Int J Adv Manuf Technol (2013) 68:2383–2391
DOI 10.1007/s00170-013-4858-4

ORIGINAL ARTICLE

Makespan minimization in a job shop with a BPM using simulated annealing

Miguel Rojas-Santiago · Purushothaman Damodaran · Shanthi Muthuswamy · Mario C. Vélez-Gallego

Received: 2 April 2012 / Accepted: 21 February 2013 / Published online: 8 March 2013

Abstract A scheduling problem commonly observed in the metal working industry has been studied in this research effort. A job shop equipped with one batch processing machine (BPM) and several unit-capacity machines has been considered. Given a set of jobs, their process routes, processing requirements, and size, the objective is to schedule the jobs such that the makespan is minimized. The BPM can process a batch of jobs as long as its capacity is not exceeded. The batch processing time is equal to the longest processing job in the batch. If no batches were to be formed, the scheduling problem under study reduces to the classical job shop problem with makespan objective, which is known to be nondeterministic polynomial time-hard. A network representation of the problem using disjunctive and conjunctive arcs, and a simulated annealing (SA) algorithm are proposed to solve the problem. The solution quality and run time of SA are compared with CPLEX, a commercial solver used to solve the mathematical formulation and with four dispatching rules. Experimental study clearly highlights the advantages, in terms of solution quality and run time, of using SA to solve large-scale problems.

Keywords Scheduling · Job shop · Batch processing · Simulated annealing · Makespan

M. Rojas-Santiago (✉)
Departamento de Ingeniería Industrial, Universidad del Norte,
Km. 5 Vía Puerto Colombia,
Barranquilla, Colombia
e-mail: miguelrojas@uninorte.edu.co

P. Damodaran
Industrial and Systems Engineering, Northern Illinois University,
Dekalb, IL 60115, USA
e-mail: pdamodaran@niu.edu

S. Muthuswamy
Department of Technology, Northern Illinois University,
Dekalb, IL 60115, USA
e-mail: smuthuswamy@niu.edu

M. C. Vélez-Gallego
Departamento de Ingeniería de Producción, Universidad EAFIT,
Carrera 49 #7 sur 50,
Medellín, Colombia
e-mail: marvelez@eafit.edu.co

1 Introduction

Most scheduling research on job shops considers discrete or unit-capacity machines (i.e., each machine can process one job at a time). However, in this paper, we consider a scheduling problem that is commonly observed in many manufacturing facilities—job shops with both discrete processing and batch processing machines (BPMs).

This research is motivated by a practical application observed at a metal working industry. Many fabrication facilities have not only discrete processing machines, but also BPMs. For example, metal working companies which make boilers, pressure vessels, and heat exchangers, use both discrete processing machines (such as press brake, bending machines, and other forming equipment) and BPM (such as a furnace). The discrete processing machines are commonly used to shape the metal (i.e., cut, bend, or punch), and the BPMs are commonly used to heat treat semifinished parts. The primary objective of this research effort is to schedule both discrete machines and a BPM in a job shop so that the makespan or the completion time of the last job is minimized. By minimizing the makespan, the utilization of the machines can be improved.

Based on our interactions with the metal working industry, the problem under study can be explained as follows. There are several discrete processing machines and one BPM in the facility. The BPM can process a batch of jobs as long as the total number of jobs and total size of all the jobs in a batch do not exceed the machine's capacity. The discrete processing machines can process one job at any

time. Each job has its own process route. The processing times of the jobs on each machine are known. The scheduler needs to schedule the jobs on both the discrete and BPMs such that the completion time of the last job or the makespan is minimized. When scheduling the BPMs, the scheduler needs to identify which jobs can be grouped into a batch and in what sequence the batches need to be scheduled. When considering jobs to be grouped in a batch, the ready times of the jobs need to be taken into account. As the jobs can follow unique routes, the completion time of each job at the machine preceding the BPM is taken as the ready time of the job. When jobs are grouped in a batch, the batch processing time is equal to the longest processing time of all the jobs in the batch. The BPM can process a batch only when all the jobs in the batch are ready (i.e., batch ready time is equal to the latest ready time of all the jobs in the batch). The scheduler faces a dilemma when it comes to batching jobs; waiting too long for a job can delay the start time of a batch, but scheduling too many batches can lead to longer completion times as the processing times at the BPM is typically longer. The objective of this research is to develop a solution approach which is capable of solving large problems in a short time with good solution quality and is easy to implement.

2 Problem description

The problem under study can be formally stated as follows. Given a set of jobs (J) and a set of machines (M), the objective is to schedule the jobs on the machines such that the makespan is minimized. The set M includes one BPM and several discrete processing machines. The route or processing sequence (π_j) for each job $j \in J$, its processing time on each machine (p_{mj}), and size (s_j) is known. The BPM can process a batch of D jobs simultaneously, where D is the capacity of the machine. For example, D could be the number of fixtures available. The batch is also constrained by the size of the jobs. The BPM can process a batch of jobs only when the total size of all the jobs in a batch does not exceed S, where S is the capacity of the machine along one of its critical dimension. When several jobs are batched together, the processing time of the batch is equal to the job which requires the largest amount of processing among all the jobs in the batch. The ready time of the batch is equal to the latest ready time of the job in the batch.

