Two-Job Shop Scheduling Problems with Availability Constraints

Riad Aggoune

Laboratoire des Technologies Industrielles
Centre de Recherche Public Henri Tudor
BP 144, L-4002 Esch-sur-Alzette, G.D. Luxembourg
Riad.Aggoune@tudor.lu

Abstract

This paper addresses the complexity of scheduling problems considering two jobs to schedule and availability constraints imposed on the machines. A polynomial algorithm called temporized geometric approach is first proposed for the minimization of the makespan, under the non-preemption constraint. Then, a generalization to the preemptive case is developed. These algorithms are extensions of the geometric approach, which allows solving the classical two-job scheduling problem.

Introduction

The classical scheduling literature commonly assumes that the machines are never unavailable during the planning horizon. However, in actual workshop, this assumption may turn false. In fact, one or several machines might become unavailable for processing jobs after a breakdown or when a preventive maintenance such as washing or control operations is scheduled.

This paper addresses the two-job shop scheduling problem under availability constraints. We consider the deterministic model of fixed and known in advance unavailability periods. Moreover we assume unless specified that the jobs are strictly non-preemptable. This means once an operation is started, its execution can be interrupted neither by an unavailability period, nor by another operation. The objective function is the minimization of the makespan.

As compared to the literature dedicated to classical scheduling problems, studies dealing with limited machine availability are very rare. Availability constraints have been firstly introduced in single machine (see Adiri et al. 1992) and parallel machines (Schmidt 1984, 1988) environments. Lee extensively investigated flow shop scheduling problems with two machines (Lee 1996, 1997 and 1999). In particular, the author defined the resumable, non-resumable and semi-resumable models. An operation is called resumable if it can be interrupted by an unavailability period and completed without penalty as soon as the machine becomes available again. If the part of the operation that has been processed before the unavailability period must be partially (resp. as a whole) re-executed, then the operation is called semi-resumable (resp. non-resumable). Recently, flow shop scheduling problems with two machines and resumable jobs have been treated in (Blazewicz et al. 2001), and (Kubiak et al. 2002). For more details, we refer to the survey of (Schmidt 2000), where existing methods for solving scheduling problems under availability constraints as well as complexity results are reviewed. To the best of our knowledge, there is no theoretical study in the scheduling literature for the two-job scheduling problems with availability constraints addressed in this paper.

The remainder of this paper is organized as follows. We first describe, in the following section, the considered problem. Secondly, the classical geometric approach is presented. It is a polynomial algorithm for solving classical scheduling problems with two jobs. Then, we propose an extension of the geometric approach, which allows dealing with unavailability periods of the machines. This extension, developed for the non-preemptive case, is finally generalized to the resumable model.

Problem Definition

The non-preemptive job shop scheduling problem with two jobs and availability constraints can be stated as follows:

- Two jobs J_1 and J_2 have to be processed on a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$.
- Each job J_i is composed by a linear sequence of n_i operations $\{O_{i1}, O_{i2}, \ldots, O_{in_i}\}$.
- Each machine M_j can perform at most one operation at a time and each operation O_{ij} needs one machine, during p_{ij} time-units.
- A maximum of K unavailability periods may occur on each machine. The starting times and durations of these tasks are known in advance and fixed.
- Operations are strictly non-preemptable, which means that their executions can be interrupted neither by unavailability periods, nor by other operations.
- The objective is to determine starting times of operations on each machine, such that the above constraints are satisfied and the makespan is minimized.

According to the terminology concerning the machines availability introduced in (Schmidt 2000), the studied problem can be denoted by J, NC_{win} | n = 2 | C_{max}, where NC_{win} means that non-availability periods are arbitrarily distributed on the machines.

Classical Geometric Approach

The geometric approach has been firstly introduced in (Akers and Friedman 1955). It consists in reducing the two-job shop scheduling problem in a shortest path one. Its exact complexity for the minimization of any regular criterion is stated in (Sotskov 1985) and (Brucker 1988). We describe in the following, the basic ideas of the geometric approach with the makespan criterion.

