<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>An enhanced ant colony optimization approach for integrated process planning and scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Zhang, S; Wong, TN</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>The 2013 International Conference on Industrial Engineering and Systems Management (IESM), Rabat, Morocco, 28-30 October 2013. In Conference Proceedings, 2013, p. 599-604</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>2013</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/209625">http://hdl.handle.net/10722/209625</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>International Conference on Industrial Engineering and Systems Management (IESM). Copyright © IEEE.; ©2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.; This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.</td>
</tr>
</tbody>
</table>
Abstract—An enhanced ant colony optimization (eACO) meta-heuristics is proposed in this paper to accomplish the integrated process planning and scheduling (IPPS) in the jobshop environments. The IPPS problem is graphically formulated to implement the ACO algorithm. In accordance with the characteristics of the IPPS problem, the mechanism of eACO has been enhanced with several modifications, including quantification of convergence level, introduction of pheromone on nodes, new strategy of determining heuristic desirability and directive pheromone deposit strategy. Experiments are conducted to evaluate the approach, while makespan and CPU time are used as measurements. Encouraging results can be seen when comparing to other IPPS approaches based on evolutionary algorithms.

Keywords— integrated process planning and scheduling; job shop scheduling; ant colony optimization

1. INTRODUCTION

Process planning and scheduling are two important manufacturing functions that are traditionally performed separately in sequence. While process planning converts the design information and constraints into the detailed manufacturing process steps and instructions, scheduling is responsible for allocating the required manufacturing resources in an efficient manner. Due to the dynamic changes of the manufacturing environment, pre-defined process plan and schedule may often become sub-optimal or even infeasible in practice. Integrated process planning and scheduling (IPPS) has therefore been proposed to combine both functions together, providing an essential solution to improve optimality and feasibility. An IPPS problem is typically figured as $N$ jobs consisted of multiple operations to be accomplished by $M$ machines, with the objective that some criteria such as makespan or mean flow time are optimised.

Various types of scheduling algorithms and techniques have been established for IPPS over the years. Classical exact and analytical approaches, such as branch and bound algorithm\(^5\) and mixed integer programming\(^10\), were suggested by researchers to solve relatively simple IPPS problems. However, for those real-life large scale problems with high complexity, the required computation time and efforts go beyond a reasonable bound. Therefore, algorithms based on heuristics, such as dispatching rules\(^5\), genetic algorithm (GA)\(^9\)\(^12\), symbiotic evolutionary algorithm (SEA)\(^6\), simulated annealing (SA)\(^11\) etc., have been established for solving larger IPPS problems. Compared to exact approaches, meta-heuristics can improve the performance significantly, producing near-optimal process plans and schedules in a reasonable computational time.

With the advent of agent technology, many researchers have spent efforts on developing agent-based systems for manufacturing applications. Bidding, auction or negotiation based approaches are the most popular. Typical examples are found in [2] and [13]. In a preliminary research\(^14\), a multi-agent system (MAS) for solving the IPPS was established with the negotiation-based approach, showing that MAS architecture is a preferable approach for solving IPPS problem.

Ant colony optimization (ACO) is a widely used meta-heuristics which was originally designed by [1] for graphical optimisation problems such as travelling salesman problem (TSP). The first attempt of using ACO to solve IPPS was reported in [7], in which only alternative machine flexibility is considered. Recently, researchers tried to collaborate different meta-heuristics with MAS. The applicability of implementing ACO on MAS was verified by a preliminary research [8] with consideration of a full set of manufacturing flexibility. A pure probabilistic procedure adopted from standard ACO algorithm was applied to the graphically formulated IPPS to search for feasible solutions. However, due to the limitation of standard ACO algorithm, the approach was not that competitive with other meta-heuristics in terms of solution quality.

This paper presents an enhanced ant colony optimisation (ACO) algorithm for solving the IPPS problem. The mechanism is well tuned to cater for the characteristics of IPPS problem and implemented on a MAS structure. Makespan and CPU time are chosen as measurements of performance. The results are compared to other approaches and a significant progress could be observed.