Table 1 presents an instance for the problem under study. There are three jobs and four machines in this instance. Here, machine 1 is assumed as the BPM, and machines 2, 3, and 4 are considered to be discrete processing machines. Column 2 shows the machining sequence for each job. Column 3 gives the processing time of the jobs in their respective machines. Column 4 provides the size of each job. Suppose $D=2$ (i.e., the total number of jobs allowed in a batch cannot exceed 2) and $S=10$ (i.e., the total size of all jobs in a batch cannot exceed 10), then there are several ways to form feasible batches. For example, the jobs can be processed individually (i.e., no batching—see Fig. 1) or batched (see Fig. 2).

Table 1 Data for a three-job and four-machine problem

Jobs	Machining sequence	Processing time	s_j
1	1→2→3	$p_{11}=9$, $p_{21}=8$, $p_{31}=4$	3
2	1→2→4	$p_{12}=5$, $p_{22}=6$, $p_{42}=3$	7
3	3→1→2	$P_{33}=10$, $p_{13}=4$, $p_{23}=9$	5

Figure 1 presents the network representation of the example problem instance shown in Table 1 when all the machines are assumed to be discrete processing machines. In Fig. 1, each node (except the source and sink nodes) represents an operation. The solid arcs (or conjunctive arcs) represent the precedence constraint (or process route) for each job. The dotted arcs (or disjunctive arcs) are used to form cliques. There is one clique for each machine indicating all the jobs that need to be processed on that machine. In Fig. 1, there is no clique for machine 4 as there is only one job that is processed in it. All arcs (both disjunctive and conjunctive) emanating from a node have a length equal to the processing time of the operation that is represented by the node. Shifting bottleneck heuristic is a well-known heuristic to minimize makespan in a job shop. It is based on the above given network representation. For more details on the network construction and shifting bottleneck heuristic, we refer the reader to [1].

Figure 2 presents the network for the example problem instance shown in Table 1 when machine 1 is assumed to be a BPM. Figure 2 shows two feasible ways to form batches:

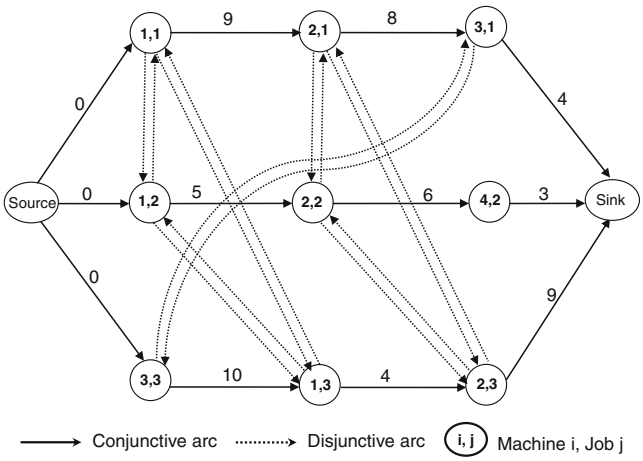

Fig. 1 A three-job and four-machine problem representation when none of the jobs are batched

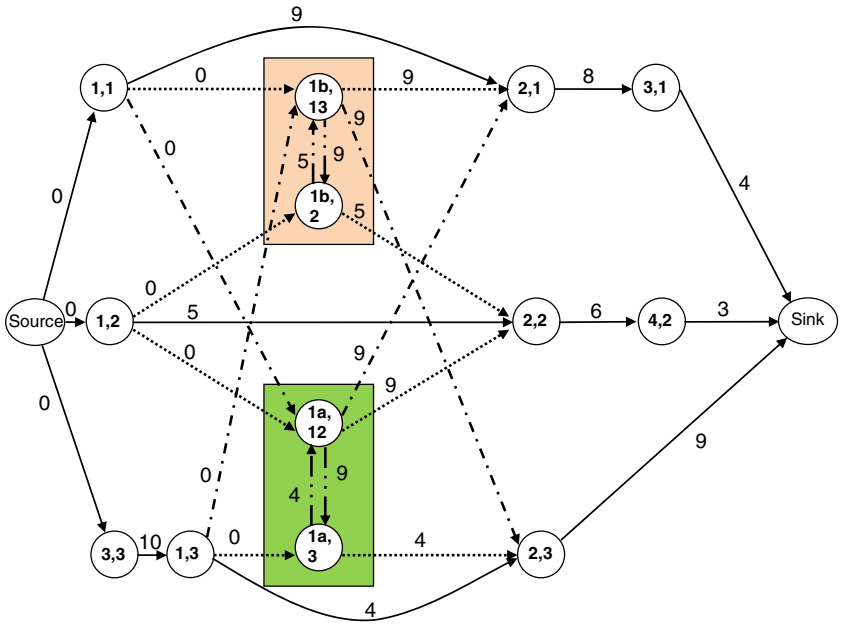

Fig. 2 A three-job and four-machine problem representation when two possible batch formations *a* (j_1 and j_2, j_3) and *b* (j_1 and j_3, j_2) are considered on machine 1

