Taylor & Francis
Taylor & Francis Group

Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: 0005-1144 (Print) 1848-3380 (Online) Journal homepage: https://www.tandfonline.com/loi/taut20

A hybrid simulated annealing for scheduling in dual-resource cellular manufacturing system considering worker movement

Jufeng Wang, Chunfeng Liu & Kai Li

To cite this article: Jufeng Wang, Chunfeng Liu & Kai Li (2019) A hybrid simulated annealing for scheduling in dual-resource cellular manufacturing system considering worker movement, Automatika, 60:2, 172-180, DOI: 10.1080/00051144.2019.1603264

To link to this article: https://doi.org/10.1080/00051144.2019.1603264

Published online: 15 Apr 2019.

AUTOMATIKA
2019, VOL. 60, NO. 2, 172–180
https://doi.org/10.1080/00051144.2019.1603264

REGULAR PAPER

OPEN ACCESS Check for updates

A hybrid simulated annealing for scheduling in dual-resource cellular manufacturing system considering worker movement

Jufeng Wang[a], Chunfeng Liu[b] and Kai Li[c]

[a]Department of Mathematics, China Jiliang University, Hangzhou, People's Republic of China; [b]School of Management, Hangzhou Dianzi University, Hangzhou, People's Republic of China; [c]School of Management, Hefei University of Technology, Hefei, People's Republic of China

ABSTRACT

This paper presents a novel linear mathematical model for integrated cell formation and task scheduling in the cellular manufacturing system (CMS). It is suitable for the dual-resource constrained setting, such as garment process, component assembly, and electronics manufacturing. The model can handle the manufacturing project composing of some tasks with precedence constraints. It provides a method to assign the multi-skilled workers to appropriate machines. The workers are allowed to move among the machines such that the processing time of tasks might be reduced. A hybrid simulated annealing (HSA) is proposed to minimize the makespan of manufacturing project in the CMS. The approach combines the priority rule based heuristic algorithm (PRBHA) and revised forward recursion algorithm (RFRA) with conventional simulated annealing (SA). The result of extensive numerical experiments shows that the proposed HSA outperforms the conventional SA accurately and efficiently.

ARTICLE HISTORY
Received 14 March 2017
Accepted 26 March 2019

KEYWORDS
Cellular manufacturing system; task scheduling; precedence constraint; simulated annealing; heuristic

1. Introduction

With rapidly changing customer expectation and global competition, cellular manufacturing system (CMS) has been an important way of producing goods in the last several decades. It shows many advantages such as an effective response to the rising product variety, a reduction in material handling cost and production lead time, streamlined production control, and enhanced productivity. Cell formation including grouping machines and tasks, and task scheduling involving decisions on task dispatching rules and timetable, are two main issues in the CMS. Consequently, these problems have attracted much investigating interest from researchers and practitioners.

For the basic cell formation problem with machine assignment, Rabbani et al. [1] used a two-phase fuzzy linear programming approach to solve a bi-objective cell formation problem with stochastic production quantities. Arkat et al. [2] proposed a branch and bound algorithm to minimize the total number of movements between each pair of machines locating in two different cells. Saraç and Ozcelik [3] used a genetic algorithm to maximize the grouping efficacy. Chung et al. [4] proposed an efficient tabu search algorithm to solve the cell formation problem with alternative routings and machine reliability considerations. Rafiei et al. [5] designed a dynamic cellular manufacturing system to pursue fundamentals of Just-In-Time production philosophy. A nonlinear programming model is proposed with two conflicting objective functions: minimizing the sum of cost, and minimizing the work-in-process. Mar-Ortiz et al. [6] presented a mathematical programming model to minimize the sum of the machine amortization cost, the machine relocation cost, and the intercellular material handling cost. By a reconfigurable approach, the cells are rearranged periodically to deal with demand variability in a multi-period planning horizon. Jayakumar and Raju [7] presented a nonlinear mixed-integer mathematical model for the cell formation problem with the uncertainty of the product mix for a single period. A solution methodology for best possible cell formation using simulated annealing (SA) is presented in order to minimize the total sum of the machine constant cost, the operating cost, the inter-cell material handling cost, and the intra-cell material handling cost. Some other methods have emerged for cell formation problems, such as particle swarm optimization method [8], clustering method [9] and scatter search algorithm [10]. It should be pointed out that the results in the aforesaid references only consider machine constrained setting.

For the cell formation problem with worker and machine assignment, Mahdavi et al. [11] presented a fuzzy goal programming-based approach for solving a multi-objective mathematical model of cell formation problem and production planning in a dynamic virtual cellular manufacturing system. Bagheri and Bashiri [12] proposed a new mathematical model to solve the worker assignment and inter-cell layout problems. The objective function of the proposed model consists of two main cost categories. The preferred solution

CONTACT Chunfeng Liu lcf_spring@163.com School of Management, Hangzhou Dianzi University, Hangzhou 310018, People's Republic of China

is obtained by a LP-metric approach. Mahdavi et al. [13] investigated an integer mathematical programming model for the design of CMSs in a dynamic environment. The aim of the proposed model is to minimize holding and backorder costs, inter-cell material handling cost, machine and reconfiguration costs, and hiring, firing and salary costs. Azadeh et al. [14] presented a simulation approach for optimization of operator allocation in the CMS. Süer et al. [15] proposed a three-phase methodology to deal with the problem of cell loading and product sequencing in labour-intensive cells. Bootaki et al. [16] designed a robust method to configure cells in a dynamic CMS to minimize the inter-cell movements and maximize the machine and worker utilisation.

