Particle Swarm Optimisation Variants and Its Hybridisation Ratios for Generating Cost-Effective Educational Course Timetables

Thatchai Thepphakorn · Saisumpan Sooncharoen · Pupong Pongcharoen

Received: 19 October 2020 / Accepted: 17 April 2021 / Published online: 8 May 2021
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd 2021

Abstract

Due to the COVID-19 pandemic, many universities across the globe are unexpectedly accelerated to face another major financial crisis. An effective university course timetabling has a direct effect on the utilisation of the university resources and its operating costs. The university course timetabling is classified to be a Non-deterministic Polynomial (NP)-hard problem. Constructing the optimal timetables without an intelligence timetabling tool is extremely difficult task and very time-consuming. A Hybrid Particle Swarm Optimisation-based Timetabling (HPSOT) tool has been developed for optimising the academic operating costs. In the present study, two variants of Particle Swarm Optimisation (PSO) including Standard PSO (SPSO) and Maurice Clerc PSO (MCPSO) were embedded in the HPSOT program. Five combinations of Insertion Operator (IO) and Exchange Operator (EO) were also proposed and integrated within the HPSOT program aimed at improving the performance of the proposed PSO variants. The statistical design and analysis indicated that five combination results of IO and EO for hybrid SPSO and MCPSO were significantly better than those obtained from the original PSO variants for all eleven problem instances. The average computational times taken by the proposed hybrid methods were also faster than the conventional SPSO and MCPSO for all cases.

Keywords Swarm intelligence · Metaheuristics · Economic timetabling · Hybrid ratio · Local search

Introduction

Universities across the world are accelerated to face another major financial crisis due to the unexpected loss of revenues caused by unprecedented pandemic. The higher education sector in Australia expects the revenue loss to be AU$3–4.6 billion in the academic year 2019–20. In the academic year 2020–21, a number of international student enrolments in US universities and colleges was expected to reduce by at least a quarter. Likewise, a 100% fall in fee income from international (Non-EU and EU) students would result in a £6.9 billion loss of income to the UK higher education sector [1]. Many universities have, therefore, tried to reduce the impact on their permanent workforce by revising their infrastructure programs, cut executives’ pay, resizing casual staff, frozen hiring and drawn on any available reserves, improving resources’ utilisation or minimising operational costs via course timetabling and so on.

Due to varying numbers of courses, curricula, students but limited numbers of resources (e.g., lecturers, classrooms), a course timetabling problem (CTP) repeatedly arises every semester in academic institutes. For the practical course timetables without any conflict or event clash, all hard constraints must be satisfied [2]. Whilst soft constraints (e.g., the quality of timetables, staff preferences and resources’ utilisation) should be optimised [2]. Solving the CTP can be accomplished using academic staffs with/without automatically course timetabling tool. Without any
automatic timetabling program, solving a large CTP is an extremely difficult task and requires a group of staff to work. Moreover, the computational time required for finding the optimal course timetables increases exponentially with the size of problem [3]. Because the CTP is classified to be a Non-deterministic Polynomial (NP)-hard problem [4]. Developing an automated timetabling tool, in which computational intelligence is embedded to generate the optimal timetables, is an alternative way to deal with these difficulties, especially for solving large sizes of CTP.

Computational Intelligence (CI) based on nature-inspired optimisation methods, e.g., Genetic Algorithm [5], Frog Leaping Algorithm [6], Artificial Immune System [7], Ant Colony Optimisation [4], Grey Wolf Optimisation [8], Bat Algorithm [9], Particle Swarm Optimisation (PSO) [10], has become very popular to solve the large sizes of combinatorial optimisation problems [11]. Although there is no guarantee to yield an optimal solution, these algorithms can practically solve the NP-hard problems within acceptable computational time [12]. Conventional PSO has been widely applied to deal with the NP-hard problems in many problem domains because of several advantages such as a few parameters to be identified, easy for learning and implementation, and little memory requirements to solve [13]. Therefore, in this present paper, we proposed to improve newer variants of PSO, such as Standard PSO (SPSO), Accelerated PSO, and Maurice Clerc PSO (MCPSO) by adopting hybridisation strategies.

In cases of very high complex constraints and very large problem size, hybridisation strategies can improve the performances of PSO by balancing exploitation and exploration searches [2]. Many types of Local Search (LS) strategies, e.g., Neighbourhood Search [14], Hill Climbing [15], Exchange Operator (EO) [2, 16], Local Beam Search [17], Insertion Operator (IO) [2], have been found to hybrid with PSO for dealing with the CTP. Besides avoiding traps in a local optimum solution, hybrid approaches can increase the opportunity to discover the global best solution quickly [18]. According to a literature survey above, only one article has considered more than one type of LS strategies (IO and EO) embedded in MCPSO for solving the CTP [2]. However, the study on hybridisation between SPSO and difference IO:EO ratios to construct the best practical course timetables having the minimal academic operating costs has not been reported.

