Int J Adv Manuf Technol (2013) 67:231–241
DOI 10.1007/s00170-013-4769-4

ORIGINAL ARTICLE

Ant colony optimization for job shop scheduling using multi-attribute dispatching rules

Przemysław Korytkowski · Szymon Rymaszewski · Tomasz Wiśniewski

Received: 30 June 2012 / Accepted: 25 November 2012 / Published online: 5 February 2013

Abstract This paper proposes a heuristic method based on ant colony optimization to determine the suboptimal allocation of dynamic multi-attribute dispatching rules to maximize job shop system performance (four measures were analyzed: mean flow time, max flow time, mean tardiness, and max tardiness). In order to assure high adequacy of the job shop system representation, modeling is carried out using discrete-event simulation. The proposed methodology constitutes a framework of integration of simulation and heuristic optimization. Simulation is used for evaluation of the local fitness function for ants. A case study is used in this paper to illustrate how performance of a job shop production system could be affected by dynamic multi-attribute dispatching rule assignment.

Keywords Ant colony optimization · Multi-attribute dispatching rules · Discrete-event simulation · Dynamic job shop

1 Introduction

In this paper, a scheduling approach is proposed using a non-preemptive method for machine dispatching rules in a dynamic job shop environment. The dispatching rule is selected through a series of computations and evaluations of the system performance measures.

The problem of scheduling in dynamic job shops has been extensively studied for many years and attracts the attention of researchers and practitioners equally. The problem is usually characterized as one in which a set of jobs is to be processed over a period of time, each job consisting of one or more operations to be performed in a specified sequence on specified machines and requiring some processing time. The objective is to determine the job schedules that minimize a measure (or multiple measures) of performance [1].

A dispatching rule is a dynamic scheduling tool. It is used to select the next job to be processed from the set of jobs waiting at a free workstation. Dynamic dispatching rules are the most used scheduling algorithms for due date-related real-time scheduling [2]. Dispatching rules are normally intended to minimize the inventory and/or tardiness costs. It has been observed that no single rule performs well for all important criteria related to flow time, job tardiness, and other system performance measures.

The job shop scheduling problem is well-known as one of the hardest combinatorial optimization problems. In the case of m operations and k dispatching rules, there are k^m possibilities of rule selection. When there are also many waiting jobs in the queues at workstations, the scheduling problem becomes even more complex. Application of dispatching rules arises therefore from two main motivations. The scheduling problem in large-scale manufacturing systems involves very difficult combinatorial problems that would be difficult or even impossible to solve with analytical approaches in a short or acceptable time. Furthermore, the production environment in which we operate is characterized by many dynamic and disturbing operational conditions with

P. Korytkowski (✉) · S. Rymaszewski
Department of Computer Science, West Pomeranian University of Technology in Szczecin, Zolnierska 49, 71-210 Szczecin, Poland
e-mail: pkorytkowski@zut.edu.pl

T. Wiśniewski
Faculty of Management and Economics of Services, University of Szczecin, Cukrowa 8, 71-004 Szczecin, Poland

unforeseen incidents, consequently offline optimal scheduling becomes useless. Exact algorithms within a reasonable time frame may only solve small problems. Thus, heuristic and metaheuristic algorithms have been widely applied to solve the issue.

In this article, an ant colony optimization (ACO) approach is evaluated in solving scheduling problems in a dynamic job shop environment. The most common approach is to assign one dispatching rule for an entire, usually linear, system. ACO is to be used as a search mechanism for the proposed simulation–optimization method in order to find a suboptimal allocation of multi-attribute dispatching rules, assuming that each workstation can be governed by one of a several dispatching rules. The aim is to increase the efficiency of the large-scale production system through the selection of dispatching rules. Simulation results will be provided to show the feasibility and effectiveness of the proposed ACO strategy.

The remainder of this paper is organized as follows: The second section summarizes relevant literature on dynamic scheduling using dispatching rules. The third section describes the proposed methodology based on ACO and introduces multi-attribute dispatching rules. The fourth section describes a case study of a commercial offset printing system and the simulation model. The fifth section presents the results from the proposed methodology. Our conclusions and directions for future study are presented in the final section.

2 Literature review

Over the recent years in literature, there have been a lot of papers with respect to scheduling problems both for non-preemptive [3–5] and preemptive disciplines [6–8]. Dispatching rules are widely accepted in the industry because of the ease of implementation, satisfactory performance, low computational requirements, and the flexibility to incorporate domain knowledge and expertise [9]. Yang [10] remarked that effective scheduling is one of the key factors in improving the efficiency of wire-bonding operations, which is a bottleneck in the manufacture of integrated-circuit packaging. Scheduling using dispatching rules was applied to semiconductor wafer fab production [11, 12] and flexible manufacturing systems [13–15]. It has been generally observed that no single rule performs well for all important criteria related to flow time, job tardiness, and other regular and non-regular performance measures [16, 17], particularly in the dynamic environment of job shop scheduling.

The earliest due date rule (EDD) is a good algorithm for minimizing the maximum lateness [18]. The shortest processing time (SPT) rule has been found to be very effective in minimizing mean flow time and also minimizing mean tardiness, while the first in first out (FIFO) rule has been quite effective in minimizing the maximum flow time and variance of flow time in many cases. In recent years, studies have therefore been carried out to find new dispatching rules that improve most of the regular tardiness-related performance measures, such as slack processing time and work in next queue (PT + WINQ + SL), slack per remaining processing time and shortest processing time (S/RPT + SPT), slack time per remaining operation (S/OPN), earliest modified operational due date (EMODD), and others [1, 9, 19–21]. These rules use a combination of dispatching rules which result in a better improvement than rules using a single job attribute like SPT, EDD, or FIFO. Due to the complexity of the job shop scheduling problem, authors generally use heuristic and metaheuristic algorithms to solve the problem, including simulated annealing [22], tabu search method [23, 24], beam search heuristic [25], and also ACO algorithms [26–29].

