International Journal of Industrial Engineering & Production Research (2016)

June 2016, Volume 27, Number 2
pp. 99-107

http://IJIEPR.iust.ac.ir/

Artificial Immune System for Single Machine Scheduling and Batching in a Supply Chain Scheduling Problem

Morteza Rasti-Barzoki[1,*], Ali Kourank Beheshti[2] & Seyed Reza Hejazi[3]

Morteza Rasti-Barzoki, *Assistant Professor, Department of Industrial and Systems Engineering, Isfahan University of Technology*
Ali Kourank Beheshti, *Ph.D., Department of Industrial and Systems Engineering, Isfahan University of Technology,*
Seyed Reza Hejazi, *Professor, Department of Industrial and Systems Engineering, Isfahan University of Technology,*

KEYWORDS

Supply chain management,
Scheduling,
Distribution,
Batching,
Artificial immune system

ABSTRACT

This paper addresses a production and outbound distribution scheduling problem in which several jobs have to be processed on a single machine for delivery to customers or to other machines for further processing. We assume that there is a sufficient number of vehicles. Also, it is assumed that the delivery cost is independent of batch size, but it is dependent on each trip. In this paper, we present an Artificial Immune System (AIS) for this problem. The objective is to minimize the sum of the total weighted number of tardy jobs and the batch delivery costs. A batch setup time has to be added before processing the first job in each batch. Using computational test, we compare our method with an existing method for the mentioned problem in literature, namely Simulated Annealing (SA). Computational tests show the significant improvement of the AIS over the SA.

1. Introduction

Two key operational functions in a supply chain scheduling are production and distribution operations. In a supply chain, it is critical to integrate or simultaneously consider these two functions and plan and schedule them jointly in a coordinated manner to achieve an optimal operational performance. Classical scheduling problems did not consider distribution and delivery cost, so considering both the delivery cost and scheduling objective in integrated model is an important subject. Chen [1] reviewed the Integrated Production and Outbound Distribution Scheduling, namely IPODS, models and classified these problems into five groups. Problems with an objective function that consider both the machine scheduling and delivery are rather complex. However, they are more practical. But, the body of literature on combined-optimization batch delivery problems, especially with large size solution, is small [2]. Hall and Potts studied the problem of production

* **Corresponding author: Morteza Rasti-Barzoki**
Email: *rasti@cc.iut.ac.ir*
Received 21 June 2015; revised 12 April 2016; accepted 25 April 2016

scheduling on a single machine under the batch availability assumption (distribution scheduling) with several objectives including the sum of flow times, maximum lateness, and the number of late jobs. Batch availability assumption means that all the jobs forming a batch become available for later processing or dispatch only when the entire batch has been processed [3-6]. They presented dynamic-programming algorithms for minimizing the mentioned objectives with delivery costs when the batches are to be delivered to several customers separately [3].

This paper addresses the minimizing sum of total weighted number of tardy jobs and delivery costs for multi-customer in a single machine environment and presents an AIS algorithm for solving it for the first time. The minimum number of tardy jobs, i.e., $1//\sum U_j$, is obtained by the polynomial Moore's algorithm for the single machine environment [7]. The weighted version of problem, i.e., $1//\sum w_j U_j$, is hard [8]. For $1//\sum w_j U_j$ A, a Fully Polynomial Time Approximation Scheme, FPTAS, was provided by Sahni [9]. Later, Gens and Levner improved it twice [10, 11]. In addition, Hallah and Bulfin developed Branch and Bound, B&B, method for this problem considering zero ready time and non-zero ready time [12, 13]. Hochbaum and Landy proposed a dynamic programming algorithm for the batching version of the problem, i.e., $1/s/\sum w_j U_j$, in which jobs are processed in batches which require setup time s [14], and later Brucker and Kovalyov improved it [15]. Nevertheless, none of these studies considered the delivery costs. Steiner and Zhang addressed the similar problem, i.e., scheduling and batching problem delivery to a customer, considering the minimizing sum of the total weighted number of tardy jobs and delivery costs on the single machine with batch setup time; they presented optimal properties and a pseudo-polynomial time DP algorithm for the optimal solution [16]. Also, they presented a pseudo-polynomial DP and an FPTAS for restricted case of multicustomer, where tardy jobs are delivered separately at the end of schedule [17]. Recently, Assarzadegan and Rasti-Barzoki [18] have studied the problem of minimizing the maximum tardiness, due date assignment, and delivery costs on a single machine. They presented two mathematical programming models and two metaheuristic algorithms, an adaptive genetic algorithm, and a parallel-simulated annealing algorithm, for solving it.