The objectives of this work were to: (i) develop SPSO and embed it into the Hybrid Particle Swarm Optimisation-based Timetabling (HPSOT) program for solving the proposed CTP; (ii) and compare the performances of the SPSO and its hybridisations using five combinations of IO:EO ratios in terms of minimal total operating costs, computational time, and convergence speed. This paper is an extension of a conference paper [2] presented in the 13th International Conference on Blended Learning (ICBL 2020). The next section describes the CTP and its constraints for this research work. The third section describes the concept of PSO and its variants followed by the processes of the HPSOT program in the fourth section. The fifth section shows the experimental results and analysis before conclusions in the last section.

Course Timetabling Problem

Scheduling or timetabling is an important problem and an active area of research with applications in many fields [19], e.g., transportation [20], sport [21], healthcare (e.g., nurse rostering [22], surgery [23]), industrial production [8], employee [24], educational institutes (e.g., secondary school [25], university [18]). The university course timetabling problem (UCTP) is a classic and famous optimisation problem in the field of computer science and management science [26]. The UCTP is to find a method to allocate all course or events into the given timeslots and classrooms, whilst all pre-defined constraints must be satisfied [27].

Most of the UCTP constraints can be mainly classified into two groups including hard constraints and soft constraints [12]. The most important constraints are hard constraints, each of which must be satisfied to guarantee the candidate timetables to be the feasible timetables [28]. Soft constraints can be more relaxed, some violations of soft constraints can be accepted but it should be minimised [4].

The characteristics of all hard constraints (H) considered in this work can be explained as follows [18]: (H1) all given courses must be scheduled, in which all lectures or laboratories for each course must be assigned to different periods; (H2) lecturers and students can only attend one lecture or event at a time; (H3) each timeslot for the given classrooms cannot assign a lecture or event more than one; (H4) all courses required by lecturers and students must not be scheduled in their unavailable periods; (H5) each course must be scheduled in the specific classrooms according to its given requirements (e.g., types of classroom, room facilities, and building location); and (H6) all events or lectures for a course required consecutive timeslots must be satisfied. (H1–H3) are related to the event-clash constraints for course timetabling problems and they can be found in most of the educational institutes [12]. (H4–H6) are particular constraints for each lecturer, curriculum, and courses found in most of the universities in Thailand. In addition, Soft Constraints (S) can be described as follows [18]: (S1) all courses should be scheduled in the proper types of the classroom to keep away from unnecessary renting or operating costs (currency unit per period); (S2) each course taught by the given lecturer(s) should be assigned according to their preferred working days and periods for saving the hiring or employing costs (currency unit per period);
and \((S_t)\) each classroom should be used in consecutive periods of a day for reducing the number of times to prepare or clear that classroom after usability (per time).

The objective functions or \(f(x_i)\) for this work are calculated by considering \(S_1 - S_3\) for minimising the total operating costs obtained from the candidate timetables as shown in Eq. (1) [29]:

\[
f(x_i) = \psi_1 S_1 + \psi_2 S_2 + \psi_3 S_3\tag{1}
\]

Subject to : \(H_k = 0, \forall k\).

The aim of Eq. (1) is to evaluate the total operating costs related to \(S_1 - S_3\). The weightings \((\psi_1, \psi_3)\) for each soft constraint depend upon the user preferences for each educational institution. In this work, \(\psi_1, \psi_3\) were set at 50 (currency units/h), 300 (currency units/h), and 2.5 (currency units/times), respectively. Equation (2) is to check the feasible timetables, each of which \(H_1 - H_k\) cannot be violated. Where \(k\) is an index for the \(k^{th}\) hard constraint \((k = 1, 2, \ldots, K)\), where \(K\) is the given number of hard constraints.

\[
\begin{align*}
\text{PSO Variants and Literature Review} & \\
\text{Particle Swarm Optimisation (PSO) was firstly introduced by Kennedy and Eberhart in 1995 [30]. The inspiration of PSO was the swarm behaviour in nature, e.g., bird flocking, fish schooling [31]. PSO is also classified to be a stochastic optimisation method based on Swarm Intelligence (SI) [13] that performs searching via a group of particles being updated from iteration to iteration [32]. Moreover, PSO has been applied to solve many optimisation areas, computational intelligence, and design applications [31]. Because of its many advantages, such as simplicity and flexibility [31], fast convergence and few parameter settings [16], PSO has become one of the popular swarm intelligent methods [31]. According to a brief literature review, many PSO variants have been continuously extended and proposed, such as original PSO [16], Accelerated PSO (APSO) [31], Standard PSO (SPSO) [15], Maurice Clerc PSO (MCPSO) [14].}
\end{align*}
\]

The general concepts of the aforementioned variants of PSO can be briefly described as follows. The essential parameters of PSO are identified in the initialisation step. Each particle or solution \(x_i\) \((i = 1, 2, \ldots, P)\) is randomly created before computing its fitness value using the objective function or \(f(x_i)\). Where \(i\) is the index of particles, whereas \(P\) is the population size (number of particles). Next step, the iteration best solution \(x_i^t\) and the global best solution \(g^t\) are specified by considering Eq. (3) [32]. Where \(t\) is the current iteration, whilst \(i\) is the index of iterations:

\[
x_i^t = \min\{f(x_i^t)\}, \quad i \in (1, 2, 3, \ldots, P),
\]

\[
g^t = \min\{f(x_i^t)\}, \quad i \in (1, 2, 3, \ldots, P), \quad t \in (1, 2, 3, \ldots, I_{\text{now}}).
\]