ACO was inspired by the pheromone trail-laying behavior of ants and their following of this trail. Artificial ants in ACO are stochastic solution construction procedures that build candidate solutions for the problem instance under concern, by exploiting artificial pheromone information that is adapted based on the ant search experience and possibly available heuristic information [30]. ACO was successfully applied to many problems such as the traveling salesman problem [31–34] and mentioned earlier job shop scheduling [26–29].

Variants of the ACO algorithm generally differ in the applied pheromone update rule. Dorgio and Blumb [35] pointed out the three main types of ACO algorithm: ant system (AS), max–min ant system (MMAS), ant colony system (ACS). ACS and MMAS are regarded as the most successful ACO variants in practice. AS was introduced by Dorgio et al. [34]. In this algorithm, each ant reinforces the value of the pheromone on their path. There are three methods for calculating the pheromone update: ant cycle, ant density, and ant quantity. MMAS was introduced in [33] and differs from AS in several important aspects. Only the best solution from a population is used to update the pheromone values and a mechanism is added to limit the strengths of pheromones in order to avoid premature convergence. In the case of ACS, the state transition rule provides a direct way to balance the exploration of new edges and the exploitation of accumulated knowledge. ACS uses a global updating rule and local pheromone updating rule.

The novelty in this article lies in the assumption that any workstation may run a different dispatching rule, rather than the one rule for a whole manufacturing system as in most literature. In our approach, an allocation of dispatching rules for job assignment, but not a schedule, is encoded and the

ACO algorithm is used. In this paper, it has modified genetic operations as the search mechanism for the proposed simulation–optimization method, in order to find a better allocation of dispatching rules. Each dispatching rule can be interpreted as an edge in the route of ant. The aim is to increase the efficiency of large-scale production systems through the selection of multi-attribute dispatching rules. We study the influence of proper selection of dispatching rules on four performance measures related to tardiness and flow time.

3 Ant colony optimization algorithm

In ACO, we use agents called ants. The set of ants is called a population. Ants from the population are searching for the solution. The pheromone value depends on the quality of the current solution (value of a fitness function). Ants travel through a graph where nodes are dispatching rules at workstations and edges are production itineraries.

A general overview of the proposed methodology for solving the problem of dispatching rule allocation is shown below.

Step 1 Initialization
Step 2 Solution construction
Step 3 Fitness function evaluation (simulation)
Step 4 Local pheromone update (if current ant number≤ total number of ants, go to step 2)
Step 5 Global pheromone update
Step 6 Stop condition (if current iteration≤total number of iterations, then go to step 2)

3.1 Initialization phase

Most ACO algorithms set $\tau_0 = \frac{1}{n \times Z}$, where Z is the objective value of a solution obtained either randomly or using some simple heuristic. Sometimes, in order to avoid premature convergence, n is removed from the denominator. We propose initializing the algorithm by assigning to all workstations:

$$\tau_0 = \frac{Q}{f_i} \tag{1}$$

where Q is a constant

f_i is the value of fitness function, evaluated using a simulation applying the same dispatching rules to all workstations.

3.1.1 Characteristics of analyzed dispatching rules

In this paper, nine both single-attribute and multi-attribute dispatching rules are examined. The selected rules were deemed to have the best potential for offering a solution to the problem under consideration. We apply the following notation:

i Index of a job
j Index of an operation carried out for job i
p Index of a workstation
r Index of a dispatching rule in a workstation
m Number of workstations
n_i Number of operations for job i
k Number of considered dispatching rules
w_i Expected waiting time per operation for job i
A_i^m Arrival time of job i to the queue at workstation m
O_i Order arrival date for job i
P_i^m Processing time of job i at workstation m
P_i Total processing time of job i
R_i^m Remaining processing time of the job i after workstation m
Q_i^m Queuing time of job i at workstation m
t Time at which the priority index is calculated (present time)
D_i Due date for job i, calculated according to dynamic processing plus waiting time method, proposed by Enns [36]:

$$D_i = o_i + \sum_{m=1}^{n_i} P_i^m + n_i \cdot w_i \tag{2}$$

S_i Slack of job i, $S_i = D_i - t - R_i^m$
L^m Total processing time of the operations at the next workstation $m+1$
K Parameter of the cost over time (COVERT) rule

The highest priority is given to the job i with minimum value of priority index Z_i at the time of decision of dispatching. Analyzed dispatching rules are as follows:

