MedCrave
Step into the World of Research

International Robotics & Automation Journal

Research Article

Open Access

CrossMark

Invasive weed optimization algorithm for minimizing total weighted earliness and tardiness penalties on a single machine under aging effect

Abstract

In this paper, minimizing total weighted tardiness and earliness criteria on a single machine is considered. Job processing time is a linear function of its starting time and each job has a distinct due date. In this study an Invasive Weed Optimization (IWO) algorithm is proposed for machine scheduling problem. Because the local search operator in IWO algorithm is designed for continuous problem only, a simple and efficient coding and decoding technique is applied to map the discrete feasible space of the permutation to a number. Then, this number is used to produce the solution. The computational results show that the performance of the proposed algorithm is much better than the GA algorithm for this problem in finding the best solutions.

Keywords: invasive weed optimization, single machine, total weighted tardiness/ earliness, scheduling; step-deterioration

Volume 2 Issue 1 - 2017

Maziyar Yazdani,[1] Reza Ghodsi[2]

[1]Department of Industrial Engineering, University College of Engineering, University of Tehran, Iran
[2]Engineering Department, Central Connecticut State University, USA

Correspondence: Maziyar Yazdani, Department of Industrial Engineering, University College of Engineering, University of Tehran, Iran, Email Maziyar.yazdani@gmail.com

Received: October 16, 2016 | **Published:** January 6, 2017

Introduction

Minimizing the total weighted tardiness and earliness time on a single machine with aging effect is discussed in the research at hand and an efficient IWO algorithm is proposed for this problem. In the classical scheduling problem, processing times for different jobs are assumed to be constant within the planning horizon. However, previous researches reveal that in many scheduling environments the processing time is an increasing function of start time.[1] This phenomenon is known as aging effect which relates to many practical and industrial applications. One of the aging effect situations, which is also considered in this study, is the linear position-dependent aging effect where jobs processing time is a linear function of job's starting time.

In recent years, most policies of modern industrial organization emphasize the need for minimizing earliness and tardiness penalty. This is more important in just in time (JIT) productions. In a JIT production environment, early job must be held in inventory until their due dates and jobs completed after their due dates can cause customer dissatisfaction, contract penalties, loss of sale and loss of reputation.[2] Therefore, many researcher pay attention to this type of scheduling. This problem is categorized as an Np-hard problem.[3] Hamidinia etal. proposed a genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system.[4] A single machine scheduling problem with controllable processing times without inserted idle time to minimize total tardiness and earliness is presented by Kayvanfar et al.[5] Kaweegitbundit developed a memetic algorithm for minimizing earliness and tardiness penalties[6] Kedad Sidhoum and Sourd proposed a neighborhood search algorithm for the single machine earliness-tardiness which jobs have distinct due dates.[7] Allaoua and Osmane proposed a new genetic algorithm inspired by the philosophy of dynamic programming, where the chromosome and the population lengths are varied from one iteration to another for earliness and tardiness problem.[8] Among the recent research in aging effect readers are referred to researches presented by.[9–14] No prior research has used IWO for the problem at hand.

This paper is organized as follows. In the next section, the problem considered in this study is described in more details. The proposed algorithm is presented in section 3. For evaluation of the performance of the suggested algorithm, a series of computational experiments is performed and the results of this study are reported in section 4. Finally, section 5 concludes the paper with a short summary of the work.

Problem definition

The problem considered in this study can be formally described as follows: Assume that there are n independent jobs $\{j_1, j_2, \ldots\ldots j_n\}$ that are available for being processed at time zero on a single machine. The machine can only process one job at a time and processing of a job cannot be interrupted until it is entirely completed (preemption is not allowed). Each job j_i has a normal processing time p_j, a due date d_j and positive weights for per unit earliness and tardiness. In addition, let p_j and $p_{[j]}$ be the normal processing time and the actual processing time of the job scheduled in a sequence, respectively. If a job processed in rth position, actual processing time is:

$$p_{[j]} = p_j + b_j r \text{ for } j, r = 0, 1, \ldots, n \quad (1)$$

Where b_j is the aging ratio of job j?

