Journal of Industrial and Systems Engineering
Vol. 1, No. 4, pp 345-359
Winter 2008

JISE

Generalized Cyclic Open Shop Scheduling and a Hybrid Algorithm

Mohammad Modarres[1*], Mahsa Ghandehari

Industrial Engineering Department, Sharif University of Technology, Tehran, Iran

ABSTRACT

In this paper, we first introduce a generalized version of open shop scheduling (OSS), called *generalized cyclic open shop scheduling* (GCOSS) and then develop a hybrid method of metaheuristic to solve this problem. Open shop scheduling is concerned with processing n jobs on m machines, where each job has exactly m operations and operation i of each job has to be processed on machine i. However, in our proposed model of GCOSS, processing each operation needs more than one machine (or other resources) simultaneously. Furthermore, the schedule is repeated more than once. It is known that OSS is NP-hard. Therefore, for obtaining a good solution for GCOSS, which is obviously NP-hard, a hybrid algorithm is also developed. This method is constructed by hybridizing ant colony optimization (ACO), beam search and linear programming (LP). To verify the accuracy of the method, we also compare the results of this algorithm with the optimal solution for some special problems.

Key words: Open shop scheduling; Cyclic open shop scheduling; Metaheuristic; ACO, Beam search.

1. INTRODUCTION

Open shop scheduling problem has a wide range of applications in many industries (Liaw, 2003). However, the assumption that each job has exactly m operations and its i^{th} operation has to be processed on machine i does not hold for many real-world problems. Therefore, in order to make it more realistic, we introduce a new version of OSS by relaxing these restrictive assumptions. In this version of generalized cyclic open shop scheduling (GCOSS), it is assumed each operation needs more than one machine or resources (such as technician or special tools) simultaneously.

OSS problems (and consequently GCOSS) are shown to be NP-Hard (Garey and Johnson, 1979). As a matter of fact, the structure of OSS is larger than any other typical scheduling problem. Thus, in this paper we develop an algorithm based on hybridizing ACO, linear programming and Beam search.

To our knowledge, a similar approach to model cyclic open shop scheduling problems, even for a single period problem, has not been applied previously in the literature, although the circular coloring concept has been applied in other scheduling problems. The only paper in the literature, regarding cyclic open shop scheduling is by Kubale and Nadolski (2005). They showed this problem is NP-hard for a 3-processor system but it is polynomially solvable for a 2-processor one.

* Corresponding Author

They proved if a given open shop can be scheduled in 3 time units then it is polynomially solvable. They also proved that in a compact open shop, the problem of minimizing $C_{\max}$ for 2-processor to determine the existence of a legal schedule is also NP-hard.

Our paper is organized as follows. Section 2 presents the definition and the structure of the problem. In section 3, we describe how to apply the concept of graph circular coloring to formulate this problem. In section 4, a metaheuristic algorithm is developed to tackle this problem. We provide an experimental evaluation in section 5.

2. GENERALIZED CYCLIC OPEN SHOP SCHEDULING PROBLEM

The generalized cyclic open shop scheduling (GCOSS) problem is defined as follows. Consider a manufacturing system in which a number of jobs must be processed. Each job consists of several operations. Each operation must be processed without preemption, *i.e.* the processing of an operation cannot be interrupted. Like OSS, operations in each job can be processed in any order. Each operation needs to be processed on a number of machines (or needs a set of resources) simultaneously. The processing time of each operation is a given specified time. Each machine can process at most one operation at a time. Similarly, each resource cannot be used by more than one operation at a time. Operations belonging to the same job cannot be processed simultaneously. We use the following notation to formulate this problem.

$M = \{M_1, M_2, ..., M_m\}$	the set of different machines (or dedicated resources);
$J = \{J_1, J_2, ..., J_n\}$	the set of different jobs to be processed;
n_j	the number of different operations of job j;
$O_j = \{o_{j1}, o_{j2}, ..., o_{jn_j}\}$	the set of operations of job j;
m_{ji}	the set of machines (or resources) that operation o_{ji} needs for processing simultaneously;
O	the set of all operations ;
$\|O\| = \sum_{j \in J} \|O_j\|$	the total number of operations.

Each permutation of operations represents a solution to this problem. The search space can easily be defined as the set of all permutations of all operations. Function $P : O \to IR^+$ assigns the processing time to operations.

There are several criteria to measure the cost of a solution. In this paper, we deal with makespan minimization while the process can be repeated more than once. In fact, we search for a solution where the makespan is minimized in a cyclic version of the problem. Adopting the other objective functions does not change the structure of the model.

Example 2.1: in this example we represent an instance of the GCOSS with four jobs of $J = \{J_1, J_2, J_3, J_4\}$ and the operations of $O = \{o_{11}, o_{12}, o_{21}, o_{22}, o_{31}, o_{41}, o_{42}\}$. There are three different machines M_1, M_2, M_3 ,and three operators $M_4 M_5, M_6$. Thus, the set of resources is $M = \{M_1, M_2, M_3, M_4, M_5, M_6\}$.

