A Robustness Measure of the Configuration of Multi-Purpose Machines

André Rossi

A Robustness Measure of the Configuration of Multi-Purpose Machines

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2007-IJPR-0907.R2
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	02-Sep-2008
Complete List of Authors:	Rossi, Andre; European University of Brittany, Lab-STICC
Keywords:	SCHEDULING, PARALLEL MACHINES
Keywords (user):	ROBUSTNESS

A Robustness Measure of the Configuration of Multi-Purpose Machines

André Rossi
Lab-STICC, European University of Brittany,
Centre de Recherche, F-56321 Lorient, France
andre.rossi@univ-ubs.fr

Abstract

This paper presents new results for assessing the robustness of a configuration for multi-purpose machines. The workshop manager is provided quantitative and meaningful information on how a configuration behaves when disturbances affect the demand. Thus, configurations can be compared on the basis of temporal performance, but also on robustness. The robustness measure of a configuration is returned by assessing the minimum magnitude of disturbances affecting the forecast demand that may lead to breaking the deadline provided by the decision-maker. First, it is shown how to minimize the completion time for a given demand. The demands such that the deadline can be met for a given configuration are characterized afterward, and two robustness measures for a configuration are also provided. The theoretical results are illustrated in detail through an application example.

Keywords: Multi-purpose machines, Robustness, Scheduling, Configuration.

1 Introduction

Multi-purpose machines are often used to model industrial workshops composed of parallel machines subject to constraints which prevent the machines from processing all the product types [5]. The configuration of such workshops is a key choice as it has a significant impact on performance. However, determining a configuration in a real workshop is a complex issue because it also has an impact on its organization, flexibility and may involves costs in the whole production process.

This paper provides some new tools to assess the temporal performance of a given configuration, as well as two indicators of its robustness. While initially designed to address the problem of configuration in the field of semiconductor industry, the present work provides a theoretical framework that can help the decision-maker to quantitatively evaluate a configuration for any workshop modelled as a set of multi-purpose machines subject to deadline constraints and demand uncertainty.

In the semiconductor industry, the photolithography process is known to be a challenging step as it involves machines that are very expensive and subject to rapid obsolescence due to the constant advances in that field [1][18]. Moreover, the production flow is not easy to manage as products may have complex reentrant routes, and it is often observed that the photolithography workshop is a bottleneck. Finally, as the production processes are constantly updated to integrate new technologies, production volumes are hard to forecast because of unpredictable production losses. In such a context, robustness [11] [4] is a highly desirable feature for a photolithography workshop configuration, this workshop being modelled as a set of multi-purpose machines. Previous works have focused on finding robust configurations using a branch-and-bound approach [3] and integer linear programming [2]. However, these approaches have appeared not to be satisfactory to practitioners. First because they return a single solution to a strongly multi-criteria problem [15] with some criteria being difficult to model, and second because in such complex contexts, decision-makers do not want a computer program with low situation awareness to make high-stakes decisions for them. That is the reason why the main purpose of this paper is to provide the decision-maker with tools allowing for the measurement of the robustness of a given configuration, by assessing the minimum magnitude of disturbances that may lead to break a given deadline. Then, the decision-maker can perform "What If" studies on several configurations in order to choose the most appropriate ones or integrate these tools into a complete decision making process. It should be stressed that the entire decision process is out of the scope of this article, since the proposed results can be applied to address the configuration to any workshop. Although the proposed approach is not always practicable (for very large of sparse configuration matrices), it has been successfully tested on real-word instances in the field of photolithography (see [14]).

This paper is organized as follows. The configuration and the underlying scheduling problem are presented in section 2. As solving the underlying scheduling problem may lead to unbalanced production plans, section 3 focuses on the characterization of the demands for which a balanced production plan can be found for a given configuration. It also shows the limits of the proposed approach through computational experiments. Then, these results are used for characterizing

the demands that can be processed by a deadline in section 4. This characterization is the basis of the robustness measures presented in section 5, as it allows for the identification of the minimum magnitude of demand disturbances that may lead to breaking the deadline. The paper's contributions are illustrated through an application example that is dealt with in detail in section 6. Finally, the results of this work are discussed in the conclusion section.

