International Journal of Applied Operational Research
Vol. 5, No. 4, pp. 39-47, Autumn 2015

Journal homepage: ijorlu.liau.ac.ir

A New Branch-and-Bound for the Problem of Mean Completion Time for the Single Machine with Release Time

R. Rashid*, S. M. Seyedhoseini, A. Bozorgi Amiri

Received: 26 March 2015 ; **Accepted:** 15 August 2015

Abstract Preemptively scheduling a set of independent jobs with release time on one processor is a historical problem. In this paper, the same problem has been considered in which objective is to minimize the mean flow time. To prepare a perfect branch-and-bound, some optimality conditions have been represented and they have been compared with the optimality conditions which have been proposed before. Our branch-and-bound has been coded with C++, and the results indicate efficiency of our algorithm and our new optimality conditions.

Keywords: Single Machine, Release Time; Branch-And-Bound, Optimality Conditions.

1 Introduction

Scheduling is one of the most important planning issues, not only in manufacturing industries but also in the service industry [1]. Single-machine scheduling has been studied extensively with different objective functions. In this paper, we consider the problem of non-preemptively scheduling a set of N independent tasks, on one processor with the objective of minimizing the mean flow time. The basic elements considered in this paper, consist of a non-negative release time r_j, that indicating the earliest possible time that the job j can be processed, processing time T_j, and a completion time C_j. Also flow time of job j, F_j, is defined to be the time between release time and completion time of job j [2]. This problem without the non-preemptive constraint can be solved optimally by the shortest remaining time (SPRT) rule [3], also solution of this problem provides a lower bound for non-preemptively version of this problem which dominates all other lower bounds [4].

In the literature, there are some researchers who dealt with the same problem. For instance, Liu and Maccarthy 1991 [5], considered mean completion time for the single

* **Corresponding Author.** (✉)
E-mail: reza.rashid69@gmail.com (R. Rashid)

R. Rashid
Department of Industrial Engineering, Iran University of Science and Technology, Narmak 16844, Tehran, Iran

S. M. Seyedhoseini
Department of Industrial Engineering, Iran University of Science and Technology, Narmak 16844, Tehran, Iran

A. Bozorgi-Amiri
Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran

machine problem with release time, and proposed three heuristics. A MILP formulation and a branch-and-bound algorithm also have been represented to prepare optimal solution. Chu 1992 [6], considered total flow time in a same problem, and examined different aspects of the problem. Firstly, he examined heuristic algorithms and pervious optimality conditions, and then he extended some optimality conditions and a new branch-and-bound algorithm. Philips et al. 1998 [7], proposed a 2-approximation algorithm for the problem which has been called PSW algorithm.

Reeves 1995 [8], proposed a heuristic for the problem, and also his heuristic has been extended by a great tabu search. Chang et al. 2006 [9], considered the same problem and represent a case-injected genetic algorithm.

Croce and T'kindt 2003 [10], dealt with the problem of one- machine dynamic total completion time scheduling, and improved the preemptive bound for the problem. Lim and Yu 2004 [2], considered minimizing total flow time and experimentally studied how well, on average, the problem can be solved.

Choi et al. 2007 [11], dealt with single machine scheduling problems with resource dependent release times, and considered two problems. In the first problem, the objective was minimizing total resource consumption with a constraint on sum of job completion times. In the second problem, minimizing total resource consumption and sum of job completion times has been considered, where some optimality conditions have been develop for the problems and it has been demonstrated that the problem is polynomially solvable.

Nessah, Yalaoui et al. 2008 [12], presented a branch and bound algorithm by adding lower bounds and dominance properties for this problem objected to minimize sum of weighted completion times. Another branch and bound method presented by [13] which was for scheduling jobs with unequal release times with the objective of minimizing the sum of maximum earliness and tardiness. To get an upper bound they used modified dispatching rules based on different release times, also they proposed a procedure that considers preemption assumption to get a lower bound. Three years later, Nessah and Kacem, 2012 [14], proposed a branch-and-bound for the weighted completion time scheduling problem on a single machine.

In summary, it is clear that despite the many contributions in the $n/1/r_i/\sum_{i=1}^{n} C_i$ problems, there is little consideration due to optimality conditions. In this paper, we considered same problem, where the main contributions of this paper can be summarized as follows:

- We developed four new optimality conditions.
- We proposed a perfect heuristic algorithm.
- We represented a filtering algorithm and a new branch-and-bound algorithm.

