Int. J. Production Economics 149 (2014) 131–144

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

An ant colony based timetabling tool

Thatchai Thepphakorn [a], Pupong Pongcharoen [a,*], Chris Hicks [b]

[a] Centre of Operations Research and Industrial Applications (CORIA), Industrial Engineering Department, Faculty of Engineering, Naresuan University, Pitsanulok 65000, Thailand
[b] Business School, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, UK

ARTICLE INFO

Article history:
Received 11 April 2012
Accepted 16 April 2013
Available online 4 May 2013

Keywords:
Best-Worst ant
Ant system
Ant colony
Local search
Course timetabling

ABSTRACT

The timetabling of lecturers, seminars, practical sessions and examinations is a core business process for academic institutions. A feasible timetable must satisfy hard constraints. An optimum timetable will additionally satisfy soft constraints, which are not absolutely essential. An Ant Colony based Timetabling (ANCOT) tool has been developed for solving timetabling problems. New variants of Ant Colony Optimisation (ACO) called the Best-Worst Ant System (BWAS) and the Best-Worst Ant Colony System (BWACS) were embedded in the ANCOT program. Local Search (LS) strategies were developed and embedded into BWAS and BWACS to enhance their efficiency and to help find the best timetable with the lowest number of soft constraint violations. Statistical tools for experimental design and analysis were adopted to investigate the factors affecting the BWAS performance. Eight benchmark problems were used for evaluating the performance. For large problems, the BWACS produced the best timetable and was better than the other ACO variants. The best proposed local search strategy enhanced the performance of both the BWAS and the BWACS by up to 74.5%, but this was at the expense of longer execution time.

1. Introduction

Effective timetabling is critical for educational institutions as it affects resource utilisation as well as staff and student satisfaction. Solving large course timetabling problems is extremely difficult and may require a group of people to work for several days (Burke and Petrovic, 2002; MirHassani, 2006). A common approach is to modify previous timetables to meet new requirements (Azimi, 2005; Daskalaki et al., 2004). However, this approach often does not work because the numbers of students, lecturers and student preferences are uncertain and vary from year to year (Azimi, 2005). In recent years, with better computing technology, automated tools based on mathematical models and algorithms are becoming increasingly effective at constructing timetables to the desired specification (Daskalaki et al., 2004; Lee and Chen, 2009).

Timetabling is a combinatorial optimisation (CO) problem. It is a non-deterministic polynomial (NP) hard problem (Daskalaki et al., 2004; Socha et al., 2003), which means that the amount of computation required increases exponentially with problem size. Enumerative search algorithms can guarantee optimal solutions (Blum, 2005), but those algorithms are often infeasible in practice because it takes too long to find an optimal solution (Blum and Roli, 2003; Dorigo et al., 2006). Approximation algorithms, such as metaheuristics, have been widely used for solving large-scale CO problems (Blum, 2005). These algorithms can produce near optimal solutions, in reduced computational time, but they do not guarantee optimum solutions (Blum and Roli, 2003; Lewis, 2008). Blum and Roli (2003) categorised metaheuristic search techniques as (i) single point, such as Tabu Search (TS) (Glover, 1989), Simulated Annealing (SA) (Kirkpatrick et al., 1983) and Iterated Local Search (ILS) (Lourenço et al., 2002) and (ii) population-based, including Genetic Algorithms (GA) (Goldberg, 1989), Particle Swarm Optimisation (PSO) (Kennedy and Eberhart, 2001), Artificial Bee Colony (ABC) optimisation (Pansuwan et al., 2010) and Ant Colony Optimisation (ACO) (Dorigo and Blum, 2005; Dorigo and Stützle, 2004).

In the last decade, ACO has been successfully used to solve various NP-hard problems such as machine layout problems (Leechai et al., 2009), bin packing problems (Thapatsuwan et al., 2008), and scheduling problems (Chainual et al., 2007; Neto and Filho, 2011). Other ACO variants called the Best-Worst Ant System (BWAS) and Best-Worst Ant Colony System (BWACS) have produced high quality solutions for the travelling salesman (Cordón et al., 2000) and quadratic assignment problems (Cordón et al., 2002a). The use of the ACO method for course timetabling has been reported in the literature. For example, the Max-Min Ant System (MMAS) has been used to solve course timetabling problems. It produces good solutions, even for large problems (Eley, 2006; Socha et al., 2003) and it has been shown to perform better than the benchmarking GRASP approach (Dino Matijaš

* Corresponding author. Tel.: +66 55964201; fax: +66 55964003.
E-mail addresses: pupongp@nu.ac.th, pupongp@gmail.com (P. Pongcharoen).

0925-5273/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2013.04.026

et al., 2010). Another variant of ACO called the Elitist Ant System (EAS) has been reported to be superior to the Ant System (AS) (Jaradat and Ayob, 2010).

Research has been conducted that has aimed to improve the ACO processes. The Rank-based Ant System (AS-rank) performance has been improved by embedding new multiple ordering heuristics in the AS-rank initialisation process to improve the efficiency of constructing feasible timetables (Thepphakorn and Pongcharoen, 2012). An improved AS-rank method has been shown to perform better than the conventional AS-rank approach with a single ordering heuristic. Djamarus and Ku-Mahamud (2009) improved the AS performance by introducing four heuristic factors and negative pheromone updating strategies in the AS initialisation and pheromone updating processes. These guide an ant so that they avoid infeasible tours. AS with these strategies help to construct better timetables than that without those strategies.

Ant Colony System (ACS) have also been hybridised with other heuristics (e.g. GA, SA and TS) and applied to solve both course and examination timetabling problems. The hybrid ACS-TS outperforms the other hybridisations for examination timetabling problems (Azimi, 2005), whilst the hybrid ACS-SA outperforms other ACO variants for solving course timetabling problems (Ayob and Jaradat, 2009). Double compact pheromone matrices (Nothegger et al., 2012) have been introduced to exploit information in the AS solution construction process for solving post enrolment course timetabling problems. The AS has produced high quality solutions even without using SA, but better solutions could be obtained when including it. The Die-Hard Co-Operative Ant Behaviour approach proposed by Ejaz and Javed (2007) was been introduced to find feasible course timetables in the first phase before getting into the optimisation phase. Five out of eleven cases related to medium-large problem sizes which produced new globally best solutions within limited time using this approach. The Hypercube framework (Johnson et al., 2006) was implemented with the MMAS (called MTH-MMAS) for solving university course timetabling problems and achieved good results for small and medium problems.

There have been a number of research articles that have focused upon improving metaheuristics by adopting optimal parameter settings (Figlali et al., 2009; Naderi et al., 2010; Pongcharoen et al., 2007; Thapatsuwan et al., 2012) or hybridisation approaches (Azimi, 2005; Pongcharoen et al., 2008a; Shelokar et al., 2007). Due to the nature and complexity of the problem domains, some of these algorithms are problem specific. The performance of the algorithms usually depends on the parameter settings (Li et al., 2010; Pongcharoen et al., 2008b; Zandieh et al., 2009). There are several ways to select parameter settings: ad hoc selection (Aytug et al., 2003); adopting recommendations of the previous work; a best-guess approach (Montgomery, 2012) or systematically identifying optimum settings through designed experiments. Due to the problem specific nature of the algorithms there is no generic optimal parameter set that can be efficiently applied to every problem domain (Figlali et al., 2009). Thus, the settings recommended by previous studies will only be applicable in similar domains. Trial-and-error experiments can be used to identify good parameter settings, but this approach is based upon experience and intuition. It can be costly and time-consuming and sometimes impossible to verify that the best values have been identified (Chen et al., 2009). The one-factor-at-a-time experimental strategy has been adopted by some researchers, but this approach is inefficient and fails to consider any possible interaction between the factors (Figlali et al., 2009). When there is interaction between factors the effect of one factor will vary according to the levels of other factors. Montgomery (2012) suggested that the correct approach for dealing with several factors is to conduct a factorial experiment, in which factors are systematically varied together, instead of one at a time. Relatively few researchers have investigated optimal parameter settings for metaheuristics by using proper experimental designs.

The objectives of this paper are to: (i) describe the development of the BWAS and BWACS for solving university course timetabling problems; (ii) demonstrate the use of experimental design and analysis for investigating the appropriate BWAS parameter settings; (iii) verify the performance of the algorithms with appropriate parameter settings; (iv) compare the performance of BWAS and BWACS with various ACO variants (including AS, ACS, MMAS, EAS and AS-rank) in terms of average results and convergence speeds; (v) improve the performance of both the BWAS and BWACS methods by combining the approaches with new local search (LS) strategies; and (vi) compare the performance of the combined approaches with the original BWAS and BWACS algorithms in terms of the quality of the results obtained, solution convergence speed, and the computational time required.

The next section describes course timetabling problems. Section 3 briefly explains the concepts of the BWAS and the BWACS. Section 4 considers the application of those methods and proposes local search strategies which are embedded in the Ant Colony based Timetabling (ANCOT) tool. Section 5 presents the experimental design, analysis and results followed by conclusions.

2. Course timetabling problem

There are many types of general timetabling problem such as nurse rostering, sports timetabling, transportation timetabling, and educational timetabling (Burke et al., 2007). In educational institutions, timetabling courses and examinations is a crucial activity, which assigns appropriate timeslots for students, lecturers, and classrooms. The general constraints in course timetabling can be classified into two types: hard constraints (HC) and soft constraints (SC) (Burke et al., 2007; Lewis, 2008). Hard constraints are the most important and must be satisfied to have a feasible timetable (Burke and Newall, 2004). For example, it is necessary to avoid the double booking of lecturers, students or classrooms. Soft constraints are more relaxed as some violations are acceptable; however, algorithms should aim to minimise the number of violations. Eighteen soft constraints have been reported in literature (Pongcharoen et al., 2008b), but they do not apply in all institutions (e.g. compulsory lunch times). A commercial version of the timetabling tool may require some customisation to cover special constraints such as those related to cultural or religious issues that may apply to universities in other countries.