1. FIFO: Rule selects the first job to enter the queue at a workstation buffer.

$$Z_i = A_i^m \tag{3}$$

2. EMODD: Rule chooses the next job to be processed from the input buffer which has the earliest operational due date.

$$Z_i = Max\{S_i,\ t + P_i^m\} \tag{4}$$

3. SPT: Rule selects the job which has the shortest processing time at the workstation.

$$Z_i = P_i^m \tag{5}$$

4. ALL + CR + SPT (the combination of the critical ratio (CR) and the SPT; ALL stands for allowance): For this

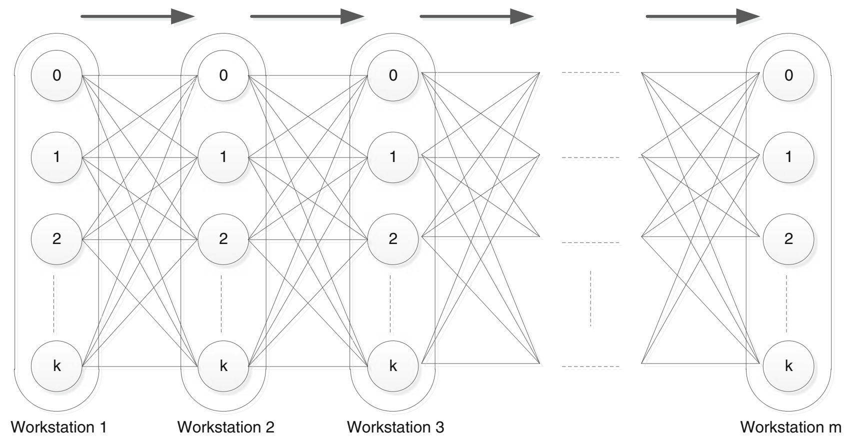

Fig. 1 Ant movement graph

rule, two separate queues are created for each workstation. The queue of the already overdue jobs has priority over the other. For this queue, the SPT rule is employed, but if this queue is empty, the CR + SPT rule is used for the secondary queue.

$$Z_i = Max\{t + \frac{D_i - t}{R_i^m} \cdot P_i^m, t + P_i^m\} \quad (6)$$

5. Minimum S/OPN: Rule selects the job with the least slack per remaining number of operations.

$$Z_i = \begin{cases} \frac{S_i}{n_i - j + 1} & \text{if } s_i \geq 0 \\ S_i \cdot (n_i - j + 1) & \text{if } s_i < 0 \end{cases} \quad (7)$$

6 S/RPT + SPT: This is combination of the slack per remaining processing time and the shortest processing time.

$$Z_i = Max\{\frac{S_i}{R_i^m} \cdot P_i^m, P_i^m\} \quad (8)$$

7. PT + WINQ + SL (combination of the slack processing time and work in next queue). WINQ selects the part from the current queue whose next process workstation has the shortest queue.

$$Z_i = S_i + P_i^m + L^m \quad (9)$$

8. PT + PW (combination of the processing time and waiting time in a given queue): Rule can achieve good performance on minimizing both mean tardiness and tardy rate.

$$Z_i = P_i^m + Q_i^m \quad (10)$$

9. COVERT rule: This is a more complicated combination of processing time-related and due date-related information. The COVERT rule is a popular benchmark rule when the mean and maximum tardiness are considered and has been shown to perform well [1].

$$Z_i = -\frac{1}{P_i^m} \cdot \left(1 - \frac{S_i}{K(R_i^m - P_i^m)}\right) \quad (11)$$

3.2 Construction of solution

An individual ant constructs solutions by iteratively adding dispatching rules for workstations until a complete candidate is generated (i.e., technological itinerary is

Table 1 Product characteristics

Parameter name	Product class				
	Leaflet	Poster	Box	Brochure	Book
Number of copies	Normal (30,000; 10,000)	Normal (5,000; 1,000)	Normal (8,000; 2,000)	Normal (5,000; 1,500)	Normal (1,000; 300)
Page format	A4, A5, A6	A2, A3	A1, A2	A4, A5, A6	A4, A5
Number of pages	1–2	1	1	Normal (30; 10)	Normal (350; 150)
Arrival rate	Expo (9)	Expo (12)	Expo (15)	Expo (10)	Expo (14)

Table 2 Technological operation parameters

	Workstation	Mean setup time per job	Standard deviation of setup time	Mean operation time	Standard deviation of operation time	Time unit
1	RIP	0	0	$20{\cdot}S_i$	0.2	min
2	CTP	0	0	$0.5{\cdot}S_i$	0	min
3	Printing	40	10	$0.005{\cdot}S_i$	0	min
4	Reversing	0	0	15	2	min
5	Drying	0	0	60	15	min
6	Folding	15	3	$0.0075{\cdot}S_i$	0	min
7	3-knife trimmer	20	5	$0.0075{\cdot}S_i$	0	min
8	Sticking cover	30	8	$0.006{\cdot}S_i$	0	min
9	Guillotine	8	2	$0.002{\cdot}S_i$	$0.0002{\cdot}S_i$	min
10	Die cutting	30	8	$0.0067{\cdot}S_i$	0	min
11	Collating	10	1	$0.0034{\cdot}S_i$	0	min
12	Binding	15	3	$0.005{\cdot}S_i$	0	min
14	Folding carton gluing	45	12	$0.0003{\cdot}S_i$	0	min

Where S_i is the lot-size parameter for job i depending on random parameters for each product shown in Table 1

finished). To construct a solution, ACO uses a state transition rule, which is the same as in the ACS algorithm. The state transition (Eqs. (12) and (14)) is called a pseudo-random-proportional rule. This state transition rule, as with the previous random-proportional rule, favors transitions with a large amount of pheromone [31]. An ant positioned in a workstation chooses the dispatching rule by applying the rule given by Eq (12), a simplified model of the problem.

