Evolutionary Intelligence (2021) 14:1915–1930
https://doi.org/10.1007/s12065-020-00473-x

RESEARCH PAPER

Hybrid particle swarm optimization with particle elimination for the high school timetabling problem

Joo Siang Tan[1] · Say Leng Goh[1] · Suaini Sura[1] · Graham Kendall[2,3] · Nasser R. Sabar[4]

Received: 13 March 2020 / Revised: 2 July 2020 / Accepted: 6 August 2020 / Published online: 28 August 2020

Abstract
In this paper, a PSO-based algorithm that hybridized Particle Swarm Optimization (PSO) and Hill Climbing (HC) is applied to high school timetabling problem. This hybrid has two features, a novel solution transformation and particle elimination. The proposed methodologies are tested on the XHSTT-2014 dataset (which is relatively new for the school timetabling problem) plus other additional instances. The experimental results show that the proposed algorithm is effective in solving small and medium instances compared to standalone HC and better than the conventional PSO for most instances. In a comparison to the state of the art methods, it achieved the lowest mean of soft constraint violations for 7 instances and the lowest mean of hard constraint violations for 1 instance.

Keywords Particle swarm optimization · Hill climbing · Hybridisation · School timetabling

1 Introduction

Timetable construction is a Non-Polynomial (NP) complete decision problem [16]. It belongs to both NP and the NP-hard complexity classes. With the exponential increase of decision problem size, it is impossible for a deterministic algorithm to find an optimal solution under polynomial time, as most scientists believed $P \neq NP$ [2]. Generally, four common parameters are used in educational timetabling problem: times, resources, meetings and constraints. The objective is to assign times and resources to the meetings by minimising the constraints violations [22].

There are many types of timetabling problems, e.g. educational [11–14], sports [27], transportation [24], nurse rostering [4] etc. High school timetabling is a variant of the educational timetabling problem. High school timetabling problem involves assigning time and resources (teachers, students, classes and rooms) to a collection of events while respecting certain constraints. Some of the usual constraints are; two subjects cannot occur simultaneously in the same room, a teacher cannot be assigned to two subjects at the same time, no idle time for students and workload limit for teachers. In practice, it is important to have an efficient timetable to prevent teachers from being overworked, as it could lead to dissatisfaction among teachers and affect their performance in teaching. The teacher is a dominant factor affecting students' learning effectiveness in class, thus influencing their achievement [23].

In this work, we introduce a novel solution transformation in the mutation and crossover operations of Particle Swarm Optimization (PSO). Next, we propose a hybrid of PSO and Hill Climbing (HC). The hybridisation intends to allow a better exploration and exploitation in the search space in finding an optimum solution. We further improve the algorithm by adding a component called Particle Elimination.

✉ Say Leng Goh
slgoh@ums.edu.my

Joo Siang Tan
jsiang@live.com

Suaini Sura
su_sura@ums.edu.my

Graham Kendall
graham.kendall@nottingham.edu.my

Nasser R. Sabar
N.Sabar@latrobe.edu.au

[1] Optimization Research Group, Faculty of Computing and Informatics, Universiti Malaysia Sabah, Sabah, Malaysia

[2] The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia

[3] The University of Nottingham, University Park, Nottingham NG7 2RD, UK

[4] Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia

PSO was employed in high school timetabling problems [25, 26], however as far as we are aware, it has never been applied to the XHSTT-2014 dataset. This dataset is complex and challenging with many constraints thus not many researchers are working on it. This motivated us to use the dataset as a testbed to test the performance of our proposed algorithm.

This paper is organized as follows. Section 2 describes high school timetabling problems and XHSTT-2014 dataset. In Sect. 3, related work is reviewed. The proposed methodology is described in Sect. 4. The experimental result is shown in Sect. 5. Discussion is presented in Sect. 6. Finally, the conclusion and future work are given in Sect. 7.

2 High school timetabling problem

The school timetabling problem differs for each country in terms of constraints and objectives, which possibly due to different cultural settings and education systems [20].

2.1 Extensible mark-up language for high school timetabling-2014 (XHSTT-2014) dataset

Extensible Mark-up Language for High School Timetabling (XHSTT) is an XML based format used to define the instances in XHSTT-2014 and International Timetabling Competition 2011 (ITC2011) datasets [21]. The XML format is popular and widely used by operation research community because of the ease of use and flexibility in defining scheduling problems. XHSTT is composed of four entities namely times, resources, events and constraints [19].

- **Times**
  Times are the available time slots for events. Times can be grouped into days, weeks and time group. Normally, times are spread out for a week. For some instances, times may be spread out for more than a week, where odd and even weeks have different schedules.
- **Resource**
  Resources are rooms, teachers, students and classes for assignment. Each resource has a type and belongs to a certain group. For example, a computer lesson needs resources such as a computer room and a computer literate teacher.
- **Events**
  This entity represents events to be scheduled. Each event can be either a single lesson or a set of lessons (which have the same starting time). It has a course/event group, duration, preassigned time, workload and a set of required resources. Event group and course are used to group certain events. Event is like a capsule of resources and time, adhering to specific constraints.
- **Constraints**
  The constraints are a set of specific conditions that have to be satisfied. There is a total of 16 hard and soft constraints [21]. The hard constraints are;

1. *Assign Time Constraint*. Assign a time to an event.
2. *Assign Resource Constraint*. Assign a resource to an event.
3. *Split Events Constraint*. Split the selected event or event groups into sub-events, within a maximum and a minimum number of amount and duration.
4. *Avoid Clashes Constraint*. Schedule the selected resources or resource groups without clashes (no resource attends two events for the same time).
5. *Limit Workload Constraint*. Limit the workload of the selected resources or resource groups.

The constraints that can be hard or soft (depending on the specific instance) are;

1. *Prefer Times Constraint*. Assign a time from a given set of time to a selected event.
2. *Prefer Resources Constraint*. Assign a resource from a given set of resource to a selected event with specific role.
3. *Link Events Constraint*. Set all events from the selected event groups to a same starting time.
4. *Spread Events Constraint*. Schedule the selected event groups to a selected time group between a minimum and a maximum number of times.
5. Avoid Split Assignments Constraint. Schedule the selected event groups with specific role to the same resource.
6. *Distribute Split Events Constraint*. Split the selected event or event groups into sub-events with a specific duration, within a maximum and a minimum number of sub-events.
7. *Order Events Constraint*. Assign times in chronological order to selected events.
8. *Avoid Unavailable Times Constraint*. Avoid the selected resources or resource groups are busy in the selected times.
9. *Limit Idle Times Constraint*. Each selected resources or resource groups should have a limited number of idle times in a time groups.
10. *Limit Busy Times Constraint*. Each selected resources or resource groups should have a limited number of occupied times in a time groups.
11. *Cluster Busy Times Constraint*. Each selected resources or resource groups should have a limited number of time groups with an assigned time.

