Journal of Industrial and Systems Engineering
Vol. 8, No. 4, pp 59-71
Autumn 2015

JISE

Maximizing service level in a β-robust job shop scheduling model

Seyed-Morteza Khatami[1*], Mohammad Ranjbar[1], Morteza Davari [2]

[1]*Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran*

[2]*Research group for Operations Management, KU Leuven, Belgium*

sm_khatami@stu.um.ac.ir, m_ranjbar@um.ac.ir, morteza.davari@econ.kuleuven.be

Abstract

In the realm of scheduling problems, different sources of uncertainty such as probabilistic durations of jobs or stochastic breakdowns of machines can arise. Given this, one highly desirable characteristic of an intelligent schedule is to bring better punctuality with less efficiency-loss because a dominant factor in customer appreciation is punctuality. It is also one of the most intangible topics for managers when a due date is predetermined to deliver jobs. In this paper, we address the β-robust job shop scheduling problem when the processing time of each operation is a normal random variable. We intend to minimize the deviation of makespan from a common due date for all jobs which corresponds to maximizing the service level, defined as probability of the makespan not exceeding the given due date. We develop a branch-and-bound algorithm to solve the problem. Using a set of generated benchmark instances, the performance of the developed algorithm has been evaluated.

Keywords: Job shop; Stochastic scheduling; Branch-and-bound algorithm.

1-Introduction

In the real production systems, "optimal" schedules, once deployed, are affected by irregularities such as variability in the job durations or resource breakdowns and are far from optimal in practice. Nevertheless, most of researches in the literature have been focused on deterministic models, which ignore uncertain deviations.

In this paper, we address a job shop scheduling problem with stochastic processing times, described by normal random variables, and a common due date for all jobs that is appointed using negotiation with the customer. We aim to find a *β-robust* schedule that maximizes the service level, which is the probability of the makespan not exceeding the predetermined due date. In the remainder of this text, we refer to this problem as the *β-robust job shop scheduling problem* (βRJSSP). We develop a branch-and-bound (B&B) algorithm to solve the proposed problem optimally.

*Corresponding author.
ISSN: 1735-8272,

The job shop scheduling problem (JSSP) is a classical NP-hard (Lenstra et al. 1977) combinatorial problem which has been widely studied in the literature. Most of the researches have considered the deterministic JSSP (DJSSP) while some of the recent works have been devoted to the stochastic JSSP (SJSSP). But to the best of our knowledge, the βRJSSP has not been studied in the literature and is introduced for the first time in this paper.

Many studies have been published on DJSSP in which all parameters are known in advance. The most classic and frequently studied problem in this field has been the DJSSP with objective of makespan minimization, denoted as $Jm||C_{max}$based upon the notation of Graham et al. (1979). Most of the exact algorithms that have been proposed for the DJSSP have B&B procedures. In this filed, we can cite the works of Carlierand Pinson (1989) and Applegate and Cook (1991). Also, numerous heuristic and metaheuristic procedures have been proposed for the DJSSP. The well-known *shifting bottleneck heuristic* procedure, developed by Adams et al. (1988), has been one of the most successful heuristic procedure for the $Jm||C_{max}$. Moreover, metaheuristic approaches such as genetic algorithm of Spanos et al. (2014), tabu search algorithm of Zhang et al. (2008), and scatter search of Ranjbar and Najafian Razavi (2012). Moreover, Pardalos et al. (2010) developed an algorithm based upon utilizing the properties of backbone and big valley, have found very good results for the hard instances of the DJSSP.

Stochastic job shop scheduling problem (SJSSP) shows an important aspect of manufacturing systems and is an extended version of the JSSP by introducing some stochastic processing conditions such as stochastic processing times. The JSSP with uncertain arrival times, processing times, due date and part priority was studied by Luh et al. (1999), who developed a solution methodology based on a combined Lagrangian relaxation and a stochastic dynamic programming. Golenko-Ginzburg and Gonik (2002) considered three sets of costs in SJSSP with stochastic processing times in normal, exponential and uniform distributions and treated the problem as the identification of the earliest start times in order to minimize the average cost of storage and tardiness from the delivery time. Tavakkoli-Moghaddam et al. (2005) proposed a hybrid method based on neural network and simulated annealing to the SJSSP.

In the area of β-robust scheduling, Daniels and Carrillo (1997) and Wu et al. (2009) considered β-robust scheduling problem in the single machine environment in which minimizing the risk that the flow time exceeds a given threshold is the main goal. Also, Ranjbar et al. (2012a) introduced the β-robust parallel machine scheduling problem in which a set of jobs with probabilistic (normal) durations must be processed on a set of identical parallel machines with the goal of maximization of the service level.

There are numerous papers in the scheduling literature in which a common due date has been considered for all jobs. In the single machine environment, we can refer to Lin et al. (2007) and Yin et al. (2013). Also, in the parallel machine environment, we can mention following research works such as Alidaee and Panwalkar (1993), and Tuong et al. (2010).

Since DJSSP is NP-hard and the βRJSSP is a generalization of the DJSSP in which processing times are stochastic and the objective function is more complicated, the βRJSSP is NP-hard as well.

The contributions of this article are twofold: (1) we provide the first description of the βRJSSP and present a non-linear formulation for it; (2) we develop an exact depth-first B&B algorithm to solve the problem.

The remainder of this paper is organized as follows. In Section 2, we provide a formal description of the problem and present a non-linear binary formulation. Section 3 describes the B&B algorithm while computational experiments are reported in Section 4. Finally, conclusions and suggestions for future works are presented in Section 5.