The problem J | n=2 | C_{max} can be represented into a two-dimension plane with obstacles, defined as follows:

- Each axis representing one job $J_i = \{O_{i1}, O_{i2}, \ldots, O_{in_i}\}$ is decomposed into n_i sub-intervals. The length L_{ij} of sub-interval I_{ij} is proportional to the processing time p_{ij} of operation O_{ij}.
- The rectangle defined by intervals I_{1j} and I_{2k} is an obstacle if the associated operations O_{1j} and O_{2k} share the same machine.
- The right and upper borders of the rectangle defined by the origin point O and the final point F, which corresponds to the completion of the two jobs, is the final obstacle.

A feasible solution of the scheduling problem is then a path going from the origin O to the point F. Such a path consists of horizontal, vertical and diagonal legs. A horizontal (resp. vertical) leg represents the exclusive progression of job J_1 (resp. J_2), whereas diagonal legs correspond to simultaneous executions of the two jobs. Moreover, any path must avoid the interior of the obstacles. This is due to the fact that two operations can not be executed simultaneously on the same machine and are not preemptable. The length L of a path, which is equal to the makespan of the associated schedule, is given by the formula:

$$L = L_{(Hor)} + L_{(Vert)} + L_{(Diag)} / \sqrt{2}, \quad (1)$$

where $L_{(Hor)}$ (resp. $L_{(Vert)}$, $L_{(Diag)}$) is the total length of horizontal (resp. vertical, diagonal) legs.

As a consequence, the search of the schedule that minimizes the makespan is equivalent to the search of the shortest path in the plane with obstacles. Figure 1 shows, in bold, the shortest path obtained for a two-job shop scheduling problem defined by the following linear sequences, where each number in brackets represents the processing time p_{ij} of job J_i on machine M_j:

- **J_1** : M_1 (1) M_2 (3) M_3 (3)
- **J_2** : M_2 (2) M_3 (2) M_1 (1)

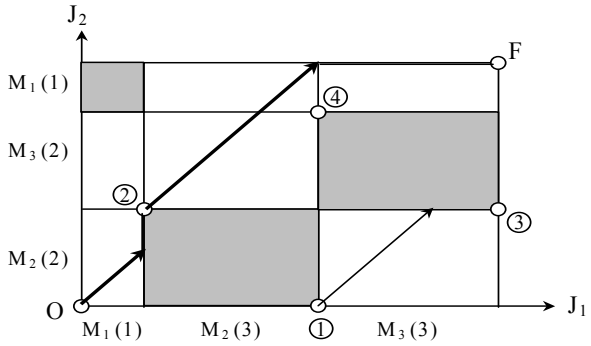
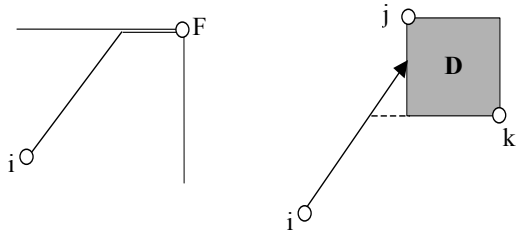


Figure 1: Shortest path in the plane

The search of a shortest path in the plane previously defined can be reduced to the search of the shortest path in an appropriate acyclic network N, this last problem being solvable in a linear time, in function of the number of obstacles. The network N = (V, E, d) is constructed as follows:

- V is the set of vertices, composed by the origin O, the final point F, and all the south-east (SE) and north-west (NW) corners of the obstacles.
- Each vertex $i \in V \setminus \{F\}$ coincides with at most two legs going from i. These legs are obtained by progressing diagonally (in the north-west direction) from i, until the border of the rectangle defined by O and F, or an obstacle D, is hit:
- In the first case, F is the unique successor of i and the leg (i, F) is composed by the path going diagonally from i to the final border and then going along it (see Figure 2.a).
- If an obstacle D is met, then two legs (i, j) and (i, k) are created, where j and k are the NW and SE corners of obstacle D, these two vertices being the direct successors of vertex i (Figure 2.b).
- The length d(i, j) of a leg (i, j) is equal to its horizontal or vertical part, to which is added the projection on one of the axis of its diagonal part.



(a) 1st case (b) 2nd case

Figure 2: Directs successors of a vertex i

An O-F path in this network corresponds to a feasible schedule for the problem J | n=2 | C_{max} and its length is equal to the makespan. So, finding the optimal makespan is equivalent to the search of a shortest path in the constructed network. Figure 3 represents the acyclic network associated with the shortest path provided by Figure 1.

