Journal of Industrial Engineering International
a SpringerOpen Journal

ORIGINAL RESEARCH Open Access

An electromagnetism-like metaheuristic for open-shop problems with no buffer

Bahman Naderi[1*], Esmaeil Najafi[2] and Mehdi Yazdani[3]

Abstract

This paper considers open-shop scheduling with no intermediate buffer to minimize total tardiness. This problem occurs in many production settings, in the plastic molding, chemical, and food processing industries. The paper mathematically formulates the problem by a mixed integer linear program. The problem can be optimally solved by the model. The paper also develops a novel metaheuristic based on an electromagnetism algorithm to solve the large-sized problems. The paper conducts two computational experiments. The first includes small-sized instances by which the mathematical model and general performance of the proposed metaheuristic are evaluated. The second evaluates the metaheuristic for its performance to solve some large-sized instances. The results show that the model and algorithm are effective to deal with the problem.

Keywords: Scheduling, Open shop with no buffer, Mixed integer linear programming, Electromagnetism algorithm

Introduction

Open-shop scheduling (OSS) can be defined by a set of n jobs that need to be processed by a set of m machines. Contrary to other shop scheduling, there are no predetermined processing routes for the jobs, which means that jobs could visit machines by any order. In other words, in open shops, there are two decisions to make: the determination of the processing routes of the jobs as well as job sequence on each machine (Pinedo 2008). The following assumptions are usually characterized to open shops. All the jobs are independent and available for their process at time 0. All machines are continuously available. Each machine can at most process one job at a time. Each job can be processed by at most one machine at a time. The process of a job on a machine cannot be interrupted. There are infinite buffers between all machines. There is no transportation time between machines. The objective function is the minimization of the total tardiness. The tardiness of each job is the amount of time that job is completed after its due date.

Bräsel et al. (2008) investigated OSS under minimization of mean flow time and proposed three types of heuristics based on matching heuristics, priority-dispatching rules, as well as insertion and appending algorithms combined with beam search. They also presented a new encoding scheme named rank matrices. Andresen et al. (2008) solved OSS with two metaheuristics, simulated annealing and genetic algorithms, with some operators. Like Bräsel et al. (2008), they used the rank matrix to encode a solution. Low and Yeh (2009) addressed OSS with some practical assumptions such as setup and removal times. To solve the problem, they presented a hybrid genetic algorithm.

The literature of scheduling is filled with different applied industrial assumptions to better represent the real nature of scheduling environments. One of the most prevailing and extremely favored assumptions by many researchers in real scheduling configurations is that there is no buffer between machines. In many industrial settings, due to characteristics of the jobs or the processing technology, the operations of a job must be performed without any interruption between machines. Applications of these scheduling problems can be found in hot-metal-rolling industries, where the heated metal has to undergo a set of operations at continuously high temperatures before it is cooled in order to prevent defects. Similarly, in the plastic molding and silverware production industries, a set of operations must be performed to immediately follow one another to prevent degradation. Other examples include chemical and pharmaceutical industries, food processing industries,

* Correspondence: Bahman_naderi62@yahoo.com
[1]Department of Industrial Engineering, Faculty of Engineering, University of Kharazmi, Karaj, Iran
Full list of author information is available at the end of the article

and advanced manufacturing environments. Hall and Sriskandarajah (1996) explained some other applications of no-buffer scheduling problems. Despite their practical applications, these problems are given far less attention than the other scheduling problems.

The two-machine open shop with no buffer (OSNB) was studied by Sidney and Sriskandarajah (1999). They introduced an approximation heuristic with a worst case performance ratio of 3:2 to solve the problem. Yao and Soewandi (2000) proposed four heuristics. One of them is a flowshop-based heuristic based on the Gilmore-Gomory algorithm. Two other heuristics are based on the 'matching' concept to minimize 'potential' idle time on either machine. The other one is a random search algorithm utilizing some features of other 'soflcomputing' metaheuristics. The experimental evaluation shows that the random search algorithm outperforms the others.

Yao and Lai (2002) proposed a genetic algorithm for the two-machine OSNB. The two-machine OSNB was also studied by Liaw et al. (2005). They developed a branch-and-bound with some dominance rules as well as a two-phase algorithm. OSNB with movable dedicated machines to minimize total occupation time of all the machines was considered by Lin et al. (2008). They introduced a mixed integer nonlinear program to model the problem. As far as we reviewed, almost none of the existing papers consider classical multi-machine OSNB. There is no attempt to linearly model the problem. There is only one metaheuristic (two-phase algorithm of Lin et al. (2008)) for a relevant problem.

