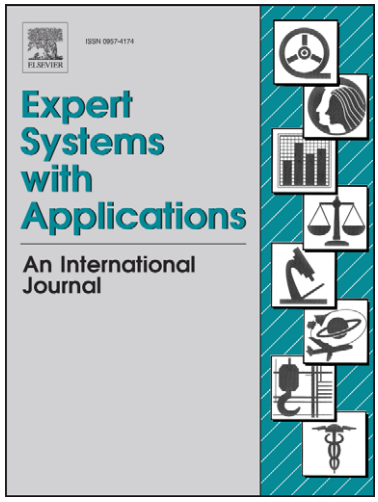


A New Bee Colony Optimization Algorithm with Idle-Time-Based Filtering Scheme for Open Shop-Scheduling Problems

Yueh-Min Huang, Jin-Chen Lin

PII: S0957-4174(10)01154-1
DOI: 10.1016/j.eswa.2010.10.010
Reference: ESWA 5381

To appear in: *Expert Systems with Applications*

Please cite this article as: Huang, Y-M., Lin, J-C., A New Bee Colony Optimization Algorithm with Idle-Time-Based Filtering Scheme for Open Shop-Scheduling Problems, *Expert Systems with Applications* (2010), doi: 10.1016/j.eswa.2010.10.010

Full Title: (ESWA5381)

A New Bee Colony Optimization Algorithm with Idle-Time-Based Filtering Scheme for Open Shop-Scheduling Problems

All authors and their affiliations

Yueh-Min Huang[1,2*]*, Jin-Chen Lin[1]

[1]Department of Engineering Science, National Cheng Kung University, Taiwan
No. 1, Ta-Hsueh Road, Tainan 701, Taiwan, R.O.C.

[2]Department of Applied Geoinformatics, Chia Nan University of Pharmacy and Science
60, Erh-Jen Rd, Sec. 1, Pao-An, Jen-Te, Tainan, 717, Taiwan, R.O.C.

*** Corresponding Author**: Yueh-Min Huang
Email: huang@mail.ncku.edu.tw
Tel: +886-6-2757575 ext. 63336
Fax: +886-6-2388123

A New Bee Colony Optimization Algorithm with Idle-Time-Based Filtering Scheme for Open Shop-Scheduling Problems

Abstract

Open Shop Scheduling Problems (OSSP) are one of the most time-consuming works in scheduling problems. Currently, many artificial intelligence algorithms can reduce the problem-solving time to an acceptable time range, and even can further downsize the range of solution space. Although the range of solution space is technically downsized, in most scheduling algorithms every partial solution still needs to be completely solved before this solution can be evaluated. For example, if there is a schedule with 100 operations, then all 100 operations must be scheduled before the scheduler can evaluate its fitness. Therefore, the time-cost of unnecessary partial solutions is no longer saved.

In order to improve the weakness stated above, this paper proposes a new bee colony optimization algorithm, with an idle-time-based filtering scheme, according to the inference of "the smaller the idle-time, the smaller the partial solution", and the smaller the makespan (Cmax) will be". It can automatically stop searching a partial solution with insufficient profitability, while the scheduler is creating a new scheduling solution, and therefore, save time-cost for the remaining partial solution. The architecture and details of the bee colony optimization heuristic rule is detailed in this paper.

Keywords: Bee Colony Optimization; Open shop scheduling

1. Introduction

The shop-scheduling problem can be simply introduced as a problem of redistribution of resources or a problem of rearrangement of operations order. Open Shop

Scheduling Problem (OSSP) is more difficult than JSSP and FSSP because its operations have no predefined order. When m>=3, OSSP is proved as a NP-complete problem [4,5].

In recent years, many heuristic rules for solving OSSP have been presented. One of the earliest heuristic rules is the branch and bound method, which regards the solution space as a tree with limited branches. Through the bound design of scientists, such as Lower Bound (the lower bound of a solution), the searching range of the solution space can be effectively downsized. The advantage of the branch and bound method is that the solution space is a clear tree structure, whose branches can be examined without omission to find a feasible solution. Its weakness is that the changes of feasible solutions are concentrated on certain branches, thus, that its feasible solution has an inherent lack of changeability.

Intelligent heuristic rules for various colonies have aroused widespread exploration and application in recent years, such as GA, ACO, PSO, BCO, and their combinations. These heuristic rules integrate the techniques of random a number generator, parallel operations, probability rules, fitness functions, tabu serial, etc., and in the face of a problem with large solution space, attempt to solve problems, where a feasible solution lacks changeability, and to effectively exclude bad solutions more quickly, in order to come close to the optima or even obtain the optima.

Sushil J. et al. [15] proposed to integrate the concepts of GA and CBRP, of which CBRP is directed at special cases to provide specific problem-solving inference principles. Christian Prins [16] suggested that, the characteristics of individuals in a GA colony were differed from each other, and their chromosomes could be rearranged to be applicable to the detection of a global optima from quasi-optimal schedules, meaning a

feasible solution with diversity, found through GA, could meet the needs of feasible solutions from a large solution space. Ching-Fang Liaw [17] developed a HGA that incorporated a local improvement procedure, based on TS, for solving the open shop scheduling problem. The incorporation of the local improvement procedure enabled the algorithm to perform a genetic search over the subspace of the local optima, and then, TS performed the local improvement procedure, which costs heavy computing time on a computer, and therefore, it is only performed in several rounds of the local improvement procedure. Jason Chao-Hsien Pan et al. [26] proposed a hybrid genetic algorithm (HGA) for no-wait job shop scheduling problems. In this paper, the chromosome was represented by a vector of integer numbers. The hybridizing method was the Order Crossover adopted by Brizuela et al. The mutation method was to randomly exchange two genes in the same chromosome. A genetic operation was defined by cutting out a section of genes from a chromosome, for treatment as a sub problem. This sub problem was then transformed into an asymmetric traveling salesman problem (ATSP), and solved with a heuristic algorithm. Subsequently, this section, with a new sequence, was returned to replace the original section of chromosome. The experimental results showed that this hybrid genetic algorithm could accelerate the convergence and improve solution quality.