In contrast with the cell formation problem, there are only a small quantity of articles addressing the problem of scheduling in the CMS. Venkataramanaiah [17] developed a SA for scheduling of parts within a cell for the objective of minimizing weighted sum of makespan, flowtime and idle time. Tavakkoli-Moghaddam et al. [18] designed a scatter search method for a multi-criteria group scheduling problem in the CMS. Halat and Bashirzadeh [19] suggested a heuristic for scheduling operations of manufacturing cells considering sequence-dependent family setup times and intercellular transportation times. Arkat et al. [20] presented a mathematical model to concurrently identify the formation of cells, cellular layout and the operations sequence with the objective of minimizing total transportation cost of parts as well as minimizing makespan. Liu et al. [21] developed a discrete bacteria foraging algorithm to solve the model of CMS with the objective of minimizing the material handling costs as well as the fixed and operating costs of machines and workers.

Because of the high complexity of CMS which is subject to dual-resource constrained conditions, the cell formation and group scheduling problems are often analyzed independently. To the best of the authors' knowledge, few related research has involved the CMS problem simultaneously considering multi-functional machines, multi-skilled workers and task sequence yet. Moreover, the impact of worker movement on task scheduling is also desired to be discussed.

The remainder of this paper is organized in the following: In Section 2, the proposed problem is stated and formulated as a mathematical model integrating cell formation and task scheduling. In Section 3, the PRBHA algorithm is suggested to obtain an initial solution with high quality. In Section 4, the HSA algorithm is designed for further search to get a global optimum. In Section 5, the performance of the proposed HSA is validated in comparison with the conventional SA by computational experiments. Finally, conclusions are drawn followed by some potential research directions in Section 6.

2. Problem statement and formulation

In this section, the problem of cell formation is formulated as a linear integer programming mathematical model. The objective is minimizing the makespan of the project which is composed of n tasks (i.e. maximum completion time of all tasks). The following hypotheses are made for the proposed problem.

(1) *Machine and worker hypotheses*: The number of machines and workers are known in advance. The number of machines is more than the number of workers.

(2) *Task hypothesis*: For each task, at least one machine has the ability to process it. For each machine, any worker has the ability to operate it. The processing of each task is not allowed to be interrupted, which implies that each task is processed on only one machine by only one worker. The processing time of task depends on the assigned machine and worker. There exists precedence relationship among tasks.

(3) *Cell size hypothesis*: The number of machines in each cell can not exceed a specified maximum, because redundant machines in a cell may generate cluttered flows in many routes.

(4) *Worker movement hypothesis*: The workers are permitted to move among different machines, and the movement time is known in advance.

Subscripts

w	Index for worker.
c	Index for cell.
j	Index for task.
t	Index for time.
m	Index for machine.
r	Index for continuous time interval (i.e. one worker's rth task is processed in the worker's rth continuous time interval).

Input parameters

W	The number of workers.
J	The number of tasks.
M	The number of machines.
N	The largest number of tasks that one worker can process, (e.g. $N=J$ if all the tasks are processed by one worker).
B_u	Upper cell size limit (measured in the number of machines).
C	The number of cells, which is the smallest integer not less than M/B_u.
Q_{jm}	1 if task j can be processed on machine m, and 0 otherwise ($j=1,\ldots,J$; $m=1,\ldots,M$).
$\mathscr{M}_j$	The set of machines that have the ability to process the task j ($j=1,\ldots,J$).
p_{jmw}	Processing time of task j on machine m by worker w ($j=1,\ldots,J$; $m\in\mathscr{M}_j$; $w=1,\ldots,W$).
$Y_{mcm'c'}$	Movement time of worker moving to machine m' in cell c' from machine m in cell c (m, $m'=1,\ldots,M$; $c,c'=1,\ldots,C$).

P_j The immediate predecessor task set of task j ($j = 1, \ldots, J$).
L A sufficient large number.
Decision variables
Z_{mc} 1 if machine m is located in cell c, and 0 otherwise ($m = 1, \ldots, M$; $c = 1, \ldots, C$).
$\mathscr{C}_m$ The cell in which machine m is located. $\mathscr{C}_m = c$ if $Z_{mc} = 1$.
x_{jmwr} 1 if task j is processed on machine m by worker w, and the task j is the rth one processed by worker w (i.e. worker w process task j on machine m in his/her rth continuous time interval), and 0 otherwise ($j = 1, \ldots, J$; $m = 1, \ldots, M$; $w = 1, \ldots, W$; $r = 1, \ldots, N$).
FT_j The finish time of task j ($j = 1, \ldots, J$).

The proposed problem is formulated as the following linear integer programming model:

$$\text{Min} \quad \mathbb{C}_{\max} = \max\{FT_j \mid j = 1, \ldots, J\} \tag{1}$$

$$\text{s.t.} \quad \sum_{c=1}^{C} Z_{mc} = 1, \quad \forall\, m \tag{2}$$

$$\sum_{m=1}^{M} Z_{mc} \le B_u, \forall c \tag{3}$$

$$x_{jmwr} \le Q_{jm}, \ \forall j,\ m,\ w,\ r \tag{4}$$

$$\sum_{m=1}^{M}\sum_{w=1}^{W}\sum_{r=1}^{N} x_{jmwr} = 1, \ \forall j \tag{5}$$

$$\sum_{j=1}^{N}\sum_{m=1}^{M} x_{jmwr} \le 1, \ \forall w,\ r \tag{6}$$

$$\sum_{i=1}^{N}\sum_{m=1}^{M} x_{imwr} - \sum_{j=1}^{N}\sum_{m=1}^{M} x_{j,m,w,r-1} \le 0, \forall w,\ r = 2, \ldots, N \tag{7}$$

$$FT_j - FT_i + L(2 - x_{jm'wr} - x_{i,m,w,r-1}) \ge p_{jmw} + Y_{m\mathscr{C}_m m'\mathscr{C}_{m'}}, \forall i \ne j,\ \forall m,\ w,\ r = 2, \ldots, N \tag{8}$$

$$FT_j - FT_i \ge \sum_{m=1}^{M}\sum_{w=1}^{W}\sum_{r=1}^{R} p_{jmw} x_{jmwr}, \forall i \in P_j,\ \forall j \tag{9}$$

$$FT_j \ge 0, \quad Z_{mc}, x_{jmwr} \in \{0, 1\}, \quad \forall\, j, m, w,\ r \tag{10}$$