Job J_1 consists of operations o_{11} and o_{12}, or by our notation, $O_1 = \{o_{11}, o_{12}\}$. Similarly, $O_2 = \{o_{21}, o_{22}\}$, $O_3 = \{o_{31}\}$ and $O_4 = \{o_{41}, o_{42}\}$. Operation o_{11} must be processed on machine M_1 under supervising operator M_4, or by our notation, $m_{11} = \{M_1, M_4\}$. Similarly,
$m_{12} = \{M_2, M_5\}, m =_{21} \{M_2, M_5\}, m_{22} = \{M_3, M_6\}, m_{31} = \{M_3, M_4\},$
$m_{41} = \{M_2, M_5\}, m_{42} = \{M_1, M_6\}$

Table 2.1. Processing times of operations, p_{ij}

j \ *i*	1	2
1	1	1
2	1	1
3	1.5	--
4	0.5	0.5

3. GRAPHICAL MODEL OF THE PROBLEM

An open shop scheduling (as well as a generalized cyclic open shop scheduling problem) can be represented by a graph.

Definition 3.1*:* A pair of operations is called *related* (or incompatible) if either both belong to the same job or if they need at least a common machine (or resource) to be processed. Let R_{ji} represent the set of all operations related to o_{ji}.

Graphical model of OSS. An OSS problem can be modeled as a bipartite graph. In this graph, each edge represents an operation with a weight of p_{ji} which is equal to the processing time of operation j on machine i.

Graphical model of GCOSS. Since in GCOSS more than one machine is needed to process an operation, this problem cannot be modeled as a bipartite graph anymore. Thus, this problem is modeled as a weighted graph $G = (V, E)$, where V and E represent the set of nodes and edges, respectively. In this graph, an operation, say o_{ji}, is represented as a node of this graph, while the weight of this node is equal to the processing time of this operation and obviously $|V| = |O|$. By definition 1, there exists an edge between a pair of nodes, if and only if the corresponding operations are related.

It can be shown easily that the chromatic number of this weighted graph is equal to the minimal makespan of the schedule.

If the production is repeated, in some cases the makespan can be improved by making the schedule compact. This is called cyclic schedule. In this case, it can be shown that the minimal makespan of the schedule is equal to the circular chromatic number of weighted graph G. The circular chromatic number of weighted graph is defined as follows.

Definition 3.2. Consider a weighted graph of (G,w) with vertex set V and edge set E, where $w:V \to \square^{+}$ are weights of vertices. Then, the *r-circular* coloring of this graph is a mapping Γ of V to open arc of an Euclidean circle C of length r, such that:

i) $\Gamma(x)$ and $\Gamma(y)$ are disjoint if $(x,y) \in E(G)$.
ii) the length of $\Gamma(x)$ is at least $w(x)$ for all vertices $x \in V$.

The circular-chromatic number $\chi_c(G,w)$ of weighted graph (G,w) is $\chi_c(G,w)$ = Inf{r : there is an *r circular* coloring of (G,w) }.

If the weight function takes constant value 1, then $\chi_c(G) = \chi_c(G,w)$, (Zhu, 1992) . It is proved that $\chi_c(G,w)$ is always attainable (Deuber and Zhu, 1997).

We use the concept of graph circular coloring to formulate GCOSS. The following mathematical model represents this problem, where $t_{ji}, \forall i, j$ are decision variables indicating the starting time of operation o_{ji} and r is the makespan.

In subsequent section, a linear model which is the relaxed version of this formulation is applied to construct our proposed algorithm of Beam-ACO.

$$
\begin{aligned}
Min \quad & Z = r & & \\
& r \geq t_{ji} & & \forall i, j \\
& t_{ji} - t_{j\acute{i}} \geq p_{ji} - M.y_{jj\acute{i}i'} & & \forall o_{ji} \in R_{j\acute{i}} \\
& t_{ji} - t_{j\acute{i}} \geq r - p_{j\acute{i}} + M.y_{jj\acute{i}i'} & & \forall o_{ji} \in R_{j\acute{i}} \\
& t_{ji} \geq t_{j\acute{i}} - M.y_{jij\acute{i}} & & \forall o_{ji} \in R_{j\acute{i}} \\
& y_{jij\acute{i}} + y_{j\acute{i}ji} = 1 & & \forall o_{ji} \in R_{j\acute{i}} \\
& y_{jij\acute{i}} \in \{0,1\} & & \forall j, i, j', i'
\end{aligned}
$$

Where *M* represents a large number.

The graph of Figure 2a corresponds to the GCOSS problem of Example 2.1. Figure 2b and Figure 2c represent a single cycle and periodic schedule of this problem, respectively.