2 Workshop configuration and the attached scheduling problem

2.1 Workshop configuration

The number of machines is denoted m, and n is the number of product types. The vector N models the demand, N_i being the quantity of products of type i that is to be processed by the workshop. The products have to be processed by one machine, provided the selected machine can process it. For some technological reasons, any machine cannot process any product type. The *technological matrix* is a boolean $n \times m$ matrix T, such that $T_{ij} = 1$ if and only if the products of type i could possibly be processed by machine j. The processing actually becomes possible if some resources are added to the machine, and if some settings are performed. In that case, the machine j is said to be *qualified* for the products of type i. The *configuration* of the workshop is a boolean $n \times m$ matrix Q, such that $Q_{ij} = 1$ if and only if the machine j is qualified for the product type i. A valid configuration must be such that $Q_{ij} \leq T_{ij}$ for all i in $I = [1, n]$ and for all j in $J = [1, m]$. Furthermore, there should be at least one machine qualified for each product type, and each machine should be qualified for at least one product type. In a photolithography workshop, the decision-maker has to design a configuration matrix periodically in order to fit the forecast demand (that is also updated periodically), by ensuring that the underlying scheduling problem has a solution which meets a given deadline, even when the actual demand differs from the forecast demand.

2.2 Underlying scheduling problem

The underlying scheduling problem involves the *speed matrix*, that is an $n \times m$ matrix V such that V_{ij} is the number of products of type i that machine j can process per unit of time (provided that $Q_{ij} = 1$). The machines are supposed to be uniform [6], i.e. V can be written as $V = VP \times VM$, where VP is a strictly positive $n \times 1$ column vector, and VM is a strictly positive $1 \times m$ row vector.

Indeed, machine j has a processing speed VM_j, that is the amount of products that machine j can process per unit of time. However, processing one unit of a product of type i may take longer than processing one unit of a product of type $i' \neq i$ on the same machine. Thus, VP_i is defined as the speed factor for the products of type i. By convention, $VP_1 = 1$: the product type 1 serves as a reference for expressing the speed factor of the other product types. More precisely, processing one unit of a product of type i can be achieved VP_i times faster than processing the same quantity of a product of type 1 on the same machine. Finally, machine j can process $V_{ij} = VP_i \times VM_j$ units of products of type i per unit of time for all (i, j) in $I \times J$. By construction, matrix V is a rank-one matrix: this characterizes uniform machines. Such a model is appropriate for most photolithography workshops.

Because the machines are very expensive and subject to rapid obsolescence, it makes sense to schedule the products to process so as to minimize the maximum completion time (this criterion is usually referred to as the *makespan* and denoted C_{max} [13]). The solution for such a problem is a value for R_{ij} for all (i, j) in $I \times J$, where R is a production plan. The numerical value for C_{max} is the problem objective, and can be deduced from R. More precisely, R_{ij} is equal to the time spent by machine j processing products of type i. Preemption and splitting are allowed in that problem, that can be referred to as $QMPM|split|C_{max}$ using the three-field notation [9]. This problem is polynomially solvable [2] and can be modelled as a linear program denoted LP.

$$(LP): \begin{cases} \text{Minimize} \quad C_{max} & & & (1) \\ \sum_{j \in J} Q_{ij} V_{ij} R_{ij} & = N_i & (\forall i \in I) & (2) \\ \sum_{i \in I} Q_{ij} R_{ij} & \leq C_{max} & (\forall j \in J) & (3) \\ R_{ij} & \geq 0 & (\forall i \in I)(\forall j \in J) & (4) \end{cases}$$

(1) is the objective function. The set of constraints (2) enforces that the demand is met for each product type. The set of constraints (3) enforces that the time workload of each machine is less than C_{max}. The set of constraints (4) ensures that the production plan is valid, as an amount of time cannot be negative. It should be stressed that R_{ij} are the decision variables while the makespan C_{max} is the problem objective. The configuration Q, the speed matrix V and the demand N are given.

