Int J Adv Manuf Technol (2013) 65:981–989
DOI 10.1007/s00170-012-4233-x

ORIGINAL ARTICLE

A hybrid imperialist competitive algorithm for single-machine scheduling problem with linear earliness and quadratic tardiness penalties

A. H. Banisadr · M. Zandieh · Iraj Mahdavi

Received: 20 April 2011 / Accepted: 8 May 2012 / Published online: 2 August 2012

Abstract In this paper, we study single-machine scheduling problem when each job is considered with linear earliness and quadratic tardiness penalties with no machine idle time. Here the objective is to find the best sequence of jobs in the reasonable time. This model was studied in several researches, and some algorithms were proposed to solve it such as genetic algorithm. As the size of problem increased, such algorithms were not effective and efficient. Hence, we proposed the hybrid imperialist competitive algorithm. The proposed algorithm is based upon the imperialist competitive algorithm and genetic algorithm concepts. This algorithm was tested in problems with different sizes. The results denoted that the hybrid algorithm can solve different size of problem in reasonable time. This procedure showed its efficiency in medium- and large-sized problems as compared with other available methods.

Keywords Hybridization · Imperialist competitive algorithm · Single-machine scheduling · Genetic algorithm

1 Introduction

Scheduling models with single facility rarely occur in practice. However this scheduling procedure occurs in several activities for example chemical industry [26]. In addition, the analysis of this scheduling procedure is very significant because the performance of many production systems is often determined by a single-bottleneck machine scheduling. The analysis of the single-machine subproblem is essential for complex system scheduling [5].

Scheduling problems with both earliness and tardiness penalties are compatible with supply chain management and just-in-time concepts. In these concepts, the objective is to produce goods only when they are required. Any earliness in producing goods causes costs such as holding cost, deterioration perishable goods, and opportunity costs. Any tardiness in producing goods results tardiness costs such as backlogging costs, lost sales, and lost goodwill. The linear earliness and quadratic tardiness penalties have been considered in this paper [21, 25]. When each job has distinct due date, the problem is NP-hard [7].

The scheduling problem with no machine idle time is suitable for most of the production systems. Some of the cases in which the assumption is used are as follows: limitation of the machine capacity accordant with the demand, high operational costs, and high costs of a new production which demands high setup costs and extended setup time. Some of the apparent instances of the production systems with no machine idle time have been proffered by Korman [13] and Landis [14].

Hybrid imperialist competitive algorithm (HICA) has been proposed to solve the problem in this article. This algorithm applies the concepts of two algorithms named imperialist competitive algorithm and genetic algorithm (GA) in a way that the high convergence speed of the ICA is used to obtain the initial solution and the genetic approach is used to achieve better final solution. Then, the performance of this algorithm is analyzed in a wide range of instances.

Literature review has been presented in Section 2. The problem has been described in Section 3, and the algorithm

A. H. Banisadr · I. Mahdavi
Department of Industrial Engineering,
Mazandaran University of Science and Technology,
Babol, Iran

M. Zandieh (✉)
Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, G.C.,
Tehran, Iran
e-mail: m_zandieh@sbu.ac.ir

to solve the problem has been proffered in Section 4. In Section 5, the computational results of the proposed approach have been compared with other algorithms. The conclusion of this study has been presented in Section 6.

2 Literature review

Sidney [20] is one of the researchers who studied the single-machine scheduling problem with earliness and tardiness penalties with distinct due date. He considered the problem of minimizing the maximum earliness or tardiness penalty. Baker and Scudder [6] provided a significant review of researches in this area until the end of the decade 1980. Several papers have presented the single-machine problem with a common due date for all jobs, for example Kanet [12], Hall [9], and Bagchi et al. [4]. Abdul-Razaq and Potts [1] and Ow and Morton [16] have considered earliness and tardiness problem with distinct due dates. The idle time was not regarded in these investigations. The problem of no machine idle time was presented by Korman [13] and Landis [14]. The problem with inserted idle time was considered by Schaller [19] in which a time table procedure was proposed.

Different approaches have been proffered to solve the problem in past. One of these approaches was branch and bound algorithm proposed by Valente [21]. Beam search and dispatching rule heuristics were suggested to solve the problem so far [15, 22, 23]. A genetic approach was presented by Valente and Goncalves [25]. Their proposed algorithm differed from the standard GA because the local search was used to obtain the initial population. This approach was more effective than the standard GA in the small-sized problem (at most 100 jobs).

A hybrid imperialist competitive algorithm is proposed in this article to solve the problem. So the related performance is explored in a wide range of instances. The imperialist competitive algorithm is one of the procedures applied in the hybrid algorithm. ICA was presented by Atashpaz and Lucas [3] which was used for solving the continuous optimization problems. This algorithm has lower computational time as compared with other calculation methods for instance the genetic approach. The concepts of ICA and GA approaches succor us to obtain an ideal final solution within an acceptable computational time for medium- and large-sized problems.