Table 1 Features of the instances in the XHSTT dataset

Instance	Times	Teachers	Rooms	Classes	Students	Events	Category	Dataset	Round in ITC2011
BrazilInstance1	25	8	–	3	–	21	S	Others	1
ItalyInstance1	36	13	–	3	–	42	S	Others	1
BR-SA-00	25	14	–	6	–	63	S	XHSTT-2014	2 & 3
BrazilInstance3	25	16	–	8	–	69	S	Others	2 & 3
BrazilInstance5	25	31	–	12	–	119	S	Others	1
BR-SM-00	25	23	–	12	–	127	S	XHSTT-2014	2 & 3
BR-SN-00	25	30	–	14	–	140	S	XHSTT-2014	2 & 3
FI-WP-06	35	18	13	10	–	172	S	XHSTT-2014	1
GR-P3-10	35	29	–	84	–	178	S	XHSTT-2014	1
ZA-LW-09	148	19	2	16	–	185	S	XHSTT-2014	1
BrazilInstance7	25	33	–	20	–	205	S	Others	1
WesternGreeceUniversity3	35	19	–	6	–	210	S	Others	2 & 3
ES-SS-08	35	66	4	21	–	225	S	XHSTT-2014	2 & 3
GR-PA-08	35	19	–	12	–	262	M	XHSTT-2014	2 & 3
ZA-WD-09	42	40	–	30	–	278	M	XHSTT-2014	2 & 3
FI-MP-06	35	25	25	14	–	280	M	XHSTT-2014	1
AU-SA-96	60	43	36	20	–	296	M	XHSTT-2014	1
AU-TE-99	30	37	26	13	–	308	M	XHSTT-2014	1
GR-H1-97	35	29	–	66	–	372	M	XHSTT-2014	–
FI-PB-98	40	46	34	31	–	387	M	XHSTT-2014	1
AU-BG-98	40	56	45	30	–	387	M	XHSTT-2014	1
FinlandElementarySchool	35	22	21	291	–	445	M	Others	2 & 3
FinlandSecondarySchool2	40	22	21	36	–	469	M	Others	2 & 3
US-WS-09	100	134	108	–	–	628	M	XHSTT-2014	–
IT-I4-96	36	61	–	38	–	748	M	XHSTT-2014	2 & 3
DK-VG-09	60	46	53	–	163	918	L	XHSTT-2014	–
DK-FG-12	50	90	69	–	279	1077	L	XHSTT-2014	–
NL-KP-09	38	93	53	48	-	1148	L	XHSTT-2014	2 & 3
NL-KP-03	38	75	41	18	453	1156	L	XHSTT-2014	1, 2 & 3
UK-SP-06	25	68	67	67	–	1227	L	XHSTT-2014	1
DK-HG-12	50	100	71	–	523	1235	L	XHSTT-2014	–
NL-KP-05	37	78	42	26	498	1235	L	XHSTT-2014	1, 2 & 3
KS-PR-11	62	101	–	63	–	1912	L	XHSTT-2014	2 & 3
GEPRO	44	132	80	44	846	2675	L	Others	1

There are 25 instances in the XHSTT-2014 dataset. We also work on 9 additional instances. Note that some instances in the XHSTT-2014 dataset can also be found in the ITC2011 dataset. There are 3 rounds in ITC2011. Participants in round 2 have to run their program within the time limit determined by running a benchmark program on their machine. Meanwhile, there are no time or technology restrictions for round 1 and 3. The features of the instances are presented in Table 1 (sorted by the number of events). We categorized the instances with less than 250 events as small (S), instances with the number of events ranging from 250 to 900 as medium (M) and instances with more than 900 events as large (L). The dash (–) symbol indicates that the feature is not relevant to the instance.

3 Related work

Vinay Kumar el at. [18] proposed an Interactive Self-Improvement based Adaptive PSO (ISI-APSO) in designing the Very-Large-Scale Integration (VLSI) circuit for the floor planning problem. An adaptive weight was updated in each iteration to adjust the exploitation and exploration in the multi-dimensional search space until a desired optimal solution is found. Their method achieved better performance than Genetic Algorithm (GA), Simulated Annealing (SA), traditional PSO, Ant Colony Optimization and Differential Evolution (DE) in terms of exploring efficiency and speed of convergence.

Feng el at. [6] proposed a multi-group particle swarm optimization algorithm (PSOEL) to solve the motion planning of redundant robotic manipulators. PSOEL consists of one elite group and several child groups. The particles of the elite group were made from the best performing particles of the child groups. Both groups evolved separately in each iteration. An interaction mechanism will be triggered when the elite group fell into a local optimum or was inferior to child groups. The bad particles from the elite group were replaced by the best particles from the child groups to allow the PSOEL to escape from the local optimum. In their simulation results, PSOEL was superior to the DE, GA and traditional PSO.

Tassopoulos and Beligiannis [26] proposed PSO algorithm in constructing Greek high school timetables. In the beginning, 150 active particles with different fitness value were initialized. Every particle in each generation was evolved with 3 procedures (swap with probability, change randomly with personal best and change randomly with global best) to produce a new solution. If the fitness value of a particle exceeded its tolerance value, the particle was inactivated in the evolution procedure until the number of the active particle was equal to 30. The purpose was to explore wider searching space and keep minimum robust particles. After 10,000 iterations, the best solution was further processed by a local search procedure to minimize the teachers' idle periods. This method outperformed GA, Constraint Programming and Evolutionary Algorithm (EA).

Tassopoulos and Beligiannis [25] improved the PSO by hybridizing it with a local search to solve high school timetabling problems in Greece [5]. 50 particles were initialized at random position in a two-dimensional matrix, and each particle went through 3 procedures (swap function, insertion from personal best and insertion from global best) to produce a feasible solution until the termination criteria were met. After PSO terminated, a local search procedure was applied to further improve the solution by minimising the soft constraint violations. Their implementation achieved better result compared to EA and SA.

3.1 Approaches specific to XHSTT instances

The winning methodology in ITC2011 was a hybrid metaheuristic that combined SA and Iterated Local Search (ILS) proposed by Fonseca et al. [10]. At first, an initial solution was generated by Kingston High School Timetabling Engine (KHE) [15] and improved by SA with reheating. Then ILS was applied to further improve the solution. There were six neighbourhood structures used by SA during the search: Event Swap (ES), Resource Swap (RS), Event Move (EM), Resource Move (RM), Event Block Swap (EBS) and Kempe Chain (KC). Meanwhile Reassign Resource Times (RRT) and KC were used in the perturbation phase of ILS. The method was successful because of its diversity of local search moves.

Post-competition, [9] continued to work on the ITC2011 dataset. They proposed a hybrid SA with stagnation-free Late Acceptance Hill Climbing (sf-LAHC). Late Acceptance Hill Climbing was adapted from the HC method with the acceptance criteria of accepting the worse solution. The proposed sf-LAHC can reheat the system during the stagnation condition, by retrieving the vector of costs from the last improvement occurred. Overall, the search was performed by SA after an initial solution was generated by the KHE solver. sf-LAHC was employed further to improve the solution. Their hybrid algorithm outperformed the winner method by producing the best-known solution for 14 out of 18 instances.

Kristiansen el at. [17] proposed an Integer Programming (IP) formulation to the XHSTT-2013 dataset. The method was based on mixed-integer linear IP (MIP) model. Their approach consisted of two steps. At first, the MIP model was built using hard constraints only. It minimised hard constraint violations until a time limit was reached or a feasible solution was found. If a feasible solution was found, the second step will be performed. The solution was further improved in the remaining time by adding all the soft constraints into the model. In their computational result, 2 optimal solutions were found and they also proved optimality for 4 other solutions out of 28 instances.

