On the performance of Scatter Search for post-enrolment course timetabling problems

Article *in* Journal of Combinatorial Optimization · April 2014

DOI: 10.1007/s10878-012-9521-8

3 authors:

Ghaith M. Jaradat
Jerash University
62 PUBLICATIONS **98** CITATIONS
SEE PROFILE

Masri Ayob
Universiti Kebangsaan Malaysia
122 PUBLICATIONS **1,016** CITATIONS
SEE PROFILE

Zulkifli Ahmad
Universiti Kebangsaan Malaysia
13 PUBLICATIONS **58** CITATIONS
SEE PROFILE

J Comb Optim
DOI 10.1007/s10878-012-9521-8

On the performance of Scatter Search for post-enrolment course timetabling problems

Ghaith Jaradat · Masri Ayob · Zulkifli Ahmad

Abstract This study presents an investigation of enhancing the capability of the Scatter Search (SS) metaheuristic in guiding the search effectively toward elite solutions. Generally, SS generates a population of random initial solutions and systematically selects a set of diverse and elite solutions as a reference set for guiding the search. The work focuses on three strategies that may have an impact on the performance of SS. These are: explicit solutions combination, dynamic memory update, and systematic search re-initialization. First, the original SS is applied. Second, we propose two versions of the SS (*V1* and *V2*) with different strategies. In contrast to the original SS, SS*V1* and SS*V2* use the quality and diversity of solutions to create and update the memory, to perform solutions combinations, and to update the search. The differences between SS*V1* and SS*V2* is that SS*V1* employs the hill climbing routine twice whilst SS*V2* employs hill climbing and iterated local search method. In addition, SS*V1* combines all pairs (of quality and diverse solutions) from the *RefSet* whilst SS*V2* combines only one pair. Both SS*V1* and SS*V2* update the *RefSet* dynamically rather than static (as in the original SS), where, whenever a better quality or more diverse solution is found, the worst solution in *RefSet* is replaced by the new solution. SS*V1* and SS*V2* employ diversification generation method twice to re-initialize the search. The performance of the SS is tested on three benchmark post-enrolment

G. Jaradat (✉) · M. Ayob
Data Mining and Optimization Research Group, Centre of Artificial Intelligence Technology, Faculty of Information Science and Technology, School of Computer Science, The National University of Malaysia, 43600 UKM B. B. Bangi, Selangor, Malaysia
e-mail: ghaith_jaradat@yahoo.com

M. Ayob
e-mail: masri@ftsm.ukm.my

Z. Ahmad
Faculty of Social Science and Humanities, School of Language Studies and Linguistics, The National University of Malaysia, 43600 UKM B. B. Bangi, Selangor, Malaysia
e-mail: dzuan@ukm.my

Published online: 07 July 2012

course timetabling problems. The results had shown that SS*V2* performs better than the original SS and SS*V1* (in terms of solution's quality and computational time). It clearly demonstrates the effectiveness of using dynamic memory update, systematic search re-initialization, and combining only one pair of elite solutions. Apart from that, SS*V1* and SS*V2* can produce good quality solutions (comparable with other approaches), and outperforms some approaches reported in the literature (on some instances with regards to the tested datasets). Moreover, the study shows that by combining (simple crossover) only one pair of elite solutions in each *RefSet* update, and updating the memory dynamically, the computational time is reduced.

Keywords Scatter Search metaheuristic · Explicit solutions combination · Dynamic memory update · Systematic search re-initialization · Post-enrolment course timetabling problem

1 Introduction

University course timetabling is considered as an NP-hard (Even et al. 1976) (non-deterministic polynomial-time hard) optimization type problem (see Lewis 2008). There are various metaheuristic approaches used to solve this course timetabling problem.

Metaheuristic designates a computational method that optimizes a problem iteratively. The method can be classified into two classes: population-based and local search (Blum and Roli 2008).

Some common population-based methods applied to the problem are the ant colony optimization (Socha 2003; Rossi-Doria et al. 2003; Mayer et al. 2008), swarm optimization (Turabieh et al. 2010; Sabar et al. 2011), hybrid evolutionary algorithm and memetic algorithm (Lewis et al. 2007b; Jat and Yang 2010; Abdullah et al. 2007, 2010a). The population-based method is utilized because of its capability to explore search space and to combine easily with local search method in order to enhance solution exploitation mechanism (Talbi 2002). On the other hand, some common local search methods applied to the problem are tabu search (Rossi-Doria et al. 2003), simulated annealing (Kostuch 2005; Chiarandini et al. 2006; Ceschia et al. 2011), and iterated local search (Rossi-Doria et al. 2003). The local search method is utilized because of its capability to exploit solution space (Blum and Roli 2008).

The strength of population-based method relied on the capability of recombining solutions to obtain new ones (Blum and Roli 2008). In population-based algorithms such as the Scatter Search (SS), a structured solution recombination of elite solutions is performed explicitly (which involve moving/swapping of assignments in a solution representing the information exchange between generations about an elite solution) using one or more recombination operators, such as crossover and mutation (Blum and Roli 2008). This process enables the search to perform structured solutions combinations (Blum and Roli 2008). The term explicit means that a solution is represented directly by the actual assignments (e.g. *course1-timeslot44*, *course1-room2*) and their fitness values.

However, a population-based method is considered weak in intensifying the search to obtain higher quality solutions. Hence, in order to enhance the intensification process, a specialized metaheuristic in exploiting the solution space (e.g. hill climbing) is usually hybridized with population-based method. Many studies have recommended this hybridization, such as Blum and Roli (2008), Rossi-Doria et al. (2003), Eiben and Smith (2003), Qu et al. (2009), Talbi (2009). On the other hand, local search method has the capability to overcome the weakness (in the population-based method) of exploiting the solution space (further enhancement of a solution's quality). In addition, the utilization of an explicit memory (e.g. elite pool or reference set), to control the search diversity, and a dynamic manipulation of the population size are also recommended for better performance of hybrid metaheuristics (Talbi 2002; Greistorfer 2000). A good performance is presented by maintaining a balance between diversification and intensification of the search (Glover et al. 2002). Therefore, we have chosen the SS for this study, due to the following characteristics (Glover et al. 2002; Laguna and Marti 2003):

i. It provides a deterministic selection of reference set of elite solutions in terms of quality and diversity. This performs a systematic neighborhood search in the Euclidean or Hamming spaces.
ii. It has structured solution combinations using diversification strategies that do not merely rely on randomization.
iii. The search evolves a strategy of updating in a form of exploiting an adaptive memory to preserve good quality and diversity.
iv. It provides useful information about the collection of elite and diverse collection solutions.

This study mainly aims at illustrating the impact of employing a number of strategies (in the SS) that might speed up the search process and effectively guide the search toward a better quality solution. These strategies include the combination of one pair of elite solutions, an explicit elite solutions combination, a dynamic update of the memory, and a search re-initialization strategy via perturbing elite solutions in the memory. Based on the presented strategies, the study concludes the performance and consistency of SS*V1* and SS*V2* metaheuristic by testing them on post-enrolment course timetabling problems. The investigation concluded that by combining only one pair of elite solutions each *RefSet* update, and updating the memory dynamically, the computational time is reduced. Moreover, by considering both good quality and diverse solutions for updating the *RefSet* and combining solutions (alongside employing the diversification generation method twice), the search converges toward better quality solutions while escaping the local optima effectively. Hence, a balance between quality and diversity of the search can be maintained. Therefore, the proposed SS*V1* and SS*V2* perform better than the original SS (in terms of solution's quality and computational time). This demonstrated the effectiveness of using dynamic memory update, the systematic search re-initialization, and combining only one pair of elite solutions. The result findings also had shown that the proposed SS*V1* and SS*V2* can produce good quality solutions (comparable with other approaches), and outperform some approaches reported in the literature (on some instances with regards to tested datasets).

2 Problem description

In post-enrolment course timetabling problem, only the number of students enrolled in a course is considered in arranging courses to a particular room (student enrolment must not exceed the room capacity) for a specific period, while satisfying some constraints (Petrovic and Burke 2004). Due to the large variety of constraints in course timetabling problem that are required by different institutions, it would be almost impossible to model a generic course timetabling model that is applicable to every institution (Lewis et al. 2007a). Therefore, this study will tackle three well known post-enrolment course timetabling benchmark datasets as follows:

- Metaheuristics Network (2001) including Socha's instances (Socha et al. 2002): the problem containing 12 instances;
- TTComp2003 announced by Metaheuristics Network (2001): the problem containing 20 instances;
- ITC2007 (Track2) (Lewis et al. 2007a): a full formulation of the problem is introduced containing 24 instances.