3 Problem description

The problem is focused to schedule a set of n independent jobs $\{J_1, J_2, \ldots, J_n\}$ on the single machine in which at most one job can handle at a moment. It is supposed that no idle time is allowed and the machine is continuously available. Each job requires a processing time p_i and distinct due date d_i. In this model, the earliness and tardiness of ith job are respectively defined as follows:

$$E_i = \max\{0,\ d_i - C_i\} \tag{1}$$

$$T_i = \max\{0,\ C_i - d_i\} \tag{2}$$

where C_i is the completion time of the ith job; thus, the objective function is designated as follows:

$$\sum_{i=1}^{n} \left(E_i + T_i^2\right) \tag{3}$$

4 Proposed algorithm description

In this section, the proposed solution procedure has been presented in details. Hence, the standard genetic algorithm and imperialist competitive algorithm ought to be expounded at first. The hybrid algorithm will be explained afterwards.

4.1 Genetic algorithm

Genetic algorithm is a statistical methodology which is applied to solve optimization problems. This procedure is based upon the evolutionary process on the natural biology. The basic principles of the genetic algorithm were first presented by Holland [11] in Michigan University. Holland [10] published the mathematics basics in a book entitled "*Adaptation in natural and artificial systems*." Then most of researchers described the genetic approach and its applications in their investigations [8, 17, 18].

In order to use genetic algorithm, the appropriate encoding method should be proposed for each specific problem first. A set of the problem parameters is considered as a string. The string is named chromosome in genetic terminology. A set of chromosomes is called population. Each chromosome is a solution or an individual in the current population. Each population has specific number of chromosomes which is called the population size. If the number of the chromosomes is small, only a little portion of the solution space will be explored by genetic algorithm. If the number of the chromosomes is so great, the algorithm will become so slow. Here, the population size is equal to the multiplication of *pop-mult* by the size of problem (number of jobs). *Pop-mult* is a parameter defined by the user. After identification of the number of solutions in the current population, the fitness value of each chromosome ought to be obtained.

4.1.1 Chromosome representation and fitness function description

Each set of chromosomes is composed of *n* jobs in which a uniform random number within the range of *0* and *1* is dedicated to each one of these jobs. Thus, the chromosome is coded based upon the vector of random numbers. The correspondent chromosome of each individual ought to be decoded, in which the jobs sequence ought to be obtained, so that the fitness value of each individual is determined. This can be achieved by arrangement of genes in each chromosome (refer to Fig. 1). The linear earliness and quadratic tardiness penalties can be achieved based upon the processing time and due dates of jobs. The fitness value of each chromosome is determined by these penalties.

The first generation is formed to explain the encoding procedure. The fitness value will be evaluated afterwards. The next generation is obtained through fundamental operators consequently available in genetics.

4.1.2 Reproduction strategy

The reproduction is a mechanism in which the best individual of the current generation is copied to the next generation. Hence, the best individual will not be lost and the least fitness value of the next generation will not escalate as compared with the previous one. In this article, the chromosome with the best fitness value is selected and then transferred to the next generation with no changes.

4.1.3 Crossover strategy

Two new chromosomes are produced in the crossover mechanism by exchange of some of the genes of a pair of the current chromosomes which have been selected on a random basis. The uniform crossover methodology has been chosen in this article.

4.1.4 Mutation strategy

Mutation operator has a vital role in avoiding the local optimal solutions and increasing of the convergence rapidness of the genetic algorithm. In this paper, one of the chromosomes is randomly selected. Then two genes will be randomly chosen and swapped their positions.

This evolutionary strategy is continued until a stopping criterion is satisfied. Here, the stopping criterion is the number of iterations with no improvements. The evolutionary strategy has been shown in Fig. 2.

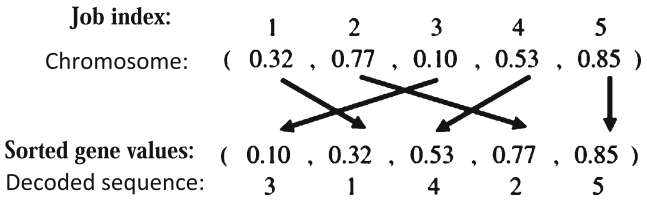

Fig. 1 Chromosome decoding example

4.2 The imperialistic competitive algorithm procedure

4.2.1 Imperialist competitive algorithm in general

Imperialist competitive algorithm is one of the new evolutionary algorithms to solve optimization problems, which uses the imperialist competitive concepts among countries. Atashpaz and Lucas [2 ,3] used this algorithm to solve the continuous optimization problems. ICA begins with the initial population like other evolutionary algorithms. Each one of the individuals of this population is known as a country. Such countries are equal to chromosomes in a genetic algorithm. In imperialist competitive algorithm, countries are classified as imperialists and colonies. An aggregation made up of one imperialist and several colonies is called an empire. The main core of this algorithm is composed of the imperialistic competition among these empires.