2. ACO APPROACH FOR IPPS

To apply the ACO meta-heuristic, the IPPS problem can typically be represented by a disjunctive graph in which the alternative processes of jobs and their processing sequences are
expressed by nodes and edges respectively. The ants are assigned as software agents to travel on the graph until a feasible solution is generated.

2.1 Graphical Formulation

First, generally, for an $N$-job $M$-machine problem with $J$ operation in total, let $N$ denotes the set of parts such that $N = \{1,2,..., n,..., N\}$, $M$ denotes the set of machines where $M = \{1,2,..., m,..., M\}$ and $J$ denotes the set of operations where $J = \{1,2,..., j,..., J\}$. The problem is visualised into a disjunctive graph $D = (O, A, B)$ where $O$ is a set of nodes, $A$ is a set of directed edges and $B$ is the set of undirected edges. Each operation is denoted as $O_i$. A dummy start node and a dummy end node are added to connect with the starting and ending operations of all jobs respectively. Fig. 1 shows an example of 2-job 5-machine IPPS problem with 10 operations, among which $O_1$-$O_5$ belong to job 1, $O_6$-$O_{10}$ belong to job 2.

Nodes denote the processes, which could be uniquely specified as $O_jM_m$, indicating that operation $O_j$ be completed by machine $M_m$. Direct edges indicate the precedence relationships among the operations, while undirected edges stand for possible ant routes.

2.2 Standard ACO heuristic

As the disjunctive graph is constructed, the ACO algorithm is applied to find a route on the graph, which corresponds to a feasible solution. The operations of solution are sequentially scheduled to the chosen machine as the visited sequence in the route. The objective is to find a solution such that the performance criteria are optimised. At the beginning of each iteration, $K$ ants are assigned at the dummy start node. They freely travel through the graph following the precedence requirements, until the dummy end node is reached. A precedence requirement indicates that, the successor operation can only be performed after its predecessor(s) has been performed. An operation may have multiple predecessors. If the relationship among all its predecessors is “OR” (shown by the OR-token), then only one of the predecessors is required to be performed before the successor; otherwise the relationship is “AND”, all the predecessors are to be performed before the successors. For example, $O_3$ could be performed after either $O_5$ or $O_4$ is performed; $O_4$ can’t be performed until both $O_5$ and $O_6$ are performed. When an OR-branch is encountered by an ant, only one of the branch routes will be visited, all operations belonging to the other branch will be neglected in its further searching. For example, if an ant has chosen $O_7$ in its route, then $O_8$ will be automatically neglected. In addition, since each operation needs to be performed by one and only one machine, once it is assigned to one machine, the other alternative machines will be neglected.

Ants are guided by two factors during the travelling: pheromone amount $\tau$ and a heuristic desirability $\eta$. At each step, ant chooses the next node from all possible destinations by a probabilistic procedure. Let $u$, $v$ denote the current and destination nodes respectively, then the probability that ant moves from $u$ to $v$ is:

$$p(u,v) = \begin{cases} \frac{\tau(u,v)^\alpha \eta(u,v)^\beta}{\sum_{w \in S} \tau(u,w)^\alpha \eta(u,w)^\beta}, & \text{if } v \in S, \\ 0, & \text{otherwise.} \end{cases}$$

where $\alpha$ and $\beta$ are respective weights of pheromone and heuristic desirability; $S$ denotes the set of possible destination nodes. Both directed and undirected edges represent possible paths for ants to pass by.

