



# Open

## Minimizing energy consumption and makespan in a two-machine flowshop scheduling problem

S Afshin Mansouri<sup>1\*</sup> and Emel Aktas<sup>2</sup>

<sup>1</sup>*Brunel Business School, Brunel University London, Uxbridge, UK; and* <sup>2</sup>*Cranfield School of Management, Cranfield University, Bedfordshire, UK*

Energy consumption has become a key concern for manufacturing sector because of negative environmental impact of operations. We develop constructive heuristics and multi-objective genetic algorithms (MOGA) for a two-machine sequence-dependent permutation flowshop problem to address the trade-off between energy consumption as a measure of sustainability and makespan as a measure of service level. We leverage the variable speed of operations to develop energy-efficient schedules that minimize total energy consumption and makespan. As minimization of energy consumption and minimization of makespan are conflicting objectives, the solutions to this problem constitute a Pareto frontier. We compare the performance of constructive heuristics and MOGAs with CPLEX and random search in a wide range of problem instances. The results show that MOGAs hybridized with constructive heuristics outperform regular MOGA and heuristics alone in terms of quality and cardinality of Pareto frontier. We provide production planners with new and scalable solution techniques that will enable them to make informed decisions considering energy consumption together with service objectives in shop floor scheduling.

*Journal of the Operational Research Society* (2016) 67(11), 1382–1394. doi: 10.1057/jors.2016.4;  
advance published online 6 April 2016

**Keywords:** scheduling; production; energy consumption; multi-objective; heuristics; genetic algorithms

The online version of this article is available Open Access

### 1. Introduction

A significant proportion of energy used in manufacturing is currently generated through fossil fuels (Rahimifard *et al.*, 2010). Therefore in the foreseeable future, energy efficiency will become the main focus in manufacturing because of both scarce resources and increasing greenhouse gases from production processes. In terms of energy-efficient manufacturing, minimization of energy use, recovery of parts, transformation of wastes into key resources are required to align the manufacturing processes with principles of sustainable production and resource efficiency.

The manufacturing sector uses massive amounts of energy and contributes to 36% of global CO<sub>2</sub> emissions (OECD-IEA, 2007). In the United Kingdom, industry electricity consumption accounts for 31% of the total. This is equivalent to 69 million metric tonnes of CO<sub>2</sub>, which approximates to annual greenhouse gas emissions from more than 14.3 million passenger vehicles (calculation obtained from EPA, 2013). This has obliged manufacturing companies to put more efforts into reducing their environmental impact and take proactive measures to consider likely energy shortages in their operations. One way to do this is by using energy-efficient operations (Duflou *et al.*, 2012) such as selectively shutting down machines

during idle time (Mouzon *et al.*, 2007; Mouzon and Yildirim, 2008) or operating them at speeds allowed by the set service level targets.

Manufacturing scheduling has traditionally been influenced by performance-oriented metrics such as makespan, float time, and tardiness. Minimizing carbon footprint on the shop floor involves multifaceted challenges that necessitate a multi-objective approach because of conflicting objectives of, for example, makespan and energy consumption. It entails complex decision making and trade-off analysis by the operations managers. As one of the first attempts in this field, Mansouri *et al.* (2016) addressed a bi-criteria two-machine flowshop scheduling problem to minimize total energy consumption and makespan. They showed the conflict between the two objectives and developed an  $O(n^3)$  heuristic to solve large size problems. Two-machine flowshop scheduling problems have attracted significant attention from practitioners and researchers. There are many real world problems that involve scheduling of two machines. These include applications for instance in printed circuit board (Sabouni and Logendran, 2013), shampoo production (Belaid *et al.*, 2012), and metalworking (Uruk *et al.*, 2013). In this paper, we extend the work of Mansouri *et al.* (2016) by developing a new  $O(n^2)$  heuristic and multi-objective genetic algorithms (MOGAs) for a sequence-dependent two-machine permutation flowshop scheduling problem to extend applicability of the concept of green scheduling in real life applications. In particular, we examine the effect of hybridizing the MOGA

\*Correspondence: S Afshin Mansouri, Brunel Business School, Brunel University London, Kingston Lane, Uxbridge UB8 3PH, UK.  
E-mail: Afshin.Mansouri@brunel.ac.uk

with the constructive heuristics to improve the efficiency and the effectiveness of the search. We validate the performance of the developed solution techniques through comprehensive experiments based on three performance metrics, namely, quality (distance with the lower bound to the problem), diversity (number of unique sequences in the solution set), and cardinality (size of the solution set) of the Pareto frontiers.

The remainder of the paper is organized as follows. Section 2 reviews the literature. Section 3 introduces the mathematical model. Solution techniques including the new constructive heuristic and MOGAs are described in Section 4. The experimental set-up and results are presented in Section 5. Finally, Section 6 discusses the results, concludes the paper, and identifies future research directions.

## 2. Literature review

Research on incorporating energy considerations into manufacturing scheduling is rather limited. Previous work focused on minimizing energy consumption and total completion time on a single machine with multi-objective mathematical programming models used for job scheduling where energy savings were achieved by turning machines off during idle time but not by considering energy used during machine operation (Mouzon *et al.*, 2007).

Processing time and energy consumption of computer numerically control machines can vary significantly by changing cutting speed, feed rate, depth of cut, and nose radius (Ahilan *et al.*, 2013). It is possible to explore opportunities for saving energy by relaxing the fixed processing times assumption. For example, Fang *et al.* (2011) developed a multi-objective mixed integer linear programming (MOMILP) model to minimize completion time and energy by varying operation speed on a single machine. In this work, decisions on operation speed affected peak load and energy consumption. Although they analysed a flowshop environment with two machines, they did not consider set-up times, which have a direct impact on the completion time.

Machining time dictates the energy demand and the specific energy consumption of a machine tool is affected by the processing speed (Diaz *et al.*, 2011). Not surprisingly, in the computing field, energy consumed increases with higher execution speeds of processors (Fang and Lin, 2013) and jobs executed at a higher machine speed for time savings incur a greater energy consumption.

It is possible to build mathematical models to predict power consumption based on machining parameters. One such work by Ahilan *et al.* (2013) reports the use of neural networks to examine the effect of turning parameters such as cutting speed, feed rate, depth of cut, and nose radius on power consumption and surface roughness. The authors develop a non-linear parametric equation that estimates power consumption based on various levels of machining parameters and report a positive relationship between power consumption and turning parameters. It is then possible to use this estimate of power

consumption in scheduling problems that consider power consumption explicitly, such as those studied by Mouzon *et al.* (2007), Fang *et al.* (2011), Liu *et al.* (2013), or this study.

Another work that should be noted is by He *et al.* (2005), who developed a bi-objective job-shop scheduling model to optimize both the energy consumption and the makespan, where energy consumption was calculated as a function of the unload power of the machine and the machining time. The authors used a heuristic algorithm based on tabu search to solve this problem.

On the other hand, energy consumption can be analysed separately during machine operation and idling. Liu *et al.* (2013) addressed this problem and developed a branch-and-bound algorithm based on the NEH heuristic (Nawaz *et al.*, 1983) to solve the permutation flowshop problem with idle energy minimization. Their objective was to minimize the total wasted energy consumption as the weighted sum of idle times on each machine.

Considering the extant work published in this field, scheduling with set-up times received relatively lower attention, probably because of the complexity of the problems. On scheduling with set-up times, Gharbi *et al.* (2013) developed lower bounds for the two-machine flowshop scheduling with sequence-independent set-up times based on waiting time-based relaxation, the single machine-based relaxation, and the Lagrangian relaxation, and recommended hybridizing the single machine-based and the Lagrangian relaxation-based lower bounds for sequence-dependent problems.

Some factors such as peak and off-peak times set by energy providers affect energy consumed on the shop floor and associated costs; yet they are outside the decision space of the manufacturer. Nevertheless, it is possible to incorporate such factors into the scheduling models. For example, Luo *et al.* (2013) studied machine electricity consumption costs in a hybrid metalworking flowshop and used constant power/speed ratios to optimize the electricity consumption during peak and off-peak hours. They recommended combining fast and slow operating machines to achieve higher energy efficiency.

Not only cost minimization goals but also environmental sustainability concerns call for minimizing energy consumption in manufacturing operations. There is usually a trade-off for the manufacturer between green and regular production technologies. Gong and Zhou (2013) analysed this trade-off from the perspective of emissions trading and observed that such a trade-off is governed by the relationship between the additional cost per energy consumption allowance saved and the trading prices.

In relation to the mathematical formulation of scheduling problems with set-up times, Allahverdi *et al.* (2008) have produced a detailed review and in terms of solution approaches Yenisey and Yagmahan (2014) report on the use of heuristic algorithms in permutation flowshop scheduling problems. Both works can serve as a good starting point for the reader who wish to deepen their knowledge in this field.

