

Integrated Due Date Setting and Scheduling on a Single Machine Considering an Unexpected Unavailability

Mehdi Iranpoor ^a, Seyed Mohammad Taghi Fatemi Ghomi ^{b,*}

^a Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan, Iran

^b Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran

Received 15 July 2016; Revised 18 June 2017; Accepted 11 November 2017

Abstract

In this paper, an integrated machine scheduling with its due date setting problem has been considered. It is assumed that the machine is subject to some kind of random unavailability. Due dates should be set in an attractive and reliable manner, implying that they should be short and possible to be met. To this end, first, long due dates are penalized in the objective function. Then, for each customer order, the probability of meeting his/her promised due date is forced to be at least as large as his/her required service level. To handle this integrated problem, first, the optimal due date formula for any arbitrary sequence is derived. By using this formula, the mathematical programming formulation of the problem, including a nonlinear non-convex expression, is developed. By defining a piecewise linear under-estimator, the solutions of the resultant mixed integer linear programming formulation have become the lower bounds of the problem. Dynasearch is a very efficient heuristic utilizing the dynamic programming approach to search exponential neighborhoods in the polynomial time. An iterated dynasearch heuristic is developed for the sequencing part of the problem. Each generated sequence is evaluated by computing its optimal due dates using the above-mentioned formula. Numerical results confirmed the high quality of the solutions found by this algorithm, as compared with the lower bound.

Keywords: Due date setting; Unexpected unavailability; Machine scheduling; Iterated dynasearch; Lower bound.

1. Introduction

This paper studies the integrated machine scheduling and due date setting problem by considering the random machine unavailability. If the quoted due dates are not competitively short, customers may opt to buy from faster competitors. So, the sales and revenues should be diminished. On the other hand, if the due dates are set at unachievable values, the promised due dates are more likely to be violated; therefore, the company's good-will is damaged. So, the objective is defined as the minimization of long due date effects while respecting the required service levels of jobs. We refer to this problem as safe scheduling with random unavailability (*SSRU*).

There is some important trade-off in due date setting decisions between attractiveness and credibility (Slotnick, 2014). Setting short due dates reflects better responsiveness to customer orders, attracting new customers and encouraging the current customers to retain. However, the tardiness is more likely to happen (Slotnick and Sobel, 2005). Tardiness, in turn, degrades the reputation of the manufacturing company and brings about contractual penalties (Slotnick and Sobel, 2005). Conversely, long due dates may not be acceptable by customers, resulting in price discount offers by the company to maintain its business (Shabtay, 2016). On the other hand, long due dates facilitate production planning and are more likely to be attained (Yin et al., 2015). Due dates cannot be determined precisely when the sequence of jobs is not known. This is why due date setting and

scheduling decisions are usually studied as an integrated problem.

Typically, each customer considers a maximum length of lead time to be acceptable (Shabtay, 2010). If the quoted due dates do not exceed the acceptable lead time, there will be no lead time penalty. However, the due dates later than these limits are subject to long due date penalties. Shabtay and Steiner studied two integrated machine scheduling and due date setting problems by considering the acceptable lead time and due date assignment cost (Shabtay and Steiner, 2006). Their problems were the minimization of total weighted earliness/tardiness and the total weighted number of tardy jobs. They proved that both problems were NP-hard and developed polynomial approaches for some special cases. Shabtay addressed some integrated machine scheduling and due date setting regarding earliness, tardiness, holding, batch delivery, and long due date penalties (Shabtay, 2010). He also assumed that there is no due date cost when a due date is assigned not greater than the acceptable lead time. He proved that the problem was NP-hard and developed polynomial algorithms for the batch partitioning part and two special cases of the problem. Gerstl and Mosheiov considered an acceptable common due window and penalized both early and late due dates in a machine scheduling setting (Gerstl and Mosheiov, 2013). They assumed that early due dates were unfavorable because customers needed some preparation time before receiving their products. They developed a polynomial approach that minimized the maximum due date and job costs. Mor et al. (Mor et al.

*Corresponding author Email address: fatemi@aut.ac.ir

2013) addressed a machine scheduling problem with the job-dependent acceptable lead time. They developed a polynomial algorithm which minimized maximum earliness, tardiness, and due date costs. Yet et al. (Yin et al., 2016), on the other hand, addressed some due date setting and machine scheduling problems in which jobs were owned by two agents. They assumed that the due dates of one agent were given, whereas the due dates of the other agents were decision variables. The objective was the minimization of the total due date assignment and tardiness costs of the latter agent while keeping the objective value of the former under a predetermined threshold. They developed some polynomial and pseudo-polynomial dynamic programming algorithms for these problems. Shabtay (Shabtay 2016) studied some due date setting and scheduling problems with upper bound constraints on the assigned due dates; therefore, the objective was the minimization of total earliness, tardiness, and long due dates penalties. He proved that all optimal solutions excluded earliness. He analyzed the problems in various parameter settings. *NP-hardness* was proved in some cases, and polynomial-time algorithms were developed for some other ones.

Due date setting problem becomes more challenging in the presence of uncertainty. In this situation, a popular approach is trying to quote tight due dates while respecting some required service level constraints (Keskinocak and Tayur 2004). The service level of each job is defined as the probability that the processing of the job is completed by its quoted due date (Elyasi and Salmasi 2013). One source of uncertainty is the randomness of processing times. Elyasi and Salmasi, for example, studied two due date assignment and scheduling problems by considering service level constraints and random processing times (Elyasi and Salmasi, 2013). They proved that sequencing jobs in a decreasing service level order minimized the maximum assigned due date. They also studied the properties of solutions with the minimum total expected earliness, tardiness, and due date costs. Baker and Trietsch studied several integrated due date settings and machine scheduling problems with regard to stochastic processing times (Baker and Trietsch, 2009). Assuming stochastically-ordered processing time distributions, they proved that sequencing the jobs in the non-decreasing order of expected processing times minimized some objective functions including the due date cost. They also proposed some static and dynamic heuristics when this assumption was not held. Baker, on the other hand, dealt with the minimization of the sum of the assigned due dates in a single machine scheduling problem with normally distributed processing times (Baker, 2014). He assumed that the service level constraints had to be met. He developed a branch and bound approach and some efficient simple dispatching rules. Baker and Trietsch developed a powerful branch and bound algorithm to minimize the sum of the total due date and expected tardiness costs in a machine scheduling environment with normally distributed processing times (Baker and Trietsch, 2015). They derived the optimal service level in any given sequence of jobs.