Some researchers have applied metaheuristic algorithms to solve scheduling problem [19-21]. In this research, we present an AIS method for this problem and compare it with SA and MINLP approaches introduced by Mazdeh et al. using computational test [22]. $1/s/V\ (\infty,\infty), direct/k/\sum_{j=1}^{n} w_j U_j + \sum_{k=1}^{K} \theta_k B_k$ is representation of our problems regarding notation that Chen [1] introduced for these types of problems. This notation means that there is the single machine for processing jobs with batch setup time, s, and sufficient vehicle by unlimited capacity and directing delivery method for sending the batches to K customers. *Directing delivery method* means that orders are transmitted to each customer without the routing problem. $\sum_{j=1}^{n} w_j U_j$ is the total weighted number of tardy jobs and $\sum_{k=1}^{K} \theta_k B_k$ is the total delivery costs where θ_k and B_k are delivery costs unit and the number of batches for each customer, respectively. Chen [1] presented an important review on the literature regarding integrated production and distribution scheduling models. So, we do not go into any more detail.

The rest of this paper is organized as follows: Section 2 contains the problem definition. AIS structure and our proposed algorithms are provided in section 3. Section 4 describes and analyzes the computational results. And, the last section contains our conclusions.

2. Notations and Problem Definition

2-1. Notations

Indexes

j Job index

k Customer index

Parameters:

n Number of jobs

K Number of customers

s_k Batch setup time for jobs belong to customer k

p_j Processing time of job j

d_j Due date of job j

w_j Weight of job j

θ_k Delivery costs for sending batch to customer k

Decision variables:

U_j One if job j be tardy and zero otherwise

B_k Number of batches for customer k

Immune system notations:

N Number of iteration in local search

PS Population size

α_i Local optimum factor in iteration i

σ Mutation rate

f Affinity value of antibody

δ Control factor of decay

2-2. Problem definition

There are K customers and one manufacturer in which each customer k orders n_k jobs to the manufacturer and $n = \sum_{k=1}^{K} n_k$ is the number of jobs. No job can be preempted. Each job has an important coefficient w_j. It is assumed that jobs need one operation and manufacturer does it by a single machine, with p_j processing time. The due date of job j is d_j. Jobs are processed and sent in batches to each customer. A batch can contain jobs only for the same customer. This assumption is common in literature for example see [4-6, 16, 17, 23-25]. We assume that each batch has a sequence-independent-batch setup time s_k. There is a sufficient number of vehicles, and the delivery cost is independent of batch size, and it is shown by θ_k for customer k for each trip . The number of batches for customer k is represented by B_k which is a decision variable. The objective is to minimize the sum of the total weighted number of tardy jobs and delivery costs. So, as mentioned earlier, this problem can be shown by $1/s/V\ (\infty,\infty), direct/k/\sum_{j=1}^{n} w_j\, U_j + \sum_{k=1}^{K} \theta_k\, B_k$.

3. Artificial Immune System

The immune system is an information processing and self-learning system that offers inspiration to design AIS. In the last decade, the immune system has drawn significant attention as a potential source of inspiration for novel approaches to solve complex computational problems [26]. Some researchers used the AIS to solve the scheduling class problems [27-33]. In this paper, a metaheuristic algorithm based on AIS for the first time is used to minimize the total weighted number of tardy jobs and delivery costs in two-level supply chain.