To summarize, machining parameters, specifics of operations, the nature of the problem at hand, and external variables have a role to play in minimizing energy consumption in manufacturing. With the advancement of manufacturing technologies, machines can now be operated at variable speeds accompanied with a corresponding energy consumption profile. However, the research on including energy consumption with variable speeds is yet to grow. Continuing on a previous work by Mansouri *et al* (2016) that analyzes the trade-off between minimizing makespan, a measure of service level and minimizing total energy consumption, an indicator of environmental sustainability, we are aiming to address this gap by furthering the modelling of energy consumption explicitly in scheduling and developing heuristic and metaheuristic algorithms to solve this problem.

### 3. Problem formulation

We address a two-machine permutation flowshop scheduling problem with sequence-dependent set-up times where machines have variable speed. Following from Ibrahimov *et al* (2014) we build a model with a high degree of fidelity, with reasonable assumptions and approximations. The general flowshop scheduling problem consists of  $n$  jobs that are to be processed in  $m$  machines with fixed, non-negative processing time for all jobs (Tiwari *et al*, 2015). Similar to the bi-criteria problem addressed by Lu and Logendran (2012), a set-up is required for processing each job on each machine, and its duration depends on both the current and the immediately preceding job. Set-up times are anticipatory, that is, a set-up can be started before the corresponding job becomes available on the machine. We adapt Graham's three-field notation ( $\alpha|\beta|\gamma$ ) (Graham *et al*, 1979) for scheduling problems (T'kindt and Billaut, 2006) where  $\alpha$  field describes the shop (machine) environment,  $\beta$  field describes the set-up information, other shop conditions, and details of the processing characteristics and  $\gamma$  field describes the objective to be minimized.

The two-machine flowshop scheduling problem to minimize total energy consumption ( $TEC$ ) and makespan (or  $C_{max}$ ) with sequence-dependent set-up times is denoted as  $F2|ST_{sd}|TEC, C_{max}$ . We refer to this problem as Problem  $P$ , which is a multi-objective optimization problem (MOP). Table 1 introduces the indexes, parameters, and variables used in the mathematical modelling of Problem  $P$ .

Problem  $P$  is NP-hard because the single objective problem  $F2|ST_{sd}|C_{max}$  is known to be NP-hard (Gupta and Darrow, 1986). Among the most common approaches to solve MOPs are sequential optimization, weighting method,  $\epsilon$ -constraint method, goal programming, goal attainment, and distance-based and direction-based methods (Collette and Siarry, 2004). Readers are referred to T'kindt and Billaut (2006) for a comprehensive survey on the theory and applications of multi-objective scheduling. In the following, we provide basic definitions of the MOP, which are needed to describe the solution techniques.

**Table 1** Indexes, parameters, and variables of the mathematical model

Indexes	
$i$	Index for machines
$j, k$	Index for jobs
$\ell$	Index for processing speeds
Parameters	
$m$	Number of machines; $i = 1, 2$
$n$	Number of jobs; $j, k = 1, \dots, n$
$p_{ij}$	Processing time (min) of job $j$ on machine $i$
$v_\ell$	Processing speed factor; $\ell = 1, 2, 3$ for fast, normal, and slow speeds, respectively
$s_{ijk}$	Sequence-dependent set-up time (min) for changing from job $j$ to job $k$ on machine $i$
$\lambda_\ell$	Conversion factor for processing speed $\ell$
$\varphi_i$	Conversion factor for idle time on machine $i$
$\pi_i$	Power of machine $i$ (kWh)
$M$	A very large number
Positive variables	
$c_{ij}$	Latest completion time (min) of job $j$ on machine $i$
$o_j$	Set-up offset for the first job on the second machine $j$
$\theta_i$	Idle time on machine $i$
$TEC$	Total energy consumption (kWh)
Binary variables	
$z_j$	$\in \{0, 1\}$ 1 if job $j$ is the first job
$x_{jk}$	$\in \{0, 1\}$ 1 if job $j$ is scheduled immediately before job $k$
$y_{j\ell}$	$\in \{0, 1\}$ 1 if job $j$ is processed at speed $\ell$ on machine $i$

A MOP seeks to determine a vector of decision variables within a feasible region to minimize a vector of objective functions that usually conflict with each other. Without the loss of generality, an MOP can take the following form:  $\min\{f_1(\tilde{x}), \dots, f_m(\tilde{x})\}$  subject to  $\tilde{x} \in \Theta$ , where  $\tilde{x}$  is the vector of decision variables and  $\Theta$  is the set of feasible solutions to  $m$  objectives. A decision vector  $\tilde{x}$  is said to dominate a decision vector  $\tilde{y}$  (also written as  $\tilde{x} \succ \tilde{y}$ ) if and only if  $f_i(\tilde{x}) \leq f_i(\tilde{y}) \forall i \in \{1, \dots, m\}$  and  $\exists i \in \{1, \dots, m\} \mid f_i(\tilde{x}) < f_i(\tilde{y})$  for a problem with all objectives to be minimized. All feasible solutions that are not dominated by any other feasible solution are called non-dominated or Pareto-optimal. These are solutions for which no objective can be improved without at least one other objective being deteriorated. Problem  $P$  can be formulated as a MOMILP problem as follows:

$$\min TEC \text{ and } C_{max} \tag{1}$$

$$\text{subject to : } M \times (1 - z_j) + o_j \geq s_{2jj} - c_{1j} \quad \forall j \tag{2}$$

$$c_{1j} \geq \frac{p_{1j}}{v_\ell} \times y_{1j\ell} + s_{1jj} \times z_j \quad \forall j, \ell \tag{3}$$

$$c_{2j} \geq c_{1j} + o_j + \frac{p_{2j}}{v_\ell} \times y_{2j\ell} \quad \forall j, \ell \tag{4}$$

$$M \times z_k + M \times (1 - x_{jk}) + c_{ik} \geq c_{ij} + \frac{p_{ik}}{v_\ell} \times y_{ik\ell} + s_{ijk} \times x_{jk} \quad \forall i, j, k, \ell \mid j \neq k \tag{5}$$

$$\sum_k x_{jk} = 1 \quad \forall j \mid j \neq k \tag{6}$$

$$\sum_j x_{jk} = 1 \quad \forall k \mid j \neq k \tag{7}$$

$$\sum_j z_j = 1 \tag{8}$$

$$\sum_l y_{ij\ell} = 1 \quad \forall i, j \tag{9}$$

$$\theta_i = C_{\max} - \sum_j \sum_\ell \frac{p_{ij}}{v_\ell} \times y_{ij\ell} \quad \forall i \tag{10}$$

$$TEC = \sum_i \sum_j \sum_\ell \lambda_\ell \times \frac{p_{ij}}{v_\ell} \times y_{ij\ell} \times \frac{\pi_i}{60} + \sum_i \varphi_i \times \theta_i \times \frac{\pi_i}{60} \tag{11}$$

$$C_{\max} \geq c_{2j} \quad \forall j \tag{12}$$

Constraint 2 ensures that the completion time of the first job on Machine 2 is delayed by taking into account the set-up offset, which allows for the set-up on the second machine to start before the first job is completed on the first machine because of the nature of anticipatory set-up assumption in the problem definition. Constraints 3 and 4 specify the earliest completion time of jobs on Machines 1 and 2, respectively. Constraint 5 ensures that set-up changeovers and completion times of the preceding jobs are taken into account in determining completion times of successive jobs. Constraints 6 and 7 specify the sequence of jobs as a tour in travelling salesman problem in which the last job is paired with the first job. Note that the decision variable  $z_j$  sets the first job in the tour and subsequently all the completion times are calculated accordingly. The subtour issue is handled by Constraint 5, which is only binding for consecutive jobs as defined by  $x_{jk}$  decision variables. For non-consecutive jobs, this constraint will be non-binding because of the presence of the big  $M$ . In this way, completion time of the last job (which is paired with the first job) will not be affected because of the big  $M$  in Constraint 5. Constraint 8 warrants that there is only one first job. Constraint 9 guarantees that exactly one speed factor is selected for each job. Machines' idle times are calculated by Constraint 10. Finally, Constraints 11 and 12 compute  $TEC$  and  $C_{\max}$ , respectively.

#### 4. Solution techniques

To address the computational complexity of the constructive heuristic developed by Mansouri *et al* (2016), which is an  $O(n^3)$  algorithm (called CH1 in this paper), we first introduce a novel constructive heuristic (CH2) with the reduced complexity  $O(n^2)$ . Subsequently, we develop regular and hybrid MOGAs

(denoted by R-MOGA and H-MOGA, respectively) to further improve the effectiveness and efficiency of search for the Pareto frontier of Problem  $P$ . Finally, we introduce an  $\epsilon$ -constraint method run on CPLEX and random search as two widely used benchmark approaches to assess performance of search heuristics.