2- Problem description and formulation

The βRJSSP may be formulated as follows. Consider n jobs $J_1, J_2, \dots, J_n$ and m different machines $M_1, M_2, \dots, M_m$. For simplicity, we assume recirculation is not allowed. Thus, each job J_j consists of at most m operations with predetermined order where O_{ij} indicates the operation of job J_j which must be processed on machine M_i. The processing time of operation O_{ij} is represented by a stochastic variable p_{ij} having normal probability distribution function (p_{ij}) and cumulative distribution function Φ, with expectation $E(p_{ij}) = \mu_{ij}$ and variance $Var(p_{ij}) = \sigma_{ij}^2$. We choose the normal random variables for operations' processing times because most of practical situations can be modeled by this distribution (Pinedo, 2014). Also, if we relax the normal distribution assumption from our assumptions, our results are approximately valid yet based on the central theorem limit (Sarin et al., 2014).

There is only one machine of each type which can process only one operation at a time. Such an operation must be processed without preemption. Moreover, a job cannot be processed by two machines at the same time. We assume that neither release dates nor due dates are imposed for any individual job but, a due date δ is put forward by the customer for the completion of the entire job set. Moreover, we suppose that the machines are always available and we define the service level Π as the probability that the makespan does not exceed the due date δ. According to these restrictions, we have to find an order of all operations O_{ij} for each machine M_i such that for the corresponding schedule the service level is maximal.

In order to model the βRJSSP, we use the *disjunctive graph model*, introduced by Roy and Sussmann (1964). A disjunctive graph $G(N, C, D)$ is a mixed graph with node set N, a set C of directed arcs (conjunctions) and a set D of undirected arcs (disjunctions). The set N of nodes represents the set of all operations plus two dummy nodes (0,0) and (m+1,n+1). The start dummy operation (0,0) is the direct predecessor of the first operation of each job while the end dummy operation (m+1,n+1) is the direct successor of the last operation of each job.

In the disjunctive graph $G(N,C,D)$, each node (i, j) indicates the event of starting operation O_{ij}. Thus, each arc emanated from node (i, j) is weighted with the stochastic processing time p_{ij} while all arcs emanated from dummy start node are weighted by zero. The set C of *conjunctive* arcs represent the precedence relations between consecutive operations of the same job. Thus, we have a conjunctive arc $(i, j) \rightarrow (i', j)$ for each pairs of directly precedence related operations (i, j) and (i', j) of jobs J_j. The meaning of the conjunctive arc $(i, j) \rightarrow (i', j)$ is that a feasible schedule must satisfy the condition $X_{ij} + p_{ij} \leq X_{i'j}$ where variable X_{ij} indicates the earliest start time of operation O_{ij}.

The set D of *disjunctive* arcs represents the different orders in which operations on the same machine may be scheduled. It consists of undirected arcs between all pairs of operations which have to be processed on the same machine, i.e. for any two operations (i, j) and (i, j'), the set D contains an undirected arc denoted by $(i, j) \leftrightarrow (i, j')$. The meaning of the disjunctive arc $(i, j) \leftrightarrow (i, j')$ is that a feasible schedule must either satisfy $X_{ij} + p_{ij} \leq X_{ij'}$ or $X_{ij'} + p_{ij} \leq X_{ij}$.

The problem of finding a feasible schedule for the βRJSSP is equivalent to the problem of fixing a direction for each disjunctive. A set $S(D)$ of fixed disjunctive arcs is called a *selection* if for each disjunction a direction has been fixed. Each selection $S(D)$ is feasible *iff* the corresponding graph $G(N,C,S(D))$ is acyclic. The Figure 1 indicates an example problem of the βRJSSP with n=4 jobs and m=4 machines in which disjunctive arcs are shown by directed dash lines.

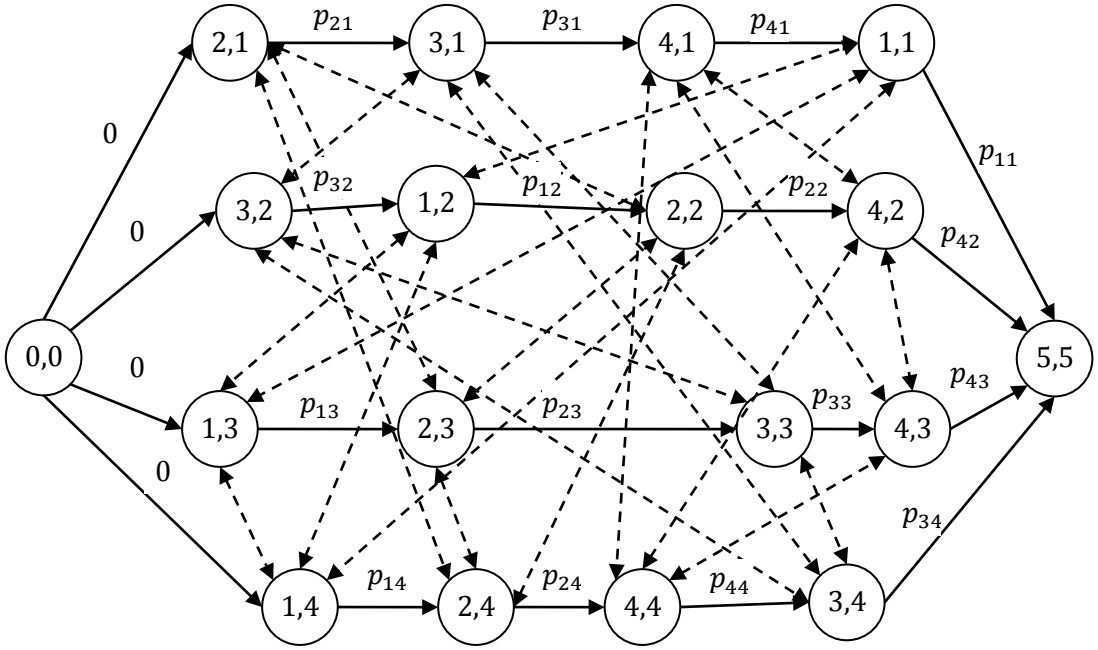

Figure 1. The example problem

The remaining job characteristics are shown in Table 1. This instance will also be used in Section 3 for further illustrations.