Christian Blum [10] hybridized ant colony optimization (ACO) with a beam search to overcome difficult combinatorial optimization problems. This method caused the probability ACO mechanism to produce a group of complete solutions, and then used the beam search to perform improvement procedures of partial solutions. This method could randomly search the solution space and direct the solution to optimal branches.

D. Y. Sha and Cheng-Yu Hsu [2] modified the particle position representation using priorities, and the particle movement using an insert operator, and implemented a modified parameterized active schedule generation algorithm (mP-ASG) to decode a particle position in a schedule. In mP-ASG, the search area between non-delay schedules and active schedules could be reduced or increased by controlling the maximum delay time allowed. Changsheng Zhang et al. [27] proposed an alternate two-phase particle swarm optimization algorithm, called ATPPSO, to solve flow shop-scheduling problems, with an objective of minimizing makespan, which included two processes, the attraction process, and the repulsive process, which executed alternatively. In the attraction process, every particle was attracted to the individual optimum location and current global optimum location. In the repulsive process, every particle could avoid the worst individual location, which could not only diversify the colonies, but also increase the speed to the convergence of ATPPSO algorithm. I-Hong Kuo et al. [28] proposed an efficient a new hybrid particle swarm optimization model, named HPSO, to solve flow shop-scheduling problems. HPSO combined a random-key (RK) encoding scheme and an individual enhancement (IE) scheme. The RK encoding scheme could encode the locations in RK virtual space as locations in a FSSP solution space. In the RK virtual space, each location was represented by a vector of real numbers, while in FSSP solution space, each location was represented by a vector of integer numbers. For example, locations (0.3, 0.5, 0.2, and 0.4) in the RK virtual space could be encoded as locations (3, 1, 4, and 2) in the FSSP solution space. The advantage of using the RK encoding scheme was that it could make full use of the query ability of PSO. The idea of an IE scheme was to exchange the order of two operations in the same job, and if the makespan after the

exchange was good, it was retained. Using an IE scheme could enhance the local search ability of particle swarm.

Chin Soon Chong et al. [18] proposed a novel approach that used the honeybee foraging model to solve job shop-scheduling problems. This approach decided whether the branch from node A or node B was formed by the side length ratio between nodes A and B, as well as the heuristic distance between nodes A and B. The branching probability was calculated according the average profitability of the previous round. Li-Pei Wong et al. [19] published the Big Valley Landscape Exploitation (BVLE) method, which proposes to define a sole search space, called a landscape, when a heuristic search approach was applied to a combinatorial optimization problem. Exploring the search space with different numbers of search operations could create different landscapes, as the content of the landscape might change with the number of heuristic operations. The structures of these landscapes could help search for the global optima. In the BVLE structure, the local optima in some clusters may tend to appear near other local optima, and every cluster would form a valley centered on the global optimum. BVLE suggested that the new starting point for a search should be based on the previous local optimum, other than a random point in the search, because good candidate solutions can often be found near a local optimum.

This study proposes a new bee colony optimization algorithm, with an idle-time-based filtering scheme, which can automatically stop searching a partial solution with insufficient profitability, while the scheduler is creating a new scheduling solution, and therefore, save on time-cost for the remaining partial solution.

The remainder of this paper is organized as follows: Section 2 introduces OSSP; Section 3 describes the born foraging behaviors of a bee colony; Section 4 defines the computerized bee colony behaviors; Section 5 describes the origin and development of ASG, and then summarizes the mP-ASG used in experiments; Section 6 discusses the experimental results.

2. The Open Shop Scheduling Problem (OSSP)

Shop-scheduling problems (SSP) can be introduced simply as problems of redistribution of resources, or a problem of rearrangement of operational order. SSP is mainly divided into following three types: 1. FSSP, 2. JSSP, and 3. OSSP

As far as the difficulty for solving, OSSP is more difficult than JSSP and FSSP, due to the fact that it has no predefined order of operations in the same job. When m>=3, OSSP is proved to be an NP-complete problem [4,5].

In this paper, the definitions [6,7,and 8] of OSSP are summarized, as follows:

Suppose the existing resources are m sets of machines, and the jobs are j pieces.

(1) Each job contains m operations

(2) Each operation must cost time pij.

(3) The operations in a same job are randomly ordered, but at a time, only an operation can be handled.

(4) No preemptive, which indicates that no operation can interrupt other operations.

(5) At one time, a machine can handle only one operation, and each job can only be handled by one machine.

In this paper, the problem is to minimize the makespan (Cmax) of OSSP. Here, makespan has the same meaning as Cmax, that is, the time needed for completing all jobs.

3. The native foraging behavior of a honeybee colony

Frisch, Karl von (The 1973 Nobel Prize winner) and his students [11] decoded the dance language of bees. They noted that the direction information of a bee dance indicated the location of a food source, relative to the sun, and the distance of the food source was signaled by different kinds of dance. Wenner, Adrian M., and Patrick H. Wells [12, 13] argued that floral odors on a forager's body were the primary cues that enabled the recruit-bees to locate new food sources. Either dance languages or floral odors indicate that there is communication between bees that fulfill foraging behaviors.

The self-organization of bees [14] is based on several simple behavioral rules of insects. The best examples are locating and collecting nectar. Each bee collects nectar according to the route indicated by the forager bee that found the foraging route. Each hive has a dance area where the forager bee that found the food source dances to tell the foraging route, and in doing so, persuades other bees to follow the route to gather nectar.

When a forager bee prepares to leave the hive to gather nectar, it will follow a dancer bee towards the nectar area. After arriving at the nectar area, the forager bee will collect nectar, fly back to the hive, and release the nectar to the bees which are in charge of food storage. After the forager bee releases the nectar, it may execute the following operations (a) or (b) or (c) according to a specific probability:

(a) abandon the existing nectar area and fly to an unvisited area.

(b) return to the nectar area to collect food but not call other bees.

(c) come back to the beehive to call other bees to collect food in this nectar area.

In the “dance area” of a beehive, the dancer bee indicates different food areas by dances, which meanings are not yet clear. However, scientists hold that “dances are

functions that denote the quality of a food source", in other words, different dances might represent different qualities of food. In addition, not all worker bees go for food at the same time. As proven by experiments, the proportion of forager bees is related to both "the total number of bees" and "the number of forager bees outside". Based on the above description and understanding, this study develops a BCO computerized model, as defined in Section 4.