The other significant source of uncertainty is machine availability. The unavailability can be a result of random breakdown or arrival of an emergent job which should be processed immediately (Kacem et al. 2014). In both cases, the start and length of the unavailability interval are not known in advance (Huo et al., 2014). To the best of our knowledge, due date setting in the presence of random unavailability has not been studied in the literature yet. It must be noted that Kacem et al. studied a machine scheduling problem with regard to an unexpected unavailability for the minimization of the maximum lateness (Kacem et al., 2014). They addressed two sources of unavailability: breakdown and entrance of an emergent job. The difference in these cases was that the arriving emergent job was considered when calculating the maximum lateness. They analyzed the approximability of these problems, proving that Jackson rule was a 2-approximation algorithm for both cases. Huo et al. also considered the two similar cases, but with the objective of the minimization of the total weighted completion time in which the weight of each job was proportional to its processing time (Huo et al. 2014). They showed that the simple a LPT rule and its variation had competitive approximation ratios for the situations with known unavailability and unknown unavailability information, respectively. Yin et al. (2017) considered parallel machine scheduling problems with possible disruptions and deteriorating processing times. They assumed that some of the machines might be disrupted at known times, but the duration of disruption was stochastic. They analyzed two cases of immediate maintenance after disruption, which returned the machine to its original efficient state, and not performing maintenance. They developed some pseudo-polynomial time approximation algorithms and polynomial-time approximation schemes for these problems. Agnetis et al. (Agnetis et al., 2017) studied a parallel machine scheduling problem under unrecoverable interruptions. They assumed that if such an interruption (failure) happened on one machine, all jobs scheduled on that machine and not processed yet could not be completed or moved to another machine. Their objective was finding the sequence maximizing the expected number of the completed jobs on all machines. They proposed a pseudo-polynomial time exact algorithm and an efficient list scheduling heuristic. Kacem and Kellerer (Kacem and Kellerer, 2016) addressed three semi-online single machine scheduling problems under a single unexpected breakdown. The first two problems were the minimization of makespan with/without jobs' release times; while the third problem was the minimization of the maximum lateness without considering the release times. They provided three approximation algorithms with tight solutions, as compared with the optimal offline solutions with known start time and duration of breakdown.

The contributions of the paper are as follows:

(1) To the best of our knowledge, the novel problem of due date setting, considering stochastic machine unavailability, either with or without the required service level constraints, is studied for the first time.

- (2) The optimal due date formula is derived for any arbitrary sequence.
- (3) A dynamic programming-based heuristic, called dynasearch, is proposed to find efficient solutions.
- (4) A tight lower bound is developed for the evaluation of the heuristic solutions.

The paper is organized as follows. In Section 2, the problem is formally defined and the notations are introduced. Section 3 derives the optimal due dates in any arbitrary given sequence. Further, in this section, it is proved that the problem is NP-hard in the strong sense. By using the optimal due date formula, the mathematical programming formulation of the problem is developed in Section 4. Section 5 describes the iterated dynasearch algorithm used to solve the problem. In order to evaluate the quality of this heuristic algorithm, a lower bound is developed in Section 6. The performance of the proposed algorithm and the tightness of the lower bound are evaluated numerically in Section 7. Finally, Section 8 summarizes the results.

2. Problem Definition and Notations

This paper studies an integrated due date setting and machine scheduling problem. Each job has an acceptable lead time. Setting due dates later than these acceptable lead times are penalized. The objective is the minimization of total long due date penalties. The machine is subject to an unexpected unavailability. This uncertainty does not allow for the quotation of 100% guaranteed due dates. In other words, breakdown may cause tardiness. However, due dates should be reliable. Hence, due dates not satisfying the required service level constraints are not allowed. As mentioned above, service level is the probability that the processing of the job is completed by its quoted due date. Each job has a specific required service level. It is assumed that the length of the time interval before the start of unavailability and the length of unavailability follow exponential distributions. All jobs are available at time zero. Further, it is assumed that preemption is allowed. In other words, the processing of the interrupted job is resumed as soon as the machine is repaired.

The parameters and decision variables are as follows.

Parameters	Decision variables
n : the number of jobs	$[j]$: the index of job scheduled as the j th job
T_b : the stochastic variable of time interval before the start of unavailability	C_j : the stochastic variable of the completion time of job j
β_b : the average time before the start of unavailability	d_j : the due date of job j
T_r : the stochastic variable of the duration of unavailability	x_{ij} : the binary variable which is 1 whenever job i precedes job j in the sequence or $i=j$
β_r : the average time of the duration of unavailability	y_j : the completion time of the j th job in the sequence when unavailability does not occur
a_j : the acceptable lead time of job j	
p_j : the processing time of job j	
P : the total processing times of all jobs	
\bar{p} : the average processing time of all jobs	
w_j : the weight of job j	
sl_j : the required service level of job j	

As mentioned above, the objective is the minimization of the total long due date penalties:

$$z = \sum_{j=1}^n w_j (d_j - a_j)^+ \quad (1)$$

$$C_{[j]} = \begin{cases} \sum_{k=1}^j P_{[k]} & ; \quad \text{with probability } e^{-\frac{\sum_{k=1}^j P_{[k]}}{\beta_b}} \\ \sum_{k=1}^j P_{[k]} + T_r & ; \quad \text{with probability } 1 - e^{-\frac{\sum_{k=1}^j P_{[k]}}{\beta_b}} \end{cases} \quad (2)$$

The first term in (2) is the summation of the processing times of jobs from the beginning of the sequence to the j th job. This situation happens when the unavailability does

wherein $(X)^+$ refers to the maximum of zero and X .

In an arbitrary sequence of jobs, the completion time of the j th job in the sequence is computed through expression (2).

not occur before completing the j th job. The second term is the complementary situation. In this case, the

completion time of the j th job includes the stochastic variable of the repair time (T_r).