LP makes possible the comparison between two configurations on the basis of temporal performance. Let $C_{max}(Q, N)$ be the optimal value for C_{max} after solving LP for configuration Q and demand N. Then, configuration Q_1 performs better than configuration Q_2 on demand N if and only if $C_{max}(Q_1, N)$ is smaller than $C_{max}(Q_2, N)$.

3 Characterization of load-balance

3.1 Balanced production plans

This section focuses on the characterization of the demands that are such that a balanced production plan can be found. Solving LP to optimality leads to one of the two following situations. The production plan R is either *balanced*, or not. More precisely, R is said to be balanced if all the machines share the same workload (i.e. $\sum_{i\in I} Q_{ij}R_{ij} = C_{max}$ for all j in J). In the rest of this paper, the workload of the machine j is denoted L_j, and $C(N)$, defined below, is a particular workload that will be proved to be equal to L_j for all j when the production plan is balanced (in [2] LP is shown to be a transportation problem [7] when a balanced production plan exists).

$$C(N) = \frac{\sum_{i\in I} \frac{N_i}{VP_i}}{\sum_{j\in J} VM_j}$$

Let us consider the following example with $m = 2$ machines and $n = 3$ product types. The machines are such that $V_{ij} = 1$ for all i and j (identical machines), the configuration matrix Q and the demands N^1 and N^2 defined below are considered.

$$V = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad N^1 = \begin{bmatrix} 3 \\ 7 \\ 2 \end{bmatrix}, \quad N^2 = \begin{bmatrix} 7 \\ 3 \\ 2 \end{bmatrix}$$

Solving LP to optimality for N^1 (respectively for N^2) leads to the production plans R^1 (respectively R^2). It should be stressed that the solutions of LP are generally not unique.

$$R^1 = \begin{bmatrix} 0 & 3 \\ 4 & 3 \\ 2 & 0 \end{bmatrix}, \quad C^1_{max} = C(N) = 6, \qquad R^2 = \begin{bmatrix} 0 & 7 \\ 3 & 0 \\ 2 & 0 \end{bmatrix}, \quad C^2_{max} = 7$$

First it can be seen that R^1 is balanced: the two machines have the same workload, that is equal to $C(N)$. Second, solving LP to optimality for N^2 leads to a non-balanced production plan R^2 with a greater makespan. This small example shows that because of non-qualifications, it is sometimes impossible to get a balanced production plan even when LP is solved to optimality. Roughly speaking, it can be said that the configuration *does not fit* the demand when no balanced production plan can be found, because the completion time is strictly greater than $C(N)$ while it could be equal to $C(N)$ for another configuration (the one defined by $Q_{ij} = 1$ for all i and j for instance).

The aim of this section is to characterize the demands that are such that a balanced production plan can be found for a given configuration. This is useful for assessing the robustness of a configuration as shown in section 5. To do so, the following lemma is used.

Lemma 1. *Let R be an admissible solution to LP. Then,*

- *There exists j_c in J such that $L_{j_c} \geq C(N)$*
- *There exists j_u in J such that $L_{j_u} \leq C(N)$*

Proof. The set of constraints (2) and (3) in LP can be rearranged as:

$$\begin{cases} \sum_{j\in J} Q_{ij}R_{ij}VM_j = \frac{N_i}{VP_i} & (\forall i \in I) \quad (2) \\ \sum_{i\in I} Q_{ij}R_{ij} = L_j \leq C_{max} & (\forall j \in J) \quad (3) \end{cases}$$