##### 4.1. Constructive heuristic 2 (CH2)

The CH2 includes a scheduling module (SM). For this module, we extended the idea of the dominance rules proposed by Gupta and Darrow (1986) for single speed two-machine sequence-dependent flowshop scheduling to minimize  $C_{\max}$  to account for variable speed problem defined in Section 3. A local search is carried out at the end to improve quality of the solution. Details of the SM are presented in Algorithm 1. As detailed in Algorithm 1, the SM routine is implemented in three steps. The search parameters are first initialized in Step 0. Subsequently, the jobs are sequenced in Step 1 by using the speed vector  $\tilde{\Delta} = [\delta_{ij}]$ ,  $i = 1, 2; j = 1, \dots, n$ , where  $\delta_{ij}$  denotes the processing speed factor of job  $j$  on machine  $i$ ;  $\delta_{ij} \in \{v_1, v_2, v_3\}$  representing fast, normal, and slow speeds, respectively. As detailed in Algorithm 1,  $\sigma_1$  and  $\sigma_2$  are two partial sequences and  $\omega$  is the set of jobs, which are not included in  $\sigma_1$  and  $\sigma_2$ . These sets are initialized in Step 0 by setting  $\sigma_1 = \sigma_2 = \emptyset$  and  $\omega = \{1, 2, \dots, n\}$ . During the sequencing routine in Step 1, jobs are taken one by one from  $\omega$  and added to the end of  $\sigma_1$  or the beginning of  $\sigma_2$ . This process continues until  $n - 1$  jobs are sequenced in either  $\sigma_1$  or  $\sigma_2$  and there is only one job left in  $\omega$ . The final complete sequence  $\sigma$  is then formed by placing  $\sigma_1$  at the beginning,  $\omega$  in the middle, and  $\sigma_2$  at the end:  $\sigma = \sigma_1 \omega \sigma_2$ . In Step 2, the jobs are scheduled according to the sequence  $\sigma$  and speed vector  $\tilde{\Delta}$ . Accordingly, the start and finish times for all jobs on both machines are calculated. Finally, a local search is carried out to improve the quality of the solution. The local search begins by examining jobs one by one to see if removing them from their position and inserting them in subsequent positions could improve  $C_{\max}$ . In this way, the first job is examined for insertion in  $(n - 1)$  subsequent positions. It is then inserted in the position that yields the maximum reduction in  $C_{\max}$  or left in its current position if no such  $C_{\max}$ -improving move can be found. The second job in the sequence is then examined for insertion in the following  $(n - 2)$  positions and so on and so forth. On the basis of a given vector of processing speed factors, Algorithm 1 schedules the jobs and calculates  $C_{\max}$  and  $TEC$ .

By using this schedule as a starting point, the CH2 (Algorithm 2) seeks for energy-efficient schedules in an iterative loop. It starts with an initial sequence in which all jobs are run at *fast* speed (ie,  $v_1$ ). In an iterative procedure, the processing speeds of operations are decreased iteratively while keeping the same sequence. The central idea in this procedure is removing idle times on either machine by slowing down machining operations that affect the idle time the most. The idle times are



identified by comparing completion times of jobs on Machine 1 (denoted by  $c_{1[k]}$ ) and their ready times on Machine 2 (represented by  $r_{2[k]}$ ). The job with maximum  $|c_{1[k]} - r_{2[k]}|$  is considered the most promising job (denoted by  $\xi$ ) for energy saving by reducing machining speed of its respective operations on either machine. If the job is completed on Machine 1 after Machine 2 is ready to process it (ie,  $c_{1[\xi]} > r_{2[\xi]}$ ), then the speed of respective operation on Machine 2 (ie,  $\delta_{2[\xi]}$ ) is slowed down by one level if not currently run at slow speed. Alternatively, if the reverse situation holds (ie,  $r_{2[\xi]} > c_{1[\xi]}$ ), the speed of respective operation on Machine 1 (ie,  $\delta_{1[\xi]}$ ) is reduced by one level. In the event that there is no idle time on either machine (except for the first job), the operation with minimum processing time (ie,  $\min p_{i[k]}/\delta_{i[k]}$ ) is chosen for speed reduction. The resultant schedule is added to Pareto frontier ( $\Omega$ ) if it is not dominated by current members of the frontier. This process continues until all operations are run at slow speed  $v_3$ .

It is clear that the new algorithm has a complexity of  $O(n^2)$ . Compared with the constructive algorithm of Mansouri *et al* (2016) with complexity of  $O(n^3)$  (which is referred to as CH1 here), the new algorithm has less computational complexity. This time efficiency has been achieved at the expense of wider exploration of job sequences. Unlike CH1, CH2 does not change the job sequence after alteration of speed levels. As a result, all solutions of the resultant Pareto frontier from CH2 have the same job sequence.

---

**Algorithm 1:** The scheduling method (SM)

---

**input:** vector of jobs processing speed factors on the two machines  
**output:** schedule  $S$  with near optimal  $C_{\max}$  and its associated  $TEC$   
**begin**  
**Step 0 (initialization)**  
 let  $\sigma_1$  and  $\sigma_2$  be two partial sequences;  $d$  the last job in  $\sigma_1$  and  $e$  the first job in  $\sigma_2$ ;  
 let  $n_1$  and  $n_2$  denote the number of jobs in  $\sigma_1$  and  $\sigma_2$  respectively;  
 let  $\omega$  denote the set of jobs not included in  $\sigma_1$  and  $\sigma_2$ ;  
 let  $\sigma$  denote a complete sequence with  $n$  jobs;  
 let  $\delta_{ij}$  denote the processing speed factor of job  $j$  on machine  $i$ ;  $\delta_{ij} \in \{v_1, v_2, v_3\}$ ;  
 let let  $\tilde{\Delta}$  denote the vector of speed factors,  
 $\tilde{\Delta} = [\delta_{ij}], i = 1, 2; j = 1, \dots, n$ ;  
 let  $J_{[k]}$  denote the job in  $k^{th}$  position of the sequence;  
 set  $\sigma_1 = \sigma_2 = \emptyset, n_1 = n_2 = 0, \omega = \{1, 2, \dots, n\}$ ;  
**Step 1 (sequencing)**  
**while**  $(n_1 + n_2) < (n - 1)$  **do**  
     find job  $a$  such that  $A = p_{1a}/\delta_{1a} + s_{1da} - s_{2da}$   
          $= \min_{j \in \omega} [p_{1j}/\delta_{1j} + s_{1dj} - s_{2dj}]$ .  
     find job  $b$  such that  $B = p_{2b} = \min_{j \in \omega} [p_{2j}/\delta_{2j}]$ .

**Algorithm 1:** The scheduling method (SM)

---

**if**  $A < B$  **then**  
     let  $\sigma_1 = \sigma_1 a, n_1 = n_1 + 1$ ;  
**else if**  $A > B$  **then**  
     let  $\sigma_2 = b \sigma_2, n_2 = n_2 + 1$ ;  
**else**  
     **if**  $a \neq b$  **then**  
         **if**  $\min[(s_{1da} + p_{1a}/\delta_{1a}), (p_{2b}/\delta_{2b} + s_{2be})]$   
              $\leq \min[(s_{1db} + p_{1b}/\delta_{1b}), (p_{2a}/\delta_{2a} + s_{2ae})]$   
             set  $\sigma_1 = \sigma_1 a, n_1 = n_1 + 1$   
         **end**  
         **else**  
             set  $\sigma_2 = b \sigma_2, n_2 = n_2 + 1$ ;  
         **end**  
     **else**  
         **if**  $(s_{1da} + p_{1a}/\delta_{1a}) \leq (p_{2a}/\delta_{2a} + s_{2ae})$  **then**  
             set  $\sigma_1 = \sigma_1 a, n_1 = n_1 + 1$ ;  
         **end**  
         **else**  
             set  $\sigma_2 = b \sigma_2, n_2 = n_2 + 1$ ;  
         **end**  
     **end**  
     **end**  
     update  $\omega$ ;  
**end**  
 let  $\sigma = \sigma_1 \omega \sigma_2$ ;  
**Step 2 (scheduling and local search)**  
 create schedule  $S$  by sequencing the jobs according to  $\sigma$  and the speed vector  $\Delta$ ;  
 let  $S_1 = S; S_2 = S; \min C_{\max} = S \rightarrow C_{\max}$ ;  
**for**  $k_1 = 1$  **to**  $n - 1$  **do**  
     let  $k_2 = k_1$ ;  
     **while**  $k_2 < n - 1$  **do**  
         move the job and processing speeds from position  $k_2$  of  $S_1$  to position  $k_2 + 1$ ;  
         **if**  $S_1 \rightarrow C_{\max} < \min C_{\max}$  **then**  
             let  $\min C_{\max} = S_1 \rightarrow C_{\max}$ ;  
             let  $S_2 = S_1$ ;  
         **end**  
         let  $k_2 = k_2 + 1$ ;  
     **end**  
     let  $S = S_2$   
**end**  
**report** schedule  $S$  and its objective vector  $[S \rightarrow C_{\max}, S \rightarrow TEC]$ .  
**end**