Each customer has a required service level, which must be respected. The service level constraint of job j is $\Pr(C_j \leq d_j) \geq sl_j$, which means that the probability of violating each due date should be less than or equal to the corresponding required service level. When this inequality holds, job j is called stochastically on time; else, it is called stochastically tardy (Baker, 2014). In order to minimize the due date costs, d_j s should have the minimum possible values. Therefore, equation (3) meets the required service levels and minimizes the long due date effects.

$$\Pr(C_j \leq d_j) = sl_j; \quad j = 1, \dots, n \quad (3)$$

3. Finding the Optimal Solutions

The problem under study (*SSRU*) has two parts: determining the due dates and sequencing the jobs. In this section, first, the optimal due dates are derived for the known sequences of jobs. Then, it is proved that the problem is NP-hard in the strong sense.

3.1 Optimal due dates in the known sequences

In this subsection, it is proved that in any given sequence of jobs, the optimal due dates can be calculated by a closed form expression. This result is used to develop the solution approach and the lower bound in the subsequent sections. To derive the optimal due dates, two complementary cases of the required service levels are handled in the following lemmas. As shown in Lemma 1, in the first case, the unexpected unavailability can be omitted.

Lemma 1. In an arbitrary sequence, for each job $[j]$ with

$$d_{[j]}^* = \sum_{k=1}^j p_{[k]} + \left(\beta_r \ln \left(\left(1 - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}} \right) / (1 - sl_{[j]}) \right) \right)^+ \quad (7)$$

In case condition (4) is held, the natural logarithm in (7) is negative; so, $d_{[j]}$ equals the resultant of lemma 1. On the other hand, in case where condition (5) is held, the logarithm is positive and the above expression equals the resultant of lemma 2.

3.2 Computational complexity

In this section, it is proved that *SSRU* is NP-hard in the strong sense. The proof is proceeded by showing that the problem contains the single machine total weighted tardiness problem as a special case. Hence, according to the widely conjectured assumption $P \neq NP$, developing an algorithm to find the optimal solution in the polynomial or pseudo-polynomial time seems impossible. Therefore, in Section 5, an efficient heuristic is developed to find near optimal solutions in reasonable times.

Theorem 2. *SSRU* is strongly NP-hard.

$$sl_{[j]} \leq e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}} \quad (4)$$

the optimal due date is $\sum_{k=1}^j p_{[k]}$.

Proof.

In this case, the probability that the unavailability starts after the completion of job $[j]$ is larger than the required service level. Thus, with neglecting the unexpected unavailability, the required service level is satisfied, and the due date costs are minimized.

The second lemma presents a closed-form expression of the optimal due dates for the complementary case. In this case, the probability of the occurrence of random breakdown affects the optimal due date of job $[j]$; hence,

$$d_{[j]}^* > \sum_{k=1}^j p_{[k]}.$$

Lemma 2. In an arbitrary sequence, for each job $[j]$ with

$$sl_{[j]} > e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}} \quad (5)$$

the optimal due date is

$$d_{[j]} = \sum_{k=1}^j p_{[k]} + \beta_r \ln \left(\left(1 - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}} \right) / (1 - sl_{[j]}) \right) \quad (6)$$

Proof.

See appendix A.

The following theorem is a straightforward combination of the above lemmas.

Theorem 1. In an arbitrary sequence, the optimal due date of job $[j]$ is

Proof.

It should be proved that there is a strongly NP-hard problem, which is reducible to *SSRU*. Single machine total weighted tardiness problem, denoted by $1||\sum wT$, is strongly NP-hard (Pinedo, 2012). Here, based on the following transformation, it is shown that $1||\sum wT$ is reducible to *SSRU*.

$$sl_j = 0; \quad j=1, \dots, n$$

According to Lemma 1, in any schedule, the optimal due dates would be:

$$d_{[j]} = \sum_{k=1}^j p_{[k]}$$

Hence, the problem can be regarded as a $1||\sum wT$ in which the due date of job j is a_j and the weight of job j is w_j .

4. Mathematical Programming Formulation

In order to formulate the problem, we use the optimal due date formula. According to the proof of Theorem 1, expression (7) can be rewritten as follows:

$$d^*_{[j]} = \max \left\{ \sum_{k=1}^j p_{[k]}, \sum_{k=1}^j p_{[k]} + \left(\beta_r \ln \left(\frac{1 - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}}}{1 - sl_{[j]}} \right) \right) \right\} \quad (8)$$

Therefore, the mathematical formulation of the problem is as follows:

$$\text{Min } \sum_{j=1}^n w_j z_j \quad (9)$$

$$\sum_{i=1}^n p_i x_{ij} - y_j = 0; j = 1, \dots, n \quad (10)$$

$$z_j \geq y_j - a_j; j = 1, \dots, n \quad (11)$$

$$z_j \geq y_j + \beta_r \ln \left(\frac{1 - e^{-\frac{y_j}{\beta_b}}}{1 - sl_j} \right) - a_j; j = 1, \dots, n \quad (12)$$

$$u_i - u_j + nx_{ij} \leq n - 1; i = 1, \dots, n, j = 1, \dots, n, j \neq i \quad (13)$$

$$x_{jj} = 1; j = 1, \dots, n \quad (14)$$

$$u_j, y_j, z_j \geq 0; j = 1, \dots, n \quad (15)$$

$$x_{ij} = 0, 1; i = 1, \dots, n; j = 1, \dots, n \quad (16)$$


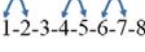
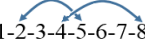
Expression (9) minimizes total long due date effects. Constraints (10-12) calculate z_j s. Constraint (10) sets y_j equal to the total processing times of jobs [1] to [j]. On the other hand, constraints (11) and (12) force that $z_j \geq d^*_j - a_j$. Also, the non-negativity of z_j and minimization of $\sum w_j z_j$ guarantee that z_j equals $(d^*_j - a_j)^+$, that is, the amount by which the optimal due date of job j violates its acceptable lead time. Constraint (13) is similar to the subtour elimination constraint in *MTZ formulation* of the traveling salesman problem (Öncan et al. 2009). However, in *MTZ formulation*, $x_{ij} = 1$ if and only if job j precedes job i immediately. Nevertheless, the proof that constraints (13-16) discard all subtours and exclude no feasible sequence is similar to that of *MTZ formulation*. So, for the sake of brevity, it is not repeated here, and the interested readers are referred to (Miller et al. 1960).