The two-machine OSNB is strongly nondeterministic polynomial time-hard (NP-hard) (Sahni and Cho 1979). Therefore, the general multi-machine OSNB is NP-hard. The presentation of mathematical programs and heuristics is commonly used to solve scheduling problems (Stafford et al. 2005). We propose a mixed integer linear programming model to formulate the OSNB. We also carry out an experiment to analyze and compare the performance of the proposed model. Besides the model, the paper solves the large-sized instances by an effective metaheuristic based on an electromagnetism-like algorithm. Model's efficiency and metaheuristics' capability to solve the problem studied here are investigated on two computational evaluations including small- and large-sized instances.

The rest of the paper is organized as follows. The 'Problem formulation' section develops the mixed integer linear program. The 'Proposed electromagnetism algorithm' section introduces the proposed metaheuristic. The 'Computational evaluation' section describes the experimental design to evaluate the proposed method including the mathematical model and algorithm. Finally, the 'Conclusions and future studies' section gives some interesting conclusions and future studies.

Problem formulation

Even though mixed integer linear programming (MILP) models might not be an efficient solution method for all problem sizes, they are a natural way to attack scheduling problems (Stafford et al. 2005). This section presents a model to formulate OSNB problems. We analyze the model by the number of binary and continuous variables as well as the number of constraints required in formulating the problem. The following notations are used in the model:

Parameters:

- n The number of jobs.
- m The number of machines.
- $O_{j,i}$ The operation of job j on machine i.
- $p_{j,i}$ The processing time of $O_{j,i}$.
- d_j The due date of job j.
- M A large positive number.

Binary variables (BVs) used in this model show the relative sequence of different operations of a job as well as the relative order of the jobs on each machine. Due to the identity of these BVs, the model is called the sequence-based model. Note that for every machine we introduce, a dummy job 0 which precedes the first job on that machine is present and that for every job, a dummy machine 0 that precedes the first operation of that job is also present. It is necessary to indicate that the model requires the big M. The following notations are established:

Indices:

- k Job index $\{0, 1, 2, \ldots, n\}$.
- j Job index $\{1, 2, \ldots, n\}$.
- l Machine index $\{0, 1, 2, \ldots, m\}$.
- i Machine index $\{1, 2, \ldots, m\}$.

Variables:

- $Y_{l,j,i}$ Binary variable that takes value 1 if $O_{j,i}$ is processed immediately after $O_{j,l}$, and 0 otherwise. $l \neq i$
- $X_{k,j,i}$ Binary variable that takes value 1 if $O_{j,i}$ is processed immediately after $O_{k,i}$, and 0 otherwise. $k \neq j$
- T_j Tardiness of job j.
- $S_{j,i}$ The starting time of $O_{j,i}$.
- B_j The starting time of job j.

The model formulates OSNB as follows:

$$\text{Minimize} \sum_{j=1}^{n} T_j,$$

which is subject to the following:

$$S_{j,i} - B_j \leq \sum_{l=1, l\neq i}^{m} p_{j,l} \qquad \forall_{j,i} \qquad (1)$$

$$S_{j,i} - B_j \geq p_{j,i} \qquad \forall_{j,i} \qquad (2)$$

$$S_{j,i} - S_{j,l} - p_{j,l} \geq M\left(Y_{l,j,i} - 1\right) \qquad \forall_{j,i,l\neq i} \qquad (3)$$

$$S_{j,i} - S_{k,i} - p_{k,i} \geq M\left(X_{k,j,i} - 1\right) \qquad \forall_{j,i,k \neq j} \qquad (4)$$

$$T_j - S_{j,i} \geq p_{j,i} - d_j \qquad \forall_{j,i} \qquad (5)$$

$$\sum_{k=0,k\neq j}^{n} X_{k,j,i} = 1 \qquad \forall_{j,i} \qquad (6)$$

$$\sum_{l=0,l\neq i}^{m} Y_{l,j,i} = 1 \qquad \forall_{j,i} \qquad (7)$$

$$\sum_{j=1,j\neq k}^{n} X_{k,j,i} \leq 1 \qquad \forall_{i,k\varepsilon\{1,2,..,n\}} \qquad (8)$$