$$s = \begin{cases} \arg\max\{[\tau_{pr}]\}, \textit{if } q \leq q_0 \\ S \end{cases} \qquad (12)$$

where q is a uniformly distributed random number [0, 1], and q_0 is a parameter ($0 \leq q_0 \leq 1$) which determines the relative importance of exploitation versus exploration. If $q \leq q_0$, then $k-$ ant takes the dispatching rule which maximizes workstation τ (exploitation); otherwise, a dispatching rule is chosen according to a random variable S (biased exploration) which is selected according to a probability distribution given by a simplified model:

$$P_{pr} = \frac{\tau_{pr}}{\sum_{r=1}^{k}\sum_{p=1}^{m}\tau_{pr}}. \qquad (13)$$

3.3 Fitness function evaluation

An individual ant keeps a set of dispatching rules (set size is equal to the number of workstations in the system). The edge at each position represents dispatching rules for the corresponding workstation. There are nine candidate dispatching rules. Each dispatching rule can be interpreted as an edge in the route of an ant. A discrete-event simulator is used to evaluate the performance of the modeled system, or in other words, the fitness

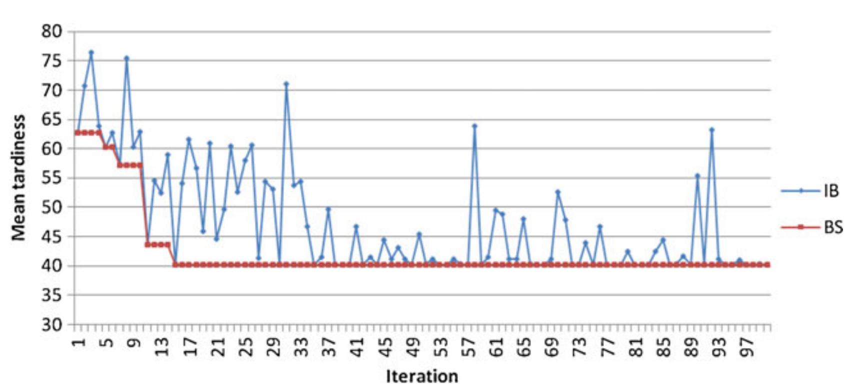

Fig. 2 Mean tardiness against number of ACO iterations

Table 3 Best results from each dispatching rule for all workstations compared to ACO and MC results

Dispatching rule	Mean tardiness	Mean flow	Max tardiness	Max flow
FIFO	165.77	256.86	492.85	546.01
EMODD	94.26	159.99	396.42	430.06
SOP/N	112.35	167.77	1,689.72	1,464.61
SPT	353.95	535.21	3,447.65	3,482.18
PT + WINQ + SL	111.38	167.51	1,633.87	1,490.19
S/RPT + SPT	139.74	207.37	1,657.32	1,098.14
ALL + CR + SPT	139.74	207.37	1,657.32	1,098.14
PT + PW	80.67	131.53	1,702.6	1,310.93
COVERT	163.05	242.64	2,838.65	2,640.66
ACO	40.21	69.34	310.52	413.91
Monte Carlo	42.98	82.23	414.79	474.87

function value for each ant. The fitness function is calculated as a mean value obtained from running a set of replications of simulation runs (Fig. 1).

3.3.1 Characteristics of analyzed dispatching rules

Fitness function in job shop manufacturing system will be adopted as one of the four measures:

1. Mean flow time—average time a job spends in the system
2. Maximum flow time—maximum time a job spends in the system
3. Mean tardiness—average tardiness of a job
4. Maximum tardiness—maximum tardiness of a job

3.4 Local search pheromone update

Local updating was introduced to the algorithm in order to dynamically change the attractiveness of edges (dispatching rules): every time an ant uses an edge, it becomes slightly less desirable. In this way, ants use pheromone information better. Without local updating, all ants would search in a narrow neighborhood of the previously best dispatching rule. In other words, the pheromone associated with the edge is modified each time the ant chooses dispatching rule r for workstation p. To locally update the pheromone value, we use Eq. (14) [37]:

$$\tau_{pr} = (1-\rho)\cdot\tau_{pr} + \rho\cdot\tau_0 \tag{14}$$

where:

p is index of workstation, $p\in(1,\ldots,m)$
r is index of dispatching rules in the workstation, $r\in(1,\ldots,k)$
τ_0 is initial pheromone value
ρ is evaporation rate

3.5 Global pheromone update

The aim of the global pheromone update is to increase pheromone values on solution components that have been found better in the sense of the fitness function value. In this case, we calculate the new value using Eq. (15):

$$\tau_{pr} = (1-\rho)\cdot\tau_{pr} + \Delta\tau_{pr} \tag{15}$$

The pheromone evaporation rate ρ [0, 1] is uniformly decreasing all pheromone values over time. Pheromone evaporation is needed to avoid a too rapid convergence of the algorithm toward a suboptimal region. It implements a useful form of forgetting, favoring the exploration of new areas in the search space [35]. We should avoid a situation where the ants construct the same solution over and over again and exploration stops while all ants are choosing the same dispatching rules at a particular workstation. MMAS imposes explicit limits on the minimum $\tau_{\min}$ and maximum $\tau_{\max}$ pheromone value and $\tau_{\min}\leq\tau_{pr}\leq\tau_{\max}$. The algorithm after each iteration has to ensure that the pheromone trail respects the limits [29], and the probability of choosing a specific solution component is never 0 if $f_{pr}\tau_{\min} > 0$. We calculate this in the following way:

$$\tau_{pr} = \begin{cases} If\ \tau_{pr} \geq \tau_{\max}\text{then } \tau_{pr} = \tau_{\max} \\ If\ \tau_{pr} < \tau_{\max}\, and \tau_{pr} > \tau_{\min}\text{then } \tau_{pr} = \tau_{pr} \\ If\ \tau_{pr} \leq \tau_{\min}\, then\ \tau_{pr} = \tau_{\min} \end{cases} \tag{16}$$

