends significantly on the population number because the
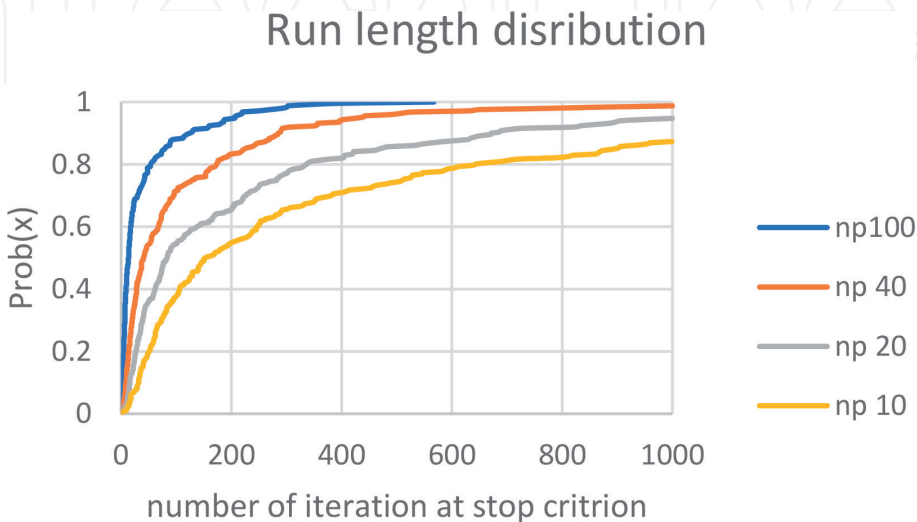

Figure 1.
RLD resulting from hybrid particle swarm optimization for population number np 10, 20, 40, and 100 and neighborhood number ne 20.
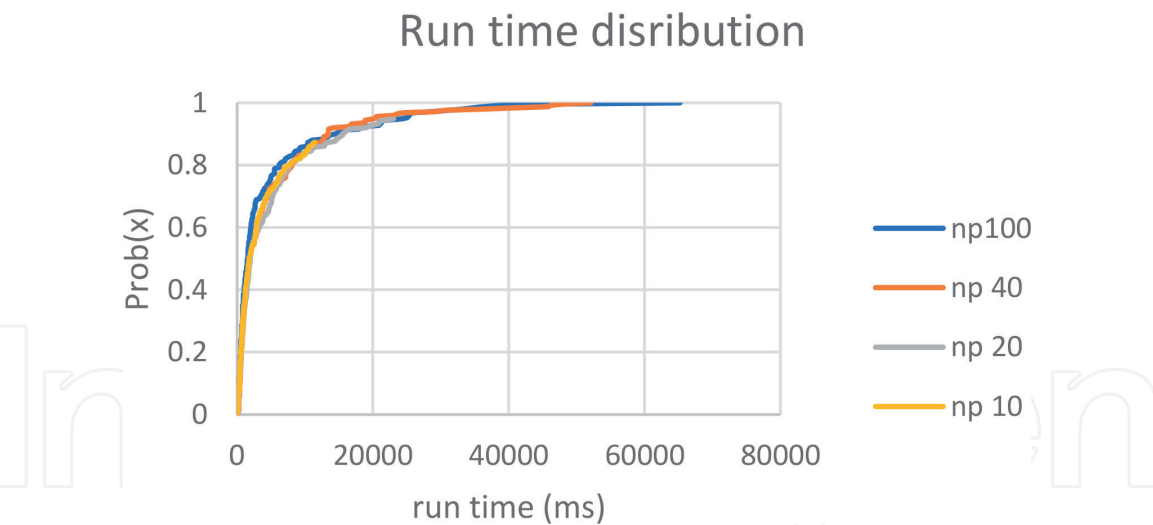
Run time disribution