There are several immune algorithms such as negative selection algorithm, clonal selection algorithm, and artificial immune networks. In this paper, solution procedure is based on the clonal selection algorithm, in which only the highest affinity antibodies proliferate. In order to understand the AIS, some preliminary biological terms are required to be characterized [34, 35]:

Immune cells: B-cells and T-cells are the two main groups of immune cells. These cells help recognize an almost limitless range of anti-genic patterns.

Antigens (Ag): These are disease-causing elements that are divided into two types of antigens: self and non-self. Non-self-antigens are disease-causing elements, whereas self-antigens are harmless to the body.

Antibodies (Ab): It is a molecule produced by a B-cell in response to an antigen and has the particular property of combining specifically with the antigen, which induced its formation.

In the biological process, when an antigen contacts with the immune system, it releases a set of B-cells, present in the immunological memory, with the function of identifying and eliminating the antigens. Those B-cells that recognize the antigens with a minimal affinity are chosen for cloning and the number of clones of a particular cell is defined according to its antigen affinity. The cells undergo somatic hypermutation after the cloning process in order to try to eliminate the antigen. The cloning and mutation processes are repeated until the antigen is eliminated. Finally, the cells with the highest affinity are included in the immunological memory[36].

Hypermutation and receptor editing are two important characteristics of the immune system. They help in the maturation of the progenies, as antibodies present in memory response must have a higher affinity than those in the earlier primary response. Hypermutation is similar to the mutation operator of the genetic algorithm. The difference lies in the rate of modification that depends on the antigenic affinity.

In general, the antibodies with lower antigenic affinity are hypermutated at a higher rate as compared to the antibodies with higher antigenic affinity. This phenomenon is known as receptor editing, which governs the hypermutation. The main task of hypermutation is to guide toward local optimal, whereas receptor editing helps to escape the local optima.

In the rest of this section, our AIS properties are introduced in detail.

3-1. Encoding schema

In the proposed algorithm, an antibody includes some genes, such that each gene shows the batch number of each job. This encoding scheme is shown in Figure 1. This Figure shows that job 1 places in batch 5, job 2 places in batch 2, and the other jobs in the similar way.

Jobs:	1	2	3	4	5	6	7	8	9	10
Batches:	5	2	2	4	1	3	5	2	1	3

Fig. 1. Antibody encoding

As mentioned before, [22] proposed simulated annealing algorithm to solve this problem. In their algorithm, solutions are encoded by a matrix depicted in Figure 2 where the rows represent the batches and the columns represent the customers. For instance, if the element in row 2 and column 1 is one, the first order of customer 1 is assigned

to batch 2. Therefore, for each solution, a matrix is formed with n^2elements, whereas base on our encoding, for each solution, an array is formed with n elements.

Decision variables
(X_{ij})

Batches $(1,\ldots,k)$ $\begin{pmatrix} 1 & 0 & \cdots \\ 0 & 1 & \\ \vdots & & \end{pmatrix}$

Fig. 2. Solution encoding of SA[22]

3-2. Affinity calculation

In the proposed algorithm, it is needed to calculate affinity of antibodies. Since the goal is to minimize the objective function and the affinity value should be maximized in the AIS algorithms, the minus objective function value is considered as the affinity value.

3-3. The proposed algorithm

The main framework of the proposed algorithm is described as follows:

1. Initialization.
2. **While** (has not met stop criterion) **do**
3. Local search.
4. Proliferation.
5. Hypermutation.
6. New generation.
7. **End.**

Each steps of this algorithm is described as follows:

I. Initialization

In this stage, a random initial population of size PS is created. For each antibody, the value of each gene is determined randomly in the range $[1, n]$ in which the value of each gene is unique. In this paper, with the help of initial experiment, the size of population (PS) is considered equal to 12.