5. Iterated Dynasearch Algorithm

As shown in Subsection 3.2, there are similarities between the problem under study and the total weighted tardiness problem. Dynasearch is a heuristic that uses a dynamic programming approach in order to search exponential size neighborhoods in a polynomial time. According to the literature, this algorithm has been used several times for machine scheduling problems with total tardiness and total weighted tardiness objectives. For instance, Angel and Bampis (2005), Congram et al.

(2002), Kedad-Sidhoum and Sourd (2010), Grosso et al. (2004), and Sourd (2006) used dynasearch for solving machine scheduling problems in which the objective function includes sum or total weighted tardiness. In another study, Ding et al. combined dynasearch with an adaptive perturbation strategy to solve the total weighted tardiness problem on a single machine (Ding et al. 2016). The efficiency of dynasearch in these studies promised successful application to the problem under study. Congram et al. introduced dynasearch heuristic as a powerful local search algorithm capable of searching the exponential size neighborhood in the polynomial time (Congram et al., 2002). Typically, while the neighborhood gets larger, the quality of solutions found by the traditional local search approaches is decreased. As a powerful local search algorithm, dynasearch mitigates this deficiency using a dynamic programming approach (Dumitrescu and Stützle, 2010). By employing dynamic programming, dynasearch conducts a series of moves in each iteration. This feature makes dynasearch superior to traditional local search algorithms in which a single move is made in each iteration. Neighborhood in dynasearch is built by a combination of independent move operations (Kedad-Sidhoum and Sourd, 2010). Two move operations are called independent when they work on separate parts of the sequence. This concept is exemplified in Table 1.

Table 1
Examples for the description of independent moves

Original sequence	Move		Independent?
1-2-3-4-5-6-7-8	swap jobs 1 and 5 swap jobs 7 and 8		YES
1-2-3-4-5-6-7-8	swap jobs 1 and 2 swap jobs 4 and 5 swap jobs 6 and 7		YES
1-2-3-4-5-6-7-8	swap jobs 2 and 5 swap jobs 4 and 8		NO

The dynasearch neighborhood comprises all sequences that can be derived from the current solution by a combination of pairwise independent moves. The dynamic programming used in dynasearch checks all these neighbor solutions and returns the best one. The detailed pseudo-code of this dynamic programming for *SSRU* is presented in appendix B.

$$MAU_j = \frac{w_j}{p_j} \text{Exp} \left[- \left(a_j - t - p_j - \left(\beta_r \ln \left(\frac{1 - e^{-\frac{t+p_j}{\beta_r}}}{1 - sl_j} \right) \right)^+ \right) / (k\bar{p}) \right] \quad (17)$$

This expression is a modified version of the apparent urgency used in (Congram et al. 2002) for the single-machine total weighted tardiness scheduling problem. In this expression, k is the parameter of the dispatching rule. We call this dispatching rule the *modified apparent urgency*.

Figure 1 depicts the proposed iterated dynasearch for *SSRU*. The algorithm starts by a sequence generated by the *modified apparent urgency* rule. Then, the dynamic programming algorithm is run within a loop until the stopping condition is met. In each iteration of the loop, the dynamic programming algorithm is recalled from the solution found in the previous step. If the objective

Dynasearch starts with an initial sequence as the input. In this paper, expression (17) is proposed to construct the initial solution. To this end, in each step, the non-sequenced job with the highest MAU_j is sequenced as the next job.

functions of two calls are the same, the found solution is a local optimum. In such a case, the search is repeated based on a perturbed solution generated from the previous local optimal solution or the best found solution until now (Congram et al., 2002). The perturbation is conducted by applying a random number of non-necessarily independent moves on the selected solution. As mentioned in (Congram et al., 2002), the dominance of dynasearch over traditional local search algorithms becomes more prominent when the search is repeated from the previous near local optimal solutions. The algorithm is stopped when the number of loops in which local optima happen becomes a predefined number.

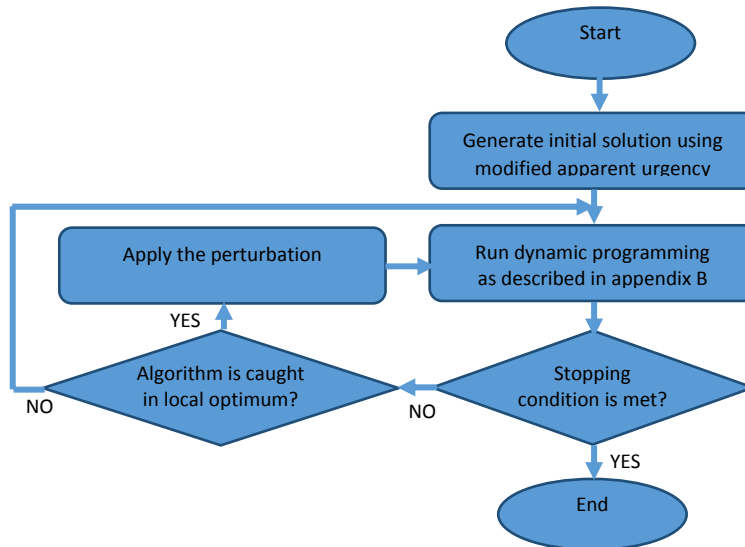


Fig. 1. Flowchart of iterated dynasearch algorithm for *SSRU*

6. Lower Bound Solution

In order to evaluate the quality of the solutions of the dynasearch algorithm, either the optimal or lower bound should be available. Problem (9-16) is a non-convex mixed integer nonlinear programming one. Therefore, general optimization packages cannot guarantee finding global optimum solutions even for small-sized problems. Further, as shown in subsection 3.2, the problem is strongly NP-hard. Thus, finding the optimal solution to this problem in a reasonable time seems impossible. A

$$f_j(y_j) = y_j + \beta_r \ln \left(\frac{1 - e^{-\frac{y_j}{\beta_b}}}{1 - sl_j} \right); \quad j = 1, \dots, n \tag{18}$$

This is the first and second parts of the right-hand side of Constraint (12). By derivation, it can be easily proved that $f_j(y_j)$ is strictly increasing and concave with respect to y_j . If it is replaced with a piecewise linear under-estimator in problem (9-16), the solution space is expanded. Hence, solving the resultant problem finds a lower bound for the optimal solution. The following approach for building the

lower bound is, therefore, developed in this section to measure the quality of the heuristic solutions.