The datasets can be downloaded from the website of Socha (http://iridia.ulb.ac.be/supp/IridiaSupp2002-001), the official website of TTComp2003 (http://www.idsia.ch/Files/ttcomp2002/), and the official website of ITC2007 (http://www.cs.qub.ac.uk/itc2007/).

The problem formulation for the post-enrolment course timetabling problems is defined by Lewis et al. (2007a). The original formulation was proposed by Metaheuristics Network (2001) and used for Socha, TTComp2003 and ITC2007 (Track2). The datasets of this formulation were originally generated by Ben Paechter (Metaheuristics Network 2001) who had designed an automatic generator of simulated real-world scenarios for the course timetabling problems. More details are provided by Rossi-Doria et al. (2003).

In this work, we use Socha, TTComp2003 and ITC2007 (Track2) datasets as platforms to evaluate the performance of proposed SS*V1* and SS*V2* methods because these datasets have gained wide interests over the past few years. Hence, this provides us with a rich comparison environment for a adequate and rational justification of the SS*V1* and SS*V2* performance and behavior over the problem. Socha et al. (2002) proposed 12 instances, but the 12th one (a large instance) has rarely been tackled in most past studies. This might be due to the extreme hardness of this dataset, where high percentage of failures of all studied methods, in finding even a feasible solution has been shown (Rossi-Doria et al. 2003; Socha et al. 2002, 2003). Therefore, the 12th instance (*aka large02* or *hard02*) was excluded from this study.

The benchmark problems are formulated as follows:

- A set of N courses needs to be scheduled into 5 working days a week of 9 timeslots each day, where $T = 45$ timeslots;
- A set of R rooms is given, where each room has a number of F features that include their capacities and other facilities;
- A number of M students will attend the course. Each student attends a number of courses with a given size of each room involved.

There are two types of constraints: hard and soft. In order to produce a good quality timetable, all hard constraints must be satisfied, whilst the violation of the soft constraints should be minimized. Each violation of soft constraints will incur a penalty cost, where lower penalty values indicate good quality solutions. The hard constraints are:

H1: No student attends more than one course at the same time;
H2: The room is big enough for all the attending students and satisfies all the features required by the course;
H3: Only one course is scheduled in each room at any timeslot;
H4: Events are only assigned to timeslots that are pre-defined as available for those events (applicable only to ITC2007-Track2);
H5: Where specified, events are scheduled to occur in the correct order in the week (applicable only to ITC2007-Track2).
H6: All courses must be scheduled to timeslots and rooms.

Then, a quality of timetable is measured by penalizing equally each violation of the following soft constraint (i.e. penalty cost = 1 for each violation on each student timetable). The soft constraints for the problem are:

S1: A student should not have a class in the last slot of the day;
S2: A student should not have more than two classes consecutively;
S3: A student should not have a single class on a day.

The objective function value of a timetable for each student is simply calculated as summation of hard (incur very high penalty cost) and soft constraint violations (as in Rossi-Doria et al. 2003). For more information on the problem formulation, please refer to Jat and Yang (2010) and Al-Betar et al. (2010).

3 Related work

During the last decade, a large variety of metaheuristics approaches were developed and applied to solve post-enrolment course timetabling problems. Recently, many studies developed hybrid metaheuristics rather than just concentrating on a standalone metaheuristic.

Some interesting implementations applied to the benchmark datasets (described in Sect. 2) are described as follows:

1. Socha et al. (2002, 2003) and Rossi-Doria et al. (2003) applied two ant colony optimization algorithms (max-min ant system, and ant colony system) to Socha's datasets (Metaheuristics Network 2001). Both algorithms performed very well in solution construction and found to be consistent in maintaining feasibility across many runs, which was achieved by the elitism strategy and the exploitation of pheromone trails. However, both algorithms could not obtain high quality solutions than the simulated annealing and iterated local search. In addition, the pheromone representation is very difficult to be utilized in estimating the improvement of a solution's quality.

2. Abdullah et al. (2007) and Abdullah and Turabieh (2008) presented two genetic based algorithms. The first is a genetic algorithm with a randomized iterative improvement. Only the mutation operator was employed while ignoring the crossover operator. Therefore, this may lead to an inefficient exploration of search space. The second is a typical genetic algorithm with a steepest descent local search to improve the quality of solutions; and a repair function to rectify infeasibility after the employment of crossover and mutation operators. Both algorithms obtained modest results compared to the best known ones.
3. Mayer et al. (2008) presented an ant system with a better utilization of pheromone trails in the form of a roulette wheel mechanism to select a feasible and good quality neighbors. The average of the pheromone trail over some period is calculated to determine whether to diversify or intensify the search at a certain point. However, their representation still not be adequate to gain a balance between diversity and quality of the search. In addition, there was no clear improvement phase (local search).
4. Turabieh and Abdullah (2009), Jat and Yang (2010), Yang and Jat (2011) developed hybrid algorithms that are based on genetic algorithms (memetic algorithms). In order to prevent reconstructing the same solution, they incorporated a memory into those algorithms, to store neighbors (or permutations) made to a solution in the improvement stage. A memory consisting of neighbors is useful for the estimation of solution improvement, but still not adequate to gain a balance between diversity and quality of the search.
5. Turabieh et al. (2009, 2010) developed two hybrid population-based metaheuristics: an electromagnetic-like mechanism with a great deluge algorithm (Turabieh et al. 2009); and a fish swarm with great deluge and steepest descent algorithms (Turabieh et al. 2010). In Turabieh et al. (2009) they utilized the concept of the electromagnetism to calculate the estimated quality for the great deluge. Whilst in Turabieh et al. (2009), they used a nelder-mead simplex algorithm to direct the search for promising areas in the search space. Although the results were found to be very good, both hybrid algorithms did not provide a clear strategy in maintaining a balance between the diversity and quality of the search. The exploration of a variety of neighborhood structures in both algorithms was also exhaustive.
6. The hybridization between simulated annealing and tabu search metaheuristics with a systematic exploration of various neighborhood structures were also widely investigated and successfully applied to the course timetabling problems. Such as Kostuch (2005), Chiarandini et al. (2006, 2008), Cambazard et al. (2012), Lewis (2010).
7. A variety of hybrid local search metaheuristics with a systematic exploration of neighborhood structures were developed by Ceschia et al. (2011), Burke et al. (2003), Di Gaspero and Schaerf (2006), Müller (2008), Atsuta et al. (2008), Shaker and Abdullah (2010), Abdullah et al. (2010b).

The best known results for Socha's datasets were obtained by simulated annealing algorithm (Ceschia et al. 2011); for the TTComp2003 datasets were obtained by three-phase simulated annealing (the official winner) (Kostuch 2005); and for the ITC2007 (Track2) datasets were obtained by hybridization of simulated annealing and tabu search (the official winner) (Cambazard et al. 2012). Generally, those approaches (hybrid local search) employed an intensive neighborhood search without a solution combination method or an explicit memory.

Most of the approaches applied to the problem did not use an explicit memory (which contains a diverse collection of elite solutions). They also lack a systematic selection strategy or a solution combination strategy, where this might be the reason behind their poor results. This may also present the lack of maintaining a balance between diversity and quality of the search. So far, the Scatter Search metaheuristic is the only standalone method that has an adaptive and explicit memory; a systematic selection; an explicit solution combination. This structure may enable it to maintain a balance between diversity and quality of the search. Therefore, the purpose of this study is to investigate its capability of solving the course timetabling problems to support the hypothesis of "*if the SS is able to maintain a balance between diversity and quality of the search, then it may be consistent in producing good quality timetables*".

4 The Scatter Search

Scatter Search (SS) is a population-based metaheuristic proposed by Glover (1977). The SS constructs solutions by combining elite solutions to exploit the problem-specific knowledge (e.g. good components of an elite solution). Recently, SS became one of the state-of-the-art methods for designing solution procedures for hard combinatorial optimization problems (Laguna 2009). Some investigation of the SS method can be found in Glover et al. (2002), Laguna and Marti (2003), Glover (1997), Martí et al. (2004), Resende et al. (2010).