Figure 2.
RTD resulting from hybrid particle swarm optimization for population number np 10, 20, 40, and 100 and neighborhood number ne 20.
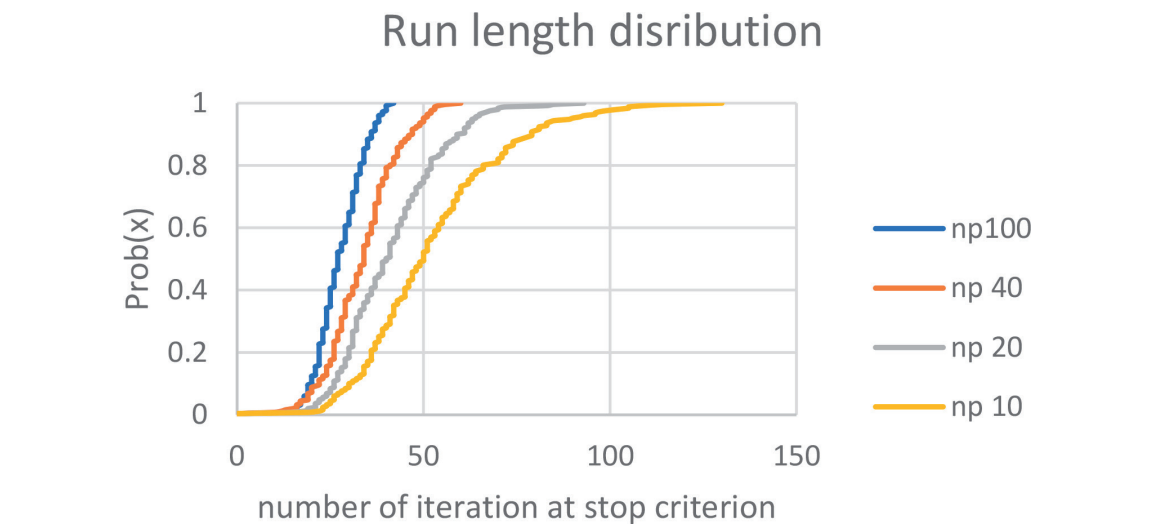
Run length disribution

Figure 3.
RLD resulting from multiple-scattered local search for population number np 10, 20, 40, and 100 and neighborhood number ne 20.
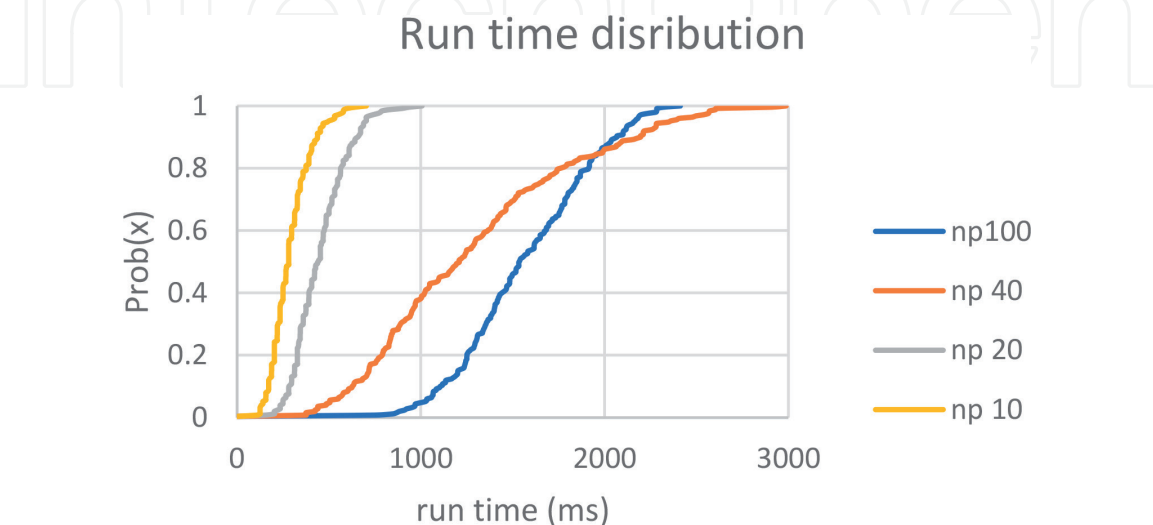
Run time disribution

Figure 4.
RTD resulting from multiple-scattered local search for population number np 10, 20, 40, and 100 and neighborhood number ne 20.

required run time growth is higher than the reduction of required run length as function of population number as shown in **Figure 4**.