The second international timetabling competition (the third tracks) described hard and soft constraints (Di Gaspero et al., 2007). The hard constraints considered in this work were: (i) all lectures within a course must be scheduled and assigned to distinct periods (HC_1); (ii) only one lecture can take place in the same classroom during the same period (HC_2); (iii) lectures within different modules or taught by the same lecturers must be scheduled in other periods (HC_3); and (iv) if a teacher for a course is not available to give a lecture during a given period, it cannot be scheduled during that period (HC_4). The soft constraints considered in this research were: (i) for each module, the number of students attending the course must be less or equal to the number of seats for all the classrooms hosting the lectures (SC_1); (ii) the lectures for each module must be spread into a minimum number of days (SC_2); (iii) lectures belonging to a programme should be adjacent to each other (i.e., in consecutive periods) (SC_3); and (iv) each lecture for a module should take place in the same classroom (SC_4).

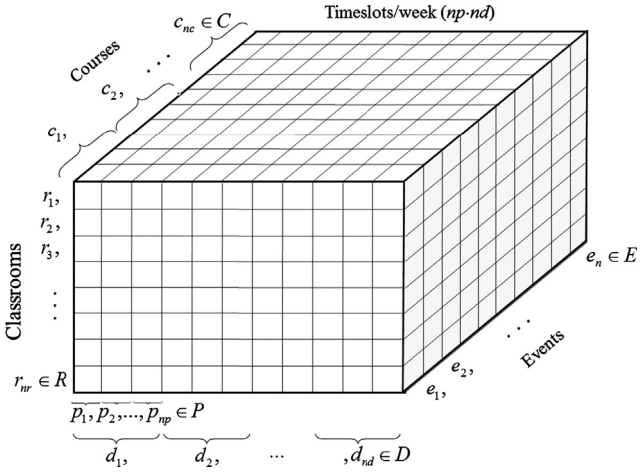

Fig. 1. Three-dimensional matrix for curriculum based course timetabling.

The soft and hard constraints considered in this study were mathematically formulated into a binary integer programming model, which was modified previous research (Burke et al., 2012; Lach and Lübbecke, 2012). In this binary decision based model, soft constraints (SC_1–SC_4) are represented by an objective function, whilst all four hard constraints determine whether potential solutions are feasible. The model used in this research can be illustrated as a three-dimensional matrix as shown in Fig. 1. The vertical axis represents the set of available classrooms (R); the horizontal axis illustrates the sets of days (D) and periods/day (P); and the third axis represents the set of courses (C). Each course has a different number of scheduled lectures (E_c), minimum required days (M_c), and enrolled students (S_c). The number of days (nd) for all classrooms is equally split into a number of periods/day (np). Each classroom may have a different seating capacity (A_r). Each curriculum consists of a set of courses (C_u). Each lecturer contributes to a set of courses (C_t). The set of total events (E) comprises the scheduling of all lectures required by all courses.

The following section describes the notation for this model including indices, parameters, and binary decision variables.

- Indices
  - r is the index of classrooms.
  - u is the index of curriculum.
  - c is the index of courses.
  - t is the index of teachers.
  - d is the index of days/week.
  - p is the index of periods/day.
  - e is the index of events.
- Binary decision variables
  - $x_{r,d,p,c}$ 1 if course c is scheduled on period p of day d in classroom r; otherwise 0.
  - $x_{c,d,p}$1 if course c belonging to curriculum u is scheduled on day d in period p; otherwise 0.
  - $x_{c,d}$ 1 if course c is scheduled on day d; otherwise 0.
  - $x_{c,r}$1 if course c is scheduled in classroom r; otherwise 0.
- Parameters
  - R, U, C, T, D, and P are the sets of classrooms, curricula, courses, teachers, days/week, and periods/day, respectively.
  - n, nc, nr, nd, and np are the numbers of events, courses, classrooms, days and periods/day, respectively.
  - C_t is the set of courses taught by teacher t.
  - C_u is the set of courses studied by curriculum u.
  - E is the set of events or lectures.
  - E_c is the number of events or lectures required by course c.
  - M_c is the minimum number of days/week required by course c.
  - S_c is the number of students enrolled in course c.
  - A_r is the number of available seats for room r.
  - $N_{c,d,p}$ is a binary matrix (1 if course c cannot teach on day d in period p; otherwise 0).
  - W_1, W_2, W_3, W_4 are weights of the corresponding soft constraint SC_1–SC_4, respectively.
  - SS, DC, and CP are the soft constraint violation numbers of spare seats, teaching days per course, and inconsecutive period, respectively.

$$SS = \begin{cases} S_c - A_r & \text{if } S_c > A_r \\ 0 & \text{otherwise.} \end{cases}$$

$$DC = \begin{cases} \left(M_c - \sum_{d \in D} x_{c,d}\right) & \text{if } M_c > \sum_{d \in D} x_{c,d} \\ 0 & \text{otherwise.} \end{cases}$$

$$CP = \begin{cases} (-x_{c,d,p-1} + x_{c,d,p} - x_{c,d,p+1}) & \text{if } (-x_{c,d,p-1} + x_{c,d,p} - x_{c,d,p+1}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

The objective was to construct timetables for students, lecturers, and classrooms that satisfied all of the hard constraints (to produce a feasible timetable) and that minimised the number of violations of the soft constraints (Kostuch, 2005). The total violation index (Z) for soft constraints can be used to determine the quality of feasible timetables by using the following equation:

$$Z = W_1\left[\sum_{r\in R}\sum_{d\in D}\sum_{p\in P}\sum_{c\in C} x_{r,d,p,c} SS\right] + W_2\left[\sum_{c\in C} DC\right] + W_3\left[\sum_{u\in U}\sum_{d\in D}\sum_{p\in P}\sum_{c\in C_u} CP\right] + W_4\left[\sum_{c\in C}\left(\left(\sum_{r\in R} x_{c,r}\right) - 1\right)\right] \quad (1)$$

Subject to:

$$\sum_{r\in R}\sum_{d\in D}\sum_{p\in P} x_{c,r,d,p} = E_c \quad \forall c\in C \quad (2)$$

$$\sum_{r\in R} x_{c,r,d,p} \le 1 \quad \forall c\in C,\ d\in D,\ p\in P \quad (3)$$

$$\sum_{c\in C} x_{c,r,d,p} \le 1 \quad \forall d\in D,\ p\in P,\ r\in R \quad (4)$$

$$\sum_{r\in R}\sum_{c\in C_t} x_{c,r,d,p} \le 1 \quad \forall d\in D,\ p\in P,\ t\in T \quad (5)$$

$$\sum_{r\in R}\sum_{c\in C_u} x_{c,r,d,p} \le 1 \quad \forall d\in D,\ p\in P,\ u\in U \quad (6)$$

$$\sum_{r\in R} x_{c,r,d,p}\cdot N_{c,d,p} = 0 \quad \forall c\in C, d\in D,\ p\in P \quad (7)$$

$$x_{c,r,d,p},\ x_{c,d,p},\ x_{c,d},\ x_{c,r} \in \{0, 1\} \quad \forall c\in C,\ r\in R,\ d\in D,\ p\in P \quad (8)$$

Eq. (1) is the objective function that evaluated the violation of four soft constraints considered in this work. The weightings (W_1–W_4) for each soft constraint are not restricted and depend upon the decision maker's preferences. Higher weightings indicate higher priorities. In this work, the weights (W_1–W_4) were adopted from the ITC2007 criteria (Di Gaspero et al., 2007) with values of 1, 5, 2 and 1 respectively. Eqs. (2) and (3) ensure that all the lectures for all of the courses were scheduled in different periods in order to avoid violating HC_1. Eq. (4) makes sure that only one lecture is

assigned to a classroom during each timeslot to satisfy HC_2. Eqs. (5) and (6) ensure that different courses undertaken by each student (or lecturer) must be scheduled in different periods to satisfy HC_3. Eq. (7) prevents courses from being assigned into unpermitted periods due to the HC_4. The binary decision variables are shown in Eq. (8).

3. Best-Worst Ant System (BWAS) and Best-Worst Ant Colony System (BWACS)

New variants of Ant Colony Optimisation (ACO), known as the Best–Worst Ant System (BWAS) and the Best-Worst Ant Colony System (BWACS) were first introduced by Cordón et al. (2000, 2002a). These methods have been successfully applied to solve travelling salesman problems (TSP) and quadratic assignment problems (QAP). The general concepts of the BWAS and BWACS are similar to other ACO variants, which were all inspired by the foraging behaviour of ants searching for the shortest path between a food source and their nest (Dorigo and Stützle, 2004).

3.1. Best-Worst Ant System (BWAS)

The BWAS mechanism is based on the Ant System (AS), which includes random proportional rules and pheromone evaporation (Cordón et al., 2002b). The random proportional rule is used to determine the probability of ant k moving from node i to j (p_{ij}^k) depending on the amount of the pheromone trail (τ_{ij}) and heuristic information (η_{ij}) from node i to j (Dorigo and Stützle, 2004). After a tour has been constructed, the pheromone evaporation process is implemented (Dorigo and Stützle, 2004). However, the obvious difference between the AS and the BWAS is that three components (including the best–worst pheromone update rule, the pheromone trails mutation, and the re-initialisation of the pheromone trails) are added into the AS in order to improve its performance (Cordón et al., 2002a; Cordón et al., 2002b).

The best–worst pheromone update rule is a crucial process for the BWAS Ant variant. The basic concept for updating the pheromone is by increasing pheromone on the arcs that belong to the best tour so far (T^{bs}) by using Eq. (9) and also decreasing pheromone on the arcs that belong to the worst tour (T^{w}) by using Eq. (10) (Cordón et al., 2002b)

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{bs} \quad \forall (i,j) \in T^{bs} \tag{9}$$

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} \quad \forall (i,j) \in T^{w} \text{ and } \notin T^{bs} \tag{10}$$

where $\Delta\tau_{ij}^{bs} = 1/(1+Z^{bs})$ is the amount of pheromone trail for tour T^{bs}, Z^{bs} is the total violation index associated with the best ant tour (T^{bs}). ρ is the pheromone evaporation rate that is uniformly distributed between 0 and 1. Afterwards, the pheromone value in the pheromone matrix is randomly mutated either by increasing or decreasing its value based on a binary random value (a). The mutation range for the pheromone value depends on the average of the pheromone trial that belongs to the T^{bs} given by the following equation (Cordón et al., 2002b);

$$\tau_{ij} \leftarrow \begin{cases} \tau_{ij} + mut(it, \tau_{threshold}), & \text{if } z \le P_m \text{ and } a = 0; \\ \tau_{ij} - mut(it, \tau_{threshold}), & \text{if } z \le P_m \text{ and } a = 1; \\ \tau_{ij}, & \text{if } z > P_m; \end{cases} \tag{11}$$

where z is a random variable and is uniformly distributed between 0 and 1, P_m is the mutation probability, which is used to control the mutation of the pheromone values in the pheromone matrix. The amount of mutated pheromone $mut(it,\tau_{threshold})$, which is based on the current iteration (it) and the average of the pheromone trial ($\tau_{threshold}$) on the T^{bs} arcs, can be calculated by the following equations (Cordón et al., 2000):

$$\tau_{\text{threshold}} = \frac{\sum_{(i,j) \in T^{bs}} \tau_{ij}}{na} \tag{12}$$

$$mut(it, \tau_{\text{threshold}}) = \left(\frac{it - it_r}{Nit - it_r}\right) \sigma \tau_{\text{threshold}} \tag{13}$$

Where na is the number of arcs, Nit is the maximum iteration, σ is the power of the mutation, and it_r is the last resetting iteration of the pheromone trails.