4. Computerized bee colony and its behavior

4.1. BCO algorithm

The foraging behaviors of bees include Forward Pass and Backward Pass. Forward Pass expresses the process of a forager bee leaving the beehive and flying towards a food source; Backward Pass denotes the process of a forager bee returning to the beehive and sharing the food source information with other forager bees (role change). The pseudo-code of the algorithmic framework of BCO developed by this paper is on the basis of foraging behaviors of bees, as shown in Fig. 1.

Figure 1 Algorithmic framework of BCO.

In Forward Pass, a forager bee will perform corresponding foraging operations in accordance with its roles (dancer, follower, or scout). The forager bee may evaluate the profitability of every partial route during a foraging trip. If the profitability of this partial route is not qualified, the forager bee will decide whether to **terminate** the foraging trip and return to the hive. In Backward Pass, upon a return to a hive from a food source, the forager bee must release the food (scheduling solution), and share the food source information with other forager bees (role change).

The termination criterion of a bee algorithm can be set as “fixed problem-solving time”, “fixed rounds”, or “fixed threshold value” (a value that is deemed as sufficiently good by the decision-maker). This paper adopts “fixed rounds” as the termination criterion. The details of forward pass and backward pass are described in Section 4.2 and Section 4.3. The termination criterion of OSSP can be defined as the fixed problem-solving time, or the fixed rounds to obtain an acceptable solution. This study employs fixed rounds as the termination criterion.

4.2. The forward pass of BCO

In OSSP, the foraging route of a forager bee is mapped according to priorities of a group of operational orders. In the forward process, if the profitability of the forager bee’s foraging route at node op meets the expectations of the bee colony, then the forager bee continues forward; however, if profitability does not meet expectations, then the forager bee abandons this forage route and returns to the hive. An algorithm of the forward pass of BCO is as shown in Fig. 2.

Figure 2 Algorithm for the forward pass of BCO.

According to Fig. 2, before leaving a hive, each forager bee colony first assigns roles. Each forager bee θ, based on the results of foraging in the last round, decides which role to play on this foraging trip. The roles fall into three types: scout, dancer, and follower, which are detailed as follows:

(1) Scout: If forager bee θ has never foraged, or the profitability of the foraging route it found in the last round is average, its role in this round would be as a scout. The computerized scout bee would randomly search for a foraging route.

(2) Dancer: If the profitability of a foraging route found by forager bee θ in last round is greater than the bee colony's expectations, its role would be as a dancer in this round. The computerized dancer bee will go to the food source, according to the foraging route of the last round; until the idle-time in that route is greater than zero, it will then go to search for a new foraging route.

(3) Follower: If the profitability of a foraging route found by forager bee θ in the last round is smaller than the bee colony's expectations, its role would be as a follower in this round. The computerized follower bee would go to the food source following the dancer bee; until the idle-time in the foraging route of the last round is greater than zero, it would randomly search for a new foraging route. The follower randomly selects a dancer to follow.

According to the inference of "the smaller the idle-time of a partial solution is, the smaller the makespan (Cmax) will be", this paper presents An Idle-Time Based Filtering Algorithm (ITBF), which automatically stops searching a partial solution of insufficient profitability, while the scheduler is creating a new scheduling solution. This filtering algorithm could be written into pseudo-code, as shown in Figure 3.

Figure 3 An Idle-Time Based Filtering Algorithm (ITBF)

In accordance with Fig. 3, when a forager bee leaves the hive, each node that forager bee θ passes through in the foraging route directly corresponds to one schedulable operation of a schedule. For example, the node op in the foraging route directly corresponds to the schedulable operation op in an actual schedule.

While the forager bee advances, if the yield rate at the current node meets the expectations of the bee colony, the forager bee will continue to advance. However, if the yield rate at the current node cannot meet the expectations of the bee colony, the forager bee would abandon its foraging and return directly to the hive.

Pf_θ is the yield rate produced by forager bee θ, upon arriving at the node.

Pf_{colony} is the yield rate generally accepted, which is produced by the whole forager bee colony, upon arriving at the node.

C^θ is the makespan ($C_{\max}$) of forager bee θ.

The new bee colony optimization algorithm developed in this paper takes each scheduling operation as a foraging area; during the course of foraging, the idle-time of partial scheduling is regarded as reciprocal of profitability, thus, the larger the idle-time is, the smaller the profitability is. When the profitability of a partial foraging route of a bee is smaller than the average acceptable profitability of the whole bee colony, this scheduler would automatically stop searching the foraging route of this bee, while the scheduler is creating a new scheduling solution, saving the time-cost of the remaining partial solution.

The paper employs particle swarm optimization [25] to divide the optimum solution into individual optimum solutions, and colony optimum solution. This paper separates yield rate into individual yield rates, and colony yield rate:

(1) Individual yield rate = Pf_θ = the yield rate produced by forager bee θ, upon arriving at the node.

(2) Colony yield rate = Pf_{colony} = the yield rate generally accepted, which is produced by a forager bee, upon arriving at the node.

Pf_{θ} should be calculated by the mathematical axiom 4.2.A, and Pf_{colony} by the mathematical axiom 4.2.B.

$$Pf_{\theta} = \frac{1}{I_{op}^{\theta}} \text{.....(4.2.A)}$$

where I_{op}^{θ} is the total idle time produced by forager bee θ, upon arriving at node op.

$$Pf_{colony} = \frac{1}{I_{op}^{best} \times \varepsilon} \text{.....(4.2.B)}$$

where I_{op}^{best} is the total idle time produced by the best forager bee, upon arriving at node op.

The results of the best forager bee represent the whole forager colony's advance, which currently produces the shortest total idle time. ε denotes one real number larger than zero, which can be set by the user at the decision-maker's discretion. A larger ε indicates the lower yield rate generally accepted by the whole forager bee colony, and the longest allowable idle time, and longest allowable time for matter solution.

When forager bee θ arrives at node op, the scheduler immediately carries out an assessment of the yield rate produced by forager bee θ from the start to the present. If the yield rate produced by forager bee θ, upon arriving at the node, is less than the yield rate generally accepted, which is that produced by the whole forager bee colony, and the makespan produced by the forager bee θ is equal to infinite, then the foraging journey would be ended. The behavior of an advancing forager bee is stopped in its search for a

food source upon arrival at the node, when its individual yield rate is less than the acceptable colony yield rate. This is the reason why this new bee colony optimization algorithm presented in this paper can greatly reduce search operation time.