II. Local search

For each antibody, the following process is done N times:

One gene is selected randomly. Then, the value of this gene that represents the related batch is changed to a new batch that includes either no job or the other jobs of the customer of that gene; consequently, the new solution is formed. Then, the affinity value of the new solution is calculated. If the blown equation is satisfied, the antibody will be replaced by the new solution.

$$\frac{AFV_{new}}{AFV_a} \leq 1 + \alpha_i \quad (1)$$

Where AFV_a is the affinity value of antibody, AFV_{new} is the affinity value of the new solution, and α_i is the local optimum factor in the iteration. This factor leads to escape algorithm from local optimum. At first, α value is equal to zero; when the best solution is not improved in three consecutive iterations, for the first time, its value will be equal to initial value. In each iteration, α value is decreased base on equation (2) as follows:

$$\alpha_{i+1} = \alpha_i - \frac{\alpha_{initial}}{ITN - IF + 1} \quad (2)$$

Where α_{i+1} is the local optimum factor in iteration $i+1$, α_i is the local optimum factor in iteration i, α_{initial} is the initial value of local optimum factor, ITN is the number of iteration, and IF is the iteration that the value of local optimum factor will be equal to initial value for the first time.

With this equation, the value of local optimum factor in the last iteration will be equal to zero. As a result, the diversity in the primary iterations is greater than the last iterations. In this paper, with the help of initial experiment, the values of $\alpha_{initial}$ and N are considered equal to 0.005 and 80, respectively.

III. Proliferation

In this process, some clones are produced from each antibody. As in Reisi and Moslehi[28], the following equation is used to calculate the number of clones that each antibody produces.

$$n_c = PS \times AFF \quad (3)$$

Where n_c is the number of clones, PS is the size of the initial population, and AFF is the cumulative probability of the antibody. For each antibody, AFF is obtained by dividing its affinity value by the sum of all the antibody affinities.

IV. Hypermutation

After the proliferation stage, the mutation operator is performed for each clone. In mutation procedure, one gene is selected randomly. Then, the value of this gene that represents its batch is changed to a new batch that includes either no job or the other jobs of the customer of that gene. As in Kumar et al.[34] and Agarwal et al.[35], the mutation rate of each clone is calculated based on the following equation:

$$\sigma = \exp(-\delta \times f) \quad (4)$$

Where σ is the mutation rate, δ is the control factor of decay, and f is the affinity value of antibody. In this paper, the value of δ is calculated based on equation (5):

$$\delta = -\frac{1}{\bar{f}} \quad (5)$$

where $\bar{f}$ is the average affinity value of the population.

V. New population

After the hypermutation process is done and the affinity of the hypermutated solutions are calculated, select the fixed number (d) of best antibody for the next generation. In this paper, based on initial experiment, d is considered equal to 40% of PS.

VI. Stop criterion

Using computational pre-test, the stop criterion is considered as follows: if the best solution is not improved after five consecutive iterations or after a total number of 200 iterations, the algorithm will be stopped.

4. Computational Results

In this section, in order to evaluate the performance of AIS, both the small- and medium-size problems are considered. The AIS and SA algorithms were coded using Matlab 2009 and run on a computer with a 2.93 GHz CPU and a 2.00 GB RAM. The MINLP model was coded in GAMS and solved by BONMIN solver, because our pre-test shows BONMIN is the most efficient solver for solving the mentioned problem. In small and medium-size problems, we have compared results of the proposed algorithm with MINLP and SA, respectively. The details will be given in the following related subsections.