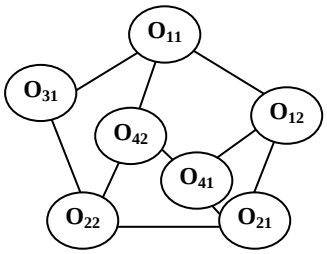

Figure 2a. Graphical representation of Example 2.1.

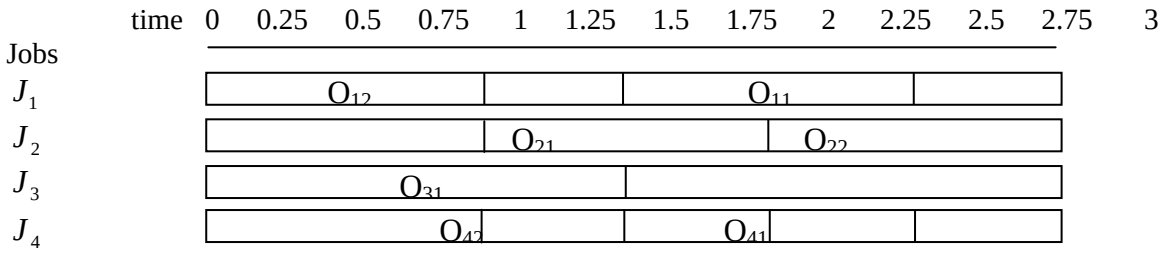

Figure 2b. Single cycle schedule of of Example 2.1.

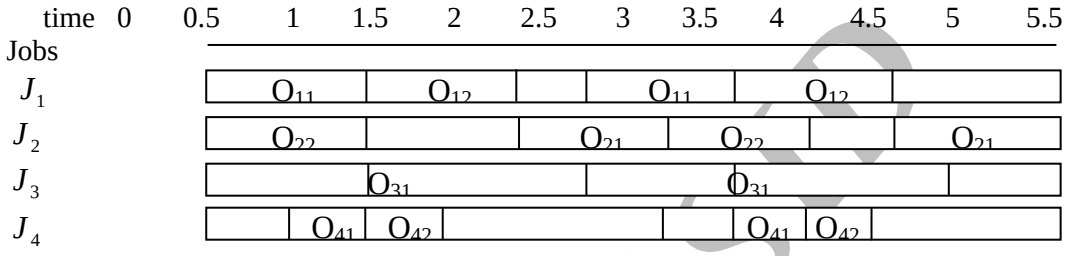

Figure 2c. Periodic schedule of of Example 2.1.

4. HYBRID ALGORITHM

In this section, we introduce an algorithm developed for obtaining a good (near optimal) solution of GCOSS problems. This algorithm is constructed by hybridizing ant colony optimization (ACO), Beam search and a local search technique. Both ACO and beam search belong to the category of constructive methods which obtain a complete solution by generating a sequence of partial solutions. A partial solution is defined as a subset of a solution in which only some variables are fixed, but not all. After assigning the value for all variables, the solution is called a complete one. In fact, rather than moving from a solution to another one, in ACO (and also in search beam), the number of assigned variables is increased in subsequent iterations (Blum, 2005). At each step, partial solutions are extended by adding a component from an allowable set. This procedure is ended when either a partial solution can not be extended any more or become a complete feasible solution.

Before presenting the proposed algorithm, we review ant colony optimization (ACO) and beam search briefly.

4.1. Ant colony optimization (ACO)

Ant colony optimization (ACO) is a metaheuristic to tackle hard combinatorial optimization problems. It was first proposed in the early 1990s (Dorigo and Stutzle, 2004. This algorithm is constructed on the basis of foraging behavior of real ants. In this algorithm, the solutions are constructed based on a probabilistic model, called pheromone model. Pheromone model consists of a set of parameters τ, called pheromone trail. The value of pheromone parameter is usually associated with the set of partial solutions obtained up to now.

At each step, a number of ants, say n_a, construct n_a partial solutions probabilistically. Some of the constructed solutions are used to update the pheromone values. These values along with another

parameter, called heuristic information, are used to determine the transition probabilities $P(c \mid \tau, \eta), \forall c \in N(s^p)$, where s_t^p is the current partial solutions at step t and $N(s_t^p)$ is the set of neighborhood of s_t^p, (as will be defined later). Transition probabilities are used to select the next component of partial solution at each step of solution construction. A quantity named convergence factor is also computed whenever the pheromones are updated. This factor takes value 1 if the ants determine a common trail.

Actually the pheromone model leads ants to generate high quality solutions over time. Heuristic information is a weighting function $\eta : N(s_t^p) \rightarrow IR^+$ and assigns a weight to each possible extension of current partial solution s_t^p. These weights reflect the benefit of extending a partial solution. The algorithm terminates, if the stopping conditions such as a maximum CPU time hold. For more information, the reader is referred to Dorigo and Stutzle, 2004.