The pheromone value $\tau$ is associated with paths with an initial value $\tau_0$ and updated by ant deposit and global evaporation. Each ant $k$ deposits pheromone on each path $(u,v)$ belonging to its route in an iteration $i$:

$$\Delta \tau_{jk}(u,v) = \frac{Q}{L_{jk}}$$

where $Q$ is a positive constant, $L_{jk}$ is the makespan by ant $k$ in iteration $i$. The objective is to minimize the makespan, it is obvious that the smaller the makespan is, the more pheromone will be deposited. During the whole algorithm, pheromone on all paths evaporate at a fixed rate $\rho$ ($0<\rho<1$). Therefore, the pheromone update in each iteration $i$ can be expressed by:

$$\tau_{jk}(u,v) = (1-\rho)\tau_{jk}(u,v) + \sum_{k=1}^{K} \Delta \tau_{jk}(u,v)$$

The heuristic desirability $\eta$ indicate the attractiveness of the destination node for an ant. It is usually associated with factors that may directly affect the solution quality. In the previous ACO approach, it is determined by a “greedy” strategy:
\[ \eta(u,v) = \frac{C}{i(v)} \]  

(4)

where \( C \) is a positive constant, \( i(v) \) is the processing time of the corresponding process of node \( v \). It can be expected that ants would tend to choose a node with the shortest processing time.

3. Enhanced ACO Approach

Even though the approach with standard ACO\(^8\) was able to generate feasible solution for IPPS, the performance was not satisfactory enough. The following problems were observed:

- Resulting from the greedy strategy, ants tend to choose the process with the shortest processing time, which may not necessarily minimize the makespan;
- Early convergence on non-optimal solutions and frequent reset of algorithm caused by high evaporation rate;
- The measures aimed at improving the solution quality by iteration were ineffective; in other words, solutions with different levels of quality cannot be quantitatively differentiated.

To improve the system, the following modifications are proposed to the algorithm:

3.1 Quantification of Convergence Level

Convergence is a commonly observed phenomenon for ACO heuristic, which is resulted from the accumulation of pheromone on specific routes: after running the algorithm for a long time, some routes may be more attractive than others due to their high pheromone trail. On one hand, convergence indicates the trend of being stable for the algorithm; on the other hand, it prevents ants from finding new solutions, which might further improve the result.

A variable indicating the level of convergence in the algorithm at iteration \( i \) is defined as follows:

\[ \sigma_i = \begin{cases} 0, & \text{if } i = 1, \\ \frac{|O_i \cap O_{i-1}|}{|O_{i-1}|}, & \text{else.} \end{cases} \]  

(5)

where \( O_i \) is the set of nodes chosen in the solution given by ants in iteration \( i \). This variable is the “number of nodes inherited from last iteration”-to-“number of nodes chosen in this iteration” ratio. Its value directly reflects how similar a solution is compared to the solution in the last iteration. A larger value indicates a higher level of convergence.

3.2 Introduction of Pheromone on Nodes

In traditional ACO algorithm, pheromone values are associated with paths (edges). This is because the algorithm was originally designed for TSP-like route searching problems, in which all cities (nodes) have to be visited to obtain a feasible solution. However for IPPS problem, not all nodes have to be visited due to the operation and machine flexibility. Besides determining the processing sequence, choosing an appropriate set of processes is also an important issue for generating solutions with good quality. This has always been ignored in traditional applications of ACO on IPPS.

In this approach, nodes will be assigned as pheromone carriers together with the edges. Let \( \mu(u) \) and \( \delta(u,v) \) denote the pheromone associated with node \( u \) and edge between \( u \) and \( v \) respectively. The two types of pheromone share a common initial value \( \tau_{0i} \), maximum and minimum value \( \tau_{\text{max}} \) and \( \tau_{\text{min}} \), and global evaporation manner with rate \( \rho \) as in standard ACO. They together constitute the pheromone trail \( \tau \) that affects the ants’ selection with different weights:

\[ \tau(u,v) = (1-\sigma)\mu(u) + \sigma\delta(u,v) \]  

(6)

The pheromone on nodes is the major factor in selection among alternative processes, while the pheromone on edges influence more in determining the sequence. It is reasonable that the former should be made dominant in early stage of the algorithm; as the algorithm goes on, ants gradually converge on a specific set of processes, the latter becomes more significant.