The objective function (1) is to minimize the makespan $\mathbb{C}_{\max}$. Constraint (2) ensures that each machine should be located in one and only one cell. Constraint (3) shows that the number of machines in each cell can not exceed the upper limit of cell size. Constraint (4) guarantees that each task is processed on one of the machine that can process the task. Constraint (5) implies that each task is processed on only one machine by only one worker, and each task should be processed in the worker's only one continuous time interval. Constraint (6) guarantees each worker operates no more than one machine and processes no more than one task in his/her continuous time interval. Constraint (7) guarantees only if certain worker processes one task in a continuous time interval, the worker must have processed another task in the previous continuous time interval. Constraint (8) shows the relationship of the completion time of two tasks which are processed in one worker's consecutive time intervals. Constraint (9) ensures the precedence relationship between consecutive tasks. Constraint (10) ensures the type of the decision variables.

The proposed model has some advantages and characteristics. (1) It is suitable for the CMS with dual-resource constrained setting, such as garment process, component assembly, and electronics manufacturing. (2) The model can handle the manufacturing project composing of some tasks with precedence order. For example, Figure 1 shows the manufacturing process of professional road bike. The task of standard wheel subassembly can not start until the tasks of wheel spokes, wheel tire and wheel rim are finished. (3) In many labour intensive companies, the workers are cross-trained with multiple skills in order to increase flexibility and reduce salary cost. The model can provide a method to assign the workers to appropriate machines. (4) In this model, each worker is permitted to move from one machine to another for performing another task, which might be able to decrease processing time.

3. Priority rule based heuristic algorithm

In this section, we develop a priority rule based heuristic algorithm (PRBHA) that is embedded in SA for determining an initial feasible schedule. The PRBHA consists of n iterations (n is the total number of takes). At each iteration, a prior task is selected according to the EFT (earliest finishing time first) rule. Assuming there is a dummy task s with 0 unit of processing time at the beginning of the project. Moreover, assuming the machine that processes the dummy task s is a dummy machine 0, and the cell where the dummy machine 0 is located is a dummy cell 0.

The variables used for the PRBHA are listed in the following:

D The decision task-set, i.e. the unscheduled tasks whose immediate predecessor tasks have been completed.
$\Re$ The completed task-set, i.e. the tasks that have been completed.
θ_j The maximum finish time of the immediate predecessors of task j.
IM_m The start idle time of machine m.
IW_w The start idle time of worker w.
W_m The worker who operates machine m.
M_w The machine operated by worker w.

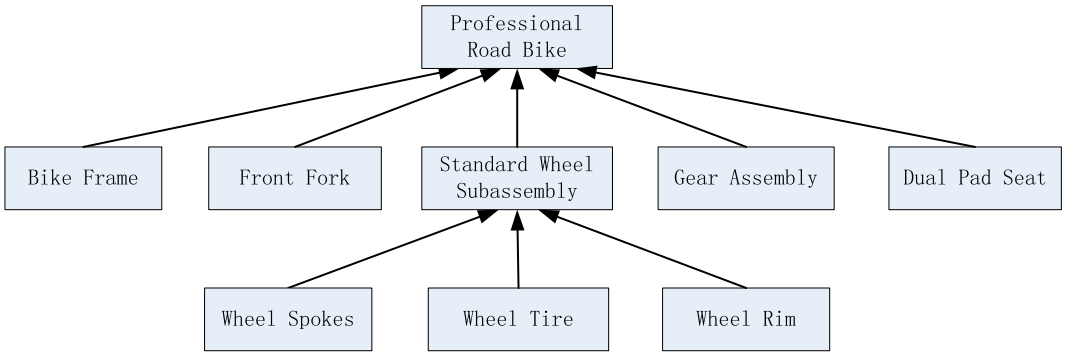

Figure 1. Manufacturing flow chart of road bike.

ft_{jwm}	The hypothesis completion time of task j which is processed by worker w on machine m.
(j^*, m^*, w^*)	The prior triple form, where j^* denotes the prior task, m^* denotes the prior machine, and w^* denotes the prior worker.
FT_j	The finish time of task j.
ST_j	The start time of task j.

We give a pseudo-code description of the PRBHA (see Algorithm 1) as follows:

Algorithm 1 Priority Rule Based Heuristic Algorithm (PRBHA)

$D = \{j \mid j \notin \Re,\ P_j \subseteq \Re\}$
1: Initialize: $FT_s = 0,\ \Re = \{s\},\ IW_w = 0,\ IM_m = 0,\ \mathscr{C}_0 = 0,\ Y_{0cm'c'} = 0,\ M_w = 0,$ $\forall w,\ \forall m,\ \forall m',\ \forall c'$, and randomly generate the values of $\mathscr{C}_m$.
2: **for** $g = 1 \to n$ **do**
3: Compute D.
4: $\theta_j = \max\{FT_h \mid h \in P_j\},\ \forall j \in D.$
5: $ft_{jwm} = \max\{\theta_j,\ IM_m,\ IW_w + Y_{M_w,\mathscr{C}_{M_w},m,\mathscr{C}_m}\} + p_{jmw},\ \forall j \in D,\ \forall m \in \mathscr{M}_j,\ \forall w.$
6: (j^*, m^*, w^*): $ft_{j^*m^*w^*} = \min\{ft_{jmw} \mid j \in D,\ m \in \mathscr{M}_j,\ \forall w\}.$
7: $FT_{j^*} = ft_{j^*m^*w^*}$
8: $IW_{w^*} = FT_{j^*},\ IM_{m^*} = FT_{j^*},\ M_{w^*} = m^*,\ \Re := \Re \cup \{j^*\}$
9: **end for**
10: $\mathbb{C}_{\max} = \max\{FT_j \mid j = 1, \ldots, J\}$

In Step 1, some variables $\Re$, IW_w, IM_m, and M_w are initialized. Let $Y_{0cm'c'} = 0$, $FT_s = 0$, $\mathscr{C}_0 = 0$. The machines are randomly assigned to the cells. In Step 2, each job is scheduled at each iteration. In each iteration, firstly, compute the decision task-set D in Step 3, and then compute the hypothesis completion time of each task j in D with different workers and different machines in $\mathscr{M}_j$ in Steps 4–5. Secondly, according to the EFT rule, select a prior task j^*, a prior machine m^*, and a prior worker w^* in Step 6. Finally, record the finish time of task j^*, and update the start idle time of machine m^*, the start idle time of worker w^*, the machine operated by the worker w^*, and the completed task-set $\Re$ in Steps 7–8. In Step 10, the makespan of the project is computed according to the finish time of each task.