For an original PSO, a new solution \(x_i^{t+1}\) for a particle \(i\) is achieved using velocity and position vectors according to Eqs. (4, 5) [33]. \(v_i\) is the velocity for particle \(i\), whereas \(c_1\) and \(c_2\) factors are the positive acceleration coefficients, \(r_1\) and \(r_2\) parameters are random variables uniformly distributed from 0 to 1 [32]:

\[
v_i^{t+1} = v_i^t + c_1 r_1 (x_i^t - x_i^t) + c_2 r_2 (g^* - x_i^t)
\]

\[
x_i^{t+1} = x_i^t + v_i^{t+1}.
\]

In the case of SPSO, a new solution \(x_i^{t+1}\) is produced using velocity and position vectors according to Eqs. (6) and (5) [32, 33]. Where \(\omega\) is the inertia weight factor that is used for balancing the global exploration and local exploitation [32]:

\[
v_i^{t+1} = \omega v_i^t + c_1 r_1 (x_i^{t+1} - x_i^t) + c_2 r_2 (g^* - x_i^t).
\]

New velocity and position for a solution \(x_i^{t+1}\) for MCPSO can be updated using Eqs. (7, 8), and (5) [33]. Where, \(k\) is a constriction factor to control particles velocity, \(\phi\) is a positive parameter depending on the given acceleration coefficients [33]:

\[
v_i^{t+1} = k v_i^t + c_1 r_1 (x_i^{t+1} - x_i^t) + c_2 r_2 (g^* - x_i^t)
\]

\[
k = \frac{2}{2 - \sqrt{\phi^2 - 4\phi}}, \quad \phi = c_1 + c_2, \quad \phi > 4.
\]

For APSO, a new solution \(x_i^{t+1}\) is produced using velocity and position vectors from Eq. (9, 10), and Eq. (5). Where \(\varepsilon_i\) is drawn from \(N(0, 1)\) [31]. Where \(c_0\) is an initial value of random parameter distributed from 0.5 to 1 [31]. Where, \(\gamma\) is a control parameter that should be set at 0.9–0.97 [31]. For easier implementation, updating velocity and position vectors for APSO can be switched to use only Eq. (11) [31]:

\[
v_i^{t+1} = v_i^t + c_1 (g^* - x_i^t) + c_2 \varepsilon_i
\]

\[
c_2 = c_0 \gamma^t, \quad (0 < \gamma < 1)
\]

\[
x_i^{t+1} = (1 - c_1) x_i^t + c_1 (g^* + c_2 \varepsilon_i).
\]
that obtained from the $f(g_{\text{best}})$, the $g_{\text{best}}$ will be replaced by a new solution $x^{i+1}_j$. These processes are repeated until the maximum iteration ($I$) or the given stop criterion is achieved.

Nowadays, using only conventional or standard PSO to solve most of the real-world constraint optimisation problems is a very difficult task [17]. Conventional PSO often suffers from low exploration ability and premature convergence, especially for very high complex multimodal problems [34]. The hybrid strategies have been widely accepted to increase the performance of algorithms for obtaining better solutions [35]. According to a literature survey on Scopus database using “course timetable*” and “particle swarm” as keywords for document search in data records (e.g., article title, abstract, and keywords) covering the period from the last two decades, several variants of PSO and their hybridisations for solving the UCTP are reported in Table 1.

According to Table 1, the real-world and benchmark course timetabling problems have been solved using many variants of PSO. Genetic Algorithm (GA), Interchange Operator, and Hill Climbing heuristic have also been applied to hybrid with PSO [16] and Standard PSO (SPSO) [15] to solve the real-world problems of course timetabling. Local Beam Search, Neighbourhood Search, and Constraint-based Reasoning have been hybridised with Maurice Clerc PSO (MCPSO) [14, 17, 36]. However, hybridising SPSO with different ratios between Insertion Operator (IO) and Exchange Operator (EO) has not been reported.

### Hybrid Particle Swarm Optimisation-Based Timetabling Tool

For this work, a HPSOT program was designed and developed using two programming languages including TCL/TK language for developing the graphical user interface and C language for generating the best timetable. The HPSOT tool was proposed to construct the practical timetables for lecturers, students, and classrooms having the lowest total university operating costs. Both Standard PSO (SPSO) and Maurice Clerc PSO (MCPSO) were integrated with the HPSOT tool with five different ratio settings of Exchange Operator (EO) and Insertion Operator (IO). The main procedure of the HPSOT tool can be divided into six sequential steps, as shown in Fig. 1.

According to Fig. 1, the first step is an initialisation process for the HPSOT tool. All input data, such as numbers of courses, lecturers, rooms, curricula, etc., are uploaded before setting the essential parameters for both SPSO and MCPSO. Then, the total number of events ($n$) is calculated from the total number of teaching periods required for all courses (see Fig. 2). An event list is then created to keep a set of $n$ events. To reduce the probability of getting infeasible timetables in this process, the sequence of events in the event list is sorted using a constructive heuristic named the Largest Unpermitted Period Degree (LUPD) [39].