Multiplying each inequality of (3) by VM_j and summing them up yields to:

$$\sum_{j\in J} \Big(\sum_{i\in I} Q_{ij}R_{ij}VM_j \Big) = \sum_{j\in J} \big(VM_j L_j \big)$$

Using (2), the last equality can be written as:

$$\sum_{i\in I} \frac{N_i}{VP_i} = \sum_{j\in J} \big(VM_j L_j \big)$$

Dividing by $\sum_{j \in J} VM_j$ leads to the following equality, denoted (A):

$$C(N) = \frac{\sum_{j \in J} \left(VM_j L_j \right)}{\sum_{j \in J} VM_j} \qquad (A)$$

Lemma 1 is now shown by contradiction using equation (A). First, it is assumed that $L_j < C(N)$ for all j in J. Then, $L_j = C(N) - \alpha_j$ where $\alpha_j > 0$ for all j. Thus, equation (A) is equivalent to:

$$\sum_{j \in J} \alpha_j = 0$$

This is a contradiction, as α_j are strictly positive numbers for all j, and so, it is proved that there exists j_c in J such that $L_{j_c} \geq C(N)$.

Second, it is assumed that $L_j > C(N)$ for all j in J. Then, $L_j = C(N) + \alpha_j$ where $\alpha_j > 0$ for all j. Equation (A) again leads to a contradiction, proving that there exists j_u in J such that $L_{j_u} \leq C(N)$.

□

It can easily be shown that the configuration Q^* defined by $Q^*_{ij} = 1$ for all (i, j) in $I \times J$ is such that any demand N can be processed in $C(N)$ units of time, and is associated with a balanced production plan. As a matter of fact, non qualifications (i.e. zero-entries in the configuration matrix Q) are responsible for non-balanced production plans. Given a configuration Q, the set of demands for which a balanced production plan can be found is characterized by a set of inequalities, as shown in the theorem 3.1 in the next section.

3.2 Characterization

If no balanced production plan exists for N, *critical* machines are defined as follows. A machine is said to be critical if its workload is equal to C_{max} for any optimal production plan. For a given optimal production plan, if a machine happens to have a load equal to C_{max}, it does not necessarily mean that it is critical, as it may exist a different production plan (which is also an optimal solution to LP) for which this machine has a load that is strictly less than C_{max}. However if a machine has a load which is strictly less than C_{max}, then this machine is not critical. It should be mentioned that identifying the critical machines does not require the computation of all the optimal production plans. It can be achieved from any single optimal solution to LP, by using a polynomial algorithm that can be found in [14]. Furthermore, it can be noted that any product type processed by a critical machine cannot be processed by a non-critical machine. If it was the case, a critical machine would be able to transfer a non-zero quantity of product to a non-critical one, leading to a new production plan for which a critical machine would actually be non-critical which is a contradiction. For the same reason, any product type that can be processed either by critical or by non-critical machines has to be exclusively processed by non-critical machines for any production plan optimal for LP.

3.2.1 Example of critical machines

The following example aims at illustrating the definition of critical machines. The workshop configuration Q, the speed matrix $V = VP \times VM$ and the forecast demand N are given hereafter.

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \qquad VP = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \qquad VM = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \qquad N = \begin{bmatrix} 10 \\ 5 \\ 1 \\ 5 \end{bmatrix}$$

The following production plans R^1 and R^2 are two optimal solutions for LP, for which the makespan is $C_{max} = 10$.

$$R^1 = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \\ 0 & 5 & 0 \end{bmatrix}, \qquad R^2 = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 5 \end{bmatrix}$$

Production plan R^1 shows that machine 3 cannot be critical, since its load is equal to 1 (and so is strictly less than $C_{max} = 10$). Besides, R^2 also shows that machine 2 is not critical, since there exists an optimal solution to LP (namely R^2) for which machine 2 has a load which is strictly less than $C_{max} = 10$. It can be proved that machine 1 is critical

because it is the only machine qualified for a product type (namely product type 1) that requires 10 units of time to be processed. As a consequence, there does not exist any other production plan which is an optimal solution to LP, and such that machine 1 has a load that is strictly less than 10.