Fonseca et al. [7] further improved Kristiansen's IP model [17] by employing new cuts and reformulations. The fractional points were reduced by improving a few valid inequalities and an extended flow-based formulation from the original formulation. Also, the unnecessary variables and constraints were removed in the pre-processing routines. A multi-commodity flow reformulation was used to drop several non-essential constraints to reduce the size of the constraints in an alternative formulation. Their proposed method reduced 32% of the average gap from the IP model and improved 11 best-known solutions out of 12 instances. It is the current state-of-the-art for the problem instances in ITC2011.

4 Proposed methodology

We employ a one-stage approach for the following methodologies where the hard and soft constraint violations are minimized simultaneously at the same time instead of minimizing each constraint violations exclusively in multiple stages. We hope the search process can move freely in the search space without restriction due to the constraints.

4.1 Particle swarm optimization (PSO)

PSO is an evolutionary algorithm that is inspired by a real swarm in solving combinatorial problems [3]. Generally, each particle in PSO operates in two equations in every iteration [5]. In Eq. 1, the particle updates by adjusting the velocity or step size. In Eq. 2, the particle moves by adding the velocity to its previous position.

$$v_i(t+1) = v_i(t) + cr_1[x_i^*(t) - x_i(t)] + cr_2[x^g(t) - x_i(t)] \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \quad i = 1, \ldots, N_p \quad (2)$$

In the equations: t is the current iteration, i is the target particle's index, $x_i$is the particle's position, v_i is the velocity, x_i^* is the best position found by the particle, x^g is the global best position obtained by any particles, c is the acceleration constant (c > 0) while r_1 and r_2 are uniform random numbers within [0, 1].

The basic steps in PSO are:

1. Initialize a population of particles with position and velocity by randomly distributing them in the search space.
2. Evaluate fitness value of each particle and assign its current position $x_i(t)$ as local best position $x_i^*(t)$.
3. Find the global best position $x^g(t)$ among all particles.
4. Update the velocity of each particle using the first equation and moving it to new position according to the second equation.
5. Calculate new fitness value of each particle.
6. Update local best position $x_i^*(t)$ and global best position $x^g(t)$ considering its new fitness value.
7. Repeat steps 4–6 until the termination criteria is met.

Based on preliminary experiments, we set the number of particles to 25 as this number allows the PSO algorithm to perform effectively within the time limit. The PSO algorithm is presented in Algorithm 1. Mutation and crossover operations employed here are slightly similar to the ones found in [25, 26] where only vertical transformation is used (see Sects. 4.3.3.1 and 4.3.3.2).

Algorithm 1:

```
1  procedure PSO
2  f(globalBest) ← ∞
3  for i = 1 to 25 do
4      p.current ← random position
5      p.localBest← p.current                          ▷ set local best position
6      if f(p.localBest) < f(globalBest) then
7          globalBest ← p.localBest                    ▷ set global best position
8      end
9      activeParticle ← activeParticle ∪ p
10 end
11 while elapsedTime < timeLimit do
12     foreach p ∈ activeParticle do                   ▷ move the position of each particle
13         MUTATION(p)
14         LOCALCROSSOVER(p)
15         GLOBALCROSSOVER(p, globalBest)
16     end
17     foreach p ∈ activeParticle do
18         if f(p.current) < f(p.localBest) then
19             p.localBest ← p.current                 ▷ update local best position
20         end
21         if f(p.localBest) < f(globalBest) then
22             globalBest ← p.localBest                ▷ update global best position
23         end
24     end
25 end
```

4.2 Hill climbing (HC)

HC is the simplest local search heuristic that is widely applied in combinatorial problems [1]. At any point, HC only accepts improving moves from the neighbouring solution. It starts from a random initial solution and iteratively moves to a local optima.

The basic steps are:

1. Initialize a current solution and calculate its fitness value.
2. Random generate a neighbourhood structure and produce a candidate solution.
3. Calculate the fitness value of candidate solution and compare it with the fitness value of the current solution.
4. Replace the current solution with candidate solution if the fitness value has improved; otherwise, do nothing.
5. Repeat steps 2–4 until the termination criteria is met.

The pseudo code of HC is presented in Algorithm 2.

Algorithm 2:

```
1  procedure HC
2  x = initial solution
3  best = f(x)
4  while elapsedTime < timeLimit do
5  |   x' = random N(x)
6  |   current = f(x')
7  |   if current < best then
8  |   |   x = x'
9  |   |   best = current
10 |   end
11 end
```

The neighbourhood structures used are Swap operator (SO), Move operator (MO) and Kempe operator (KO). Each neighbourhood structure consists of neighbourhood functions as shown in Table 2. The neighbourhood functions (similar to the ones used by Fonseca el at. [8]) are listed below.

Table 2 Neighbourhood structures and functions

Neighbourhood structure	Neighbourhood function
SO	ES and RS
MO	EM and RM
KO	KM

- Event Swap (ES)—The times t_1 and t_2 from two randomly selected events are swapped.
- Resource Swap (RS)—Resources (of the same type) r_1 and r_2 assigned to two randomly selected events are swapped. e_2 are swapped.
- Event Move (EM)—A randomly selected event is moved to an empty time.
- Resource Move (RM)—The resource r_1 of a randomly selected event is replaced by another resource r_1.
- Kempe Move (KM)—The events from two randomly selected time t_1 and t_2 are represented as nodes in a graph. The nodes in conflict (sharing the same resources) in distinct times are connected by edges to produce a bipartite graph (known as conflict graph). The connected nodes in the distinct times are swapped.

A neighbourhood structure is selected using probabilistic selection with probability P (SO) = 0.3, P (MO) = 0.6 and P (KO) = 0.1. A relevant neighbourhood function is then selected with equal probability.

4.3 Hybrid PSO with particle elimination (HPSO-PE)

4.3.1 Initial solution

Our initial solution is built constructively. Our method comprises of structural phase, resource assignment phase and time assignment phase. In the structural phase, we split each event into sub-events randomly based on split event constraint and distributed split event constraint. If the constraints do not apply to the event, the event will be split into sub-events with a duration of either 1 or 2, to allow flexibility during the event assignment. In the resource assignment phase, resources are randomly assigned based on prefer resource constraint, avoid split constraint and limit workload constraint. If an event does not require any specific resource, a random resource is selected and assigned based on the resource type. Lastly, in the time assignment phase,

Fig. 1 Particle encoding

	$Time_1$	$Time_2$	…	…	$Time_{n-1}$	$Time_n$
Teacher $_1$	event	event	…	…	event	event
…	…	…	…	…	…	…
Teacher $_t$	event	event	…	…	event	event
Student $_1$	event	event	…	…	event	event
…	…	…	…	…	…	…
Student $_s$	event	event	…	…	event	event
Class $_1$	event	event	…	…	event	event
…	…	…	…	…	…	…
Class $_c$	event	event	…	…	event	event
Room $_1$	event	event	…	…	event	event
…	…	…	…	…	…	…
Room $_r$	event	event	…	…	event	event

we assign a time to each event randomly based on prefer time constraint and link event constraint. If an event does not require any specific time, any time can be assigned to the event based on event duration. Note that, these phases do not solve the constraints completely, however they do minimize the cost violations. The initial solution is then mapped into a two-dimensional matrix.