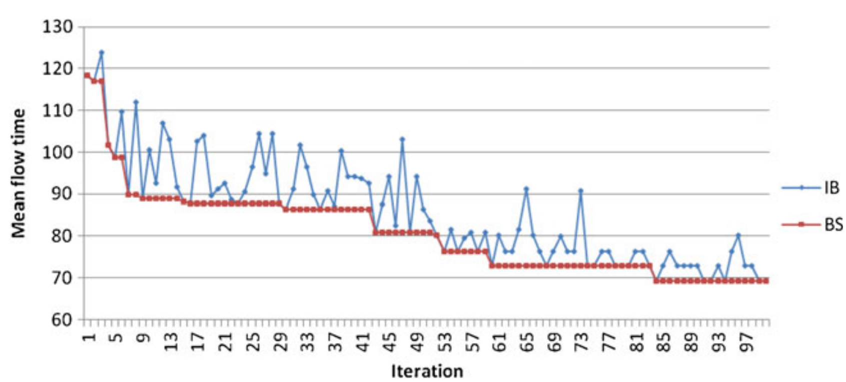

Fig. 3 Mean flow time against number of ACO iterations

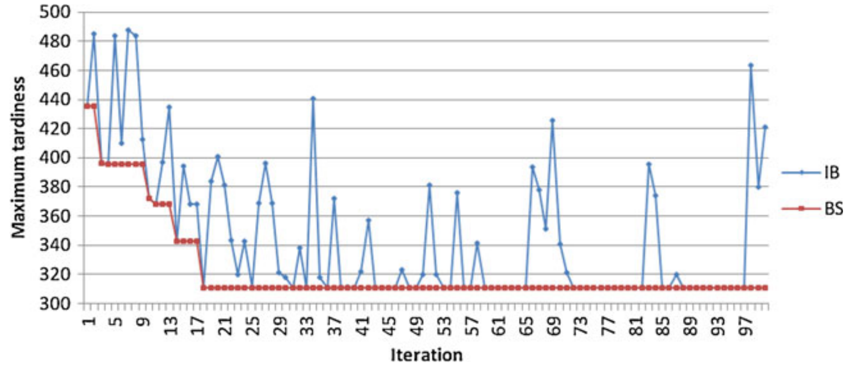

Fig. 4 Maximum tardiness against number of ACO iterations

After the search, the pheromone trail value of the new solution is updated proportionally to the improvement of the fitness function value. Moreover, in order to converge in a reasonable time, we additionally introduce iteration-best (IB) solution and best so far (BS) solution update rules. These amplify pheromone values on the ant path that led to the best in the last iteration or the best so far solution, i.e., it attracts more ants in the following iteration. The IB-update and BS-update rules introduce a strong bias towards the good solutions. This does however increase the danger of premature convergence. This is the reason why we used a modified construct solution model from ACS and pheromone update model from MMAS to avoid premature convergence:

$$\textit{If}\ \mathrm{pr} \in \mathrm{IB}\ \text{then}\ \Delta\tau_{\mathrm{pr}} = \frac{\mathrm{Q_{IB}}}{\mathrm{f_{pr}}}$$
$$\text{If}\ \mathrm{pr} \in \mathrm{BS}\ \text{then}\ \Delta\tau_{\mathrm{pr}} = \Delta\tau_{\mathrm{pr}} + \frac{\mathrm{Q_{BS}}}{\mathrm{f_{pr}}}$$

where f_{pr} is the value of fitness function for solution, IB is the best solution in last iteration, and BS is the best so far solution.

3.6 Stop condition

In literature [30, 34, 35], two stopping criteria are common: the process is iterated until the tour counter reaches the user-defined maximum number of cycles $\mathrm{NC_{MAX}}$, or all ants make the same tour. The last case is called stagnation behavior as it denotes a situation in which the algorithm stops searching for alternative solutions. In this paper, we use the first of these stopping criteria, the maximum number of iterations.

4 Case study: commercial offset printing system

To illustrate the methodology of evaluation and optimization of the dispatching rules, a discrete-event simulation model of a commercial offset printing system was built and is summarized as follows. There were three work areas (prepress, press, finishing), including 14 workstations consisting of single or multiple identical machines. There were five different processing flows for five different product types: softcover books, booklets, posters, leaflets, and boxes.

The commercial offset printing system was derived from a real-life company. Five types of products using different technical flows as shown in Table 1 were assumed to be processing simultaneously. Processing and setup times as shown in Table 2 are randomly distributed with known first- and second-order parameters (i.e., mean and standard deviation). In all the experiments below, the length of each simulation is 15,480,000 time units, with the first 57,600 time units being the warm up period. We ran the simulation seven times and used the average as the simulation result.