Table 1: Data of the example problem

Job	Sequence of operations	Mean of processing times	Variance of processing times
J_1	$M_2 - M_3 - M_4 - M_1$	$64 - 97 - 36 - 61$	$23.2 - 302.4 - 119.2 - 28.1$
J_2	$M_3 - M_1 - M_2 - M_4$	$84 - 15 - 16 - 55$	$398.1 - 27.1 - 5.1 - 56.5$
J_3	$M_1 - M_2 - M_3 - M_4$	$31 - 39 - 89 - 63$	$43.9 - 13.7 - 559.0 - 76.2$
J_4	$M_1 - M_2 - M_4 - M_3$	$25 - 16 - 48 - 85$	$57.1 - 18.3 - 83.1 - 274.8$

If we assume $E(X_{ij}) = \lambda_{ij}$ and $Var(X_{ij}) = \theta_{ij}^2$, it is obvious that λ_{ij} and θ_{ij}^2 are variables depending on *S*(*D*). Now, we define the service level as $\Pi = Pr(X_{m+1,n+1} \leq \delta) = \Phi\left(\frac{\delta - \lambda_{m+1,n+1}}{\theta_{m+1,n+1}}\right)$ where stochastic variable $X_{m+1,n+1}$ has normal distribution and $\lambda_{m+1,n+1}$ and $\theta_{m+1,n+1}$ are calculated based upon the *program evaluation and review technique* (PERT) developed by Malcolm et al. (1959). In the PERT, it is assumed that for each node (i,j), $\lambda_{ij}$and θ_{ij}^2 are calculated using the expected value and variance of processing times for the relevant operations and by working forward through the network. In other words, if h_{ij} indicates the longest path in the graph *G*(*N*,*C*,*S*(*D*)) from start dummy node $(0,0)$ to node(i,j), then $\lambda_{ij}(\theta_{ij}^2)$ equals to the summation of expected values (variances) of operations belonging to h_{ij}.

It should be mentioned here that there are some weaknesses in the statistical reasoning of PERT but we ignore them for simplification. For example, it supposes the stochastic independence of the nodes as well as of their relations which is not completely fulfilled in the βRJSSP. Therefore, the objective function value that we calculate is a heuristic approximation to the real service level.

In order to formulate the βRJSSP, in the following we introduce variable$Y_{ijj'}$ as follows.

$$Y_{ij'j} = \begin{cases} 1 & \text{if disjunctice arc } (i,j') \leftrightarrow (i,j) \text{ is fixed as } (i,j') \rightarrow (i,j) \\ 0 & \text{otherwise} \end{cases}$$

The model will be as follows:

$$Max\Pi = Pr\left(X_{m+1,n+1} \leq \delta\right) = \Phi\left(\frac{\delta - \lambda_{m+1,n+1}}{\theta_{m+1,n+1}}\right) \quad (1)$$

Subject to:

$$Y_{ijj'} + Y_{ij'j} = 1; \qquad \forall (i,j') \leftrightarrow (i,j) \in D \quad (2)$$

$$\lambda_{ij} \geq \lambda_{i'j} + \mu_{i'j}; \qquad \forall (i',j) \rightarrow (i,j) \in C \quad (3)$$

$$\lambda_{ij} \geq \left(\lambda_{ij'} + \mu_{ij'}\right) + M(Y_{ij'j} - 1); \qquad \forall (i,j') \leftrightarrow (i,j) \in D \quad (4)$$

$$\theta_{ij}^2 \geq M\left(\lambda_{i'j} + \mu_{i'j} - \lambda_{ij}\right) + \theta_{i'j}^2 + \sigma_{i'j}^2; \qquad \forall (i',j) \rightarrow (i,j) \in C \quad (5)$$

$$\theta_{ij}^2 \geq M\left(\left(\lambda_{ij'} + \mu_{ij'}\right) + M(Y_{ij'j} - 1) - \lambda_{ij}\right) + \theta_{ij'}^2 + \sigma_{ij'}^2; \qquad \forall (i,j') \leftrightarrow (i,j) \in D \quad (6)$$

$$\lambda_{0,0} = 0 \quad (7)$$

$$\theta_{0,0}^2 = 0 \quad (8)$$

$$\lambda_{ij} \in \mathbb{R}^{\geq 0}; \qquad i=1,\ldots,m; j=1,\ldots,n \quad (9)$$

$$\theta_{ij}^2 \in \mathbb{R}^{\geq 0}; \qquad i=1,\ldots,m; j=1,\ldots,n \quad (10)$$

$$Y_{ijj'} \in \{0,1\}; \qquad \forall (i,j') \leftrightarrow (i,j) \in D \quad (11)$$

The objective function (1) maximizes the service levelΠ. The constraint (2) indicates that only one direction should be fixed for each disjunctive arc $(i,j') \leftrightarrow (i,j)$. The constraints (3) and (4) impose the calculation rule of λ_{ij} for conjunctive and disjunctive arcs, respectively. Similarly, the constraints (5) and (6) show how the variance is calculated in each node (i,j) in which constraints (5) are developed for conjunctive arcs while constraints (6) are established for disjunctive arcs. When there are two or more paths with the maximum expected value and terminated to a single node, the path with the maximum variance should be considered. In this model, for each node (i,j), constraints (5) and (6) impose lower bounds for θ_{ij}^2. Thus, we have $maximum\ lower\ bound \leq \theta_{ij}^2 \leq \infty$, but since the objective function tries to minimize θ_{ij}^2, the model set $\theta_{ij}^2 = maximum\ lower\ bound$.