Next process is to create a candidate solution or timetable $x_i$ as shown in Fig. 2 for an example. The dimension size of a candidate timetable is considered from the numbers of given classrooms, working days per week ($d$), and periods per day ($p$). Then, all events in the sorted list are assigned into an empty timetable for producing a candidate timetable $x_i$ ($i = 1, 2, 3, ..., P$), where $P$ is the number of given population or particles. From Fig. 2, it can be seen that all encoded events representing Physics, Calculus, and Sport subjects were successfully scheduled in a candidate solution. However, the “0” values located in the candidate solution illustrated in Fig. 2 represent the idle (empty) periods or timeslots.

After that, a random key technique [40] is applied for each $x_i$ solution to accommodate a solution evolution or movement process. A new list of a random key having the same dimension size of the $x_i$ timetable is created. Each slot of a random key list is assigned to have a random number between 0 and 1 using the Uniform Distribution. Both initial timetable construction and random key processes

### Table 1: Literature survey of PSO variants with/without hybridisation for solving the UCTP

| Authors                      | Years | Problems  | PSO variants | Hybridisations            | Hybrid Ratio |
|------------------------------|-------|-----------|--------------|---------------------------|--------------|
| Irene et al. [36]            | 2009  | Real world| MCPSO        | Constraint-based reasoning| No           |
| Irene et al. [14]            | 2009  | Benchmark | MCPSO        | Neighbourhood Search      | No           |
| Sheau Fen Ho et al. [37]     | 2009  | Benchmark | MCPSO        | No hybridisation          | n/a          |
| Ahandani and Vakil Baghmisheh [15] | 2013 | Benchmark | SPSO         | GA, Hill Climbing         | No           |
| Chen and Shih [16]           | 2013  | Real world| SPSO, SPSO   | Interchange operator      | No           |
| Kanoh and Chen [38]          | 2013  | Real world| SPSO         | No hybridisation          | n/a          |
| Oswald and Anand Deva Durai [17] | 2014 | Real world| MCPSO        | Local beam search         | No           |
| Thepphakorn and Pongcharoen [29] | 2019 | Real world| SPSO, SPSO, MCPSO | No hybridisation          | n/a          |
| Thepphakorn et al. [2]       | 2020  | Real world| MCPSO        | Local search (IO, EO)     | Yes          |
| This research                |       | Real world| SPSO         | Local search (IO, EO)     | Yes          |
for each particle \(x_i\) will be repeated until getting to the maximum number of particles \((P)\).

The second step is the movement (evolution) processes for both SPSO and MCPSO. In the case of MCPSO, updating velocity for each solution or particle \(x_i\) is produced using Eq. (7)-(8), whereas updating velocity according to Eq. (6) is used for SPSO. The position of particle \(x_i^{t+1}\) for both SPSO and MCPSO is then updated using Eq. (5).

For the university course scheduling case, an example of a movement process is shown in Fig. 3. If the first event of subject number 2 in the \(x_i^t\) solution is selected for moving, then the next step is to calculate a new value of velocity (or

---

**Fig. 1** Pseudocode of the HPSOT program

**Fig. 2** Initialisation of a candidate solution or timetable
random key) for $x_i^{(t)}$ based on the values obtained from
the first event of subject number 2 embedding in both $x_i^{best}$ and
the global best $g^{\ast}_{best}$ solutions. If a new value of velocity
is equal to 0.07, it will be replaced into the same position of
subject number 2 in the $x_i^{t}$ solution. After that, the move-
ment process of the remaining events in the $x_i^{t}$ solution is
also implemented similar to the aforementioned steps. Then,
a list of random keys is increasingly sorted before moving
all events (courses) of the $x_i^{t}$ solution to the new positions
according to the ascending order of new random keys.

After the movement process, a new solution or timeta-
ble $x_i^{t+1}$ may be either feasible or infeasible timetable. In
the case of an infeasible solution, the repair process was
adopted from previous research [18] to rectify all infeasible
timetables to obtain feasible timetables. The main rectifying
steps can be described as follows: (i) find the encoded events
having some violations of hard constraints ($H_1$–$H_6$); (ii)
for each violated event, seek the possible idle or empty timeslots
without $H_1$–$H_6$ violations (called the feasible timeslots) for
that violated event before swapping randomly; (iii) if the fea-
sible timeslots from the previous step cannot found, search
the possible non-empty timeslots (events) without any viola-
tion of $H_1$–$H_6$ for that violated event and vice versa before
random swapping; and (iv) repeat the steps (ii) and (iii) until
all violated events in a candidate solution are rectified.

The third step is to determine the quality of solution $x_i^{t+1}$
using Eq. (1). If $f(x_i^{t+1})$ is better than $f(x_i^{best})$, the current best
particle $x_i^{best}$ is replaced by the $x_i^{t+1}$ solution. Moreover, a
particle $g^{\ast}_{best}$ is replaced by the $x_i^{t+1}$ if the quality of $x_i^{t+1}$
solution is also the best-so-far timetable. These processes
are repeated until the quality of the solution obtained from
all particles is determined.