option a—jobs 1 and 2 can be batched and job 3 is processed individually (i.e., $\{j_1$ and $j_2, j_3\}$) and option b—jobs 1 and 3 can be batched and job 2 is processed individually (i.e., $\{j_1$ and $j_3, j_2\}$). In Fig. 2, for the sake of clarity, only the disjunctive arcs that belong to jobs related with the BPM are shown. Two rectangles are used to denote two possible batch formations; the inside of each rectangle has two nodes representing operations in the BPM. The first feasible batch formation is denoted by appending *a* to the node representing "dummy" machine 1 in which the job is processed. Similarly, *b* is used to denote the second batching option. The shifting bottleneck heuristic was applied on this network to find a feasible solution. The Cmax value for this example problem was found to be 32. The final solution from the shifting bottleneck heuristic is shown in Fig. 3 (i.e., red thicker line). As the number of jobs increase, the different combinations in which jobs can be batched also increase. This would result in a bigger network with a longer run time to find the solution.

The problem under study can be represented using the standard three field notation as *Jm|batch|Cmax*. It is well known that *Jm||Cmax* is nondeterministic polynomial time (NP)-hard. When the BPM capacity is adjusted such that it can process one job at a time, then the problem under consideration will reduce to the classical job shop problem. Since the special case of the problem under study is a well-known NP-hard problem, the problem under study is also NP-hard. The research objective is to develop a solution approach which takes

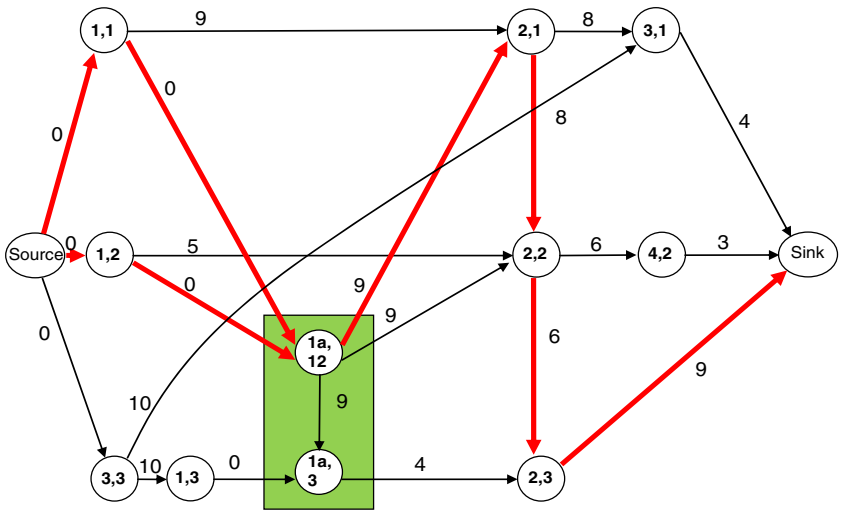

Fig. 3 Graphical solution for the three-job and four-machine problem with machine 1 as BPM

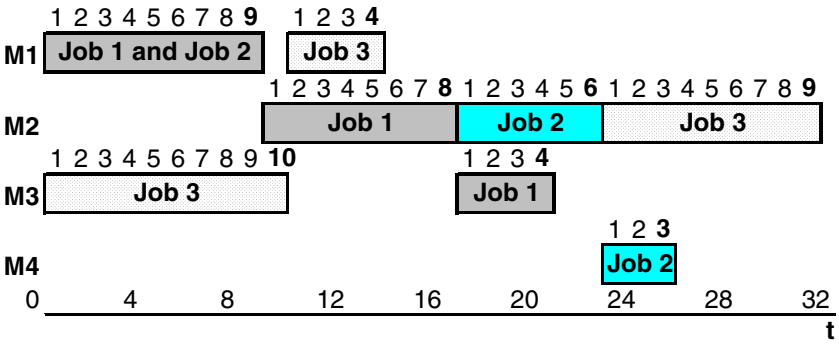

Fig. 4 Gantt chart for the three-job and four-machine problem with machine 1 as BPM

less time to find a good solution and is easy to implement.

3 Literature review

Most research on job shop scheduling to date has focused on unit-capacity machines. Job shop with BPMs has received very little attention. Unit-capacity machines can process one job at a time, while BPMs can process a number of jobs simultaneously as a batch. Some operations research analysts and engineers call these production systems as complex job shops [2], because they are characterized by different types of work centers consisting of multiple identical machines with one or more BPMs. An example of a complex job shop is a wafer fabrication facility, where integrated circuits are fabricated on silicon wafers using a variety of chemical and thermal processes [3].

Majority of research on complex job shop scheduling has been carried out in the semiconductor industry. Gupta and Sivakumar [4] have classified the research into four categories based on the solution approaches used: dispatching rules, analytical methods, heuristics, and artificial intelligence techniques. More specifically, decomposition methods based on the shifting bottleneck heuristic [5] and Lagrangian relaxation [6] are the most popular techniques applied to this kind of manufacturing environment.