Moreover, the initial values $\lambda_{0,0}$ and $\theta_{0,0}^2$ are set to zero in constraints (7) and (8). The last three constraints indicates the range of variables λ_{ij}, θ_{ij}^2 and $Y_{ijj'}$ where $\mathbb{R}^{\geq 0}$ represents the set of non-negative real numbers. It should be noticed that the objective function (1) is non-linear non-convex; hence, this model cannot be solved using available commercial solvers.

3- A branch-and-bound algorithm

In light of the NP-hardness of the βRJSSP, an exact algorithm with better than exponential time complexity is unlikely to exist for the βRJSSP, and we will devise a depth-first B&B algorithm to implicitly enumerates the solution space. We use the concept of "arc insertion" as described by Singer and Pinedo (1998) to determine a solution for minimizing the total weighted tardiness in DJSSP.

In our B&B algorithm, we first develop a heuristic procedure to generate an initial solution, considered as a lower bound (*LB*) for the problem and it will be updated whenever a better solution is found by the B&B algorithm. In order to avoid a complete enumeration procedure, we develop two dominance rules.

3-1- Definitions and properties

In order to describe the rest of the paper, we first need to propose some definitions and properties as follows.

***Definition* 1:** For simplification, after this, we show each node (i, j) by a single and unique index $k = mj + i$. Thus, the example problem will be represented as shown in Figure 2.

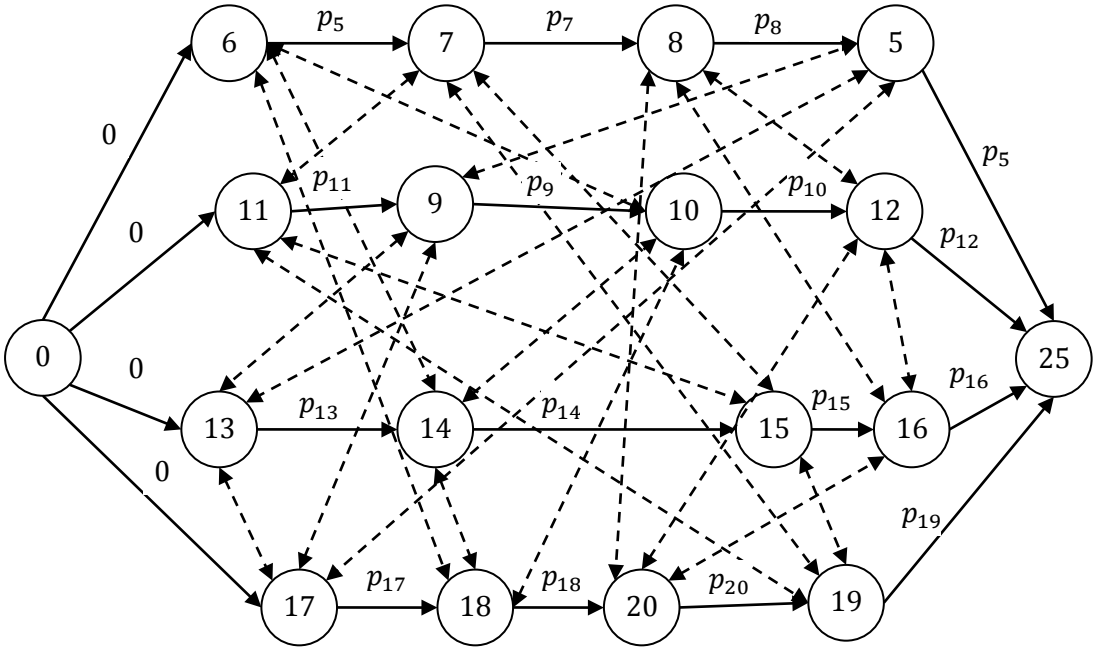

Figure 2. new representation of the example problem

***Definition* 2:** For each operation *k*, we define the expected value of tail (head) $E(q_k)\big(E(h_k)\big)$ as a lower bound for the expected value of the longest path between dummy end (start) node and node *k*. Although our definitions for tail and head are a bit different from the ones developed by Brucker et al. (1994), their developed procedures with complexity order $O(|C|)$ can be used here. We develop equations (14) and (15) to calculate $E(q_k)$and $E(h_k)$, respectively.

$$E(q_k) = max\{\mu_k + E(q_l);\ (k \to l) \in C\} \quad (14)$$
$$E(h_k) = max\{\mu_l + E(h_l);\ (l \to k) \in C\} \quad (15)$$

In other words, $E(h_k)$ and $E(q_k)$ are lower bounds on the expected value of the earliest and the latest start timeof operation *k*, respectively, in which μ_k is considered as the processing time of operation *k*. Equations (14) and (15) require initialization which are given by $E\big(q_{m(n+1)+(m+1)}\big) = E(h_0) = 0$. It should be noticed that $E\big(h_{m(n+1)+(m+1)}\big) = E(q_0) = max\{E(h_k) + E(q_k); k \in N\}$ indicates the expected value of the critical path length, determined based on critical path method (CPM).

***Definition* 3:** For each operation *k*, we define the variance of tail (head) $Var(q_k)\big(Var(h_k)\big)$ as the summation of variances of operations which belong to the longest path between dummy end (start) node and node *k*.

***Definition* 4:** Schedule $S^p(D)$ indicates a partial selection of disjunctive arcs in which some disjunctive arcs are unfixed such as $k \leftrightarrow l$ and Π^p shows its corresponding objective value.

3-2- A heuristic procedure for an initial solution

We develop a heuristic procedure to generate an initial solution, leading to a lower bound (*LB*) that can be computed in a very short running time. In this procedure, described in Algorithm1, the graph $G\big(N, C, S^p(D)\big)$ in which $S^p(D) = \emptyset$ is given as input. Assume P indicates the critical path in this graph. Algorithm 1 is an iterative algorithm in which a direction is fixed for a disjunctive arc in each iteration. It should be noticed that only directions are considered which make an acyclic graph.