Finally, the re-initialisation of the BWAS is performed by resetting all the components of the pheromone matrix to the initial pheromone value (τ_0). It is reset if the percentage of total pheromone trails between the iteration best tour (T^{ib}) and T^{w} are less than or equal to the percentage for pheromone resetting (P_r) (Cordón et al., 2002a). The value of τ_0 proposed by Dorigo and Stützle (2004) was modified in order to solve timetabling problems. The value can be determined from the formula $\tau_0 = max\_ants/(1+Z^{nn})$ for BWAS. max_ants is the number of ants, and Z^{nn} is the total violation index of the ant tour that was constructed by using the nearest-neighbourhood rule.

3.2. Best-Worst Ant Colony System (BWACS)

The main BWACS processes were developed from the ACS, which are different from the AS in terms of the pseudorandom proportional rules, local pheromone update rules, and global pheromone update rules. The accumulated search experience (pseudorandom proportional rule) is used to determine the probability of ant k moving from node i to j (Dorigo and Stützle, 2004). Afterwards, a local pheromone update rule is used immediately after having crossed an arc (i, j) during the tour construction (Dorigo and Stützle, 2004). After the tour construction procedure, the pheromone evaporation process is implemented. Then, an ant deposits some pheromone on the arcs belonging to the T^{bs} called the global pheromone update rule (Dorigo and Stützle, 2004).

The BWACS was improved by adding three procedures to the ACS: (i) the best–worst pheromone update rule, which was used to improve the convergence of solutions by using Eqs. (9) and (10); (ii) the pheromone trail mutation procedure, which was used to reduce exploitation and increase exploration to find a new solution by using Eqs. (11)-(13); and (iii) re-initialisation - setting the pheromone value to τ_0 if no pheromone difference was found between T^{ib} and T^{w}. This was to avoid becoming trapped in a local sub-optimum solution. Although both the BWAS and BWACS have been successfully applied to solve combinatorial optimisations, its application to solve course timetabling problem have not been studied by other researchers or reported in the international literature databases.

4. Application of BWAS and BWACS with Local Search for timetabling problems

The applications of BWAS and BWACS to address timetabling challenges were developed in an automated timetabling program, which was coded using the TCL/TK programming language (Ousterhout, 2009). These algorithms, which were embedded in the ANCOT program, which consists of four main parts including initialisation (lines 1–3), solution construction (lines 4–11), local search (lines 12–15), and pheromone update (lines 16–21) as shown in Fig. 2.

```
1   upload problem data and assign parameters setting                    /* Initialisation procedure */
2   create events list (E); sort it by priority; create pheromone matrix
3   create candidate list and encode tour components {r_nr, d_nd, p_np}
4   while iteration ≤ maximum_iteration do                               /* Construct solution procedure */
5       for ant k = 1 to max_ants do
6           set empty tour (T^k) of ant k
7           for event = 1 to n do
8               check feasible timeslots in candidate list
9                 choose timeslot into T^k
10                if BWACS do update local pheromone
11          calculate Z of T^k
12      for ant k = 1 to max_ants do                                     /* Local Search procedure */
13          T^k produce LS strategies
14          calculate Z of T^k after LS
15          record the T^bs, T^ib and T^w
16  pheromone evaporation                                                /* Pheromone update procedure */
17  update pheromone trail of T^bs
18  evaporate pheromone trail of T^w
19      for event = 1 to n do
20          if random_value ≤ Pm do mutate pheromone trail
21      if different percent between T^ib and T^w ≤ Pr do reset pheromone matrix into τ_0
22  end while
23  output the best timetables ( T^bs)
```

Fig. 2. Pseudo code of BWAS and BWACS with Local Search strategies.

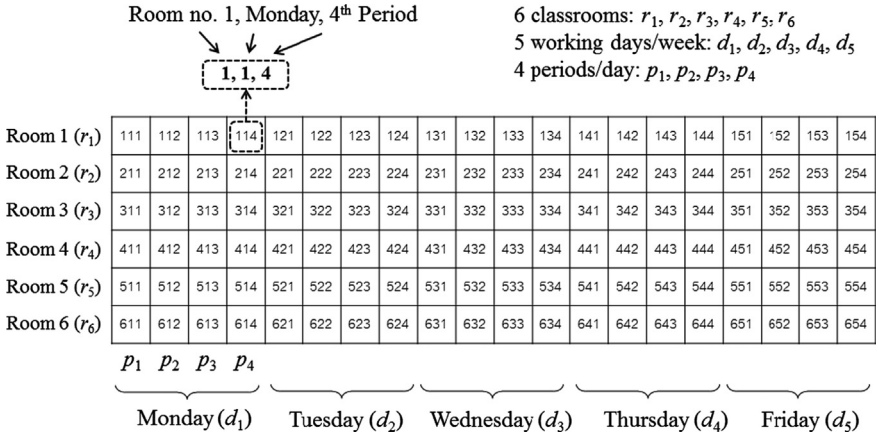

Fig. 3. Tour components encoding in candidate list of ants.

4.1. Initialisation procedure

After uploading the course timetable data and assigning the algorithm parameters, the total number of events (n) can be determined from the number of teaching periods required for all modules (courses). An event list (E) containing a set of n events (e_1, e_2,…, e_n) was initialised. The event sequence in the list was sorted by using the largest enrolment first heuristic (Burke et al., 2007). This rule reduces the probability of getting infeasible timetables that generally occur in the process of constructing solutions.

The next step is to create the ACO candidate list, which contains the components required for ant tour construction. The candidate list represents the total number of available timeslots (nr•nd•np), which were encoded into integer values as shown in Fig. 3. The timeslot components consisted of three coded numbers {r_{nr}, d_{nd}, p_{np}} which included a set of R classrooms (r_1, r_2,…, r_{nr}), a set of D days/week (d_1, d_2, …, d_{nd}), and a set of P possible periods/day (p_1, p_2, …, p_{np}). For example, if there are 6 classrooms, 5 working days/week, 4 periods/week, the available timeslots contained in ACO candidate list are shown in Fig. 3.

The next step in the initialisation process was to create the pheromone matrix for ACO, which was used for solution construction and by the pheromone update mechanism. The size of the pheromone matrix was equal to the total number of possible timeslots, from which ants could generate tours (timetables) in the search space. The pheromone matrix used in this work is similar to Fig. 1, which represents three dimensions including sets of events (E), classrooms (R), days/week (D) and periods/day (P), each of which also contained a small pheromone value ($\tau_0 > 0$).

4.2. Construct solution procedures

The construction of solutions by the BWAS and BWACS for timetabling was based on artificial ants gradually building an ant tour (timetable) by using state transition rules as shown in Fig. 4. At the first event, an ant (k) considers the feasible tour components (timeslots) from the candidate list. The characteristics of the feasible timeslots must not have been previously visited by the current ant and must not guide the ant into an infeasible tour, which would violate the hard constraints (HC_1–HC_4). Then, the state transition rule is used to randomly select the feasible timeslots in the partial tour (T^k) based on the pheromone trail and heuristic information. Once the timeslot is selected, the candidate list is updated while the local pheromone update is produced by BWACS methods. After that the ant (k) from the first event is moved to the next event in order to choose a new feasible timeslot from the candidate list.

The process is repeated until the ant (k) completes a full tour (T^k). Each complete tour is then determined for the quality by using Eq. (1) before the next ant (k+1) starts to construct a new tour. The number of tours depends on the number of ants (*max_ants*) previously defined. The efficiency of the tours can be enhanced by introducing Local Search strategies.

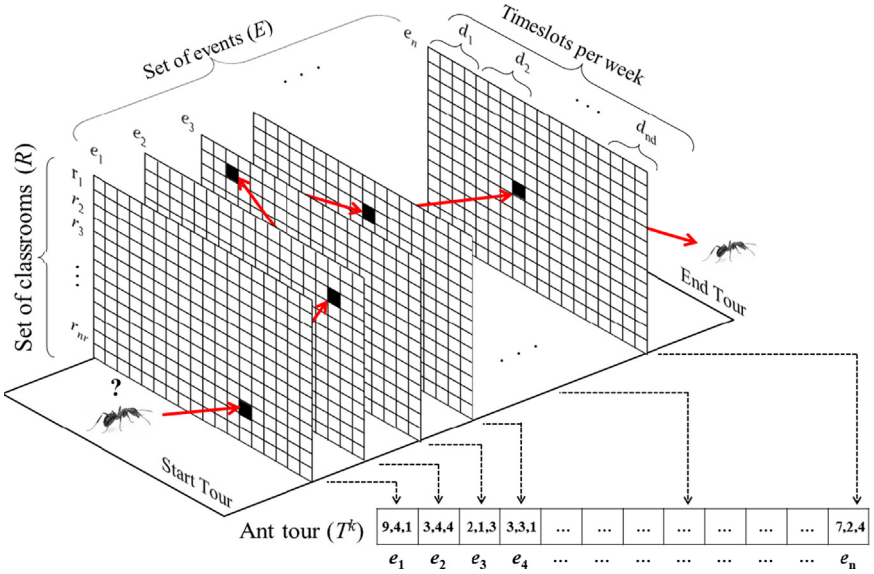

Fig. 4. The timetable construction of BWAS and BWACS.

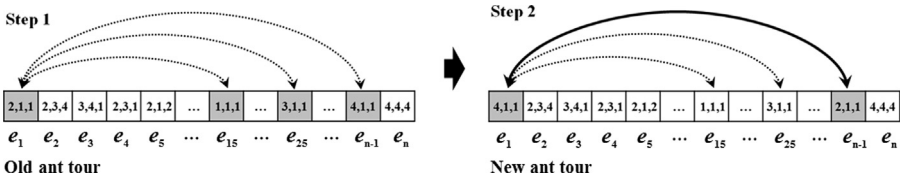

Fig. 5. The Local Search type I (LS1) procedure.