Imperialists and colonies ought to be determined in this algorithm at first. The best countries and the remaining ones are categorized as imperialists and colonies, respectively. The number of imperialists (*N_imp*) is a portion of the population size where *N_imp* is a parameter determined by the user. The number of the colonies belonging to each imperialist depends upon the cost of the imperialist. Hence, an imperialist with the higher cost than another one has lower power and fewer number of colonies.

The first strategy explained in this algorithm is the assimilation policy which aims to attract the culture and the structure of each colony toward its related imperialist. The colonies of an empire commence moving towards their

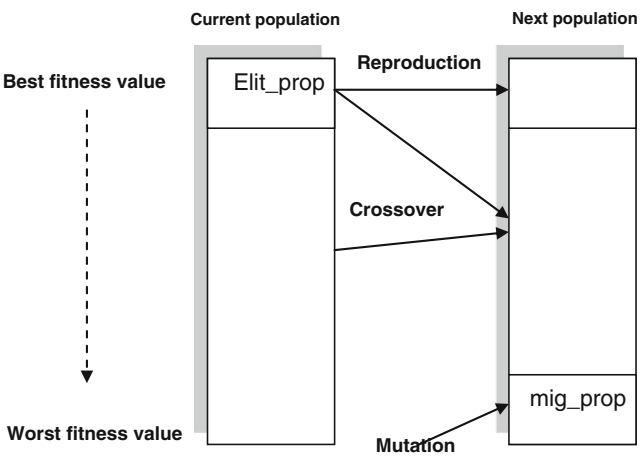

Fig. 2 Evolutionary strategy on GA

imperialist. The assimilation policy has been shown in Fig. 3.

Sometimes fitness value of one of colonies will be better than the relevant imperialist's fitness value in assimilation strategy. This is the situation where the algorithm changes the position of the imperialist and the colony. Then, this procedure continues with the new imperialist in a new position and colonies move towards the position of new imperialist.

The total power of empires ought to be calculated in the imperialistic competition. The total power of each empire can be obtained by the total power of the imperialist and that of all the colonies. In the imperialist competition, the weakest empire loses its own colonies and more powerful ones take possession of such colonies to increase their power. Hence, the weak empire is weakened and the more powerful one will increase its power during this strategy. The empire is collapsed when it loses all its colonies. Therefore all the empires will be lost except the most powerful one, and all the colonies will have the same positions and costs. Figure 4 displays a flowchart of the basic ICA.

4.2.2 Encoding and decoding procedures

A unified random number between 0 and 1 is used to encode the imperialistic competitive algorithm. So each country is encoded based upon a vector of random numbers. The length of this vector is equal to the number of the jobs.

Each country should be decoded to the corresponding solution (sequence of jobs) so that its fitness value can be obtained. Such a sequence is determined by a sorting of jobs.

4.2.3 Evolutionary strategy

Here we explain the evolutionary strategy by means of the imperialistic competition approach. The population size is obtained by the multiplication of the *pop-mult* and the size of problem. *pop_mult* is the user-defined

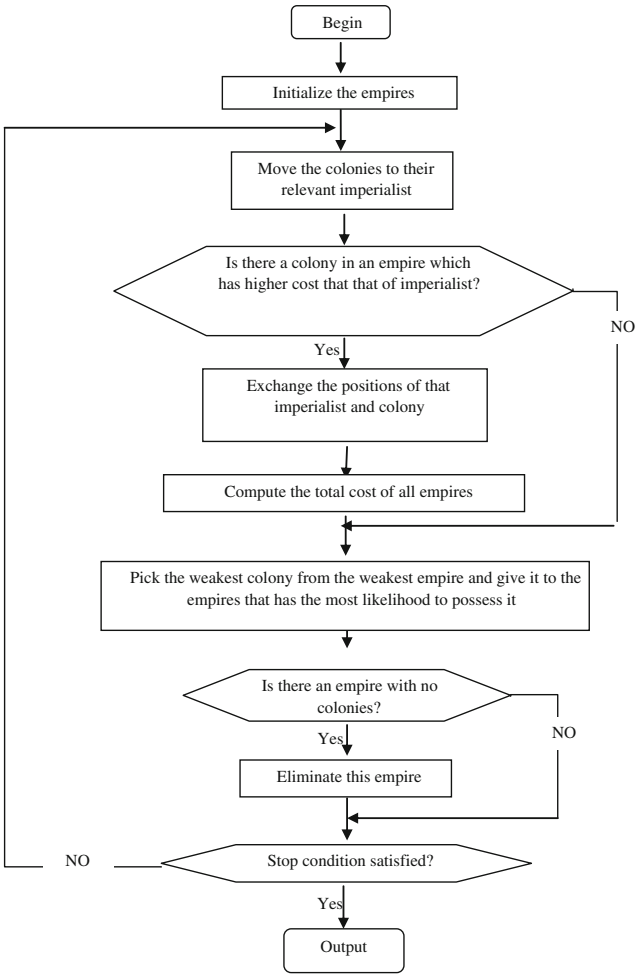

Fig. 4 Basic ICA flowchart

parameter and constant. The population size is kept fixed throughout the algorithm. The number of the imperialists (*N_imp*) and the colonies ought to be determined. Consequently the number of the imperialists can be obtained by multiple of the population size. *impp* is a proportion of the population size which is a user-defined parameter. Imperialists are selected from countries with higher fitness value than others. Number of colonies (*N_col*) is