The fourth step is to produce the hybridisation for SPSO
and MCPSO using two strategies of Local Search (LS)
including an Insertion Operator (IO) and Exchange Operator
(EO), as shown in Figs. 4 and 5, respectively. Both operators
are demonstrated for their ability to improve the solution
quality according to Eqs. (12) and (13) [2]:

$$x_i^{t+1'} = x_i^{t+1} + IO$$

(12)

$$x_i^{t+1'} = x_i^{t+1} + EO.$$  

(13)

In this present work, five strategies of IO:EO ratios
including IO(100%), IO(75%):EO(25%), IO(50%):EO(50%),
IO(25%):EO(75%), and EO(100%), were proposed for
both SPSO and MCPSO hybridisations. For examples,
MCPSO + IO(100%) means that all particles or solution
$x_i^{t+1}$
($i = 1, 2, 3, \ldots, P$) must be produced using only IO strategy for
each iteration. Likewise, the MCPSO + IO(75%):EO(25%)
indicates that 75% of particles in the population were hybrid-
ised using IO, whereas the hybridisation of the remaining
particles (25%) were carried out using EO. After this pro-
cess, a new improvement solution was changed from
$x_i^{t+1}$ to $x_i^{t+1'}$.

In the fifth step, if some infeasible timetables were found
from the previous step ($x_i^{t+1}$), these timetables were then
rectified using the repair process again. The quality of solu-
tion $x_i^{t+1}$ was determined using Eq. (1) before updating

![Fig. 3 Example of movement processes using a random key technique for SPSO](image-url)
The processes from step 2 to step 5 were repeated until reaching the maximum iterations ($I$). For the final step, the best timetable for each lecturer, student, and room obtained from the proposed program was displayed together with the computational report.

**Computational Experiment**

The HPSOT tool for constructing the efficient course timetables with the minimum total operating costs in the university has previously been developed. Therefore, this experiment was designed to explore and compare the performances of hybrid SPSO and MCPSO with five combinations of IO:EO ratios, which were 0%:100%; 25%:75%; 50%:50%; 75%:25% and 100%:0%. In this work, the best settings for both SPSO and MCPSO parameters were adopted from previous work [29]. Eleven real-world course timetabling problem instances were also adopted from the previous research [41].

The timetable was based on 1 h per period. Due to the different sizes of the problem instances, the parameters were ranged from 173 to 1,009 events; 53 to 142 classrooms; 19 to 66 curricula; and 30 to 143 lecturers, which need to be scheduled in timetable ranged from 10 to 13 periods per day; 5 to 7 days per week. The number of computational run for each instance was repeated ten times using different random seeds. Each instance was tested on a personal computer with Core i7 3.20 GHz of CPU and 8 GB of Ram. The computational results for hybrid SPSO and MCPSO were analysed in terms of minimum (Min), maximum (Max), Mean (currency unit), standard deviation (SD), computational time (minute) and $P$-value obtained from the analysis of variants (ANOVA) as shown in Table 2.

According to Table 2, the hybrid performances of both SPSO and MCPSO with five combinations of IO:EO ratios to generate the preferable course timetables with the minimum total operating costs are better than the performances obtained from their conventional variants for all problem instances.
Table 2: Performance comparisons between the conventional SPSO and MCPSO as well as its hybridisations