Algorithm 1: The pseudo-code of the initial solution

1. Let graph $G(N, C, S^p(D))$ as input where with $S^p(D) = \emptyset$ and critical path P.
2. For each direction $k \to l$ of disjunctive arcs $k \leftrightarrow l \in D$, if $G(N, C, S^p(D))$ is acyclic,
Do time calculation of PERT.
3. If P has not been changed, calculate $\Pi^{k \to l}$ and let $k \to l$ in set A_1.
4. End for
5. If $A_1 \neq \emptyset$, then add arc $k^* \to l^* \in A_1$ to $S^p(D)$ and remove it from $A_1$where
$\Pi^{k^* \to l^*} = max\{\Pi^{k \to l}; \forall k \to l \in A_1\}$; let $A_1 = \emptyset$ and go to step 2.
6. For each direction $k \to l$ of disjunctive arcs $k \leftrightarrow l \in D$, if $G(N, C, S^p(D))$ is acyclic,
do time calculation of PERT, calculate $\Pi^{k \to l}$ and let $k \to l$ in set A_2.
7. End for
8. If $A_2 \neq \emptyset$, then add arc $k^* \to l^* \in A_2$ to $S^p(D)$ remove it from $A_2$where
$\Pi^{k^* \to l^*} = max\{\Pi^{k \to l}; \forall k \to l \in A_2\}$; let $A_1 = A_2 = \emptyset$ and go to step 2.
9. Stop.

In overall, Algorithm 1 is divided into two main phases including steps 2 to 5 and steps 6 to 8. In the first phase, we select directions for disjunctive arcs that do not change the current critical path, shown as set A_1. In step 5, the best direction $k^* \to l^*$among all $k \to l \in A_1$ will be fixed where $\Pi^{k^* \to l^*} = max\{\Pi^{k \to l}; \forall\, k \to l \in A_1\}$ and $\Pi^{k^* \to l^*}$ indicates the objective value after fixing $k^* \leftrightarrow l^*$as $k^* \to l^*$. Steps 2 to 5 will be repeated for the new obtained network while for updated A_1we have $A_1 \neq \emptyset$. Next, we follow similar strategy but for all disjunctive arcs. In other words, in step 6, we calculate the selection impact of each direction for each disjunctive arc on the objective value. Next, similar to step 5, the best direct is selected in step 8. For the new obtained network, the algorithm will be resumed from step 2 and it will end when $A_2 = \emptyset$.

If Algorithm 1 is applied to the example problem, the order and selected direction for disjunctive arcs are as follows:$13 \to 5, 17 \to 5, 6 \to 10, 18 \to 10, 13 \to 9, 17 \to 9, 14 \to 10, 20 \to 8, 20 \to 16, 20 \to 4, 9 \to 5, 6 \to 18, 13 \to 17, 14 \to 18, 11 \to 19, 12 \to 6, 7 \to 19, 15 \to 19, 11 \to 15, 6 \to 14,\ 8 \to 16, 11 \to 7, 12 \to 8, 7 \to 15$. Also, the initial solution equals to *LB*=0.73.

3-3- The branching scheme

Usually, each B&B algorithm includes two main schemes, i.e. *branching* and *bounding*. In the *branching* scheme, the search (branching) tree is constructed and different feasible schedules are investigated while in the *bounding* scheme, the goal is to fathom the nodes using suitable and efficient dominance rules. Our branching scheme works based upon the arc insertion scheme in which each node in the branching tree represents a partial selection $S^p(D)$ of disjunctive arcs where the resulting graph $G(N, C, S^p(D))$ is acyclic. Two offspring nodes are generated by branching on a disjunctive arc $k \leftrightarrow l$ that is selected; one node for k → l and another node for→ k. Thus, in the worst case, the complexity order of our B&B algorithm is $O\big(2^{|D|}\big)$ where $|D|$ indicates number of disjunctive arcs.

One important factor in our developed B&B algorithm is the sequence of the arcs insertion. As shown by Ranjbar et al. (2012b), in a deterministic scheduling problem of a project network, the best sequence is based upon non-increasing order of tails. Using a try and error approach, we found this rule as the best rule for the sequence of arcs insertion in this research but we consider the expected values of tails instead of tails (using non-decreasing smaller (larger) operation numbers as the first (second) tie-breaker). For the example problem, the sequence of arcs insertion will be as follows:
$6 \leftrightarrow 14, 6 \leftrightarrow 18, 13 \leftrightarrow 17, 7 \leftrightarrow 11, 7 \leftrightarrow 15, 14 \leftrightarrow 18, 6 \leftrightarrow 10, 11 \leftrightarrow 15, 9 \leftrightarrow 13, 5 \leftrightarrow 13,$
$7 \leftrightarrow 19, 10 \leftrightarrow 14, 9 \leftrightarrow 17, 11 \leftrightarrow 19, 15 \leftrightarrow 19, 5 \leftrightarrow 17, 8 \leftrightarrow 20, 10 \leftrightarrow 18, 16 \leftrightarrow 20,$
$12 \leftrightarrow 20, 8 \leftrightarrow 16, 8 \leftrightarrow 12, 5 \leftrightarrow 9$ and $12 \leftrightarrow 16$.

3-4- The bounding scheme

In order to detect infeasible or low quality solutions, some dominance rules are corporate with our B&B algorithm. To prevent the extension of infeasible solution including cycle, we use the strategy developed by Ranjbar et al. (2012b). In order to detect whether there is any cycle in each node of the search tree, they defined a path matrix $PM_{(nm+2)(nm+2)}$ in which the $PM(k,l) = 1$, if there is a path from node *k* to node *l* in the graph$G(N, C, S^p(D))$and $PM(k,l) = 0$, otherwise. Obviously, there is no cycle in the initial graph $G(N, C, S^p(D))$in which $S^p(D) = \emptyset$ and the following procedure monitor and prohibit the cycle creation in each node of the search tree.