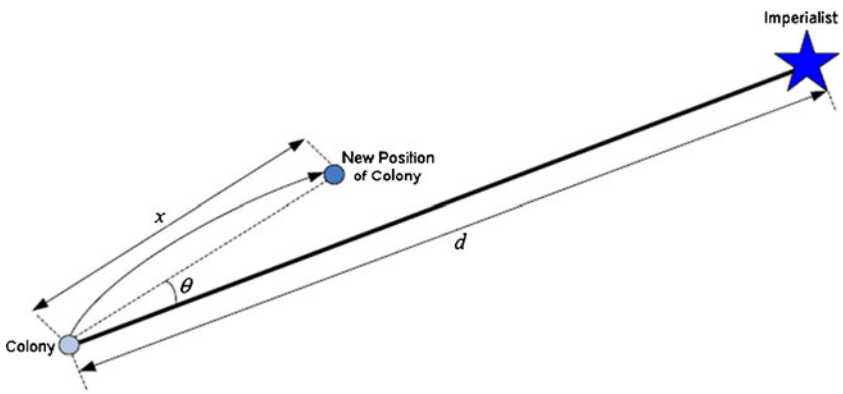

Fig. 3 Movement of colonies toward their relevant imperialist

achieved by:

$$N\_col = \text{populationsize} - N\_imp \tag{4}$$

To form the initial empires, the colonies are divided among imperialists based upon their power. The initial number of colonies of each empire is achieved by normalized comparative power of its imperialist. The initial number of ith empire's colonies $ncol\_imp(i)$ can be obtained by:

$$ncol\_imp(i) = round\,(p\_imp(i) \times N\_col) \tag{5}$$

where p_imp is the normalization comparative power of imperialist. The normalized power of ith imperialist is defined by:

$$p\_imp(i) = \left| \frac{NC_i mp(i)}{\sum_{i=1}^{N\_imp} NC\_imp(i)} \right| \tag{6}$$

The normalized cost of ith imperialist ($NC\_imp(i)$) can be achieved as follows:

$$NC\_imp(i) = C\_imp(i) - \max\{C\_imp(i)\}_i \tag{7}$$

where $C\_imp(i)$ is the cost of ith imperialist.

Evolutionary strategy is a mechanism which is used to generate a new generation from the current one. This mechanism is begun with the assimilation strategy in imperialist competitive algorithm where colonies move towards their imperialist's position. If the fitness value of one of colonies is better than the imperialist, the algorithm swaps the position of the colony and the imperialist. Hence, colonies move toward the position of the new imperialist.

The power of each empire can be achieved by the total costs for the imperialistic competition. Then the total cost of the empire can be obtained by the combining cost of the imperialist and a proportion of the cost of colonies.

$$TC\_emp(i) = C\_imp(i) + S \times \left(\sum_j tc\_col(i,j) \right) \tag{8}$$

where $TC\_emp(i)$ is the total cost of ith empire and $tc\_col(i, j)$ is the total cost of jth colony of ith empire. In addition, S is an attenuation coefficient that is used to reduce the affect of colonies cost. S is the number between 0 and 1.

To start the imperialistic competition, the possession probability of each empire must be obtained by its total power. The normalized total cost of ith empire ($NTC\_emp(i)$) will be:

$$NTC\_emp(i) = TC\_emp(i) - \max\{TC\_emp(i)\} \tag{9}$$

Then the power of each empire is computed as follows:

$$P\_emp(i) = \left| \frac{NTC\_emp(i)}{\sum_j NTC\_emp(j)} \right| \tag{10}$$

where $P\_emp(i)$ is the power of ith empire.

An empire with the highest total cost is the weakest one. The weakest colony is chosen from the weakest empire. The imperialistic competition commences among empires to take this colony afterward. An appropriate procedure is used for imperialistic competition with relation to the power of empires. The calculated power of empires is set in a vector named *P-emp*. Then another same-size vector called *R_emp* is made whose arrays should be unified random number between 0 and 1. A vector called *D_emp*, same-sized as these two vectors, can be given by:

$$D\_emp = P\_emp - R\_emp \tag{11}$$

The colony is allocated to one of empires based upon the *D-emp* Vector which has a greater quantity. The more powerful empire may have the higher quantity in *D_emp* Vector with more probability. The powerless empire will be collapsed during the imperialistic competition. This situation can be encountered when the empire losses all of its colonies.

This evolutionary strategy is continued until the stopping criterion is met. Here the stopping criterion is the number of the iterations without any improvement.

4.3 Hybridized ICA

When imperialist competitive algorithm was executed and compared with other evolutionary algorithms such as the genetic ones, we perceived that its computational time was better than GA's one; however, it did not yield a better final solution. Therefore, we proposed the hybrid imperialist competitive algorithm.