---

**Algorithm 2:** Constructive heuristic 2 (CH2)

---

**input :** Set of jobs  
**output:** Approximation of Pareto frontier ( $\Omega$ )  
**begin**

**Algorithm 2:** Constructive heuristic 2 (CH2)

```

set iterator  $\rho = 0$ ;
let  $[k]$  denote the job in position  $k$  of the sequence;
let  $o_{i[k]}$  denote the operation in position  $k$  on machine  $i$ ;
let  $r_{i[k]} = c_{i[k-1]} + s_{i[k-1][k]}$  denote the ready time of  $k^{th}$ 
job on machine  $i$ ;
initialize the speed vector at fast speed  $\Delta_\rho: \{\delta_{i[k]} \leftarrow v_1;$ 
 $\forall i, k\}$ ;
develop schedule  $S_\rho$  by running the scheduling method
SM (Algorithm 1) using  $\Delta_\rho$ ;
if  $\nexists S_\mu \in \Omega \mid S_\mu \succ S_\rho$  then
    set  $\Omega = \Omega \cup S_\rho$ ;
end
let  $\Xi = \{o_{i[k]} \mid \delta_{i[k]} \neq v_3\}$ ;
while  $\Xi \neq \emptyset$  do
    set  $\Lambda = \{k \in S_\rho \mid c_{1[k]} \neq r_{2[k]}\}$ ;
    find  $\xi$  such that  $|c_{1[\xi]} - r_{2[\xi]}| = \max_{k \in \Lambda} |c_{1[k]} - r_{2[k]}|$ ;
    if  $\xi = 1 \wedge c_{1[\xi]} > r_{2[\xi]}$  then
        let  $\Lambda = \Lambda - \xi$ ;
        find  $\xi$  such that  $|c_{1[\xi]} - r_{2[\xi]}|$ 
         $= \max_{k \in \Lambda} |c_{1[k]} - r_{2[k]}|$ ;
    end
    if  $c_{1[\xi]} > r_{2[\xi]} \wedge \delta_{2[\xi-1]} \neq v_3$  then
        update  $\Delta_\rho : \{\delta_{2[\xi-1]} \leftarrow v_{(\ell+1)}\}$ ;
    else if  $r_{2[\xi]} > c_{1[\xi]} \wedge \delta_{1[\xi]} \neq v_3$  then
        update  $\Delta_\rho : \{\delta_{1[\xi]} \leftarrow v_{(\ell+1)}\}$ ;
    else
        find  $o_{i[\xi]} \in \Xi$  such that
         $p_{i[\xi]} / \delta_{i[\xi]} = \min_k p_{i[k]} / \delta_{i[k]}$ ;
        update  $\Delta_\rho : \{\delta_{i[\xi]} \leftarrow v_{(\ell+1)}\}$ ;
    end
    set  $\rho = \rho + 1$ ;
    recalculate  $c_{ij}$ 's,  $r_{ij}$ 's,  $C_{\max}$  and  $TEC$  of schedule  $S_\rho$ 
    using  $\Delta_\rho$ ;
    if  $\nexists S_\mu \in \Omega \mid S_\mu \succ S_\rho$  then
        set  $\Omega = \Omega \cup S_\rho$ ;
    end
    update  $\Xi$ ;
end
end
report  $\Omega$  as an approximation of the Pareto frontier;

```

4.2. Multi-objective genetic algorithms

Genetic algorithms (GAs) are adaptive search methods that have been shown to be robust for a variety of combinatorial optimization problems (Jog *et al.*, 1991). GAs have successfully been applied to solve a wide range of complex MOPs (Coello *et al.*, 2002). In a typical GA, a set of solutions (called population) are improved (or evolved) over a number of iterations (called generations) using a combination of operators

(named genetic operators) such as reproduction, crossover, and mutation (Goldberg, 2006). In this section we provide details of the MOGAs that we have developed to solve Problem  $P$ .

4.2.1. *Chromosome structure.* To represent solutions for Problem  $P$ , we use a two-dimensional chromosome structure, including three rows and  $n$  columns. Figure 1 illustrates the solution structure for given sequence  $J_1$  to  $J_n$ .

4.2.2. *Fitness assignment.* Fitness of solutions are calculated according to the concept of non-dominated sorting (Deb and Sinha, 2009). To assign appropriate fitness to the individuals in a population taking into account both objectives, we selected the non-dominated sorting method proposed by Srinivas and Deb (1994), which is one of the most commonly used methods for multi-objective optimization using GAs. In this procedure, the population is ranked based on individuals' level of non-domination. The non-dominated individuals of the population are first identified and assumed to constitute the first non-dominated frontier. These individuals are assigned a large dummy fitness value. To maintain diversity in the population, these dummy fitness values are then shared with solutions in their close neighbourhood, called their *niche*. In this way, the dummy fitness of an individual is divided by the number of solutions in its niche. The niche dimensions in a given population is calculated using the concept of *niche cubicle* proposed by Hyun *et al.* (1998). The niche size is calculated at every generation. A solution in a less dense cubicle will have a higher chance to survive in the next generation. After sharing, these non-dominated individuals are ignored and the same process is implemented to identify individuals for the second non-dominated frontier. These non-dominated solutions are then assigned a new dummy fitness value, which is smaller than the minimum shared dummy fitness of the previous frontier. The dummy fitness values are then shared and this process is repeated until the whole population is classified into several frontiers and individuals are assigned fitness values.

4.2.3. *Selection.* An elitist strategy is used to preserve non-dominated solutions found over generations in an archive called *Elite Set*. The *Elite Set* is updated at the end of each iteration by adding non-dominated solutions of the current generation and eliminating dominated solutions from the set. Selected solutions from each population are copied into the mating pool. To give fitter solutions more chance to contribute to the next generation, the expected number of each solution is

Job string:	$J_1$	$J_2$	...	$J_n$
Speed on Machine 1	$\ell$	$\ell$	...	$\ell$
Speed on Machine 2	$\ell$	$\ell$	...	$\ell$

**Figure 1** The chromosome structure for the sequence  $J_1, \dots, J_n$ .

set proportionate to their fitness. For a given solution  $x$  with fitness  $f_x$ , the expected number of copies is calculated as follows:  $E[x] = PopSize \times (f_x / \sum_x f_x)$ , where  $PopSize$  denotes *Population Size*. Integer parts of  $E[.]$  will determine the number of chromosomes to be copied into the mating pool. The remainder of mating pool is filled by randomly selected solutions from the *Elite Set*.

**4.2.4. Genetic operators.** Order crossover (Michalewicz, 1996) is used for recombination. Figure 2 represents an example of recombination using Order Crossover in a problem with seven jobs. The bold lines in Figure 2(a) designate the segment of the sequences that will remain intact and form part of the offspring. This segment is selected randomly for each crossover operation. A new parent is then created by moving all the genes appearing to the right of this segment (of the original parent) to the beginning of the sequence (Figure 2(b)). Subsequently, the characters that match the characters between the bold lines of the other original parent are removed (Figure 2(c)). Finally, the sequence between the bold lines of the original parent and this shortened list from the other parent is used to generate an offspring (Figure 2(d)).

To further diversify the population, a small proportion of the chromosomes will be mutated. Four strategies are used for mutating selected solutions. These include: *Inversion*, *Insertion*, *Swap*, and *Speed Alteration*. The four mutation strategies are demonstrated in Figure 3. Inversion simply reverses the order of genes in a randomly selected segment of the chromosome (Figure 3(a)). Through Insertion, a single gene is taken out from a random position along the chromosome and is inserted in another random position (Figure 3(b)). Swap operator exchanges the position of two randomly selected genes

(Figure 3(c)). Finally, Speed Alteration changes the speed factors of randomly chosen genes on either machines from one value to another (Figure 3(d)).