Reducing the number of population until *np* = 10, we found that using multiple-scattered local search, finding of a feasible schedule is assured if one lets the algorithm run up to 1000 iterations. We have investigated thoroughly this algorithm for lower population number until it becomes a single-scattered local search as presented in **Figures 5** and **6**. Note that we set maximum iteration of 3000 for this test. We found that a success run probability of 100% at maximum iteration of 3000 needs at least *np* = 4 population members and the minimum population number is getting higher, i.e., *np* = 8, for larger schedule problem, i.e., scheduling problem set II as shown in **Figure 6**.

Figures 7 and **8** show the result of hybrid genetic algorithm. In this case, increasing number of population *np* merely improves the run length distribution as shown in **Figure 7**. **Figure 8** shows that the increasing number only made things worse, as logically higher population number needs more run time for the same level of probability.
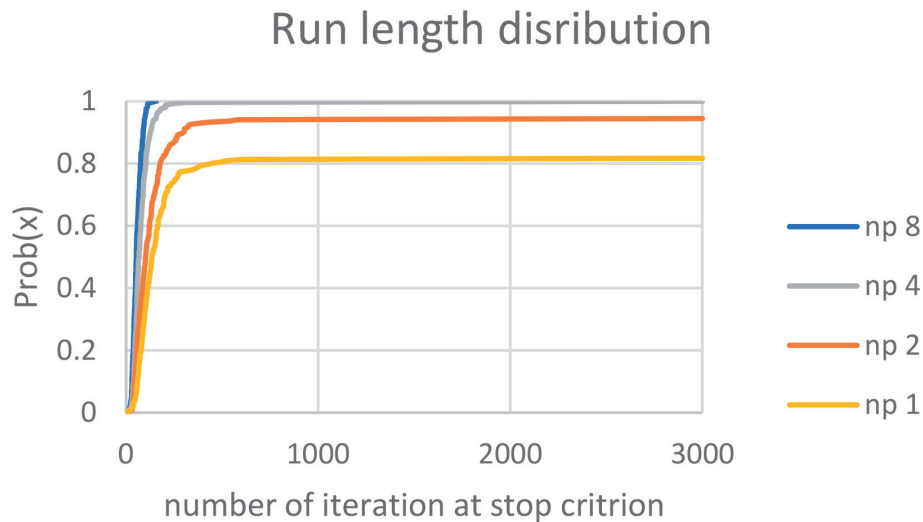

Figure 5.
RLD resulting from multiple-scattered local search for population number np 1, 2, 4, and 8 and scheduling in two rooms.
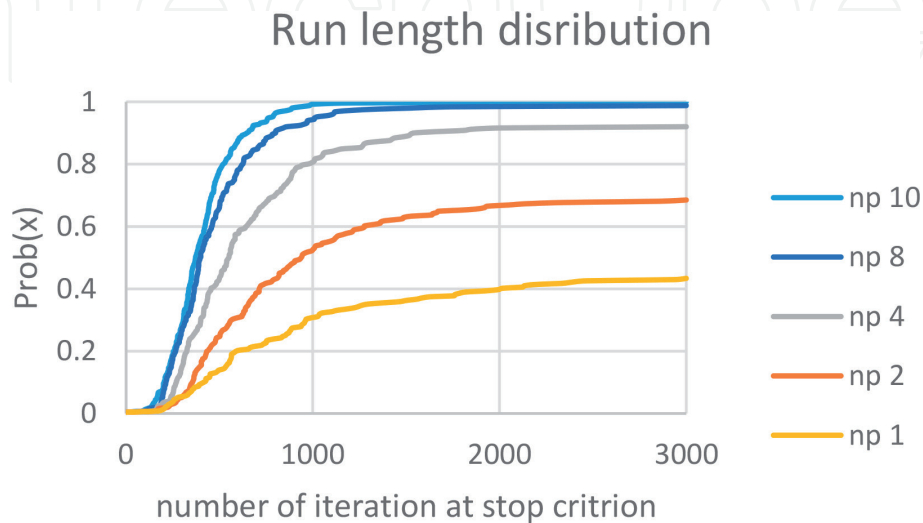