4. 2. Beam search

Beam search (BS) is a classical tree search that was introduced in the context of scheduling and since then has been applied to many other combinatorial optimization problems successfully. This technique is an incomplete derivative of branch and bound algorithm (Ow and Morton, 1998).

The central idea behind BS is to extend the partial solution in several possible ways. At each step, a partial solution from beam set B is extended at most in k_{ext} possible ways. Each newly obtained partial solution is either a complete solution and is stored in the set of complete solution B_C, or again a partial solution and is stored in the set of further extensible partial solution B_{ext}. At the end of each step, when all members of B are extended, up to k_{bw} (called beam width) solutions are selected from the set of B_{ext}, based on the value of their lower bounds, provided B_{ext} is not empty. Actually, for each partial solution, a lower bound is computed for the objective function of all complete solutions which are constructed from this partial solution.

At the first step, the set beam B consists of only one partial solution, say $s^p =< o_{11} >$. The reason is that in a cyclic scheduling, the order of operations is not important. Thus, one operation is selected arbitrarily to build a schedule. This algorithm terminates when set B_{ext} is empty.

One major difference between ACO and beam search is that in ACO the set of partial solutions is constructed probabilistically, while beam search is a method that solutions are constructed deterministically. We use a hybridized method to take advantage of the benefit of combining both ways of exploring the search space.

4. 3. The algorithm

Ant colony optimization is the basic framework of the proposed algorithm. In standard ACO a group of ants generate the partial solutions probabilistically by considering two guides, the pheromone values and the heuristic information. However, in this method by adopting beam search rules, a component is selected deterministically to append the solution. In fact, the probabilistic choice of another component to append a partial solution in ACO is replaced by the deterministic choice of beam search.

The best complete solutions are stored in two sets, s_{bs} and S_{best} representing the best solutions so far and the best solutions after the previous restart, respectively.

The main algorithm is called Beam-ACO (or simply Algorithm 1). However, another supporting algorithm, called Beam-ACO-Construction (or simply Algorithm 2), constructs the structure of the partial solution. Actually, it represents the steps of this probabilistic beam search.

Some components of Algorithm 1 are outlined in more details as follows.

Initialize pheromone values (τ): all pheromone values are initialized to 0.5, at the start of algorithm.

Beam-ACO-GCOSS solution construction (τ). From our notation, the set of all operations is represented by O. Let the current partial solution be s_t^p. Then, O is partitioned into two sets of O_t^- and O_t^+, where $O_t^- = \{o_{ji} \mid o_{ji} \in s_t^p\}$ and $O_t^+ = O \mid O_t^-$. Actually, O_t^- represents the set of operations which are already fixed in the sequence of this partial solution and O_t^+ is the set of operations which are still free and can be appended

If $O_t^+ \cap R_{ji} = \varnothing$ for an operation $o_{ji} \in O_t^+$, then o_{ji} is not related to any other remaining operations. In that case, the position of this operation in the sequence is not important. Actually, this partial solution can be appended with the remaining operations in any arbitrary sequence.

Therefore, we define the set of allowed operations (neighborhood of s_t^p) as follows:

$$N(s_t^p) \leftarrow \{o_{ji} \mid o_{ji} \in O_t^+, R_{ji} \cap O_t^+ \neq \varnothing\}$$

As mentioned before, the next operation in ACO is selected probabilistically. This is done by computing transition probabilities. Thus, it has to determine pheromone values and the weighting function that is called heuristic information. The pheromone values are computed by Algorithm 2. However, the weighting function (heuristic information) is defined as:

$$\eta(o_{ji}) \leftarrow \frac{\dfrac{1}{t_{es}(o_{ji}, s_t^p)+1}}{\displaystyle\sum_{o_{lk} \in N(s_t^p)} \frac{1}{t_{es}(o_{lk}, s_t^p)+1}} \qquad \forall o_{ji} \in N(s_t^p)$$

Where $t_{es}(o_{ji}, s_t^p)$ is the earliest starting time of o_{ji} with respect to the partial solution s_t^p. At the beginning, s_t^p has no operation. Then, the earliest starting of $N(s_t^p)$ is 0.

The transition the probability of selecting $o_{ji} \in N(s^p)$ is defined as follows.

$$p(o_{ji} \mid \tau, \eta) \leftarrow \frac{(\min_{o_{ji'} \in R_{ji} \cap O_t^+} \tau_{jij'i'})^{\alpha} \eta(o_{ji})}{\displaystyle\sum_{o_{lk} \in N(s_t^P)} (\min_{o_{ji'} \in R_{lk} \cap O_t^+} \tau_{lkji'})^{\alpha} \eta(o_{lk})} \qquad \forall o_{ji} \in N(s^p)$$

Clearly, if the minimum of $\tau_{jij'i'}$ is lower than that of o_{lk}, then $p(o_{ji} \mid \tau, \eta) \leq p(o_{lk} \tau, \eta)$. It means that there is at least one operation left that probably should be scheduled before $c = o_{ji}$.