There are two main differences between SS and other classical population-based metaheuristics (such as GAs) (Glover et al. 2002; Marti et al. 2006):

i. The size and content of elite solutions (*RefSet*): SS has a relatively small or moderate size (typical sizes between 10 and 40, according to Laguna and Marti 2003, Laguna 2009) that stores good and diverse solutions. Whilst, in other population-based approaches (such as GAs), the population size is typically larger (e.g. $\geq$100), which contains a randomly selected good quality solutions.
ii. The way the method combines the existing solutions to provide new ones: the SS systematically combines good quality and diverse solutions (parents) for reproduction. Whilst in GAs (for example), a population of solutions are evolved by using the mutation and crossover operators, which rely on randomization to choose parents for reproduction.

The SS consists of five component processes as described by Glover et al. (2002), Laguna and Marti (2003), Laguna (2009), Greistorfer and Voß (2005):

i. A *Diversification Generation Method* to generate a collection of diverse initial solutions as an input.
ii. An *Improvement Method* to enhance the quality of a trial solution using any local search to explore the neighbors of a solution.
iii. A *Reference Set Update Method* to build and maintain a reference set consisting of elite solutions, organized to provide structured solution combinations. The size

of the set is usually not more than 20 solutions. The Reference Set presents a huge diversity of the search.

iv. A *Subset Generation Method* to select solutions from the reference set, to produce a subset of its solutions as a basis for creating combined solutions. The most common subset generation method is to generate all pairs of reference solutions, namely Type-I selection (e.g. all subsets of size 2).
v. A *Solution Combination Method* to generate one or more solutions by combining good parts of a given subset of solutions produced by the *Subset Generation Method*. The combination method is analogous to the crossover operator in genetic algorithms but it must be capable of combining two or more solutions in the aspect of combining two elite solutions.

A generic pseudo code of our proposed SS is illustrated in Fig. 1. The SS systematically generates combinations of the reference solutions to create new ones, each of which is mapped into an associated feasible solution. An adaptive memory is exploited which attempts to avoid the search from re-investigating solutions that have already been evaluated. This is achieved by preventing duplication of reference solutions in the memory, which contains a diverse collection of elite solutions. This study will utilize a dynamic update method in the proposed SS*V1* and SS*V2* rather than the static update method applied in the original SS in order to converge effectively and efficiently toward an elite solution.

According to Glover et al. (2002, 2004), Burke et al. (2010), the basic design (shown in Fig. 1) can be expanded and improved in different ways. The SS methodology is very flexible, since each of its elements can be implemented in a variety of ways and degrees of sophistication.

Recently, SS has been successfully applied to a variety of combinatorial optimization problems, such as nurse rostering (Greistorfer and Voß 2005; Maenhout and Vanhoucke 2006), vehicle routing (Campos et al. 2008), examination timetabling (Mansour et al. 2009; Sabar and Ayob 2009) and flow shop scheduling (Engin et al. 2009) problems. The strategies and mechanisms of the SS have been comprehensively investigated, as well as the advances and applications have been reviewed in many studies such as Glover et al. (2002, 2004), Maenhout and Vanhoucke (2006). Further discussions on SS design and implementation can be found in Glover et al. (2002), Laguna and Marti (2003), Laguna (2009), Martí et al. (2004), Resende et al. (2010), Cotta (2004), Moscato and Cotta (2007).

5 The Scatter Search for course timetabling problems

In this study, the proposed SS*V1* and SS*V2* differ from the original SS algorithm (proposed by Glover 1977) in terms of population initialization; reference set update; subset generation method (possible combinations); solution combination; and search re-initialization. The proposed SS*V1* and SS$V2$ contribution is the utilization of methods and strategies in the SS for solving post-enrolment course timetabling problem, where:

i. The proposed SS*V1* and SS*V2* employ the *Diversification Generation method* twice (Steps 1 and 4 in Fig. 1) to maintain a high diversity of the search. First, is to generate a population of solutions; and then to generate a whole new population

```
Step 1: Start with $P = 0$.
        Use Diversification Generation Method to construct a solution.
Step 2: Apply the Improvement Method. Let $x$ be the resulting solution.
        If $x \notin P$ then add $x$ to $P$ (i.e., $P = P \cup x$), otherwise, discard $x$.
          Repeat this step until $|P| = PSize$.
          // employ Largest Degree ordering heuristic to construct a population of initial solutions
          // employ a repair function to rectify infeasible solutions
          // apply hill climbing to the whole population to enhance the quality of the initial solutions
        Step 3: Use the Reference Set Update Method to build RefSet (divided into two sets) with the
                "best quality" $b1 = \{x^1, \ldots, x^b\}$ solutions and; the "most diverse" $b2 = \{y^1, \ldots, y^b\}$
                solutions in $P$.
                Order the solutions in RefSet b1 according to their objective function values such that
                $x^1$ is the best solution and $x^b$ is the worst.
                Order the solutions in RefSet b2 according to their dissimilarity values (using
                $d_{\min}(p, q)$) such that $y^1$ is the best solution and $y^b$ is the worst.
                Make NewSolutions = TRUE.
        while   (NewSolutions and Stopping criterion is not met) do
        Step 4: Generate NewSubsets with the Subset Generation Method.
                  // explore only one pair of elite solutions (best quality $x^1$ and most diverse $y^1$) from
                  b1 and b2
                  Make NewSolutions = FALSE.
                while (NewSubsets $\neq \emptyset$) do // this is repeated 5 times as a maximum tries to update the
                RefSet
                  Step 5: Select the next subset $s$ in NewSubsets.
                          //subset s is the selected pair of solutions to be combined
                          // if the first pair did not update the RefSet, then explore another pair from b1
                          and b2
                  Step 6: Apply the Solution Combination Method to s to obtain one or more new so-
                          lutions $x$.
                          // apply crossover to the pair of elite solutions from RefSet
                  Step 7: Apply the Improvement Method to the trial solutions.
                          // apply a local search to the resulted solution from the Solution Combination
                          Method
                  Step 8: Apply the Reference Set Update Method.
                          // replace the worst quality solution $x^b$ in b1 by the newly generated solu-
                          tion $x$,
                          // or replace the worst diverse solution $y^b$ in b2
                  if      (RefSet has changed) then
                          Step 9: Make NewSolutions = TRUE. Delete $s$ from NewSubset.
                  else    Employ Diversification Generation Method to construct a new population of
                          solutions by performing some perturbations to the elite solutions in RefSet.
                          Apply the Improvement Method to the new population. // apply hill climbing
                  end if
                  Delete  $s$ from NewSubset.
                end while
        end while
        Step 10: Return the best found Solution
```

Fig. 1 The pseudo code of the SS based on Marti et al. (2006)

from solutions in the *RefSet* when the *Reference Set Update method* fails to update its contents within a predetermined number of trials (which is equal to 5 trial in this case). Whilst, in the original SS, the method is employed only once to generate a population of different solutions at the beginning of the search (only Step 1 in Fig. 1). This indicates that the original SS is "*very aggressive in trying* *to improve upon the quality of the solutions in the current reference set, to the extent that it sacrifices search diversity*" (Glover et al. 2004).

ii. The proposed SS*V1* and SS*V2* generate the whole population using a constructive heuristic rather than generating it by random construction. In the proposed SS*V1* and SS*V2*, a reference set was created (divided into two reference subsets) of best quality solutions and most diverse solutions (with respect to the best quality solutions in the first subset) from the population. Then, the *Subset Generation method* will select a pair of quality and diverse solutions to be combined, which represents the diversification of the search (Maenhout and Vanhoucke 2006).

iii. In the proposed SS*V2*, the *Subset Generation method* consists of generating only one pair of solutions in *RefSet*, in which only the best quality solution and the most diverse solution are subjected to the *Solution Combination method*. By generating one pair, it prevents wasting computational time in generating all possible subset combinations. Whilst, in the original SS the *Subset Generation method* consists of generating all pairs of solutions in *RefSet* that contain at least one new solution.

iv. The proposed SS*V2* updates the *RefSet* dynamically rather than statically to reduce computational time. In the *Dynamic Update strategy*, a new solution is immediately admitted to the *RefSet*. The goal is to allow this new solution to be applied in the *Solution Combination method* as quickly as possible (Glover et al. 2004). Whereas, in the *Static Update strategy*, the new solutions that become members of *RefSet* are not combined until all pairs in *Subset Generation method* are subjected to the *Solution Combination method* (Glover et al. 2004).