4.3. The backward pass of BCO

When returning, the forager bees returns to the hive, puts down the nectars and pollens for the food-storing bees, and then decides and executes the following rules of bee classification:

(1) Changing into a dancer bee: the dancer bees persuade other bees to return to a food source through performing dances. To change into a "dancer bee", the conditions of formula 4.3.A1 must be met, in which C^{dancer} is the minimum standard for changing into a dancer bee. The makespan consumed by forager bee θ must be less than, or equal to, C^{dancer} in order to change into a dancer bee. C^{dancer} is calculated by formula 4.3.A2, in which β is calculated by 4.3.A3. The dancer ratio included in formula 4.3.A3 is the parameter value introduced from outside the scheduler. Dancer_ratio controls the quantity of dancer bees, and dancer_ratio=50 means that the quantity of the dancer bees must account for 50% of the forager bees.

$$C^{\theta} \le C^{dancer} \text{.....(4.3.A1)}$$

$$\{ C^{\theta} \mid C^{\theta} >= P_{\beta}, C^{\theta} \ne \infty, P_{\beta} = \text{the } \beta\text{-}th \text{ percentile of } C \} \text{.....(4.3.A2)}$$

$$\beta = (dancer\_ratio * 100) \text{.....(4.3.A3)}$$

(2) Changing into follower bees: the follower bees go to a better food source following the dancer bees. To change into a "follower bee", the conditions of the formula 4.3.B1 must be met, in which $C^{follower}$ is the minimum standard for changing into a

follower bee. The makespan consumed by forager bee θ must be less than, or equal to, $C^{follower}$ in order to change into a dancer bee. γ value is included in formula 4.3.B2, and calculated by formula 4.3.B3, in which the follower ratio is a parameter value introduced from outside the scheduler. Follower ratio controls the quantity of follower bees. Follower ratio=50 means that the quantity of the follower bees account for 50% of the forager bees.

$$C^{\theta} > C^{follower} \text{.....(4.3.B1)}$$

$$\{ C^{\theta} \mid C^{\theta} >= P_{\gamma}, C^{\theta} \neq \infty, P_{\gamma} = \text{the } \gamma\text{-}th \text{ percentile of C} \}\text{.....(4.3.B2)}$$

$$\gamma = (follower\_ratio * 100)\text{.....(4.3.B3)}$$

(3) Changing into scout bees: the scout bees look for food sources in an independent and random manner, without calling together other bees or following any dancer bee. All forager bees that fail to meet formulas 4.3. A1 and 4.3.B1 would be changed into scout bees.

Whole bee colony optimization algorithm: the returning course could be transformed into pseudo codes, as shown in Figure 4.

Figure 4 New bee colony optimization algorithm: returning course

5. Active Schedule Generation

Active Schedule Generation (ASG) algorithms may be derived from the definition of left-shift adjustment scheduling, by Wiest [20]. Arno Sprecher et al. [21], with respect to the Resource-Constrained Project Scheduling Problem (RCPSP), gave a formal

definition to semi-active schedules, active schedules, and non-delay schedules, respectively.

Goncalves et al. [22] argued that the best schedule lies in a set of all active schedules. However, such a set is extremely large, and contains schedules with relatively large time delays. Therefore, such schedules have poor quality results in solving such matters as makespan and Cmax. To lessen solution space, Goncalves et al. turned to the P-ASG, whose basic concept lies in the control of permitted time delays of each operation. By means of controlling the maximum allowable time delay, the scheduler is able to decrease or increase its solution space.

The mP-ASG algorithm introduced by D. Y. Sha et al. [2, 3] is a modification of the original P-ASG [22], called the Modified Parameterized Active Schedule Generation (mP-ASG). Such an algorithm can more precisely control the search range of a solution space. By using such superior characteristics, the "Bee Foraging Route" is decoded into an actual schedule. The mP-ASG published by D. Y. Sha et al. is shown in Figure 5.

Figure 5 The mP-ASG algorithm, developed by D. Y. Sha et. al.

6. Experimental results

6.0. Experimental Process

The computer utilized in this experiment is a Lemel M8066 notebook computer, produced by the Synnex Technology International Corporation.

The central processing unit (CPU) used is Intel® Pentium® M 740(1.73 GHz).

The operating system used is Microsoft Windows XP, with SP2.

The program language used is Microsoft Visual Basic 6.0.

According to the paper concerning the application of the ASG series algorithm (ASG, P-ASG, mP-ASG), by Goncalves et al. [22] and D. Y. Sha et al. [2, 3], this paper summarizes two phases required for an ASG series algorithm, when applied in an actual solution of scheduling.

(1) The precedence of all schedulable operations is determined based on select heuristic rules.

(2) The ASG series algorithm is applied to decode the precedence of schedulable operations into an actual schedule (scheduling solution)

Based on the above-mentioned two phases required for an ASG series algorithm in an actual application of a scheduling solution, this paper designs experimental processes, through which the efficiency of two algorithms are compared, as shown in Figure 6.

Figure 6 The flowchart of Experimental Process

Algorithm A utilized in this paper is the PSO-mP-ASG, developed by D. Y. Sha et al. [2]; Algorithm B is the BCO-mP-ASG developed by this paper. The test questions and results applied herein are described in the next section.

6.1. Test set of experiments

The test questions are the tai_20 × 20_* OSSP test set presented by Taillard [23]. The test set provides two groups of experimental parameters, i.e. “processing time” and “machine configuration”, each of which is composed of a 20x20 numerical matrix. For OSSP, question complexity reaches 400!

6.2. Test of experimental fairness

The computer utilized in this experiment is a Lemel M8066 notebook computer, produced by the Synnex Technology International Corporation.

The central processing unit (CPU) used is Intel® Pentium® M 740(1.73 GHz).

The operating system used is Microsoft Windows XP, with SP2.

The program language used is Microsoft Visual Basic 6.0.