Let α be the adjusting factor which indicates the importance of pheromone values with respect to the heuristic information. Ø

4. 4. Algorithm 1, Beam-ACO for GCOSS problem

Input: an GCOSS problem instance
$s_{bs} \leftarrow <\ >$, $S_{best} \leftarrow \varnothing, cf \leftarrow 0$
Initialize pheromone values (τ)
while termination condition not satisfied do
$S_{iter} \leftarrow \varnothing$
for $j = 1$ to n_a
$S_{iter} \leftarrow S_{iter} \cup$ Beam-ACO solution construction (τ)
end for
Apply local search (S_{iter}) (optional step)
Find the set of obtained best solutions (S_{iter})
Apply pheromone update (τ, S_{best}, s_{bs})
$cf \leftarrow$ compute convergence factor ()
if $cf > cf\_\lim it$ then
Reset pheromone values (τ)
$S_{best} \leftarrow \varnothing$
end if
end while
output: s_{bs}, a set of best solutions S_{best} found

4. 5. Algorithm 2. Beam-ACO-Construct for structure of GCOSS solution.

Some details of Algorithm 2 are presented as follows.

Since our scheduling is a cyclic one, the first operation can be selected arbitrarily. So $s_1^p =< o_1 >$.
Reduce to related $(N(s_t^p), o_{ji})$: this procedure is used to restrict the neighborhood set, after the first extension of partial solution s_t^p was performed. This is done as follows:

$$N(s_t^p) = \{o_{j'i'} \in N(s_t^p) \mid o_{j'i'} \in R_{ji}\}$$

This effective procedure prevents the construction of permutations that have no effect on the final solution.

Rank the partial solution s_t^p *using a lower bound LB(.)*: we compute a lower bound from the definition of $t_{es}(o_{ji}, s_t^p)$.

As we said $t_{es}(o_{ji}, s_t^p)$ is the earliest starting time of operation o_{ji} with respect to the partial solution s_t^p. Thus, the earliest completion time $t_{es}(o_{ji}, s_t^p)$ of operation o_{ji} is $t_{es}(o_{ji}, s_t^p) + p_{ji}$. The approximate earliest completing time is constructed by extending s_t^p as follows:

$$LB(.) = \max\{X, Y\}$$

$$X = \max_{o_{ji} \in s_t^p} \{tec(o_{ji}, s_t^p) + \sum_{o \in R_{tji}^+} p(o)\}, Y = \max_{o_{ji} \in O_t^-} \{\sum_{o \in R_{tji}^+} p(o)\}$$

Similar to the partition of O into O_t^- and O_t^+, R_{ji} is also partitioned into two subsets of R_{tji}^+ and R_{tji}^-.

Input data for k_{bw} *and* k_{ext} *:* $k_{bw} = \max\{1, |O|/10\}$ and $k_{ext} = \{1, |N(s_t^p)|/2\}$.

The set of best solutions (S_{iter})*:* The objective value of each solution in S_{iter} must be determined and then a number of the best solutions equal to $\max\{\frac{|S_{iter}|}{10}, 1\}$ are selected. However, To determine the objective value (makespan) of a solution $< o^1, o^2, ..., o^{|O|} >$ where $o^k \in O, t^k \geq t^{k'}$ *if* $k \geq k', o^k \in R^{k'}$, we use the following linear programming:

$$\begin{aligned} &Min \quad Z = r \\ &r \geq t^k && \forall k \\ &t^k - t^{k'} \geq p^k && \forall k, k'; o^k \in R^{k'} \\ &t^k - t^{k'} \geq r - p^k && \forall k, k'; o^k \in R^{k'} \end{aligned}$$

Updating pheromone τ : $\tau_{jij'i'} \leftarrow \tau_{jij'i'} + \rho(\frac{\sum_{s \in S_\tau} (\delta(o_{ji}, o_{j'i'}, s) - \tau_{jij'i'})}{|S_\tau|}$, where $\delta(o_{ji}, o_{j'i'}, s) = 1$ if o_{ji} is scheduled before $o_{j'i'}$ and $\delta(o_{ji}, o_{j'i'}, s) = o$, otherwise. $\rho \in [0,1]$ is a constant called evaporation rate.

Compute convergence factor: This numerical factor controls the algorithm. This factor is denoted as $cf \in [0,1]$ and is computed as follows:

$$cf \leftarrow 2(\frac{\sum_{o_{ji} \in O} \sum_{o_{j'i'} \in R_i} \{\tau_{\max} - \tau_{jij'i'}, \tau_{ijij'i'} - \tau_{\min}\}}{|\tau|(\tau_{\max} - \tau_{\min})} - 0.5)$$

We set the limit of convergence factor, $cf\_\lim it$ as 0.9. The initial value of cf is set 0.