4.3.2 Particle encoding

Each particle is encoded based on two variables: resources (teachers, students, classes, rooms) and times in a two-dimensional matrix. We allow more than one event in each cell of the matrix to ease the placement of events. For example, a particle p [5][20] = {47, 33}, events 47 (x7B_English_1) and 33 (x7D_Maths_1) are assigned to resource 5 (Teacher05) and time 20 (Wed_1_4).

Figure 1 shows a particle in a two-dimensional matrix. A row represents a resource and a column represents a time. The matrix can contain up to four resource types, depending on the instance. Resources of similar type are grouped in rows. For example, there are 56 teachers and 300 students. The rows from 0 to 55 represent teachers while the rows from 56 to 355 represent students.

We use a single dimension to represent multiple resource types to save on memory utilization. We refrain from using one dimension to represent each resource type even though it may ease data manipulation as its application is memory intensive.

4.3.3 The main algorithm

The pseudo-code of the proposed HPSO-PE is presented in Algorithm 3. Note that a particle consists of a current position and a local best position. In lines 3–10, we append 25 particles to the set *activeParticle* (swarm). For each particle, we set the current position to a random position and the local best position to the current position. The global best position is set as the best local best position among the particles (lines 5–8).

In lines 12–16, we attempt to move the position of each particle utilizing mutation, local and global crossover operators. <horizontal and vertical> position transformation is applied in these operators. A new position is accepted only if its fitness value is better or equal to the current position. A detail explanation of the operators is given in Sects. 4.3.3.1 and 4.3.3.2.

In lines 17–19, we apply 1 iteration of HC (shown in Sect. 4.2) on 300 (set based on computational experience) particles selected randomly from the swarm. Then, we update the local best position of each particle and the global best position (lines 20–27).

When 20% of the time limit is exceeded, we start to eliminate the worst particle (in terms of fitness value) from the swarm until there are 5 (set based on computational experience) particles left (lines 28–37). We are trying to switch the focus of the search from exploration to exploitation towards the end.

Algorithm 3:

```
1  procedure HPSO-PE
2  f(globalBest) ← ∞
3  for i = 1 to 25 do
4      p.current ← random position
5      p.localBest← p.current                          ▷ set local best position
6      if f(p.localBest) < f(globalBest) then
7          globalBest ← p.localBest                    ▷ set global best position
8      end
9      activeParticle ← activeParticle ∪ p
10 end
11 while elapsedTime < timeLimit do
12     foreach p ∈ activeParticle do                   ▷ move the position of each particle
13         MUTATION(p)
14         LOCALCROSSOVER(p)
15         GLOBALCROSSOVER(p, globalBest)
16     end
17     for i = 1 to 300 do
18         apply HC on random p ∈ activeParticle
19     end
20     foreach p ∈ activeParticle do
21         if f(p.current) < f(p.localBest) then
22             p.localBest ← p.current                 ▷ update local best position
23         end
24         if f(p.localBest) < f(globalBest) then
25             globalBest ← p.localBest                ▷ update global best position
26         end
27     end
28     if elapsedTime > timeLimit * 0.2 AND activeParticle.size() > 5 then
29         worstFitness ← 0
30         foreach p ∈ activeParticle do
31             if f(p) > worstFitness then
32                 worstFitness ← f(p)
33                 worstParticle ← p
34             end
35         end
36         activeParticle ← activeParticle – worstParticle
37     end
38 end
```

4.3.3.1 Mutation operator The mutation operator (line 13 of the Algorithm 3) swap two rows (similar resource type) or columns in the solution matrix to produce a different solution. As shown in Algorithm 4, there is an equal probability for mutation by column and row. If mutation by column is chosen, two columns are randomly selected and swapped. An example is given in Fig. 2. Otherwise, if mutation by row is chosen, a resource type is randomly picked, two rows of the resource type are randomly selected and swapped. An example is given in Fig. 3.

Matrix A

	79		36	45
78	20	30		
2	94		52	
17				5
	45	97	17	
93	54			16
		35	74	83
62			71	
50	64	81		78
	11	10	77	

→

Matrix B

	45		36	79
78		30		20
2			52	94
17	5			
		97	17	45
93	16			54
	83	35	74	
62			71	
50	78	81		64
		10	77	11

Fig. 2 Example of mutation in the matrix by column (Matrix A mutated to produce Matrix B)

Matrix A

	79		36	45
78	20	30		
2	94		52	
17				5
	45	97	17	
93	54			16
		35	74	83
62			71	
50	64	81		78
	11	10	77	

→

Matrix B

	79		36	45
78	20	30		
93	54			16
17				5
	45	97	17	
2	94		52	
		35	74	83
62			71	
50	64	81		78
	11	10	77	

Fig. 3 Example of mutation in the matrix by row (Matrix A mutated to produce Matrix B)

Fig. 4 Example of crossover by column (Matrix X crossover with Matrix Y to produce Matrix Z)

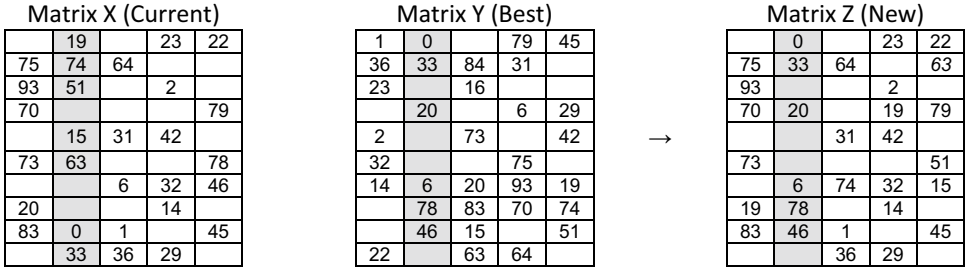
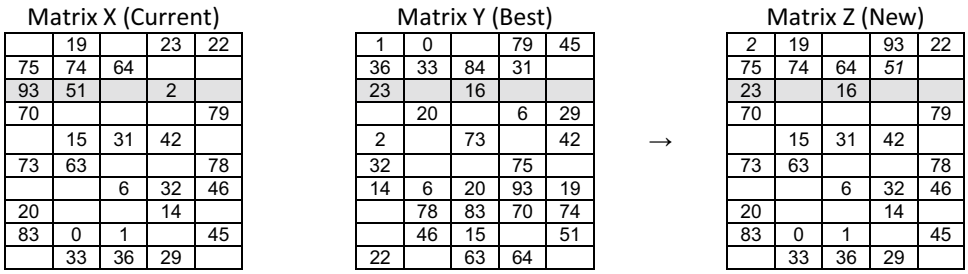


Fig. 5 Example of crossover by row (Matrix X crossover with Matrix Y to produce Matrix Z)



Algorithm 4:

```
1 procedure MUTATION(p)
2 if RANDOM[0,2) = 1 then
3 |   SWAPCOLUMN(p.current)          ▷ mutation by column
4 else
5 |   SWAPROW(p.current)             ▷ mutation by row
6 end
```

4.3.3.2 Crossover operator As the mutation operator, this operator aims to produce a potential new solution matrix with a better fitness value. There are two types of crossover operator namely local and global crossover.

The local crossover operator (line 14 of the Algorithm 3) replaces a column or row of events in the current solution matrix with the matching one in the local best solution matrix. As shown in Algorithm 5, there is an equal probability for local crossover by column and row.