| Inst | Analysis | SPSO | MCPSO | P value |
|------|----------|------|-------|---------|
| 1 Min | 203,018 | 202,669 | 0.144 | |
| Max | 203,158 | 203,136 | | |
| Mean | 203,099 | 203,031 | | |
| SD | 41.28 | 134.34 | | |
| Time | 4.67 | 4.69 | | |
| 2 Min | 382,309 | 382,239 | 0.370 | |
| Max | 383,588 | 383,410 | | |
| Mean | 382,900 | 382,744 | | |
| SD | 364.81 | 391.59 | | |
| Time | 15.68 | 19.94 | | |
| 3 Min | 306,227 | 304,349 | 0.000** | |
| Max | 306,969 | 306,026 | | |
| Mean | 306,722 | 305,460 | | |
| SD | 213.26 | 596.58 | | |
| Time | 27.43 | 30.23 | | |
| 4 Min | 309,488 | 307,856 | 0.000** | |
| Max | 310,573 | 309,820 | | |
| Mean | 310,215 | 308,821 | | |
| SD | 382,26 | 597.81 | | |
| Time | 18.71 | 20.47 | | |
| 5 Min | 492,272 | 492,429 | 0.596 | |
| Max | 493,584 | 493,737 | | |
| Mean | 492,892 | 493,002 | | |
| SD | 409,66 | 500.36 | | |
| Time | 30.09 | 36.48 | | |
| 6 Min | 410,641 | 409,362 | 0.001** | |
| Max | 410,481 | 410,345 | | |
| Mean | 410,996 | 410,329 | | |
| SD | 292.61 | 409.93 | | |
| Time | 44.79 | 41.47 | | |
| Inst | Analysis | SPSO | MCPSO | P value | Hybrid IO% + EO% ratios (Unit: money currency) | 0% + 100% | 25% + 75% | 50% + 50% | 75% + 25% | 100% + 0% |
|------|----------|------|-------|---------|---------------------------------|----------|-----------|----------|----------|----------|
|      |          |      |       |         | SPSO | MCPSO | SPSO | MCPSO | SPSO | MCPSO | SPSO | MCPSO | SPSO | MCPSO | SPSO | MCPSO |
| 7    | Min      | 419,594 | 418,732 | 0.005** | 420,399 | 413,717 | 420,017 | 414,015 | 419,317 | 414,227 | 418,836 | 415,049 | 418,483 | 417,363 |
|      | 100% + 0% | 421,157 | 420,788 | 420,363 | 420,594 | 420,356 | 420,440 | 419,257 | 419,792 | 418,471 | 419,535 | 417,272* | 419,533 | 418,967 |
|      | Max      | 420,805 | 419,216 | 420,381 | 419,792 | 418,471 | 419,535 | 417,272* | 419,533 | 418,967 |
|      | Mean     | 638.07  | 688.56  | 16.26 | 14.00 | 18.01 | 14.22 | 18.94 | 14.75 | 19.68 | 15.42 | 18.85 | 17.38 |
|      | SD       | 40.96   | 35.03   |        |        |        |        |        |        |        |        |        |        |
| 8    | Min      | 588,163 | 587,955 | 0.935  | 587,979 | 586,708 | 588,130 | 588,516 | 588,210 | 587,702 | 587,978 | 587,130 | 588,506 | 588,248 |
|      | 100% + 0% | 589,553 | 589,477 | 589,605 | 589,442 | 589,742 | 589,487 | 589,598 | 588,858 | 589,684 | 589,308 |
|      | Max      | 589,128 | 588,707 | 589,085 | 588,910 | 589,042 | 588,616 | 589,007 | 589,220 | 589,762 |
|      | Mean     | 486.07  | 623.7   | 499.39 | 878.90 | 515.34 | 327.79 | 586.63 | 528.00 | 528.32 | 620.12 | 393.05 | 381.66 |
|      | SD       | 74.13   | 86.97   | 37.18 | 30.98 | 39.59 | 36.44 | 39.79 | 37.77 | 43.83 | 40.60 | 46.94 | 42.71 |
| 9    | Min      | 614,461 | 616,349 | 0.411  | 615,543 | 608,437 | 615,221 | 612,932 | 613,511 | 611,974 | 614,460 | 611,149 | 615,073 | 613,422 |
|      | 100% + 0% | 617,858 | 617,638 | 616,754 | 616,545 | 617,371 | 616,097 | 616,855 | 616,664 | 616,291 |
|      | Max      | 616,870 | 617,237 | 616,881 | 615,962 | 615,510 | 616,071 | 614,466* | 615,729 | 614,656 | 616,065 | 615,294 |
|      | Mean     | 1,232.41 | 612.54 | 707.98 | 3,474.48 | 502.07 | 1,151.97 | 1,125.19 | 1,428.04 | 896.23 | 1,623.18 | 583.80 | 966.60 |
|      | SD       | 46.16   | 59.02   | 24.98 | 21.63 | 26.74 | 21.71 | 26.71 | 23.04 | 26.73 | 23.89 | 28.99 | 24.65 |
| 10   | Min      | 565,280 | 566,033 | 0.279  | 567,163 | 555,947 | 566,345 | 555,039 | 564,691 | 561,096 | 565,046 | 560,539 | 565,435 | 563,621 |
|      | 100% + 0% | 568,886 | 568,574 | 567,813 | 566,741 | 568,310 | 566,868 | 566,847 | 566,242 | 566,860 | 565,769 |
|      | Max      | 567,075 | 567,616 | 567,822 | 565,822 | 566,068 | 566,421 | 566,057 | 564,316 | 563,858 |
|      | Mean     | 1,155.39 | 1,051.13 | 504.42 | 3,631.00 | 504.42 | 4,090.85 | 968.22 | 1,991.10 | 506.26 | 1,571.29 | 427.61 | 718.15 |
|      | SD       | 72.94   | 80.95   | 39.88 | 31.38 | 40.76 | 32.56 | 45.10 | 33.86 | 44.70 | 35.21 | 46.00 | 36.82 |
| 11   | Min      | 956,907 | 956,529 | 0.932  | 956,847 | 943,597 | 952,822 | 952,592 | 954,706 | 950,951 | 953,321 | 950,469 | 953,975 | 952,467 |
|      | 100% + 0% | 960,907 | 961,157 | 959,917 | 958,927 | 957,559 | 957,897 | 957,590 | 958,116 | 956,378 | 955,668 | 955,467 |
|      | Max      | 958,784 | 958,843 | 958,601 | 956,589 | 956,559 | 956,085 | 956,409 | 955,665 | 955,593 | 953,674* | 956,120 | 953,943 |
|      | Mean     | 1,279.39 | 1,734.03 | 834.12 | 4,789.36 | 2,009.42 | 1,741.44 | 1,055.34 | 1,961.10 | 1,622.97 | 1,858.93 | 1,212.54 | 1,176.38 |
|      | SD       | 155.73  | 186.13  | 82.63 | 72.87 | 87.02 | 74.36 | 89.22 | 78.92 | 94.18 | 82.36 | 94.89 | 83.14 |

*The best average total operating costs for each problem instance

**The significance with a 95% confidence interval
instances. MCPSO + IO(75%):EO(25%) performed best as it produced timetables with the lowest mean operating costs for four out of eleven problem instances including instance numbers 6, 7, 8 and 11. Secondly, SPSO + IO(75%):EO(25%) was also the best performance for problem instance numbers 2 and 5. According to the mean values of the total operating costs associated with the generated timetables, both MCPSO + EO(100%) and MCPSO + IO(50%):EO(50%) were the best hybridisation ratio for two problem instances, whereas MCPSO + IO(25%):EO(75%) was the best only one problem instance. Moreover, the differences in the mean values achieved by SPSO and MCPSO methods were statistically significant with a 95% confidence interval using ANOVA (P value ≤ 0.05), especially in medium–large problem instances. The standard deviations obtained from all proposed methods were moderately different for all problem instances.