The best efforts to implement a solution using decomposition methods were due to Mason et al. [7], Mönch and Driessel [8], and Uzsoy and Wang [9]. They proposed a heuristic called modified shifting bottleneck to minimize the total weighted tardiness and the maximum lateness, which was later compared with several dispatching rules [10], and with the Theory of Constraints [11]. Another technique used in complex job shops is Lagrangian relaxation with dynamic programming by Sun et al. [12] and Kaskavelis and Caramanis [13] with $\sum w_j T_j$ as the main objective.

Jm|batch|Cmax has not received due attention in the literature. Moreover, most of the research in complex job shop used total weighted tardiness as a performance measure; however, Pfund et al. [14] developed a desirability function to solve a multicriteria scheduling problem which combined different objectives, like Cmax, $\sum C_j$, and $\sum w_j T_j$. Damodaran and Rojas [15] proposed a mathematical model for the *Jm|batch|Cmax* problem. Commercial solvers require prohibitively long run times to solve these models for larger problem instances. This paper presents a simulated annealing approach and highlights its advantages over using the commercial solvers, both in terms of solution quality and run time, especially on large problem instances, through an experimental study.

4 Solution approach: simulated annealing

In order to schedule the jobs on the BPM in a job shop, jobs are first grouped into batches such that the capacity of the BPM machine is not violated. The total number of jobs in the batch cannot exceed D, and the total size of all the jobs in a batch cannot exceed S. If the BPM is not the first machine for a job to visit, then the job ready time should be taken into account. The ready time is the completion time of the job in the machine preceding the BPM. The ready time and the processing time of the batch depend upon the jobs in the batch. The ready

Table 2 Sizes and processing times for the BPM for six jobs with six machines problem instance

Job	1	2	3	4	5	6
Size	5	1	9	5	7	8
P_j	3	10	9	5	3	10

Table 3 Composition of each batch using FF along with MWKR

Batch	1	2	3	4
Jobs	{1, 4}	{3}	{2, 6}	{5}
Total size	10	9	9	7
P_j	5	9	10	3

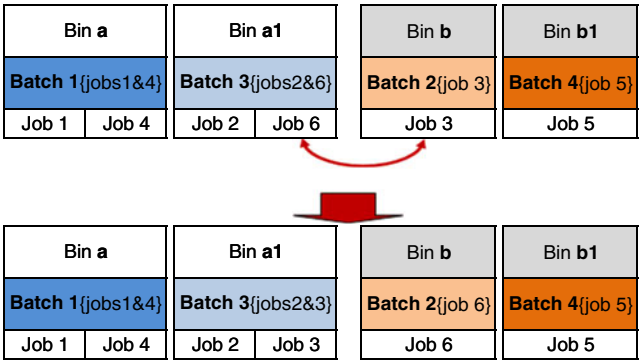

Fig. 5 Example of swapping two jobs

time and processing time of the batch will be equal to the longest ready time and processing time of the jobs in the batch, respectively. Consequently, when forming batches, the ready times of the jobs should be taken into account. In order to minimize the number of batches formed and to improve the utilization of the BPM, it may be better to form as little batches as possible, but waiting too long for a job to be ready can delay the completion time of the entire batch.

Simulated annealing (SA) is a widely used metaheuristic that enables the search process to escape from a local optimum. This technique was adopted from metallurgy, where the metal is heated to a very high temperature and then cooled very slowly to improve its strength. At any given temperature T, during this process, the energy level is fluctuating, so a cooling schedule should be adopted (i.e., the temperature is decreased in a regular fashion or decremented by a certain value that results in a new state). Since SA's introduction several decades ago [16, 17], it has been applied successfully to many scheduling problems including the classical job shop problem [18] and flexible job shop problem [19].

SA starts with an initial solution $F(x)$ and iteratively improves it. At each iteration, the process moves from one schedule to another (i.e., it moves from the current trial solution to an immediate neighbor in the local neighborhood of this solution). In order to form the batches, three different heuristics are considered: (1) modified delay (MD) [20], (2) first fit (FF) [21], and (3) the modified first fit decreasing (MFFD) heuristic. The last approach, MFFD, is a modification of a prominent heuristic used to solve the bin-packing problem, called first fit decreasing (FFD) heuristic [21]. To obtain an initial solution for SA, two dispatching rules, most work remaining (MWKR) and most operations remaining (MOPNR) [22] were used with MD, FF, and MFFD.