Figure 6.
RLD resulting from multiple-scattered local search for population number np 1, 2, 4, 8, and 10 and scheduling in four rooms.
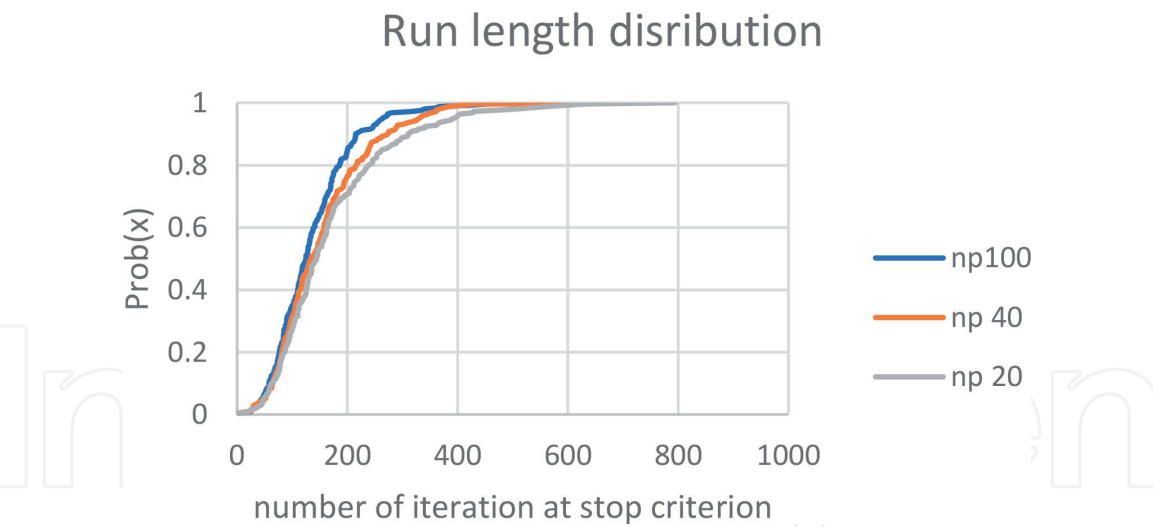
Figure 7.
RLD resulting from hybrid genetic algorithm with population number np 20, 40, 100 and neighborhood number ne 20.
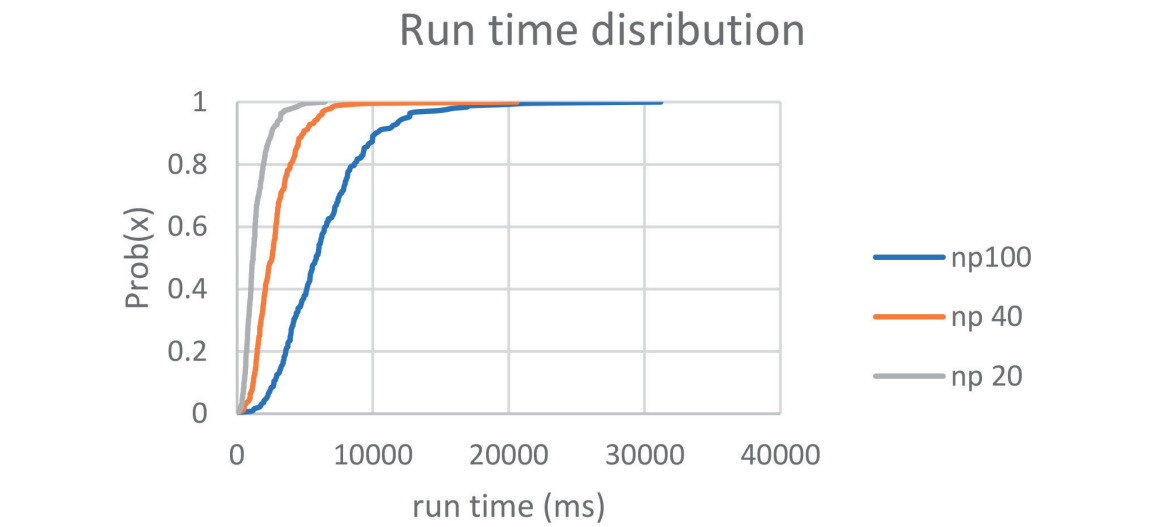

Figure 8.
RTD resulting from hybrid genetic algorithm for population number np 20, 40, and 100 and neighborhood number ne 20.