3.3 Redefinition of Heuristic Desirability

The “greedy” strategy for determining \( y \) is the major factor lowering the performance of standard ACO approach for IPPS. A more reasonable manner based on the “earliest finishing time” is used in this approach:

\[ \eta(u,v) = \begin{cases} \frac{C}{\Delta L(v)}, & \text{if } \Delta L(v) \geq 1, \\ \frac{C}{2-\Delta L(v)}, & \text{if } \Delta L(v) < 1. \end{cases} \]  

(7)

where \( C \) is a positive constant, \( \Delta L(v) \) is the increment of makespan when scheduling the corresponding process of node \( v \). In this case, ants will not simply judge the node by the processing time, but by the actual influence of the node on makespan. This is much more effective in minimizing the makespan.

3.4 Redesign of Pheromone Deposit Strategy

It has been found that the pheromone deposit strategy in the previous research \(^8\) is not capable of intellectually directing ants to favourable routes. Nor could ants discover better routes based on previous findings. Ants often quickly converge to a local optima, and the algorithm will have to be reset in a few iterations.

This approach used an elite strategy which has been proved effective in many ACO applications. With the elite strategy, only the ant which found the best solution in an iteration is allowed to deposit pheromone on its path:

\[ \tau_i(u,v) = (1-\rho)\tau_{i-1}(u,v) + \Delta \tau_{\text{elite}}(u,v) \]  

(8)

Besides, two factors are added to affect the amount of pheromone to be deposited:

- The general solution quality of the route. Note that this is not the makespan value itself, but the relative fitness of the solution compared to all the previous finding, expressed as:
Initialization
- Formulate of the IPPS problem
- Determine algorithm parameters
- Initialize algorithm variables

Iteration
- Deploy $K$ ants at dummy start node
- Each ant $k$ finds a feasible solution
- Find the best solution among $K$ solutions
- Apply pheromone deposit
- Apply global evaporation and variables update

Termination control
- Better solution found?
  - $r = 0$
  - $r = r + 1$
  - $r \geq R$?
- Output the best solution found

Output

Fig. 2. Flow chart of the enhanced ACO approach

3.7 Implementation

The approach is implemented on a fully distributed MAS architecture designed by JADE (Java Agent Development Environment). An environment manager agent, a heuristic manager agent and multiple heuristic agents are constructed. The environment manager is responsible for the recording the global variables and iteration between ants and the environment, while the heuristic manager is responsible for the algorithm functions of each run, such as initialization, iteration and termination control. The multiple heuristic agents behave as individual ants to generate solutions.

4. EXPERIMENTAL STUDY

Experiments have been conducted to illustrate the approach. The experiments are designed into two parts: one to find the most appropriate set of parameters, the other to test the performance of the enhanced ACO algorithm. The widely used 24-problem set adopted from [6] is used for all test cases, which has taken a full set of manufacturing flexibilities into consideration, including: process flexibility, sequence flexibility and alternative machine flexibility, summarized by [4]. The set contains 24 problems with different combinations of 18 jobs, which are summarized in TABLE I.

\[
A = \begin{cases} 
1, & \text{if } i = 1, \\
\frac{L_{\text{avg}} - L_{a,i}}{L_{\text{avg}} - L_{\text{best}}}, & \text{else.}
\end{cases}
\]  

(9)

where $L_{\text{avg}}$ is the average makespan since the start, $L_{\text{best}}$ is the best makespan since the start, $L_{a,i}$ is the makespan found by ant $k$ in iteration $i$.

- The pheromone level on individual node or edge. Lower pheromone level indicates that the component is less popularly chosen by ants. As for a good route, these components with low pheromone level should be its critical components that make it stand out from other routes. This factor is expressed by:

\[
B = \frac{\tau_{\text{max}} - \tau}{\tau_{\text{max}} - \tau_{\text{min}}}
\]  

(10)

The amount to be added is given by:

\[
\Delta \tau_{a,i} = \begin{cases} 
Q L_{a,i}^a D, & \text{if } L_{a,i} < L_{\text{avg}}, \\
0, & \text{otherwise.}
\end{cases}
\]  

(11)

where $Q$ is a positive constant, $D$ is a positive constant larger than 1. Larger value of both $A$ and $B$ will encourage more pheromone deposit.