Upper bound $\tau_{\max}$ *and lower bound* $\tau_{\min}$: to prevent the algorithm from converging to a local solution, as proposed by Stutzle and Hoos (2000), we set 0.01 and 0.09 as the lower and the upper bound of τ. After applying pheromone update rule, we check to make sure the pheromone remains within the range of $[\tau_{\min}, \tau_{\max}]$. When the pheromone values take their limited bound, then $cf = 1$.

The steps of Algorithm 2 are as follows.

Input: partial solution $s_1^p =< o_1 >$, beam width k_{bw}, maximum number of extensions k_{ext}

$B \leftarrow \{s_1^p\}, B_C \leftarrow, t \leftarrow 1$
while $B \neq$ do
  $B_{ext} \leftarrow \varnothing$
  for $s_t^p \in B$ do
    $count \leftarrow 1$
    $N(s_t^p) \leftarrow \text{PreSelect}(N(s_t^p))$
    while $count \leq k_{ext}$ AND $N(s_t^p) \neq$ do
      choose $o_{ji} \in N(s_t^p)$ with transition probability $p(o_{ji} \mid \tau, \eta)$
      $s_{t+1}^p \leftarrow$ extend s_t^p by appending operation o_{ji}
      $N(s_t^p) \leftarrow N(s_t^p) \mid \{o_{ji}\}$
      if $N(s_t^p) \neq \varnothing$ then
        $B_{ext} \leftarrow B_{ext} \cup \{s_{t+1}^p\}$
      else
        $B_c \leftarrow B_c \cup \{s_{t+1}^p\}$
      end if
      if $count = 1$ then
        $N(s_t^p) \leftarrow \text{Reduce To Related}(N(s_t^p), o_{ji})$
      end if
      $count \leftarrow count + 1$
    end while
  end for
$B \leftarrow$ select the $\{k_{bw}, \mid B_{ext}\}$ highest ranked partial solutions from B_{ext}
$k \leftarrow k + 1$
end while
output: a set of feasible solutions B_{ext}

In fact, in Algorithm 2 each ant constructs a set of solutions B_C. But in ACO algorithm each ant constructs only one solution not a set of different solutions.
Apply local search (S_{iter}): we can apply a suitable local search procedure to every solutions $s \in S_{iter}$.

5. EXPERIMENTAL RESULTS

To illustrate our proposed method, we present a number of GCOSS instances and obtain the solution of these problems by applying Beam-ACO-GCOSS algorithm in this section. Two sets of problems are solved by this algorithm. The first one is presented by random generation of parameters. Some solutions of this set of problems are compared with the corresponding optimal solutions, obtained analytically. The second set consists of the problems with a certain amount of optimal objective function, as we discuss it later. In this case, we set the value of α equal to 1.

For the first set of GCOSS instances, we propose the problems with 3, 4, 5, 6, 7, 8, 9, 10, 15 machines and from 4 to 30 jobs. Job j has n_j non-preemptive operations, while an operation, say o_{ji} has to be performed on $|m_{ji}|$ machines (or use some other resources). Any processing order of jobs is allowed and the order of the operations of every job is totally free.

Given n (the number of jobs) and m (the number of machines), the other parameters of the problems are generated randomly, drawn from discrete uniform distributions as follows.

$|O_j|$, the number of operations of job $j : U[1, \lfloor m/2 \rfloor]$, where $\lfloor x \rfloor = \max\{\text{integer} \le x\}$

$|m_{ji}|$, the required number of machines (or other resources) to perform operation, o_{ji}: $U[1, \max\{\lfloor m/2 \rfloor, 1\}]$

m_{ji}, the set of machines related to operation $o_{ji} : \{M_k \mid M_k \in M\}$

$k : U[1, m]$

p_{ij}, the processing time of operation: $U[1,99]$

We use Matlab7.1 for programming Beam-ACO-GCOSS algorithm. For generating random numbers we call, state, one of the generators using by Matlab7.1 and set "*rand*" to specified state, S, for each problem. Table 5.1 represents the generated instances as well as their solutions.

S: the number of initialized string (represented by its number of jobs, number of machines and number of initialized random string);
No jobs: The number of jobs;
No machines: The number of machines;
No operations: The total number of operations;
OV: value of objective function obtained by Beam-ACO-GCOSS;
LB: The number of vertices of maximum complete subgraph of (G, w);
OP: The optimum solution obtained by an exact method.

At first, the numbers of operations for each job is generated. Then, the number of machines to execute each operation are determined and at last the machine numbers are generated. In these instances all processing times are equal to 1.