In this problem, the objective function is minimizing the total weighted tardiness and earliness.

Proposed IWO algorithm

Main IWO algorithm

IWO is a continuous, stochastic numerical algorithm that mimics the colonizing behavior of weeds.[15] First, a population of initial seeds is randomly spread over the entire search space. These weeds will

finally grow up and execute the further steps of the algorithm. There are four steps of the algorithm as below:

Initialization: A finite number of weeds are initialized randomly in the feasible search space via a uniform function.

Reproduction: Every individual in the population is permitted to produce seeds based on its own fitness as well as the colony's lowest and highest fitness, such that the number of seeds produced by a weed increases linearly from lowest predefined possible seed for a weed with the worst fitness to the maximum number of seeds for a plant with the best fitness so far. S_{min} and S_{max} denote minimum and maximum number of weeds respectively and are predefined.

Spatial distribution: The generated seeds are being randomly scattered over the d-dimensional search space by normally distributed random numbers with the mean equal to location of parent plant and a varying variance. This step guarantees that the produced seeds will be generated around the parent weed. However, the Standard Deviation (SD) of the random function decreases over the iterations from initial value $sd_{initial}$ to final value sd_{final}. Standard deviation for a particular iteration is given as in equation 2 below:

$$sd_{iter} = \left(\frac{iter_{max} - iter}{iter_{max}}\right)^{n} \left(sd_{initial} - sd_{final}\right) + sd_{final} \qquad (2)$$

Where n is nonlinear modulation index. This step means that the probability of dropping a seed in the area around the parents decreases nonlinearly with iterations, which results in better plants and elimination of unsuitable plants. Hence, this is the selection mechanism of Invasive Weed Optimization Algorithm.

Competitive exclusion: When the maximum number of individual in a colony is reached (P_{max}), each weed is allowed to produce seeds and scatter them according to the mechanism indicated in the previous step. After that, parents and their new seeds are ranked together according to their fitness. Next, weeds with highest fitness are selected to reach the maximum allowable population size in a colony.

Termination condition: The entire process continues until the maximum number of iterations has been reached, and then the plant with the best fitness is the closest one to the optimal solution.

Because the IWO algorithm is designed for continues problem and the local search operator in this algorithm uses normal distribution to search the feasible space, this algorithm cannot be used directly for discrete problems without modification. Here a coding method is applied to map every permutation to a number and then use this number as the mean for the normal distribution to produce new seed (new solution). Then, using a decoding technique matches this new number to the permutation. This coding-decoding technique is described as follows.

Factoradic base

Factoradic is a specific constructed number system and a lexicographical index arrangement for permutations.[16] The concept of the factoradic is closely connected to that of the Lehmer code.[16] This numbering system is a factorial-based mixed radix numerical system: the ith digit from right side is to be multiplied by i! And the rightmost digit is always 0, the second digit is 0, or 1, the third 0, 1 or 2 and so on.[16] For instance, 44 in decimal base can be shown as $(1_4 3_3 1_2 0_1 0_0)$ in factoradic base.

$$(1_4 3_3 1_2 0_1 0_0) = 1\times 4! + 3\times 3! + 1\times 2! = 44_{10} \quad (3)$$

The factoradic numbering system never has two possible interpretations and no number can be described in more than one way due to the fact that the sum of consecutive factorials multiplied by their index is always the next factorial minus one.[16]

$$\sum_{i=0}^{n} i \times i! = (n+1)! - 1 \quad (4)$$

Relation between factoradic base and permutations

There is a natural mapping between the **permutations** of n elements in lexicographical order and the integers 0, 1,…, n!-1, when the integers are expressed in factoradic form. This mapping has been called the Lehmer code. For instance, with n=3, this mapping is illustrated in Table 1.

Table 1 Natural mapping between factoradic numbers and permutations when n=3.