$$\sum_{i=1,i\neq l}^{m} Y_{l,j,i} \leq 1 \qquad \forall_{j,l\varepsilon\{1,2,..,m\}} \qquad (9)$$

$$\sum_{j=1}^{n} X_{0,j,i} = 1 \qquad \forall_i \qquad (10)$$

$$\sum_{i=1}^{m} Y_{0,j,i} = 1 \qquad \forall_j \qquad (11)$$

$$X_{k,j,i}, Y_{l,j,i} \in \{0,1\} \qquad (12)$$

$$S_{j,i}, B_j, T_j \geq 0 \qquad (13)$$

Constraint sets (1) and (2) are to assure that no-buffer restrictions are held. Constraint set (3) ensures that $O_{j,i}$ cannot start before $O_{j,i}$ completes if $O_{j,i}$ precedes $O_{j,i}$. Constraint set (4) specifies that if $O_{j,i}$ is processed immediately after $O_{k,i}$, it cannot begin before $O_{k,i}$ completes. Constraint set (5) calculates tardiness of jobs. Constraint sets (6) and (7) state that every operation is scheduled once. Constraint sets (8) and (9) ensure that every operation must have at most one succeeding operation in the processing route of the jobs and in the job order of every machine. Constraint sets (10) and (11) enforce that dummy job 0 and machine 0 must have exactly one successor. Ultimately, Constraint sets (12) and (13) define the decision variables.

The model is analyzed with respect to the required numbers of binary variables, continuous variables, and constraints for a problem with n jobs and m machines. The model requires $nm(n+m)$ binary variables, i.e., $O(nm(n+m))$, $2n+nm$ continuous variables, i.e., $O(nm)$, and $nm\left(4+\frac{3n}{2}+\frac{3m}{2}\right)+n+m$ constraints, i.e., $O(nm(n+m))$.

The proposed electromagnetism algorithm

Electromagnetism algorithm (or EA) is a solution method for optimization problems with bounded variables. It is a population-based metaheuristic firstly proposed by Bïrbïl and Fang (2003). EA is inspired from the attraction-repulsion mechanism of the electromagnetism theory. Each candidate solution is regarded as a particle whose charge is related to its objective function value. The direction and magnitude of attraction or repulsion over each candidate solution in the population are computed by this charge. The direction of this charge for candidate solution i is determined by vectorially adding the forces from each of the other solutions on candidate solution i. In this mechanism, a candidate solution with good objective function value attracts the other ones, whereas a candidate solution with worse objective function repels the other candidate solutions. The better (or worse) the objective function value is, the higher the size of attraction (or repletion). After the movement for each candidate solution in the current population is made, a new population is generated. The procedure is repeated until a stopping criterion is met.

Figure 1 shows the general outline of EA which includes four phases: *initialization* of algorithm, computation of *total force* exerted on each particle, *movement* along the direction of the force, and *local search*.

Encoding scheme and initialization

The encoding scheme is the procedure of making a solution recognizable for algorithms and plays an important role in the effectiveness of the algorithms. The decoding scheme is a complementary procedure that turns an encoded solution into a schedule. The proposed encoding scheme has two main parts: a permutation showing relative order of the jobs rather than the operations and a matrix whose rows determine the relative sequence of operations of each job. In this case, we just determine the relative order of the jobs and the processing route of each job. For each job permutation, we assign the random key of 0 to the first job and 1 to the last one. The jobs (i.e., $n-2$ other jobs) in between are assigned real numbers of $\frac{1}{n-1},\ldots,\frac{n-2}{n-1}$, respectively. For each job j, we assign the random key of 0 to the first operation and 1 to the last operation. The operations in between are assigned real numbers $\frac{1}{n_i-1}$, $\frac{2}{n_i-1},\ldots,\frac{n_i-2}{n_i-1}$, respectively. For example, consider a problem with $n = 5$. One

Procedure ***Electromagnetism_algorithm***

Initialization
While ***stopping criterion is not met*** **do**
    Local search
    Computation of total forces
    Movement by forces
End while

Figure 1 The general outline of an EA.

```
Procedure Local_search

for k = 1 to popsize do
    iter = 0
    j = 1
    while j ≤ n do
        while iter < LSITER do
            The random key of job j in candidate solution k is randomly regenerated
            if improvement is made then
                The new generated random key is accepted
                iter = LSITER
                j = n

            elseif
                iter = iter + 1

            endif
        endwhile
        j = j + 1
    endwhile
endfor
```

Figure 2 The procedure of the local search.

possible permutation is {4, 2, 5, 1, 3}. The real values assigned to jobs 4 and 3 are 0 and 1, respectively. For middle jobs, the values $\frac{1}{4}, \frac{2}{4}$, and $\frac{3}{4}$ is assigned to jobs 2, 5 and 1, respectively.