The chromosome produced by *mutation* will be compared with the original chromosome and will replace the original if it dominates it. However, a dominated offspring will still have a chance to replace its parent. Probability of such degrading moves will be high at the beginning and decreases as the search converges to final solution. The probability of accepting a dominated offspring, starting from 1.0, will be decreased exponentially over generations. The probability of accepting a dominated offspring resulted by *mutation* at a given time  $t$  is denoted by  $P_a(t)$  and is calculated as follows:

$$P_a(t) = \exp\left(\frac{-t}{t_{max} - t}\right), t = 0, \dots, t_{max} \quad (13)$$

where  $t_{max}$  represents the maximum execution (CPU) time.

**4.2.5. The search procedure and hybridization.** The overall structure of the MOGAs is represented in Algorithm 3. We develop three variants of MOGAs in this paper, including a regular (R-MOGA) and two hybrid MOGAs (or H-MOGAs). Performance of GAs can often be improved through hybridization with local search and heuristic approaches (Costa *et al*, 2012). The main difference between regular and hybrid MOGAs in this research are in the initiation of the *Elite Set*. The initial *Elite Set* of the R-MOGA is generated randomly whereas in the H-MOGAs, the *Elite Set* is initiated by the outputs of the two constructive heuristics, namely CH1 and CH2. The resultant hybrid methods are denoted by H-MOGA1 and H-MOGA2, respectively. This will give the H-MOGAs an initial advantage in terms of better starting points.

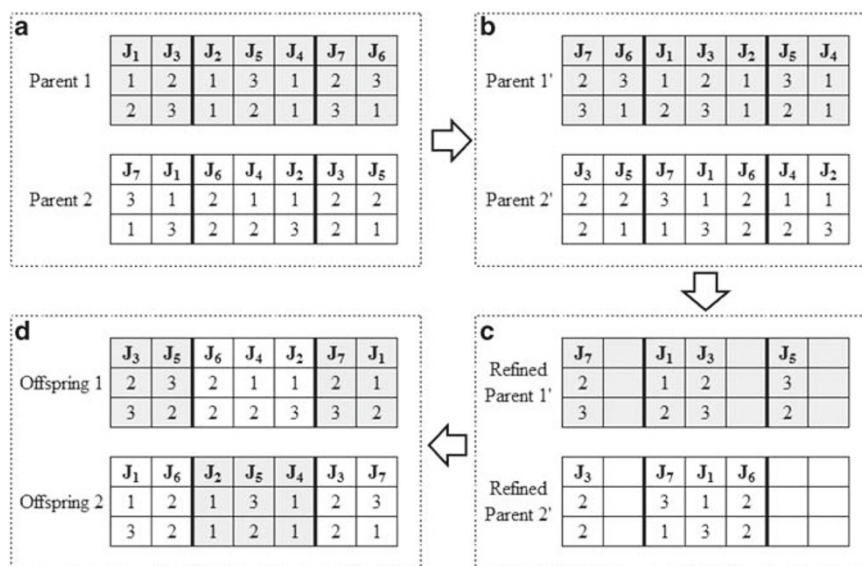
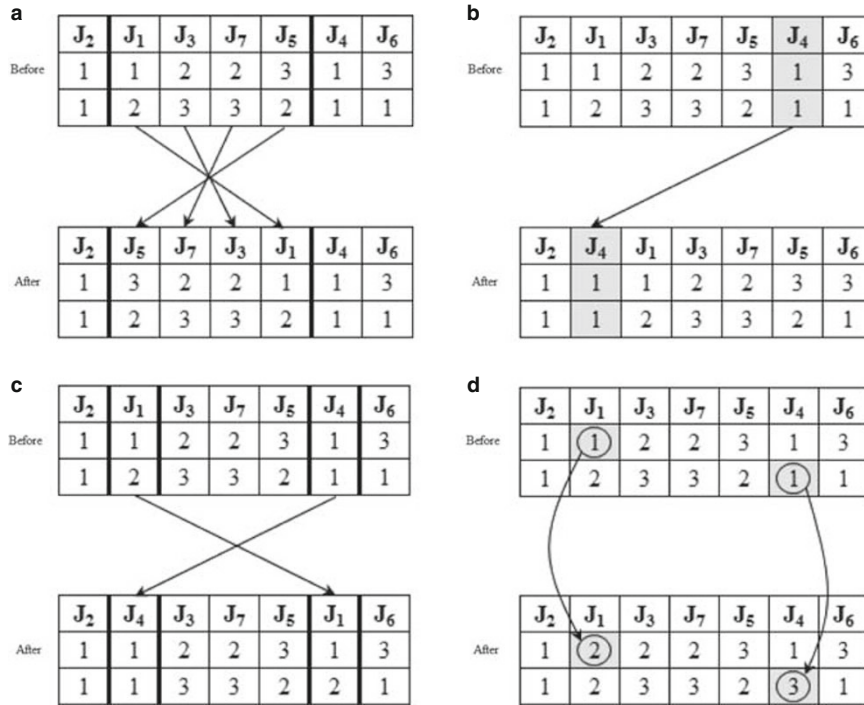


Figure 2 An example of recombination using Order Crossover in a problem with seven jobs.



**Figure 3** Four mutation strategies. (a) Inversion; (b) Insertion; (c) Swap; (d) Alteration.

Moreover, the solutions found by the constructive heuristics will have the chance to contribute to the generation of offspring.

**Algorithm 3:** Pseudocode of the MOGA approaches

**input :** Search parameters  
**output:** Approximation of the Pareto frontier  
 Let time counter  $t=0$ ;  
 Initialize *population*;  
 Initialize search parameters;  
 Initialize *{Elite Set}*  
**while**  $t < t_{\max}$  **do**  
     Perform nondominated sorting;  
     Select individuals for *mating pool*;  
     From *mating pool*, generate *new generation* using genetic operators;  
     Let *current generation* = *new generation*;  
     Identify  $F^1$  = nondominated frontier of *current generation*;  
     Let  $\{Elite Set\} = \{Elite Set\} \cup F^1$ ;  
     Refine  $\{Elite Set\}$ ;  
     Let  $t = t + 1$ ;  
**end**  
 Report  $\{Elite Set\}$ ;

4.3. Other benchmark methods

To have baselines for comparing performance of the above search methods, we use an  $\epsilon$ -constraint method implemented in CPLEX and a random search (RS). Using CPLEX provides the opportunity to examine the practicality of existing optimization tools in solving Problem  $P$ . Comparisons with RS on the other hand, will serve as a sanity check to ensure that the performance of the guided search methods is not just a result of computational power of the hardware used to implement them.

5. Experiments and results

5.1. Performance metrics

In this research, we have used three metrics to compare the performance of the solution techniques. These include distance with the lower bound ( $DLB$ ) as a measure of quality, diversity ( $DVR$ ) as a measure of the variety of the solution set, and cardinality ( $CRD$ ) as a measure of the size of the solution set of the final Pareto frontier found by each algorithm. The accuracy of the Pareto frontier  $\Omega$  is measured by its distance with the lower bound, denoted by  $DLB_{\Omega}$  and calculated as follows:

$$DLB_{\Omega} = \frac{\sum_{\tau \in \Omega} \min \left\{ \frac{(C_{\max}^{\tau} - C_{\max}^{LB})}{C_{\max}^{LB}}, \frac{(TEC^{\tau} - TEC^{LB})}{TEC^{LB}} \right\}}{|\Omega|} \tag{14}$$



where  $C_{\max}^{LB}$  and  $TEC^{LB}$  are lower bounds for  $C_{\max}$  and  $TEC$ , respectively, as defined in Mansouri *et al* (2016). In short,  $C_{\max}^{LB}$  and  $TEC^{LB}$  are calculated by solving a sequence-independent variant of Problem  $P$  (defined in Section 3), in which the minimum set-up changeovers from all other jobs are considered as the required set-up for any given job irrespective of their position in the sequence. The  $C_{\max}$  of the sequence-independent problem can be found in polynomial time using the algorithm of Yoshida and Hitomi (1979). Mansouri *et al* (2016) proved that  $C_{\max}$  of the sequence-independent problem is a lower bound for  $C_{\max}$  of Problem  $P$  when all operations are run at slow speed. Moreover, they showed that  $TEC$  of the sequence-independent problem is lower bound for  $TEC$  of Problem  $P$  when operations are executed at slow speed and under some non-restrictive conditions. The diversity of the Pareto frontier  $\Omega$  is denoted by  $DVR_{\Omega}$  and it represents the number of unique sequences in the final frontier. It should be mentioned that the objective values of a solution are primarily defined by two factors: the sequence of jobs and speed of operations. As such, a single sequence of jobs may lead to different  $C_{\max}$  and  $TEC$  values depending on speed factors of operations on the two machines. Considering  $n$  jobs and three speed levels for every operations, there are  $(n^3)^2 = n^6$  possible combinations of speed factors for any sequence of jobs. The  $DVR$  metric captures the number of unique sequences regardless of speed in the Pareto frontier, which serve as an indicator of flexibility in production planning. The more diverse is the Pareto frontier, the more flexibility production planners will have in job sequencing. Finally, the total number of solutions in the frontier is the cardinality of the Pareto frontier  $\Omega$  and denoted by  $CRD_{\Omega}$ , which reflects the total number of solutions in the Pareto frontier regardless of the uniqueness of sequences.