4-1. Problems with $n = 4, 7$ *and* 10

The number of jobs in small-sized problems was set $4, 7$ *and* 10. The number of customers for each n was defined by a uniform distribution in the interval $[1, n]$. Processing times, batch setup times, and job weights were randomly-generated integers from the uniform distribution defined on $[1\ 100]$, $[0\ 0.5\bar{p}]$, and $[1\ 100]$, respectively. Based on the batch delivery costs values, we generated two classes of problems, namely A and B, for each given number of the job. For class A and class B, the intervals that the delivery costs were generated randomly are $[0\ \bar{w}]$ and $[0\ 10\bar{w}]$, respectively. For each class, we generated three subclasses, namely 1, 2, and 3, based on the due dates values; therefore, we have six groups, namely A-1, B-1, A-2, B-2, A-3, B-3. For groups (A-1, B-1), (A-2, B-2), and (A-3, B-3), the intervals that the due dates values were generated randomly are $[0\ 0.5n(\bar{s}+\bar{p})]$, $[0\ n(\bar{s}+\bar{p})]$ and $[0\ 5n(\bar{s}+\bar{p})]$, respectively.

For each job number in each group, 10 problems were generated randomly. Hence, totally 180 (3*2*3*10) problems in small-sized problems were being generated and solved. A 300-second time constraint was considered, and if the problem could not be solved regarding this constraint, then the procedure would no longer be used for that problem. The results of the experiment for small-sized problems are shown in Table 1. Column "Number of the solution in which" of Table 1 shows that for all problems, AIS has produced objective function, i.e., total cost, less or equal to MINLP model. In detail, AIS has produced objective value the same as MINLP for 67.22% of problems and absolutely a better result for 32.78% of problems.

In problems with four jobs, both MINLP model and AIS algorithm have found the optimum solution for all problems in each group, but the average run time of AIS algorithm is smaller than the average run time of MINLP model for each group of four jobs. The average run time of problems with $n = 4$ is 0.38 second and 32.26 seconds for AIS and MINLP model, respectively. In problems with 7 and 10 jobs, the average of deviation in A-3 is smaller than A-2 and is smaller than A-1 in A-2. This means that as the due dates decrease the difference between AIS and MINLP, objective function increases. In addition, the average of deviation in B-1 is smaller than A-1; in B-2, is smaller than A-2; in B-3, is smaller than A-3. Therefore, as delivery costs decrease the difference between AIS and MINLP, objective function increases.

Tab. 1. The result of experiment for small-sized problems, comparing AIS with MINLP

n	Delivery Costs Cclasses	Due Date Subclass	Number of the solution in which			Ave. of CPU time (s)	
			AIS<MINLP*	AIS=MINLP	MINLP<AIS	MINLP	AIS
4	A	1	0	10	0	48.02	0.39
		2	0	10	0	41.62	0.37
		3	0	10	0	7.82	0.37
	B	1	0	10	0	30.95	0.36
		2	0	10	0	50.21	0.36

n	Delivery Costs Cclasses	Due Date Subclass	Number of the solution in which			Ave. of CPU time (s)	
			AIS<MINLP*	AIS=MINLP	MINLP<AIS	MINLP	AIS
7	A	3	0	10	0	14.97	0.41
		1	8	2	0	-	-
		2	6	4	0	-	-
		3	0	10	0	-	-
	B	1	3	7	0	-	-
		2	4	6	0	-	-
		3	0	10	0	-	-
10	A	1	9	1	0	-	-
		2	10	0	0	-	-
		3	1	9	0	-	-
	B	1	9	1	0	-	-
		2	8	2	0	-	-
		3	1	9	0	-	-

*AIS<MINLP means that AIS has a better result (less total cost) than MINLP

-Since some problems have not been solved within 300 seconds by GAMS, the ave. of CPU time could not calculated for them

4-2. Problems with $n = 50, 80, 110\ and\ 140$

In this section, we have compared results of our proposed algorithm, i.e., AIS, with SA proposed by [22] in medium-sized problems. The number of jobs in medium-sized problems was set $50, 80, 110, and\ 140$. All parameters were generated similar to the previous, but the number of problems for each job number in each group was set 20; hence, totally 480 (4*2*3*20) problems in medium-sized problems were generated. Table 2 shows the result of the computational test for this problems.