The average computational times required by hybrid SPSO and MCPSO with five combinations of IO:EO ratios were quicker than that obtained from their conventional variants for all instances. According to the largest instance in Table 2, the execution time required by both hybrid SPSO and MCPSO was reduced up to 60.30% and 60.85%, respectively. This was because the number of generated solutions in the current population is double after using IO and EO strategies. To limit the amount of search at the 24,000 candidate solutions, the number of iterations (I) for the proposed hybrid methods was, therefore, reduced by half.

The convergence speed comparisons to construct the best so far timetable for hybrid SPSO and MCPSO with five combinations of IO:EO ratios are shown in Fig. 6 and Fig. 7, respectively. The problem instance number 11, which was the largest size, was selected to be an example for this comparative study. Although SPSO + IO(75%):EO(25%) had a lower average of best so far solutions than other methods in the initial generations, the average value obtained from the middle to the last generations for this hybrid approach was the best, as shown in Fig. 6. Likewise, the aforementioned results using MCPSO + IO(75%):EO(25%) had a lower average of best so far solutions than other approaches during the middle generations. The average value obtained from the hybrid approach was also the best in the last generation, as shown in Fig. 7. Therefore, the proposed hybrid strategies with combinations of IO:EO ratios improved the performances of both SPSO and MCPSO in terms of the solution quality and convergence speed.

**Conclusions**

A Hybrid Particle Swarm Optimisation-based Timetabling (HPSOT) program was developed for solving the real-world university course timetabling problem in Thailand. Both hybrid SPSO and MCPSO with five combinations of IO:EO ratios were proposed and embedded into the HPSOT tool for constructing the desirable timetables with minimum total operating costs. The statistical analysis on the computational results suggested that the differences in average values achieved by conventional SPSO and MCPSO methods were statistically significant with a 95% confidence interval. The averages of the total operating costs associated with the timetables produced by hybrid SPSO and MCPSO with IO:EO ratios were better than those results obtained from their original variants for all problem instances. According to fair comparison in term of the amount of search, the computational times required by all proposed hybrid
approaches were faster than their original variants for all problem instances. Moreover, the convergence speeds generated from the hybrid SPSO and MCPSO using the IO:EO ratio of 75%:25% were the best configurations comparing with the other hybrid ratios.

**Funding** The work was financially supported by Thailand Science Research and Innovation and Office of the Higher Education Commission (MRG6080066) and Faculty of Engineering, Naresuan University (R2564E008).

**Declarations**

**Conflict of Interest** On behalf of all authors, the corresponding author states that there are no conflicts of interest.

**Ethical Approval** This article does not contain any studies with human participants performed by any of the authors.

**References**

1. Burki TK. COVID-19: consequences for higher education. Lancet Oncol. 2020;21:758.
2. Thepphakorn T, Sooncharoen S, Pongcharoen P. Academic operating costs optimisation using hybrid MCPSO based course timetabling tool. Lect Notes Comput Sci. 2020;12218:338–50.
3. Pongcharoen P, Promtet W, Yenradee P, Hicks C. Stochastic optimisation timetabling tool for university course scheduling. Int J Prod Econ. 2008;112:903–18.
4. Thepphakorn T, Pongcharoen P, Hicks C. An ant colony based timetabling tool. Int J Prod Econ. 2014;149:131–44.
5. Vitayasak S, Pongcharoen P, Hicks C. A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a Genetic Algorithm or modified Backtracking Search Algorithm. Int J Prod Econ. 2017;190:146–57.
6. Dapa K, Loreunthug P, Vitayasak S, Pongcharoen P. Bat algorithm, genetic algorithm and shuffled frog leaping algorithm for designing machine layout. In: Ramanna S, Lingras P, Sombatheera C, Krishna A, editors. Lecture Notes in computer science. Berlin: Springer; 2013. p. 59–68.
7. Pongcharoen P, Chainate W, Pongcharoen S. Improving artificial immune system performance: inductive bias and alternative mutations. In: Bentley PJ, Lee D, Jung S, editors. Lecture notes in computer science. Berlin: Springer; 2008. p. 220–31.
8. Sooncharoen S, Pongcharoen P, Hicks C. Grey wolf production scheduling for the capital goods industry. Appl Soft Comput. 2020;94:106480.
9. Chansombat S, Musikapun P, Pongcharoen P, Hicks C. A hybrid discrete bat algorithm with Krill Herd-based advanced planning and scheduling tool for the capital goods industry. Int J Prod Res. 2019;57:6705–26.
10. Dahmani I, Hifi M, Saadi T, Yousef L. A swarm optimization-based search algorithm for the quadratic knapsack problem with conflict graphs. Expert Syst Appl. 2020;148:113224.
11. Yang X-S. Swarm Intelligence Based Algorithms: A Critical Analysis. Evol Intel. 2014;7:17–28.
12. Lewis R. A survey of metaheuristic-based techniques for University Timetabling problems. OR Spectr. 2008;30:167–90.
13. Rana S, Jasola S, Kumar R. A review on particle swarm optimization algorithms and their applications to data clustering. Artif Intell Rev. 2011;35:211–22.
14. Irene SFH, Deris S, Mohd HSZ. A combination of PSO and local search in university course timetabling problem. In: Proceedings 2009 International Conference on Computer Engineering and Technology, ICCET 2009, 2009, pp. 492–5.
15. Ahandani MA, Vakil Baghmisheh MT. Hybridizing genetic algorithms and particle swarm optimization into a hyper-heuristic system for solving university course timetabling problem. WSEAS Trans Comput. 2013;12:128–43.
16. Chen RM, Shih HF. Solving university course timetabling problems using constriction particle swarm optimization with local search. Algorithms. 2013;6:227–44.
17. Oswald C, Anand DDC. Novel hybrid PSO algorithms with search optimization strategies for a University Course Timetabling
18. Thepphakorn T, Pongcharoen P. Performance improvement strategies on Cuckoo Search algorithms for solving the university course timetabling problem. Expert Syst Appl. 2020;161:113732.