4.3. Local search procedures

Local Search (LS) procedures may be adopted to enhance the efficiency of metaheuristics. In this work, two new strategies based upon hill climbing local search were proposed and embedded into the conventional variants of the BWACS and BWAS. The aims were to: (i) improve the quality of the timetable (ant tour or T^k); (ii) exploit better solutions from its neighbour; and (iii) increase the probability of discovering the global optimum quickly. The new local search procedures can be used independently or in combination.

4.3.1. Local Search type I (LS1)

Local Search type I (LS1) is proposed for improving the efficiency of the ant tour (T^k) by reducing the number of violations of the first and the fourth soft constraints (SC_1 and SC_4). The LS1 process randomly interchanges two components (timeslots) within the ant tour. Only the classroom index (r) for randomised timeslots is allowed to swap position when the day (d) and period (p) indexes for those components are fixed in order to prevent the ant tour from violating hard constraints. For example (see Fig. 5), the first event (e_1) of the old ant tour was randomly selected and it was assigned the {2,1,1} timeslot ($r=2$, $d=1$, and $p=1$).

The first step of the LS1 was to search the components (timeslots) within the tour to find a timeslot with the same day (d) and period (p) as the first event, but with a different classroom (r) such as {1,1,1}, {3,1,1}, and {4,1,1} as shown in Fig. 5. Then, the timeslots found were collected into a group of feasible timeslots, and their local fitness was calculated taking into account the number of students and the number of classroom seats (SC_1) and keeping a course in the same classroom (SC_4). In Step 2, the feasible timeslot that had the best local fitness from the previous step was selected to switch timeslot positions within the tour. If the {4,1,1} timeslot had the best local fitness, the {2,1,1} timeslot for the first event was allowed to interchange with {4,1,1} the timeslot, while the remaining feasible timeslots were restored to their old positions. This process was repeated until all events in the tour were improved.

4.3.2. Local Search type II (LS2)

The objective of the Local Search 2 (LS2) was to reduce the number of violations of the second and the third soft constraints (SC_2 and SC_3) of the ant tour (T^k) by paying more attention to the working days of each course and the compactness of the timetable. This is illustrated in Fig. 6; the first-five events (e_1–e_5) of the ant tour (T^k) violated the soft constraint SC_2 or SC_3. The first step of LS2 was to restore the timeslots of those events (e_1–e_5) into the remaining candidate list of ant (k), whilst leaving the other timeslots for that tour unchanged. Then, the total timeslots in the current candidate list were checked for hard constraint violations (HC_1-HC_4). The local fitness (SC_2-SC_3) of feasible timeslots that did not violate the hard constraints was calculated. After that, the feasible timeslots were selected and restored back into the empty tour slots using the greedy rule (Odajima et al., 2008), based upon the local fitness (i.e. in terms of SC_2 and SC_3). This process was repeated until all the empty components of the ant tour (T^k) were completely scheduled.

4.3.3. LS1+LS2 and LS2+LS1

There are two combinations of local search strategies: LS1 followed by LS2 (LS1+LS2) or LS2 followed by LS1 (LS2+LS1). The quality of each T^k was then determined by using the objective function from Eq. (1). The global-best ant tour (T^{bs}), the iteration-best ant tour (T^{ib}), and the worst ant tour (T^w) in the current iteration were recorded before performing the next procedure.

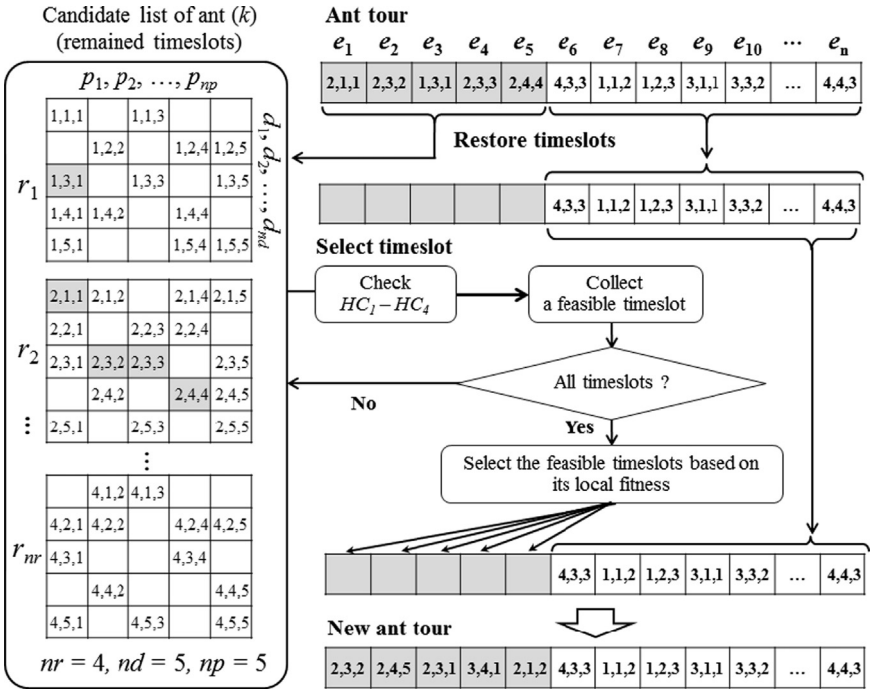

Fig. 6. The Local Search type II (LS2) procedure.

4.4. Pheromone update procedures

The BWAS and BWACS pheromone updating process for timetabling is directly related to the pheromone matrix. First, the pheromone values for all timeslots (small boxes) in the pheromone matrix are evaporated. The pheromone values for the timeslots on the best ant tours (T^{bs}) are then increased, whilst the values for the timeslots on worst ant tour (T^{w}) are decreased by both methods. The next step is for all the events in the pheromone matrix to be randomly selected for pheromone mutation; the selection is based upon the difference between a random value (z) and the mutation parameter (P_m). A timeslot for selected events is again randomised for mutation. The pheromone value obtained for random timeslots may be increased or decreased by mutation based on a binary random value (a). Finally, the percentage difference in average pheromone values between T^{ib} and T^{w} is calculated. If the percentage difference is less than the probability of pheromone resetting (a forbidden condition), the pheromone values for all timeslots in the pheromone matrix are reset to the initial value (τ_0). The construction solution procedures, local search procedures, and pheromone update procedures are repeated until the maximum iteration is satisfied. The ANCOT program then reports the best-so-far timetables (T^{bs}) for students, teachers and classrooms.

5. Experimental design and analysis

The computational experiments were designed in four steps to: i) verify the significance and identify the appropriate setting of the BWAS factors; ii) confirm the best BWAS parameter settings; iii) compare the performance of the BWAS and BWACS with the well-known ACO variants (including AS, ACS, MMAS, EAS and AS-rank); and iv) establish the performance of the BWAS and BWACS with/without Local Search (LS) strategies. Due to limited computational time and resources, two variants of ACO were considered with/without the proposed local search heuristics. Eight instant datasets of course timetabling problems provided by the third track of ITC2007 (Di Gaspero et al., 2007) were selected. The selected instant problems were different sizes and ranged from small to large (see more details in Table 1). All the computational runs were performed on personal computers with a Core 2 Quad 2.66 GHz CPU and 4 GB DDR3 RAM.

5.1. BWAS's screening experiment

The first experiment aimed to demonstrate the use of advanced statistical design and analysis to investigate the influence of factors within the BWAS. The factors and levels are summarised in Table 2. The factors considered were a combination of the number of ants multiplied by the number of iterations (AI), pheromone weight (α), heuristic information weight (β), pheromone evaporation rate (ρ), power of mutation (σ), probability of pheromone mutation (P_m), and probability of pheromone resetting (P_r). The amount of search, determined by the combination of the number of ants multiplied by the number of iterations (AI), should be increased when the problem size increases. The high value of this combination usually increases the probability of getting the best solution, but also requires longer computational time and resources. In practice, when a computational limitation is imposed, this combination can be fixed to suit the time limit.

The values of the parameters selected were based upon previous research (Cordón et al., 2002a; Dorigo and Stützle, 2004; Figlali et al., 2009). Due to the number of parameters and their levels applying a full factorial design would have led to excessive computation. To overcome this difficulty, an efficient fractional factorial experimental design was used.

The one-half fraction of the 2_{VII}^{7-1} experimental design (Montgomery, 2012) was adopted for the screening experiment, which reduced the number of computational runs by 50% per replication. The first instant problem was considered in this experiment and was repeated five times by using different random seed numbers. The computational results obtained from 320 (2^{7-1}∗5) runs were analysed by using a general linear model form of analysis of variance (ANOVA). Table 3 shows an ANOVA table consisting of Source of Variation (Source), Degrees of Freedom

Table 1
Characteristics of benchmarking instance problems considered in this work.

Problems	Characteristics of course timetabling problems							
	No. courses	No. of events	No. classrooms	Days/week	Periods/day	No. lecturers	No. curriculums	Unavailability constraints
1	30	160	6	5	6	24	14	53
2	30	162	5	5	9	24	13	94
3	54	152	9	6	6	47	139	771
4	72	251	16	5	5	61	68	382
5	79	286	18	5	5	70	57	396
6	85	275	17	5	5	68	60	486
7	121	390	19	5	5	95	78	691
8	131	434	20	5	5	99	77	677

Table 2
Experimental factors and its levels.

Factors	Levels	Values	
		Low (−1)	High (+1)
AI	2	20*45	45*20
α	2	1	5
β	2	1	5
ρ	2	0.1	0.9
σ	2	1	5
P_m	2	0.1	0.9
P_r	2	5%	95%

Table 3
ANOVA on the BWAS's parameters.