The reminder of this paper is organized as follows: In section 2, firstly we represented our optimality conditions. A branch and bound algorithm has been proposed in section 3, and the branch-and-bound has been validated in section 4. We conclude our study in section 5.

2 Dominance properties

Let N denotes set of jobs that need to be scheduled, k denote a partial schedule, J(k) the set of jobs in this partial schedule, and φ_k denotes the completion time of the last job in k. $k \mid i$ is the new partial schedule obtained by adding the job i behind the given partial schedule k, and r_i denotes release time of job i. $\sum(k,i)$ is the schedule composed of $k \mid i$, completed by partial optimal schedule of jobs belonging to N-J($k \mid i$) starting from moment $\varphi(k \mid i)$ [11].

$R_i(\varphi(k))$ and $E_i(\varphi(k))$ denote the earliest beginning time and earliest completion time where job i can be scheduled after φ(k). r*(k) denotes earliest release time for set k. $\sum_k C$ denotes sum of completion time of set k. $s_j[N - J(k)]$ denotes number of jobs in set $N - J(k)$ which have less processing time than job j. k_A and $k_B$denote set of jobs scheduled in branch A and B. in this paper, Δ_i denotes completion time of the job which immediately preceding job i in schedule. If i is the first job of schedule, $\Delta_i = -\infty$.

In this section, number of dominance properties found in the literature have been reviewed, and also some dominance properties have been represented and proved.

Table 1. optimality conditions in previous studies

Theorem	Reference	Condition	Conclusion
1	[11]	$\exists j \in N - J(k): j \neq i; r_i \geq E_j(\varphi(k))$	$\sum(k,i)$ is dominated
2	[11]	$\exists j \in N - J(k): j \neq i;\ p_j \geq p_i;\ E_j(\varphi(k)) \leq E_i(\varphi(k))$	
3	[15]	$\exists j \in N - J(k): j \neq i;\ E_i(\varphi(k)) \geq E_j(\varphi(k));\ E_i(\varphi(k)) - E_j(\varphi(k)) \geq (p_i - p_j)[card(N - J(k) - 1]$	
4	[15]	$\exists j \in N - J(k): j \neq i;\ E_i(\varphi(k)) \leq E_j(\varphi(k));\ p_i - p_j \leq \left(E_i(\varphi(k)) - E_j(\varphi(k))\right) * card[(N - J(k)]$	
5	[11]	$\exists j \in J(k): j \neq i;\ E_i(\Delta_j) \leq E_j(\Delta_j);\ E_i(\Delta_j) - E_j(\Delta_j) \leq (p_i - p_j) * card[N - J(k)]$	
6	[11]	$\exists j \in J(k)\ in\ the\ kth\ position: p_i \geq p_j;\ p_i - p_j \geq (E_i(\Delta_j) - E_j(\Delta_j) * (card[J(k)] - k + 2)$	
7	[4]	There is another partial solution $\acute{k}$: $J(\acute{k})$ =J(k)∪{i}; $F(\acute{k}) \leq F(k\|i)$; $card[N - J(\acute{k})] * R_j(\varphi(\acute{k}))+ F(\acute{k}) \leq card[N - J(\acute{k})] * R_j(\varphi(k\|i)) + F(k\|i)$, where j is a job of N-J($\acute{k}$) with the smallest release date	

Considering reviewed properties, we presented some new dominance properties in 4 lemmas:

Lemma 1:

If k is a partial schedule and job j has minimum release time in set N-J(k),then only jobs which have below conditions could be nominated for next sequence.

1. $\varphi(k\ |i) \leq \varphi(k\ |j)$

Proof:

Considering SPT rule if job i has bigger processing time than j, job i could not be scheduled before j. if job i has less processing time, and $\varphi(k\ |i) \geq \varphi(k\ |j)$, then as described in Fig. 1, $\sum_{N-j(k\ |i)} C$ would be bigger than $\sum_{N-j(k\ |j)} C$, consequently equation 1 would be proved.

$$\sum_k C + \varphi(k\ |i) + \sum_{N-j(k\ |i)} C \geq \sum_k C + \varphi(k\ |j) + \sum_{N-j(k\ |j)} C \quad (1)$$

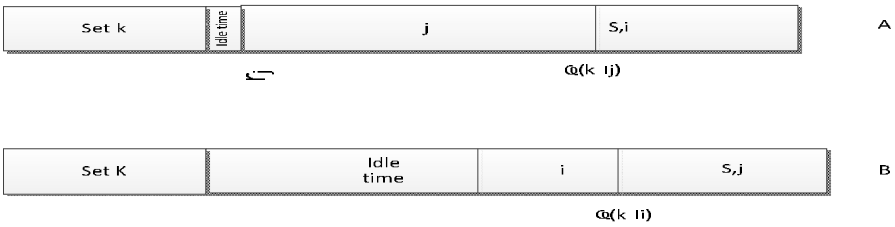

Fig. 1 comparison of two branches of A and B under lemma 1

Lemma 2.