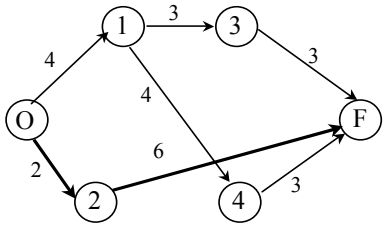


Figure 3: Acyclic network N

In (Brucker 1988), it was shown that the network N could be constructed in O(r log r) steps, where r is the number of obstacles, at most equal to $O(n_1 n_2)$. Moreover, the shortest path in an acyclic network can be found in a time proportional to O(r). So, the complexity of the problem $J \mid n=2 \mid C_{max}$ is at most equal to O(r log r). Since the works of Sotskov (1985), several generalizations of the method have been proposed in the literature. Applications of the geometric approach are reviewed in (Aggoune 2002).

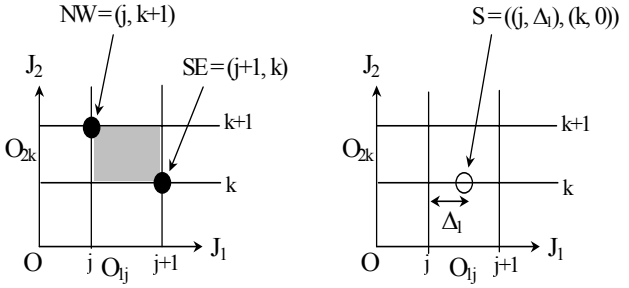
Temporized Geometric Approach

We develop in this section an extension of the geometric approach which exactly solve the problem J, $N_{Cwin} \mid n=2 \mid C_{max}$. This extension, called *temporized geometric approach*, allows to integrate the evolution of time and so the availability of the machines. It is based on the definition and the introduction of new vertices, as well as a new and dynamic way to progress from one vertex to its successors.

Vertices Characterization and Definitions

In the classical geometric approach, vertices of the network are the north-west (NW) and south-east (SE) corners of the obstacles hit when going diagonally in the plane. These corners are located at the extremities of the intervals corresponding to operations in conflict. Each vertex can then be defined thanks to its coordinates in the plane: the x-coordinate (resp. y-coordinate) of the vertex corresponds to operation of job J_1 (resp. J_2) to be executed (Figure 4.a).

In our approach, some vertices can be located between two lines bounding an operation that is to say inside intervals. So, we add, for each coordinate of the vertices, an additional information related to the duration already processed of the associated operation (Figure 4.b).



(a) classical characterization (b) new characterization

Figure 4: Vertices characterization

Furthermore, an *earliest date*, noted h(v), is combined with each vertex v obtained by a progression in the plane with obstacles. This value corresponds to the smallest duration which is necessary to reach vertex v starting from the origin O (note that h(O) = 0). The earliest date h(v) is then equal to the length of the shortest path going from O to v.

In order to take into account the unavailability periods of the machines, tests are realized at each consideration of a vertex, so as to verify if the machines needed for processing the next operations are available. Knowing the earliest date h(v) of vertex v, we try to process at the earliest, and if it is possible simultaneously, operations of the two jobs to be started. An *availability test* on direction J_1 (resp. J_2) consists in determining the availability time $T_{1,h(v)}$ (resp. $T_{2,h(v)}$) of the machine associated with the next operation of job J_1 (resp. J_2).

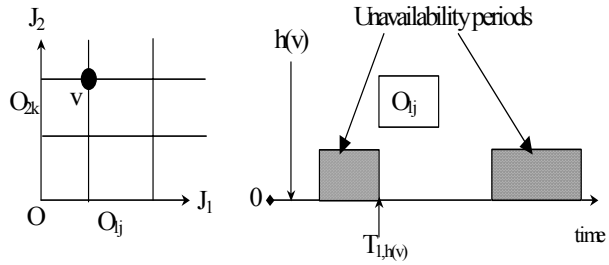


Figure 5: Availability test on direction J_1

In the example provided by Figure 5, operation O_{1j} can not be processed at time h(v), since it can not be completed before the first availability period of the machine. Operations being strictly non-preemptable, the execution of O_{1j} can start (at the earliest) at time $T_{1,h(v)}$ which is equal to the time in which the problematical availability period is finished.

We describe in what follows the principle of the temporized geometric approach, namely the set of vertices, the progression method from one vertex to its successors, and the way to calculate the distance between two vertices. Then, the construction of the network associated with the shortest path determination is developed.