19. Bettinelli A, Cauchiani V, Roberti R, Toth P. An overview of curriculum-based course timetabling. TOP. 2015;23:313–49.

20. Xu XM, Li KP, Yang LX, Gao ZY. An efficient train scheduling algorithm on a single-track railway system. J Sched. 2019;22:85–105.

21. Januario T, Urrutia S. A new neighborhood structure for round robin scheduling problems. Comput Oper Res. 2016;70:127–39.

22. Legrain A, Omer J, Rosat S. An online stochastic algorithm for a dynamic nurse scheduling problem. Eur J Oper Res. 2020;285:196–210.

23. Silva TAO, de Souza MC, Saldanha RR, Burke EK. Surgical scheduling with simultaneous employment of specialised human resources. Eur J Oper Res. 2015;245:719–30.

24. De Komarudin FT, Guerry M-A, Vanden BG. The extended roster quality staffing problem: addressing roster quality variation within a staffing planning period. J Sched. 2020;23:253–64.

25. Muggy L, Easton T. Generating class schedules within a complex modular environment with application to secondary schools. J Sched. 2015;18:369–76.

26. Rezaeipanah A, Matoori SS, Ahmadi G. A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Appl Intell. 2020;51:467–92.

27. Babaei H, Karimpour J, Hadidi A. A survey of approaches for university course timetabling problem. Comput Ind Eng. 2015;86:43–59.

28. Thepphakorn T, Pongcharoen P, Hicks C. Modifying regeneration mutation and hybridising clonal selection for evolutionary algorithms based timetabling tool. Math Probl Eng. 2015. https://doi.org/10.1155/2015/841748.

29. Thepphakorn T, Pongcharoen P, variants and parameters investigations of particle swarm optimisation for solving course timetabling problems. Lect Notes Comput Sci. 2019;11655:177–87.

30. Kennedy J, Eberhart R. Particle swarm optimization. In: IEEE International Conference on Neural Networks, 1995, pp. 1942–8.

31. Yang X-S. Nature-inspired optimization algorithms. Amsterdam: Elsevier; 2014.

32. Zhang Y, Wang S, Ji G. A comprehensive survey on particle swarm optimisation algorithm and its applications. Math Probl Eng. 2015. https://doi.org/10.1155/2015/931256.

33. Thangaraj R, Pant M, Abraham A, Bouvry P. Particle swarm optimization: Hybridization perspectives and experimental illustrations. Appl Math Comput. 2011;217:5208–26.

34. Vafashoar R, Meybodi MR. Multi swarm bare bones particle swarm optimization with distribution adaption. Appl Soft Comput. 2016;47:534–52.

35. Fong CW, Asmuni H, McCollum B. A hybrid swarm-based approach to university timetabling. IEEE Trans Evol Comput. 2015;19:870–84.

36. Irene HSF, Safaai D, Mohd H, Zaiton S. University course timetable planning using hybrid particle swarm optimization. In: Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, GEC’09, 2009, pp. 239–45.

37. Sheau FHI, Safaai D, Sitii ZMH. A study on PSO-based university course timetabling problem. Int Conf Asv Comput Control. 2009. https://doi.org/10.1109/ICACC.2009.112.

38. Kanoh H, Chen S. Particle swarm optimization with transition probability for timetabling problems. Berlin: Springer; 2013. p. 256–65.

39. Thepphakorn T, Pongcharoen P. Heuristic ordering for ant colony based timetabling tool. J Appl Oper Res. 2013;5:113–23.

40. Khadwilard A, Chansombat S, Thepphakorn T, Thapatsuwan P, Chainate W, Pongcharoen P. Application of firefly algorithm and its parameter setting for job shop scheduling. J Ind Technol. 2012;8:49–58.

41. Thepphakorn T, Pongcharoen P, Vitayasak S. A new multiple objective cuckoo search for university course timetabling problem. Lecture notes in computer science 10053 LNAI. Cham: Springer; 2016. p. 196–207.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.