The decoding scheme turns an encoded solution into a schedule. We set the starting time of the first job in part 1 as 0. Then, to find the earliest possible starting time of the subsequent job, we first set the starting time to be 0. The first operation of that job is scheduled. If any overlap happens between this operation and previously scheduled operations on the same machine, the starting time is set to as the first possible starting time that avoids any overlap. The second operation in the processing route of the job is then scheduled. Again, if any overlap occurs between this operation and previously scheduled operations on that machine, we have to restart from the first operation in

```
Procedure Computation_of_total_forces

for k = 1 to popsize do
    calculate (↓k)
    F_k = 0
endfor
for k = 1 to popsize do
    for l = 1 to popsize do
        if l ≠ k & f(x_l) ≠ f(x_k) do
            if f(x_l) < f(x_k) then
                F_k = F_k + (x_l − x_k) (q_k · q_l) / ||x_l − x_k||^2   (attraction)
            else
                F_k = F_k − (x_l − x_k) (q_k · q_l) / ||x_l − x_k||^2   (repulsion)
            endif
        endif
    endfor
endfor
```

Figure 3 The procedure of calculating *total forces*.

Procedure *Movement_by_ total_forces*

for k = 1 **to** *popsize* **do**
  if $k \neq best$ **then**
    β = random [0, 1]
    $F^k = \frac{F_k}{\|F_k\|}$ %Normalization of the total forces
    for j = 1 **to** n **do**
      if $F_j^k > 0$ **then** % F_j^k is j -th element of F^k
        $x_{kj} = x_{kj} + \beta \cdot F_j^k(1 + m_1 - x_{kj})$ % x_{kj} is random key of job j in solution k
      else
        $x_{kj} = x_{kj} + \beta \cdot F_j^k(x_{kj} - 1)$
      endif
    endfor
  endif
endfor

Figure 4 The procedure of *movement* by *total forces.*

order to hold no buffer constraints. The procedure repeats until all the operations of the job are scheduled.

Local search

To enhance the algorithm's performance, EA is hybridized with a local search. The procedure of this local search can be described as follows: for each candidate solution k, and then for a given job j, we generate a temporary random number between 0 and 1. The jobs are then sorted according to the random keys. Considering the previous example, the real values are $\{\frac{3}{4}, \frac{1}{4}, 1, 0, \frac{2}{4}\}$. Suppose the real value of job 2 is randomly regenerated and it becomes 47, therefore, the new solution is as follows: $\{\frac{3}{4}, \frac{4}{7}, 1, 0, \frac{2}{4}\}$.

If the new sequence results in a better objective function value, then the new permutation is accepted. Otherwise, it is left. This procedure iterates at most *LSITER* times. If any improvement is made in an iteration less than *LSITER*, then the search for the current candidate solution terminates. Figure 2 shows the procedure.

Computation of total forces

To obtain the total force exerted from the other solutions on candidate solution k, we first need to calculate the charge of that candidate solution. The charge of each candidate solution k is denoted by q_k and calculated by the following formula:

$$q_k = \exp\left(-n \times m \frac{f(x_k) - f(x_{\text{best}})}{\sum_{l=1}^{\text{popsize}} (f(x_l) - f(x_{\text{best}}))}\right), \forall_k \quad k = 1, 2, \ldots, \text{popsize}$$

where x_k and x_{best} are candidate solution k and the best candidate solution. This formula ensures that solutions with better objective values are assigned higher charges. The *total force* (F_k) vector exerted on candidate solution k is also obtained by the sum of all the forces exerted from each of other solutions in the current population on candidate solution k. If solution l is better than solution k (i.e., if $f(x_l) < f(x_k)$), it attracts solution k by the

Table 1 The results obtained by the model for OSNB

$n \times m$	Instances	Time
3 × 3	3	0.07
3 × 4	3	1.26
4 × 3	3	2.71
4 × 4	3	218
5 × 3	1	753
5 × 4	0	-