The optimality of the solutions are checked either by GAMS or by (k/d) property. If an example is solvable by GAMS, then the objective value of the model is marked by OV*. For the second set of generated problems, the obtained solutions are compared with the exact optimal solution (k/d). From Table 5-1, the solutions obtained from the proposed algorithm resulting in the optimal value (indicated by OV *) are distinguished from the ones with a lower bound.

In Table 5.1, OV marked by * indicates the optimal value of the objective function has been reached. The number of operations of the instances is up to 60.

We executed the program for at most 20 iterations where in many instances the convergence factor didn't exceed its limit and collected the information. For the problems with smaller number of operations, the objective value of the jobs is equal to LB, as Table 5.1 shows. For larger size instances, the results are good enough by considering the fact that only 20 iterations were executed. The second set of instances are the ones that have special structure and their objective function are equal to k/d, where k and d are two positive integers, from the following definition:

Definition 5.1. For two integers $1 \le d \le k$, a (k,d) coloring of a graph G is a coloring of the vertices of G with colors $\{0,1,...,k-1\}$ such that,

$$(x,y) \in E(G) \Rightarrow d \le |c(x) - c(y)| \le k - d$$

The circular chromatic number is defined as

$\chi_c(G) = \inf\{k/d : G$ has a (k,d) coloring$\}$

Let an integer from set $\{0,1,...,k-1\}$ be assigned to each vertex of the set of vertices V and also an edge between two vertices of o_i and o_j be connected. If $d \le |I(o_i) - I(o_j)| \le k - d$ where $I : V \to \{0,1,...,k-1\}$, then $|O|$ is the total number of operations, $V = O = \{o_1,...,o_{|O|}\}$.

Let the number of operations of a job be n. For $k \le n$, we assign an integer which is $i-1$ to each operation o_i of $\{o_1,...,o_k\}$. The remaining operations are assigned randomly. Thus, the vertices of this graph are marked by the colors of $\{0,1,...,k-1\}$. The edges are assigned as described by Definition 5.1.

We examine the efficiency of our algorithm by solving these instances. Each problem is solved 10 times. The average number of iterations to reach the optimal solution is shown in Table 5.2.

6. CONCLUSIONS

In this paper we have introduced a generalized cyclic open shop scheduling (GCOSS) problem which is a NP-hard problem. To solve this problem, a hybrid algorithm of ACO, Beam search and LP was developed. We generated two types of instances of GCOSS problems and solved them by this method. It was shown that this algorithm works reasonably well by comparing the objective value of these instances to the similar values obtained by an exact method or by checking their lower values.

Table 5.1 The results of Beam-ACO-GCOSS for some instances

	LB	OP	OV	Instanc	LB	OP	OV	Instanc	LB	OP	OV
4×3_0	3	_	3*	6×7_0	7	7	8	6×9_0	9	_	9*
4×3_20	5	_	5*	6×7_20	6	_	6*	6×9_20	6	_	6*
4×3_50	3	_	3*	6×7_50	4	_	4*	6×9_50	7	_	7*
4×3_80	6	_	6*	6×7_80	6	_	6*	6×9_80	10	_	10*
4×3_100	5	_	5*	6×7_100	7	7	8	6×9_100	9	9	10
3×5_0	5	_	5*	3×8_0	6	_	6*	7×9_0	11	14	14*
3×5_20	5	_	5*	3×8_20	6	_	6*	7×9_20	9	10	10*
3×5_50	3	_	3*	3×8_50	5	_	5*	7×9_50	10	10	11
3×5_80	5	_	5*	3×8_80	9	_	9*	7×9_80	10	12	12*
3×5_100	6	_	6*	3×8_100	7	_	7*	7×9_100	13	_	13*
4×5_0	4	_	4*	4×8_0	5	_	5*	8×9_0	9	11	11*
4×5_20	5	_	5*	4×8_20	6	_	6*	8×9_20	11	_	11*
4×5_50	3	_	3*	4×8_50	6	_	6*	8×9_50	10	_	10*
4×5_80	4	_	4*	4×8_80	6	_	6*	8×9_80	9	9	10
4×5_100	4	_	4*	4×8_100	6	_	6*	8×9_100	10	11	11*
3×6_0	6	_	6*	5×8_0	8	10	10*	3×10_0	8	_	8*
3×6_20	5	_	5*	5×8_20	8	_	8*	3×10_20	10	_	10*
3×6_50	4	_	4*	5×8_50	6	_	6*	3×10_50	7	_	7*
3×6_80	5	_	5*	5×8_80	7	7	9	3×10_80	10	_	10*
3×6_100	7	_	7*	5×8_100	9	10	10*	3×10_10	9	_	9*
4×6_0	5	_	5*	6×8_0	7	8	9	5×10_0	8	9	9*
4×6_20	4	_	4*	6×8_20	7	_	7*	5×10_20	8	8	9
4×6_50	3	_	3*	6×8_50	6	_	6*	5×10_50	7		8
4×6_80	5	_	5*	6×8_80	7	_	7*	5×10_80	9	_	9*
4×6_100	5	_	5*	6×8_100	9	_	9*	5×10_10	12		13
5×6_0	8	9	9*	7×8_0	11	12	12*	7×10_0	13		14
5×6_20	8	_	8*	7×8_20	7	9	9*	7×10_20	9		11
5×6_50	5	_	5*	7×8_50	8	_	8*	7×10_50	11		12
5×6_80	5	_	5*	7×8_80	11	12	12*	7×10_80	13		14
5×6_100	6	6	7	7×8_100	12	13	13*	7×10_10	13		15
3×7_0	6	_	6*	3×9_0	7	_	7*	9×10_0	14		20
3×7_20	8	_	8*	3×9_20	7	_	7*	9×10_20	12		14
3×7_50	5	_	5*	3×9_50	6	_	6*	9×10_50	10		13
3×7_80	7	_	7*	3×9_80	7	_	7*	9×10_80	12		17
3×7_100	7	_	7*	3×9_100	8	_	8*	9×10_10	16		20.971
4×7_0	6	_	6*	4×9_0	7	_	7*	3×15_0	13		14
4×7_20	6	_	6*	4×9_20	6	_	6*	3×15_20	11	_	11*
Instance	LB	OP	OV	Instance	LB	OP	OV	Instance	LB	OP	OV
4×7_50	6	_	6*	4×9_50	7	_	7*	3×15_50	12	_	12*
4×7_80	6	_	6*	4×9_80	6	_	6*	3×15_80	10	_	10*
4×7_100	6	_	6*	4×9_100	8	_	8*	3×15_10	13		14