4. Hybrid simulated annealing algorithm

Since SA was introduced by Kirkpatrick et al. [22]. It has become one of the most popular metaheuristic methods to solve complex optimization problems in manufacturing systems [23,24]. The name of SA and its inspiration comes from annealing in metallurgy. The main mechanism is that SA decreases the probability of accepting worse solutions as the temperature drops gradually. Accepting worse solutions allows for a more extensive search in the solution space, and provide the chances to jump out the local optima. In this section, we propose a hybrid simulated annealing (HSA) which combines the PRBHA approach with conventional SA algorithm.

4.1. Initial solution

By the PRBHA, the value of the $\mathscr{C}_m$, FT_j, $\mathbb{C}_{\max}$ and (j^*, m^*, w^*) are generated. From the prior triple form $(j^*,\ m^*,\ w^*)$, we can see that task j^* is processed on machine m^* by worker w^*. From the values of $\mathscr{C}_m$, FT_j and $\mathbb{C}_{\max}$, we know the location cell $\mathscr{C}_m$ for machine m, the finish time of task j, and the makespan $\mathbb{C}_{\max}$. Therefore, the PRBHA generates a feasible initial solution in the HSA.

4.2. Neighborhood generation strategy

It is important to design superior solution mutation(SM) operators for the search of HSA. In this research, three different mutation strategies are provided in the following:

(1) Machine-cell mutation(SM1): Randomly select a machine m in cell c and move it to different cell c' if the number of machines in cell c' does not reach the upper cell size limit, otherwise randomly

select a machine m' in cell c', and then exchange machines m and m' between the two cells.

(2) Task-machine mutation(SM2): A task which can be processed by more than one machine is randomly selected, and then randomly reassigned to another machine that can process the task.

(3) Task-worker mutation(SM3): A task is randomly selected, and then randomly reassigned to another worker.

The objective function values (i.e. makespan $\mathbb{C}_{\max}$) of the neighbourhood solutions can be calculated by the revised forward recursion algorithm (see Algorithm 2).

Algorithm 2 Revised Forward Recursion Algorithm (RFRA)

1: Initialize: The unallocated task set $U = \{1, \ldots, n\}$; The start idle time of machine m and worker w, $IM_m = 0,\ \forall m, IW_w = 0,\ \forall w$.
2: **while** $U \neq \phi$ **do**
3: Randomly select a job j from U.
4: Allocate task j to machine m operated by worker w (from solution schedule, the values m and w are known).
5: If the finish time of predecessor i of task j has not been determined, recursively execute Step 4 for predecessor i.
6: Compute the start time of task j, $ST_j = \max\{\theta_j,\ IM_m,\ IW_w + Y_{M_w,\mathscr{C}_{M_w},m,\mathscr{C}_m}\}$, and the finish time of task j, $FT_j = ST_j + p_{jmw}$.
7: Update the unallocated task set, the start idle time of machine m and worker w, and the value of M_w: $U := U\backslash\{j\}$, $IM_m = FT_j$, $IW_w = FT_j$, $M_w = m$.
8: **end while**
9: $\mathbb{C}_{\max} = \max\{FT_j \mid j = 1, \ldots, J\}$.

4.3. Cooling schedule

(1) Parameters of HSA algorithm: Initial temperature T_0, final temperature T_f, cooling rate α and Markov chain length $L_{\max}$ are set to 200,0.5, 0.95 and 200, respectively. The temperature T is decreased by using the following common equation: $T := \alpha T$.

(2) Termination condition: Let the best schedule up to now is x_{best}, the HSA algorithm is stopped if x_{best} is not changed after three consecutive temperature levels or the final temperature T_f is reached.

4.4. The pseudo code of the HSA

Algorithm 3 provides the pseudo-code of the HSA. The major optimization procedure is that: Generate an initial solution by the PRBHA. If $\mathbb{C}_{\max}$ of neighbourhood solution x_l is less than $\mathbb{C}_{\max}$ of current solution x_c, accept x_l as current solution x_c, otherwise neighbourhood solution x_l is accepted as current solution x_c by certain probability, which can escape from local optima to reach a global optimum. At the start, the probability of accepting nonimproving solutions is high, but as the search continues (i.e. the temperature drops), the probability of accepting nonimproving solutions decreases.