Set of Vertices

The set of vertices V of the network constructed by our approach is composed by the three following kinds of vertices:

- *Regular* vertices, located at the intersection of horizontal and vertical lines, whose NW and SE corners of obstacles belong to. They correspond to the end of at least one operation.
- *Singular* vertices, located on a horizontal (resp. vertical) line, which means that the execution of the operation of job J_1 (resp. J_2) has already started (see Figure 4.b). They correspond to the fact that the execution of one job is going on, while the other is starting.

• *Waiting* vertices, also located at the intersection of two lines, for which the execution of operations of jobs J_1 and J_2 has not started yet.

A singular vertex is created if at the current time t, the progression of only one job is possible, whereas waiting vertices are created if the progression is possible for none of the two jobs. In the two-dimension plane, a waiting vertex is always a duplication of the regular vertex having the same geometric coordinates.

Progression Method

Let us consider a vertex v =((j, Δ_1), (k, Δ_2)), and its earliest date h(v). Availability tests are first realized on the operations of the two jobs to execute. According to these tests results, several ways to progress and to determine the successors of v are to be considered:

First Case. If the operations of the two jobs can not start at time h(v), vertex v is duplicated. A waiting vertex v_w = ((j, 0), (k, 0)) is then created, and v_w is the only successor of v.

Second Case. If there is an availability problem in the direction J_1 (resp. J_2), the progression is made along the vertical (resp. horizontal) line, that corresponds to only execute the operations of job J_2 (resp. J_1), until job J_1 (resp. J_2) becomes available again. A singular vertex s, from which a diagonal progression is possible, is added as the unique successor of v.

Third Case. If there is no availability problem, that is to say if the operations of the two jobs can be executed at time h(v), the progression works as in the classical geometric approach. It is made in the diagonal direction until an obstacle is hit. If this obstacle represents a machines conflict, its NW and SE corners are the two direct successors of vertex v. If the hit obstacle is the final one, point F is added as the unique successor of v.

Distance Between Two Vertices

Let v = ((j, Δ_1), (k, Δ_2)) be an arbitrary vertex that belongs to a path of the plane with obstacles, and let v'= ((j', Δ_1'), (k', Δ_2')) be one of its direct successors. If v' is a duplication of vertex v (see 1st case), then the distance between v and v' corresponds to the waiting time of the availability of one of the two jobs, which is given by the formula:

$$d(v, v') = \min \{T_{1v}, T_{2v}\} - h(v). \quad (2)$$

If v and v' have not the same geometric coordinates (Figure 6), let us suppose without loss of generality that they are located on horizontal lines, that is to say $\Delta_2 = \Delta_2' = 0$.

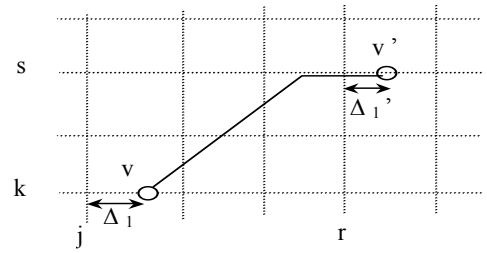


Figure 6: Distance between two vertices

In this case, $d(v, v') = \sum_{x=j}^{r-1} P_{1x} - \Delta_1 + \Delta_1'$, (3)

where $\sum_{x=j}^{r-1} P_{1x}$ is the distance between lines j and r.

Construction of the Network N = (V, E, d)

We have described in the previous parts the progression method from one vertex. Each time a new vertex v' (that is to say a successor of an explored vertex v) is generated, it is inserted in the set V of the vertices. A leg connecting the two vertices is created and the associated weight is equal to the distance d(v, v').

The initial vertex of network N is the origin point O. Its successors are generated, following the progression method, and added to the set V of vertices to be explored. Vertices of V are ordered according to their coordinates in the plane with obstacles.

More precisely, let us consider two vertices v = ((j, Δ_1), (k, Δ_2)) and v' = ((j', Δ_1'), (k', Δ_2')). Vertex v is inserted before v' in the set V if one of the following conditions is verified:

• (i): j < j' and k < k': v is located at the left and below v'.
• (ii): j < j' and k = k': v is at the left of v' on the plane.
• (iii): k < k' and j = j': v is below v' on the plane.
• (iv): j = j', k = k' and ($\Delta_1 < \Delta_1'$ or $\Delta_2 < \Delta_2'$): v is regular vertex and v' is a singular one.