For every two operations k ,l where$k \leftrightarrow l \in D$, we can change disjunctive arc $k \leftrightarrow l$ to arc $k \rightarrow l$ if $\mathrm{PM(l, k)} = 0$. Now, if $\mathrm{PM(l, k)} = 0$ and arc $\mathrm{k \rightarrow l}$ is inserted to graph *G*, the*PM*is updated using the following four rules: a) $\mathrm{PM(k, l)} = 1$,b) $PM(k,j) = 1;\ \forall j \in Suc(l)$,c) $PM(i,l) = 1;\ \forall i \in Pred(k)$d)$PM(i,j) = 1; \forall i \in Pred(k), \forall j \in Suc(l)$. In these four rules, $Pred(k)$ and $Suc(k)$indicate all (direct and indirect) predecessors and successors of operation*k*, respectively, in graph $G(N, C, S^p(D))$.Now, we can establish the *dominance rule* 1 as follows.

Dominance rule 1: Assume in a node of the search tree, disjunctive arc $k \leftrightarrow l$ has to be fixed. If $PM(l,k) = 1$, fathom the node of the search tree in which arc $k \rightarrow l$ is selected.

Proof: Straightforward.

In addition of infeasible solutions, there are partial solutions in the search tree where all of their offspring solutions will have objective values smaller than the best found solution so far. In order to detect such solutions, we need to construct an upper bound (*UB*) in each node of the search tree. For this purpose, consider the optimal graph $G(N, C, S^*(D))$ and its corresponding optimal objective value Π^*. Also, assume Γ_i indicates the set of operations which must be processed by machine $i; i = 1, \dots, m$ and $N^i \subset N$indicates a set of nodes corresponding to operations which should be processed by M_i. Suppose Π_i^* shows the objective function value calculated based upon optimal graph $G(N^i, C, S^*(D))$. It is clear that $\Pi^* \leq min\{\Pi_i^*; i = 1, \dots, m\}$. Thus, if we can calculate an upper bound for each Π_i^*, it is also a valid upper bound for Π^*. Assume Π_i^p represents the objective function value calculated based upon operations $k \in \Gamma_i$in graph $(N, C, S^p(D))$. In the Algorithm 2, we develop an upper bound $UB_i$for each Π_i^p which is an upper bound for Π_i^* and Π^* as well.

Algorithm 2: The pseudo-code of UB_i

1. Let $k_1^* = arg\left\{min_{k\in\Gamma_i}\{E(h_k)\}\right\}$and $k_2^* = arg\{min_{k\in\Gamma_i}\{E(q_k)-\mu_k\}\}$. Put $k_i^*; i = 1,2$ in set $H_i; i = 1,2$ (initialized as an empty set).
2. Let $l_1^* = arg\{min_{l\in\Gamma_i}\{Var(h_l)\}\}$and $l_2^* = arg\{min_{l\in\Gamma_i}\{Var(q_l)-\sigma_l^2\}\}$. Put $l_i^*; i = 1,2$ in set $Q_i; i = 1,2$ (initialized as an empty set).
3. If $|H_1 \cap H_2| = 1$ and $|H_1| = |H_2| = 1$, then for the common operation k_i^*, let $mean_i = min(\alpha_i, \beta_i)$ where α_i and β_i are calculated as follows:

$$\alpha_i = E\left(h_{k_i^*}\right) + \sum_{k\in\Gamma_i}\mu_k + min_{k\neq k_i^*\in\Gamma_i}\{E(q_k)-\mu_k\}$$

$$\beta_i = min_{k\neq k_i^*\in\Gamma_i}\{E(h_k)\} + \sum_{k\in\Gamma_i}\mu_k + \left\{E\left(q_{k_i^*}\right)-\mu_{k_i^*}\right\}$$

4. Else, calculate *mean* as follows:

$$mean_i = min_{k\in\Gamma_i}\{E(h_k)\} + \sum_{k\in\Gamma_i}\mu_k + min_{k\in\Gamma_i}\{E(q_k)-\mu_k\}$$

5. If $|Q_1 \cap Q_2| = 1$ and $|Q_1| = |Q_2| = 1$, then for the common operation l_i^*, let $variance_i = min(\omega_i, \gamma_i)$ where ω_i and γ_i are calculated as follows:

$$\omega_i = Var\left(h_{l_i^*}\right) + \sum_{l\in\Gamma_i}\sigma_l^2 + min_{l\neq l_i^*\in\Gamma_i}\left\{Var(q_l)-\sigma_l^2\right\}$$

$$\gamma_i = min_{l\neq l_i^*\in\Gamma_i}\{Var(h_l)\} + \sum_{l\in\Gamma_i}\sigma_l^2 + \left\{Var\left(q_{l_i^*}\right)-\sigma_{l_i^*}^2\right\}$$

6. Else, calculate *variance* as follows:

$$variance_i = min_{l\in\Gamma_i}\{Var(h_l)\} + \sum_{l\in\Gamma_i}\sigma_l^2 + min_{l\in\Gamma_i}\left\{Var(q_l)-\sigma_l^2\right\}$$

7. Calculate the upper bound of Π_i^p as $UB_i = \Phi\left(\frac{\delta-mean_i}{\sqrt{variance_i}}\right)$.

In the above algorithm, the operator *arg* used in relation $arg\left\{min_{k\in\Gamma_i}\{\dots\}\right\}$ return the index (operation) to the minimum value. If we consider only operations$k \in \Gamma_i$, $mean_i$ is a lower bound on the length of path including all of these operations. In order to find a better (larger) value for $mean_i$, we consider two situations, represented in steps 3 and 4, respectively. In step 3, we assume the operation having the minimum $E(h_k)$, has also the minimum value of $E(q_k)-\mu_k$. Thus, we consider the impact of this operation on calculation of $mean_i$ either in $E(h_k)$ or in $E(q_k)-\mu_k$. This idea has been implemented by definition of parameters $\alpha_i$and β_i. Moreover, in order to find UB_i, we must establish a lower bound for variance of the path including all operations $k \in \Gamma_i$. Similar to steps 3 and 4, we develop steps 5 and 6 and define parameters ω_i and γ_i to calculate $variance_i$.