If local crossover by column is chosen, in the replaceColumn procedure, a column (time) in the current solution matrix is randomly selected. All the events (except the ones that can be found in the matching column in the local best solution matrix) in that column are randomly distributed to other cells. That column is then replaced with the matching column in the local best solution matrix. An example is given in Fig. 4.

Otherwise, if local crossover by row is chosen, in the replaceRow procedure, a resource type is randomly selected. A row of that resource type is randomly picked. All the events (except the ones that can be found in the matching row in the local best solution matrix) in the row are distributed to other cells. That row (similar resource type) is then replaced by the matching row in the local best solution matrix. An example is given in Fig. 5.

The global crossover operator (line 15 of the Algorithm 3) is similar to the local crossover operator where it replaces a column or row of events in the current solution matrix with the matching one in the global best solution matrix instead of the local best solution. The details are shown in Algorithm 6.

Algorithm 5:

```
1 procedure LOCALCROSSOVER(p)
2 if RANDOM[0,2) = 1 then
3 |   REPLACECOLUMN(p.current, p.localBest)   ▷ local crossover by column
4 else
5 |   REPLACEROW(p.current, p.localBest)      ▷ local crossover by row
6 end
```

Table 3 Result comparison between <vertical> and <horizontal and vertical> solution transformations

Instance	Vertical		Horizontal and vertical		*t*-test (*p*-value)
	Best	Mean	Best	Mean	
BrazilInstance1	10	**10.9**	10	11.1	0.251
ItalyInstance1	16	**21.7**	15	22.8	0.060
BR-SA-00	9	**10.1**	9	**10.1**	0.500
BrazilInstance3	16	**18.9**	16	19.1	0.344
BrazilInstance5	27	29.8	26	**29.7**	0.362
BR-SM-00	37	**41.5**	38	**41.5**	0.471
BR-SN-00	37	**38.5**	36	38.7	0.262
FI-WP-06	23	26.3	24	**26.0**	0.233
GR-P3-10	151	**166.0**	156	168.7	0.050
ZA-LW-09	84	**89.2**	95	103.1	**0.000**
BrazilInstance7	53	56.8	51	**56.6**	0.363
WesternGreeceUniversity3	10	**10.8**	10	11.2	**0.003**
ES-SS-08	251	**281.3**	263	308.0	**0.000**
GR-PA-08	40	44.2	40	**44.1**	0.401
ZA-WD-09	110	123.3	112	**121.7**	0.075
FI-MP-06	38	41.5	38	**41.2**	0.202
AU-SA-96	2572	2675.2	2523	**2547.4**	**0.000**
AU-TE-99	2324	6469.4	1292	**1358.3**	**0.000**
GR-H1-97	7	9.8	7	**9.7**	0.471
FI-PB-98	115	121.7	112	**120.4**	**0.042**
AU-BG-98	22,455	31,806.1	14,491	**23,077.4**	**0.000**
FinlandElementarySchool	3	**3.6**	3	**3.6**	0.286
FinlandSecondarySchool2	64	67.5	61	**65.7**	**0.007**
US-WS-09	3142	**3234.8**	3169	3253.1	**0.012**
IT-I4-96	657	809.2	541	**704.5**	**0.000**
DK-VG-09	5276	**5535.9**	5504	5681.1	**0.000**
DK-FG-12	4905	**6235.8**	4932	6521.1	0.145
NL-KP-09	143,038	157,681.9	145,982	**156.338.1**	0.193
NL-KP-03	62,170	71,873.8	56,450	**64.657.9**	**0.000**
UK-SP-06	1822	2175.2	1615	**2074.2**	**0.023**
DK-HG-12	20,927	**23,859.0**	21,230	23,967.9	0.383
NL-KP-05	48,650	55,778.0	41,979	**50.015.9**	**0.000**
KS-PR-11	2073	2176.1	2010	**2141.0**	**0.010**
GEPRO	309,674	337,971.8	316,064	**334.885.3**	0.149

Shown is the best and mean value of (hard + soft) constraint violations

Algorithm 6:

```
1 procedure GLOBALCROSSOVER(p, globalBest)
2 if RANDOM[0,2) = 1 then
3 |   REPLACECOLUMN(p.current, globalBest)        ▷ global crossover by column
4 else
5 |   REPLACEROW(p.current, globalBest)           ▷ global crossover by row
6 end
```

Table 4 Result comparison between PSO and HPSO. Shown is the best and mean value of (hard + soft) constraint violations

Instance	PSO		HPSO		*t*-test (*p*-value)
	Best	Mean	Best	Mean	
BrazilInstance1	10	11.1	10	**11.0**	0.286
ItalyInstance1	15	22.8	16	**21.5**	**0.041**
BR-SA-00	9	10.1	9	**9.8**	**0.034**
BrazilInstance3	16	19.1	14	**18.1**	**0.035**
BrazilInstance5	26	29.7	24	**28.0**	**0.000**
BR-SM-00	38	41.5	37	**40.0**	**0.000**
BR-SN-00	36	38.7	35	**37.2**	**0.000**
FI-WP-06	24	26.0	20	**23.5**	**0.000**
GR-P3-10	156	168.7	140	**154.2**	**0.000**
ZA-LW-09	95	103.1	71	**77.0**	**0.000**
BrazilInstance7	51	56.6	50	**53.4**	**0.000**
WesternGreeceUniversityInstance3	10	11.2	9	**9.8**	**0.000**
ES-SS-08	263	308.0	227	**265.4**	**0.000**
GR-PA-08	40	44.1	36	**42.3**	**0.002**
ZA-WD-09	112	121.7	105	**114.3**	**0.000**
FI-MP-06	38	41.2	35	**38.1**	**0.000**
AU-SA-96	2523	2547.4	2470	**2502.9**	**0.000**
AU-TE-99	1292	1358.3	1263	**1310.5**	**0.000**
GR-H1-97	7	9.7	4	**8.9**	**0.041**
FI-PB-98	112	120.4	97	**106.6**	**0.000**
AU-BG-98	14,491	23,077.4	4597	**7766.8**	**0.000**
FinlandElementarySchool	3	3.6	3	**3.5**	0.323
FinlandSecondarySchool2	61	65.7	48	**52.3**	**0.000**
US-WS-09	3169	3253.1	3194	**3248.5**	0.282
IT-I4-96	541	**704.5**	619	819.8	**0.000**
DK-VG-09	5504	5681.1	5437	**5536.3**	**0.000**
DK-FG-12	4932	6521.1	4888	**6354.7**	0.234
NL-KP-09	145,982	**156.338.1**	137,028	156,397.7	0.488
NL-KP-03	56,450	64.657.9	46,437	**58,178.4**	**0.000**
UK-SP-06	1615	2074.2	1555	**1896.6**	**0.000**
DK-HG-12	21,230	**23,967.9**	22,666	25,209.0	**0.000**
NL-KP-05	41,979	50.015.9	39,050	**47,800.6**	**0.010**
KS-PR-11	2010	2141.0	1958	**2048.4**	**0.000**
GEPRO	316,064	**334.885.3**	320,391	338,135.5	0.123

5 Experimental results

Experiments were performed on machines running on Intel® Xeon® E-2124 3.30 GHz with 16 GB RAM and Windows Server 2019. The algorithms were coded in Java and compiled with Eclipse IDE. The results obtained were validated using HSEval validator[1] . All the experiment instances were supplied by University of Twente[2] and the time limit T (determined by running a benchmark program on the computer) allowed for every single run is 540 s. The statistical data were gathered by executing 31 runs for each instance.