Under the new strategy, only a few ants can deposit pheromone, whose solution is: 1) the best solution in an iteration; 2) better than the average value of fitness since the beginning of algorithm. It is believed that this manner is able to direct later ants to route with better solution fitness.

3.5 Termination of algorithm

The most commonly used method to determine when to terminate the algorithm is to limit the total number of iterations, which is quite easy to implement. However, the number of iterations needed to obtain a satisfactory solution varies in each run, due to the randomness. This approach uses a more reasonable manner by limiting the total number of consecutive “non-improving iteration” (iteration in which no better solution is found). Resetting strategy is not applied.

3.6 Walkthrough

The flow chart for the procedures of the entire approach is presented in Fig. 2.

The algorithm parameters are: ant number $K$; weights of pheromone and heuristic desirability $\alpha$ and $\beta$; pheromone evaporation rate $\rho$; initial, maximum and minimum value of pheromone $\tau_0$, $\tau_{\text{max}}$ and $\tau_{\text{min}}$; constants $C$, $Q$ and $D$; limitation of consecutive non-improving iterations $R$. The variables are: pheromone values on nodes and edges; level of convergence $\sigma$; average and best makespan since started, $L_{\text{avg}}$ and $L_{\text{best}}$; counter of consecutive non-improving iterations $r$. The heuristic process is replicated until $R$ consecutive non-improving iterations occur. The best solution found during the whole process is then given as the final output of the algorithm.
4.1 Experiment I – determining algorithm parameters

In most ACO applications, the algorithm constants $C$ and $Q$ are usually arbitrarily assigned to ensure the range of $\tau$ and $\eta$ within reasonable expectation. The values depend on the features of the problem set. As for the problem set in this paper, $C = 150$ and $Q = 600$ are found to be appropriate, which give $\eta$ values of around 10 for most nodes and edges, and $\Delta r$ around 1.5 for pheromone deposit. The initial, maximum, and minimum value of pheromone $\tau_0$, $\tau_{\text{max}}$, and $\tau_{\text{min}}$ are respectively set to 1.0, 20 and 0.1 to limit the pheromone value with a reasonable range.

The other parameters, including pheromone evaporation rate $\rho$, relative weights of pheromone and desirability $\alpha$ and $\beta$, number $K$, elite strategy constant $D$, and limitation of consecutive non-improving iterations $\Delta_r$, are tested with different values. The aim of running these testing experiments is to obtain solutions with satisfactory quality in as few as possible iterations. The following values are found most suitable for the problems in the test cases: $\rho = 0.15$, $\alpha = 1$, $\beta = 2$, $K = 50$, $D = 15$, $R = 100$.

4.2 Experiment II – algorithm performance evaluation

The 24 problems have been run with the parameters determined in Experiment I. Each problem is run for 10 times. The average result is taken. The results are compared with SEA[6], IGA[12] and standard ACO, shown in Table II.

We can see that the enhanced ACO approach stands out for most of the problems (19 of 24), with criterion of minimizing makespan, especially for more complex problems. Even though the CPU time is increased from standard ACO (since more computations are involved), and the algorithm is slower than IGA, the time taken is still acceptable (only 3 minutes for the most complex problem, No. 24).

5. CONCLUSION AND FUTURE WORK

An enhanced ACO approach for IPPS is proposed in this paper. A graph-based formulation of IPPS is used to apply the ACO heuristics. By studying the algorithm and conduction of tests, the weaknesses and limitation of standard ACO for IPPS are pointed out, and corresponding modifications are designed to enhance the performance. The advantages of the enhanced algorithm have been illustrated by experimental evaluation.

For further study, dynamic scheduling involving unexpected disturbances could be a potential orientation. Optimizing multiple performance criteria, such as mean flow time, machine utilization, may also be considered in our future work.

ACKNOWLEDGMENT

The work described in this paper is fully supported by a grant from the Research Grants Council of Hong Kong (Project Code HKU 718809E).

REFERENCES