4.3.2 Exploitation for local search

The novelty of the presented hybrid algorithms is the introducing of exploitation force in the original algorithm where exploration power is essential. This enhancement is affected by the number of scattered candidates in the neighborhood of the elites, denoted as *ne*. If this neighborhood number *ne* is zero, it means the hybrid algorithm is exactly the same as the original versions.

Figure 9 depicts effect of local search in the hybrid genetic algorithm on the run length probability distribution. Even a small neighborhood number *ne* = 2 affects the probability function significantly compared to the original genetic algorithm.

Figure 10 depicts effect of local search in the hybrid particle swarm optimization on the run length probability distribution. Using 6000 iterations as the maximum stop criterion, the original particle swarm optimization fails to yield any feasible solution; hence no result is given in **Figure 10** for the original version. Even a small neighborhood number *ne* = 2 affects the probability function significantly compared to the original genetic algorithm.
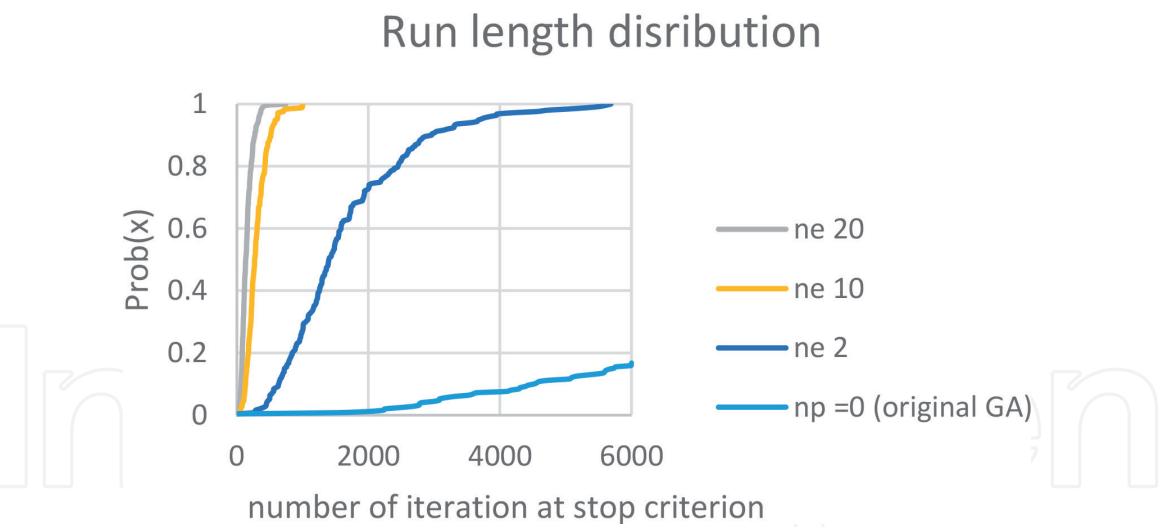
Run length disribution