If job j has minimum release time, then Job i is dominated if below condition be satisfied:

$$\text{p} \quad (2)$$

Proof.

As described in Fig 2, if the condition be satisfied schedule B is dominated against schedule A, Where schedule A may not be optimal. Jobs in G must have less processing time than job j, and number of unscheduled jobs in set $\mathrm{N}-\mathrm{J}(\mathrm{k})$ which have less processing time than J is $\mathrm{s_j}[\mathrm{N}-\mathrm{J}(\mathrm{k})]-1$. consequently, number of jobs in the set G is less than $\mathrm{s_j}[\mathrm{N}-\mathrm{J}(\mathrm{k})]-1$, and below expression is true.

$$\sum_{\mathrm{B}} \mathrm{C} - \sum_{\mathrm{A}} \mathrm{C} \geq \mathrm{p_j} - \mathrm{p_i} - \left(\mathrm{s_j}[\mathrm{N}-\mathrm{J}(\mathrm{k})]-1\right) * \left(\mathrm{C_j} - \mathrm{C_i}\right) \quad (3)$$

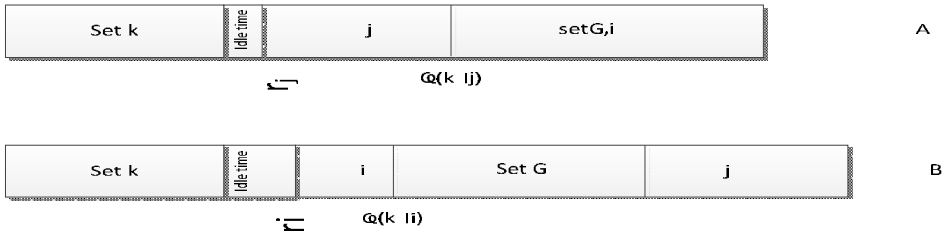

Fig. 2 comparison of two branches of A and B under lemma 2

Considering equation 3, if the condition be satisfied, then sum of completion time in schedule A is less than schedule B.

Lemma 3.

If job j has minimum release time, and $\mathrm{r_{ij}}$ be computed as follows:

$$r_{ij} = C_j - \frac{p_j - p_i}{\mathrm{s_j}[\mathrm{N}-\mathrm{J}(\mathrm{k})]-1} \quad (4)$$

Then i is dominated if below conditions satisfied:

- minimum release time for jobs of set {N-J(k)- (i U j)} be bigger than $\mathrm{r_{ij}}$
- $2(C_j - C_i) \leq p_j - p_i$

Proof.

Considering second condition, if job i and j be scheduled as described in Fig. 3, schedule B is dominated.

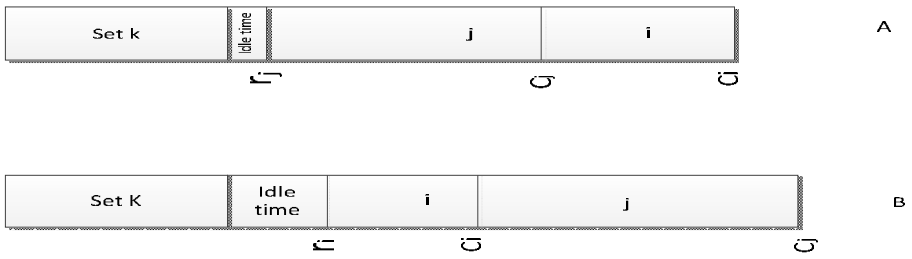

Fig. 3 comparison of two branches of A and B under lemma 3

And if some jobs scheduled between job i and j as described in Fig. 4, then number of them is less than $s_j[N - J(k)] - 1$, and their minimum release time is $r^*(G)$. Then if $r^*(G)$ is between r_{ij} and C_j, below equation would be satisfied.

$$\sum_{A} C - \sum_{B} C \leq p_j - p_i - \left(s_j[N - J(k)] - 1\right) * \left(C_j - r^*(G)\right) \leq 0 \tag{5}$$