5.1 The effect of solution transformation

In this section, we compare the use of <vertical> and <horizontal and vertical> transformations in the mutation and crossover operations of PSO. Table 3 shows the best and mean values of the constraint (hard + soft) violations. From *t*-tests, the *p*-values (less than 0.05) revealed a significant difference between the means of PSO utilizing <vertical> and <horizontal and vertical> transformation on 15 instances. PSO utilizing our proposed solution

[1] http://www.it.usyd.edu.au/ jeff/cgi-bin/hseval.cgi.
[2] https://www.utwente.nl/en/eemcs/dmmp/hstt/.

Table 5 Result comparison between HPSO and HPSO-PE

Instance	HPSO		HPSO-PE		*t*-test (*p*-value)
	Best	Mean	Best	Mean	
BrazilInstance1	10	11.0	10	**10.9**	0.435
ItalyInstance1	16	21.5	15	**20.7**	0.134
BR-SA-00	9	9.8	8	**9.6**	0.163
BrazilInstance3	14	**18.1**	16	18.6	0.118
BrazilInstance5	24	28.0	24	**27.5**	0.076
BR-SM-00	37	40.0	35	**38.4**	**0.000**
BR-SN-00	35	37.2	35	**36.1**	**0.000**
FI-WP-06	20	23.5	14	**20.9**	**0.000**
GR-P3-10	140	154.2	102	**121.2**	**0.000**
ZA-LW-09	71	77.0	37	**49.2**	**0.000**
BrazilInstance7	50	53.4	47	**49.8**	**0.000**
WesternGreeceUniversity3	9	9.8	9	**9.2**	**0.000**
ES-SS-08	227	265.4	175	**215.0**	**0.000**
GR-PA-08	36	42.3	31	**38.3**	**0.000**
ZA-WD-09	105	114.3	72	**82.1**	**0.000**
FI-MP-06	35	38.1	28	**31.3**	**0.000**
AU-SA-96	2470	2502.9	2388	**2425.5**	**0.000**
AU-TE-99	1263	1310.5	1266	**1308.2**	0.340
GR-H1-97	4	**8.9**	3	9.2	0.325
FI-PB-98	97	106.6	61	**69.8**	**0.000**
AU-BG-98	4597	7766.8	2369	**3152.8**	**0.000**
FinlandElementarySchool	3	**3.5**	3	**3.5**	0.411
FinlandSecondarySchool2	48	52.3	16	**21.2**	**0.000**
US-WS-09	3194	3248.5	3164	**3243.1**	0.233
IT-I4-96	619	819.8	171	**198.2**	**0.000**
DK-VG-09	5437	5536.3	3475	**3675.3**	**0.000**
DK-FG-12	4888	6354.7	2846	**4695.3**	**0.000**
NL-KP-09	137,028	156,397.7	76,393	**98,053.2**	**0.000**
NL-KP-03	46,437	58,178.4	14,280	**17,354.1**	**0.000**
UK-SP-06	1555	1896.6	1134	**1604.5**	**0.000**
DK-HG-12	22,666	25,209.0	15,622	**18,689.8**	**0.000**
NL-KP-05	39,050	47,800.6	17,529	**22,046.1**	**0.000**
KS-PR-11	1958	2048.4	1086	**1152.4**	**0.000**
GEPRO	320,391	338,135.5	310,275	**334,886.1**	0.077

Shown is the best and mean value of (hard + soft) constraint violations

transformation <horizontal and vertical> managed to achieve a lower mean value for 10 out of the 15 instances. The proposed solution transformation is effective for most of the medium and large instances.

5.2 Comparing PSO and HPSO

We compare the performance of PSO and hybrid PSO (HPSO) in minimising constraint violations. To show the effectiveness of hybridization, <horizontal and vertical> transformation is used in both algorithms. As shown in Table 4, the *p*-values (less than 0.05) revealed a significant difference between the means of HPSO and PSO in 28 instances. No significant difference is observed for 6 other instances (BrazilInstance1, FinlandElementarySchool, US-WS-09, DK-FG-12, NL-KP-09 and GEPRO). A lower mean value is obtained by HSPO for 26 out of the 28 instances, showing that synergy is achieved by the hybridization.

5.3 The effect of particle elimination

In this section, we present the effect of eliminating insignificant particles in HPSO-PE in Table 5. The *p*-values (less than 0.05) revealed a significant difference between the means of HPSO and HPSO with particle elimination

Table 6 Result comparison between HC, PSO and HPSO-PE

Instance	HC		PSO		HPSO-PE	
	Best	Mean	Best	Mean	Best	Mean
BrazilInstance1	12	21.9	**10**	**10.9**	**10**	**10.9**
ItalyInstance1	28	46.0	16	21.7	**15**	20.7
BR-SA-00	14	19.0	9	10.1	**8**	**9.6**
BrazilInstance3	25	40.7	**16**	18.9	**16**	**18.6**
BrazilInstance5	34	39.3	27	29.8	**24**	**27.5**
BR-SM-00	46	57.2	37	41.5	**35**	**38.4**
BR-SN-00	42	53.9	37	38.5	**35**	**36.1**
FI-WP-06	30	39.4	23	26.3	**14**	**20.9**
GR-P3-10	124	147.7	151	166.0	**102**	**121.2**
ZA-LW-09	47	59.8	84	89.2	**37**	**49.2**
BrazilInstance7	58	66.2	53	56.8	**47**	**49.8**
WesternGreeceUniversity3	10	11.7	10	10.8	**9**	**9.2**
ES-SS-08	216	280.1	251	281.3	**175**	**215.0**
GR-PA-08	46	59.1	40	44.2	**31**	**38.3**
ZA-WD-09	101	130.1	110	123.3	**72**	**82.1**
FI-MP-06	32	42.7	38	41.5	**28**	**31.3**
AU-SA-96	2425	2497.4	2572	2675.2	**2388**	**2425.5**
AU-TE-99	1317	1412.9	2324	6469.4	**1266**	**1308.2**
GR-H1-97	14	24.8	7	9.8	**3**	**9.2**
FI-PB-98	63	84.7	115	121.7	**61**	**69.8**
AU-BG-98	2382	**2555.8**	22,455	31,806.1	**2369**	3152.8
FinlandElementarySchool	5	8.6	**3**	3.6	**3**	**3.5**
FinlandSecondarySchool2	**11**	**18.0**	64	67.5	16	21.1
US-WS-09	3243	3331.1	**3142**	**3234.8**	3164	3243.1
IT-I4-96	**102**	**144.2**	657	809.2	171	198.2
DK-VG-09	**2595**	**2713.3**	5276	5535.9	3475	3675.3
DK-FG-12	4973	6276.3	4905	6235.8	**2846**	**4695.3**
NL-KP-09	**54,335**	**68,105.5**	143,038	157,681.9	76,393	98,053.2
NL-KP-03	**2772**	**3741.3**	62,170	71,873.8	14,280	17,354.1
UK-SP-06	1446	2761.9	1822	2175.2	**1134**	**1604.5**
DK-HG-12	21,534	26,135.4	20,927	23,859.0	**15,622**	**18,689.8**
NL-KP-05	**2561**	**3217.5**	48,650	55,778.0	17,529	22,046.1
KS-PR-11	**613**	**673.4**	2073	2176.1	1086	1152.4
GEPRO	**142,949**	**169,605.7**	309,674	337,971.8	310,275	334,886.1
Average rank	2.25		2.33		**1.34**	

Shown is the best and mean value of (hard + soft) constraint violations

(HPSO-PE) for 24 out of the 34 instances. HPSO-PE attained a lower mean value for all the 24 instances, indicating the effectiveness of particle elimination in HPSO-PE.