Algorithm 3 Hybrid Simulated Annealing (HSA)

1: Initialize parameters: $T = 200, T_f = 0.5, \alpha = 0.95, Total = 0, Change = 0, Unchange = 0$.
2: By algorithm 1, generate initial schedule x_0, set $x_c = x_0$, and compute the objective value of schedule, $\mathbb{C}_{\max}$.
3: **while** $T > T_f$ **do**
4: **while** $Total < L_{\max}$ **do**
5: A random number r_1 is generated from the uniform distribution over the interval [0, 1]
6: **if** $r_1 < 1/3$ **then**
7: Generate neighborhood x_l by SM1, compute $\mathbb{C}_{\max}(x_l)$ by algorithm 2
8: **else if** $1/3 \leq r_1 < 2/3$ **then**
9: Generate neighborhood x_l by SM2, compute $\mathbb{C}_{\max}(x_l)$ by algorithm 2
10: **else**
11: Generate neighborhood x_l by SM3, compute $\mathbb{C}_{\max}(x_l)$ by algorithm 2
12: **end if**
13: **if** $\mathbb{C}_{\max}(x_l) \leq \mathbb{C}_{\max}(x_c)$ **then**
14: $x_c = x_l$
15: **if** $\mathbb{C}_{\max}(x_c) \leq \mathbb{C}_{\max}(x_{best})$ **then**
16: $x_{best} = x_c, Change := Change + 1, Unchange = 0$
17: **end if**
18: **else**
19: Randomly generate a number $r_2 \sim U(0, 1)$, let $\Delta = \mathbb{C}_{\max}(x_l) - \mathbb{C}_{\max}(x_c)$
20: **if** $e^{-\Delta/T} > r_2$ **then**
21: $x_c = x_l$
22: **end if**
23: **end if**
24: $Total := Total + 1$
25: **end while**
26: **if** $Change = 0$ **then**
27: $Unchange := Unchange + 1$
28: **end if**
29: **if** $Unchange = 3$ **then**
30: **Return** $\mathbb{C}_{\max}(x_{best})$
31: **end if**
32: $T := \alpha \times T, Total = 0, Change = 0$
33: **end while**
34: **Return** $\mathbb{C}_{\max}(x_{best})$

Table 1. Comparison between SA and HSA with different number of parts (J) ($M = 30$, $W = 25$, $T_{pkmw} \sim DU[5, 20]$, $\mathscr{N}_m \sim DU[1, 3]$, $B_u = 4$).

	SA-$\mathbb{C}_{max}$			HSA-$\mathbb{C}_{max}$						
J	MIN	MAX	AVE	MIN	MAX	AVE	D-$\mathbb{C}_{max}$ (%)	SA-CPU (s)	HSA-CPU (s)	D-CPU (%)
150	1000	1167	1104.5	332	336	335.3	69.64	45.99	32.58	29.17
200	1200	1293	1256.6	314	324	318.0	74.69	101.73	63.76	37.32
250	1305	1427	1368.5	239	268	251.4	81.63	131.55	110.72	15.84
300	1704	1904	1792.3	394	422	408.4	77.21	205.60	169.80	17.41

Table 2. Comparison between SA and HSA with different number of machines (M) ($J = 200$, $W = 20$, $T_{pkmw} \sim DU[5, 20]$, $\mathscr{N}_m \sim DU[1, 10]$, $B_u = 4$).

	SA-$\mathbb{C}_{max}$			HSA-$\mathbb{C}_{max}$						
M	MIN	MAX	AVE	MIN	MAX	AVE	D-$\mathbb{C}_{max}$ (%)	SA-CPU (s)	HSA-CPU (s)	D-CPU (%)
30	1348	1514	1443.1	404	414	407.7	71.75	82.38	63.44	23.00
40	1504	1608	1558.6	433	439	435.2	72.08	80.58	63.41	21.30
50	1133	1251	1184.1	224	230	226.8	80.85	86.64	63.87	26.28
60	1316	1487	1409.1	504	509	506.5	64.06	84.14	63.34	24.72

Table 3. Comparison between SA and HSA with different number of worker types (W) ($M = 50$, $J = 200$, $T_{pkmw} \sim DU[5, 20]$, $\mathscr{N}_m \sim DU[1, 10]$, $B_u = 4$).

	SA-$\mathbb{C}_{max}$			HSA-$\mathbb{C}_{max}$						
W	MIN	MAX	AVE	MIN	MAX	AVE	D-$\mathbb{C}_{max}$ (%)	SA-CPU (s)	HSA-CPU (s)	D-CPU (%)
30	1106	1204	1161.5	256	262	258.1	77.78	87.15	66.81	23.33
40	1005	1203	1100.8	255	260	257.4	76.62	92.61	67.00	27.65
50	952	1133	1064.2	210	210	210.0	80.27	91.03	69.78	23.34
60	1119	1382	1258.1	420	420	420.0	66.62	88.89	66.89	24.75

5. Computational experiments

In order to evaluate the performance of the HSA and SA algorithms for the problem, extensive numerical experiments are conducted. Several impact factors are used for the evaluation, including the number of tasks (J), the number of machines (M), the number of workers (W), upper cell size limit (B_u), the number of machines that can process task m ($\mathscr{N}_m$), and the processing time (T_{pkmw}).

To test the effects of varying J, M, W and B_u, four different values of J are used, including 150,200,250 and 300, four different values of M and W are used, which are 30,40,50,60, and four different values of B_u are used, including 4,6,8 and 10. Moreover, to determine whether the range of T_{pkmw} and $\mathscr{N}_m$ may have any impact on the performance of the HSA algorithm, four different distributions of T_{pkmw} are used, including $T_{pkmw} \sim DU[5, 10]$, $T_{pkmw} \sim DU[5, 20]$, $T_{pkmw} \sim DU[5, 30]$ and $T_{pkmw} \sim DU[5, 40]$, and four different distributions of $\mathscr{N}_m$ are used, including $\mathscr{N}_m \sim DU[1, 4]$, $\mathscr{N}_m \sim DU[1, 6]$, $\mathscr{N}_m \sim DU[1, 8]$ and $\mathscr{N}_m \sim DU[1, 10]$, where DU[a, b] represents a discrete uniform distribution with an integer range from a to b.