The lower bound is developed through replacing the nonlinear part of constraint (12) by a piecewise under-estimator. This replacement results in a mixed integer-programming problem with additional continuous and binary variables. Thanks to the modern codes for solving pure and mixed integer programming problems, piecewise linear approximations, even with extra continuous and integer variables, are more manageable than the original nonlinear problems (Bergamini et al. 2008). Let $f_j(y_j)$ be defined as

piecewise linear under-estimator lower bound is called delta method (Bergamini et al. 2008).

Let y_j^a be the value of y_j , for which $f_j(y_j)$ equals a_j . y_j^a can be found numerically by simple arithmetic algorithms. Further, assume that $\Psi_j = \max\{p_j, y_j^a\}$. Consider a set of K grid points g_{kj} , $k=1, \dots, K$, which divides the interval $[\Psi_j, P]$ into $K-1$ equal subintervals. Figure 2 shows an example with $K=4$.

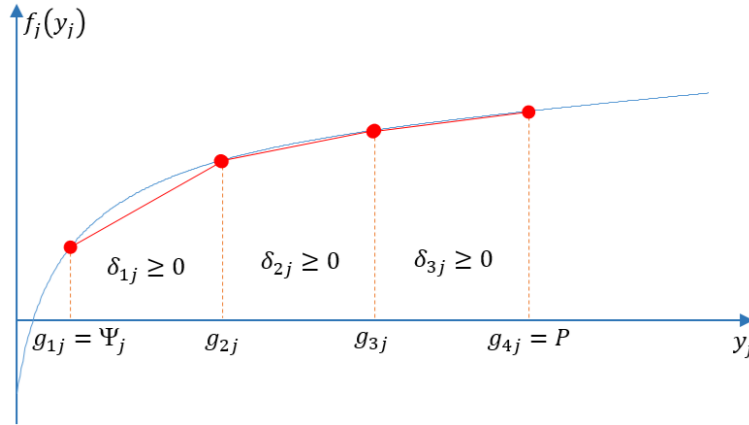


Fig. 2. Division of $[\Psi_j, P]$ to equidistant intervals

Theorem 3 shows the modification in problem (9-16), resulting in replacing the non-convex curve $f_j(y_j)$ with a piecewise linear under-estimator. This replacement expands the feasible solution space and, hence, generates the lower bounds for the optimal solutions.

Theorem 3. By replacing $f_j(y_j)$ with expression (19) and adding Constraints (21-25) in problem (9-16), the optimal solution to the resultant mixed integer linear programming problem gives a lower bound for the optimal solution of SSRU.

$$f_j(\psi_j) + \sum_{k=1}^{K-1} \left(\frac{(K-1)(f_j(g_{k+1,j}) - f_j(g_{kj}))}{P - \psi_j} \delta_{kj} \right) \tag{19}$$

$$y_j \leq \psi_j + \sum_{k=1}^{K-1} \delta_{kj} \tag{20}$$

$$\frac{P - \psi_j}{K-1} v_{1j} \leq \delta_{1j} \leq \frac{P - \psi_j}{K-1}; \quad j = 1, \dots, n \tag{21}$$

$$\frac{P - \psi_j}{K-1} v_{kj} \leq \delta_{kj} \leq \frac{P - \psi_j}{K-1} v_{k-1,j}; \quad j = 1, \dots, n, \quad k = 2, \dots, K-1 \tag{22}$$

$$\delta_{k-1,j} \leq \frac{P-\psi_j}{K-1} v_{k-2,j} ; j = 1, \dots, n \quad (23)$$

$$\delta_{kj} \geq 0 ; j = 1, \dots, n, \quad k = 1, \dots, K-1 \quad (24)$$

$$v_{kj} \in \{0,1\} ; j = 1, \dots, n, \quad k = 1, \dots, K-2 \quad (25)$$

Proof.

The following fraction is the slope of the piecewise linear under-estimator of $f_j(y_j)$ between g_{kj} and $g_{k+1,j}$.

$$(K-1)(f_j(g_{k+1,j}) - f_j(g_{kj})) / (P - \psi_j)$$

So, if δ_{kj} equals $\min\{y_j - g_{kj}, g_{k+1,j} - g_{kj}\}$, expression (19) truly calculates the under-estimator. Whenever $\delta_{k-1,j}$ is less than $(P - \psi_j)/(K-1)$, binary variables v_{kj} in constraints (21-23) force δ_{kj} to be zero. This shows that $f_j(y_j)$ is underestimated truly on the piecewise linear under-estimator.

Note that, as shown in Figure 2, by increasing the value of K , the under-estimator is tightened. However, this improvement is done under the cost of adding more additional binary and continuous variables into the model.

7. Numerical analysis

In this section, numerical results are reported. The aim of this section is two-fold. First, the performance of the proposed dynasearch algorithm is evaluated. Second, the tightness of the provided piece-linear lower bound is assessed. To this end, the solutions of dynasearch algorithm are compared with those of the lower bounds. In order to conduct this comparison, 1100 instances, i.e., 100 instances for 11 problem sizes, were generated. The parameter values were selected randomly from the following intervals.

$$sl_j \in U[0.2, 0.95]$$

$$\beta \in U[10n, 30n]$$

$$b \in U[30n, 100n]$$

$$p_j \in U[50, 200]$$

$$a_j \in U[p_j, p_j + 300]$$

All the experiments were implemented on a laptop computer with 2.40GHz of CPU and 2.00GB of RAM. Dynasearch was coded in C++ and the lower bound model was solved by GAMS/CPLEX.