The proposed SS*V1* and SS*V2* start with the *Diversification Generation method* (*Step 1*) by generating a population of 50 initial solutions (as in Maenhout and Vanhoucke 2006) by using the largest degree ordering heuristic (LD) with a repair function (as in Shaker and Abdullah 2010; Landa-Silva and Obit 2008).

All solutions in the population are then improved in the first *Improvement method* (*Step 2*) using hill climbing (HC) local search until the search trapped in local optima (unable to find improve solution). The procedure tries to reach feasibility of solutions, by scheduling any unscheduled courses, performing free-clash moves, and hopefully improving the overall quality of population.

In the *Reference Set Update method* (*Step 3*), an adaptive memory (with the size of 20) is created (only in the first iteration) containing an initial reference set of solutions called (*RefSet*). These reference solutions are elite solutions selected from the population considering best quality and best diversity. This is to maintain diversity of the search. Elite solutions are determined here in this study as follows (similar to Mansour et al. 2009, where the size of the $RefSet = 0.4 \cdot |Pop\_size|$ is divided into two subsets):

i. select the 10 best quality solutions from the population and store them in a subset *b1*;

ii. measure the diversity of the rest of the solutions in the population by measuring their similarities to the best 10 solutions in the *RefSet*. This is done by counting the number of courses that occupies the same resources (timeslots) in two or more solutions;

iii. the 10 least similar solutions (from the population) to the best ones (in the *RefSet*) are selected as the best diverse solutions and stored in *b2* (as in Mansour et al. 2009). This means that those least similar ones have different solutions' structures derived from different regions of the search space. Similar update strategy was employed by Maenhout and Vanhoucke (2006).

The *Subset Generation method* (*Step 4* and *Step 5*) involves selecting two solutions from the *RefSet* to be combined to generate new solutions. In this study, the two solutions (one from *b1* and one from *b2*) are selected to produce off-spring, in which this process is considered as a deterministic selection similar to the "*Combination method 10*" in Martí et al. (2004), Burke et al. (2010), Campos et al. (2005). This deterministic or systematic selection is also known as the "controlled selection based on fitness/diversity value" as in Maenhout and Vanhoucke (2006). The combination of solutions is tackled using *Type-I* method, that combines all 2-elements subsets, meaning that combining all possible unrepeated two solutions. This solution combination type may maintain a balance between intensification and diversification of the search (Campos et al. 2005). Basically, the *Subset Generation method* is capable of covering different promising regions of the solution space, to avoid solution duplication (Laguna and Marti 2003); see Laguna (2009) for more details.

The *Solution Combination method* (*Step 6*) is performed using the *one-point crossover* operator to generate two new solutions at a time and selects the best. We implement the *one-point crossover* operator in a way of selecting 80 % (*crossover_rate*) of the first solution's assignments and swap them with the second solution's 80 % assignments. Hence, the proposed SS*V1* and SS*V2* might be able to maintain a small degree of randomization as a tie breaker in order to maintain a balance between diversification and intensification of the search.

Then the *Improvement method* (*Step 7*) will employ iterated local search (ILS) (from Rossi-Doria et al. 2003) and stops when reaching a number of non-improvement iterations (30 iterations). The idea is to significantly improve the quality of the combined solution (improve only one solution at a time) in order to obtain a better quality solution by exploring some of the solution's neighborhoods intensively and to escape from the local minimum. By exploring the neighbors of a solution via the ILS, the proposed SS*V2* may prevent reconstructing the same solutions (Dowsland and Thompson 2008). If the newly generated solution is infeasible, then a repair function is applied to rectify the solution as in Shaker and Abdullah (2010). If the repair function fails to rectify, then the solution combination is discarded. A neighborhood structure is randomly selected and performed in the improvement method. For example, the ILS selects four courses out of the total number of courses in the timetable/solution (e.g. 100) and performs moves or swaps to those four courses (*aka mutation_rate* value equals to 0.04). Three neighborhood structures are adopted from Socha (2003). These are: move a randomly selected course to a random feasible room and timeslot; swap timeslots and rooms of two randomly selected courses; and swap all courses of two randomly selected timeslots and rooms.

In the proposed SS*V2* (as recommended by Glover et al. 2004), the *RefSet* is dynamically updated by replacing the worst quality solution when a new solution has better quality than the worst solution in *b1*; otherwise, replace the worst diverse solution in *b2*. When there is no more good quality solution generated, the *Diversification*

Generation method is commenced again to generate a new population from solutions in the *RefSet* by performing some shakings.

The same similarities measurement is used as in Marti et al. (2006), Glover et al. (2004), Mansour et al. (2009), where the minimum distance (i.e. Manhattan distance) or dissimilarity $d_{\min}(p,q)$ between each solution (p) in the population and the solutions (q) currently in *RefSet* is calculated as follows:

$$d_{\min}(p,q) = \sum_{i=1}^{n} |p_i - q_i| \tag{1}$$

where, p_i is the ith solution in the population pool after the *Improvement method* in *Step 2*, whilst q_i is the ith solution in the *RefSet* and precisely in the subset *b1* (good quality solutions). As in Glover et al. (2004), then the proposed SS*V1* and SS*V2* select the solution that maximizes $d_{\min}(p) = \min_{q \in RefSet}\{d(p,q)\}$. This solution is then added to *RefSet* (*b*2) and deleted from the population. This process is repeated $b/2$ times. The resulting reference set b has $b/2$ high quality solutions and $b/2$ diverse solutions. The calculation of distance between solutions depends on the type of problem being solved (Marti et al. 2006). In the proposed SS*V1* and SS*V2*, the differences of course-timeslot assignments were counted as dissimilarity in both solutions (as in Mansour et al. 2009). That is, the distance between two solutions is a measure for diversity and is calculated as the number of different assignments between the two solutions (Maenhout and Vanhoucke 2006).

More specifically, as in Laguna and Marti (2003), Glover et al. (2004), for each generation (offspring solutions), the *RefSet* is updated as follows (*Step 8*):

i. if the newly generated solution is better than the worst solution in *b1*, then it replaces the worst solution in *b1*;
ii. if not, and if it is more diverse from those in *b2* w.r.t. to solutions in *b1*, then it replaces the worst diverse in *b2*;
iii. otherwise, the new solution is discarded.

Furthermore, if the pools in *b1* and *b2* are not modified for a fixed number of iterations (maximum number of *non_RefSetupdate* iterations) and the stopping criterion is not met, then repeat the *Diversification Generation method* (*Step 9*) to generate new solutions from the ones stored in the *RefSet*. Otherwise, the *Subset Generation method* is repeated until the stopping condition is met (goto *Step 4*). The SS is repeated until the stopping condition is met, which is the maximum number of iterations (*Step 10*).

6 Experimental results and discussions

In this work, the proposed SS*V1* and SS*V2* tested three versions of the SS on three sets of benchmark post-enrolment course timetabling instances (see Sect. 2). The proposed SS*V1* and SS*V2* run 25 times (for each version) on each instance of the three datasets, using the same parameters settings as in Table 1. The experiments were performed on Intel Pentium Core2 Duo 2.16 GHz processor, 2 GB RAM, and implemented in Java NetBeans IDE v 6.9. The best suited parameters and settings

Table 1 Parameters settings used by our SS versions

Parameter	Description and value
Pop_size	Population size = 50
max_iter	Maximum number of iterations = 100,000
stag_iter	Number of non-improvement iterations = 30
RefSet_size	Size of *RefSet* = 20 (*b1* = 10, *b2* = 10)
Local Search	Hill Climbing (for the original SS and SS*V1*); Iterated Local Search (for SS*V2*)
Subset_generation	Type-I selection, 2-elements subsets
Solution_combination	One-point crossover
Similarity_measurement	Least similar = best diverse
RefSet_update	Dynamic, once a better solution is found
Crossover_rate	Crossover rate = 0.8
Mutation_rate	Mutation rate = 0.04

of the proposed SS*V1* and SS*V2* were determined experimentally, and according to some studies from the literature, such as Glover et al. (2002), Laguna (2009), Marí et al. (2004, 2006), Greistorfer and Voß (2005).