Tab. 2. The result of experiment in medium-sized problems, comparing AIS with SA

n	Delivery Costs Classes	Due Date Subclass	Number of the solution in which			Ave. of CPU time (s)		$\frac{SA-AIS}{AIS}$ (%)		$\frac{AIS-SA}{SA}$ (%)	
			AIS<SA	AIS=SA	SA<AIS	SA	AIS	Avg.	max	Avg.	max
50	A	1	15	0	5	1.500	5.495	3.328	18.613	0.647	5.687
		2	20	0	0	1.612	4.525	13.687	58.402	0.000	0.000
		3	14	6	0	1.193	3.049	51.752	475.000	0.000	0.000
	B	1	18	0	2	3.548	4.639	2.738	10.109	0.041	0.768
		2	16	0	4	2.840	5.031	1.992	10.634	0.043	0.412
		3	14	6	0	2.402	3.123	16.253	156.690	0.000	0.000
80	A	1	20	0	0	2.645	12.279	8.976	15.738	0	0
		2	19	0	1	2.564	10.788	39.997	109.396	0.089	1.772
		3	20	0	0	1.917	5.701	35.094	215.205	0	0
	B	1	20	0	0	4.939	12.007	6.011	28.028	0	0
		2	20	0	0	4.428	12.196	8.645	61.892	0	0
		3	19	1	0	4.190	7.401	24.106	155.787	0	0
110	A	1	20	0	0	4.022	22.207	14.758	42.654	0	0
		2	20	0	0	3.810	15.892	30.030	95.338	0	0
		3	18	2	0	2.895	9.891	50.717	300.000	0	0
	B	1	20	0	0	7.781	22.657	10.256	44.755	0	0
		2	20	0	0	7.302	20.461	13.377	75.304	0	0
		3	19	1	0	5.488	11.302	34.832	218.156	0	0
140	A	1	20	0	0	5.037	34.630	14.564	72.348	0	0
		2	20	0	0	5.250	26.626	45.090	88.867	0	0

n	Delivery Costs Classes	Due Date Subclass	Number of the solution in which			Ave. of CPU time (s)		$\frac{SA-AIS}{AIS}$ (%)		$\frac{AIS-SA}{SA}$ (%)	
			AIS<SA	AIS=SA	SA<AIS	SA	AIS	Avg.	max	Avg.	max
		3	19	1	0	3.963	17.211	108.856	595.625	0	0
		1	20	0	0	9.941	39.466	19.169	93.312	0	0
	B	2	20	0	0	10.794	31.236	28.379	89.023	0	0
		3	19	1	0	6.759	17.689	42.434	187.731	0	0

Table 2 shows that the AIS algorithm has found a better solution, less objective function than SA algorithm in 450 (93.75%) problems, and its objective function is equal to SA for 18 (3.75%) problems; hence, AIS has solved 97.5% of all problems with less equal total cost with respect to SA; SA has presented a better solution for only 2.5% of problems. However, the average run time of AIS is larger than SA. Column $\frac{SA-AIS}{AIS}$ shows the deviation of SA from AIS when AIS has presented the better result than SA; Column $\frac{AIS-SA}{SA}$ shows the deviation of AIS from SA when SA has presented the better result than AIS. The average deviation of SA from AIS for 93.75% of problems, for which AIS has presented better result than SA, is 26.04%, while the average deviation of AIS from SA for 2.5% of problems that SA has presented better result than AIS, is 0.20%. The maximum deviation for SA and AIS is 595.63% and 5.69% respectively. These results shows that AIS is more efficient than SA.

It is obvious from Table 2 that problems in class B has more average run time, for both SA and AIS, than problems in class A. So, as delivery costs increase, more time was required until the stopping criteria hold. In general, the $\frac{SA-AIS}{AIS}$ value for subclass 3 is greater than subclass 2, and for subclass 2 is greater than subclass 1. Therefore, as due dates increase, the deviation of SA from AIS increases.