Source	DF	SS	MS	F	P
AI	1	21,929	21,929	6.880	0.009
α	1	1092	1092	0.340	0.559
β	1	31,141,969	31,141,969	9770.000	0.000
ρ	1	128,040	128,040	40.170	0.000
σ	1	7059	7059	2.210	0.138
P_m	1	84,273	84,273	26.440	0.000
P_r	1	112,538	112,538	35.310	0.000
$AI*\alpha$	1	2767	2767	0.870	0.352
$AI*\beta$	1	31,383	31,383	9.850	0.002
$AI*\rho$	1	4300	4300	1.350	0.246
$AI*\sigma$	1	17,746	17,746	5.570	0.019
$AI*P_m$	1	1679	1679	0.530	0.469
$AI*P_r$	1	16,951	16,951	5.320	0.022
$\alpha*\beta$	1	376	376	0.120	0.731
$\alpha*\rho$	1	372	372	0.120	0.733
$\alpha*\sigma$	1	1092	1092	0.340	0.559
$\alpha*P_m$	1	7022	7022	2.200	0.139
$\alpha*P_r$	1	1092	1092	0.340	0.559
$\beta*\rho$	1	72,391	72,391	22.710	0.000
$\beta*\sigma$	1	216	216	0.070	0.795
$\beta*P_m$	1	59,323	59,323	18.610	0.000
$\beta*P_r$	1	105,306	105,306	33.040	0.000
$\rho*\sigma$	1	9277	9277	2.910	0.089
$\rho*P_m$	1	65,009	65,009	20.390	0.000
$\rho*P_r$	1	128,040	128,040	40.170	0.000
$\sigma*P_m$	1	5994	5994	1.880	0.171
$\sigma*P_r$	1	7059	7059	2.210	0.138
P_m*P_r	1	166,030	166,030	52.090	0.000
Error	287	927,565	3188		
Total	319	33,127,888			

(*DF*), Sum of Square (*SS*), Mean Square (*MS*), and *F* and *P* values. A factor with value of $P \leq 0.05$ was considered statistically significant with a 95% confidence interval. All seven BWAS parameters were considered as the main sources of variation as well as the interaction effects between the parameters.

From Table 3, it can be seen that all BWAS's parameters except α and σ were statistically significant in terms of the main effect or interaction with a 95% confidence interval. The most influential factor was β followed by ρ, P_r, P_m, and *AI*, respectively. The most influential two-way interaction was P_m*P_r followed by ρ*P_r, β*P_r, β*ρ, ρ*P_m, β*P_m, *AI**β, *AI**σ, *AI**P_r, respectively. The main effect plots are shown in Figs. 7 and 8, suggesting that the main factors including *AI*, α, β, ρ, P_m, and P_r should be specified as 20∗45, 1 or 5, 5, 0.9, 0.1, and 5%, respectively. The power of mutation (σ) was found to be insignificant as a main effect, but it had a significant interaction effect with the *AI* parameter. The interaction effect plot for *AI*∗σ shown in Fig. 8. This suggests that the σ parameter should be set to 1 whilst the *AI* parameter should be 20∗45.

5.2. Confirming the parameter settings for the BWAS

The aim of this experiment was to confirm the appropriate parameter setting of BWAS that had been previously identified by the earlier experiment. The experiment was repeated 10 times using different random seeds. The total violation index (Z) results were analysed statistically in terms of their average and standard deviation (*SD*) from the best-so-far solutions. The BWAS results that adopted the optimised parameter setting were compared with the BWAS's results obtained using other randomised parameter settings and Cordón et al. (2000) settings as shown in Table 4. For a fair comparison, the total number of candidate solutions created during the stochastic search for exploring a solution space must be similarly defined. In this case, the total number of candidate solutions was determined by the number of ants multiplied by the number of iterations (*AI*), which were fixed at 20∗45 (900 solutions). From Table 4, it can be seen that the average best-so-far solutions obtained from the BWAS that adopted the optimised parameters identified in the previous experiment were significantly lower (i.e. less soft constraint violations) than the BWAS's results based upon the adoption of the settings used by other researchers (Cordón et al., 2000). The parameter settings identified by the previous experiment proved to be the optimal parameter setting for BWAS. The standard deviation (*SD*) of the BWAS results obtained from the optimised parameter showed lower variability compared with the other results. This demonstrated that the improved performance of ACO algorithms depended on both the mechanisms and the parameter settings. However, the appropriate parameter setting for the BWAS may vary depending upon the type of timetabling problem.

5.3. Performance comparison of BWAS and BWACS with five well-known ACO variants

The performance of the BWAS and BWACS has been successfully used to solve both the TSP (Cordón et al., 2000) and the QAP

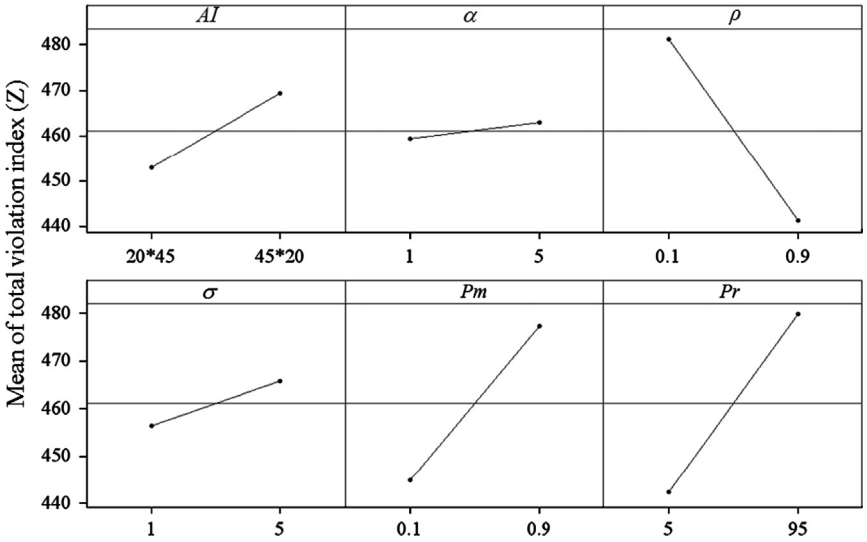

Fig. 7. Main effect plot of *AI*, α, β, ρ, P_m, and P_r factors.

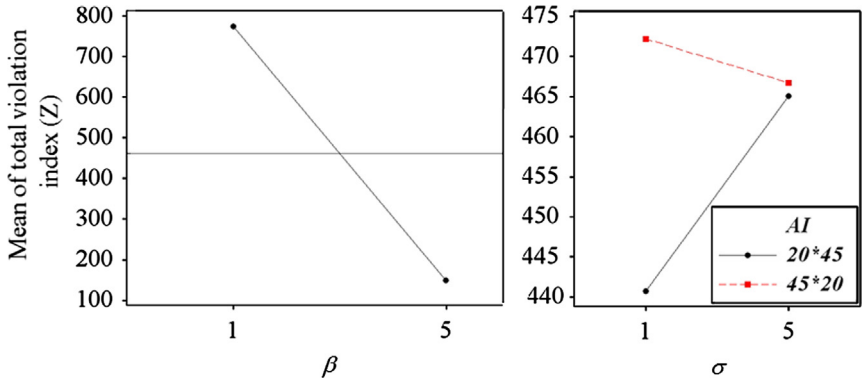

Fig. 8. Main effect plot of β factor and interaction plot of *AI**σ.

Table 4
BWAS's results from different parameter settings.

Parameter setting types	Factors and its levels (uncoded values)							Best-so-far solutions (violation index Z)	
	AI	α	β	ρ	σ	P_m	P_r	Average	SD
Optimised	20*45	1	5	0.9	1	0.1	5	100.30	28.21
Low setting	20*45	1	1	0.1	1	0.1	5	779.30	61.96
Medium setting	30*30	3	3	0.5	3	0.5	50	231.10	25.49
High setting	45*20	5	5	0.9	5	0.9	95	154.20	20.82
Cordón et al. (2000)	20*45	1	2	0.2	4	0.3	5	346.00	42.17

(Cordón et al., 2002a). However, the performance and speed of convergence of the methods have not been tested for course timetabling problems. The goal of this experiment was to compare the performance of BWAS and BWACS with five well-known ACO variants (including AS, EAS, AS-rank, MMAS, and ACS) based upon the construction of the timetables with the lowest total violation index (Z). The optimal parameter setting for the BWAS identified by the previous experiment was adopted, whilst the optimal parameter settings of other variant have been recommended by other research work, for example, MMAS and ACS (Lutuksin and Pongcharoen, 2009); BWACS (Lutuksin and Pongcharoen, 2010); AS and EAS (Dorigo and Stützle, 2004); and AS-rank (Thepphakorn and Pongcharoen, 2012).

Eight course timetabling problems (detailed in Table 1) were used to benchmark performance in terms of the minimum (*Min*), maximum (*Max*), average (*Avg*), and standard deviation (*SD*) for the best solutions obtained. Each proposed method was replicated 10 times using different random seed numbers. The statistical analysis on computational results shown in Table 5 indicates that the BWACS performed better than the other ACO variants for relatively large problems, whilst the AS-rank, MMAS, and ACS outperformed the other variants for relatively small and medium problems, respectively. This indicates that no particular variant of ACO method performed best for all sizes of the instant problems considered. The fifth problem was selected for comparing the amount of computation required to find the best-so-far solution. From Fig. 9, the BWACS convergence speed in early iterations was obviously better than the other ACO variants and remained so until the end of computational run. It can also be seen that the performance of BWAS as well as the EAS and MMAS was clearly faster than the AS, but the convergence result for BWAS was slower than the BWACS.

5.4. Performance comparison of BWAS and BWACS with/without LS strategies

This experiment was designed to demonstrate the performance improvement gained by embedding the proposed LS strategies into the BWAS and the BWACS algorithms. The optimal parameter setting for the BWAS was based on the previous experiment, whilst the optimal parameter setting of the BWACS was adopted from the previous research (Lutuksin and Pongcharoen, 2010), which systematically investigated and verified the results by using

Table 5
Performance comparison between seven ACO variants (measured in terms of violation index Z).