First, the original SS is applied exactly as presented in Glover et al. (2002, 2004) Laguna (2009). The implementation of the original SS employs:

i. the *Diversification Generation method* only once;
ii. the HC in both *Improvement methods*;
iii. *Type-I Subset Generation method* by generating all possible subset combinations;
iv. a static *Reference Set Update method*;
v. discards all solutions in the reference set and generate totally new population built from scratch using *Diversification Generation method* when the search fails to update the contents of the reference set; and
vi. the reference set keeps only good quality solutions.

Then, some modifications to the SS algorithm are made, namely SS*V1*, and SS*V2*. Both SS*V1* and SS*V2* have the same structure of the original SS (as in Glover et al. 2002, 2004) except for one component, which is the local search. The local search employed in SS*V1* is the HC routine, and in SS*V2* is the ILS. Both SS*V1* and SS*V2* apply the *Diversification Generation method* twice.

Furthermore, SS*V2* has two significant modifications, which are: the *RefSet* is updated dynamically; and the *Subset Generation method* generates only one pair of best quality and most diverse solutions to be combined. The results obtained by the three versions are illustrated in Table 2.

The results (solutions' quality) obtained by the original SS are clearly far worse than the ones obtained by SS*V1* and SS*V2*. The computational time spent in obtaining those results was greater than the ones obtained by SS*V1* and SS*V2*. For example, for *S1* instance SS took nearly 300 seconds, whilst SS*V1* and SS*V2* took less than 30 seconds to obtain the optimal solution (penalty = 0). Note that, massive quality

Table 2 Computational statistics of the SS versions applied to Socha's instances

Instance	Original SS	SS*V1*				SS*V2*			
	best	best	Std.	median	worst	best	Std.	median	worst
S1	**0**	**0**	.89	0	2	**0**	.58	1	2
S2	**0**	**0**	.55	0	1	**0**	.51	1	1
S3	**0**	**0**	1	1	2	**0**	1.0	1	3
S4	**0**	**0**	.45	0	1	**0**	.33	0	1
S5	**0**	**0**	.45	0	1	**0**	0	0	0
M1	377	82	6.78	88	98	**70**	7.12	86	93
M2	344	80	11.39	90	107	**77**	6.04	86	98
M3	398	124	19.39	156	174	**115**	6.39	126	140
M4	236	82	6.87	92	101	**67**	8.37	73	88
M5	172	71	10.04	80	98	**64**	9.31	82	94
L	909	663	66.73	770	821	**555**	55.1	677	745

Note: *Std*, *median*, *best* and *worst* are the standard deviation, median value, best and worst quality scores of the 25 runs, respectively

differences (for *M1* and *L* instances) between the original SS and both SS*V1* and SS*V2*. This might be due to (in the original SS):

i. the *Diversification Generation method* is implemented only once, where once the *RefSet* is created, the successive population size will be always restricted to the size of the *RefSet*. Therefore, the search will rapidly converge toward a local minimum;
ii. all solutions in the *RefSet* are improved using the HC routine, which is known to be too weak to escape a local minimum. Therefore, no significant improvement of a solution's quality can be achieved;
iii. all possible and unrepeated 2-elements subsets (pairs) are generated. Therefore, the computational time is considerably exhaustive;
iv. the *RefSet* considers only keeping good quality solutions. Therefore, a small degree of diversity is maintained;
v. the *RefSet* is updated based on improving the quality of the worst solution which means a *Static Update strategy*;
vi. the search terminates when no new solutions are admitted to the *RefSet*. Therefore, enforcing the search to get stagnate in early stages.

Both SS*V2* and SS*V1* significantly outperformed the original SS (with regards to the quality of solution as reported in Table 2).

Results shown in Tables 2 indicate that SS*V2* has better performance (produces better quality of solution) and consistent (small standard deviation) than SS*V1* in obtaining good quality results in all runs (indicated by a small difference between the *best* and *median* values). As in the original SS, SS*V1* employed the HC routine two times, after the *Diversification Generation method* and the *Solution Combination method* to fulfill two purposes: (i) to enhance the quality of the initial population and to rectify infeasible solutions; and (ii) to direct the search toward the local optima

Table 3 The t-test of SS*V1* against SS*V2* applied to Socha's instances

	S1	*S2*	*S3*	*S4*	*S5*	*M1*	*M2*	*M3*	*M4*	*M5*	*L*
t-test	2.03	1.89	3.24	[a]	[a]	3.93	2.64	3.82	2.45	1.9	4.98
p-value	.053	.071	.004	–	–	.0006	.014	.0008	.021	.069	.00004

[a] t cannot be computed because the standard deviation is 0

(as mentioned by Petrovic and Burke 2004). The HC in SS*V1* is outperformed by the perturbation phase and the acceptance criterion of the ILS in SS*V2*. Since the HC is most likely to get trapped in local minima, the SS and SS*V1* has less chance to further diversify the search and then the solution combination process would have less effect to generate new promising and good quality solution. Therefore, this study overcomes this shortage by hybridizing the ILS into the SS (in SS*V2*), where ILS perturbs the newly generated solution and/or escapes the local minima by accepting poor quality solution.

Table 3 shows the t-test and p-values to demonstrate the significant improvement of SS*V2* over SS*V1*. The t-test is carried out with 24 degree of freedom at a .05 level of significance. SS*V2* shows its consistency in obtaining optimal results (penalty = 0) for *S4* and *S5* across 25 runs (indicated by t-test values). Indicated by p-value, SS*V2* shows a relatively high value of differences (in term of consistent results) compared to SS*V1* for most of the datasets (e.g. see *S2* and *M5*). Apart from that, SS*V1* and SS*V2* seemed to be performing similarly on the dataset *L* (.00004).

Table 4 shows the best results obtained by SS*V2* compared to similar population-based approaches and state-of-the-art metaheuristics. The results of this study are competitive to all approaches in Table 4. SS*V2* outperforms all GA-based approaches in most of the datasets (see Table 4) and relatively consistent in producing optimal solution (penalty = 0) for all small sized instances (*S1*–*S5*) (refer to the small difference between *best* and *median*, and small *Std.* in Table 2). The consistency of our SS in obtaining feasible and good quality results for some datasets is excellent, especially for the *S5* dataset (indicated by the standard deviation value, *Std.* = 0; and the difference between *median* and *best* is 0).

Furthermore, in an attempt to support our hypothesis of the effectiveness, efficiency and consistency of our implementation (SS*V2*), this study had carried out further experiments by testing SS*V2* on two other well-known datasets of post-enrolment course timetabling problems (TTComp2003 and ITC2007-Track2). Tables 5 and 6 show the results obtained by SS*V2* (under competitions' stopping condition, i.e. 474 seconds for TTComp2003 and 494 seconds for ITC2007) compared to the best known and approaches. This study had performed 25 runs for each instance for both competitions. Only the best results (out of 25 runs) obtained by SS*V2* are presented in Tables 5 and 6.

Notice that, across all runs for each instance of both competitions' datasets (Tables 5 and 6), SS*V2* was able to obtain feasible solutions. SS*V2* had obtained high quality results compared to most of the presented approaches in Table 5 (either similar to or better than other approaches for most of the instances).

Table 4 The SS*V2* compared to similar approaches applied on the Socha's test instances

Instance	Population-based									Local search		
	SS*V2*	*MM AS*	*GA SD*	*EM GD*	*HEA*	*EGS GA*	*TMA*	*FSI*	*HB MO*	*RRM*	*DSA*	*SA*
S1	**0**	1	2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
S2	**0**	3	4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
S3	**0**	1	2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
S4	**0**	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
S5	**0**	**0**	4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
M1	70	195	254	96	221	139	50	45	75	117	93	**9**
M2	77	184	285	96	147	92	70	40	88	108	98	**15**
M3	115	248	251	135	246	122	102	61	129	135	149	**36**
M4	67	164.5	321	79	165	98	32	35	74	75	103	**12**
M5	64	219.5	276	87	130	116	61	49	64	160	98	**3**
L	555	851.5	1027	683	529	615	653	407	523	589	680	**208**