The set V of vertices to be explored being partially ordered, the first vertex of V is the one selected at each step of the network construction. The explored vertex is then deleted from V and its successors are added to the set, if it is not already done. Note that this exploration method guarantees for every selected vertex, that its predecessors have been previously treated.

Shortest Path Determination

Before the exploration of a vertex v, its earliest date h(v) is calculated. This is done by finding the predecessor v_i of v such that the value $h_i = h(v_i) + d(v_i, v)$ is minimum, which is equivalent to determine the shortest path from origin O to vertex v.

The determination of shortest paths during the construction of the network has two advantages: First, it guarantees at each step the optimality of the used paths and

second, it avoids making a shortest path search after the network construction. In fact, when the progression is stopped at the final point F, the shortest path of the network is completely determined and its length (which is the optimal makespan of the two-job scheduling problem) is equal to h(F).

Complexity Results

The following algorithm *TGA* allows building network N = (V, E, d) associated with the shortest path determination.

Algorithm 1: *TGA*

Data:	Two jobs and unavailability periods of the machines
Result	Optimal schedule and associated makespan
Step 1:	*Initialization* - V = {O = ((1, 0), (1, 0))} */* set of vertices */* - E = ∅ */* set of created arcs */*

Step 2:	*Construction*
	While V ≠ {F} **Do**
	2.1. Select the first vertex s of V and compute h(s).
	2.2. Remove s from V : V = V \ {s}.
	2.3. Apply the progression method to obtain successors of s:
	For each successor v_i **Do**
	- Insert v_i in the sorted set V : V = V ∪ {v_i}.
	- Add arc (s, v_i) in E: E = E ∪ {(s, v_i)}.
	End For.
	End While.

Step 3:	The makespan value is given by the earliest date h(F) of F. The optimal schedule is obtained by a backward scanning.

Theorem 1. The set of vertices and the weights of arcs constructed by algorithm *TGA* are sufficient to determine a shortest path in the plane and so an optimal solution for the scheduling problem J, N_{Cwin} | n = 2 | C_{max}.

Proof. For sake of shortness, we only give the basic ideas of the proof. We refer to (Aggoune 2002) for the complete demonstration. It consists in showing that a shortest path in the plane with obstacles is only composed by series of arcs that belong to the network constructed by the algorithm.

Let C_{opt} = (v_0 = O, v_1, v_2,…, v_s = F) be the optimal path obtained by algorithm *TGA*, where v_i = (x_i, y_i, h(v_i)) and consider a shortest path *C*. We suppose without loss of generality that C is chosen so as the common part between C_{opt} and *C* is maximal. Let $v = v_k$ be the first point of C_{opt} such that arc (*v*, *v'*) does not belong to the network constructed by algorithm *TGA*, *v'* being the successor of *v* in path *C*. Note that in the worst-case *v* = O.

If k = s, then the proof is finished. Otherwise k < s, and arc (*v*, *v'*) (and eventually vertex *v'*) does not belong to the path constructed by the algorithm as illustrated in Figure 7.

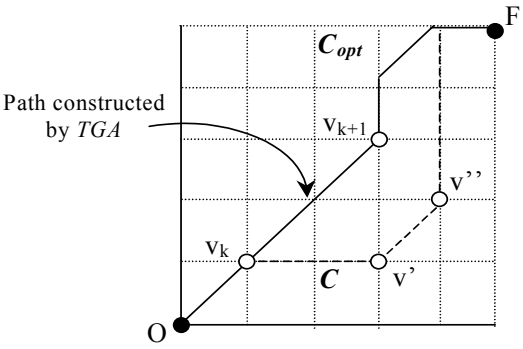


Figure 7: Shortest paths

In (Aggoune 2002), it is proved that, whatever nature of successor v_{k+1} of v_k in the network is, the shortest path *C* can be transformed in a path with a shorter or equal length, but having an additional arc of the network. This allows contradicting hypothesis on the maximality of index k. As a consequence v_k cannot be the first vertex of path C_{opt} such that arc (v_k, v') does not belong to the network obtained by our approach. By repeating this argument (considering v_{k+1} and its successor v'' in path *C*, etc.) we finally reach point F and we can conclude that a shortest path is only composed by series of arcs that belong to the network constructed by algorithm *TGA*. □

Theorem 2. The scheduling problem J, N_{Cwin} | n = 2 | C_{max} is polynomial and its complexity is at most equal to $O(Ks^4)$, where K is the maximum number for all the machines of the unavailability periods, and s = max {n_1, n_2}.