A popular index for analyzing the quality of a solution approach is the average relative percentage deviation, RPD% for short. This index calculates the relative difference between heuristic solutions and the reference

ones. The reference solutions can be optimal, best known, or lower bound. As shown in expression (26), RPD% is defined here as the relative difference between the dynasearch and lower bound solutions. In this expression, *Dyna* denotes the dynasearch solution and *LB* implies the lower bound.

$$RPD\% = \frac{Dyna - LB}{LB} \times 100\% \quad (26)$$

The summary of the numerical results is reported in Table 2. As shown in this table, the numerical studies were performed on 11 different problem size categories ranging from 20 to 70 jobs ($n=20$ to 70). Each category included 100 randomly generated problem instances (#Instances=100). Since the proposed lower bound is a mixed integer linear programming problem, the optimal solutions may not be found in a reasonable time. GAMS has some stopping criteria. In this paper, we used *reslim* and *optcr*. When using *optcr*, GAMS stops as soon as the proportional difference between the incumbent solution and the best possible solution is less than the *optcr* parameter. On the other hand, *reslim* limits the execution time spent for solving the problem. The *best found* and *best possible*, i.e., the lower bound, solutions are reported at the termination time. Obviously, the lower bounds of the proposed model, can be used as the bounds for the original problem. These *best possible* values are used as *LB* in expression (26).

As shown in Table 2, the average run time of dynasearch algorithm is less than a few seconds for all problem sizes. Moreover, the average RPD% over all problems is less than 2%. Note that this gap is measured between the lower bound and heuristic solutions. Therefore, the average proportional difference between dynasearch and *optimal* solutions may be even less than 2%.

Box plots can provide more information about the distribution of data than average. These plots include the minimum, first quartile, median, third quartile, and maximum of data, respectively. Further, in the presence of extreme values, the median may be preferred over the simple average. In the last column of Table 2, the box plot of each problem category is depicted. For instance, according to the last row of this table, 75% of test problems with 70 jobs have the RPD% of less than 3.

Table 2
Gap between the dynasearch and the lower bound

n	# Grid points (K)	# Instances	Termination conditions for LB	Average run time of LB (seconds)	Average run time of dynasearch (seconds)	Average RPD%	Boxplot for RPD%
20	11	100	reslim=600 optcr=0.000	<1	<1	1.16	
25	11	100	reslim=600 optcr=0.000	2.4	<1	0.86	
30	11	100	reslim=600 optcr=0.000	<1	<1	0.72	
35	11	100	reslim=600 optcr=0.000	24.1	<1	0.72	
40	11	100	reslim=120 optcr=0.0005	44.0	<1	0.82	
45	5	100	reslim=600 optcr=0.0005	30.9	<1	1.21	
50	5	100	reslim=600 optcr=0.0005	46.3	1.1	1.63	
55	5	100	reslim=180 optcr=0.002	47.8	1.4	1.57	
60	5	100	reslim=180 optcr=0.002	70.1	1.8	1.76	
65	5	100	reslim=180 optcr=0.002	88.8	2.4	1.90	
70	5	100	reslim=180 optcr=0.002	124.0	2.9	2.37	

In order to make statistical inference about the average *RPD%*, one sample *z-test* was performed for each problem size. The statistical tests were performed in Minitab 16. The results of these tests are reported in Figure 3. All *p-values* were less than 0.05. Hence, all null hypotheses were accepted at 95% confidence level. For instance, according to this figure, the mean of the average

difference between dynasearch solutions and lower bounds for problems with 70 jobs was not greater than 2.1%. Hence, the mean difference of heuristic solutions and *optimal solutions* was *strictly* less than 2.1%. Regarding the complexity of the problem discussed in Subsection 3.2, the proposed dynasearch heuristic seems efficient.

n	Results of hypothesis testing						
20	Test of mu = 1.1 vs > 1.1						
	The assumed standard deviation = 0.765489						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	1.1572	0.7655	0.0765	1.0313	0.75 0.227
25	Test of mu = 0.8 vs > 0.8						
	The assumed standard deviation = 0.531588						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	0.8642	0.5316	0.0532	0.7768	1.21 0.113
30	Test of mu = 0.65 vs > 0.65						
	The assumed standard deviation = 0.480157						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	0.7244	0.4802	0.0480	0.6454	1.55 0.061
35	Test of mu = 0.65 vs > 0.65						
	The assumed standard deviation = 0.480157						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	0.7223	0.5164	0.0516	0.6373	1.40 0.081
40	Test of mu = 0.75 vs > 0.75						
	The assumed standard deviation = 0.712152						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	0.8192	0.7122	0.0712	0.7021	0.97 0.165
45	Test of mu = 1.1 vs > 1.1						
	The assumed standard deviation = 0.714515						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	1.2114	0.7145	0.0715	1.0939	1.56 0.060
50	Test of mu = 1.5 vs > 1.5						
	The assumed standard deviation = 0.996387						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	1.6300	0.9964	0.0996	1.4661	1.30 0.096
55	Test of mu = 1.5 vs > 1.5						
	The assumed standard deviation = 0.936846						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	1.5708	0.9368	0.0937	1.4167	0.76 0.225
60	Test of mu = 1.6 vs > 1.6						
	The assumed standard deviation = 1.15098						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	1.760	1.151	0.115	1.571	1.39 0.082
65	Test of mu = 1.7 vs > 1.7						
	The assumed standard deviation = 1.26307						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	1.899	1.263	0.126	1.691	1.57 0.058
70	Test of mu = 2.1 vs > 2.1						
	The assumed standard deviation = 1.67306						
	Variable	N	Mean	StDev	SE Mean	95% Lower Bound	Z P
	rpD%	100	2.367	1.673	0.167	2.092	1.60 0.055

Fig. 3. Z-test of RPD% measure

8. Conclusion and Future Research

In this paper, an integrated solution approach was developed for a machine scheduling problem with reliable due date setting by considering an unexpected unavailability. Given the sequence of jobs, the closed-form formula of optimal due dates was derived. It was proved that the problem is NP-hard in the strong sense. The nonlinear mathematical programming formulation of the problem was built. The formulation was the base for designing a tight lower bound. An iterated dynasearch algorithm was then proposed to solve the problem. Due to the proved NP-hardness and nonlinearity of the problem, it was very tough to find the optimal solutions to evaluate the efficiency of dynasearch algorithm. Hence, a mixed integer programming lower bound was developed. The small gaps between heuristic and lower bound solutions revealed the good quality of the proposed dynasearch as well as the tightness of the lower bound.