Notes:
- MMAS: Max-Min Ant System (Socha et al. 2002), results shown are the average values
- GASD: Hybrid of Genetic Algorithm and Steepest Descent (Abdullah and Turabieh 2008)
- EMGD: Hybrid of Electromagnetic-like Mechanism with force decay rate Great Deluge (Turabieh et al. 2009)
- HEA: Hybrid Evolutionary Algorithm—genetic operators and Randomized Iterative Improvement (Abdullah et al. 2007)
- EGSGA: Extended Guided Search Genetic Algorithm (Yang and Jat 2011)
- TSMA: Incorporation of Tabu Search into Memetic Algorithm (Turabieh and Abdullah 2009)
- FSI: Fish Swarm Intelligence (Turabieh et al. 2010)
- HBMO: Honey Bee Mating Optimization (Sabar et al. 2011)
- RRM: Controlling multi algorithms (Simulated Annealing, Great deluge, Hill Climbing) using Round Robin (Shaker and Abdullah 2010)
- DSA: Dual Simulated Annealing (Abdullah et al. 2010b)
- SA: Simulated Annealing approach (Ceschia et al. 2011)

In the 2nd competition ITC2007 (Track2), Table 6 shows that SS*V2* had obtained an optimality (penalty cost = 0) for one third of the instances (*comp5*, *comp6*, *comp8*, *comp13*, *comp14*, *comp15*, *comp17*, *comp18*, and *comp21*) and comparable to the best known results reported in the literature (e.g. *comp24*, cost = 3) in others instances. SS*V2* had also obtained a new best result for the *comp12* (cost = 14) instance. In addition, SS*V2* outperformed some approaches for most instances shown in Table 6 especially the population-based approaches in terms of quality of solution.

A good performance of SS*V2* might be attributed to the search evolution behavior (search experience update), where strategic update (deterministic rules) was performed to preserve quality and diversity of the search. This behavior relies on the use of an adaptive and explicit memory (*RefSet*), which is confined during the search process to maintain a balance between the diversification and intensification. A structured (explicit) solution combination considers good quality and good diverse solutions (from the *RefSet*) was employed to ensure that balance. This was achieved by combining good parts (dissimilar parts/course-timeslot assignments within two

Table 5 The SS*V2* compared to similar approaches applied on the TTComp2003 instances

Instance	Population-based					Local search				
	SS*V2*	*MMAS*	*AS*	*EMGD*	*EGSGA*	*3-SAx*	*GD*	*LS*	*HA*	*SA*
comp01	37	65	55	52	54	**16**	85	63	45	45
comp02	12	36	43	20	25	**2**	42	46	14	20
comp03	40	69	61	78	44	**17**	84	96	45	43
comp04	75	138	134	74	132	**34**	119	166	71	87
comp05	54	143	134	71	97	**42**	77	203	59	71
comp06	**0**	24	32	6	3	**0**	6	92	1	2
comp07	**2**	24	52	6	12	**2**	12	118	3	**2**
comp08	**0**	28	48	15	23	**0**	32	66	1	9
comp09	14	36	39	32	21	**1**	184	51	8	15
comp10	58	75	77	58	53	**21**	90	81	52	41
comp11	32	50	39	30	46	**5**	73	65	30	24
comp12	64	95	102	88	96	**55**	79	119	75	62
comp13	57	79	94	105	69	**31**	91	160	55	59
comp14	20	73	109	51	13	**11**	36	197	18	21
comp15	18	31	47	34	35	**2**	27	114	8	6
comp16	5	23	26	10	12	**0**	300	38	5	6
comp17	68	108	78	121	104	**37**	79	212	46	42
comp18	20	26	35	26	39	**4**	39	40	24	11
comp19	40	108	119	57	63	**7**	86	185	33	56
comp20	**0**	5	19	5	2	**0**	**0**	17	**0**	**0**

Notes:

- MMAS: Max-Min Ant System (Socha 2003)
- AS: Ant System (Mayer et al. 2008)
- EMGD: Hybrid of Electromagnetic-Like mechanism with force decay rate Great Deluge (Abdullah et al. 2010a)
- EGSGA: Extended Guided Search Genetic Algorithm (Yang and Jat 2011)
- 3-SAx: Extended work of the official winner with some refinements and greater number of iterations equals to 1 million (Kostuch 2005)
- GD: Great Deluge (Burke et al. 2003)
- LS: Local Search (Di Gaspero and Schaerf 2006)
- HA: Hybrid metaheuristic, a mixture of constructive heuristics (including local search and Tabu Search), Variable Neighborhood Descent & Simulated Annealing (Chiarandini et al. 2006)
- SA: Simulated Annealing approach (Ceschia et al. 2011)

solutions) from both good quality and good diverse solutions in the form of *one-point crossover* in order to reach a global optimum solution.

On the other hand, this study might not be able to outperform the best known results due to the lack of efficiency of the intensification mechanism. Although our SS*V2* has two improvement methods (HC and ILS), SS*V2* may have not performed well in making a significant modification to a solution due to the way this study explored the neighbors of the elite solution, which was carried out using random exploration method. A systematic exploration of an elite solution's neighbors could

Table 6 The SS*V2* compared to similar approaches applied on the ITC2007 (Track2) instances

Instance	Local search							Population-based	
	SS*V2*	*CTI*	*HA*	*LSA*	*HSAT*	*TSSA*	*SA*	*AS*	*HGATS*
comp1	470	61	1482	1861	1166	571	59	**15**	523
comp2	920	547	1635	*inf*	1665	993	**0**	**0**	342
comp3	194	382	288	272	251	**164**	148	391	379
comp4	219	529	385	425	424	310	**25**	239	234
comp5	**0**	5	559	8	47	5	**0**	34	**0**
comp6	**0**	**0**	851	28	412	**0**	**0**	87	**0**
comp7	6	**0**	10	13	6	6	**0**	**0**	**0**
comp8	**0**	**0**	**0**	6	85	**0**	**0**	4	**0**
comp9	979	**0**	1947	*inf*	1819	1560	*inf*	**0**	1102
comp10	447	**0**	1741	*inf*	2091	2163	*inf*	**0**	515
comp11	233	548	240	263	288	178	**142**	547	246
comp12	**14**	869	475	804	474	146	267	32	241
comp13	**0**	**0**	675	285	298	**0**	1	166	**0**
comp14	**0**	**0**	864	110	127	1	**0**	**0**	**0**
comp15	**0**	379	**0**	5	108	**0**	**0**	**0**	**0**
comp16	1	*inf*	1	132	138	2	**0**	41	**0**
comp17	**0**	1	5	72	**0**	**0**	**0**	68	**0**
comp18	**0**	**0**	3	70	25	**0**	**0**	26	**0**
comp19	1531	*inf*	1868	*inf*	2146	1824	*inf*	22	**121**
comp20	534	1215	596	878	625	445	543	*inf*	**304**
comp21	**0**	**0**	602	40	308	**0**	*inf*	33	36
comp22	2359	**0**	1364	889	*inf*	29	5	**0**	1154
comp23	982	430	688	436	3101	**238**	*inf*	1275	963
comp24	3	720	822	372	841	21	**0**	30	274

Notes:
- AS: Ant System (Mayer et al. 2008)
- HGATS: Hybrid Genetic Algorithm with Tabu Search (Jat and Yang 2010)
- CTI: Combination of general purpose Constraint Satisfaction Solver, Tabu Search, & Iterated Local Search (Atsuta et al. 2008)
- HA: Combination of constructive procedure to achieve feasibility & a Simulated Annealing (Chiarandini et al. 2008)
- LSA: Local search algorithm taken from the Constraint Solver Library combined with Variable Neighborhood Search algorithms (Müller 2008)
- TSSA: Combination of Tabu Search & Simulated Annealing in conjunction with various neighborhood operators (Cambazard et al. 2012). The official winner
- HSAT: Hybrid approach which combines a constructive heuristic, two separate phases of Simulated Annealing & Neighborhood operators, & it is time dependent (Lewis 2010)
- SA: Simulated Annealing approach (Ceschia et al. 2011)
- *inf*: no feasible solution was obtained, with distance to feasibility ($DF > 0$)

be a better alternative, such as exploiting a *roulette wheel selection* mechanism to select a neighborhood structure.

However, an efficient exploration was achieved by utilizing an elite collection of good quality and diverse solution (*RefSet*) to prevent a premature convergence of the search. Solutions from the *RefSet* were utilized to generate new promising solutions (rather than from scratch) to restart the search with a new diversified population but close in quality to the elite solutions. The *RefSet* provided useful information about the location and structure of the global solution presented by the (dis)similarity measurement between solutions in the population and the elite ones.