## 5.2. Experimental design

We used a full factorial design of experiments for a  $F2|ST_{sd}|C_{\max},TEC$  problem with 480 problem instances (30 for each combination of factors) based on the number of jobs ( $n$ ) set-up to processing time ratio ( $s_{ijk}$ ) generated using the experimental setting given in Table 2. Then we solved these problem instances using seven algorithms: CPLEX, CH1, CH2, H-MOGA1, H-MOGA2, R-MOGA, and RS.

In Table 2, we based the number of jobs in our experiments on previous flowshop scheduling research by Naderi *et al* (2009). Following from the classical problem introduced by Taillard (1990) and revisited by Ruiz *et al* (2005) we used uniformly distributed processing times in the experiments. In order to gain insights about the impact of set-up times, we followed the set-up to processing time ratio investigated by Ruiz and Stützle (2008). We took the idle time energy consumption parameter from Mouzon *et al* (2007). The work of Ahilan *et al* (2013) was instrumental to estimating processing speed and energy conversion rate. We obtained energy conversion rate distribution from Ahilan *et al* (2013) and drew this parameter from the given distributions for each problem set. In accordance with the work of Ahilan *et al* (2013) and Mouzon *et al* (2007), each problem instance in the data set satisfied the condition:  $\min\{(\lambda_1 - \lambda_2)$  and  $(\lambda_2 - \lambda_3)\} \geq \max(\varphi_1, \varphi_2)$ .

**5.2.1. Setting up the search methods.** Parameters of the MOGAs were tuned empirically on a number of test problems in comparison with true Pareto frontiers found by CPLEX. The following are the set of parameters used for the MOGAs: *Population Size* =  $n$ ; *Crossover Rate* = 0.70; *Inversion Rate* = 0.10; *Swap Rate* = 0.10; *Insertion Rate* = 0.08; *Speed Alteration Rate* = 0.10; and  $t_{\max} = n$  seconds.

For a fair comparison, we ran CPLEX under  $\epsilon$ -constraint approach for the same time ( $n$  seconds) as other search heuristics. To allow for the formation of frontier and to avoid spending too much time at any  $\epsilon$  level, we set a limit for 10% of the total time for each  $\epsilon$  level before proceeding with the reduced  $\epsilon$  value. Incidentally, in deciding on the time spent at each stage, there is a trade-off among the three performance metrics, that is, accuracy, diversity, and cardinality. More time at any given  $\epsilon$  level would allow CPLEX to improve accuracy but at the expense of less iterations and hence lower cardinality and diversity. We examined a number of problem instances and observed that 10% provides a fair opportunity for exploration and exploitation of the search space at the same time. The best solution found at each stage is archived and ultimately refined by removing dominated solutions to obtain the set  $\Omega$ . All algorithms were coded in C++ and run on an Intel Xeon CPU 3.50GHz with 32.0GB RAM under Windows 7 Enterprise.

**Table 2** Summary of experimental design

Input	Levels	Based on
Number of jobs ( $n$ )	20, 50, 80, 120	Naderi <i>et al</i> (2009)
Processing timedistribution ( $p_{ij}$ )	Unif (1,99)	Ruiz <i>et al</i> (2005); Taillard (1990)
Sequence dependentset-up timedistribution ( $s_{ijk}$ )	Unif (1,25), Unif (1,50), Unif (1,99), Unif (1,125)	Ruiz and Stützle (2008)
Powerdistribution ( $\pi_i$ )	Unif (60,80)	Heidenhain (2011); Brooks (2012)
Idle time energyconsumption ( $\varphi_i$ )	0.05	Mouzon <i>et al</i> (2007)
Processing speed ( $v_{\ell}$ )	1.2, 1, 0.8	Ahilan <i>et al</i> (2013)
Conversion rate ( $\lambda_{\ell}$ )	LogN (6.395, 0.220), LogN (6.225, 0.229), LogN (5.804, 0.303)	Ahilan <i>et al</i> (2013)

Moreover, we used CPLEX 12.5 in Concert Technology to code the MILP model in C++. Statistical analyses were done on IBM SPSS Version 20.

5.3. Results

The *LB* we defined is very conservative. For each job, it takes the shortest set-up time from job *k* to job *j*. Then, it uses the Yoshida and Hitomi’s (1979) algorithm to find  $C_{max}$ . When the ratio of set-up to processing time is low, the *LB* is closer to the optimal solution. So for problem sets with larger set-up to processing time ratio the *LB* becomes much farther away from the optimal solution. Part of the gap is because of the looseness of the *LB* and part of it is because of the factors that affect problem complexity. As our goal was to improve the constructive heuristic developed in Mansouri *et al* (2016), we initially checked how well CH2 performed against CH1 in terms of CPU time and the three performance metrics we defined in Section 5.1. As it can be seen from Table 3, CH2 outperforms CH1 in CPU time required for the algorithm, the distance with the lower bound and the number of non-dominated solutions in the Pareto frontier, but not diversity. So when the production planner is looking for quick solutions to his scheduling problem, he may prefer CH2 over CH1 unless the diversity of solutions does not have priority over other performance metrics.

Table 4 shows the performance of algorithms with respect to *DLB*. CPLEX is outperformed by even the random search, where the difference between CPLEX and algorithms that employ a GA (H-MOGA1, H-MOGA2, R-MOGA) are about 10%. There is no difference between CH1 and CH2 in terms of the *DLB* and both are outperformed by all algorithms that employ a GA. Both CH1 and CH2 perform better than CPLEX and RS. There is no statistically significant difference between the performance of H-MOGA1 and H-MOGA2 in terms of the *DLB*. As expected, H-MOGA1 and H-MOGA2 perform better than R-MOGA and RS.

In Table 5, it is clear that CH1 outperforms all the rest of the algorithms with respect to diversity. Following CH1, hybrid MOGAs are second best performers. Between the two hybrid MOGAs, H-MOGA1 performs better than H-MOGA2. Interestingly, RS performs better than R-MOGA. One possible explanation for this result is the fact that quality of the solution is not a criteria for assessing diversity performance and random search has an inherent

ability to explore a wider search space, and therefore finds more unique sequences than R-MOGA. Finally, the worst performers in terms of diversity are CPLEX and CH2. Although CPLEX found more solutions than CH2, the difference in the number of unique solutions is not statistically significant.

Table 6 shows the mean differences between algorithms with respect to the number of solutions they found in the Pareto frontier. CH2 produced the largest number of solutions on the Pareto frontier. As can be seen in Table 6 CPLEX is outperformed by all algorithms where a constructive heuristic is used. There is no statistically significant difference between CPLEX and the MOGA starting with random search (R-MOGA) and the random search (RS). CH2 and H-MOGA2 outperform CH1 in terms of the number of solutions on the Pareto frontier; however, as it is highlighted in the previous section on diversity, all these solutions represent only a single unique sequence. CH1 performs better than its counterpart with GA (H-MOGA1) and random solutions (R-MOGA and RS).

5.3.1. An illustrative example. We show here, on a single problem instance with 20 jobs and set-up to processing time ratio = 25%, how the Pareto frontier and the performance of the algorithms compare with each other. In Figure 4(a) we display the Pareto frontier of each algorithm and the lower bounds for  $C_{max}$  and *TEC*. A few interesting observations can be made: Random search (RS) has a frontier that is inferior compared with all other frontiers except that of CPLEX.