5.4 Comparing HC, PSO and HPSO-PE

In this section, we compare the performance of HC, conventional PSO and HPSO-PE in Table 6. Table 6 shows the best and mean values of the constraint (hard + soft) violations. We also compare the performance of the three algorithms using a ranking system used in the second round of ITC2011. The three algorithms are ranked (1 or 2 or 3) based on the constraint (hard + soft) violations in each run on each instance. The average rank shown in the last row of Table 6 is calculated as the sum of ranks/ number of ranks. As we run each algorithm for 31 times for each of the 31 instances, therefore the number of ranks is 961. The average rank suggests that HPSO-PE is the best followed by HC and PSO. From observation, the HPSO-PE performed well on small and medium-sized instances while HC is effective on the larger instances.

Table 7 Result comparison between HC, PSO and HPSO-PE using extended runtime (5*T*)

Instance	HC			PSO			HPSO-PE		
	Best	Mean	Improvement (%)	Best	Mean	Improvement (%)	Best	Mean	Improvement (%)
DK-VG-09	**2231**	**2333.7**	14	3377	3466.9	37	2526	2591.8	29
DK-FG-12	2677	5211.1	17	2675	3142.0	50	**1785**	**2669.3**	43
NL-KP-09	**31,936**	**48,502.2**	29	75,031	85,380.9	46	40,116	49,329.5	50
NL-KP-03	**1610**	**2473.5**	34	11,026	12,282.0	83	3163	3711.3	79
UK-SP-06	1416	2897.6	− 5	1452	1931.4	11	**1218**	**1564.2**	3
DK-HG-12	17,576	23,763.8	9	13,255	13,486.3	43	**11,219**	**11,963.6**	36
NL-KP-05	**1481**	**1887.6**	41	7800	9415.9	83	2811	3035.8	86
KS-PR-11	**309**	**359.5**	47	948	973.7	55	437	470.5	59
GEPRO	**30,712**	**37,589.6**	78	235,784	259,832.5	23	178,709	224,355.9	33

Shown is the best and mean value of (hard + soft) constraint violations

Table 8 Solver details

Solver	Reference	Technique
A	Fonseca et al. [10]	Simulated annealing + iterative local search
B	Fonseca et al. [9]	Simulated annealing + stagnation-free late acceptance hill climbing
C	Kristiansen et al. [17]	Mixed-integer programming
D	Fonseca et al. [8]	Matheuristic + variable neighbourhood search

Table 9 Comparing HPSO-PE with the state of the art methods on instances in round 2 of ITC2011. Shown is (mean of hard constraint violations/ mean of soft constraint violations)

Instance	A	B	C	D	HPSO-PE
BR-SA-00	(1.0/63.9)	(**0.0**/78.0)	(**0**/46)	(**0.0**/5.8)	(9.6/**0.0**)
BrazilInstance3	(**0.0** /127.8)	(**0.0**/160.2)	(**0**/39)	(**0.0**/31.2)	(18.6/**0.0**)
BR-SM-00	(17.2/99.6)	(2.4/164.2)	(5/286)	(**0.0**/63.6)	(38.4/**0.0**)
BR-SN-00	(4.0/223.5)	(**0.0**/221.0)	(**0**/682)	(**0.0**/51.6)	(36.1/**0.0**)
WesternGreeceUniversity3	(**0.0** /5.6)	(**0.0**/5.2)	(**0**/25)	(**0.0**/5.6)	(7.1/**2.0**)
ES-SS-08	(**0.0** /865.2)	(**0.0**/856.8)	(1454/11,020)	(**0.0**/474.8)	(105.3/**109.7**)
GR-PA-08	(**0.0** /7.4)	(**0.0**/6.6)	(**0**/81)	(**0.0/5.0**)	(29.7/8.6)
ZA-WD-09	(2.0/15.8)	(2.0/**12.0**)	(1801/705)	(**0.0**/57.8)	(42.9/39.2)
FinlandElementarySchool	(**0.0** /4.0)	(**0.0**/3.8)	(**0/3**)	(**0.0/3.0**)	(**0.0**/3.5)
FinlandSecondarySchool2	(**0.0** /0.4)	(**0.0**/0.4)	(1604/3878)	(**0.0/0.0**)	(18.6/2.6)
IT-I4-96	(**0.0** /658.4)	(**0.0**/302.2)	(**0**/17,842)	(**0.0**/32.2)	(168.9/**29.2**)
NL-KP-09	(36.6/154,998.5)	(25.8/112,335.0)	(17,512/**140**)	(**11.8**/11545.0)	(92,068.6/5984.5)
NL-KP-03	(0.4/89,132.2)	(0.6/90,195.8)	(8491/6920)	(**0.0/1257.8**)	(3390.6/13,963.4)
NL-KP-05	(30.2/33,169.6)	(33.9/27,480.4)	(2567/**53**)	(**14.4**/5677.8)	(4267.6/17,778.4)
KS-PR-11	(14.0/6934.4)	(6.3/6383.8)	(3626/2620)	(**0.0/9.0**)	(996.4/156.0)

Out of curiosity, we extended the runtime to 5 times the time limit or 5*T* (2700 s) for the large instances. It took around 23 h to run each instance for 31 times. As shown in Table 7, the result is consistent with the previous one and HC still performed better than HPSO-PE on the large instances. However, it seems that the improvement rate for PSO and HSPO-PE is higher compared to HC for all the large instances except GEPRO.

5.5 Comparing HPSO-PE with the state of the art methods

Note that as we follow the time limit restriction as stipulated in round 2 of ITC2011, therefore we only compare HPSO-PE with the state of art methods (which are following the same restriction) on instances in round 2 of ITC2011. As shown in Table 9, HPSO-PE performs well in minimizing soft constraint violations but it is not that competitive in minimizing hard constraint violations. It achieved the lowest mean of soft constraint violations for 7 instances and the lowest mean of hard constraint violations for 1 instance. The solver details are given in Table 8.

6 Discussion

From observation, HC outperformed HPSO-PE in larger instances. HC is relatively lighter in terms of processing compared to HPSO-PE (a population based method). Therefore, it can perform more iterations than HPSO-PE. Furthermore, the time limit does not allow a sufficient number of iterations for HPSO-PE to perform effectively on larger instances. HC is also better than HPSO-PE in exploiting the search space by making small changes to the current solution and moving to its neighbour solutions.

Meanwhile, HPSO-PE performed better than HC in smaller instances. This is possibly because HPSO-PE now has a sufficient number of iterations to function effectively. Besides, HC can easily get stuck in local optima as its capability in exploring the search space is rather limited. On the other hand, HPSO-PE is better at exploring the search space. When one of the particles (solutions) is stuck in local optima, the other particles will attempt to guide it out of the local optima and move it towards the global best position.