And if the next job in schedule B released after C_j, then

$$\sum_{A} C - \sum_{B} C \leq 2\left(C_j - C_i\right) - p_j + p_i \leq 0 \tag{6}$$

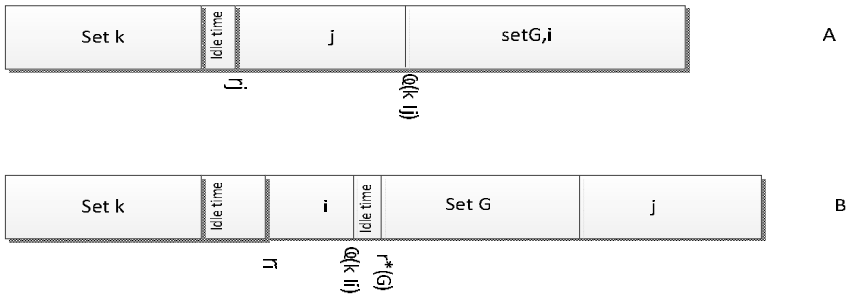

Fig. 4 comparison of two branches of A and B under lemma 3, when some jobs scheduled between i and j

Lemma 4.

If branch A and B have same number of scheduled jobs, then branch B can be eliminated if below conditions be satisfied.

1. $c_{max}^{B} \geq c_{max}^{A}$
2. $\sum_{k_B} C \geq \sum_{k_A} C$
3. Equation 7 must be satisfied, where j_e^A and j_e^B demonstrate e'th job in criteria of process time among jobs of branch A and B which are not scheduled yet, and $p_{j_e^A}$ and $p_{j_e^B}$ demonstrate processing time of them.

$$p_{j_e^A} \leq p_{j_e^B} \qquad \forall\, all\ i's \tag{7}$$

Proof.

The third condition states that, processing time of jobs in set (N-J(k_B) is at least equal to jobs in set (N-J(k_A)). considering condition 1 equation 8 would be emerged:

$$\sum_{(\mathrm{N-j}(k_A)} C \leq \sum_{(\mathrm{N-j}(k_B)} C \tag{8}$$

Considering condition 2 below expression would be emerged, which proves lemma 4.

$$\sum_{A} c \leq \sum_{B} c \tag{9}$$

Also it is clear that lemma 4 could be improved by replacing condition 1 with equation 10.

$$\begin{aligned} \sum_{k_B} C + \left(\sum_{\mathrm{N-j}(k_A)} \mathrm{p} - \sum_{\mathrm{N-j}(k_B)} \mathrm{p} \right) + \mathrm{c}^{\mathrm{B}}_{\max} * (card[N - J(k_B)] - 1) \\ \geq \sum_{k_A} C + \mathrm{c}^{\mathrm{A}}_{\max} * (card[N - J(k_A)] - 1) \end{aligned} \tag{10}$$

Where $\sum_{\mathrm{N-j}(k_B)} \mathrm{p}$ and $\sum_{\mathrm{N-j}(k_A)} \mathrm{p}$ are sum of processing time for jobs in branch B and A, that are not scheduled yet.

3 Filtering

In this section a heuristic has been prepared for filtering the data, where the main goal is to decompose the data and also determined some optimal schedules. Below algorithm prepares the basic sequences for filtering section.

- Step 1.Using lemma 1 choose the candidates for the next sequence
- Step 2. Considering lemma 2, 3 and 4 eliminate candidates which could be omitted.
- Step 3. Between remaining candidates, chose the job with minimum completion time

Lemma 5

If the jobs be scheduled with proposed heuristic, then two sets of k2 and $k1$ could be scheduled in depend if below condition be satisfied:

- $r^*_{k2} \geq \varphi_{k1}$

Where r^*_{k2} denotes the minimum release time of set k2.

Proof

When proposed heuristic be used, the maximum completion time of set $k1$ is bigger than maximum completion time of set $k1$ when optimal schedule prepared. Consequently, optimal schedule for set k1 would finish before r^*_{k2}, and it would be scheduled independent to set k2.

4 Branch and bound algorithm

In this paper we proposed a branch-and-bound algorithm, which uses the same scheme as the one used in (chengbin chu, 1992): During the computation, a list of unexplored nodes are kept, which are arranged in increasing order according to the lower bounds of nodes, with ties broken by a nonincreasing number of scheduled jobs. Each node represents a partial schedule which is also a partial list. This algorithm tries to develop the head of the list. Before any new node is created, some dominance properties are applied. For each node of search tree, which

cannot be eliminated by dominance properties, a lower bound is calculated, and if the lower bound is greater than or equal to the upper bound, this node is also eliminated.