5. Conclusion

This paper presents an AIS algorithm for the scheduling and batching a set of jobs on a single machine with batch setup time for delivery to customers. In order to evaluate the performance of the AIS algorithm, computational tests are used. The computational results show that the proposed AIS framework is more efficient than the MINLP and the SA proposed by [22]. Considering some constraints such as the number of vehicle and capacity for each vehicle, other machine configurations for a manufacturer, such as the parallel machine or flow shop, routing delivery method, instead of directing delivery method, can be suggested for future works. In addition, another function for the total costs, such as total weighted lateness and delivery costs are suggested as well.

References

[1] Chen ZL. Integrated production and outbound distribution scheduling: review and extensions, Operations Research, Vol. 58, No.1, (2010), pp. 130-148.

[2] Mazdeh MM, et al. Single-machine batch scheduling minimizing weighted flow times and delivery costs, Applied Mathematical Modelling, Vol. 35, No. 1, (2011), pp. 563-570.

[3] Hall NG, CN Potts. Supply chain scheduling: Batching and delivery, Operations Research, Vol. 51, No. 4, (2003), pp. 566-584+674.

[4] Rasti-Barzoki M, Hejazi SR. Minimizing the weighted number of tardy jobs with due date assignment and capacity-constrained deliveries for multiple customers in supply chains, European Journal of Operational Research, Vol. 228, No. 2, (2013), pp. 345-357.

[5] Rasti-Barzoki M, Hejazi SR. Pseudo-polynomial dynamic programming for an integrated due date assignment, resource allocation, production, and distribution scheduling model in supply chain scheduling, Applied Mathematical Modelling, (2015).

[6] Rasti-Barzoki M, Hejazi SR, Mazdeh MM. A Branch and Bound Algorithm to Minimize the Total Weighed Number of Tardy Jobs and Delivery Costs. Applied Mathematical Modelling, Vol. 37, No. 7, (2013), pp. 4924-4937.

[7] Moore JM. An n job, one machine sequencing algorithm for minimizing the

number of late jobs, Management Science, Vol. 15, No. 1, (1968), pp. 102-109.

[8] Karp RM. Reducibility among combinatorial problems, Springer, (1972).

[9] Sahni SK. Algorithms for scheduling independent tasks, Journal of the ACM (JACM), Vol. 23, No. 1, (1976), pp. 116-127.

[10] Gens GV, Levner EV. Discrete optimization problems and efficient approximate algorithms, Engineering Cybernetics, Vol. 17, No. 6, (1979), pp. 1-11.

[11] Gens GV, Levner EV. Fast approximation algorithm for job sequencing with deadlines. Discrete Applied Mathematics, Vol. 3, No. 4, (1981), pp. 313-318.

[12] M'Hallah R, Bulfin R. Minimizing the weighted number of tardy jobs on a single machine, European Journal of Operational Research, Vol. 145, No. 1, (2003), pp. 45-56.

[13] M'Hallah R, Bulfin R. Minimizing the weighted number of tardy jobs on a single machine with release dates, European Journal of Operational Research, Vol. 176, No. 2, (2007), pp. 727-744.

[14] Hochbaum DS, Landy D. Scheduling with batching: minimizing the weighted number of tardy jobs, Operations Research Letters, Vol. 16, No. 2, (1994), pp. 79-86.

[15] Brucker P, Kovalyov MY. Single machine batch scheduling to minimize the weighted number of late jobs, Mathematical Methods of Operations Research, Vol. 43, No. 1, (1996), pp. 1-8.

[16] Steiner G, Zhang R. Minimizing the weighted number of late jobs with batch setup times and delivery costs on a single machine, Multiprocessor Scheduling, (2007), pp. 85-98.

[17] Steiner G, Zhang R. Approximation algorithms for minimizing the total weighted number of late jobs with late deliveries in two-level supply chains, Journal of Scheduling, Vol. 12, No. 6, (2009), pp. 565-574.

[18] Assarzadegan P, Rasti-Barzoki M. Minimizing sum of the due date assignment costs, maximum tardiness and distribution costs in a supply chain scheduling problem, Applied Soft Computing.