Problems	Descriptive statistics	ACO variants						
		AS	EAS	AS-rank	MMAS	ACS	BWAS	BWACS
1	*Min*	162	54	53	64	71	80	49
	Max	216	122	81	121	120	169	130
	Avg	183.4	85.7	64.4	81.7	105.6	100.3	82.4
	SD	19.4	22.2	7.2	15.5	14.1	28.2	24.9
2	*Min*	139	21	24	13	52	19	30
	Max	185	56	40	40	60	43	54
	Avg	168.8	34.0	32.2	25.2	56.8	33.8	38.4
	SD	15.2	11.2	6.0	8.0	2.6	6.6	7.6
3	*Min*	889	556	515	495	509	615	517
	Max	1315	720	607	729	652	754	572
	Avg	1097.2	624.8	546	562.8	542	684.2	543.5
	SD	121.9	63.1	24.6	72.8	43.3	50.7	19.6
4	*Min*	554	405	362	400	344	397	314
	Max	643	457	423	458	385	467	356
	Avg	592.5	434.2	404.3	432.0	361.9	438.6	341.4
	SD	28.3	17.5	18.2	18.8	14.1	18.8	13.3
5	*Min*	492	273	290	320	247	330	245
	Max	561	348	349	363	286	364	289
	Avg	525.9	321.9	323.3	346.1	271.2	349.5	261.9
	SD	24.3	21.9	19.4	12.6	11.4	12.0	12.1
6	*Min*	620	377	385	407	318	409	312
	Max	682	453	439	461	352	455	344
	Avg	648.5	410.7	403.4	433.4	336.2	441.8	329
	SD	16.4	25.8	16.0	15.3	9.3	13.5	10.8
7	*Min*	1267	610	583	642	487	658	473
	Max	1402	756	704	780	520	800	508
	Avg	1346.6	668.3	634.4	692.2	499.7	740.3	492.5
	SD	46.0	40.5	41.7	45.0	9.3	46.1	10.6
8	*Min*	936	545	550	620	421	639	419
	Max	1032	643	654	667	478	682	505
	Avg	983	598.2	578.4	646	455.2	659.2	453.1
	SD	27.4	27.7	29.8	16.2	17.7	14.5	21.9

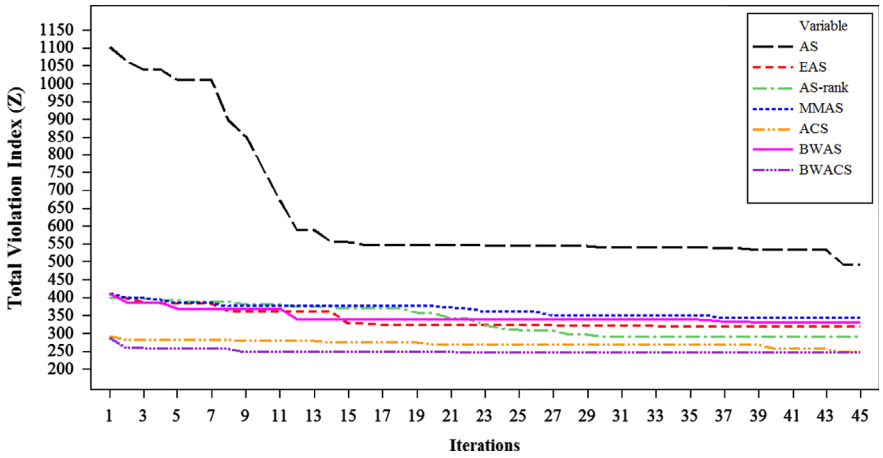

Fig. 9. Convergence graph of the best-so-far solution from ACO.

rigorous statistical tools. The computational results obtained from both methods included 10 replications with different random seeds. The results were analysed in terms of the minimum (*Min*), average (*Avg*), standard deviation (*SD*), execution time (*Time*) required to find the best so far solutions (timetables), and percentage improvement (*%Imp*) after applying the local search strategies.

The analysis of the experimental results is summarised in Table 6. According to the *Min*, *Avg*, and *SD* values of total violation index (Z) obtained from the BWAS and BWACS without LS, the BWAS only produced timetables with a lower total violation index (Z) than the BWACS for problem number 2. For the remaining problems, the BWACS produced better timetables than the BWAS. Table 6 shows that the total violation index of the timetables obtained from both BWAS and BWACS methods with LS strategies were better than those methods without LS strategies. Moreover, the combinations of local search strategies (LS1+LS2 and LS2+LS1) improved the performance of the BWAS and BWACS methods compared to using only LS1 or LS2 for all the instant problems except the third problem, in which the BWAS and BWACS with LS1 outperformed other approaches. For example, in the second instant problem, the combined local search strategy produced an

Table 6
Computational results obtained from the BWAS and BWACS with/without LS.

Problems	Methods	Local search	Best-so-far solutions																%Imp
			SC_1			SC_2			SC_3			SC_4			Total violation index (Z)				
			Min	Avg	SD	Min	Avg	SD	Min	Avg	SD	Min	Avg	SD	Min	Avg	SD	Time (hour)	
1	BWAS	No	4	18.1	25.5	0	13.0	8.6	20	37.2	12.0	30	32.0	2.3	80	100.3	28.2	1.1	0.0
		LS1	4	5.6	1.0	0	7.5	7.9	18	28.8	5.2	15	19.0	2.3	42	60.9	12.1	1.2	39.3
		LS2	4	13.0	15.7	0	5.0	10.0	6	18.4	9.9	24	29.2	3.3	40	65.6	22.5	1.4	34.6
		LS2+LS1	4	5.8	0.9	0	0.0	0.0	2	6.4	4.0	15	19.7	3.1	24	31.9	6.6	1.5	68.2
		LS1+LS2	5	6.7	3.0	0	1.5	2.4	4	11.8	5.5	21	24.5	3.4	32	44.5	7.1	1.5	55.6
	BWACS	No	4	26.7	26.7	10	14.5	3.7	8	22.8	8.1	15	18.4	3.3	49	82.4	24.9	1.1	0.0
		LS1	4	4.2	0.4	5	18.5	7.1	12	19.0	5.9	12	17.4	3.0	48	59.1	8.4	1.1	28.3
		LS2	4	49.2	24.1	0	4.0	6.2	2	13.6	8.6	17	22.2	3.7	51	89.0	22.4	1.3	−8.0
		LS2+LS1	4	5.0	0.7	0	0.5	1.6	0	6.0	3.7	14	20.1	4.0	23	31.6	6.5	1.4	61.7
		LS1+LS2	4	7.1	7.4	0	1.0	2.1	6	11.8	5.0	18	20.9	2.2	34	40.8	8.1	1.5	50.5
2	BWAS	No	0	1.3	3.2	5	11.5	7.1	10	13.2	2.9	4	7.8	2.6	19	33.8	6.6	2.3	0.0
		LS1	0	1.0	2.1	5	9.0	3.2	4	9.2	4.9	3	8.8	2.5	21	28.0	5.7	2.3	17.2
		LS2	0	0.0	0.0	0	0.0	0.0	0	2.4	2.5	5	9.3	2.1	7	11.7	3.4	2.8	65.4
		LS2+LS1	0	0.0	0.0	0	0.5	1.6	0	2.0	1.3	4	6.6	1.3	6	9.1	2.3	2.8	73.1
		LS1+LS2	0	0.0	0.0	0	0.0	0.0	0	1.6	1.6	5	7.2	1.4	6	8.8	2.0	3.0	74.0
	BWACS	No	0	0.5	1.6	10	17.5	7.9	4	13.0	4.2	6	7.4	1.2	30	38.4	7.6	2.2	0.0
		LS1	0	0.0	0.0	10	17.5	7.6	4	13.0	5.1	4	7.2	1.8	28	37.7	7.4	2.3	1.8
		LS2	0	0.3	1.0	0	1.0	2.1	0	3.2	2.5	6	10.2	2.7	6	14.7	4.1	3.3	61.7
		LS2+LS1	0	0.0	0.0	0	1.5	2.4	0	1.2	1.4	4	7.1	1.9	5	9.8	2.6	3.4	74.5
		LS1+LS2	0	0.0	0.0	0	1.0	2.1	0	2.0	1.9	6	9.2	1.7	10	12.2	1.4	3.3	68.2
3	BWAS	No	22	53.5	18.7	200	229.5	16.2	252	357.2	61.9	36	44.0	4.6	615	684.2	50.7	2.9	0.0
		LS1	0	6.5	6.3	220	247.5	19.9	220	291.6	37.4	11	16.0	2.7	509	561.6	29.3	2.7	17.9
		LS2	0	37.7	36.7	160	193.0	23.5	330	414.2	61.4	28	38.5	5.6	586	683.4	66.7	3.2	0.1
		LS2+LS1	0	11.5	10.0	135	186.5	32.3	292	394.4	69.0	17	20.8	2.4	502	613.2	64.5	4.0	10.4
		LS1+LS2	0	16.7	10.8	145	204.5	31.3	266	368.8	73.9	17	25.4	4.5	529	615.4	69.5	3.9	10.1
	BWACS	No	0	8.5	5.8	220	255.5	25.2	208	260.6	40.0	15	18.9	2.7	517	543.5	19.6	3.2	0.0
		LS1	0	7.5	7.9	225	255.5	18.6	220	255.2	30.9	8	11.3	2.3	493	529.5	22.3	2.9	2.6
		LS2	0	4.7	5.4	175	223.0	32.6	242	308.2	52.8	16	23.7	4.8	509	559.6	30.5	3.0	−3.0
		LS2+LS1	0	2.0	2.6	190	231.5	31.6	196	281.0	71.2	12	16.4	2.8	472	530.9	47.6	3.8	2.3
		LS1+LS2	0	7.0	6.8	205	239.5	19.9	226	282.8	37.3	14	19.9	3.7	502	549.2	28.8	3.4	−1.1
4	BWAS	No	0	10.0	7.3	120	155.5	18.9	140	155.6	12.9	110	117.5	4.2	397	438.6	18.8	3.9	0.0
		LS1	0	0.0	0.0	125	156.5	24.8	120	148.8	23.5	66	72.8	6.4	336	378.1	24.2	3.7	13.8
		LS2	0	3.6	5.5	25	61.0	24.1	154	187.6	20.2	104	112.6	4.6	297	364.8	28.3	5.5	16.8
		LS2+LS1	0	0.0	0.0	15	51.0	19.0	156	192.0	18.0	72	81.7	5.5	269	324.7	29.3	5.6	26.0
		LS1+LS2	0	0.0	0.0	35	65.5	24.9	162	184.8	14.3	73	88.8	11.9	303	339.1	29.2	5.5	22.7
	BWACS	No	0	0.0	0.0	140	168.5	16.3	102	124.0	17.8	36	48.9	8.6	314	341.4	13.3	4.5	0.0
		LS1	0	0.0	0.0	140	176.0	20.0	94	130.2	16.0	27	35.2	4.7	327	341.4	12.6	4.7	0.0
		LS2	0	0.8	2.5	25	61.0	24.5	138	183.0	26.9	78	88.4	6.7	304	333.2	18.5	7.8	2.4
		LS2+LS1	0	0.0	0.0	40	66.5	20.7	156	180.2	18.1	60	74.8	7.2	305	321.5	15.5	10.5	5.8
		LS1+LS2	0	0.0	0.0	30	65.0	21.7	146	185.6	23.4	62	76.7	9.1	315	327.3	13.0	9.9	4.1
5	BWAS	No	0	5.3	7.2	65	92.5	18.1	118	131.8	14.9	107	119.9	6.4	330	349.5	12.0	6.0	0.0
		LS1	0	0.3	1.0	60	84.0	16.0	120	138.2	13.6	79	90.9	7.1	293	313.4	10.0	6.5	10.3
		LS2	0	0.2	0.4	0	5.0	7.8	90	105.6	9.7	113	121.3	3.8	216	232.1	15.1	7.8	33.6
		LS2+LS1	0	0.0	0.0	0	9.0	7.4	100	111.4	9.0	76	83.7	4.5	183	204.1	16.6	8.4	41.6
		LS1+LS2	0	0.0	0.0	0	4.0	5.2	90	112.0	14.9	77	85.0	6.1	172	201.0	18.0	8.3	42.5
	BWACS	No	0	0.0	0.0	95	115.0	15.1	74	93.4	13.4	43	53.5	6.6	245	261.9	12.1	5.9	0.0
		LS1	0	0.0	0.0	105	125.5	15.7	76	89.0	9.0	41	46.6	5.2	239	261.1	12.9	6.5	0.3
		LS2	0	1.2	2.5	0	7.5	7.9	100	117.6	14.0	89	101.5	5.7	202	227.8	17.9	7.4	13.0
		LS2+LS1	0	0.0	0.0	0	10.5	12.1	96	117.6	14.8	76	81.6	3.5	179	209.7	19.4	7.8	19.9
		LS1+LS2	0	0.0	0.0	0	7.5	5.4	102	119.6	11.8	71	81.3	6.9	179	208.4	16.4	8.6	20.4