Analysis of large and complex stochastic systems is a difficult task due to the complexities that arise when

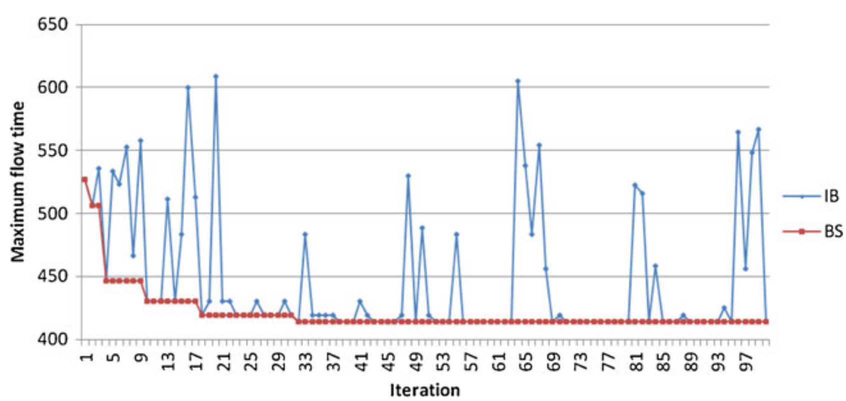

Fig. 5 Maximum flow time against number of ACO iterations

Table 4 ACO improvement for all performance criteria

Iterations	Mean tardiness			Mean flow time			Maximum tardiness			Maximum flow time		
	IB	BS	BS %	IB	BS	BS %	IB	BS	BS %	IB	BS	BS %
1	62.7	62.7	0	118.4	118.4	0	435.1	435.1	0	527.2	527.2	0
25	57.91	40.2	35.9	96.50	90.61	23.5	311.4	310.5	28.6	418.9	418.9	20.5
50	45.5	40.2	0	86.5	81.02	31.6	319.7	310.5	0	488.2	413.9	21.5
75	40.2	40.2	0	72.9	72.87	38.4	310.5	310.5	0	413.9	413.9	0
100	40.2	40.2	0	69.3	69.3	41.5	421.4	310.5	0		413.9	0
Improvement		22.6	35.9		49.1	41.5		124.5	28.6			21.5

randomness is embedded within a system. Since testing priority rules in real-world production is absolutely impossible, discrete-event simulation has often been adopted to evaluate the performance of dispatching rules for rule selection. Simulation modeling as an evaluative tool for stochastic systems has facilitated the ability to obtain performance measure estimates under any given system configuration. Simulation experiments were conducted to determine a suboptimal allocation of dispatching rules in the meaning of minimizing performance measures (mean flow time, mean tardiness, and max tardiness) for a typical commercial offset printing facility. ARENA simulation software from Rockwell Software was used for modeling the manufacturing system.

5 Results

In this section, simulation results and comparisons are provided to show the feasibility and effectiveness of the proposed ACO strategy. Experiments were carried out for four performance measures: (1) mean tardiness, (2) maximum tardiness, (3) mean flow time, and (4) maximum flow time. In each experiment, 100 iterations were performed of the ACO algorithm which gave 280,000 replications (100 iterations × 100 ants in each iteration × 7 replications of each simulation run × 4 performance measures).

Figure 2 presents the results of the ACO algorithm for mean tardiness (in hours). Already, in the first iteration, the created allocation of dispatching rules has given better results both for mean tardiness and mean flow time than. In Table 3, it can be seen that ACO gave better results for all performance measures than one rule for the entire system. In the case of mean tardiness, it is better than the best single-attribute dispatching rule (75 % better than FIFO) and best multi-attribute dispatching rule (50 % better than PT + PW). For validation, we have compared the results from a Monte Carlo simulation (20,000 instances). Validation showed that ACO is capable of finding the best suboptimal solution among the tested strategies (Table 3). On a typical PC (Intel Core i5 2.3 GHz, 4 GB Ram), ACO needed about

Table 5 Derived allocation of dispatching rule for all performance criteria

Workstation	Mean tardiness		Mean flow		Max tardiness		Max flow	
	Edge number	Dispatching rules	Edge number	Dispatching rules	Edge number	Dispatching rules	Edge number	Dispatching rules
RIP	7	PT + PW	4	PT + WINQ + SL	0	FIFO	1	EMODD
CTP	7	PT + PW	7	PT + PW	0	FIFO	1	EMODD
Offset 1	4	PT + WINQ + SL	7	PT + PW	0	FIFO	1	EMODD
Offset 2	7	PT + PW	7	PT + PW	0	FIFO	1	EMODD
Reversing	7	PT + PW	8	COVERT	0	FIFO	1	EMODD
Folding	7	PT + PW	2	SOP/N	1	EMODD	1	EMODD
3-knife trimmer	7	PT + PW	0	FIFO	0	FIFO	1	EMODD
Binding	1	EMODD	2	SOP/N	5	S/RPT + SPT	8	COVERT
Guillotine	8	COVERT	8	COVERT	2	SOP/N	1	EMODD
Die cutting	2	SOP/N	0	FIFO	1	EMODD	7	PT + PW
Folding carton	5	S/RPT + SPT	1	EMODD	1	EMODD	8	COVERT
Collating	2	SOP/N	1	EMODD	0	FIFO	1	EMODD

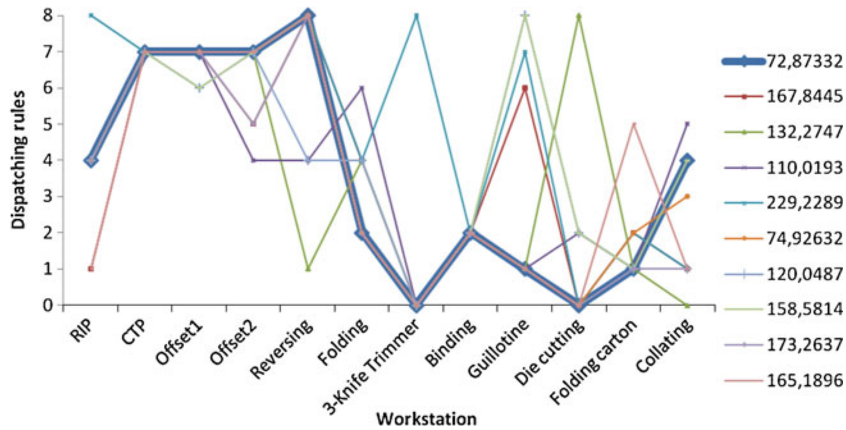

Fig. 6 Movement of the first 10 ants from sample population

23.5 h to find a suboptimal solution while Monte Carlo (for 20,000 scenarios) needed more than 46 h, without finding a better solution.