Theorem 3.1. *For a given configuration Q, there exists a balanced production plan for the demand N if and only if the following set of constraints is satisfied.*

$$(C_k) : \sum_{i \in I_k} \frac{N_i}{VP_i} \sum_{j \in J_k} VM_j \leq \sum_{i \in \bar{I}_k} \frac{N_i}{VP_i} \sum_{j \in \bar{J}_k} VM_j \quad (\forall k \in K)$$

In the above set of constraints, I_k is a non-empty strict subset of I and $\bar{I}_k$ is the complement of I_k in I. Analogously, J_k is a non-empty strict subset of J and $\bar{J}_k$ is the complement of J_k in J. For all k in K, I_k and J_k are such that $Q_{ij} = 0$ for all (i, j) in $I_k \times J_k$. The Cartesian product $I_k \times J_k$ is assumed to be a maximum rectangle of zeros in Q. A rectangle of zeros is said to be *maximum* if it is not included in another one, i.e. there does not exist $i \in \bar{I}_k$ such that $(i \cup I_k) \times J_k$ is a rectangle of zeros, and there does not exist $j \in \bar{J}_k$ such that $I_k \times (j \cup J_k)$ is a rectangle of zeros.

All the *maximum rectangles of zeros* (denoted MRZ for short) in Q are computed using an algorithm shown in section 3.4, derived from Nourine and Raynaud algorithm [12]. They are numbered from 1 to $|K|$, where $|K|$ can be up to $2^n - 2$ for $m = n$ when Q is the identity matrix. Although such an enumerative algorithm is obviously non-polynomial [8], it can be used for some real-life instances. Section 3.4 shows the limits the proposed approach.

The theorem 3.1 is shown below.

Proof. First, it is shown that if there exists a balanced production plan for N, then constraint (C_k) is satisfied for all k in K.

This implication is proved by contraposition, i.e. it is shown that if there exists k in K such that constraint (C_k) is violated, then no balanced production plan can be found for N.

As $I_k \times J_k$ is a maximum rectangle of zeros, all the products of type i in I_k are processed by some machines in $\bar{J}_k$. These machines process at least all the products of type i in I_k, as a consequence, lemma 1 ensures that there exists j_c in $\bar{J}_k$ such that:

$$L_{j_c} \geq \frac{\sum_{i \in I_k} \frac{N_i}{VP_i}}{\sum_{j \in \bar{J}_k} VM_j}$$

$I_k \times J_k$ is a maximum rectangle of zeros, which also means that the machines in J_k only process products of type i in $\bar{I}_k$. Then, lemma 1 ensures that there exists j_u in J_k such that:

$$L_{j_u} \leq \frac{\sum_{i \in \bar{I}_k} \frac{N_i}{VP_i}}{\sum_{j \in J_k} VM_j}$$

Since constraint (C_k) is violated, it can be written that:

$$L_{j_u} \leq \frac{\sum_{i \in \bar{I}_k} \frac{N_i}{VP_i}}{\sum_{j \in J_k} VM_j} < \frac{\sum_{i \in I_k} \frac{N_i}{VP_i}}{\sum_{j \in \bar{J}_k} VM_j} \leq L_{j_c}$$

This clearly shows that no balanced production plan can be found for the demand N.

Second, it is shown that if constraint (C_k) is satisfied for all k in K, then there exists a balanced production plan for N.

This implication is proved by contraposition, i.e. it is shown that if an optimal production plan for N is not balanced, then there exists k in K such that (C_k) is violated.