Table 6 (*continued*)

Problems	Methods	Local search	Best-so-far solutions																
			SC_1			SC_2			SC_3			SC_4			Total violation index (Z)				%Imp
			Min	Avg	SD	Min	Avg	SD	Min	Avg	SD	Min	Avg	SD	Min	Avg	SD	Time (hour)	
6	BWAS	No	0	3.3	4.7	125	146.5	19.3	144	163.4	16.0	121	128.6	4.0	409	441.8	13.5	5.5	0.0
		LS1	0	0.4	1.3	100	139.5	31.1	124	160.4	28.3	74	87.9	6.4	371	388.2	13.4	6.2	12.1
		LS2	0	1.2	2.0	0	12.5	7.6	156	176.2	12.3	113	123.5	5.9	295	313.4	17.7	7.7	29.1
		LS2+LS1	0	0.0	0.0	5	12.0	7.5	150	185.0	20.7	86	90.8	4.2	247	287.8	24.0	8.1	34.9
		LS1+LS2	0	0.0	0.0	5	11.5	5.8	134	169.4	17.3	84	93.0	7.2	233	273.9	23.7	8.1	38.0
	BWACS	No	0	0.0	0.0	120	149.5	19.4	102	127.4	13.1	37	52.1	10.5	312	329.0	10.8	5.7	0.0
		LS1	0	0.4	1.3	95	144.0	19.6	110	130.6	12.1	38	47.7	7.0	311	322.7	7.5	6.2	1.9
		LS2	0	0.4	1.3	10	16.5	6.7	158	175.4	12.9	96	104.7	5.0	274	297.0	16.8	7.2	9.7
		LS2+LS1	0	0.0	0.0	0	18.0	10.1	154	186.0	15.9	79	84.4	2.5	258	288.4	19.3	7.9	12.3
		LS1+LS2	0	0.0	0.0	10	18.0	6.3	160	186.8	15.7	80	86.9	4.3	263	291.7	13.7	8.1	11.3
7	BWAS	No	46	101.4	42.8	175	219.5	25.0	210	243.8	27.7	165	175.6	6.9	658	740.3	46.1	7.8	0.0
		LS1	0	10.4	12.2	170	200.5	22.9	182	219.2	27.5	111	125.5	6.6	533	555.6	24.9	7.8	25.0
		LS2	9	48.6	24.9	15	37.5	20.0	202	238.4	23.6	168	176.5	6.7	427	501.0	52.4	8.0	32.3
		LS2+LS1	0	5.1	8.9	0	21.0	14.5	160	203.6	28.5	117	132.1	8.2	314	361.8	29.1	12.6	51.1
		LS1+LS2	0	6.1	7.4	10	23.5	12.5	188	219.8	22.6	133	142.3	7.7	349	391.7	36.8	13.0	47.1
	BWACS	No	0	11.4	8.1	180	225.0	19.7	140	173.6	21.6	63	82.5	13.2	473	492.5	10.6	9.6	0.0
		LS1	0	8.7	10.5	205	232.5	15.0	160	175.0	13.5	64	78.4	12.0	470	494.6	18.8	9.8	−0.4
		LS2	7	19.3	10.1	5	29.5	17.6	172	227.8	25.5	147	159.0	8.5	418	435.6	14.0	9.9	11.6
		LS2+LS1	0	4.8	4.1	10	29.0	15.2	188	220.0	17.1	130	136.5	6.1	367	390.3	17.8	12.2	20.8
		LS1+LS2	0	1.9	2.7	10	19.5	10.9	176	215.2	23.4	127	138.7	6.9	323	375.3	25.1	12.2	23.8
8	BWAS	No	5	23.6	9.8	185	194.5	5.0	216	233.4	15.5	197	207.7	6.6	639	659.2	14.5	10.2	0.0
		LS1	0	4.2	5.4	160	185.0	15.1	190	213.0	14.5	151	167.8	10.7	552	570.0	12.4	11.3	13.5
		LS2	1	13.0	11.8	0	21.0	15.4	170	197.6	21.9	189	208.3	9.3	398	439.9	29.6	16.2	33.3
		LS2+LS1	0	0.2	0.6	0	14.5	14.8	138	195.8	25.0	158	173.7	9.2	315	384.2	36.3	17.7	41.7
		LS1+LS2	0	1.2	2.1	0	12.5	8.9	168	188.6	15.0	168	175.7	5.2	351	378.0	21.4	17.6	42.7
	BWACS	No	0	2.8	4.5	185	204.5	22.3	130	143.4	9.9	89	102.4	6.8	419	453.1	21.9	10.4	0.0
		LS1	0	0.0	0.0	180	218.5	24.7	116	147.0	17.3	81	93.5	9.2	432	459.0	27.1	12.1	−1.3
		LS2	0	4.3	4.4	5	25.0	14.9	162	178.8	10.4	159	180.9	12.1	372	389.0	12.6	14.7	14.2
		LS2+LS1	0	0.9	2.2	5	21.5	12.9	170	189.2	16.4	155	162.9	5.3	344	374.5	23.3	16.5	17.4
		LS1+LS2	0	1.2	2.1	5	20.5	8.0	160	186.6	14.6	161	169.5	5.4	336	377.8	19.4	17.1	16.6

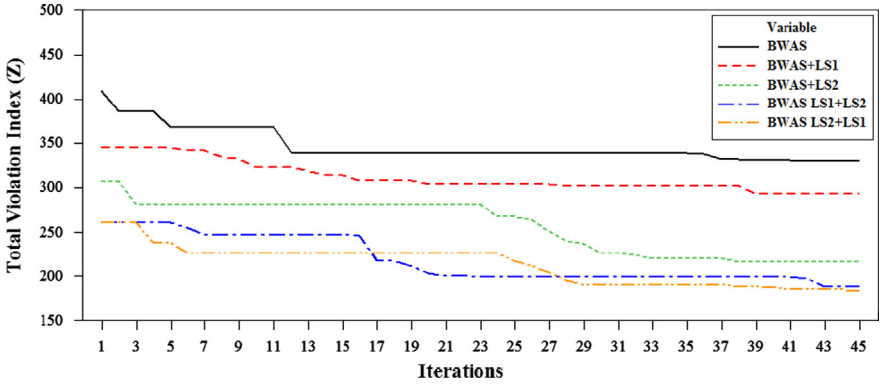

Fig. 10. Convergence graph of the best-so-far solution from BWAS with/without LS.

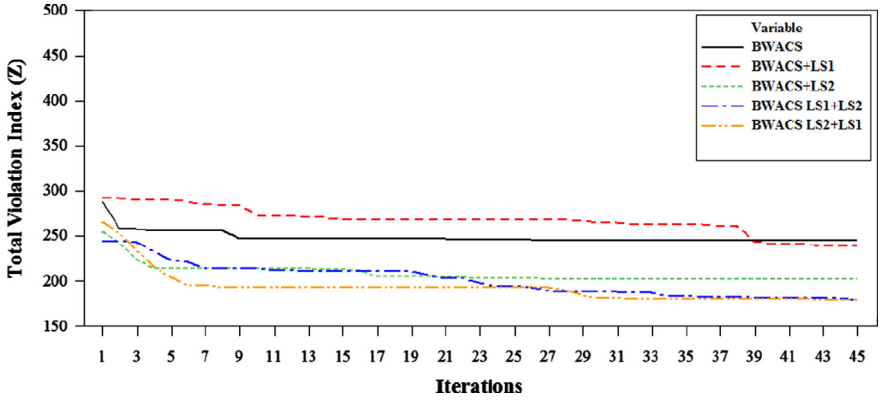

Fig. 11. Convergence graph of the best-so-far solution from BWACS with/without LS.

improvement of up to 70% compared to the classical BWAS and BWACS. The negative values of *%Imp* shown in Table 6 indicated that, in a few cases, the performance of BWACS with LS2 dropped slightly when compared to the results without local search. The possible reason was that the goal of the LS2 process was only to consider the violation of SC_2 and SC_3. There may have been an increase in SC_1 and SC_4 violations for problem number 1, whilst problem number 3 had more violations of SC_2 and SC_3 than the other problems.