Figure 3 presents the results of the ACO algorithm for mean flow time (in hours). It can be seen that during the operation of the algorithm, it significantly improves (minimizes) the value of this performance measure. When we compare results from the first and last iteration, the improvement was 41.5 %.

Figures 4 and 5 show the effect of the ACO algorithm for maximum tardiness and maximum flow time (in hours), respectively. Results for both performance measures demonstrate proper operation of the developed ACO algorithm to minimize each of these measures. The algorithm significantly improved results in subsequent iterations, up to 18 iterations (for maximum tardiness) and up to 30 iterations (for maximum flow time). For all presented figures, the process of the IB values indicates that the ants in each population do not always go exactly the same way, which means a search for more possible solutions. After about 30 iterations, a notable increase in IB value can be seen. This is caused by the pheromone reset mechanism used in the algorithm in the case where there is no improvement over a given number of iterations. The algorithm thus converges again to a minimum.

Table 4 shows IB results, BS results, and percentage improvement of BS (BS%) for each performance criterion. Each performance criterion gained significant improvement. Results are shown for every 25th ACO iteration. For mean tardiness and maximum tardiness, the suboptimal solution was obtained after just 25 iterations.

Table 5 contains allocations of dispatching rules for each performance criterion. It can be seen that for minimizing mean tardiness and mean flow time, the best rules are multi-attribute rules like PT + WINQ + SL and PT + PW—rules which take into account the situation at the next workstation, especially for workstations that are the most loaded and arise as bottlenecks in the system, like offset machines.

Figure 6 presents the movement of the first 10 ants from a sample population. It can be seen how ants move among

Table 6 Sample population of ants for mean flow time

Mean flow time	Ant number	Mean flow time	Ant number	Mean flow time	Ant number
72.87332	0	197.022	34	307.1297	67
167.8445	1	149.7544	35	191.2246	68
132.2747	2	219.1308	36	143.8165	69
110.0193	3	164.4793	37	203.6097	70
229.2289	4	207.0687	38	231.7816	71
72.87332	5	165.0864	39	312.9008	72
120.0487	6	72.87239	40	292.3497	73
165.1896	7	173.6926	41	250.2933	74
158.5814	8	162.8692	42	148.8748	75
173.2637	9	315.1846	43	239.669	76
154.5076	10	242.0819	44	219.0832	77
272.4236	11	292.6162	45	179.6459	78
291.6677	12	122.3642	46	154.0787	79
153.2452	13	168.773	47	139.6005	80
205.0173	14	241.3758	48	256.8527	81
173.9866	15	171.705	49	137.2193	82
196.924	16	137.0676	50	148.8225	83
138.6102	17	231.2122	51	96.69304	84
212.7054	18	324.2665	52	195.2554	85
76.42042	19	147.6645	53	180.172	86
196.8091	20	123.93	54	163.7891	87
118.6621	21	152.6104	55	165.7132	88
212.5578	22	221.7794	56	165.285	89
231.4182	23	191.6432	57	154.3732	90
315.9376	24	208.8572	58	149.5411	91
80.34066	25	249.4601	59	263.5935	92
146.6493	26	168.3177	60	271.1757	93
176.4914	27	155.9873	61	208.1513	94
189.5219	28	147.571	62	128.5893	95
174.1748	29	140.4601	63	205.6723	96
207.8188	30	167.042	64	123.469	97
219.1089	31	271.8226	65	311.853	98
181.3684	32	187.4176	66	132.9907	99
170.6405	33				

dispatching rules in workstations. One color stands for one ant. Results presented in the legend show mean flow time for each ant after passing the path.

In Table 6, the sample population of 100 ants is presented for mean flow time criterion, presenting the differentiation of one population. The large dispersion of results proves the effect of the local pheromone update and also the scale of how dispatching rules may change the results of a performance measure. When we combine results from Fig. 6 with data from Table 6, it is clear that the ACO algorithm works well in combing the vast searching area.

6 Conclusion

The conducted experiments and analyses show that proper management of the allocation of orders in the system can improve the efficiency by several percent. Optimization of the job processing order does not entail the need for additional investment in machinery or equipment.

The simulation model works well and answers many important questions. It proves firstly that the dispatching rules do change the results: flow time tardiness, size of queues, amount of finished products, etc.

The presented ACO algorithm worked well and found an allocation of dispatching rules that gave better results for all criteria than for just one rule in an entire system. For all performance criteria, the ACO algorithm converged and gave, in an acceptable time, good results for the scheduling problem.