[19] Rui Z, et al. An ant colony algorithm for job shop scheduling problem with tool flow, Proceedings of the Institution of Mechanical Engineers, Part B, Journal of Engineering Manufacture, (2014), pp. 0954405413514398.

[20] Raja K, et al. Earliness-tardiness scheduling on uniform parallel machines using simulated annealing and fuzzy logic approach, Proceedings of the Institution of Mechanical Engineers, Part B, Journal of Engineering Manufacture, Vol. 222, No. 2, (2008), pp. 333-346.

[21] Bathrinath S, et al. An improved meta-heuristic approach for solving identical parallel processor scheduling problem, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, (2015), pp. 0954405414564410.

[22] Mahdavi Mazdeh M, Hamidinia A, Karamouzian A. A mathematical model for weighted tardy jobs scheduling problem with a batched delivery system, International Journal of Industrial Engineering Computations, Vol. 2, No. 3, (2011), pp. 491-498.

[23] Steiner G, Zhang R. Minimizing the weighted number of tardy jobs with due date assignment and capacity-constrained deliveries, Annals of Operations Research, Vol. 191, No. 1, (2011), pp. 171-181.

[24] Assarzadegan P, Rasti-Barzoki M. Minimizing sum of the due date assignment costs, maximum tardiness and distribution costs in a supply chain scheduling problem, Applied Soft Computing, Vo. 47, (2016), pp. 343-356.

[25] Hassanzadeh A, Rasti-Barzoki M, Khosroshahi H. Two new meta-heuristics for a bi-objective supply chain scheduling problem in flow-shop environment, Applied Soft Computing, Vol. 49, (2016), pp. 335-351.

[26] Dasgupta D, Yu S, Nino F. Recent advances in artificial immune systems: models and applications, Applied Soft Computing, Vol. 11, No. 2, (2011), pp. 1574-1587.

[27] Amin-Tahmasbi H, Tavakkoli-Moghaddam R. Solving a bi-objective flowshop scheduling problem by a Multi-objective Immune System and comparing with SPEA2+ and SPGA. Advances in Engineering Software, Vol. 42, No. 10, (2011), pp. 772-779.

[28] Reisi M, Moslehi G. Minimizing the number of tardy jobs and maximum earliness in the single machine scheduling using an artificial immune system, The International Journal of Advanced Manufacturing Technology, Vol. 54, Nos. 5-8, (2011), pp. 749-756.

[29] Chandrasekaran M, et al. Solving job shop scheduling problems using artificial immune system, The International Journal of Advanced Manufacturing Technology, Vol. 31, Nos. 5-6, (2006), pp. 580-593.

[30] Ge HW, Sun L, Liang YC. Solving job-shop scheduling problems by a novel artificial immune system, in AI 2005: Advances in Artificial Intelligence, Springer, (2005), pp. 839-842.

[31] Engin O, Döyen A. A new approach to solve hybrid flow shop scheduling problems by artificial immune system, Future Generation Computer Systems, Vol. 20, No. 6, (2004), pp. 1083-1095.

[32] Coello CAC, Rivera DC, Cortes NC. Use of an artificial immune system for job shop scheduling, in Artificial Immune Systems, Springer, (2003), pp. 1-10.

[33] Gao J. A novel artificial immune system for solving multiobjective scheduling problems subject to special process constraint, Computers & Industrial Engineering, Vol. 58, No. 4, (2010), pp. 602-609.

[34] Kumar A, et al. Psycho-Clonal algorithm based approach to solve continuous flow shop scheduling problem, Expert Systems with Applications, Vol. 31, No. 3, (2006), pp. 504-514.

[35] Agarwal R, Tiwari M, Mukherjee S. Artificial immune system based approach for solving resource constraint project scheduling problem, The International Journal of Advanced Manufacturing Technology, Vol. 34, Nos. 5-6, (2007), pp. 584-593.

[36] Diana ROM, et al. An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimization, Neurocomputing, Vol. 163, (2015), pp. 94-105.