Table 5.1 (Continued)

S	L	O	OV	Instan	L	O	OV	Instan	L	O	OV
5×7_0	7	8	9	5×9_0	8	9	9*	5×15_0	15		17
5×7_20	6	_	6*	5×9_20	7	_	7*	5×15_20	13	_	13*
5×7_50	5	_	5*	5×9_50	7	_	7*	5×15_50	12	_	12*
5×7_80	6		7	5×9_80	8	8	9	5×15_80	15		15.9805*
5×7_100	8	8	9	5×9_1	9	10	11	5×15_	18		18.9642*
8×15_0	14		19.9	3×20_	14	_	14*	3×25_	21		22
8×15_20	14		16.9	3×20_	14	_	14*	3×25_	20		21
8×15_50	12		17.9	3×20_	14		15.9	3×25_	19		21
8×15_80	14		16.9	3×20_	15	_	15*	3×25_	17	_	17*
8×15_100	15		19.9	3×20_	16	_	16*	3×25_	23		22.9673*
5×20_20	18		19	3×40_	29		31	3×30_	22		24
5×20_50	18	_	17.9	5×20_	21		24	3×30_	27		29
5×20_80	20		23.9	3×30_	33	_	33*	3×30_	22		24

Table 5.2 The average number of iterations to reach the optimal solution in 10 executions

No Operations	*k*	*d*	Average iterations
5	5	2	1
7	5	2	1
7	7	2	1
9	5	2	1
9	7	2	1.1
10	5	2	1.1
10	7	3	1.2
15	5	2	1.4
15	7	2	3.3
15	9	4	2.1
15	11	3	4.5
15	15	4	51
20	5	2	1.2
20	11	3	41.2
20	17	8	2.9
30	5	2	14

REFERENCES

[1] Blum C. (2005), Beam-ACO-Hybridizing ant colony optimization with beam search: an application to open shop scheduling; *Computers and Operations Research* 32; 1565-1591.

[2] Deuber W., Zhu X. (1997), Circular Coloring of Weighted Graphs; *Journal of Graph Theory* 23; 365-376.

[3] Dorigo M., Stutzle T. (2004), Ant colony optimization; Boston, MA: MIT Press.

[4] Garey M.R., Johnson D.S. (1979), Computers and intractability; A guide to the theory of NP-Completeness, Freeman, San Francisco.

[5] Kubale M., Nadolski A. (2005), Chromatic scheduling in a cyclic open shop; *European Journal of Operation Research*; 164(99); 585-591.

[6] Liaw C.F. (2003), An efficient tabu search approach for the two-machine preemptive open shop scheduling; *computers & Operations Research* 30, 2081-2095.

[7] Ow P.S., Morton T.E. (1988), Filtered beam search in scheduling; *International Journal of Production Research* 26; 297-307.

[8] Stutzle T, Hoos H.H. (2000), MAX-MIN ant system; *Future Generation Computer Systems* 16(8); 889-914.

[9] Zhu X. (1992), Star-chromatic numbers and products of graphs; *Journal of Graph Theory* 16; 557-569.