Table 4 Comparison of algorithms with respect to *DLB*

(I) Algorithm	(J) Algorithm	Mean difference (I-J)	Standard error	Significance
CPLEX	CH1	0.076	0.007	0.000
	CH2	0.079	0.007	0.000
	H-MOGA1	0.131	0.007	0.000
	H-MOGA2	0.123	0.007	0.000
	R-MOGA	0.103	0.006	0.000
CH1	RS	0.049	0.006	0.000
	CH2	0.003	0.007	1.000
	H-MOGA1	0.055	0.008	0.000
	H-MOGA2	0.047	0.007	0.000
	R-MOGA	0.026	0.007	0.004
CH2	RS	-0.027	0.007	0.002
	H-MOGA1	0.052	0.007	0.000
	H-MOGA2	0.044	0.006	0.000
	R-MOGA	0.024	0.006	0.003
	RS	-0.030	0.006	0.000
H-MOGA1	H-MOGA2	-0.008	0.007	0.924
	R-MOGA	-0.029	0.007	0.001
	RS	-0.082	0.007	0.000
H-MOGA2	R-MOGA	-0.021	0.006	0.014
	RS	-0.074	0.006	0.000
R-MOGA	RS	-0.054	0.006	0.000

Table 3 Paired comparison of the performance of CH1 and CH2

	CPU time		DLB		DVR		CRD	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
CH1	161.382	191.930	0.159	0.089	17.150	12.639	42.035	23.661
CH2	58.482	76.571	0.157	0.087	1.000	0.000	98.002	68.775
<i>p</i>	0.000		0.014		0.000		0.000	

**Table 5** Comparison of algorithms with respect to *DVR*

(I) Algorithm	(J) Algorithm	Mean difference (I-J)	Standard error	Significance
CPLEX	CH1	-14.910	0.473	0.000
	CH2	1.240	0.422	0.052
	H-MOGA1	-9.890	0.466	0.000
	H-MOGA2	-5.360	0.415	0.000
	R-MOGA	-1.500	0.402	0.004
	RS	-2.450	0.402	0.000
CH1	CH2	16.150	0.467	0.000
	H-MOGA1	5.020	0.507	0.000
	H-MOGA2	9.550	0.461	0.000
	R-MOGA	13.410	0.449	0.000
	RS	12.460	0.449	0.000
CH2	H-MOGA1	-11.130	0.460	0.000
	H-MOGA2	-6.600	0.408	0.000
	R-MOGA	-2.740	0.394	0.000
	RS	-3.690	0.394	0.000
H-MOGA1	H-MOGA2	4.530	0.454	0.000
	R-MOGA	8.390	0.441	0.000
	RS	7.430	0.441	0.000
H-MOGA2	R-MOGA	3.860	0.387	0.000
	RS	2.900	0.387	0.000
R-MOGA	RS	-0.950	0.373	0.139

**Table 6** Comparison of algorithms with respect to *CRD*

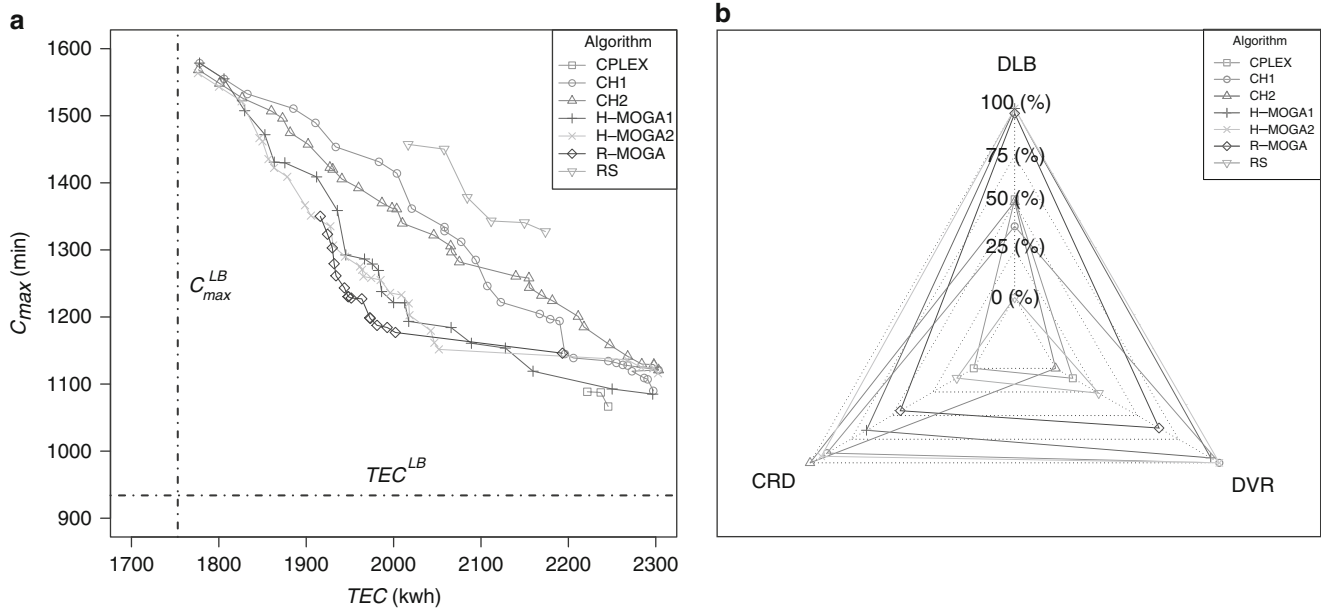
(I) Algorithm	(J) Algorithm	Mean difference (I-J)	Standard error	Significance
CPLEX	CH1	-27.600	1.839	0.000
	CH2	-68.980	1.640	0.000
	H-MOGA1	-14.640	1.812	0.000
	H-MOGA2	-33.130	1.614	0.000
	R-MOGA	-2.930	1.562	0.495
	RS	-2.140	1.562	0.819
CH1	CH2	-41.380	1.815	0.000
	H-MOGA1	12.950	1.972	0.000
	H-MOGA2	-5.530	1.791	0.033
	R-MOGA	24.660	1.745	0.000
	RS	25.460	1.745	0.000
CH2	H-MOGA1	54.330	1.788	0.000
	H-MOGA2	35.850	1.586	0.000
	R-MOGA	66.040	1.533	0.000
	RS	66.840	1.533	0.000
H-MOGA1	H-MOGA2	-18.490	1.764	0.000
	R-MOGA	11.710	1.716	0.000
	RS	12.510	1.716	0.000
H-MOGA2	R-MOGA	30.190	1.505	0.000
	RS	30.990	1.505	0.000
R-MOGA	RS	0.800	1.450	0.998

There are solutions found by RS with TEC lower than that of CPLEX. Another observation is that the frontier found CH1 for this problem instance is inferior to the frontier by H-MOGA1. A similar comment can be made for CH2 and H-MOGA2 pair. This in fact shows the level of improvement achieved by hybrid MOGAs since H-MOGA1 and H-MOGA2 start from the solutions found by CH1 and CH2, respectively. Hybrid MOGAs cannot be considered superior to R-MOGA; however, the frontier found by R-MOGA is much more compact than the frontiers found by hybrid MOGAs. R-MOGA has performed better than other search techniques in the middle section of the frontier but performed worse than some of the other methods in other areas. Figure 4(b) shows normalized performance matrix of these algorithms in a radar chart. We normalized the performance metrics *DLB*, *CRD*, and *DVR* given this problem so that the best performer has a score 100% and the worst performer has a score 0%. Considering the three performance measures, we cannot say there is an algorithm that is absolutely inferior. It is clear that none of the algorithms outperform the others in all three dimensions. H-MOGA1 and H-MOGA2 have the best performance in terms of *DLB*. CH2 has the best performance in terms of *CRD*. When it comes to *DVR*, CH1 and H-MOGA2 have the best performance. On the other hand, RS is inferior to CPLEX when it comes to *DLB*; however, it has more solutions than CPLEX and its diversity considering the unique sequences of solutions is higher than that of CPLEX. Apart from CH2 and CPLEX, the RS is inferior to the rest of the algorithms when considering all three performance measures.

## 6. Discussion and conclusion

Manufacturers are under pressure to incorporate greener practices in their operations. In this paper we addressed a complex two-machine sequence-dependent permutation flowshop scheduling problem to minimize energy consumption and makespan. Our research contributes to the literature on green scheduling by developing a new heuristic and GAs for finding the Pareto optimal frontiers. We compared performance of the heuristics and GAs against each other and also with truncated CPLEX (ie, time constrained) and random search. More specifically, we developed a new  $O(n^2)$  heuristic (named CH2) and compared its performance against an existing  $O(n^3)$  heuristic (called CH1). Our experiments show that CH2 takes much less time than CH1 to run, which depends on the problem size. On average, CH2 takes one-third of the time it takes CH1 to solve a problem. In terms of the distance of the resultant Pareto frontiers with the lower bounds, although statistically significant, the difference between the two heuristics is marginal. With regard to diversity, CH1 is able to produce much more diverse set of sequences, while CH2 produces only one sequence. This is indeed the main reason for faster implementation of CH2. While CH2 produces more solutions on the Pareto frontier, CH1 produces many more unique sequences, which provides higher flexibility to production planners.

Our results show that hybrid GAs perform better than regular GAs. In terms of diversity, CH1 showed the best performance in producing unique sequences, followed by H-MOGA1. In this respect, the poorest performance is shown by CH2.



**Figure 4** Comparison of algorithms on an illustrative problem instance. (a) Pareto frontiers; (b) Performance metrics.