To strengthen experimental fairness, this experiment employs the same program language, and four tests are conducted on the same hardware platform, in order to determine: (1) whether there is a proportional relationship between PSO-mP-ASG and BCO-mP-ASG, with no ITBF, in terms of the number of solutions achieved, within a specified time; (2) whether there is a proportional relationship between PSO-mP-ASG and BCO-mP-ASG, with no ITBF, in terms of the difference in optima achieved, within the same quantity of colony individuals, and the same execution time; (3) whether there is a proportional relationship between PSO-mP-ASG and BCO-mP-ASG, with no ITBF, in terms of test results of the same specified quantity of colony individuals, and the same execution time; (4) What effect may be exerted upon the number of solutions achieved by BCO-mP-ASG, to which ITBF is added, within a specified time. The test results are shown in Figure 7, Figure 8, Figure 9, and Figure 10, respectively.

Figure 7 Number of solutions achieved by PSO-mP-ASG within the specified time

Figure 8 Number of solutions achieved by BCO-mP-ASG, with no ITBF, within the specified time

The test results, as shown in Figure 7 and Figure 8, reveal that the number of solutions achieved by PSO-mP-ASG is fully identical with that by BCO-mP-ASG, with no ITBF.

Figure 9 Test results of PSO-mP-ASG and BCO-mP-ASG with no ITBF

The test results shown in Figure 9 reveal that no great gap occurs among all experimental results of PSO-mP-ASG and BCO-mP-ASG, with no ITBF, with the same number of colony individuals and the same execution time. For example, the average gap of Best solutions achieved by PSO-mP-ASG is 1.32%; the average gap of Best solutions achieved by BCO-mP-ASG, with no ITBF, is 1.35%; the average gap of Average solutions achieved by PSO-mP-ASG is 10.60%; the average gap of Average solutions achieved by BCO-mP-ASG, with no ITBF, is 10.02%; the average t (time for achieving optima) required for PSO-mP-ASG is 11.67s, while the average t (time for achieving optima) required for BCO-mP-ASG, with no ITBF, is 11.72s.

Experimental parameters used in Figure 9 are as follows:

(1) Parameters for the PSO-mP-ASG

The particles = 30, run time duration = 30 seconds.

The inertia weight w was decreased linearly from wmax to wmin during a run.

The parameter wmax= 0.9, wmin=0.3. The initial value of w = 0.9.

The parameter c1=0.2, c2=0.8.

The parameter delayweight was increased linearly from 0 to 1 during a run.

The initial value of delayweight = 0.01.

No mutation operator was implemented.

(2) Parameters for the BCO-mP-ASG with no ITBF

The bees = 30, run time duration = 30 seconds

The dancer ratio = 0.3.

The follower ratio = 0.7.

The imitation ratio = 0.6.

The parameter ε = 11.88.

The parameter delayweight was increased linearly from 0 to 1 during a run.

The initial value of delayweight = 0.01.

6.2.4. Effect on number of solutions achieved with ITBF added within the specified time

To understand what effect may be exerted upon the number of solutions achieved by BCO-mP-ASG, to which ITBF is added within a specified time, one test is conducted to observe the difference between the number of solutions achieved by BCO-mP-ASG with BCO-mP-ASG, and that of BCO-mP-ASG with no BCO-mP-ASG. The test method is as follows: BCO-mP-ASG, with ITBF, is utilized to continue solving each test question of tai_20 x 20_* series within the specified time, and when one integral scheduling solution is obtained, add one to the counter.

Figure 10 Effect on number of solutions achieved with ITBF added to BCO-mP-ASG within a specified time

Figure 10 shows that with respect to the test question tai_20 x 20_1, the number of solutions achieved by BCO-mP-ASG is increased from 90 to 450 within 10s after ITBF is

added, of which S&B is 437, and DCS is 13. S&B refers to the times of forager bees returning to the hive from showing the way to a foraging location, and DCS means the times of the forager bees returning to the hive after completing their foraging journey.

6.3. Test and comparison of BCO-mP-ASG with ITBF and PSO-mP-ASG

6.3.1. Parameters for the PSO defined by D. Y. Sha et. al.

The particles = 30.

The inertia weight w was decreased linearly from wmax to wmin during a run.

The parameter wmax= 0.9, wmin=0.3. The initial value of w = 0.9.

The parameter c1=0.2, c2=0.8.

The parameter delayweight was increased linearly from 0 to 1 during a run.

The initial value of delayweight = 0.01.

No mutation operator was implemented.

Run until the average gap of Best solutions below 1%.

6.3.2. Parameters for the BCO defined by Y.M. Huang and J.C. Lin

The bees = 30.

The dancer ratio = 0.3.

The follower ratio = 0.7.

The imitation ratio = 0.6.

The parameter ε = 11.88.

The parameter delayweight was increased linearly from 0 to 1 during a run.

The initial value of delayweight = 0.01.

Run until the average gap of Best solutions below 1%.

6.3.3. Test results and Tables

Figure 11 The test results of PSO-mP-ASG and BCO-mP-ASG with ITBF

Figure 11 shows the experimental results of PSO-mP-ASG and BCO-mP-ASG with ITBF solving the test questions of tai_20 x 20_* series. In this figure, BKS (Best-Known Solution) is a known optimum solution, Best is the optimum solution, T is the total time to achieve all solutions, t is the time to achieve the optimum solution, Average (average solution) is the average value of the sum of all solutions, Average time is the average time consumed, which is preferably small, and Average gap is the gap of average solutions, in which gap (gap value) the equal to (Cmax – BKS) / BKS (the smaller gap of one solution indicates that the solution has a smaller gap with BKS, and thus, it is a better quality solution).

The average gap of Best solution achieved by PSO-mP-ASG solving the test question of tai_20 x 20_* series is 0.5%, while that of the BCO-mP-ASG solution of the same question is 0.65; the difference between them is only 0.15%. However, the average T (time for achieving all solutions) required for PSO-mP-ASG is 52.86s, while that of BCO-mP-ASG with ITBF is only 12.04. Furthermore, the average t (time for achieving optima) required for PSO-mP-ASG is 29.74, while that of BCO-mP-ASG with ITBF is only 5.89.

Figures 12, 13, 14, and 15 show the differences, through bar charts, between the Best-Known solutions, PSO-mP-ASG solutions, and BCO-mP-ASG solutions.