To the best of our knowledge, this study is the first one dealing with due date setting by considering random unavailability within a scheduling environment. Therefore, a promising avenue for future research is to study various integrated due date setting and scheduling problems with the consideration of random unavailability intervals. Further, the proposed dynasearch heuristic has good quality solutions. Hence, this heuristic can be combined as an initial solution, i.e., upper bound, with a relaxation of the proposed lower bound to develop a branch and bound algorithm for the studied problem.

References

- Agnetis, A., Detti, P. & Martineau, P. (2017). Scheduling nonpreemptive jobs on parallel machines subject to exponential unrecoverable interruptions. *Computers & Operations Research*, 79(1), 109–118.
- Angel, E., & Bampis, E. (2005). A multi-start dynasearch algorithm for the time dependent single-machine total weighted tardiness scheduling problem. *European Journal of Operational Research*, 162(1), 281–289.
- Baker, K.R. (2014). Setting optimal due dates in a basic safe-scheduling model. *Computers & Operations Research*, 41(1), 109–114.
- Baker, K.R., & Trietsch, D. (2015). Trading off due-date tightness and job tardiness in a basic scheduling model. *Journal of Scheduling*, 18(3), 305-309.
- Baker, K.R., & Trietsch, D. (2009). Safe scheduling: Setting due dates in single-machine problems. *European Journal of Operational Research*, 196(1), 69–77.
- Bergamini, M.L., Grossmann, I., Scenna, N., & Aguirre, P. (2008). An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. *Computers & Chemical Engineering*, 32(3), 477–493.
- Congram, R. K., Potts, C.N., & Velde, S.L.V.D. (2002). An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem. *INFORMS Journal on Computing*, 14(1), 52-67
- Ding, J., Lu, Z., Cheng, T.C.E., & Xu, L. (2016). Breakout dynasearch for the single-machine total weighted tardiness problem. *Computers & Industrial Engineering*, 98(1), 1–10.
- Dumitrescu, I., & Stutzle, T. (2010). Usage of Exact Algorithms to Enhance Stochastic Local Search Algorithms, In: *Matheuristics*. Springer, New York.
- Elyasi, A., & Salmasi, N. (2013). Due date assignment in single machine with stochastic processing times. *International Journal of Production Research*, 51(8), 2352-2362.
- Gerstl, E., & Mosheiov, G. (2013). Minmax due-date assignment with a time window for acceptable lead-times. *Annals of Operations Research*, 211(1), 167–177.
- Grosso, A., Croce, F.D., & Tadei, R. (2004). An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters*, 32(1), 68–72.
- Huo, Y., Reznichenko, B., & Zhao, H. (2014). Minimizing total weighted completion time with an unexpected machine unavailable interval. *Journal of Scheduling*, 17(2), 161–172.
- Kacem, I., & Kellerer, H. (2016). Semi-online scheduling on a single machine with unexpected breakdown. *Theoretical Computer Science*, 646(1), 40–48.
- Kacem, I., Nagih, A., & Seifaddini, M. (2014). Maximum lateness minimization with positive tails on a single machine with an unexpected non-availability interval. *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, IEEE Press, 1-5.
- Kedad-Sidhoum, S., & Sourd, F. (2010). Fast neighborhood search for the single machine earliness-tardiness scheduling problem. *Computers & Operations Research*, 37(8), 1464–1471.
- Keskinocak, P., & Tayur, S. (2004). Due date management policies, In: *Handbook of Quantitative Supply Chain Analysis*. Kluwer Academic Publishers, Boston.
- Miller, C.E., Tucker, A.W., & Zemlin, R.A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4), 326-329.
- Montgomery, D.C., & Runger, G.C. (2014). *Applied Statistics and Probability for Engineers*. 6th edition, Wiley, Danvers.
- Mor, B., Mosheiov, G., & Shabtay, D. (2013). A note: Minmax due-date assignment problem with lead-time cost. *Computers & Operations Research*, 40(8), 2161–2164.
- Öncan, T., Altınel, İ.K., & Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3), 637–654.

- Pinedo, M. L. (2012). Scheduling: theory, algorithms, and systems. 4th edition, *Springer Science & Business Media*, New York.
- Shabtay, D. (2010). Scheduling and due date assignment to minimize earliness, tardiness, holding, due date assignment and batch delivery costs. *International Journal of Production Economics*, 123(1), 235–242.
- Shabtay, D. (2016). Optimal restricted due date assignment in scheduling. *European Journal of Operational Research*, 252(1), 79–89.
- Shabtay, D., & Steiner, G. (2006). Two due date assignment problems in scheduling a single machine. *Operations Research Letters*, 34(6), 683–691.
- Slotnick, S.A. (2014). Lead-time quotation when customers are sensitive to reputation. *International Journal of Production Research*, 52(3), 713-726.
- Slotnick, S.A., & Sobel, M.J. (2005). Manufacturing lead-time rules: Customer retention versus tardiness costs. *European Journal of Operational Research*, 163(3), 825–856.
- Sourd, F. (2006). Dynasearch for the earliness–tardiness scheduling problem with release dates and setup constraints. *Operations Research Letters*, 34(5), 591–598.
- Yin, Y., Cheng, T.C.E., Yang, X., & Wu, C.C. (2015). Two-agent single-machine scheduling with unrestricted due date assignment. *Computers & Industrial Engineering*, 79, 148–155.
- Yin, Y., Wang, D. J., Wu, C.C., & Cheng, T.C.E. (2016). CON/SLK Due Date Assignment and Scheduling on a Single Machine with Two Agents. *Naval Research Logistics*, 63(5), 416-429.
- Yin, Y., Wang, Y., Cheng, T.C.E., Liu, W., & Li, J. (2017). Parallel-machine scheduling of deteriorating jobs with potential machine disruptions. *Omega*, 69(1), 17–28.

This article can be cited: Iranpoor M. & Fatemi Ghomi S.M.T (2019). Integrated Due Date Setting and Scheduling on a Single Machine Considering an Unexpected Unavailability. *Journal of Optimization in Industrial Engineering*. 12 (1), 1-13



http://www.qjie.ir/article_538023.html

DOI: 10.22094/JOIE.2017.644.1415

Appendix A: Proof of lemma 2.