Figure 9.
RLD resulting from hybrid genetic algorithm with variation of neighborhood number ne 2, 10, and 20 and the original genetic algorithm.
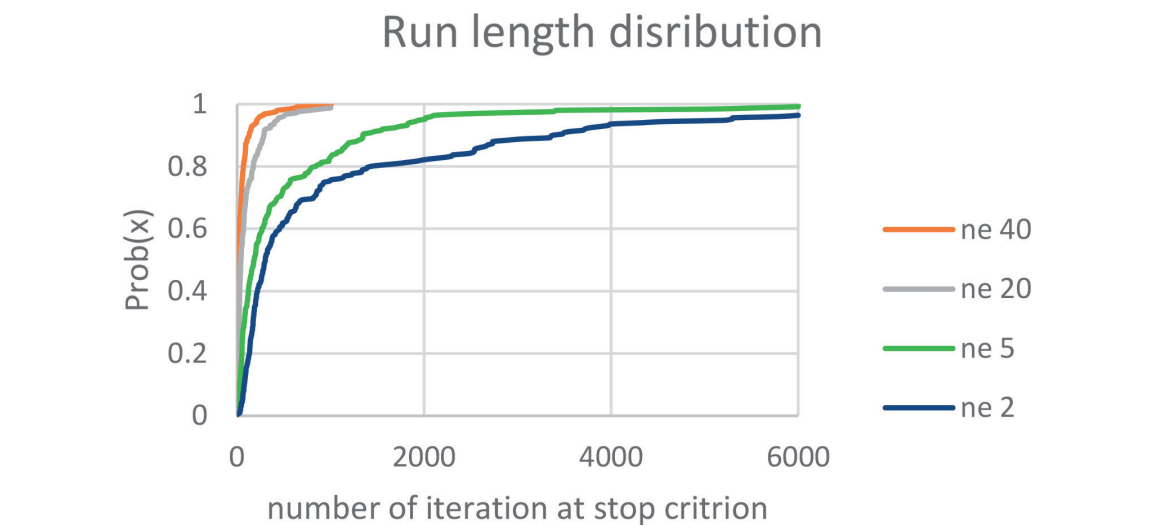
Run length disribution

Figure 10.
RLD resulting from hybrid particle swarm optimization with variation of neighborhood number ne 2, 5, 20, and 40.
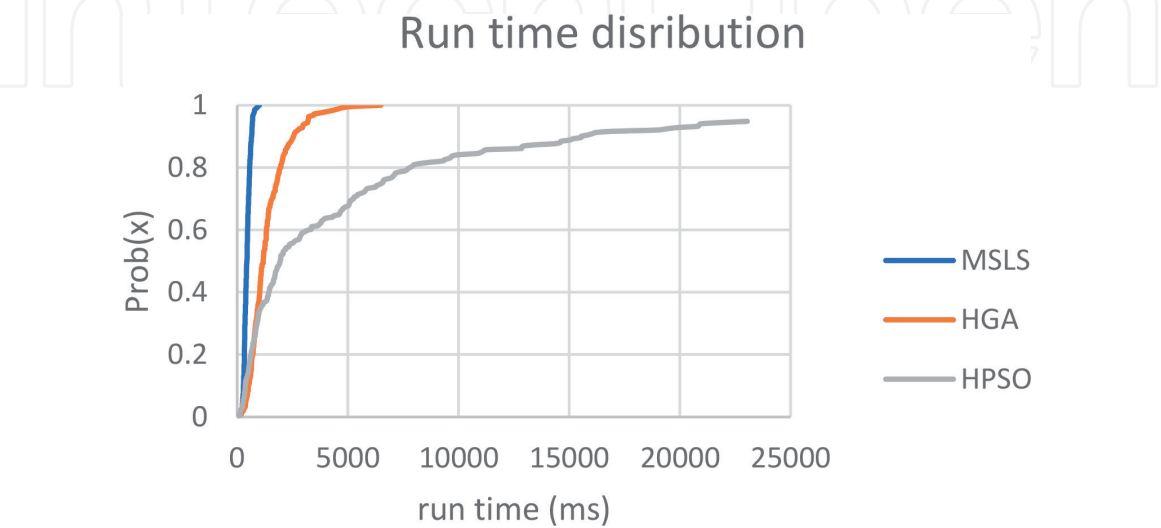
Run time disribution

Figure 11.
RTD resulting from the three algorithms for population number np 20 and neighborhood number ne 20, which are multiple-scattered local search, hybrid genetic algorithm, and hybrid particle swarm optimization.

4.3.3 Comparison between three algorithms

Figure 11 depicts RTD for three algorithms for the same computational size, i.e., $np = 20$ and $ne = 20$. It shows that multiple-scattered local search is the most efficient in time and also in the required iteration number, while hybrid particle swarm optimization is the worst.