In terms of the comparing the convergence of the results obtained from both the BWAS and the BWACS with/without LS strategies, the computational results from problem number 5, which was a relatively large problem, are shown in Figs. 10 and 11. From Fig. 10, the convergence of the results created by the BWAS with LS2+LS1 was the best followed by the BWAS with LS1+LS2, whilst the original BWAS without LS was the worst. From Fig. 11, the convergence in the results created by the BWACS with LS2+LS1 was the best followed by the BWACS with LS1+LS2. It can be seen that the LS strategies helped both BWAS and BWACS to find near optimal solutions quicker than the classical ant system methods. However, the average computational times for both methods with LS strategies was slightly increased in some problem sizes.

6. Conclusions

The BWAS and BWACS were used to solve timetabling problems. New Local Search (LS) strategies were developed and embedded into BWAS and BWACS to enhance the efficiency. This paper demonstrated the use of the experimental design and analysis tools to investigate the appropriate parameter settings before sequentially conducting a comparative study on the performance of the proposed methods. The analysis indicated that the efficiency of BWAS was dramatically improved by using the optimised parameter settings. This was investigated via the statistical design and analysis tools. The performance of the BWAS and BWACS in terms of the quality of the solutions obtained and its convergence speed was better than that of the original variants of ACO for course timetabling. The BWACS produced timetables with a lower total violation index for large problems than the other ACO variants, while AS-rank, MMAS, and ACS are better than other ACO methods for small problem. Finally, LS strategies help to improve the performance of both BWAS and BWACS, but required longer computational time than without introducing the strategies.

Acknowledgements

This work was partly supported by the Naresuan University Research Fund; grant number EN-AR-053/2552.

References

Ayob, M., Jaradat, G., 2009. Hybrid ant colony systems for course timetabling problems. In: Proceedings of the Second Conference on Data Mining and Optimization, 120–126.
Aytug, H., Khouja, M., Vergara, F.E., 2003. Use of Genetic Algorithms to solve production and operations management problems: a review. International Journal of Production Research 41, 3955–4009.
Azimi, Z.N., 2005. Hybrid heuristics for examination timetabling problem. Applied Mathematics and Computation 163, 705–733.
Blum, C., 2005. Ant colony optimization: introduction and recent trends. Physics of Life Reviews 2, 353–373.
Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surveys 35, 268–308.
Burke, E.K., Mareček, J., Parkes, A.J., Rudová, H., 2012. A branch-and-cut procedure for the Udine Course Timetabling problem. Annals of Operations Research 194, 71–87.
Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R., 2007. A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research 176, 177–192.

Burke, E.K., Newall, J.P., 2004. Solving examination timetabling problems through adaption of heuristic orderings. Annals of Operations Research 129, 107–134.
Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. European Journal of Operational Research 140, 266–280.
Chainual, A., Lutuksin, T., Pongcharoen, P., 2007. Computer based scheduling tool for multi-product scheduling problems. International Journal of the Computer, the Internet and Management 15, 26.1–26.6.
Chen, W., Fu, G., Tai, P., Deng, W., 2009. Process parameter optimization for MIMO plastic injection molding via soft computing. Expert Systems with Applications 36, 1114–1122.
Cordón, O., de Viana, I., Herrera, F, 2002a. Analysis of the Best-Worst Ant System and its variants on the QAP. Lecture Notes in Computer Science 2463, 228–234.
Cordón, O., de Viana, I., Herrera, F., Moreno, L., 2000. A new ACO model integrating evolutionary computation concepts: the best-worst ant system. In: Dorigo, M., Middendorf, M., Stützle, T. (Eds.), ANTS2000—From Ant Colonies to Artificial Ants: A Series of International Workshops on Ant Algorithms. Université Libre de Bruxelles, Belgium, pp. 22–29.
Cordón, O., Herrera, F., Stützle, T., 2002b. A review on the Ant Colony Optimization metaheuristic: basis, models and new trends. Mathware and Soft Computing 9, 141–175.
Daskalaki, S., Birbas, T., Housos, E., 2004. An integer programming formulation for a case study in university timetabling. European Journal of Operational Research 153, 117–135.
Di Gaspero, L., McCollum, B., Schaerf, A., 2007. The second international timetabling competition (ITC-2007): curriculum-based course timetabling track. In: Gavanelli, M., Mancini, T. (Eds.), Proceedings of the Fourteenth RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion, Rome, Italy.
Dino Matijaš, V., Molnar, G., Čupić, M., Jakobović, D., Dalbelo Bašić, B., 2010. University course timetabling using ACO: a case study on laboratory exercises. Lecture Notes in Computer Science 6276, 100–110.
Djamarus, D., Ku-Mahamud, K.R., 2009. Heuristic factors in ant system algorithm for course timetabling problem. In: Proceedings of the Ninth International Conference on Intelligent Systems Design and Applications, 232–236.
Dorigo, M., Birattari, M., Stutzle, T., 2006. Ant colony optimization - artificial ants as a computational intelligence technique. IEEE Computational Intelligence Magazine 1, 28–39.
Dorigo, M., Blum, C., 2005. Ant colonoty optimization theory: a survey. Theoretical Computer Science 344, 243–278.
Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge, Massachusetts.
Ejaz, N., Javed, M.Y., 2007. A hybrid approach for course scheduling inspired by die-hard co-operative ant behavior. In: Proceedings of the IEEE International Conference on Automation and Logistics, 3095–3100.
Eley, M., 2006. Some experiments with Ant Colony Algorithms for the exam timetabling problem. Lecture Notes in Computer Science 4150, 492–499.
Figlali, N., Ozkale, C., Engin, O., Figlali, A., 2009. Investigation of Ant System parameter interactions by using design of experiments for job-shop scheduling problems. Computers & Industrial Engineering 56, 538–559.
Glover, F., 1989. Tabu search – Part I. ORSA Journal on Computing 1, 190–206.
Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley, Massachusetts.
Jaradat, G.M., Ayob, M., 2010. An Elitist-Ant System for solving the post-enrolment course timetabling problem. Communications in Computer and Information Science 118, 167–176.
Johnson, F., Crawford, B., Palma, W., 2006. Hypercube framework for ACO applied to timetabling. International Federation for Information Processing 217, 237–246.
Kennedy, J., Eberhart, R.C., 2001. Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco.
Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.
Kostuch, P., 2005. The university course timetabling problem with a three-phase approach. Lecture Notes in Computer Science 3616, 109–125.
Lach, G., Lübbecke, M.E., 2012. Curriculum based course timetabling: new solutions to Udine benchmark instances. Annals of Operations Research 194, 255–272.
Lee, Y., Chen, C.Y., 2009. A heuristic for the train pathing and timetabling problem. Transportation Research Part B: Methodological 43, 837–851.
Leechai, N., Iamtan, T., Pongcharoen, P., 2009. Comparison on rank-based Ant System and shuffled frog leaping for designing multiple row machine layout. SWU Engineeing Journal 4, 102–115.
Lewis, R., 2008. A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum 30, 167–190.
Li, X., Baki, M.F., Aneja, Y.P., 2010. An ant colony optimization metaheuristic for machine-part cell formation problems. Computers & Operations Research 37, 2071–2081.
Lourenço, H.R., Martin, O., Stützle, T., 2002. Iterated local search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Kluwer Academic Norwell, pp. 321–353.
Lutuksin, T., Pongcharoen, P., 2009. Experimental design and analysis on parameter investigation and performance comparison of ant algorithms for course timetabling problem. Naresuan University Engineering Journal 4, 31–38.
Lutuksin, T., Pongcharoen, P., 2010. Best-worst ant colony system parameter investigation by using experimental design and analysis for course timetabling problem. In: Proceedings of the International Conference on Computer and Network Technology, 467–471.
MirHassani, S.A., 2006. A computational approach to enhancing course timetabling with integer programming. Applied Mathematics and Computation 175, 814–822.
Montgomery, D.C., 2012. Design and Analysis of Experiments, 8th ed. John Wiley & Sons, New York.
Naderi, B., Ghomi, S.M.T.F., Aminnayeri, M., 2010. A high performing metaheuristic for job shop scheduling with sequence-dependent setup times. Applied Soft Computing 10, 703–710.
Neto, R.F.T., Filho, M.G., 2011. An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. Computers & Operations Research 38, 1286–1293.
Nothegger, C., Mayer, A., Chwatal, A., Raidl, G.R., 2012. Solving the post enrolment course timetabling problem by ant colony optimization. Annals of Operations Research 194, 325–339.
Odajima, K., Hayashi, Y., Tianxia, G., Setiono, R., 2008. Greedy rule generation from discrete data and its use in neural network rule extraction. Neural Networks 21, 1020–1028.
Ousterhout, J.K., 2009. Tcl and the tk Toolkit, 2 ed. Addison-Wesley, Massachusetts.
Pansuwan, P., Rukwong, N., Pongcharoen, P., 2010. Identifying optimum Artificial Bee Colony (ABC) algorithm's parameters for scheduling the manufacture and assembly of complex products. In: Proceedings of the International Conference on Computer and Network Technology, 339–343.
Pongcharoen, P., Chainate, W., Pongcharoen, S., 2008a. Improving artificial immune system performance: inductive bias and alternative mutations. Lecture Notes in Computer Science 5132, 220–231.
Pongcharoen, P., Chainate, W., Thapatsuwan, P., 2007. Exploration of genetic parameters and operators through travelling salesman problem. ScienceAsia 33, 215–222.
Pongcharoen, P., Promtet, W., Yenradee, P., Hicks, C., 2008b. Stochastic optimisation timetabling tool for university course scheduling. International Journal of Production Economics 112, 903–918.
Shelokar, P.S., Siarry, P., Jayaraman, V.K., Kulkarni, B.D., 2007. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Applied Mathematics and Computation 188, 129–142.
Socha, K., Sampels, M., Manfrin, M., 2003. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. Lecture Notes in Computer Science 2611, 334–345.
Thapatsuwan, P., Chainate, W., Pongcharoen, P., 2008. Optimising multiple container packing using ant system. In: Proceedings of the Twelfth Annual Symposium on Computational Science and Engineering (ANSCSE 12), Ubon Ratchathani, Thailand, pp. 375–380.
Thapatsuwan, P., Pongcharoen, P., Hicks, C., Chainate, W., 2012. Development of a stochastic optimisation tool for solving the multiple container packing problems. International Journal of Production Economics 140, 737–748.
Thepphakorn, T., Pongcharoen, P., 2012. Heuristic ordering for ant colony based timetabling tool. Lecture Notes in Management Science 4, 87–96.
Zandieh, M., Amiri, M., Vahdani, B., Soltani, R., 2009. A robust parameter design for multi-response problems. Journal of Computational and Applied Mathematics 230, 463–476.