Table 2 The average RPD obtained by EA against TPSA in the Brucker et al. (1997) benchmark

Instance	EA	TPSA
j3	0.02	2.69
j4	0.82	2.17
j5	1.61	2.51
j6	1.79	3.22
j7	1.91	2.65
j8	2.43	4.73
Average	1.43	2.99

Table 3 The average RPD obtained by EA against TPSA in the Guéret and Prins (1998) benchmark

Instance	EA	TPSA
gp03	0.75	2.55
gp04	0.89	3.89
gp05	1.47	5.42
gp06	1.99	5.39
gp07	2.08	5.14
gp08	1.36	3.20
gp09	1.64	4.35
gp010	2.74	4.18
Average	1.62	4.26

force of $(x_l - x_k)\frac{q_k \cdot q_l}{\|x_l - x_k{}^2\|}$. Otherwise, (i.e., if $f(x_l) > f(x_k)$), it repulses solution k by the force of $(x_k - x_l)\frac{q_k \cdot q_l}{\|x_l - x_k{}^2\|}$. Figure 3 shows the procedure of calculation of *total force* vector exerted on candidate solutions.

Let us further illustrate the procedure by an example. Consider a problem with $n = 4$ and $m = 2$. We have popsize = 5 where $f(x_1) = 120$, $f(x_2) = 140$, $f(x_3) = 90$, $f(x_4) = 100$, and $f(x_5) = 110$. Therefore, we have $q_1 = 0.11$, $q_2 = 0.03$, $q_3 = 1$, $q_4 = 0.48$, and $q_5 = 0.23$. Suppose $x_1 = \{2.87, 2.05, 1.82, 2.52\}$ and $x_4 = \{1.22, 2.65, 1.43, 2.12\}$. The vector force exerted from x_4 on x_1 becomes as follows:

$$\left(\begin{bmatrix}1.22\\2.56\\1.43\\2.12\end{bmatrix} - \begin{bmatrix}2.87\\2.05\\1.82\\2.52\end{bmatrix}\right)\frac{0.11 \times 0.48}{1.84} = \begin{bmatrix}-0.06\\0.01\\-0.01\\-0.02\end{bmatrix}$$

Movement by total forces

To move according to the total forces obtained in the previous step, we first normalize the vector F_k. If $F_j^k > 0$ (F_j^k is the j th element of the normalized vector F_k), x_{kj} is increased (x_{kj} is the random key of job j in solution k). Otherwise, it is decreased. This move for each candidate solution is in direction of *total force* exerted on it by a random step length. This length is generated from a uniform distribution between [0, 1]. By selecting random length, we can guarantee that candidate solutions have a nonzero probability to move to the unvisited solution along this direction. Moreover by normalizing *total force* exerted on each candidate solution, we can avoid producing infeasible solutions. All the candidate solutions are moved with the exception of the current best solution. Figure 4 shows the outline of *movement* procedure.

Table 4 The average RPD obtained by EA against TPSA in the Taillard (1993) benchmark

Instance	EA	TPSA
Tail 4 × 4	0.42	3.16
Tail 5 × 5	2.20	3.58
Tail 7 × 7	1.86	2.49
Tail 10 × 10	2.10	4.58
Tail 20 × 20	2.03	6.10
Average	1.72	3.98

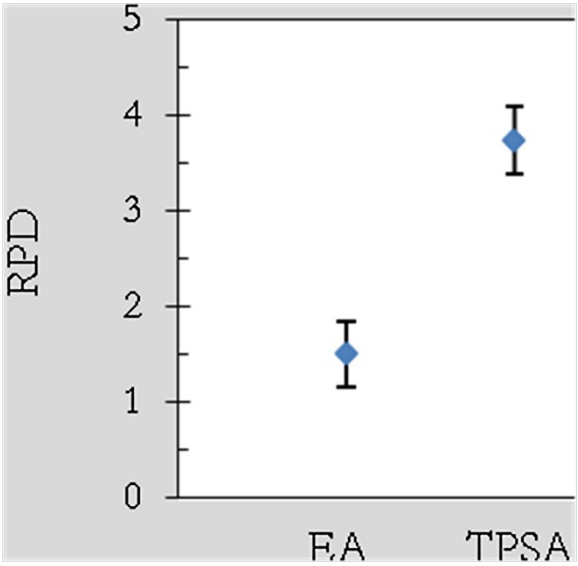

Figure 5 Mean plots and LSD intervals for the metaheuristics.