Effective solution exploitation was achieved by the *Solution Combination method* (a systematic selection of solutions and a structure recombination) and the ILS routine (a significant quality improvement). Where, they performed further exploration of an elite solution's neighbors to significantly enhance the quality of the solution. Besides, the dynamic update of the *RefSet* provided a fast exchange of useful information (about the elite solution in hand) between generations, such as, whether to keep or discard a number of solutions in each generation to maintain a good quality while diversifying the search.

In addition, a reasonable (that is neither too strong, nor too weak) degree of the search diversity was maintained by considering the measurement of dissimilarity course-timeslot assignments occurred in the solutions compared to the elite solution. In which, from this study point of view, if the course-room assignments were also considered in the dissimilarity measurements, it might be very restrictive and hence, a few number of diverse solutions will have the possibility to update the *RefSet*. The reason behind the selection of the course-timeslot pair regardless the room, was due to the timeslot permutations which had a larger impact on the quality of the timetable significantly more than the room (Petrovic and Burke 2004) (actually, in these datasets, no soft constraint was associated with room assignment, i.e. it did not affect the quality of solution but the course must be assigned to a room).

By contrast, SS*V2* had failed to produce a good quality results (for the *comp22*, in Table 6) compared to others. This is due to the diversity control which was not quite efficient enough to take such a hard scenario hence the dataset was considered as the hardest among others. The observations of SS*V2* behavior over this dataset indicated that their convergence toward good quality solutions was somewhat concentrated on the same regions that contain probably similar solutions' structures regardless their fitness cost. This might be caused by the sequential constructive algorithm (e.g. *LD*) that this study had used in the SS, where feasibility were granted easily but the fitness cost value of all initial solutions for this dataset were relatively small. This means that the search will proceed with a small cost value that is nearly close to its neighbors cost values. Most likely, the structures of these neighbors were extremely similar to many initial solutions produced earlier in the population. This might decreased the strength of diversity control process. Therefore, there were not many chances left for the local search routine to significantly enhance the quality of elite solutions. This issue left an unanswered question of "*how to find the right degree of search diversification*". Hence, the question is subjected to the future work. It can also be assumed that the employed neighborhood structures (or the way they are explored) in SS*V2* may be inappropriate or insufficient for this particular instance.

7 Conclusions

This study had investigated the importance of employing the following strategies in Scatter Search (SS): exploring a smaller number of explicit combinations, a dynamic update of the memory, and search re-initialization strategy via perturbing solutions in the memory. Those strategies provided a deterministic search by maintaining a balance between diversity and quality of the population. Where, a small number of combinations and a dynamic update reduce the computational time. In addition, the *RefSet* considers quality and diverse solutions in which to guide the search effectively toward better quality solutions while escaping the local optima. Finally, by utilizing the elite solutions in the memory, the search was re-initialized within promised regions of the search space once the *RefSet* was unable to be updated.

This study had made some modifications to the SS algorithm, namely SS*V1*, and SS*V2*. Both modified versions have the same structure of the original SS except for the local search employed in SS*V1* is the HC (hill climbing) routine, and ILS (iterated local search) in SS*V2*. Both, SS*V1* and SS*V2* applied the *Diversification Generation method* twice to trigger the search again once it stagnates. Both, SS*V1* and SS*V2* update the *RefSet* dynamically to speed up the search. SS*V2* has a major difference from SS*V1* and the original SS, which is the number of pairs obtained from the *Subset Generation method*. Where, SS*V2* selects only one pair of elite solutions to be combined in the *Combination method* (a good quality solution and a high diverse solution). This is to speed up the search while concentrating on good quality solutions (searching around the neighbors of a local optimum solution).

The SS*V2* is able to produce good quality solutions and in some cases has obtained high quality ones (which are competitive or even better than others reported in the literature). This was due to maintaining a balance between diversification and intensification of the search (by injecting ILS in SS*V2* to intensify the search). SS*V2* maintains the search diversity by employing the diversification generation method twice, first to construct a population from scratch, and then reinitializes the population by performing simple shakings to the solutions in the *RefSet*. Hence, SS*V2* keeps good quality and diverse solutions in hand, where the worst solution is replaced once a better quality or better diversity solution is found.

However, the SS is still not good enough in determining the right degree of the search diversity especially in larger size problem. In future, this study will investigate some alternative selection and/or recombination mechanisms of elite solutions and/or neighborhood exploration mechanisms. This is needed so as to further understand how to achieve an efficient convergence toward better quality solutions, as well as better performance and consistency. In order to fulfill the purpose of generalization of a metaheuristic across a variety of applications, it might be interesting to apply this proposed SS*V2* metaheuristic to other combinatorial optimization problems.

Acknowledgements The authors wish to thank Ministry of Higher Education (Malaysia) for supporting this work under the Fundamental Research Grant Scheme (FRGS) no. (FRGS/1/2012/SG05/UKM/02/11).

References

Abdullah S, Turabieh H (2008) Generating university course timetable using genetic algorithms and local search. In: Proceedings of the 3rd international conference on convergence and hybrid information technology (ICCIT 2008). IEEE Comput Soc, Los Alamitos, pp 254–260

Abdullah S, Burke EK, McCollum B (2007) A hybrid evolutionary approach to the university course timetabling problem. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2007) pp 1764–1768. ISBN 1-4244-1340-0

Abdullah S, Turabieh H, McCollum B, McMullan P (2010a) A hybrid metaheuristic approach to the university course timetabling problem. J Heuristics. doi:10.1007/s10732-010-9154-y

Abdullah S, Shaker K, McCollum B, McMullan P (2010b) Dual sequence simulated annealing with round-robin approach for university course timetabling. In: Cowling P, Merz P (eds) EVOCOP 2010. LNCS, vol 6022. Springer, Heidelberg, pp 1–10

Al-Betar M, Khader A, Liao I (2010) A harmony search with multi-pitch adjusting rate for the university course timetabling. In: Geem, ZW (ed) Recent advances in harmony search algorithm. SCI, vol 270. Springer, Heidelberg, pp 147–161

Atsuta M, Nonobe K, Ibaraki T (2008) ITC2007 track 2: an approach using general CSP solver. http://www.cs.qub.ac.uk/itc2007 [20 December 2010]

Blum C, Roli A (2008) Hybrid metaheuristics: an introduction, studies in computational intelligence. In: Blum C, Aguilera MJB, Roli A, Samples M (eds) Hybrid metaheuristics: an emerging approach to optimization. SCI, vol 114. Springer, Berlin, pp 1–30

Burke EK, Bykov Y, Newall JP, Petrovic S (2003) A time-predefined approach to course timetabling. Yugosl J Oper Res 13(2):139–151

Burke EK, Curtois T, Qu R, Berghe GV (2010) A scatter search approach to the nurse rostering problem. J Oper Res Soc 6:1667–1679

Cambazard H, Hebrard E, O'Sullivan B, Papadopoulos A (2012) Local search and constraint programming for the post enrolment-based course timetabling problem. Ann Oper Res 194(1):111–135. Url:http://dx.doi.org/10.1007/s10479-010-0737-7

Campos V, Laguna M, Martí R (2005) Context-independent scatter search and tabu search for permutation problems. INFORMS J Comput 17(1):111–122. ISSN 0899-1499

Campos V, Corberan A, Mota E (2008) A scatter search algorithm for the split delivery vehicle routing problem. In: Fink A, Rothlauf F (eds) Advances in computational intelligence. SCI, vol 144. Springer, Berlin, pp 137–152

Ceschia S, Di Gaspero L, Schaerf A (2011) Simulated annealing approach to post-enrolment course timetabling problem. J Comput Oper Res. doi:10.1016/j.cor.2011.09.014

Chiarandini M, Birattari M, Socha K, Rossi-Doria O (2006) An effective hybrid algorithm for university course timetabling. J Sched 9(5):403–432

Chiarandini M, Fawcett C, Hoos HH (2008) A modular multiphase heuristic solver for post enrollment course timetabling. In: Proceedings of the 7th international conference on the practice and theory of automated timetabling (PATAT 2008)

Cotta C (2004) Scatter search and memetic approaches to the error correcting code problem. In: Gottlieb J, Raidl GR (eds) EvoCOP 2004. LNCS, vol 3004. Springer, Berlin, pp 51–61

Di Gaspero L, Schaerf A (2006) Neighborhood portfolio approach for local search applied to timetabling problems. J Math Model Algorithms 5(1):65–89