Six sets of numerical experiments are conducted. In the first set, J is allowed to vary, given $M = 30$, $W = 25$, $T_{pkmw} \sim DU[5, 20]$, $\mathscr{N}_m \sim DU[1, 3]$, $B_u = 4$. In the second set, M is allowed to vary, given $J = 200$, $W = 20$, $T_{pkmw} \sim DU[5, 20]$, $\mathscr{N}_m \sim DU[1, 10]$, $B_u = 4$. In the third set, W is allowed to vary, given $M = 50$, $J = 200$, $T_{pkmw} \sim DU[5, 20]$, $\mathscr{N}_m \sim DU[1, 10]$, $B_u = 4$. In the fourth set, B_u is allowed to vary, given $M = 40$, $W = 30$, $J = 200$, $T_{pkmw} \sim DU[5, 20]$, $\mathscr{N}_m \sim DU[1, 3]$. In the fifth set, the distribution of generating T_{pkmw} is allowed to vary, given $M = 40$, $J = 200$, $W = 30$, $\mathscr{N}_m \sim DU[1, 2]$, $B_u = 5$. In the sixth set, the distribution of generating $\mathscr{N}_m$ is allowed to vary, given $M = 40$, $J = 200$, $W = 30$, $T_{pkmw} \sim DU[5, 40]$, $B_u = 5$.

In the experiments, we use the approach presented in [25] for generating the precedence constraints of tasks. To do this, let $\mathscr{P}_{ij} = \Pr\{\text{arc}(i, j) \text{ exists in the immediate precedence graph}\}$, and let D represent the target density of the precedence constraint graph, that is $D = \Pr\{\text{arc}(i, j) \text{ exists in the precedence constraint graph}\}$, for $1 \le i < j \le J$. The D and $\mathscr{P}_{ij}$ satisfy:

$$\mathscr{P}_{ij} = \frac{D(1-D)^{j-i-1}}{1 - D(1-(1-D)^{j-i-1})}, \tag{11}$$

where $D \in (0, 1)$.

Randomly generate a number r_{ij} from the uniform distribution over the interval [0, 1]. If $r_{ij} < D$, then arc(i, j) exists in the immediate precedence graph. Given $Y_{mcm'c'} = 0$ for $m = m'$, $Y_{mcm'c'} = 2$ for $c = c'$ and $m \neq m'$, $Y_{mcm'c'} = 20$ for $c \neq c'$, and $D = 0.5$ in these experiments.

The experiments have been performed on a Pentium-based Dell computer with 2.30 GHz clock-pulse and 4.00 GB RAM. The HSA and SA algorithms have been coded in C++, compiled with the Microsoft Visual C++ 9.0 compiler, and tested under Microsoft Windows 7 (32-bit) operating system.

Table 4. Comparison between SA and HSA with different upper cell size limit (B_u) ($M = 40$, $W = 30$, $J = 200$, $T_{pkmw} \sim DU[5, 20]$, $\mathcal{N}_m \sim DU[1, 3]$).

	SA-$\mathbb{C}_{max}$			HSA-$\mathbb{C}_{max}$						
B_u	MIN	MAX	AVE	MIN	MAX	AVE	D-$\mathbb{C}_{max}$ (%)	SA-CPU (s)	HSA-CPU (s)	D-CPU (%)
4	1282	1386	1332.1	391	395	392.6	70.53	83.88	62.47	25.53
6	1036	1183	1110.0	258	263	260.9	76.50	66.66	62.71	5.92
8	1085	1293	1183.8	325	332	327.5	72.33	83.87	62.81	25.11
10	1216	1336	1262.5	327	338	330.3	73.84	74.05	63.85	13.77

Table 5. Comparison between SA and HSA with different processing times (T_{pkmw}) ($M = 40$, $J = 200$, $W = 30$, $\mathcal{N}_m \sim DU[1, 2]$, $B_u = 5$).

	SA-$\mathbb{C}_{max}$			HSA-$\mathbb{C}_{max}$						
T_{pkmw}	MIN	MAX	AVE	MIN	MAX	AVE	D-$\mathbb{C}_{max}$ (%)	SA-CPU (s)	HSA-CPU (s)	D-CPU (%)
DU[5, 10]	851	912	887.9	560	561	560.9	36.83	84.18	61.27	27.22
DU[5, 20]	907	1040	959.3	179	198	186.6	80.55	74.00	62.81	15.12
DU[5, 30]	1683	2016	1847.2	351	356	353.3	80.87	89.28	62.66	29.81
DU[5, 40]	1880	2111	1992.2	288	302	296.7	85.11	86.79	62.53	27.95

Table 6. Comparison between SA and HSA with different number of machines that can process the operation ($\mathcal{N}_m$) ($M = 40$, $J = 200$, $W = 30$, $T_{pkmw} \sim DU[5, 40]$, $B_u = 5$).

	SA-$\mathbb{C}_{max}$			HSA-$\mathbb{C}_{max}$						
$\mathcal{N}_m$	MIN	MAX	AVE	MIN	MAX	AVE	D-$\mathbb{C}_{max}$ (%)	SA-CPU (s)	HSA-CPU (s)	D-CPU (%)
DU[1, 4]	2135	2588	2303.8	397	410	401.0	82.59	99.04	64.11	35.26
DU[1, 6]	2280	2539	2432.3	478	486	483.9	80.11	99.87	65.85	34.06
DU[1, 8]	1713	1923	1840.7	171	177	174.4	90.53	86.95	67.80	22.02
DU[1, 10]	1729	2114	1914.7	342	356	349.2	81.76	88.48	66.04	25.35

Table 7. Statistical t-test results from SPSS for sample entries 1–12.