References

1. Rajendran C, Holthaus O (1999) A comparative study of dispatching rules in dynamic flowshops and jobshops. Eur J Oper Res 116:156–170
2. Lengyel A, Hatono I, Ueda K (2003) Scheduling for on-time completion in job shops using feasibility function. Comput Ind Eng 45:215–229
3. Iravani F, Balcıoglu B (2008) On priority queues with impatient customers. Queueing Syst 58:239–260
4. Pardo MJ, de la Fuente D (2007) Optimizing a priority-discipline queuing model using fuzzy set theory. Comput Math Appl 54:267–281
5. Krishnamoorthy A, Babu S, Narayanan VC (2009) The MAP/(PH/PH)/1 queue with self-generation of priorities and non-preemptive service. Eur J Oper Res 195:174–185
6. Walraevens J, Steyaert B, Bruneel H (2006) A preemptive repeat priority queue with resampling: performance analysis. Ann Oper Res 146:189–202
7. Van Houdt B, Blondia C (2006) Analyzing priority queues with 3 classes using tree-like processes. Queueing Syst 54:99–109
8. Katayama T (2007) Analysis of a time-limited service priority queueing system with exponential timer and server vacations. Queueing Syst 57:169–178
9. Chiang TC, Fu LC (2007) Using dispatching rules for job shop scheduling with due-date based objectives. Int J Prod Res 45:3245–3262
10. Yang T, Tseng LP (2002) Solving a multiple objective simulation model using a hybrid response surface method and lexicographical goal programming approach—a case study on IC ink marking machines. J Oper Res Soc 53:211–221
11. Kim Y-D, Kim J-U, Lim S-K, Jun H-B (1998) Due-date based scheduling and control policies in a multiproduct semiconductor wafer fabrication facility. IEEE T Semiconduct M IEEE T Semiconduct M 11:155–164
12. Zhang H, Jiang Z, Guo C (2009) Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. Int J Adv Manuf Technol 41:110–121
13. Bocewicz G, Wójcik R, Banaszak Z. (2008) AGVs distributed control subject to imprecise operation times. In: Agent and multi-agent systems: technologies and applications, LNAI 4953: 421–430
14. Chan FTS, Chan HK, Lau HCW, Ip RWL (2003) Analysis of dynamic dispatching rules for a flexible manufacturing system. J Mater Process Tech 138:325–331
15. Chan FTS (2003) Effects of dispatching and routing decisions on the performance of a flexible manufacturing system. Int J Adv Manuf Technol 21:328–338
16. Blackstone JH, Philips DT, Hogg GL (1982) A state-of-the survey of dispatching rules for manufacturing job shop operations. Int J Prod Res 20:27–45
17. Dominic PDD, Kaliyamoorthy S, Saravana Kumar M (2004) Efficient dispatching rules for dynamic job shop scheduling. Int J Adv Manuf Technol 24:70–75
18. Jayamohan MS, Rajendran C (2000) New dispatching rules for shop scheduling: a step forward. Int J Prod Res 38:563–586
19. Holthaus O, Rajendrant C (1997) New dispatching rules for scheduling in a job shop—an experimental study. Int J Adv Manuf Technol 13:148–153
20. Yin YL, Rau H (2006) Dynamic selection of sequencing rules for a class-based unit-load automated storage and retrieval system. Int J Adv Manuf Technol 29:1259–1266
21. Vinod V, Sridharan R (2011) Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system. Int J Prod Econ 129:127–146
22. Kim DW, Kim KH, Jang W, Chen FF (2002) Unrelated parallel machine scheduling with setup times using simulated annealing. Robot Cim-Int Manuf 18:223–231
23. Nowicki E, Smutnicki C (1996) A fast taboo search algorithm for the job-shop problem. Manage Sci 42:797–813
24. Pezella F, Merelli E (2000) A tabu search method guided by shifting bootleneck for the job shop scheduling. Eur J Oper Res 120:297–310
25. Sabuncuoglu I, Bayiz M (1999) Job shop scheduling with beam search. Eur J Oper Res 118:390–412
26. Huang M, Wu T, Liang X (2010) GA-ACO in job-shop schedule problem research. Comput Intel Syst 107:226–233
27. Avila Rondon RL, Carvalho AS (2009) Solving a real job shop scheduling. Industrial Electronics. IECON '09. 35th Annual Conference of IEEE: 2494–2498
28. Lu MS, Romanowski R (2012) Multi-contextual ant colony optimization of intermediate dynamic job shop problems. Int J Adv Manuf Technol 60:667–681
29. Liao CJ, Tsai YL, Chao CW (2011) An ant colony optimization algorithm for setup coordination in a two-stage production system. Appl Soft Comput 11:4521–4529

30. Dorigo M, Stutzle T (2010) Ant colony optimization: overview and recent advances. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics, vol 146, Secondth edn. Springer, New York, p 561, Chapter 8
31. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE T Evolut Comput 1:53–66
32. Nonsiri S, Supratid S (2008) Modifying ant colony optimization. IEEE Conference on Soft Computing in Industrial Applications. June 25–27. Muroran. Japan: 95–100
33. Stützle T, Hoos HH (2000) MAX-MIN ant system. Future Gen Comput Systems 16(8):889–914
34. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Systems Man Cybernet-Part B 26:29–41
35. Dorigo M, Blumb C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344:243–278
36. Enns ST (1995) A dynamic forecasting model for job shop flow time prediction and tardiness control. Int J Prod Res 33:1295–1312
37. Udhayakumar P, Kumanan S (2011) Integrated scheduling of flexible manufacturing system using evolutionary algorithms. Int J Adv Manuf Technol Int J Adv Manuf Technol 61:621–635, 2012