The proposed HPSO-PE is superior compared to HC and PSO. The performance is attributed to the features in the algorithm. The proposed solution transformation <horizontal and vertical> is more effective than vertical transformation (E.g. swapping by columns only) in mutation and crossover operations. The connectivity of the search space is improved by having an additional method in generating a new solution. In HPSO-PE, the particles in the PSO component are used to explore the search space while the HC component is employed to exploit the search space. The combination of PSO and HC helps to reduce the probability of a solution getting stuck in local optima. Also, the number of particles in the swarm is reduced (by eliminating a certain number of insignificant particles) towards the end of HPSO-PE. Particle elimination works because it allows the search to switch from exploration to exploitation towards the end.

7 Conclusion and future work

We have presented the effect of various solution transformation used in the mutation and crossover operations of PSO. We found that the proposed solution transformation <horizontal and vertical> is better than vertical transformation. We have compared PSO and HPSO. Results indicate that synergy is achieved by hybridizing the PSO with HC. We have made a comparison of the performance of HPSO with and without particle elimination. Results show that HPSO-PE is superior to HPSO. We also compared the performance of HC, PSO and HPSO-PE. Average rank suggests that HPSO-PE is the best followed by HC and PSO. The HPSO-PE performed well on small and medium-sized instances while HC is effective on larger instances. Overall, the HPSO-PE performed better than HC and PSO. Lastly, we compared HPSO-PE with the state of the art methods.

In this work, we employed a one-stage approach where the hard and soft constraint violations are minimized simultaneously at the same time. The disadvantage of this approach is that the feasibility of a solution is not guaranteed. In future, we may develop a repair function to fix the feasibility of the solution at the end of the search. We may also try a two-stage approach where hard and soft constraints are minimized separately in stages.

Neighbourhood structures play an important role in the search for better quality solutions. Therefore, we look forward to developing a variety of novel neighbourhood structures to improve the connectivity of the search space.

The current HC component in HPSO-PE only accepts candidate solutions with better fitness value which strictly limits the exploration capability of the search. We are considering to relax the acceptance criteria to conditionally accept worse candidate solutions. We may also try to hybridize the PSO with other local search methods such simulated annealing, tabu search, iterated local search etc.

References

1. Al-Betar MA (2017) $\beta $-Hill climbing: an exploratory local search. Neural Comput Appl 28(s1):153–168. https://doi.org/10.1007/s00521-016-2328-2
2. Arora S, Barak B (2009) Computational complexity—modern approach. Cambridge University Press. http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264
3. Burke EK, Kendall G et al (2005) Search methodologies. Springer, Berlin
4. Ceschia S, Dang N, De Causmaecker P, Haspeslagh S, Schaerf A (2019) The second international nurse rostering competition. Ann Oper Res 274(1–2):171–186
5. Du KL, Swamy MNS (2016) Search and optimization by metaheuristics. Springer, Berlin. https://doi.org/10.1007/978-3-319-41192-7

6. Feng Z, Chen L, Chen CH, Liu M, Yuan M (2020) Motion planning for redundant robotic manipulators using a novel multi-group particle swarm optimization. Evol Intel. https://doi.org/10.1007/s12065-020-00382-z
7. Fonseca GH, Santos HG, Carrano EG, Stidsen TJ (2017) Integer programming techniques for educational timetabling. Eur J Oper Res 262(1):28–39. https://doi.org/10.1016/j.ejor.2017.03.020
8. Fonseca GHG, Santos HG, Carrano EG (2016) Integrating matheuristics and metaheuristics for timetabling. Comput Oper Res 74:108–117
9. Fonseca GHG, Santos HG, Carrano EG (2016) Late acceptance hill-climbing for high school timetabling. J Sched 19(4):453–465. https://doi.org/10.1007/s10951-015-0458-5
10. da Fonseca GHG, Santos HG, Toffolo TÂM, Brito SS, Souza MJF (2016) GOAL solver: a hybrid local search based solver for high school timetabling. Ann Oper Res 239(1):77–97. https://doi.org/10.1007/s10479-014-1685-4
11. Goh SL, Kendall G, Sabar NR (2017) Improved local search approaches to solve the post enrolment course timetabling problem. Eur J Oper Res 261(1):17–29. https://doi.org/10.1016/j.ejor.2017.01.040
12. Goh SL, Kendall G, Sabar NR (2018) Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem. J Oper Res Soc 70(6):873–888. https://doi.org/10.1080/01605682.2018.1468862
13. Goh SL, Kendall G, Sabar NR (2019) Monte carlo tree search in finding feasible solutions for course timetabling problem. Int J Adv Sci Eng Inf Technol 9(6):1936. https://doi.org/10.18517/ijaseit.9.6.10224
14. Goh SL, Kendall G, Sabar NR, Abdullah S (2020) An effective hybrid local search approach for the post enrolment course timetabling problem. OPSEARCH. https://doi.org/10.1007/s12597-020-00444-x
15. Kingston JH (2014) KHE14: An algorithm for high school timetabling. In: Proceedings of the tenth international conference on practice and theory of automated timetabling, 269–291. http://www.it.usyd.edu.au/~jeff/khe/khe14.pdf
16. Kingston JH (2014) Timetable construction: the algorithms and complexity perspective. Ann Oper Res 218(1):249–259
17. Kristiansen S, Sørensen M, Stidsen TR (2015) Integer programming for the generalized high school timetabling problem. J Sched 18(4):377–392. https://doi.org/10.1007/s10951-014-0405-x
18. Vinay Kumar SB, Rao PV, Singh MK (2019) Optimal floor planning in VLSI using improved adaptive particle swarm optimization. Evol Intel. https://doi.org/10.1007/s12065-019-00256-z
19. Post G, Ahmadi S, Daskalaki S, Kingston JH, Kyngas J, Nurmi C, Ranson D (2012) An XML format for benchmarks in high school timetabling. Ann Oper Res 194(1):385–397. https://doi.org/10.1007/s10479-010-0699-9
20. Post G, Kingston JH, Ahmadi S, Daskalaki S, Gogos C, Kyngas J, Nurmi C, Musliu N, Pillay N, Santos H, Schaerf A (2014) XHSTT: an XML archive for high school timetabling problems in different countries. Ann Oper Res 218(1):295–301
21. Post G, Di Gaspero L, Kingston JH, McCollum B, Schaerf A (2016) The third international timetabling competition. Ann Oper Res 239(1):69–75. https://doi.org/10.1007/s10479-013-1340-5
22. 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(1):55–89
23. Sanders WL, Wright SP, Horn SP (1997) Teacher and classroom context effects on student achievement: implications for teacher evaluation. J Pers Eval Educ 11(1):57–67
24. Schöbel A (2017) An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. Transp Res Part C Emerg Technol 74:348–365
25. Tassopoulos IX, Beligiannis GN (2012) A hybrid particle swarm optimization based algorithm for high school timetabling problems. Appl Soft Comput 12(11):3472–3489
26. Tassopoulos IX, Beligiannis GN (2012) Using particle swarm optimization to solve effectively the school timetabling problem. Soft Comput 16(7):1229–1252. https://doi.org/10.1007/s00500-012-0809-5
27. Yi X, Goossens D, Nobibon FT (2020) Proactive and reactive strategies for football league timetabling. Eur J Oper Res 282(2):772–785