Computational evaluation

This section first evaluates the efficiency of the proposed MILP model on a computational experiment including small-sized instances. Afterwards, it investigates the general performance of the proposed metaheuristic against the optimal points obtained by the model. Moreover, it further evaluates the performance of the algorithms

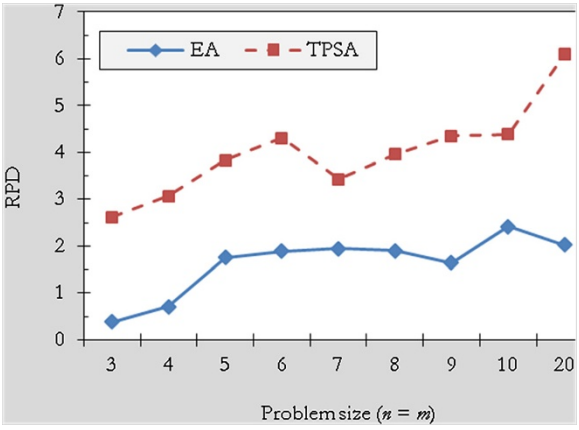

Figure 6 Mean plots for the metaheuristics versus the problem size.

against benchmarks in the literature of basic open shops. Since existing optimal solutions of the benchmarks are for basic open shops, they cannot be used for the evaluation. In this case, we use a performance measure named relative percentage deviation (RPD) obtained by the following formula:

$$\mathrm{RPD} = \frac{\mathrm{Alg}_{\mathrm{sol}} - \mathrm{Min}_{\mathrm{sol}}}{\mathrm{Min}_{\mathrm{sol}}} \times 100,$$

where $\mathrm{Min}_{\mathrm{sol}}$ and $\mathrm{Alg}_{\mathrm{sol}}$ are the lowest total tardiness for a given instance obtained by any of the algorithms and the solution obtained by a given algorithm. We implement the MILP models in CPLEX 10 and the other algorithms in Borland C++ and run on a PC with 2.0-GHz Intel Core 2 Duo and 2 GB of RAM memory. The stopping criterion used when testing all instances with the metaheuristics is set to a computational time limit fixed to $n \times m \times 0.5$ s. This stopping criterion permits for more time as the number of jobs or machines increases. Our proposed electromagnetism algorithm includes only one parameter of popsize. To set this parameter, we consider the sizes of 5, 10, and 20. We randomly generate 30 instances in different sizes and solve them by the three EA obtained by different levels of popsize. The popsize of 10 obtained the best results.

Evaluation of the MILP model

This section first evaluates the efficiency of the MILP model to solve NW-OSS problems. We generate a set of different instances as follows. We have 10 problem sizes ranging from $(n \times m) = (3 \times 3)$ up to (5×4). The processing times are randomly distributed over (1, 99). For each problem size, we generate three instances. The MILP model is allowed a maximum of 900 s (15 min) of computational time. Table 1 shows the results obtained by the model including the number of instances solved to optimality by the model in each problem size and the average computational time required to solve the instances. The model is capable of solving instances up to (5×3).

Evaluation of the metaheuristic

This section evaluates the algorithm against the optimal solution obtained by the model in the previous subsection. EA optimally solves 12 instances out of 13 instances. After having investigated the general performance of the proposed electromagnetism algorithm, we further evaluate the proposed algorithm against the adaptation of a relevant algorithm, the two-phase solution algorithm (TPSA) of Lin et al. (2008). To do so, we use standard benchmarks in the literature, such as those of Taillard (1993), Brucker et al. (1997), and Guéret and Prins (1998).

We use RPD measure to compare the algorithms. Tables 2, 3, and 4 show the results of the experiments for benchmarks of Brucker et al. (1997), Guéret and Prins (1998), and Taillard (1993), respectively. EA outperforms TPSA in all the three benchmarks. It obtains the RPD of 1.43%, 1.62%, and 1.72% in Brucker et al. (1997), Guéret and Prins (1998), and Taillard (1993) benchmarks, respectively. To further statistically analyze the results, we carry out an 'analysis of variance' or ANOVA. The results demonstrate that there are significant differences between the two algorithms with the p value very close to 0. Figure 5 shows the mean plots and least significant difference or LSD intervals at 95% confidence level for the different algorithms. As could be seen, EA statistically outperforms TPSA. It is also interesting to plot the performance of the algorithms versus the problem size. Figure 6 shows the means obtained by the algorithms in the different problem sizes. As shown, EA significantly performs well in all the problem sizes.