It is assumed that R is a non-balanced production plan for N that is also an optimal solution to LP. By definition, critical machines are processing products that non-critical machines cannot process (because they are not qualified for these product types). This means that $I_k \times J_k$ is a maximum rectangle of zeros in Q, where $\bar{J}_k$ is the set of the critical machines, I_k is the set of product types that are processed by these machines, and J_k is the set of the non-critical machines. Consequently, critical machines share the same load (this common load sets the value for C_{max})

$$L_j = \frac{\sum_{i \in I_k} \frac{N_i}{VP_i}}{\sum_{j \in \bar{J}_k} VM_j} = C_{max} \quad (\forall j \in \bar{J}_k)$$

Since there does not exist any balanced production plan for N, it can be written that $L_j \leq C_{max}$ for any non-critical machine j. For any j in J_k, multiplying L_j by VM_j and summing up all these inequalities yields to:

$$\sum_{j \in J_k} \left(VM_j L_j\right) < \sum_{j \in J_k} \left(VM_j C_{max}\right) \qquad (B)$$

The last inequality is strict because, as no balanced production plan can be found for N, there exists (at least) one machine j_u in J_k, the load of which is strictly less than C_{max}.

The machines j in J_k are not qualified for the products of type I_k. Consequently, $Q_{ij}R_{ij} = 0$ for all i in I_k. By definition of L_j, it can be written that:

$$L_j = \sum_{i \in I} Q_{ij}R_{ij} = \sum_{i \in \bar{I}_k} Q_{ij}R_{ij} \qquad (\forall j \in J_k)$$

The inequality (B) can then be written as follows:

$$\sum_{i \in \bar{I}_k} \left(\sum_{j \in J_k} Q_{ij}R_{ij}VM_j \right) < C_{max} \sum_{j \in J_k} VM_j$$

And so, the quantity $\sum_{j \in J_k} Q_{ij}R_{ij}VM_j$ can be replaced by $\frac{N_i}{VP_i}$ for all i, as specified by the set of constraints (2) in LP. Finally, replacing C_{max} by its value leads to:

$$\sum_{i \in \bar{I}_k} \frac{N_i}{VP_i} \sum_{j \in \bar{J}_k} VM_j < \sum_{i \in I_k} \frac{N_i}{VP_i} \sum_{j \in J_k} VM_j$$

This clearly shows that (C_k) is violated. □

3.3 Expressing C_{max} for unbalanced production plans

Knowing that $C_{max} = C(N)$ for any balanced production plan, this section shows that the makespan value can be computed analogously when the demand is such that any optimal solution to LP is not balanced.

Theorem 3.1 shows that if an optimal solution to LP is not balanced, then there exists a maximum rectangle of zeros in Q denoted $I_0 \times J_0$ such that the products of type in I_0 are processed by critical machines only. This maximum rectangle of zeros can be found by using a polynomial algorithm (see [14]). Furthermore the machines in J_0 only process the product types in I_0 and these product types are only processed by the machines in J_0.

The value of C_{max} only depends on the critical machines that share the same workload on the product types they are processing. This situation can be modelled as a subproblem of the original problem, where the considered product types are in I_0 and where the only considered machines are those in $\bar{J}_0$. This subproblem turns out to have a balanced production plan as the critical machines share the same workload. Thus, the makespan expression for a non-balanced production plan is:

$$C_{max} = \frac{\sum_{i \in I_0} \frac{N_i}{VP_i}}{\sum_{j \in \bar{J}_0} VM_j}$$

3.4 Computing the maximum rectangles of zeros

3.4.1 Proposed algorithm

The computation of all the MRZ for a given configuration is a necessary step for characterizing the demands that can be processed with a balanced production plan. It can be achieved using Algorithm 1, that is derived from Nourine and Raynaud algorithm [12].