Table 4 Composition of each batch after exchanging job 6 with job 3

Batch	1	2	3	4
Jobs	{1, 4}	{6}	{2, 3}	{5}
Total size	10	8	10	7
P_j	5	10	10	3

Table 5 Results for job instances with less than or equal to 100 operations

Instance	Run code	Cmax			% GAP
		DR	SA	CPLEX	
ft06	6j06m_1	60	53	53	0 %
la01	10j05m_1	858	666	666	0 %
la02	10j05m_2	838	699	655	6.7 %
la03	10j05m_3	715	602	586	2.7 %
la06	15j05m_1	982	926	926	0 %
la07	15j05m_2	1,052	904	890	1.6 %
la08	15j05m_3	1,014	863	863	0 %
abz5	10j10m_1	1,462	1,279	1,238	3.3 %
ft10	10j10m_2	1,260	988	897	10.1 %
la16	10j10m_3	1,255	1,041	945	10.2 %
ft20	20j05m_1	1,672	1,309	1,194	9.6 %
la11	20j05m_2	1,297	1,222	1,222	0 %
la12	20j05m_3	1,179	1,027	1,027	0 %

To illustrate how an initial solution is found, consider the data shown in Table 1. Batching heuristic FFD along with MWKR is used to find an initial solution. The first step is to form batches with FFD, which sorts the jobs in decreasing order by s_j (i.e., jobs 2, 3, and 1); then, it batches jobs 2 and job 1; job 3 is processed individually. Jobs 2 and 3 cannot be on the same batch as the total size of the batch would violate the machine capacity of 10. Once the batches are formed for the BPM, the last step is to obtain a sequence for each machine using MWKR; the best job sequence for each machine is as shown in Fig. 4. Note that for the BPM (machine 1), jobs 1 and 2 should be scheduled at the same time and job 3 should be processed individually.

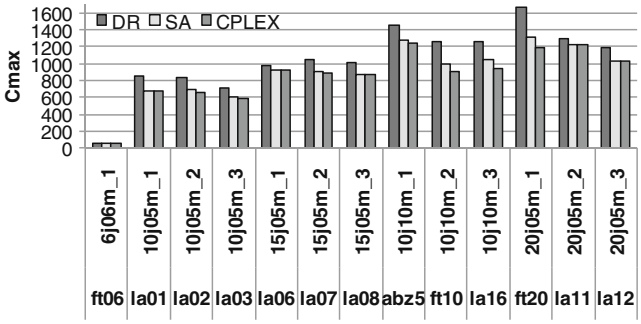

Fig. 6 Results for job instances with less than or equal to 100 operations

Preliminary experimentation was used to guide the selection of the parameters of the algorithm. Initially, a relatively large value of T, cooling rate α, and the desired number of iterations were specified. The rest of details are provided in the following pseudo code for the SA heuristic:

1. Initialize heuristic parameters (Total iterations, Initial Temperature (T), $0 \leq \alpha \leq 1$).
2. Obtain an initial solution:
   - Form batches using a heuristic like Modified Delay, FF, or MFFD.
   - Select an initial neighborhood x using heuristics like MWKR or MOPNR.
   - Calculate the initial *Cmax* or *F(x)*, and set *Best F(x)= Cmax*.
3. Obtain a neighboring solution using a local search technique.
   For each machine, do the following:
   If Machine is a BPM
   Classify batches with two or more jobs as bin "*a*," and with one job as bin "*b*."
   Generate a random number, and depending on the value,
   i. Select a job from bin "*a*," and swap it with another job in bin "*a+j*," or in bin "*b*." (*j*=1,2...B-1). B= Total bins with two or more jobs.
      - *If* the capacity of BPM is violated,
        go to the next bin, and repeat the previous step;
        else,
        go to next step.
        end
      - Continue the job swapping process until the last batch or bin in the sequence has been reached.

   ii. Assign k = random number, and h = random number.
   Generate a random number, and depending on the value,
      - Remove a bin from position k, and insert it in position h.
      - Swap a bin from position k, and replace it with a bin in position h.
      - Exchange a bin in position k, with a bin in the adjacent position ($k+1$).

   End
   If Machine is not a BPM
   Set k = random number, and h = random number
   Generate a random number, and depending on the value,
      - Remove a job from position k, and insert it in position h.
      - Swap a job from position k, and replace it with a job in position h.
      - Exchange a job in position k, with a job in the adjacent position ($h+1$).

   End
4. Calculate the makespan of the neighboring solution $F(x')$.
5. Compare $F(x')$ to the current solution.
   If $F(x') < F(x)$
   Replace current neighborhood (x) by the new neighborhood (x'), $x = x'$.
   Set $F(x) = F(x')$.
   Else,
   $\Delta = F(x') - Best\ F(x)$.
   calculate probability of acceptance $p = e^{(-\Delta/T)}$.
   If $p >$ random number
   Accept the move, $F(x) = F(x')$.
   Else
   Reject the move.
   Check for another neighboring solution.
   End
   End
6. Update counters and parameters ($T = \alpha * T$).
7. Repeat steps 3-6 until stopping criteria is met.