After calculating UB_i for all $i = 1, \dots, m$ in each node of the search tree, we determine upper bound of the main problem as $UB = min\{UB_i; i = 1, \dots, m\}$. Now, we establish dominance rule 2.

***Dominance rule* 2**: In each node of the search tree, if $UB < LB$or $\Pi^p < LB$, then fathom the node.

Proof: Straightforward.

4- Computational results

4-1- Computational setup

All the procedures were coded in Visual C++ 2013; all computational experiments were performed on a computer with Intel® Core™ i7-4790k CPU @ 4.00 GHz processor, 32 GB of internal memory

and a 64-bit operating system. In order to evaluate the performance of the developed B&B algorithm, four values for the number of jobs, $n = 10, 20, 30$ and 40 and three values for the number of machines, $m = 3, 4$ and 5, have been considered. For each operation *k*, the value of μ_k is drawn from a uniform distribution $U[5,99]$ and the value of σ_k^2 is drawn from a uniform distribution on interval $\left(0, 0.1\mu_j^2\right]$. The uniform distribution for σ_k^2 is chosen so as to obtain non-negative processing times in virtually all cases. Five instances are generated for each combination of *n* and *m*, leading to 60 test instances in total. The due date is set to the makespan obtained from the well-known *shifting bottleneck method* (Adams et al., 1988) in which μ_k is considered as the deterministic processing time of operation *k*.

4-2- Summary results

In this section, the total run time of the B&B algorithm, referred as T_{Total}, is reported in Table 2. Each cell of this table indicates the average T_{Total} over five instances for the corresponding *n* and *m*. Also, in the last row and last column, the average T_{Total} is presented for each value of *n* and *m*. All times are expressed in seconds.

Table 2. Average of T_{Total} for different values of *n* and *m*

m \ *n*	10	20	30	40	*Avg.*
3	0.01	0.70	0.83	83.21	21.19
4	0.11	1.12	95.80	755.62	213.16
5	0.17	6.82	337.30	2181.11	631.35
Avg.	0.10	2.88	144.64	1006.65	288.5

As expected, we observe an increase in T_{Total} B&B algorithm when the number of jobs or machines is increased. The average of T_{Total} over all test instances has been around 288.5 seconds.

4-3-Results with time limits

In order to evaluate the performance of our developed B&B algorithm in short running times, we executed it for five time limits (*TL*), i.e. $TL = 1, 10, 100$ and 1000 seconds. For each time limit, we consider the average percentage of deviation (*APD*) of the best obtained solutions from the best found solutions so far and the number of best solutions found (#*best*). The results of Table 3 indicate that the performance of the B&B algorithm is improved by increasing the time limit.

Table 3. Performance of the B&B algorithm in limited times

TL=	1	10	100	1000
APD	19.5	15.3	7.1	3.7
#*best*	35	42	48	51

4-4- Performance of the heuristic procedure

The performance of the heuristic procedure, developed for the initial solution, is evaluated in Table 4. Each cell of this table contains the *APD* between the solutions produced by the heuristic procedure and the best found solutions so far. The total *APD* is almost 55% which shows the efficiency of this procedure. Maybe it is expected that deviation will be increased with increasing number of jobs and machines but there are some exceptions in the reported results in Table 4 which is due to the non-optimal solutions. The running times for all test instances by the heuristic procedure are almost zero.

Table 4. Performance of the heuristic procedure

m \ *n*	10	20	30	40
3	45.1	66.1	18.7	39.2
4	75.8	44.2	57.4	52.9
5	77.8	53.6	60.4	68.9

4-5- Comparative results

If we have a linear objective function instead of Π, we will be able to run the binary model, developed in section 2, using CPLEX and compare its performance with the B&B algorithm. For this purpose, we consider$Max\ \Pi' = -\lambda_{m+1,n+1} - \theta^2_{m+1,n+1}$ instead of$Max\ \Pi$in the objective function. We should confess that Π' is not a proper linear approximation for Π and there are some errors in this conversion, but we do this just to run the developed model using CPLEX. Also, we simply change the other components of the B&B algorithm based on the new objective function Π'. Moreover, it is obvious that when we consider Π' instead of Π, the dominance rule 2 will not cut the optimal solutions.

Based upon the a fore mentioned assumption and considering previous test instances, we run the developed binary model using CPLEX 12.3 and compare its results with the results obtained from B&B algorithm in Table 5. Each cell of this table indicates T_{Total} based on seconds.

Table 5. Comparative results of B&B algorithm and CPLEX

	B&B				CPLEX			
m \ *n*	10	20	30	40	10	20	30	40
3	0.00	0.53	0.64	71.94	0.01	0.92	1.02	118.01
4	0.08	0.86	74.27	643.12	0.12	1.24	150.49	909.72
5	0.11	5.07	274.19	1945.63	0.18	8.83	449.37	3218.97

The results of Table 6 reveal that B&B outperforms the CPLEX such that the average T_{Total} over all test instances for B&B and CPLEX are 251.37 seconds and 404.91 seconds, respectively.