Decimal	Factoradic	Permutation
0_{10}	$0_2 0_1 0_0$	1,2,3
1_{10}	$0_2 1_1 0_0$	1,3,2
2_{10}	$1_2 0_1 0_0$	2,1,3
3_{10}	$1_2 1_1 0_0$	2,3,1
4_{10}	$2_2 0_1 0_0$	3,1,2
5_{10}	$2_2 1_1 0_0$	3,2,1

For mapping permutations into factoradic numbers and vice versa, two algorithms are proposed in McCaffrey[17] and further used in Behroozi.[18] Computational complexity of both algorithms are O(n) which makes the algorithms able to efficiently map decimal numbers to factoradic numbers, factoradic numbers to the permutations and vice versa. These two algorithms are presented as Algorithm 1 and Algorithm 2 below.

Algorithm 1: mapping permutation to factoradic.

Step 1: Consider $f = \{0,1,\ldots,n-1\}$ as a list of possible digits for factoradic base in ascending order, and also a list of possible numbers in permutation $p = \{1,2,\ldots,n\}$

Step 2: For $k = 1, 2, \ldots, n$, select the kth digit in permutation, and find this digit in p. Suppose that this digit is the ith digit in $p = \{1,2,\ldots,n\}$. Choose the ith element of f, set this element as the kth element of factoradic, and omit this element from p. Table 2 demonstrates the mapping of $(2 - 3 - 1) \rightarrow (1_2 1_1 0_0)$ based on Algorithm 1.

Algorithm 2: mapping factoradic to permutation

Step 1: Like algorithm 1, consider a list of possible digits of factoradic base in ascending order $f = \{0,1,\ldots,n-1\}$, and also a list of possible

numbers in permutation $p = \{1,2,\ldots,n\}$

Step 2: For k=1,2,…,n: select the k_{th} digit in factoradic representation, and find this digit in $f = \{0,1,\ldots,n-1\}$. Assume that this digit is the i_{th} digit in $f = \{0,1,\ldots,n-1\}$. Select the i_{th} element of p; set this element as the k_{th} element of permutation, and omit this element from p. Table 3 demonstrates the mapping of $(1_2 1_1 0_0) \rightarrow (2-3-1)$ based on Algorithm 2.

Proposed IWO algorithm

Step 1: Set the values of the control parameters: N_{int} (number of initial seed), P_{max} (maximum number of plant can be survived in every iteration), $iter_{max}$ (maximum number of iterations), S_{max} (maximum number of seed that a plant can produced according to that's fitness), S_{min} (minimum number of seed that a plant can produced according to that's fitness), n (nonlinear modulation index), $sd_{initial}$ (initial value of standard deviation), and sd_{final} (final value of standard deviation).

Step 2: Generate initial solutions (initial seeds' population) to initiate solution space exploration. For single machine problem with n jobs, any permutation of n number can be a feasible initial solution for problem.

Step 3: This step is the fitness evaluation of each seed. Calculate the values of the fitness functions for the solutions according to objective function. The seed is called a plant after assigning each fitness value to the corresponding seed.

Step 4: Rank the plants and plants reproduce new seeds. Plants are ranked according to the corresponding fitness values. Plants generate seeds based on the rank in the colony. The number of new seeds to be produced is computed for each plant using the linear relationship considering S_{min} and S_{max}.

Step 5: Spread the newly produced seeds. The produced seeds are spread over the search area based on the standard deviation of each step. Because this algorithm uses normal distribution to spread newly produced seed, factoridc coding is applied to map jobs permutation to a number and by using that number as the mean of normal distribution the new seed is spread.

Step 6: Compute the fitness values of the new plants.

Step 7: Is the number of the plants in the current step less than maximum number of plant?

Yes: Go to step 9.

No: Then do competitive exclusion. In proposed algorithm, k% of the plants with better fitness values is selected and other population of next iteration is selected randomly through remaining plants.

Step 8: Run the intensification procedure. Intensification is the search in the neighborhood of the current solutions for constructing better solutions closer to optimum solution. This is done by switching the positions of the first two jobs and then, the objective function value of this new permutation is calculated. If the objective function of the new switched permutation is better, the algorithm records this switch and its objective function value. Next, two jobs will move back to their original position before switching. Next, the position of the first job and the third job are switched. This procedure continues and is applied for the first job and all other jobs and finally the switch with the most improvement in objective function value is accepted. The same procedure is applied for job two in the new position and all jobs after second position. Thus, these steps are repeated for job in position one to job in position n-1.