5. Conclusions

Course scheduling problem is a discrete optimization problem and considered as NP complete problem which is hard to solve. Therefore, we presented three algorithms which are modification from three popular algorithms. The first algorithm is multiple-scattered local search which is an enhancement of the hill-climbing search. The improvement is achieved by introducing exploration search power in the original single local search.

The hybrid genetic algorithm and the hybrid particle swarm optimization are the last two presented algorithms. The original version of these two algorithms is well known for their powerful of exploration ability in searching of global solution for a sufficiently large number of iterations. The hybrid enhancement of these two algorithms was implemented by implanting scattered local search on the small elite group. The enhancement aimed to accelerate searching process.

Course scheduling problems used for the evaluation of three algorithms are curriculum-based or pre-enrollment course schedules which must fulfill eight hard constraints and three soft constraints. The course schedule problem was modeled using a list of 3-tuple which consists of course event, room, and time slot $<C_i, R_j, T_k>$ which further simplified as a vector containing time–space allocation of every course. The essential advantage of using this course model is to resolve the problem of missing courses and double allocated of the same courses in the evolution mechanism. Furthermore, this course model can be used for all three algorithms under consideration with comparable computation model size, i.e. the population number np the neighborhood number ne.

The experimental test results have proven that the population number consistently governs the exploration power for better global search in term of required iteration numbers at cost of more time needed. In the case of multiple-scattered local search, only a small number of population are needed to obtain a good probability behavior of success run.

The effect of implanting local search in two originally global search algorithms is more remarkable. Letting a very small-scattered local search in the elite group has improved the cumulative probability distribution function of hybrid genetic algorithm and hybrid particle swarm optimization compared to the original versions. However, comparing all three algorithms for the same problem condition yields the multiple-scattered local search as the superior algorithm over the other two hybrid algorithms for cumulative time probability distribution and cumulative run length distribution.

Acknowledgements

Author details

Ade Jamal
University of Al-Azhar Indonesia, Jakarta, Indonesia

*Address all correspondence to: adja@uai.ac.id

IntechOpen

References

[1] Goh SL, Kendall G, Sabar NR. Improved local search approaches to solve the post enrolment course timetabling problem. European Journal of Operational Research. 2017;**261**(1):17-29. DOI: 10.1016/j.ejor.2017.01.040

[2] Elmohamed MAS, Fox G, Coddington P. A comparison of annealing techniques for academic course scheduling. DHPC-045, SCSS-777; 1998

[3] Myszkowski P, Norbeciak M. Evolutionary algorithms for timetable problems. Annales UMCS, Informatica. 2003;**1**(1):115-125. Available from: http://www.annales.umc.lublin.pl

[4] Phillips AE, Walker CG, Ehrgott M, Ryan, DM. Integer programming for minimal perturbation problems in university course timetabling. In: Proceeding of 10th International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014); August 2014; York, United Kingdom; 2014. pp. 26-29

[5] Al-Betar MA, Abdul Khader AT. A harmony search algorithm for university course timetabling. Annals of Operations Research. 2012;**194**(1):3-31. DOI: 10.1007/s10479-010-0769-z

[6] Moody D, Kendall G, Bar-Noy A. Constructing initial neighborhoods to identify critical constraints. In: Burke EK, Gendreau M, editors. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT '08); August 2008; Montréal, Canada; 2008

[7] Lewis R, Paechter B. Application of the grouping genetic algorithm to university course timetabling. In: Raidl G, Gottlieb J, editors. Evolutionary Computation in Combinatorial

Optimization. Berlin, Germany: Springer; 2005. pp. 144-153. LNCS 3448

[8] Massoodian S, Esteki A. A hybrid genetic algorithm for curriculum based course timetabling. In: Burke EK, Gendreau M, editors. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT'08); August 2008; Montréal, Canada; 2008

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

[10] Abramson D. Constructing school timetables using simulated annealing: Parallel and sequential solutions. Management Science. 1991;**37**(1):98-113. DOI: 10.1287/mnsc.37.1.98

[11] Meyers C, Orlin JB. Very large scale neighborhood search in timetabling problems. In: Proceeding of the 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT '06); Brno, Czech Republic; 2006

[12] Akinwale OC, Olatunde OS, Olusayo OE, Temitayo F. Hybrid metaheuristic of simulated annealing and genetic algorithm for solving examination timetabling problem. International Journal of Computer Science and Engineering - IJCSE. 2014;**3**(5):7-22

[13] Lawal HD, Adeyanju IA, Omidiora EO, Arulogun OT, Omotosho OI. University examination timetabling using Tabu Search. International Journal of Scientific and Engineering Research. 2014;**5**:10. Available from: http://www.ijser.org

[14] Leighton FT. A graph coloring algorithm for large scheduling problems.