HICA is composed of two stages. The imperialist competitive algorithm is carried out in the first stage. Some of the countries with the best fitness value will be selected when the stopping criterion is met. These countries are the final solution of the first stage. Then the second stage is initiated. The initial population is included the chromosomes corresponding with the final solution of the first stage and other chromosomes created randomly. GA is implemented with this initial population. The chromosomes achieved by the first stage cause genetic procedure to be initiated with a total cost less than before. The second stage continues until the stopping criterion is met. A better final solution can be obtained in an acceptable computational time by utilization of this procedure. Such results are much better than the results of ICA and GA algorithms. The implementation procedure of HICA is described as follows:

1. The hybrid algorithm will be started with ICA procedure. At first we introduce initial parameters. N means the number of the countries, N_imp is the number of the

```
Generate initial population
 Calculate fitness values
 Create empires (Consider some of countries with best fitness values as imperialists and
remaining them as colonies)

    While (iteration_withoutimprove < stop _itr)
        Perform assimilation approach
        Calculate fitness value

        If fitness value of colony < fitness value of imperialist
            Swap colony and its imperialist
        End if

        Calculate empires power (summation ofimperialists'fitness value and
        proportion of their colonies)
        Select weakest colonies of weakest empires
        Perform imperialist competition strategy

        If new_bestsolution is found
            Update best solution
            Itration_withoutimprove=0
        Else
            Itration_withoutimpove=itration_withoutimprove+1
        End if

    End while

Select five best different best fitness values of country and insert them to chromosomes

Generate initial population (5 chromosomes are the best solutions of ICA and remaining
them are generated randomly)
Calculate fitness values of chromosomes (P)

    While ( itration_noimprove < stop_itr)
        Perform crossover;
        Perform mutation;
        Perform elitism;
        Calculate fitness values of newchromosomes (newP)
        Set P=newP

        If new_bestsolution is found
            Update best solution
            itration_noimprove=0
        Else
            itration_noimpove=itration_noimprove+1
        End if

    End while
```

Fig. 5 Main steps of hybrid imperialist competitive algorithm description

imperialists, *N_col* is the number of the colonies, where the following equality should be given.

$$N = N\_imp + N\_col \tag{12}$$

Here the number of the imperialist is obtained by multiple *impp* of total number of countries. Also, the

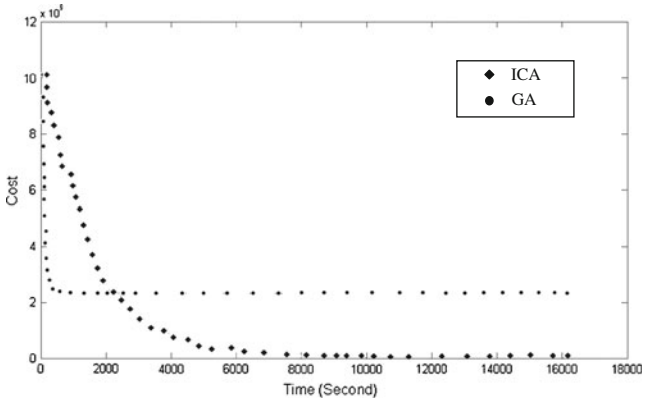

Fig. 6 Comparison chart between ICA and GA with 500 jobs

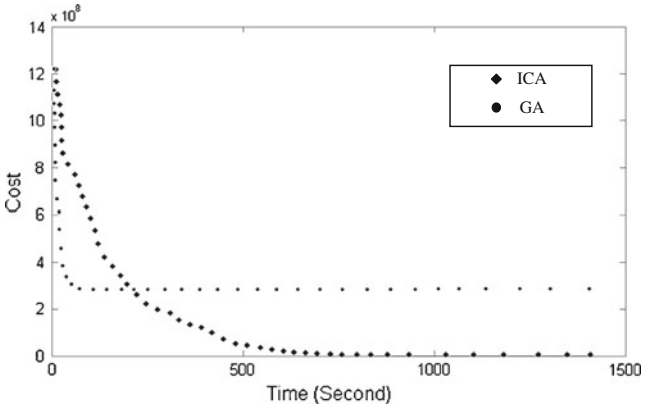

Fig. 7 Comparison chart between ICA and GA with 250 jobs

stopping criterion, the crossover (*cross_prop*) and mutation (*mig_prop*) parameters are proffered in this section.