The implementation of step 3 (local search) of the pseudo code is illustrated with an example with six jobs and six machines (refer to Fisher and Thompson ft06 job shop instance for more details [23]). The initial solution required

Table 6 Results for job instances with greater than 100 and less than 300 operations

Instance	Run code	Cmax			% GAP
		DR	SA	CPLEX	
la21	15j10m_1	1,427	1,183	1,148	3.0 %
la22	15j10m_2	1,360	1,035	953	8.6 %
la23	15j10m_3	1,469	1,109	1,032	7.5 %
la26	20j10m_1	1,752	1,399	1,444	−3.1 %
la27	20j10m_2	1,791	1,422	1,497	−5.0 %
swv01	20j10m_3	2,436	1,887	1,636	15.3 %
la36	15j15m_1	1,677	1,388	1,369	1.4 %
la37	15j15m_2	2,098	1,593	1,470	8.4 %
la38	15j15m_3	1,698	1,417	1,287	10.1 %

to start the algorithm is constructed as follows: (1) the jobs are scheduled using MWKR; (2) then, batches are formed using FF. For this example, machine 1 is the BPM, S is equal to 10, and D is equal to 3. Table 2 shows the size for each job and the processing times for the BPM. Before batching, the initial job sequence for machine 1 is jobs 4, 1, 3, 6, 2, and 5. By applying FF along with MWKR, four batches were formed (see Table 3), with a Cmax of 69. This makespan was reduced by ten units (Cmax=59) when job 6 from batch 3 and job 3 from batch 2 were swapped in machine 1 (see Fig. 5 and Table 4).

5 Computational experiments

To test the efficacy of SA, several benchmark job shop instances were chosen from the OR Library [23]. The instances available were for the classical job shop problem with unit-capacity machines. For this study, the BPM capacities (S and D) were assumed to be equal to 10 and 3, respectively. The job sizes were sampled from a discrete uniform (DU) random variable, $s_j \sim \mathrm{DU}[1, S]$. The instances were renamed as njkm_c, where n is number of jobs, k is number of machines, and c is the instance number. For example, 6j06m_1 means that this is the first problem instance (c=1) with six jobs (n=6) and six machines (k=6).

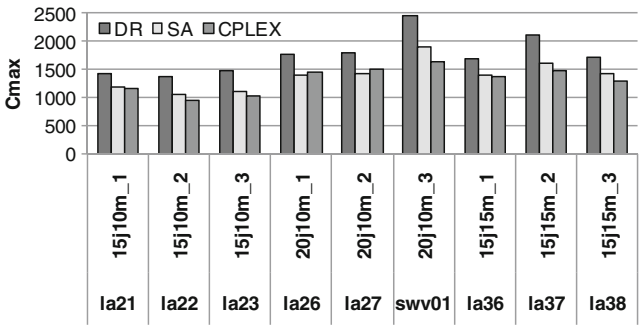

Fig. 7 Results for job instances with greater than 100 and less than 300 operations

Table 7 Results for job instances with 300 operations

Instance	Run code	Cmax			% GAP
		DR	SA	CPLEX	
abz7	20j15m_1	893	800	862	−7.2 %
abz8	20j15m_2	1003	796	868	−8.3 %
swv06	20j15m_3	2,709	2,165	2,536	−14.6 %
la31	30j10m_1	2,215	1,889	2,102	−10.1 %
la32	30j10m_2	2,421	1,946	2,180	−10.7 %
la33	30j10m_3	2,139	1,765	1,913	−7.7 %

SA was implemented using Matlab 7.0, and the experiments were conducted on a Core Duo PC, clocked at 1.86 GHz with 1 GB of RAM. Each problem instance was run three times based on which machine (machine 1, 2, or 3) was considered as a BPM. The results of SA in terms of solution quality and run time were compared against four dispatching rules and the CPLEX mathematical model proposed by Damodaran and Rojas [15]. The dispatching rules were shortest processing time, critical ratio, MWKR, and MOPNR [24]. Given that the branch and bound approach used by CPLEX may take a prohibitively long computational time to report the optimum solution, the run time was restricted initially to 1,800 s for each instance. However, for instances with more than 150 operations [15 jobs×10 machines], it was necessary to modify the specified run time. CPLEX was allowed to run until it reported a feasible solution. Some instances took approximately 28,800 s (8 h) to find a solution. However, on two problem instances, CPLEX did not report a feasible solution even after running for 8 h.

The parameters of SA, after fine tuning in the experimentation phase, were set to the following values: total number of iterations=60, initial temperature=100, and α=0.05. To develop this algorithm, two dispatching rules (MWKR and

Table 8 Results for job instances with more than 300 (up to 500) operations

Instance	Run code	Cmax			% GAP
		DR	SA	CPLEX	
yn1	20j20m_1	1,229	1,092	1,131	−3.4 %
yn2	20j20m_2	1,232	1,077	979	10.0 %
yn3	20j20m_3	1,268	1,083	1,364	−20.6 %
swv11	50j10m_1	5,324	4,373	4,661	−6.2 %
swv12	50j10m_2	5,167	4,505	5,387	−16.4 %
swv13	50j10m_3	5,767	4,623	5,250	−11.9 %