In this case, in order to satisfy the required service level, the probability of the occurrence of unavailability should be considered. Accordingly, the feasible due date should be

$$d_{[j]} > \sum_{k=1}^j p_{[k]}$$

Let $\rho_{[j]}$ be an extra time added to $\sum_{k=1}^j p_{[k]}$ to obtain an acceptable due date for job $[j]$. In other words,

$$d_{[j]} = \sum_{k=1}^j p_{[k]} + \rho_{[j]}$$

The stochastic variable of the completion time of job $[j]$ before $d_{[j]}$ can be calculated in two ways.

First, $\sum_{k=1}^j p_{[k]} < C_{[j]} \leq d_{[j]}$ and $C_{[j]} \leq \sum_{k=1}^j p_{[k]}$ are mutually exclusive events. So, the general addition rule (Montgomery and Runger, (2014)) results in (27).

$$\Pr\left(\sum_{k=1}^j p_{[k]} < C_{[j]} \leq d_{[j]}\right) = \Pr\left(C_{[j]} \leq d_{[j]}\right) - \Pr\left(C_{[j]} \leq \sum_{k=1}^j p_{[k]}\right) \Rightarrow$$

$$\Pr\left(\sum_{k=1}^j p_{[k]} < C_{[j]} \leq d_{[j]}\right) = sl_{[j]} - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}} \quad (27)$$

Second, the definition of conditional probability (Montgomery and Runger, (2014)) results in (28).

$$\Pr\left(\sum_{k=1}^j p_{[k]} < C_{[j]} \leq d_{[j]}\right) = \Pr\left(\left(C_{[j]} \leq d_{[j]}\right) \cap \left(C_{[j]} > \sum_{k=1}^j p_{[k]}\right)\right)$$

$$\begin{aligned} &= \Pr\left(C_{[j]} \leq d_{[j]} \mid C_{[j]} > \sum_{k=1}^j p_{[k]}\right) \times \Pr\left(C_{[j]} > \sum_{k=1}^j p_{[k]}\right) \\ &= \Pr\left(T_r \leq \rho_{[j]}\right) \times \left(1 - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}}\right) \Rightarrow \end{aligned}$$

$$\Pr\left(\sum_{k=1}^j p_{[k]} < C_{[j]} \leq d_{[j]}\right) = \left(1 - e^{-\frac{\rho_{[j]}}{\beta_b}}\right) \times \left(1 - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}}\right) \quad (28)$$

The left hand sides of equations (27) and (28) are equal. After putting the right hand sides in an equality and doing some calculations, it is concluded that

$$\rho_{[j]} = \beta_r \ln \left(\frac{\left(1 - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}}\right)}{\left(1 - sl_{[j]}\right)} \right)$$

By the substitution of $\rho_{[j]}$ in $d_{[j]} = \sum_{k=1}^j p_{[k]} + \rho_{[j]}$, the proof is completed.

$$d_{[j]} = \sum_{k=1}^j p_{[k]} + \beta_r \ln \left(\frac{\left(1 - e^{-\frac{\sum_{k=1}^j p_{[k]}}{\beta_b}}\right)}{\left(1 - sl_{[j]}\right)} \right)$$

Appendix B. The detailed pseudo-code of dynasearch heuristic

Let σ be an arbitrary sequence and obj the value of its objective function. The pseudo-code of the dynamic

programming algorithm for *SSRU*, which is a modification of one used in (Angel and Bampis, (2005)), is shown in Figure 4.

As shown in Figure 1, the dynamic programming algorithm is called at each iteration with an input sequence. The search within this dynamic programming algorithm is conducted in a backward manner. This

algorithm applies the best set of pairwise independent moves on the input solution. Symbols i, j and k denote the jobs in these positions of the sequence. $dyna(i)$ is the best objective function of jobs in positions i through n . $forward_cost[j]$ represents the total costs of jobs in positions i through n when jobs i and j are swapped.

```

Procedure Dynamic_programming
  obj = 0
  For i=n to 1 step -1
    is_computed[j]←false  ∀j∈{1,...,n};  obj=obj+dynasearch(i);
  End procedure
Function dynasearch(i)
  t = ∑j=1i-1 p[j]
  if i=n+1 return 0

  if i=n  return w[n] ⎡ t + p[n] + ⎣ βr ln ⎛  $\frac{1-e^{-\frac{t+p_{[n]}}{\beta_b}}}{1-sI_{[n]}}$  ⎞ ⎦ - a[n] ⎤

  elseif is_computed[i] return best_forward_cost[i]

  forward_cost [ i ] = w[i] ⎡ t + p[i] + ⎣ βr ln ⎛  $\frac{1-e^{-\frac{t+p_{[i]}}{\beta_b}}}{1-sI_{[i]}}$  ⎞ ⎦ - a[i] ⎤ + dynasearch ( i + 1)

  for j=i+1 to n
    if j=i+1 tj=t+p[j] else tj=tk-1+p[k-1];

    forward_cost [ j ] = w[j] ⎡ t + p[j] + ⎣ βr ln ⎛  $\frac{1-e^{-\frac{t+p_{[j]}}{\beta_b}}}{1-sI_{[j]}}$  ⎞ ⎦ - a[j] ⎤
    + ∑i<k<j ⎡ w[k] ⎡ tk + p[k] + ⎣ βr ln ⎛  $\frac{1-e^{-\frac{t_k+p_{[k]}}{\beta_b}}}{1-sI_{[k]}}$  ⎞ ⎦ - a[k] ⎤ ⎤
    + w[i] ⎡ tj + p[i] + ⎣ βr ln ⎛  $\frac{1-e^{-\frac{t_j+p_{[i]}}{\beta_b}}}{1-sI_{[i]}}$  ⎞ ⎦ - a[i] ⎤ + dynasearch ( j + 1)

  j* = arg mini≤j≤n forward_cost[j] ;  swap jobs [i] and [j*] in σ ; is_computed[i]=true;
  best_forward_cost[i]=forward_cost[j*]; return best_forward_cost[i];
end function
  
```

Fig. 4. Detailed pseudo-code of dynasearch algorithm for *SSRU*