2. The fitness value of all the countries is achieved. Then imperialists are selected from the most powerful countries which have the best fitness values. The remaining countries are the colonies. Such colonies are possessed by empires through normalized power of their imperialists. Therefore initial empires are obtained. Then ICA approach can be carried out.
3. The colonies of each empire move towards the position of imperialist in the assimilation strategy. The fitness value of each colony is determined after this policy. If the fitness value of each colony is better than the imperialist's one in each empire, their position will be swapped. The imperialistic competition begins afterwards. The weakest colony of the most powerless empire is chosen for the imperialistic competition. The *D_emp* Vector is formed afterwards, and one of the rival empires is selected to possess this colony. This procedure continues until the stopping criterion is met. The stopping criterion is the number of the generations with

Table 1 Comparison between GA and ICA costs with fixed time

N	Fixed time	Cost		Ratio
		GA	ICA	
150	12.1	1.50E+08	5.46E+07	3
200	30.6	3.35E+08	1.88E+08	2
250	70.0156	7.07E+08	2.87E+08	2
300	102.5	1.78E+09	8.81E+08	2
350	193.6	2.67E+09	1.17E+09	2
400	301.3	3.32E+09	1.91E+09	2
450	415.4	4.12E+09	1.38E+09	3
500	540.4	7.55E+09	2.36E+09	3
550	762.3	1.00E+10	4.62E+09	2
600	1,059.8	1.78E+10	6.17E+09	3

Table 2 Comparison between GA and ICA computational times with fixed cost

N	Fixed cost	Computational time		Ratio
		GA	ICA	
150	54,611,000	39.1	12.1	3
200	1.88E+08	78.6	30.6	3
250	2.87E+08	200.1	70	3
300	8.81E+08	256.2	102.5	3
350	1.17E+09	795.4	193.6	4
400	1.91E+09	946.3	301.3	3
450	1.38E+09	1,684.3	415.4	4
500	2.36E+09	2,330.8	540.5	4
550	4.62E+09	2,633.1	762.3	3
600	6.17E+09	4,481.4	1,059.8	4

no improvement. Some of the best countries with the best dissimilar fitness values will be selected. Here the number of these countries is considered five. The chromosomes similar to these countries are formed afterwards.

4. Genetic procedure is begun in the second stage. An initial population is formed in which five chromosomes inserted from the imperialistic competitive procedure and the remaining chromosomes have been produced randomly. The genetic algorithm is carried out afterwards until the stopping criterion is met.

The main steps in the proposed algorithm are summarized in the pseudo code shown in Fig. 5.

5 Computational results

We applied three algorithms (the genetic algorithm, the imperialist competitive algorithm, and the hybrid imperialist competitive algorithm) to solve the problem. These algorithms have been coded in Matlab 7.5 and executed in the Intel® Core(TM) 2 Duo 1.80 GHz. The hybrid algorithm has been compared with other algorithms by means of diverse instances. The different instances have different number of jobs, and each of them has been executed for five times. The processing time of each job has been generated by a uniform distribution random number between 1 and 100 for each instance. The distinct due dates are considered for each job which are obtained by a uniform random distribution between $P(1-T+R/2)$ and $P(1-T-R/2)$. Here, P is the sum of processing times of all jobs and T is the tardiness parameter and R is the range of due dates. T and R quantities are severally equal to 0.8 and 0.4 [24, 25].

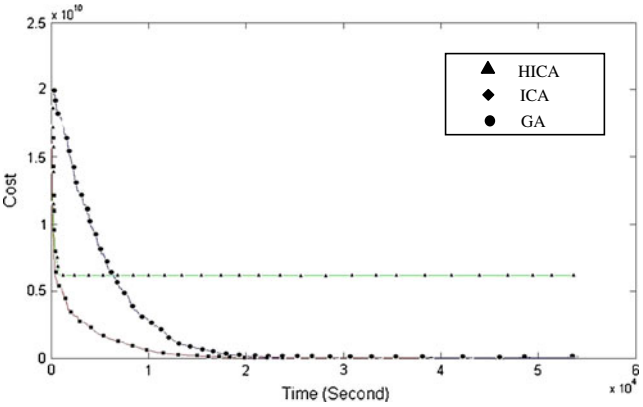

Fig. 8 Comparison among HICA, ICA, and GA with 600 jobs

The parameters related to imperialist competitive algorithm have been set as follows: population size is a multiple *pop_-mult* of the size of problem. The *pop-mult* quantity has been presumed to be 3. The crossover and mutation possibilities are 0.75 and 0.25, respectively, and the stopping criterion is equal to 100 generations without any improvement. The number of imperialists is a multiple *impp* of the population size. The *impp* quantity has been presumed to be 0.1.

At first we use the genetic and imperialist competitive algorithms to solve the problem. There is no apparent

Table 3 Comparison among three algorithms for computational time and final solution

Problem	Computational time			Final solution		
N	GA	ICA	HICA	GA	ICA	HICA
150	255.2313	13.35626	234.3125	258,502.4	59,959,690	135,423.8
200	531.5039	29.28752	502.2313	284,042	2.02E+08	260,993.8
250	1,460.48	63.0469	1,153.9	321,467.8	2.93E+08	303,694.4
300	4,416.52	109.6906	2,824.94	500,019.6	6.61E+08	460,578.2
350	5,170.74	158.3875	3,808.14	785,406.4	1.19E+09	603,092.8
400	10,102.52	207.3971	6,178.046	1,236,206	1.37E+09	1,077,498
450	12,604	225.6847	10,785	1,248,094	1.96E+09	1,111,476
500	15,587	394.8813	12,385	1,359,554	2.56E+09	1,162,506
550	30,747	470.8609	21,144.65	1,663,568	3.46E+09	1,453,846
600	52,606.33	469.043	33,977.37	1,887,919	6.38E+09	1,217,173