						Paired Differences				
		Samples Statistics					99% Confidence Interval			
The *n*th Sample Entry	Algorithm	Mean	Std. Deviation	Std. Error Mean	Pair	Mean	Lower	Upper	T-Value	P-Value
1	SA	1104.50	49.646	15.699						
	HSA	335.30	1.337	0.423	SA–HSA	769.200	718.432	819.968	49.239	0.000
2	SA	1256.60	27.549	8.712						
	HSA	318.00	3.232	1.022	SA–HSA	938.600	911.774	965.426	113.707	0.000
3	SA	1368.50	37.340	11.808						
	HSA	251.40	9.606	3.038	SA–HSA	1117.100	1075.718	1158.482	87.729	0.000
4	SA	1792.30	61.651	19.496						
	HSA	408.40	8.884	2.810	SA–HSA	1383.900	1316.473	1451.327	66.701	0.000
5	SA	1443.10	53.702	16.982						
	HSA	407.70	2.869	0.907	SA–HSA	1035.400	978.845	1091.955	59.497	0.000
6	SA	1558.60	36.467	11.532						
	HSA	435.20	2.486	0.786	SA–HSA	1123.400	1084.898	1161.902	94.823	0.000
7	SA	1184.10	39.411	12.463						
	HSA	226.80	2.300	0.727	SA–HSA	957.300	915.665	998.935	74.722	0.000
8	SA	1409.10	57.311	18.123						
	HSA	506.50	1.716	0.543	SA–HSA	902.600	843.669	961.531	49.775	0.000
9	SA	1161.50	35.877	11.345						
	HSA	258.10	1.853	0.586	SA–HSA	903.400	866.384	940.416	79.313	0.000
10	SA	1100.80	63.215	19.990						
	HSA	257.40	1.578	0.499	SA–HSA	843.400	778.770	908.030	42.410	0.000
11	SA	1064.20	58.284	18.431						
	HSA	210.00	0.000	0.000	SA–HSA	854.200	794.302	914.098	46.345	0.000
12	SA	1258.10	83.043	26.260						
	HSA	420.00	0.000	0.000	SA–HSA	838.100	752.758	923.442	31.915	0.000

Note: Sample size of each pair $N = 10$, degree of freedom df $= 9$, significancelevel $= 0.01$

The six experiment results are presented in Tables 1–6, respectively. Let HSA-$\mathbb{C}_{max}$ and SA-$\mathbb{C}_{max}$ denote the objective function values (i.e. makespan) of the problem using the HSA and SA, respectively. Each table entry represents the minimum, maximum and average of its associated 10 instances. D-$\mathbb{C}_{max}$ denotes the declining percentage of average HSA-$\mathbb{C}_{max}$ over average SA-$\mathbb{C}_{max}$. Let HSA-CPU and SA-CPU denote the mean CPU time of the HSA and SA algorithms without including input and output time,

Table 8. Statistical t-test results from SPSS for sample entries 13–24.

The *n*th Sample Entry	Algorithm	Samples Statistics			Pair	Paired Differences			T-Value	P-Value
							99% Confidence Interval			
		Mean	Std. Deviation	Std. Error Mean		Mean	Lower	Upper		
13	SA	1332.10	37.323	11.802						
	HSA	392.60	1.265	0.400	SA–HSA	939.500	901.007	977.993	79.319	0.000
14	SA	1110.00	46.985	14.858						
	HSA	260.90	1.792	0.567	SA–HSA	849.100	800.908	897.292	57.259	0.000
15	SA	1183.80	69.262	21.903						
	HSA	327.50	2.506	0.792	SA–HSA	856.300	785.098	927.502	39.084	0.000
16	SA	1262.50	35.120	11.106						
	HSA	330.30	2.983	0.943	SA–HSA	932.200	898.085	966.315	88.803	0.000
17	SA	887.90	20.344	6.433						
	HSA	560.90	0.316	0.100	SA–HSA	327.000	306.236	347.764	51.180	0.000
18	SA	959.30	42.266	13.366						
	HSA	186.60	5.420	1.714	SA–HSA	772.700	732.839	812.561	62.997	0.000
19	SA	1847.20	103.971	32.878						
	HSA	353.30	1.767	0.559	SA–HSA	1493.900	1386.208	1601.592	45.082	0.000
20	SA	1992.20	91.742	29.011						
	HSA	296.70	4.473	1.415	SA–HSA	1695.500	1599.442	1791.558	57.362	0.000
21	SA	2303.80	130.843	41.376						
	HSA	401.00	4.447	1.406	SA–HSA	1902.800	1766.682	2038.918	45.430	0.000
22	SA	2432.30	87.323	27.614						
	HSA	483.90	3.143	0.994	SA–HSA	1948.400	1857.460	2039.340	69.628	0.000
23	SA	1840.70	73.314	23.184						
	HSA	174.40	2.271	0.718	SA–HSA	1666.300	1589.943	1742.657	70.920	0.000
24	SA	1914.70	113.355	35.846						
	HSA	349.20	4.237	1.340	SA–HSA	1565.500	1447.051	1683.949	42.952	0.000

Note: Sample size of each pair $N = 10$, degree of freedom df $= 9$, significancelevel $= 0.01$

respectively. Let D-CPU denote the declining percentage of HSA-CPU over SA-CPU, i.e. D-CPU = (SA-CPU−HSA-CPU)/SA-CPU.

From the obtained results, we can see that D-$\mathbb{C}_{\max}$ reaches 36.83%–90.53% and D-CPU reaches 5.92%–37.32%. That is to say, the HSA performs more accurately and efficiently than the SA in spite of the variation of six impact factors J, M, W, B_u, $\mathscr{N}_m$ and T_{pkmw}. The reason lies in that the selection method for the prior job, machine and worker in the PRBHA algorithm may be effective to get a good initial solution.

The paired-samples t-test experiment is conducted in SPSS (Statistical Product and Service Solutions) software, so that the performance of HSA and SA can be further validated. Tables 7–8 display the experimental results according to 24 sample entries in Tables 1–6. For example, the first entry shows that t-value is 49.239 ($> t_{0.01}(9) = 2.8214$), and p-value is 0.000 ($< \alpha = 0.01$). Since all t-values and p-values satisfy the conditions, the HSA is significantly better than the SA with respect to solution quality in statistics.

6. Conclusions

This paper gives a new optimization model of cellular manufacturing system (CMS) under dual resources and task precedence constrained setting. The objective of the problem is to minimize the makespan. The PRBHA, which schedules a prior task on a prior machine by a prior worker according to the priority rule at each iteration, is embedded to the HSA for initial feasible solution that can be improved in further stages. Computational experiments are conducted to show that the quality of results obtained by the HSA is better than the SA regardless of the variation of some important parameters.