Dowsland KA, Thompson JM (2008) An improved ant colony optimisation heuristic for graph colouring. Discrete Appl Math 156:313–324

Eiben AE, Smith JE (2003) Introduction to evolutionary computing, 1st edn. Springer, Berlin. Corrected 2nd printing, 2007. ISBN 978-3-540-40184-1

Engin O, Kahraman Y, Yilmaz MK (2009) A scatter search method for multiobjective fuzzy permutation flow shop scheduling problem: a real world application. In: Chakraborty UK (ed) Comput intel in flow shop and job shop sched. SCI, vol 230. Springer, Berlin, pp 169–189

Even S, Itai A, Shamir A (1976) On the complexity of timetable and multi commodity flow problem. SIAM J Comput 5:691–703

Glover F (1977) Heuristics for integer programming using surrogate constraints. Decis Sci 8:156–166

Glover F (1997) A template for scatter search and path relinking. In: Hao JK, Lutton E, Ronald E, Schoenauer M, Snyers D (eds.) LNCS, vol 1363, pp 13–54

Glover F, Laguna M, Martí R (2002) Scatter search. In: Ghosh A, Tsutsui S (eds) Theory and applications of evolutionary computation: recent trends. Springer, Berlin, pp 519–529

Glover F, Laguna M, Martí R (2004) Scatter search and path relinking: foundations and advanced designs. In: Onwubolu G, Babu BV (eds) New optimization techniques in engineering. Springer, Berlin

Greistorfer P (2000) On the algorithmic design in heuristic search. In: European conference on operational research (EURO XVII), Budapest, Ungarn, pp 16–19

Greistorfer P, Voß S (2005) Controlled pool maintenance in combinatorial optimization. In: Rego C, Alidaee B (eds) Conference on adaptive memory and evolution: tabu search and scatter search, University of Mississippi. Kluwer Academic, Dordrecht, pp 387–424. Chap 18

Jat SN, Yang S (2010) A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. J Sched. doi:10.1007/s10951-010-0202-0

Kostuch P (2005) The university course timetabling problem with a 3-phase approach. In: Burke EK, Trick M (eds) The practice and theory of automated timetabling V (PATAT 2004). LNCS, vol 3616. Springer, Berlin, pp 109–125

Laguna M, Marti R (2003) Scatter search: methodology and implementations in C. Kluwer Academic, Boston

Laguna M (2009) Scatter search and path relinking. In: Ehrgott M et al (eds) EMO 2009. LNCS, vol 5467. Springer, Berlin, p 1

Landa-Silva D, Obit JH (2008) Great deluge with nonlinear decay rate for solving course timetabling problems. In: Proceedings of the 2008 IEEE conference on intelligent systems (IS 2008). IEEE Press, New York, pp 8.11–8.18

Lewis R (2008) A survey of metaheuristic-based techniques for university timetabling problems. OR Spektrum 30:167–190

Lewis R (2010) A time-dependent metaheuristic algorithm for post enrolment-based course timetabling. J Ann Oper Res. doi:10.1007/s10479-010-0696-z

Lewis R, Paechter B, McCollum B (2007a) Post enrolment based course timetabling: a description of the problem model used for track two of the second international timetabling competition. In: Cardiff working papers in accounting and finance A2007-3, Cardiff Business School, Cardiff University, Wales. ISSN 1750-6658

Lewis R, Paechter B, Rossi-Doria O (2007b) Metaheuristics for university course timetabling. In: Dahal K, Chen Tan K, Cowling P (eds) Evolutionary scheduling. Studies in computational intelligence, vol 49. Springer, Berlin, pp 237–272

Maenhout B, Vanhoucke M (2006) New computational results for the nurse scheduling problem: a scatter search algorithm. In: Gottlieb J, Raidl GR (eds) EvoCOP 2006. LNCS, vol 3906. Springer, Berlin, pp 159–170

Mansour N, Isahakian V, Ghalayini I (2009) Scatter search technique for exam timetabling. Applied intelligence. Springer, Berlin

Martí R, Laguna M, Campos V (2004) Scatter search vs. genetic algorithms: an experimental evaluation with permutation problems. In: Rego C, Alidaee B (eds) Metaheuristic optimization via adaptive memory and evolution: tabu search and scatter search. Kluwer Academic, Dordrecht, pp 263–282. Chap 12

Marti R, Laguna M, Glover F (2006) Principles of scatter search. Eur J Oper Res 169:359–372

Mayer A, Nothegger C, Chwatal A, Raidl G (2008) Solving the post enrolment course timetabling problem by ant colony optimization. In: Proceedings of the 7th international conference on the practice and theory of automated timetabling (PATAT 2008)

Metaheuristics Network (2001). The official website: http://www.idsia.ch/Files/ttcomp2002/. Accessed in 20 June 2011

Moscato PA, Cotta C (2007) Memetic algorithms. In: Handbook of approximation algorithms and metaheuristics. Taylor & Francis, Boca Raton, pp 27-1–27-12

Müller T (2008) ITC2007 solver description: a hybrid approach. In: Proceedings of the 7th international conference on the practise and theory of automated timetabling (PATAT 2008)

Petrovic S, Burke EK (2004) University timetabling. In: Leung J (ed) Handbook of scheduling: algorithms, models and performance analysis. CRC Press, Boca Raton. Chap 45

Qu R, Burke EK, McCollum B, Merlot LTG, Lee SY (2009) A survey of search methodologies and automated system development for examination timetabling. J Sched 12:55–89

Resende MGC, Ribeiro CC, Glover F, Marti R (2010) Scatter search and path-relinking: fundamentals, advances, and applications. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics, 2nd edn. Springer, Berlin.

Rossi-Doria O, Samples M, Birattari M, Chiarandini M, Dorigo M, Gambardella LM, Knowels J, Manfrin M, Mastrolilli M, Paechter B, Paquete L, Stultzle T (2003) A comparison of the performance of different metaheuristics on the timetabling problem. In: Burke EK, De Causmaecker P (eds) PATAT 2002. LNCS, vol 2740. Springer, Heidelberg, pp 329–354

Sabar NR, Ayob M (2009) Examination timetabling using scatter search hyper-heuristic. In: The 2nd data mining and optimization conference (DMO 2009), vol I, pp 127–131

Sabar R, Ayob M, Kendall G, Qu R (2011) A honey-bee mating optimization algorithm for educational timetabling problems. Eur J Oper Res. doi:10.1016/j.ejor.2011.08.006

Shaker K, Abdullah S (2010) Controlling multi algorithms using round robin for university course timetabling problem. In: Zhang Y et al (eds) DTA/BSBT 2010. CCIS, vol 118. Springer, Heidelberg, pp 47–55

Socha K, Knowles J, Samples M (2002) A max-min ant system for the university course timetabling problem. In: Dorigo M, Di Caro GA, Sampels M (eds) Ant algorithms 2002. LNCS, vol 2463. Springer, Heidelberg, pp 1–13

Socha K (2003) The influence of run-time limits on choosing ant system parameters. In: Cantu-Paz E et al (eds) GECCO 2003. LNCS, vol 2723. Springer, Berlin, pp 49–60

Socha K, Samples M, Manfrin M (2003) Ant algorithm for the university course timetabling problem with regard to the state-of-the-art. In: Proceedings of the 3rd European workshop on evolutionary computation in combinatorial optimisation, Essex, UK. LNCS, vol 2611. Springer, Berlin, pp 334–345

Talbi EG (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8:541–564

Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, New York

Turabieh H, Abdullah S (2009) Incorporating tabu search into memetic approach for enrolment-based course timetabling problems. In: The 2nd data mining and optimization conference (DMO 2009), pp 122–126

Turabieh H, Abdullah S, McCollum B (2009) Electromagnetism-like mechanism with force decay rate great deluge for the course timetabling problem. In: Wen P, Li Y, Polkowski L, Yao Y, Tsumoto S, Wang G (eds) RSKT 2009. LNCS, vol 5589. Springer, Heidelberg, pp 497–504

Turabieh H, Abdullah S, McCollum B, McMullan P (2010) Fish swarm intelligent algorithm for the course timetabling problem. In: Yu J et al (eds) RSKT 2010. LNAI, vol 6401. Springer, Heidelberg, pp 588–595

Yang S, Jat SN (2011) Genetic algorithms with guided and local search strategies for university course timetabling. IEEE Trans Syst Man Cybern, Part C, Appl Rev 41(1):93–106