This algorithm uses filtering stage to prepare an initial solution and calculate an upper bound for the problem, it also uses lemma 5 to prepare smaller independent sets of jobs which needs to be scheduled. The dominance properties used in this algorithm are theorem 7, lemma 1, 2,3, and 4. The lower bound used is LB-SPRT and for each node.

5 Computational results

In this section we prepared some numerical results in order to evaluate efficiency of the proposed branch-and-bound algorithm, and also we considered PB&B algorithm, the branch-and-bound with same procedure of our algorithm and optimality conditions which have been presented in table 1. We coded both of the algorithms in C++, and compared them in table 2. For this table, examples have been constructed randomly, where p_i is between $(0, 10)$ and four types have been presented four construction of release times, where the results indicate efficiency of our algorithm. In each size and each range of r_i five examples have been produced with the same procedure discussed.

Table 2 mean computational time(ms)

algorithm \ Size		10	20	40	60	80	120	160	200
$r_i \in (0, \frac{n}{4}.10)$	Our algorithm	780	2100	20310	55604	142523	291782	603842	2422990
	PB&B	978	3463	29409	78083	394781	843029	194019	-
$r_i \in (0, \frac{n}{2}.10)$	Our algorithm	792	2504	28003	89351	198620	377602	780422	3001388
	PB&B	1403	14352	47822	230141	740801	2891120	-	-
$r_i \in (0, \frac{3n}{4}.10)$	Our algorithm	977	3661	35110	144899	296146	422038	1106708	5687011
	PB&B	5233	33401	80872	644065	3796908	-	-	-
$r_i \in (0, n.10)$	Our algorithm	10934	28553	68114	187441	366499	601394	1654110	7920133
	PB&B	18388	104011	964330	4065055	-	-	-	-

To prepare a better comparison among optimality conditions, we considered a branch-and-bound algorithm with theorems of table 1, and 4 lemmas which have been proposed in this paper. Performance of each condition has been defined as follows:

$$performance\ of\ condition\ i = \frac{\sum_{k=1}^{N}(N-k)n_{ik}}{\sum_{k=1}^{N}\sum_{i=1}^{11}(N-k)n_{ik}} \tag{11}$$

Where n_{ik} is number of nodes which could be closed with condition i when k jobs have been planned. Equation 11 states that if a condition can close branches in higher levels it has better performance. Table 3 compares different optimality conditions for 5 random examples with size of 50 jobs for each range of r_i, where lemma 1 and 4 have more effectiveness then the other condition.

Table 3 a brief comparison among optimality conditions

Average performance for n=50	$r_i \in (0,\frac{n}{4}.10)$	$r_i \in (0,\frac{n}{4}.10)$	$r_i \in (0,\frac{n}{4}.10)$	$r_i \in (0,\frac{n}{4}.10)$
Theorem 1	.1	.12	.09	.08
Theorem 2	.33	.35	.33	.32
Theorem 3	.08	.06	0.06	.09
Theorem 4	.04	.05	.05	.06
Theorem 5	.09	.08	.06	.09
Theorem 6	.06	.07	.06	.07
Theorem 7	.19	.15	.15	.13
Lemma 1	.62	.63	.61	.57
Lemma 2	.12	.13	.12	.14
Lemma 3	.15	.17	.16	.17
Lemma 4	.45	.40	.48	.54

We also examined our heuristics performance, where same examples which have been produced for table 2, considered, and also optimality gap and computational time of the heuristic for each job size and range of release time has been analyzed in table 4. The results indicate high performance of the heuristic that mean optimality gap for all the examples is only 0.07.

Table 4 analysis of proposed heuristic, mean computation time (ms) and mean optimality gap

algorithm \ Size		10	20	40	60	80	120	160	200
$r_i \in (0,\frac{n}{4}.10)$	Optimality gap	.02	.02	.04	.03	.06	.06	.09	.13
	Computational time	632	1110	3137	5440	8185	10930	17282	18233
$r_i \in (0,\frac{n}{2}.10)$	Optimality gap	.04	.03	.05	. 10	.08	.08	.11	.12
	Computational time	510	985	3420	4738	8812	13475	16210	21156
$r_i \in (0,\frac{3n}{4}.10)$	Optimality gap	.01	.03	.04	.05	.07	.08	.12	.11
	Computational time	694	1321	2984	5913	7203	13537	17502	22580
$r_i \in (0,n.10)$	Optimality gap	.07	.06	.07	.09	.16	.12	.13	.19
	Computational time	819	1290	3530	5101	9244	12509	18055	21677

6 Conclusion

In this paper the single machine sequencing problem with different release time in which the objective is to minimize the mean flow time has been considered. By considering constraints for this problem, some optimality conditions have been developed, and they have been compared with the optimality conditions which have been represented in the previous studies. We also proposed a new branch-and-bound, a new heuristic, and a new filtering algorithm.