Figure 12 The comparison of Best Cmax

The X axle, in Figure 12, represents the number of test questions, where 1 denotes the test question tai_20 × 20_1, and 2 denotes the test question tai_20 × 20_2, and 3 denotes the test question tai_20 × 20_3, etc. The Y axle denotes Best Cmax (optimum value of makespan). The left bar in the bar chart denotes Best-Known Solutions, the middle bar denotes the experimental results produced by PSO-mP-ASG, and the right bar denotes the experimental results produced by BCO-mP-ASG. The figure shows that Best Cmax of both BCO-mP-ASG and PSO-mP-ASG very closely approach the Best-Known Solution.

Figure 13 The comparison of Average Cmax

The X axle in Figure 13 represents the number of test questions, where 1 denotes the test question tai_20 × 20_1, and 2 the test question tai_20 × 20_2, and 3 the test question tai_20 × 20_3, etc. The Y axle represents Average Cmax (average value of makespan). The left bar in the bar chart denotes the experimental results produced by PSO-mP-ASG, and the right bar denotes the experimental results produced by BCO-mP-ASG. The figure shows that the Average Cmax of BCO-mP-ASG is slightly closer to the Best-Known Solution, than that of PSO-mP-ASG.

Figure 14 The comparison of T (Time cost for all solutions)

The X axle in Figure 14 represents the number of test questions, where 1 denotes the test question tai_20 × 20_1, and 2 the test question tai_20 × 20_2, and 3 the test question tai_20 × 20_3, etc. The Y axle represents T (time for achieving all solutions). The left bar

in the bar chart denotes the experimental results produced by PSO-mP-ASG, and the right bar denotes the experimental results produced by BCO-mP-ASG. The figure shows that T time required by BCO-mP-ASG is obviously less than that required by PSO-mP-ASG.

Figure 15 The comparison of t (Time cost for best solution)

The X axle in Figure 15 represents the number of test questions, where 1 denotes the test question tai_20 × 20_1, and 2 the test question tai_20 × 20_2, and 3 the test question tai_20 × 20_3, etc. The Y axle represents t (time for achieving optima). The left bar in the bar chart denotes the experimental results produced by PSO-mP-ASG, and the right bar denotes the experimental results produced by BCO-mP-ASG. The figure shows that t time required by BCO-mP-ASG is obviously less than that required by PSO-mP-ASG.

7. Conclusions

This paper proposes a novel bee colony optimization algorithm for solving OSSP, which could largely decrease the T (the summation of all time-costs for obtaining all solutions) and t (the time-cost for obtaining the optima) times of the algorithm.

The concrete details of heuristic rules of solving combinatorial optimization problems would vary with different problems. In order to shorten the time-cost for obtaining a possible bad solution, an idle-time-based filtering scheme is presented based on the processes of the Forward Pass of bee's foraging behaviors. The scheme takes each scheduling operation as a foraging area; and in the course of bee colony foraging, the idle-time of partial scheduling is regarded as the reciprocal of profitability, therefore, the larger the idle-time is, the smaller the profitability is. When the profitability of a partial foraging route of a bee is smaller than the average profitability accepted by the colony,

this scheduler would automatically stop searching the foraging trip of this particular bee, while the scheduler continues creating a new scheduling solution, thus, saving time-cost for remaining partial solutions.

Teodorovic and Dusan et al. [1] adopted the operations of abandon, continue, dance, and recruitment of the peers in the Backward Pass. In order to highlight the classification operations of information sharing and division of work by forager bees in the Backward Pass, this paper adjusts the Backward Pass for the roles of classification operations for dancer, follower, and scout. This adjustment converts the classification of abandon routes, dance, and recruitment the peers into the classification of follower or dancer, and turns the operation of continuing in the classification of scout. This classification makes clear the roles played by different forager bees.

For the BCO heuristic rules, future study could attempt to develop a new "profitability equation", and new methods to "follow" the dancer bee.

For OSSP, future study can use a "different decoder" to decode the priorities of various heuristic rules into a schedule.

The new BCO framework developed by this paper provides a novel method of viewing computerized bee colony heuristic rules. Future researchers could use or add this new BCO framework into their fields of application for solving other scheduling problems or combinatorial optimization problems.

References

[1] Teodorovic, Dusan; Lucic, Panta; Markovic, Goran; Orco, Mauro Dell'. Bee Colony Optimization: Principles and Applications. NEUREL 2006. 8th Seminar on 25-27 Sept. 2006;151-156.

[2] D. Y. Sha, Cheng-Yu Hsu. A new particle swarm optimization for open shop scheduling problems. Computers & Operations Research. 2008;35:3243-3261.

[3] D. Y. Sha, Cheng-Yu Hsu. A modified parameterized active schedule generation

algorithm for job shop-scheduling problems. The 36th CIE Conference on Computers & Industrial Engineering. 2006;

[4] Garey MR, Johnson DS. Computers and Intractability: a guide to the theory of NP-completeness. Freeman, San Francisco. 1979;

[5] Gonzalez T, Sahni S. Open shop scheduling to minimize finish time. Journal of the Association for Computing Machinery, 1976: 23(4):665-679.

[6] Brucker P, Hurink J, Jurisch B,Wostmann B. A branch and bound algorithm for open-shop problems. Discrete Applied Mathematics. 1997;76:43–59.

[7] Christelle Gueret, Christian Prins. Classic and novel heuristics for open-shop problems: A computational evaluation. European Journal of Operational Research. 1998;107:306-314.

[8] Ching-Fang Liaw. A tabu search algorithm for open shop scheduling problems. Computers & Operations Research. 1999;26:109-126.

[9] Giffler B, Thompson GL. Algorithms for solving production-scheduling problems. Operations Research, 1960 8(4):487–503.

[10] Christian Blum. Beam-ACO - hybridizing ant colony optimization with a beam search: an application to open shop scheduling. Computers & Operations Research. 2005;32:1565-1591.