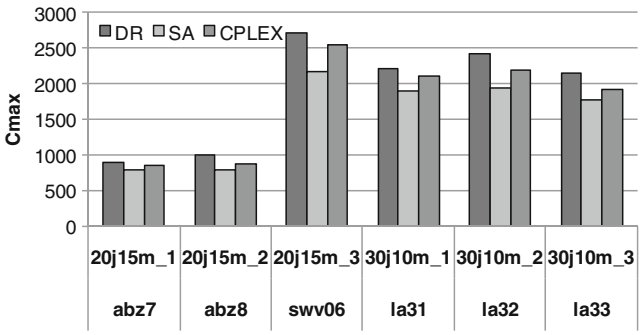

Fig. 8 Results for job instances with 300 operations

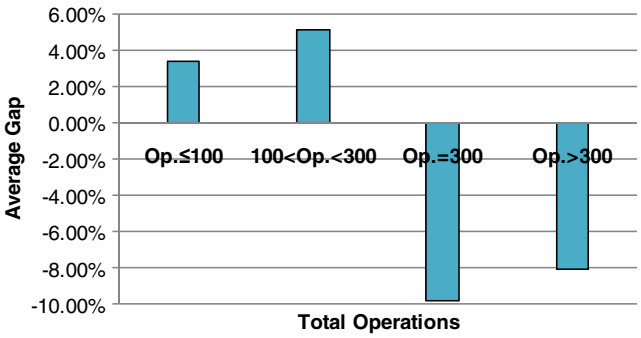

Fig. 10 Average gap vs. total operations

MOPNR) with three batch forming heuristics (MD, FF, MFFD) were combined and used as the initial solution. Through this combination, six approaches of the simulated annealing algorithm were developed. The best solution ($Cmax^{SA}$) of these approaches was recorded, and the percentage difference in solution (or gap) between SA and the solution from commercial solver (C_{max}^{CPLEX}) was computed as shown in Eq. (1).

$$\%Gap = \frac{C_{\max}^{SA} - C_{\max}^{CPLEX}}{C_{\max}^{CPLEX}} \times 100\% \quad (1)$$

Tables 5 through 8 and Figs. 6 through 9 compare the results obtained from SA with CPLEX and with the best Cmax from DRs for various problem instances. DR does not perform well when it is compared with SA and CPLEX; moreover, CPLEX was better than SA on several instances with less than 100 operations (see Table 5 and Fig. 6). However, SA and CPLEX reported the optimum for some of the instances. The original instance code and the run code used in this research are shown along the *x*-axis in each figure. The Cmax is shown along the *y*-axis. In the tables, the original code [23], run code, makespan, and the percent gap are shown. The optimum solutions are shown with bold font.

SA outperformed CPLEX on a couple of instances when the total number of operations was between 100 and 300 (see Table 6 and Fig. 7). SA outperformed CPLEX on all instances with 300 to 500 operations (see Tables 7 and 8, and Figs. 8 and 9).

Figure 10 summarizes the average gap in comparing SA with CPLEX. CPLEX performs well when the total number of operations is less than 300 operations. SA outperforms CPLEX when there are more than 300 operations.

Table 9 and Fig. 11 compare the run time required by DR, SA, and CPLEX. The DR finds the solution in the shortest period of time. The average run time required by SA is far less compared to CPLEX, which makes it more attractive for practitioners to implement. The DR solution is inferior compared to SA on all the instances.

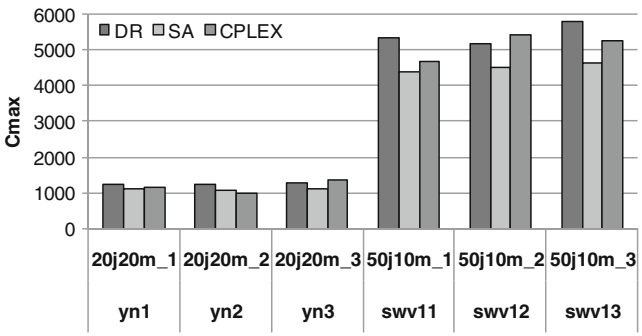

Fig. 9 Results for job instances with more than 300 operations

6 Conclusions

This study has extended the class of job shop scheduling problems typically researched (i.e., only discrete processing machines are considered) by considering both discrete machines and BPMs. The classical job shop problem is a well-known NP-hard problem. Consequently, the problem studied in this research is also NP-hard. A new network representation of the problem under study was proposed. However, the size of the network exploded in size with the increase in the number of machines and jobs, and the shifting bottleneck heuristic required prohibitively long run times to solve the problems. In an effort to develop a more practical solution approach, simulated annealing was considered. The effectiveness of the proposed approach was evaluated against four dispatching rules and CPLEX, a

Table 9 Results for average run time between DR, SA, and CPLEX

Total operations	Avg. run time (s)		
	DR	SA	CPLEX
Operations≤100	0.42	94.23	2,545.79
100<operations<300	0.53	1,264.69	5,877.70
Operations=300	0.77	4,462.96	12,979.14
Operations>300	1.17	18,196.67	25,557.37