A valuable future research direction is to consider the impact of learning and forgetting effects of workers on their assignment and movement. The other possible extension to this research would investigate various efficient priority rules and corresponding heuristics for SA. It is also desired to linearize the proposed model in the future, so that the HSA can be compared with the branch-and-bound approach (B&B) under the ILOG CPLEX software for small or medium sized instances. Moreover, some state-of-the-art heuristics, such as firefly algorithm, league championship algorithm, and migrating birds optimization, can be developed from various aspects on the basis of CMS characteristics. Therefore, we wish to design these heuristic based HSA approaches, and compare them with the proposed HSA in this paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was supported by the Zhejiang Provincial Natural Science Foundation of China under Grants LY19A010007 and LY19G020015, the Humanities and Social Sciences Foundation of the PRC Ministry of Education under Grants 19YJA630078 and 17YJC630093, and the National Natural Science Foundation of China under Grant 71871076.

References

[1] Rabbani M, Jolai F, Manavizadeh N, et al. Solving a bi-objective cell formation problem with stochastic production quantities by a two-phase fuzzy linear programming approach. Int J Adv Manuf Technol. 2012;58(5–8):709–722.

[2] Arkat J, Abdollahzadeh H, Ghahve H. A new branch and bound algorithm for cell formation problem. Appl Math Model. 2012;36(10):5091–5100.

[3] Saraç T, Ozcelik F. A genetic algorithm with proper parameters for manufacturing cell formation problems. J Intell Manuf. 2012;23(4):1047–1061.

[4] Chung S-H, Wu T-H, Chang C-C. An efficient tabu search algorithm to the cell formation problem with alternative routings and machine reliability considerations. Comput Ind Eng. 2011;60(1):7–15.

[5] Rafiei H, Rabbani M, Nazaridoust B, et al. Multi-objective cell formation problem considering work-in-process minimization. Int J Adv Manuf Technol. 2015;76(9–12):1947–1955.

[6] Mar-Ortiz J, González-Velarde JL, Adenso-Díaz B. A VNS algorithm for a disassembly cell formation problem with demand variability. Eur J Ind Eng. 2014;8(1):22–49.

[7] Jayakumar V, Raju R. A simulated annealing algorithm for machine cell formation under uncertain production requirements. Arab J Sci Eng. 2014;39(10):7345–7354.

[8] Rafiee K, Rabbani M, Rafiei H, et al. A new approach towards integrated cell formation and inventory lot sizing in an unreliable cellular manufacturing system. Appl Math Model. 2011;35(4):1810–1819.

[9] Boutsinas B. Machine-part cell formation using biclustering. Eur J Oper Res. 2013;230(3):563–572.

[10] Tavakkoli-Moghaddam R, Ranjbar-Bourani M, Amin GR, et al. A cell formation problem considering machine utilization and alternative process routes by scatter search. J Intell Manuf. 2012;23(4):1127–1139.

[11] Mahdavi I, Aalaei A, Paydar MM, et al. Multi-objective cell formation and production planning in dynamic virtual cellular manufacturing systems. Int J Prod Res. 2011;49(21):6517–6537.

[12] Bagheri M, Bashiri M. A new mathematical model towards the integration of cell formation with operator assignment and inter-cell layout problems in a dynamic environment. Appl Math Model. 2014;38(4):1237–1254.

[13] Mahdavi I, Aalaei A, Paydar MM, et al. Designing a mathematical model for dynamic cellular manufacturing systems considering production planning and worker assignment. Comput Math Appl. 2010;60(4):1014–1025.

[14] Azadeh A, Sheikhalishahi M, Koushan M. An integrated fuzzy DEA-Fuzzy simulation approach for optimization of operator allocation with learning effects in multi products CMS. Appl Math Model. 2013;37(24):9922–9933.

[15] Süer GA, Cosner J, Patten A. Models for cell loading and product sequencing in labor-intensive cells. Comput Ind Eng. 2009;56(1):97–105.

[16] Bootaki B, Mahdavi I, Paydar MM. New bi-objective robust designbased utilisation towards dynamic cell formation problem with fuzzy random demands. Int J Comput Integ M. 2015;28(6):577–592.

[17] Venkataramanaiah S. Scheduling in cellular manufacturing systems: an heuristic approach. Int J Prod Res. 2008;46(2):429–449.

[18] Tavakkoli-Moghaddam R, Javadian N, Khorrami A, et al. Design of a scatter search method for a novel multi-criteria group scheduling problem in a cellular manufacturing system. Expert Syst Appl. 2010;37(3):2661–2669.

[19] Halat K, Bashirzadeh R. Concurrent scheduling of manufacturing cells considering sequence-dependent family setup times and intercellular transportation times. Int J Adv Manuf Technol. 2015;77(9–12):1907–1915.

[20] Arkat J, Farahani MH, Ahmadizar F. Multi-objective genetic algorithm for cell formation problem considering cellular layout and operations scheduling. Int J Comput Integ M. 2012;25(7):625–635.

[21] Liu C, Wang J, Leung JY-T, et al. Solving cell formation and task scheduling in cellular manufacturing system by discrete bacteria foraging algorithm. Int J Prod Res. 2016;54(3):923–944.

[22] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science. 1983;220(4598):671–680.

[23] Liu S, Wu W-H, Kang C-C, et al. A single-machine two-agent scheduling problem by a branch-and-bound and three simulated annealing algorithms. Discrete Dyn Nat Soc. 2015;2015:1–8.

[24] Liu B, Chen H, Li Y, et al. A pseudo-parallel genetic algorithm integrating simulated annealing for stochastic location-inventory-routing problem with consideration of returns in e-commerce. Discrete Dyn Nat Soc. 2015;2015:1–15.

[25] Hall NG, Posner ME. Generating experimental data for computational testing with machine scheduling applications. Oper Res. 2001;49(7):854–865.