[11] Frisch, Karl von. The Dance Language and Orientation of Bees. The Belknap Press of Harvard University Press. 1967;

[12] Wenner, Adrian M., and Patrick H. Wells. Anatomy of a Controversy: The Question of a "Language" Among Bees. Columbia University Press. 1990;

[13] David R. Tarpy. THE DANCE LANGUAGE OF THE HONEY BEE. Dissertation, NC state university. 2009; http://www.cals.ncsu.edu/entomology/apiculture/Dance_language.html http://en.wikipedia.org/wiki/Honey_bee

[14] S. Camazine, J. Sneyd, A Model of Collective Nectar Source by Honey Bees: Self-organization Through Simple Rules. Journal of Theoretical Biology. 1991;149:547-571.

[15] Sushil J. Louis, Zhijie Xu. Genetic algorithms for open shop scheduling and re-scheduling. Dissertation. Department of Computer Science. University of Nevada. 1996;

[16] Christian Prins. Competitive genetic algorithms for open-shop scheduling problems. Dissertation. University of Technology of Troyes, Department of GSI. 2000;

[17] Ching-Fang Liaw. A hybrid genetic algorithm for open shop scheduling problems. European Journal of Operational Research. 2000;124:28-42.

[18] Chin Soon Chong, Appa Iyer Sivakumar, Malcolm Yoke Hean Low, Kheng Leng Gay. A bee colony optimization algorithm for job shop scheduling. Proceedings of the 38th conference on Winter simulation, Winter Simulation Conference, IEEE. 2006;

[19] Li-Pei Wong, Chi Yung Puan, Malcolm Yoke Hean Low, Chin Soon Chong. Bee

Colony Optimization algorithm with Big Valley landscape exploitation for Job Shop Scheduling problems. Simulation Conference, 2008. WSC 2008. Winter 7-10 Dec, IEEE. 2008;2050 - 2058.

[20] Wiest, J.D. Various properties for schedules of large projects with limited resources. Operations Research. 1964;12:395-418.

[21] Arno Sprecher, Rainer Kolisch, Andreas Drexl. Semi-active, active, and non-delay schedules for resource-constrained project-scheduling problems. European Journal of Operational Research. 1995;80:94-102.

[22] Gonçalves JF, Mendes JJ, de M, Resende MGC. A hybrid genetic algorithm for job shop-scheduling problems. European Journal of Operational Research, 2005 167(1):77–95.

[23] Taillard E. Benchmarks for basic scheduling problems. European Journal of Operations Research, 1993 64:278–85.

[24] Michael L. Pinedo. Scheduling Theory, Algorithms, and Systems. Springer, 2008.

[25] James Kennedy and Russell Eberhart. Particle Swarm Optimization, IEEE.1995.

[26] Jason Chao-Hsien Pan, Han-Chiang Huang. A hybrid genetic algorithm for no-wait job shop scheduling problems. Expert Systems with Applications, 2009 36:5800–5806.

[27] Changsheng Zhang, Jigui Sun. An alternate two phases particle swarm optimization algorithm for flow shop scheduling problems. Expert Systems with Applications, 2009 36:5162–5167.

[28] I-Hong Kuo, Shi-Jinn Horng, Tzong-Wann Kao, Tsung-Lieh Lin, Cheng-Ling Lee, Takao Terano, Yi Pan. An efficient flow-shop scheduling algorithm, based on a hybrid particle swarm optimization model. Expert Systems with Applications, 2009 36:7027–7032.

do

Forward Pass:

for each forager bee θ

Find a new foraging route and

evaluate the profitability of every partial route during the foraging trip.

next for

Backward Pass:

for each forager bee θ

Map a complete foraging route to a real solution and

do the role transform in the hive.

next for

while the termination criterion is not met.

for each forager bee θ

if bee scout *then*

Find a new foraging route randomly and

evaluate the profitability of every partial route using ITBF.

elseif bee dancer *then*

Find a new foraging route after the node of idle - time > 0 and

evaluate the profitability of every partial route using ITBF.

elseif bee follower *then*

Follow the foraging route of dancer before the node of idle - time > 0 and

evaluate the profitability of every partial route using ITBF.

endif

next for

for each scheduled operation *op* of bee θ

if $Pf_{\theta} < Pf_{colony}$ *then*

$C^{\theta} = \infty$

Stop this foraging trip and back to the beehive.

endif

next for

for each forager bee θ back to the beehive

if $C^{\theta} <= C^{dancer}$ *then*

bee $\theta \rightarrow$ bee *dancer*

elseif $C^{\theta} > C^{follower}$ *then*

bee $\theta \rightarrow$ bee *follower*

else

bee $\theta \rightarrow$ bee *scout*

endif *endif*

next for

mP - ASG algorithm : Generate a schedule between non - delay and active.

Initialize S $\leftarrow \phi$, Ω is initialized to contain all operations.

repeat

Determine $s^* \leftarrow \min_{o_{ij} \in \Omega}\{s_{ij}\}$.

Determine $f^* \leftarrow \min_{o_{ij} \in \Omega}\{f_{ij}\}$.

$delay \leftarrow (f^* - s^*) \times delayweight$

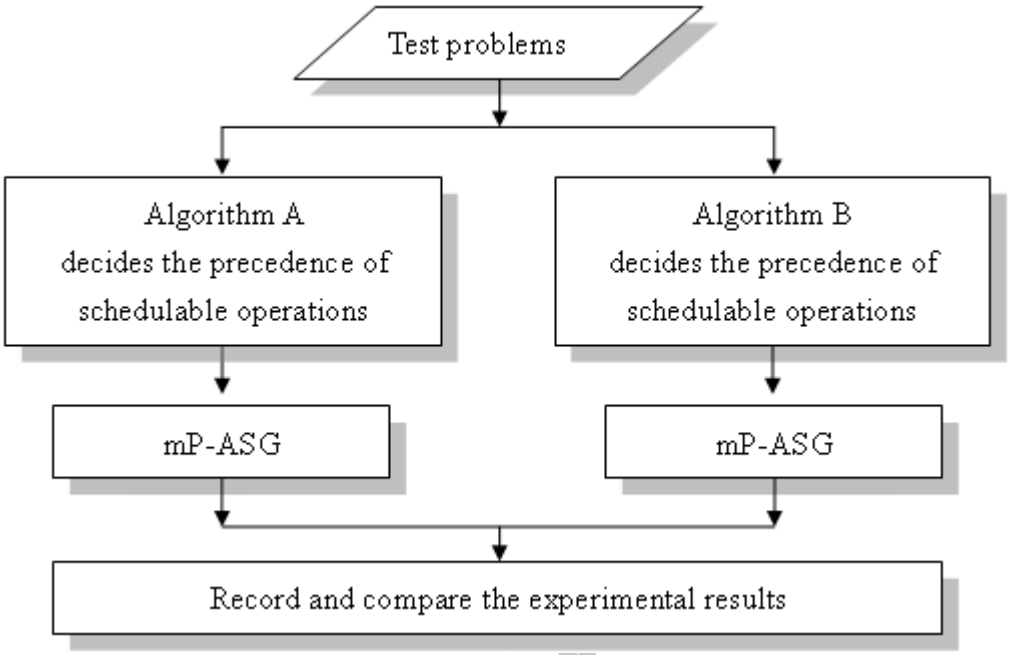
Identify the operation set O $\leftarrow \{o_{ij} \mid s_{ij} < s^* + delay, o_{ij} \in \Omega\}$.