Proof. The determination of a shortest path in the network from the origin O to a vertex *v* being calculated before *v* is explored; the complexity of the approach depends on the time that is necessary to explore all the vertices.

The vertices of the network are the regular, the singular and the waiting ones. An upper bound on the number of regular vertices that can be generated is $O(n_1 \times n_2)$, which represents the number of obstacles. The number of waiting vertices that can be generated by the exploration defined by algorithm *TGA* is proportional to $O(n_1 \times n_2)$, representing the number of crossings between horizontal and vertical lines. However, to compute an upper bound on the singular vertices, we must determine the number of vertices generated from an arbitrary vertex v = ((x, Δ_1), (y, Δ_2)). Starting from v, and before reaching point F, the path goes in the worst case successively on a vertical and a horizontal line, and we can add at most one vertex, each time. So, vertex v can generate at most $(n_1 - x) + (n_2 - y)$ vertices, which is bounded by $(n_1 + n_2)$.

As a consequence, the cardinality of the set of vertices V is at most equal to $(n_1 \times n_2)(4 + n_1 + n_2)$. The number of

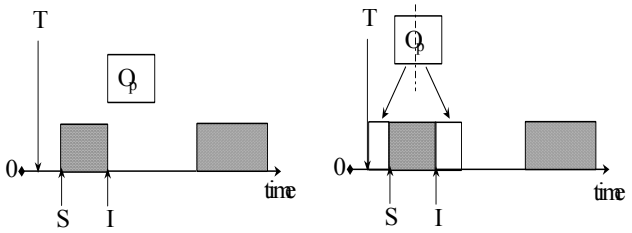
availability tests performed from a vertex is at most equal to $(n_1 + n_2) + K \max\{n_1, n_2\}$, the first term corresponding to the diagonal progressions, the second to horizontal and vertical ones. So, the number of elementary operations performed by the algorithm is at most equal to $(K s^4)$, where $s = \max\{n_1, n_2\}$. □

The Resumable Model

We focus in this sub-section on the case where operations are resumable (Lee 1996), that is to say they can be interrupted by unavailability periods and completed without penalties. We propose an extension of the temporized geometric approach to deal with this assumption.

The basic principles of the approach developed in the previous sub-section remains applicable to the resumable model. In fact, the non-preemption assumption is only considered in the progression method and more precisely during the availability tests. In the non-preemptive model, if at a given time T the machine, which must process an operation, is unavailable, then this operation can start at the earliest when the machine becomes available (see Figure 8.a).

In the resumable model, it is possible to start processing operation O_p from time T and until time S, then to achieve it from time I, as shown in Figure 8.b.



(a) non-preemptable operation (b) resumable operation

Figure 8: Availability tests

The problem that appears is then the systematic marking, in the plane representation of the scheduling problem, of the portions of operations executed before and after the availability periods.

To settle this situation, we propose to decompose each interval corresponding to an operation. More precisely, a grid is inserted on the classical plane representation, such that unit-length intervals are obtained. In this way, any interruption in the execution of operations can be modeled by adding new vertices and by changing the direction of the path representing the realization of jobs. Moreover, all the vertices are then located on at least one line of the new grid.

Example. The following Figure 9 represents a part of the plane with obstacles, namely the two first operations of two jobs J_1 and J_2. Let us consider operations O_{11} and O_{21}, which processing times are respectively equal to 5 and 6 units. The machine (not defined here) that must execute O_{11} is supposed to be unavailable between times 2 and 4, and the one that must process O_{21}, between times 4 and 5.

At time 0, the two operations can start and the progression is made in diagonal direction. Then, at time 2, the execution of O_{11} is interrupted during two time units and only O_{21} can continue, what corresponds to a vertical progression. The execution of O_{11} restarts at time 4 (there is a part of 3 units to produce left), but operation O_{21} must be interrupted during one time unit. The progression goes on the horizontal direction then starts again at time 5 in the diagonal direction. Finally, at time 7, the two first operations are finished and the path progression can continue with the next two operations.