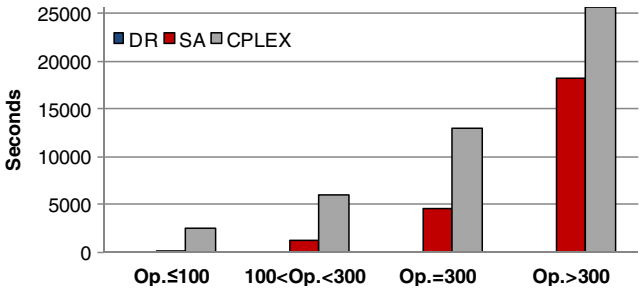

Fig. 11 Comparison of average run time between DR, SA, and CPLEX

commercial solver used to solve a mathematical formulation of the problem under study. The experimental study conducted clearly shows that the dispatching rules are not capable of reporting good-quality solutions, although they are quick to find a solution. Commercial solvers require prohibitively long run times to report feasible solutions for large-sized problem instances. SA outperformed CPLEX in computation time for all the problem instances and in Cmax values for 300 to 500 operation problem instances.

It is typical to observe a large number of operations even in a small job shop. As the experimental study suggests, our approach will help practitioners to schedule their resources more effectively than using dispatching rules (which is also the current practice at the facility where this study stemmed from). The program files written in Matlab are converted into executable files for the company to use. The data are fed in as a text file, and the output is written to a text file.

This research work can be extended to study other due date-related objectives in the job shop setting. In addition, other dispatching rules used in scheduling job shops can also be explored to see if it helps to improve the makespan.

References

1. Pinedo M (2002) Scheduling theory, algorithms, and systems, Secondth edn. Prentice-Hall, Upper Saddle River
2. Mason SJ, Fowler JW (2000) Maximizing delivery performance in semiconductor wafer fabrication facilities. Simul Conf Proc 2:1458–1463, o.2
3. Mason SJ, Fowler JW, Carlyle WM, Montgomery DC (2005) Heuristics for minimizing total weighted tardiness in complex job shops. Int J Prod Res 43(10):1943–1963
4. Gupta AK, Sivakumar AI (2006) Job shop scheduling techniques in semiconductor manufacturing. Int J Adv Manuf Technol 27(11–12):1163–1169
5. Ovacik IM, Uzsoy R (1997) Decomposition methods for complex factory scheduling problems. Kluwer, Boston
6. Fang L, Luh PB, Chen H (2000) Improving the Lagrangian relaxation approach for large job-shop scheduling. 39th IEEE Conference on Decision and Control; Dec 12–15 2000; Sydney, NSW: Institute of Electrical and Electronics Engineers Inc.
7. Mason SJ, Fowler JW, Carlyle WM (2002) A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. J Sched 5(3):247–262
8. Mönch L, Driessel R (2005) A distributed shifting bottleneck heuristic for complex job shops. Comput Ind Eng 49(3):363–380
9. Uzsoy R, Wang CS (1997) Decomposition procedures for global scheduling of complex job shops. Proceedings of the 1997 21st IEEE/CPMT International Electronics Manufacturing Technology (IEMT) Symposium; Oct 13–15 1997; Austin, TX, USA: IEEE, Piscataway, NJ, USA
10. Demirkol E, Mehta S, Uzsoy R (1997) A computational study of shifting bottleneck procedures for shop scheduling problems. J Heuristics 3(2):111–137
11. Uzsoy R, Wang CS (2000) Performance of decomposition procedures for job shop scheduling problems with bottleneck machines. Int J Prod Res 38(6):1271–1286
12. Sun T, Luh PB, Liu M (2006) Lagrangian relaxation for complex job shop scheduling. Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006; May 15–19 2006; Orlando, FL, USA: Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ 08855–1331, USA
13. Kaskavelis CA, Caramanis MC (1998) Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems. IIE Trans (Inst Ind Eng) 30(11):1085–1097
14. Pfund ME, Balasubramanian H, Fowler JW, Mason SJ, Rose O (2008) A multi-criteria approach for scheduling semiconductor wafer fabrication facilities. J SCHED 11(1):29–47
15. Damodaran P and Rojas M (2011). Mathematical formulation to minimize makespan in a job shop with a batch processing machine. Ninth LACCEI Latin American and Caribbean Conference (LACCEI'2011), Engineering for a Smart Planet. http://www.laccei.org/LACCEI2011-Medellin/…/PE166_Rojas.pdf. Accessed 25 Jan 2012
16. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
17. Cerny V (1985) A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J Optim Theory Appl 45:41–51
18. Yamada T, Nakano R (1996) Job shop scheduling by simulated annealing combined with deterministic local search, *meta-heuristics*: *theory and applications*. Kluwer, MA, pp 237–248
19. Fattahi P, Mehrabad MS, Jolai F (2007) Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J Intell Manuf 18(3):331–342
20. Chung SH, Tai YT, Pearn WL (2008) Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes. Int J Prod Res. doi:10.1080/00207540802010807
21. Yao AC (1980) New algorithms for bin packing. J ACM 27(2):207–227
22. Askin R, Standridge C (1993) Modeling and analysis of manufacturing systems. Wiley, New York, 451 p
23. Beasley JE (1990). Set of 82 JSP test instances. OR-library: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt. Accessed 30 June 2009
24. Panwalkar SS, Iskander W (1977) A survey of scheduling rules. Oper Res 25:45–61