Choose o_{ij}^* from the operation set O with the highest priority,

add o_{ij}^* to S, and assign s_{ij} as the starting time of o_{ij}^*.

Delete o_{ij}^* from Ω.

Until Ω is empty.



PSO-mP-ASG

Problem	10sec	20sec	30sec	40sec	50sec
tai_20 x 20_1	90	150	240	330	390
tai_20 x 20_2	90	150	240	330	390
tai_20 x 20_3	90	150	240	330	390
tai_20 x 20_4	90	150	240	330	390
tai_20 x 20_5	90	150	240	330	390
tai_20 x 20_6	90	150	240	330	390
tai_20 x 20_7	90	150	240	330	390
tai_20 x 20_8	90	150	240	330	390
tai_20 x 20_9	90	150	240	330	390
tai_20 x 20_10	90	150	240	330	390

BCO-mP-ASG (No ITBF)

Problem	10sec	20sec	30sec	40sec	50sec
tai_20 x 20_1	90	150	240	330	390
tai_20 x 20_2	90	150	240	330	390
tai_20 x 20_3	90	150	240	330	390
tai_20 x 20_4	90	150	240	330	390
tai_20 x 20_5	90	150	240	330	390
tai_20 x 20_6	90	150	240	330	390
tai_20 x 20_7	90	150	240	330	390
tai_20 x 20_8	90	150	240	330	390
tai_20 x 20_9	90	150	240	330	390
tai_20 x 20_10	90	150	240	330	390

		PSO-mP-ASG				BCO-mP-ASG (No ITBF)			
Problem	BKS	Best	Average	T	t	Best	Average	T	t
tai_20 x 20_1	1155	1171	1305.43	30.00	0.124	1181	1292.67	30.00	4.202
tai_20 x 20_2	1241	1271	1387.3	30.00	12.66	1260	1381.93	30.00	12.28
tai_20 x 20_3	1257	1266	1363.2	30.00	0.406	1267	1362.13	30.00	4.452
tai_20 x 20_4	1248	1251	1388.53	30.00	0.125	1263	1380.4	30.00	20.48
tai_20 x 20_5	1256	1269	1361.6	30.00	20.3	1265	1358.23	30.00	12.05
tai_20 x 20_6	1204	1211	1317.5	30.00	12.37	1215	1312.37	30.00	18.67
tai_20 x 20_7	1294	1315	1427.3	30.00	28.52	1315	1425.7	30.00	20
tai_20 x 20_8	1169	1199	1330.7	30.00	24.91	1195	1318.23	30.00	0.124
tai_20 x 20_9	1289	1310	1411.77	30.00	12.75	1303	1396.87	30.00	4.437
tai_20 x 20_10	1241	1253	1365.3	30.00	4.499	1255	1359.03	30.00	20.5
Average time				30.00	11.67			30.00	11.72
Average gap		1.32%	10.60%			1.35%	10.02%		

BCO-mP-ASG (With ITBF)

Problem	10sec	S&B	DCS	20sec	S&B	DCS	30sec	S&B	DCS
tai_20 x 20_1	450	437	13	660	637	23	1200	1158	42
tai_20 x 20_2	360	347	13	840	816	24	1020	975	45
tai_20 x 20_3	330	322	8	1050	1036	14	1200	1185	15
tai_20 x 20_4	420	398	22	810	771	39	1890	1858	32
tai_20 x 20_5	480	472	8	630	610	20	1500	1472	28
tai_20 x 20_6	390	381	9	990	977	13	1560	1544	16
tai_20 x 20_7	480	475	5	900	895	5	1530	1525	5
tai_20 x 20_8	330	314	16	840	804	36	1380	1316	64
tai_20 x 20_9	300	287	13	990	959	31	1470	1427	43
tai_20 x 20_10	480	477	3	660	652	8	1440	1435	5

※ S&B = The number of times of Stop halfway and Back to the beehive.

※ DCS = The number of times of Doing a Complete Search.

		PSO-mP-ASG				BCO-mP-ASG			
Problem	BKS	Best	Average	T	t	Best	Average	T	t
tai_20 x 20_1	1155	1165	1274.50	55.43	41.90	1163	1184.00	11.92	6.25
tai_20 x 20_2	1241	1254	1378.65	54.07	13.45	1256	1268.33	15.04	8.84
tai_20 x 20_3	1257	1261	1356.30	52.10	44.64	1264	1273.00	9.34	1.88
tai_20 x 20_4	1248	1251	1384.05	51.36	20.29	1248	1262.67	11.01	7.22
tai_20 x 20_5	1256	1259	1349.30	51.55	7.87	1258	1271.00	10.62	7.07
tai_20 x 20_6	1204	1205	1304.25	51.67	33.11	1211	1213.00	12.69	1.21
tai_20 x 20_7	1294	1301	1407.65	54.60	38.17	1308	1338.50	10.35	0.13
tai_20 x 20_8	1169	1186	1312.20	51.90	46.93	1185	1201.67	17.47	13.86
tai_20 x 20_9	1289	1290	1389.00	53.75	30.14	1295	1301.50	12.03	7.66
tai_20 x 20_10	1241	1243	1349.80	52.11	20.87	1246	1246.00	9.90	4.76
Average time				52.86	29.74			12.04	5.89
Average gap		0.50%	9.36%			0.65%	1.67%