As is the non-preemptive model, the number of vertices obtained by this progression method is bounded. In fact, from a regular vertex, we can generate at most $2 K s$ vertices before reaching point F. Moreover, the additional grid is only used to mark out the coordinates of the generated vertices. It does not make the number of vertices increase. The result is that:

Theorem 3. The scheduling problem J, N_{Cwin} | rs, n = 2 | C_{max} is polynomial and its complexity is at most equal to $O(Ks^4)$, where rs indicates that operations are resumable.

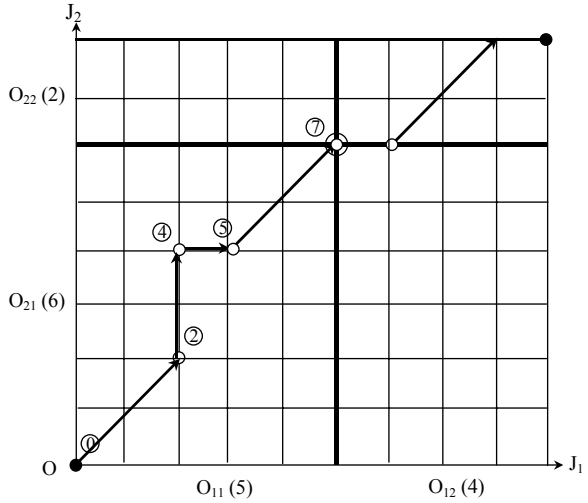


Figure 9: Shortest path with resumable operations

Conclusion and Perspectives

We have investigated in this paper two-job shop scheduling problems under availability constraints. The strictly non-preemptive problem has first been considered. After having recalled the classical geometric approach, we have proposed an extension of this approach to deal with the machines availability. This extension is based on a new characterization of the vertices, on the introduction of additional vertices, and mostly on the taking into account of time evolution during the scheduling procedure.

The developed algorithm, namely the temporized geometric approach, is polynomial, in function of numbers of operations and availability periods. It allows exactly solving the non-preemptive job shop scheduling problem

with two jobs and availability constraints. An extension of this algorithm to the resumable model was then described.

The proposed approaches can be used to develop quite powerful solution methods, exact as well as heuristic ones, for the general job shop scheduling problem with more than two jobs (Aggoune 2002).

As future research, we are studying a similar approach for solving scheduling problems under limited availability in the case where operations can be interrupted by availability periods and resumed after, with some penalties.

References

Adiri, I.., J. Bruno, E. Frostig, and , A. H. G. Rinnooy Kan (1989). Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, 26, 679-696.

Aggoune, R. (2002). Ordonnancement d'Ateliers sous Contraintes de Disponibilité des Machines. *Ph.D. Thesis*, Universite de Metz, France.

Akers, S. B. and J. Friedman (1955). A non-numerical approach to production scheduling problems. *Operations Research*, 3, 429-442.

Blazewicz, J., J. Breit, P. Formanowicz, W. Kubiak and G. Schmidt. (2001). Heuristic algorithms for the two-machine flowshop problem with limited machine availability. *Omega Journal*, 29, 599-608.

Brucker, P. (1988). An efficient algorithm for the job-shop problem with two jobs. *Computing*, 40, 353-359.

Kubiak, W., J. Blazewicz, P. Formanowicz, J. Breit. and G. Schmidt. (2002). Two-machine flow shops with limited machine availability. *European Journal of Operational Research*, 136, 528-540.

Lee, C. Y. (1996). Machine scheduling with an availability constraint. *Journal of Global Optimization*, 9, 395-416.

Lee, C. Y. (1997). Minimizing the makespan in two-machine flowshop scheduling problem with an availability constraint. *Operations Research Letters*, 20, 129-139.

Lee, C. Y. (1999). Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research*, 114, 420-429.

Schmidt, G. (1984). Scheduling on semi-identical processors. *Z. Oper. Res.*, A28, 153-162.

Schmidt, G. (1988). Scheduling independent tasks with deadlines on semi-identical processors. *Journal of Operational Research Society*, 39, 271-277.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 121, 1-15.

Sotskov, Y. N. (1985). Optimal servicing two jobs with a regular criterion, In: *Automation of Designing Processes*, Minsk, 86-95 (in Russian).