4-6- Impact of the dominance rules

In order to evaluate the impact of the dominance rules, we run our B&B algorithm in which these rules are excluded. We define algorithms $B\&B^{-(1,2)}$, $B\&B^{-(1)}$and $B\&B^{-(2)}$ respectively as the B&B algorithm in which both dominance rules, dominance rule 1 and dominance rule 2 are excluded. . By considering four already mentioned time limits, Table 6 indicates the results in terms of *APD.*

Table 6. Impact of the dominance rules

	Time limits			
	1	10	100	1000
B&B	19.5	15.3	7.1	3.7
$B\&B^{-(1,2)}$	64.3	51.0	28.7	15.74
$B\&B^{-(1)}$	41.8	33.3	18.6	10.0
$B\&B^{-(2)}$	29.4	23.3	12.8	7.6

Comparing the results of B&$B$with B&$B^{-(1.2)}$, B&$B^{-(1)}$ and B&$B^{-(2)}$, we find that the average *APD* over all time limits is respectively around 3.5, 2.3 and 1.6 times worse than B&B. These results also indicate that dominance rule1 that cut infeasible solutions has more impact on the efficiency of the developed B&B algorithm.

5- Conclusions and outlook on future work

In this paper, the β-robust job shop scheduling problem was studied, which aims to maximize service level in a job shop environment. A branch-and-bound algorithm accompanied with a set of dominance rules was developed to solve the problem. It is able to solve problem with size of 40 activities in reasonable times.

As future research opportunities, we suggest finding efficient dominance rules which do not cut optimal solutions. Also, developing more capable dominance rules can increase the efficiency of algorithm. The development of other exact, heuristic or meta-heuristic procedures for the βRJSSP may also be interesting research topics.

References

Adams J, Balas E and Zawack D (1988). The shifting bottleneck procedure for job-shop scheduling. Management Science 34: 391-401.

Alidaee B and Panwalkar SS (1993). Single stage minimumabsolute lateness problem with a common due date on non-identicalmachines. Journal of the Operational ResearchSociety 44: 29–36.

Applegate D and Cook W (1991). A computational study of job-shop scheduling. ORSA Journal of Computing 3: 149-156.

Brucker P, Jurisch B and Kramer A (1994). The job-shop problem and immediate selection. Annals of Operations Research 50: 73-114.

Carlier J and Pinson E (1989). An algorithm for solving the job-shop problem. Management Science 35: 164-176.

Daniels RL and Carrillo JE (1997). Beta-Robust scheduling for single-machine systems with uncertain processing times. IIE Transactions 29: 977-985.

Golenko-Ginzburg D and Gonik A (2002). Optimal job-shop scheduling with random operations and cost objectives. International Journal of Production Economics 76: 147-154.

Graham RL, Lawler EL, Lenstra JK and RinnooyKan AHG (1979). Optimization and approximation in deterministic sequencing and scheduling theory: a survey. Annals of Discrete Mathematics 5: 287-326.

Lenstra JK, RinnooyKan AHG and Brucker P (1977). Complexity of machine scheduling problems. In: P.L. Hammer, E.L. Johnson, B.H. Korte& G.L. Nemhauser, Studies in integer programming (Vol. 1, pp. 343-362). North-Holland, Amsterdam: Annals of Discrete Mathematics.

Lin S W, Chou S Y and Ying K C (2007). A sequential exchange approach for minimizing earliness–tardiness penalties of single-machine scheduling with a common due date. European Journal of Operational Research 177: 1294-1301.

Luh PB, Cheng D and Thakur LS (1999). An effective approach for job shop scheduling with uncertain processing requirements. IEEE Transactions on Robotics and Automation 15: 328–339.

Malcolm DG, Roseboom JH, Clark CE ,Fazar W (1959). Application of a Technique for Research and Development Program Evaluation. Operations Research 7(5): 646-669.

Pardalos P, Shylo O, Vazacopoulos A (2010). Solving job shop scheduling problems utilizing the properties of backbone and “big valley”. Computational Optimization and Applications 47: 61-76.

Pinedo ML (2014). Scheduling: theory, algorithms, and systems. Springer.

Ranjbar M and NajafianRazavi M (2012). A hybrid metaheuristic for concurrent layout and scheduling problem in a job shop environment. Advanced Manufacturing Technology 62: 1249-1260.

Ranjbar M, Davari M and Leus R (2012a). Two branch-and-bound algorithms for the robust parallel machine scheduling problem. Computers & Operations Research 39: 1652-1660.

Ranjbar M, Khalilzadeh M, Kianfar F, Etminani K (2012b). An optimal procedure for minimizing total weighted resource tardiness penalty costs in the resource-constrained project scheduling problem. Computers & Industrial Engineering 62: 264-270.

Roy B and Sussmann B (1964). Les problemesd'ordonnancement avec contraintesdisjonctives. In: Note D. S. 9. Paris: SEMA.

Sarin SC, Nagarajan B, Liao L (2014). Stochastic scheduling: expectation-variance analysis of a schedule. Cambridge University Press.

Singer M and Pinedo M (1998). A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. IIE Transactions 30: 109-118.

Spanos AC, Ponis ST, Tatsiopoulos IP, Christou IT and Rokou E (2014). A new hybrid parallel genetic algorithm for the job-shop scheduling problem. International Transaction in Operational Research 21: 479-499.

Tavakkoli-Moghaddam R, Jolai F, Vaziri F, Ahmed PK and Azaron A (2005). A hybrid method for solving stochastic job shop scheduling problem. Applied Mathematics and Computation 170: 185-206.

Tuong NH, Soukhal A, Billaut JC (2010). A new dynamic programming formulation for scheduling independent tasks with common due date on parallel machines. European Journal of Operational Research 202: 646-653.

Wu CW, Brown KN and Beck CJ (2009). Scheduling with uncertain durations: modeling beta-robust scheduling with constraints. Computers & Operations Research 36: 2348-2356.

Yin Y, Cheng TCE, Cheng SR, Wu CC (2013). Single-machine batch delivery scheduling with an assignable common due date and controllable processing times. Computers & Industrial Engineering 65: 652-662.

Zhang CY, Li PG, Rao YQ and Guan ZL (2008). A very fast TS/SA algorithm for the job shop scheduling problem. Computers & Operations Research 35: 282-294.