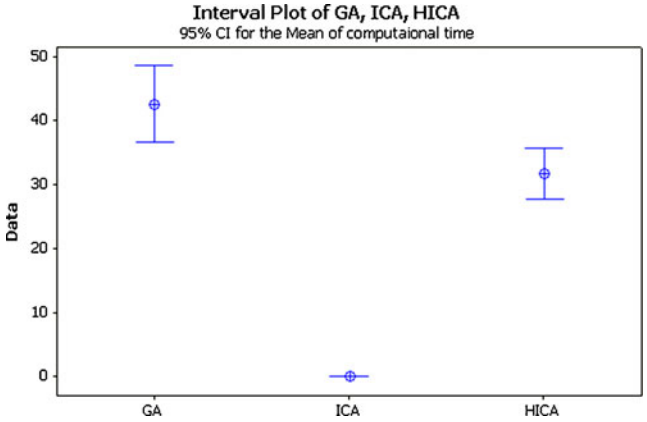

Fig. 9 Confidence interval of three algorithms for computational time

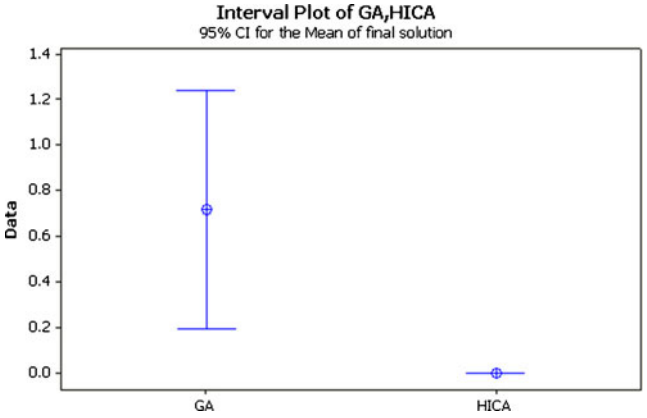

Fig. 11 Confidence interval of GA and HICA for final solution

difference in small-sized instances (smaller than 100 jobs) between these two algorithms. However the final solution of the genetic algorithm is better than ICA in medium- and large-sized instances (more than 150 jobs), but the computational time of GA is much more than ICA. The comparative performances of these two algorithms have been shown in Figs. 6 and 7.

We used two assumptions to show differences between two algorithms in different instances. One of them refers to fixed time (ICA convergence time). The costs of GA and ICA are then extracted on this time. Then we concluded that ICA costs better than those of GA with regard to the fixed time. The word "ratio" in Table 1 shows the proportion of GA costs to that of ICA. Another assumption concerns the fixed cost. The costs of these algorithms are assumed to be fixed and same. The computational times of two algorithms are extracted, and it is shown that the computational time of ICA with fixed cost is less than GA. The word "ratio" in Table 2 displays the proportion of the computational times of GA to ICA.

The hybrid imperialist competitive algorithm has been proposed to achieve appropriate final solution with acceptable computational time especially in medium- and large-sized instances. The performance of this algorithm was compared with GA and ICA. We perceived that the performance of this algorithm is better than others. Table 3 shows that HICA has more efficient performance for solving medium- and large-sized problems. The computational time of the hybrid algorithm is less than the genetic one. The final solution of the proposed algorithm is better than the other two algorithms. For instance in a problem whose number of jobs is 600, the final solution of GA is equal to 1,887,919 and its computational time is equal to 52,606 s, but the final solution of HICA is equal to 1,217,173 and the computational time is equal to 33,977 s. The average costs and computational time of five times of each instance implementation have been given in Table 3. An example has been presented in Fig. 8 to show comparative performances of every three algorithms. One example for comparison of HICA, ICA, and GA is shown in Fig. 8.

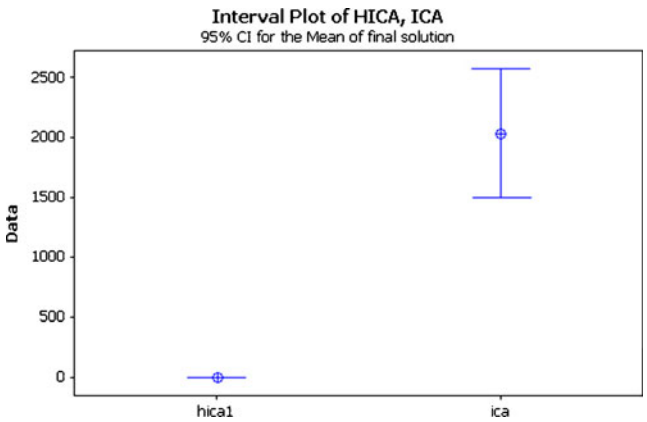

Fig. 10 Confidence interval of HICA and ICA for final solution

The results of three algorithms have been tested by Tukey's method. The normalized data have been used for this method. These normalized data have been computed as follows:

$$\text{Normalized } S_i = (S_i - \min(S_1 : S_3))/\min(S_1 : S_3) \qquad (13)$$

where S_i is the computational time or final solution of ith algorithm.