We coded our algorithm with C++ which could solve examples with size of 200 jobs. Performance of proposed algorithm has been evaluated for eight sizes and different ranges of release time, where computational time of algorithms became bigger when range of release times became bigger. Our algorithm had less computational time than the other algorithm which has been represented before. In addition, our algorithm was capable to solve examples with size of 200 for each range of release time, in which, the other algorithm has been failed.

In this paper, efficiency of our optimality conditions has been examined, where lemma 1 and lemma 2 have best performance among all conditions. We also evaluated performance of

the heuristic and the results indicated high performance of the heuristic that mean optimality gap for all the examples was only 0.07.

References

1. Pinedo, M. (1995). Scheduling: theory, algorithms, and systems. Englewood Cliff, NJ: Prentice Hall.
2. Gue, Y., Lim, A., Rodrigues, B., Yu, S., (2004). Minimizing total flow time in single machine environment with release time: an experimental analysis. Computers & Industrial Engineering 47, 123-140.
3. Baker, K. R. (1974). Introduction to sequencing and scheduling. New York: Wiley.
4. Ahmadi, Reza h., Baghchi, Uttarayan. (1990). Lower Bounds for single-Machine Scheduling Problems. Naval Research Logistics 37, 967-979.
5. Liu, Jiyn., Maccarthy, B. L. (1991). Effective heuristics for the single machine sequencing problem with ready times. Int. J. PROD. RES, Vol.29, No. 8, 1521-1533.
6. Chu, Chengbin. (1992). A Branch-and-Bound algorithm to Minimize Total Flow Time with Unequal Release Dates. Naval Research Logistics 39. 859-875.
7. Phillips, C., Stein, C. & Wein, J. (1998). Minimizing average completion time in the presence of release dates. Mathematical programing B, 82, 199-224.
8. Reeves, Colon., (1995). Heuristics for scheduling a single machine subject to unequal job release times. European Journal of operation research 80, 397-403.
9. Chang, Pei-Chann., Hsieh, Jih-Chang., Liu, Chen-Hao. (2006). A case-injected genetic algorithm for single machine scheduling problems with release time. Int. J. Production Economics 103, 551-564.
10. Croce, F. Della., T'kindt, V. (2003). Improving the preemptive bound for the one-machine dynamic total completion time scheduling problem. Operation Research Letters 31, 142-148.
11. Chu, Chengbin. (1992). A Branch-and-Bound algorithm to Minimize Total Flow Time with Unequal Release Dates. Naval Research Logistics 39. 859-875.
12. Nessah, R., Yalaoui, F., Chu, C. (2008).A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates. Computers & Operation Research 35, 1176-1190
13. Mahnam, Mehdi., Moslehi, Ghasem. (2009). A branch-and-bound algorithm for minimizing the sum of maximum earliness and tardiness with unequal release times. Engineering Optimization 41. 521-535.
14. Nessah, Rabia., Kacem, Imed. (2012). Branch-and-bound method for minimizing the weighted completion time scheduling problem on a single machine with release dates. Computers & Operation Research 39, 471-478.
15. Dessouky, M. I., Deogun, J. S. (1981). Sequencing Jobs with Unequal Ready times to Minimize Mean Flow Time. SIAM Journal of Computing 10, 192-202.
16. Bianco, L. Ricciardelli. S. (1982). Scheduling of a Single Machine to Minimize Total Weighted Completion Time Subject to Release Dates. Naval Research Logistics Quarterly 29, 151-167.
17. Choi, Byung-Cheon., Yoon, Suk-Hun., Chung, Sung-Jin. (2007). Single machine scheduling problems with resource dependent release times
18. Liu, Ming., Zheng, Feing., Chu, Chengbin., Xu, Yinfeng. (2012). Single-machine scheduling with past-sequence-dependent delivery times and release times. Information processing letters 112, 835-838.