This, however, is mainly because of the way that CH2 was designed based on time efficiency rather than solution quality or diversity. In terms of cardinality, CH2 produces the largest number of solutions. It should be noted that these solutions are produced for one sequence with different speed vectors.

On this basis of the results, it could be concluded that if time is not an important criterion, CH1 is the preferred solution method for small- and medium-sized problems. For large-sized problems, hybrid GAs are recommended. However, in situations where diversity of Pareto frontier is more important, H-MOGA1 is preferred. If cardinality is more important regardless of the number of unique sequences, then CH2 is preferable. Finally, if the distance with the lower bound and diversity are more important, then H-MOGA2 is preferable, followed by H-MOGA1 and CH1.

The current research could be extended in a number of directions. New mathematical models for green scheduling of manufacturing operation need to be developed. This includes new models for general  $m$ -machine flowshop, job-shop, and open-shop problems including other energy- and power-related objectives alongside performance-oriented metrics. As an example, minimizing maximum power and makespan in general job-shop environment has interesting applications for manufacturing companies to reduce their need to peak power while maintaining their service level. Majority of research in this field have been conducted at machine level, focusing on optimizing energy consumption considering various machining parameters. Future research could look into integrating this line of research with factory-level optimization of energy consumption. By optimizing individual components of a system, one does not necessarily optimize the entire system.

Such an integration would allow for wider energy savings at system level.

## References

- Ahilan C, Kumanan S, Sivakumaran N and Dhas JER (2013). Modeling and prediction of machining quality in CNC turning process using intelligent hybrid decision making tools. *Applied Soft Computing* **13**(3): 1543–1551.
- Allahverdi A, Ng C, Cheng T and Kovalyov MY (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* **187**(3): 985–1032.
- Belaid R, Tkindt V and Esswein C (2012). Scheduling batches in flowshop with limited buffers in the shampoo industry. *European Journal of Operational Research* **223**(2): 560–572.
- Brooks R (2012). High-performance cutting of hard aero-alloys, <http://americanmachinist.com/machining-cutting/high-performance-cutting-hard-aero-alloys>, accessed 20 November 2014.
- Coello CAC, Van Veldhuizen DA and Lamont GB (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Vol. 242. Kluwer Academic: New York.
- Collette Y and Siarry P (2004). *Multiobjective Optimization: Principles and Case Studies*. Springer-Verlag: Berlin Heidelberg.
- Costa L, Espirito Santo IA and Fernandes EM (2012). A hybrid genetic pattern search augmented Lagrangian method for constrained global optimization. *Applied Mathematics and Computation* **218**(18): 9415–9426.
- Deb K and Sinha A (2009). Solving bilevel multi-objective optimization problems using evolutionary algorithms. In: Ehrgott M *et al* (eds). *Evolutionary Multi-Criterion Optimization*. Springer-Verlag: Berlin Heidelberg, pp 110–124.
- Diaz N, Redelsheimer E and Dornfeld D (2011). Energy consumption characterization and reduction strategies for milling machine tool use. In: Hesselbach J and Herrmann C (eds). *Glocalised Solutions for Sustainability in Manufacturing*. Springer-Verlag: Berlin Heidelberg, pp 263–267.



- Dufflou JR *et al* (2012). Towards energy and resource efficient manufacturing: A processes and systems approach. *CIRP Annals-Manufacturing Technology* **61**(2): 587–609.
- EPA (2013). Greenhouse gas equivalencies calculator, <http://www.epa.gov/cleanenergy/energy-resources/calculator.html>, accessed 8 November 2013.
- Fang K, Uhan N, Zhao F and Sutherland JW (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems* **30**(4): 234–240.
- Fang K-T and Lin BM (2013). Parallel-machine scheduling to minimize tardiness penalty and power cost. *Computers & Industrial Engineering* **64**(1): 224–234.
- Gharbi A, Ladhari T, Msakni MK and Serairi M (2013). The two-machine flowshop scheduling problem with sequence-independent setup times: New lower bounding strategies. *European Journal of Operational Research* **231**(1): 69–78.
- Goldberg DE (2006). *Genetic Algorithms*. Pearson Education: India.
- Gong X and Zhou SX (2013). Optimal production planning with emissions trading. *Operations Research* **61**(4): 908–924.
- Graham RL, Lawler EL, Lenstra JK and Kan A (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics* **5**: 287–326.
- Gupta JN and Darrow WP (1986). The two-machine sequence dependent flowshop scheduling problem. *European Journal of Operational Research* **24**(3): 439–446.
- He Y, Liu F, Cao H-j and Li C-b (2005). A bi-objective model for job-shop scheduling problem to minimize both energy consumption and makespan. *Journal of Central South University of Technology* **12**(2): 167–171.
- Heidenhain (2011). Aspects of energy efficiency in machine tools, [http://www.heidenhain.us/enews/stories\\_1011/MTmain.php](http://www.heidenhain.us/enews/stories_1011/MTmain.php), accessed 20 November 2014.
- Hyun CJ, Kim Y and Kim YK (1998). A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. *Computers & Operations Research* **25**(7): 675–690.
- Ibrahimov M, Mohais A, Schellenberg S and Michalewicz Z (2014). Scheduling in iron ore open-pit mining. *The International Journal of Advanced Manufacturing Technology* **72**(5–8): 1021–1037.
- Jog P, Suh JY and Van Gucht D (1991). Parallel genetic algorithms applied to the traveling salesman problem. *SIAM Journal on Optimization* **1**(4): 515–529.
- Liu G-S, Zhang B-X, Yang H-D, Chen X and Huang GQ (2013). A branch-and-bound algorithm for minimizing the energy consumption in the PFS problem. *Mathematical Problems in Engineering* **2013**: 1–6.
- Lu D and Logendran R (2012). Bi-criteria group scheduling with sequence-dependent setup time in a flow shop. *Journal of the Operational Research Society* **64**(4): 530–546.
- Luo H, Du B, Huang GQ, Chen H and Li X (2013). Hybrid flow shop scheduling considering machine electricity consumption cost. *International Journal of Production Economics* **146**(2): 423–439.
- Mansouri SA, Aktas E and Besikci U (2016). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. *European Journal of Operational Research* **248**(3): 772–788.
- Michalewicz Z (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag: Berlin Heidelberg.
- Mouzon G and Yildirim MB (2008). A framework to minimise total energy consumption and total tardiness on a single machine. *International Journal of Sustainable Engineering* **1**(2): 105–116.
- Mouzon G, Yildirim MB and Twomey J (2007). Operational methods for minimization of energy consumption of manufacturing equipment. *International Journal of Production Research* **45**(18–19): 4247–4271.
- Naderi B, Zandieh M and Roshanaei V (2009). Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness. *The International Journal of Advanced Manufacturing Technology* **41**(11–12): 1186–1198.
- Nawaz M, Ensore Jr EE and Ham I (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* **11**(1): 91–95.
- OECD-IEA (2007). Tracking industrial energy efficiency and CO<sub>2</sub> emissions. Technical Report International Energy Agency.
- Rahimifard S, Seow Y and Childs T (2010). Minimising embodied product energy to support energy efficient manufacturing. *CIRP Annals—Manufacturing Technology* **59**(1): 25–28.
- Ruiz R, Maroto C and Alcaraz J (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research* **165**(1): 34–54.
- Ruiz R and Stützle T (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research* **187**(3): 1143–1159.
- Sabouni MY and Logendran R (2013). Carryover sequence-dependent group scheduling with the integration of internal and external setup times. *European Journal of Operational Research* **224**(1): 8–22.
- Srinivas N and Deb K (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* **2**(3): 221–248.
- Taillard E (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research* **47**(1): 65–74.
- Tiwari A, Chang P-C, Tiwari M and Kollanoor NJ (2015). A Pareto block-based estimation and distribution algorithm for multi-objective permutation flow shop scheduling problem. *International Journal of Production Research* **53**(3): 793–834.
- T'kindt V and Billaut J-C (2006). *Multicriteria Scheduling. Theory, Models and Algorithms*. Springer: Berlin.
- Uruk Z, Gultekin H and Akturk MS (2013). Two-machine flowshop scheduling with flexible operations and controllable processing times. *Computers & Operations Research* **40**(2): 639–653.
- Yenisey MM and Yagmahan B (2014). Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega* **45**: 119–135.
- Yoshida T and Hitomi K (1979). Optimal two-stage production scheduling with setup times separated. *AIIE Transactions* **11**(3): 261–263.

Received 5 January 2015;

accepted 12 January 2016 after two revisions



This work is licensed under a Creative Commons Attribution 3.0 Unported License. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in the credit line; if the material is not included under the Creative Commons license, users will need to obtain permission from the license holder to reproduce the material. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>