In this proposed IWO algorithm, the described procedures is applied to q% of the population in every iteration randomly.

Step 9: Is the iteration number less than maximum number of iteration?

Yes: Repeat the algorithm again.

No: Stop. Calculate the objective function values of the current plants and choose the best among them.

Table 2 Mapping of (2 – 3 – 1) → (121100) based on Algorithm 1

Iteration	k=1	k=2	k=3
Permutation	2	3	1
P	{1,2,3}	{1,3}	{1}
F	{0,1,2}	{0,1,2}	{0,1,2}
Factoradic	1	1	0

Table 3 Mapping of (121100) → (2 – 3 – 1) based on Algorithm 2

Iteration	k=1	k=2	k=3
Factoradic	1	1	0
F	{0,1,2}	{0,1,2}	{0,1,2}
P	{1,2,3}	{1,3}	{1}
Permutation	2	3	1

Table 4 Name of problems generated with variable parameters obtained are shown in Table 5

Computational results

In this section, the performance of the proposed IWO is presented. Problems with different sizes are considered. The small size problems are associated with 10 jobs, medium size with 15 and 20 jobs, and large size with 40 and 60 jobs. The processing times are generated from the discrete uniform distribution [40,120] and weight of earliness–tardiness penalties are drawn from the discrete uniform distribution [1, 4]. The aging ratio of each job is drawn from the uniform distribution [0, 2] and the due dates of each job is drawn from the uniform distribution $[dmin-\lambda, dmin+\lambda]$, Where

$$dmin = P(1-TEF), \mathrm{P} = \left(\sum_{i=1}^{n} p_i\right) + \left(\frac{n}{2} \times \sum_{i=1}^{n} b_i\right) \text{ and } \lambda = P(RDD/2).$$

The two parameters RDD and TEF are the relative range of due dates and tardiness/earliness function, respectively. RDD gets the values of 0.2 and 0.5. Also TEF gets the values of 0.2 and 0.35. Table 4 shows the generated problems.

The following experimentally derived values are proposed for the parameters:

$$Nint = n, Pmax = n, itermax = 5 \times n, Smax = 3, Smin = 1, n = 2, sdinitial = n^2, sdfinal = 2, k = 10\%$$ and $q = 5\%$. And n is the number of jobs.

The proposed algorithm is evaluated against the Genetic Algorithm (GA), GA parameters are as below:

Number of initial population=50, crossover operator: uniform (80%), mutation operator: Inversion (0.02), selection operator: roulette wheel. stop condition: 10n.

For each instance both algorithms run five times independently and best and worst solutions obtained are shown in Table 5.

In addition, to compare the two methods, Relative Percentage Deviation (RPD) is used, which is computed in the following way:

$$PRD = \frac{alg_{sol} - min_{sol}}{min_{sol}} \quad (5)$$

Where Alg_{sol} as the objective function value is obtained for a given algorithm and $Minsol$ is the best solution obtained for each instance by any of the two algorithms. Figure 1 compares the mean of PRD for the two algorithms. The computational results show that the proposed IWO has less deviation from the best solution and that the algorithm is more promising in finding near-optimum solutions.

Table 5 Compare best, mean and worst solutions that obtained for algorithms

Problem name	Number of jobs	GA			IWO		
		best	average	worst	best	average	worst
J101	10	4916	4916	4916	4916	4916	4916
J102	10	5515	5515	5515	5515	5515	5515
J151	15	11081	11104.6	11127	11081	11081	11081
J152	15	17169	17190.6	17227	17169	17169.8	17171
J201	20	25901	25949.2	25981	25901	25901	25901
J202	20	31598	31621.8	31647	31581	31581	31581
J401	40	97783	97927.2	98026	97135	97139.8	97159
J402	40	128239	128322	128461	127989	127990.8	127995
J601	60	249017	249757.8	250785	247707	247987	248057
J602	60	202926	203481	204003	201689	201702.2	201711

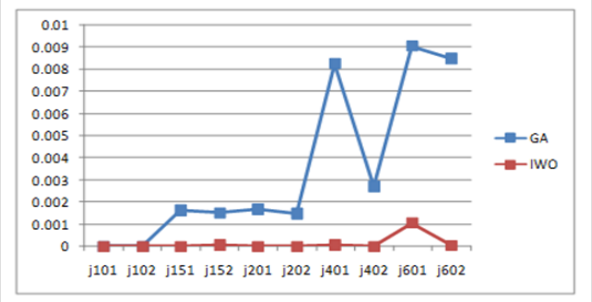

Figure 1 mean of PRD for two algorithms in 5 run.