Conclusions and future studies

This paper studied the problem of open-shop scheduling, with no intermediate buffer to minimize total tardiness. To optimally solve the problem, one mathematical model in the form of mixed integer linear program is developed. Next, we proposed a metaheuristic in form of an electromagnetism algorithm to solve the large-sized problems in an acceptable computational time. We carried out two computational experiments to evaluate the performances of model and metaheuristic. In the first one, we had some small-sized instances by which we assessed the mathematical model and evaluated the general performance of the proposed metaheuristic. In the second experiment, the proposed algorithm was further evaluated against an algorithm in the literature. All the results supported that the model and metaheuristic were effective to tackle open-shop problems with no intermediate buffer between machines. As an interesting future research, one can study the multi-objective case of the problem under consideration. Another impressive research is to present a branch-and-bound or any other exact method for the problem.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

BN, EN, and MY formulated the problem of scheduling no-wait open shops with no intermediate buffer. To solve the problem, we proposed a metaheuristic in the form of electromagnetism-like algorithm. All authors read and approved the final manuscript.

Author details

[1]Department of Industrial Engineering, Faculty of Engineering, University of Kharazmi, Karaj, Iran. [2]Department of Industrial Engineering, Science &

Research Branch, Islamic Azad University, Tehran, Iran. [3]Department of Industrial Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran.

Received: 25 June 2012 Accepted: 21 November 2012
Published: 27 December 2012

References

Andresen M, Bräsel H, Morig M, Tusch J, Werner F, Willenius P (2008) Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. Math Comput Model 48(7–8):1279–1293

Bïrbïl I, Fang SC (2003) An electromagnetism-like mechanism for global optimization. J Glob Optim 25:263–282

Bräsel H, Herms A, Morig M, Tautenhahn T, Tusch T, Werner F (2008) Heuristic constructive algorithms for open shop scheduling to minimize mean flow time. Eur J Oper Res 189(3):856–870

Brucker P, Hurink J, Jurisch B, Wöstmann BA (1997) A branch and bound algorithm for the open-shop problem. Discret Appl Math 76:43–59

Guéret C, Prins C (1998) Classical and new heuristics for the open shop problem: a computational evaluation. Eur J Oper Res 107:306–314

Hall NC, Sriskandarajah CA (1996) Survey of machine scheduling problems with blocking and no-wait in process. Oper Res 44:510–525

Liaw CF, Cheng CY, Chen M (2005) Scheduling two-machine no-wait open shops to minimize makespan. Comput Oper Res 32:901–917

Lin HT, Lee HT, Pan WJ (2008) Heuristics for scheduling in a no-wait open shop with movable dedicated machines. Int J Prod Econ 111(2):368–377

Low C, Yeh Y (2009) Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated. Robot Comput Integr Manuf 25(2):314–322

Pinedo ML (2008) Scheduling: theory, algorithms, and systems, 3rd edn. Springer Science + Business Media, New York

Sahni S, Cho Y (1979) Complexity of scheduling shops with no-wait in process. Math Oper Res 4:448–457

Sidney JB, Sriskandarajah C (1999) A heuristic for the two-machine no-wait open shop scheduling problem. Nav Res Logist 46:129–145

Stafford EF Jr, Tseng FT, Gupta JND (2005) Comparative evaluation of MILP flowshop models. J Oper Res Soc 56:88–101

Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285

Yao MJ, Lai CW (2002) A genetic algorithm for the two machine open shop scheduling problem with blocking. In: The second Japanese-Sino optimization meeting (JSOM 2002), Rihga Royal Hotel Kyoto, Kyoto, 25–27 Sept 2002

Yao MJ, Soewandi H (2000) Simple heuristics for the two machine open shop problem with blocking. J Chin Inst Ind Eng 17(5):537–547

doi:10.1186/2251-712X-8-29
Cite this article as: Naderi *et al.*: **An electromagnetism-like metaheuristic for open-shop problems with no buffer.** *Journal of Industrial Engineering International* 2012 **8**:29.