Journal of Research - The National Bureau of Standards. 1979;**84**(6):489-506. DOI: 10.6028/jres.084.024

[15] Dandashi A, Al-Mouhamed M. Graph coloring for class scheduling. In: Proceeding of the 8th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2010); Hammamet, Tunisia; May 2010

[16] Soria-Alcaraz JA, Özcan E, Swan J, Kendall G, Carpio M. Iterated local search using an add and delete hyper-heuristic for university course timetabling. Applied Soft Computing. 2016;**40**:581-593. DOI: 10.1016/j.asoc.2015.11.043

[17] Jamal A. University course scheduling using the evolutionary algorithm. In: Proceeding of International Conference on Soft Computing, Intelligent System, and Information System (ICSIIT 2010); Bali, Indonesia; 2010. pp. 86-90

[18] Jamal A. A three stages approach of evolutionary algorithm and local search for solving the had-m and soft constrained course scheduling problem. In: Proceeding of the 11th Seminar on Intelligence Technology and its Application (SITIA2010); Surabaya, Indonesia; 2010. pp. 324-328

[19] Lutuksin T, Pongcharoen P. Experimental design and analysis on parameter investigation and performance comparison of ant algorithms for course timetabling problem. Naresuan University Engineering Journal. 2009;**4**:31-38

[20] Oner A, Ozcan S, Dengi D. Optimization of university course scheduling problem with a hybrid artificial bee colony algorithm. In: Proceeding of 2011 IEEE Congress of Evolutionary Computation (CEC 2011); 2011. pp. 339-346

[21] Bolaji AL, Khader AT, Al-Betara MA, Awadallah MA. University course timetabling using hybridized artificial bee colony with hill climbing optimizer. Journal of Computational Science. 2014;**5**(5):809-818. DOI: 10.1016/j.jocs.2014.04.002

[22] Ojha D, Sahoo RK, Das S. Automatic generation of timetable using firefly algorithm. International Journal of Advanced Research in Computer Science and Software Engineering. 2016;**6**(4):589-593

[23] Poli R, Kennedy J, Blackwell T. Particle swarm optimization: An overview. Swarm Intelligence. 2007;**1**(1):33-57. DOI: 10.1007/s11721-007-0002-0

[24] Shiau DF. A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences. Expert Systems with Applications. 2011;**38**(1):235-248. DOI: 10.1016/j.eswa.2010.06.051

[25] Jamal A. Multiple local scattered local search for course scheduling problem. In: Proceeding International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT 2017) IEEE; September 2017; Bali, Indonesia; 2017. DOI: 10.1109/ICSIIT.2017.22

[26] Forrest S, Mitchell M. Relative building-block fitness and the building-block hypothesis. In: Whitley D, editor. Foundations of Genetic Algorithms 2. San Mateo, CA: Morgan Kaufmann; 1993

[27] Trabzon SA, Pehlivan H, Dehkharghani R. Adaptation and use of artificial bee colony algorithm to solve curriculum-based course time-tabling problem. In: Proceeding of the 5th International Conference on Intelligent Systems, Modelling and Simulation; 2014. pp. 77-82

[28] Burke EK, Elliman DG, Weare RF. A genetic algorithm based university timetabling system. In: Proceeding of the 2nd East-West International Conference on Computer Technology in Education; September 1994; Crimea, Ukraine; 1994. pp. 35-40

[29] Hoos HH. Stochastic local search: Methods, models, application [Thesis]. Darmstadt, Germany: Technisen Universitat Darmstadt; 1998

[30] Hoos HH, Stutzle T. Stochastic local search: Foundation and applications. San Francisco: Elsevier; 2004