These normalized results should be tested by Tukey's method. An interval plot is regarded for the final solution and the computational time. The error ratio of these interval plots has been assumed 0.05. The interval plot of these algorithms has been shown for the computational time in Fig. 9. Interval plots of the three algorithms have been depicted for the final solution in Figs. 10 and 11.

We perceived that the means of normalized computational results of three algorithms have significant differences in Fig. 9, and also, a lot of notable differences among the conclusions of final solutions of ICA and the hybrid algorithm in Fig. 10 are observed. Moreover, the considerable difference is depicted between the means of normalized final solution of genetic algorithm and HICA in Fig. 11.

According to the results of Tukey's method, the results of three algorithms are proved consequently. Considering the executing of these three algorithms, we can clearly conclude the effectiveness of the proposed hybrid algorithm in comparison to the other ones.

6 Conclusion

We applied the hybrid imperialist competitive algorithm for single-machine scheduling with linear earliness and quadratic tardiness with no machine idle time. This approach is based upon the genetic algorithm and imperialist competitive algorithm concepts. The performance of the genetic algorithm and that of the imperialist competitive algorithm was compared in this problem. According to the computational results, it was observed that ICA has a lower computational time than GA but its final solution is worse than that of GA. Hence, HICA was proposed. The proposed algorithm was compared with GA and ICA. Thus, this algorithm achieved a better final solution with acceptable computational time as compared with the other two algorithms. This algorithm can be applied to solve medium- and large-sized problems.

For future researches, the following subjects can be considered:

- The machine idle time and setup time;
- The other type of objective function in single-machine scheduling environment such as quadratic earliness and tardiness penalties; and
- Other types of scheduling environment such as parallel machine, job shop, and so on.

References

1. Abdul-Razaq TS, Potts CN (1988) Dynamic programming state–space relaxation for single machine scheduling. J Oper Res Soc 39:141–152
2. Atashpaz E, Hashemzadeh G, Rajabioun F (2008) Colonial competition algorithm: a novel approach for PID controller design in MIMO distillation column process. Int J Intell Comput Cybern 1 (3):337–355
3. Atashpaz E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialist competitive. IEEE Congress on Evolutionary Computation 4661–4667
4. Bagchi U, Sullivan RS, Chang YL (1986) Minimizing mean absolute deviation of completion times about a common due date. Nav Res Logist Q 33:227–240
5. Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York
6. Baker KR, Scudder GD (1990) Sequencing with earliness and tardiness penalties: a review. Oper Res 38:22–36
7. Du J, Leung JY (1990) Minimizing total tardiness on one machine is NP-hard. Math Oper Res 15:483–495
8. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading
9. Hall NG (1986) Single and multiple-processor models for minimizing completion time variance. Nav Res Logist Q 33:49–54
10. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, re-issued by MIT Press (1992)
11. Holland JH (1962) Outline for a logical theory of adaptive systems. J Assoc Comput Mach 9(3):297–314
12. Kanet JJ (1981) Minimizing the average deviation of job completion times about a common due date. Nav Res Logist Q 28:643–651
13. Korman K (1994) A pressing matter. Video 46–50
14. Landis K (1993) Group technology and cellular manufacturing in the Westvaco Los Angeles VH department. Project report in IOM 581, School of Business, University of Southern California
15. Ow PS, Morton TE (1988) Filtered beam search in scheduling. Int J Prod Res 26:35–62
16. Ow PS, Morton TE (1989) The single machine early/tardy problem. Manag Sci 35:177–191
17. Reeves CR (1997) Genetic algorithms for the operations researcher. INFORMS J Comput 9:231–250
18. Reeves C (2003) Genetic algorithms. In: Glover F, Kochenberger GA (eds) Handbook of metaheuristics. Kluwer, Dordrecht, pp 55–82
19. Schaller J (2004) Single machine scheduling with early and quadratic tardy penalties. Comput Ind Eng 46:511–532
20. Sidney JB (1977) Optimal single machine scheduling with earliness and tardiness penalties. Oper Res 25:62–69
21. Valente JMS (2008) An exact approach for the single machine scheduling problem with linear early and quadratic tardy penalties. Asia Pac J Oper Res 25:169–186
22. Valente JMS (2008) Beam search heuristics for the single machine early/tardy scheduling problem with no machine idle time. Comput Ind Eng 55:663–675
23. Valente JMS (2009) Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs. Asia Pac J Oper Res 26:319–339
24. Valente JMS (2007) Heuristics for the single machine scheduling problem with early and quadratic tardy penalties. Eur J Ind Eng 1:431–448
25. Valente JMS, Goncalves JF (2009) A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. Comput Oper Res 36:2707–2715
26. Wagner BJ, Davis DJ, Kher H (2002) The production of several items in a single facility with linearly changing demand rates. Decis Sci 33:317–346