Conclusion

This research considered the single machine scheduling problem where the processing time of jobs depends on the position of the job scheduled. This problem is NP-hard and finding optimum solution for the large instances of this problem is not possible in a reasonable time. An Invasive Weed Optimization Algorithm was proposed to find near-optimal solutions for this problem. The proposed algorithm employs a coding procedure to map the permutation of jobs into a continuous space where Invasive Weed Optimization Algorithm can efficiently spread the new produced seed sand perform exploration. Afterward, the algorithm maps the new seed to a permutation. Moreover, a powerful intensification sub-algorithm was applied to search the promising areas of the feasible space more comprehensively. Performance of the proposed algorithm was tested using several generated problems and compared to GA algorithm. The computational results prove that the proposed algorithm is more promising than the GA algorithm used.

Acknowledgements

None.

Conflict of interest

The author declares no conflict of interest.

References

1. Pinedo M. *Scheduling: Theory, Algorithms and Systems*. Springer; 1995.
2. Liao CJ, Cheng CC. A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Computers and Industrial Engineering*. 2007;52(4):404–413.
3. Liao CJ, Choi JY. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers and Operations Research*. 1995;22(8):857–869.
4. Hamidinia A. A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Computers and Industrial Engineering*. 2012;62(1):29–38.
5. Kayvanfar V I, Mahdavi, Komaki GM. Single machine scheduling with controllable processing times to minimize total tardiness and earliness. *Computers and Industrial Engineering*. 2011;65(1):166–175.

6. Kaweegitbundit P. A Single machine scheduling problem with earliness and tardiness penalties using memetic algorithm. *Advanced Materials Research*. 2011;314:2353–2357.

7. Kedad-Sidhoum S, Sourd F. Fast neighborhood search for the single machine earliness-tardiness scheduling problem. *Computers and Operations Research*. 2010;37(8):1464–1471.

8. Allaoua H, Osmane I. Variable parameters lengths genetic algorithm for minimizing earliness-tardiness penalties of single machine scheduling with a common due date. *Electronic Notes in Discrete Mathematics*. 2010;36(C):471–478.

9. Fan J, Xu D. *Some single-machine scheduling problems with aging effect*. IEEE Explore; 2010.

10. Janiak A, Rudek R. Scheduling jobs under an aging effect. *Journal of the Operational Research Society*. 2010;61(6):1041–1048.

11. Ji M, Hsu CJ, Yang DL. Single-machine scheduling with deteriorating jobs and aging effects under an optional maintenance activity consideration. *Journal of Combinatorial Optimization*. 2011;26(3):437–447.

12. Rudek R. Some single-machine scheduling problems with the extended sum-of-processing-time-based aging effect. *International Journal of Advanced Manufacturing Technology*. 2012;59(1-4):299–309.

13. Rudek R. The strong NP-hardness of the maximum lateness minimization scheduling problem with the processing-time based aging effect. *Applied Mathematics and Computation*. 2012;218(11):6498–6510.

14. Yang SJ, Yang DL. Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. *Omega*. 2010;38(6):528–533.

15. Mehrabian AR Lucas C. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*. 2006;1(4):355–366.

16. Knuth DE. Semi-numerical algorithms. 3rd ed. *The art of computer programming*. MA, USA: Addison-Wesley Longman Publishing Co Boston; 1973.

17. McCaffrey J. *Using permutations*. NET for improved systems security; 2003.

18. Behroozi M. *A meta-heuristic approach for a special class of job shop scheduling problem*. Faculty of industrial engineering. Tehran, Iran: Sharif University of Technology; 2009.