input : Q, a $n \times m$ configuration matrix
output: I_k and J_k the h MRZ defined by $I_k \times J_k$ for all $k \in [1, h]$
Initialization
$h = 1$ is the number of MRZ currently found
$I_0 = \{1, 2, \ldots, n\}$
$J_0 = \{\emptyset\}$
for $b = 1$ **to** n **do**
  A $\leftarrow$ Set of column indexes of zeros entries located in the row b of matrix Q
  for $k = 0$ **to** $h - 1$ **do**
    Building a candidate $r = I' \times J'$
    $I' = A \cap I_k$
    $J' = J_k \cup \{b\}$
    Finding out if there exists $q \in [0, h-1]$ *such that* $I_q = I'$
    $q = 0$
    $stop = 0$
    while $stop == 0$ **do**
      if $I' == I_q$ **then**
        $J_q = J_q \cup J'$
        $stop = 1$
      else
        $q = q + 1$
      end
      if $q \geq h$ **then**
        $h = h + 1$
        $I_h = I'$
        $J_h = J'$
        $stop = 1$
      end
    end
  end
end
Delete I_0 and J_0

Algorithm 1: Enumerating the maximum rectangles of zeros in a configuration

This algorithm takes configuration Q as input, and aims at building all the MRZ in Q. The MRZ are found iteratively (it is an enumerative algorithm), and h is the number of MRZ currently found. Initially, h is set to one and a dummy MRZ denoted $I_0 \times J_0$ is built. This MRZ does not exist in practice (J_0 is the empty set) and will be deleted at the very end of the algorithm. Its purpose is to let actual MRZ being built (this step is called *Building a candidate* $r = I' \times J'$). The rows of Q serve as generators: element A (that is the set of indexes of zero elements in the row b of Q) is used to enlarge any preexisting MRZ r_k by adding machine b to J_k, and computing the intersection of I_k and A. Thus, the dummy MRZ plays the role of a "neutral element" allowing to build *candidates* from the rows of Q. The resulting rectangle of zeros $r = I' \times J'$ (the so-called candidate) is looked for in the list of existing MRZ. If I' already exists, the preexisting MRZ $I_q \times J_q$ is enlarged by including the machines of J' in J_q. If I' cannot be found in the preexisting MRZ, the list grows longer as the candidate becomes a new MRZ, leading h to be incremented by one. When the algorithm terminates, the dummy MRZ $I_0 \times J_0$ is deleted and $h = |K|$.

It must be stressed that Algorithm 1 is a simplified version of the C program actually implemented, as many technical improvements can be added to decrease computation time. The purpose of this algorithm is to provide the reader with an overview of the necessary steps to compute all the MRZ. Thus, it can be observed that since the list of rectangles of zeros has to be searched for each new candidate, finding a new MRZ takes more and more time as the list grows longer. If each candidate is to be added to the list, the number of MRZ can be up to $2^n - 2$ [14]. Furthermore, as Nourine and Raynaud [12] say that finding h (i.e. the total number of MRZ for a given configuration) without enumerating them all is an open problem, it is not easy to forecast the algorithm running time. The next section reports computational experiments allowing to characterize the configurations that can be searched for MRZ in a reasonable amount of time.

3.4.2 Computational results

Since Algorithm 1 may return a non-polynomial number of maximum rectangles of zeros, some experiments have been conducted for determining the limits of the approach. The tested instances are randomly generated configurations characterized by the number of product types n, the number of machines m and the matrix density d, that is the probability for every single entry Q_{ij} to be set to one (i.e. a matrix with a low density is sparse). Algorithm 1 is run on twenty instances randomly generated for each triplet (n, m, d). Tables 3.4.2, 3.4.2 and 3.4.2 show the average and maximum number of maximum rectangles of zeros (referred to as Avg. #MRZ and Max. #MRZ), as well as the average and maximum execution time in seconds for each couple (n, m), for $d = 0.25$, $d = 0.5$ and $d = 0.75$ respectively (referred to as Avg. CPU time and Max. CPU time in the Tables). The computer used is powered by a 3.2 GHz Intel® Pentium IV Processor with 1GB RAM under Microsoft® Windows XP. Algorithm 1 is implemented in C language and compiled with GCC.

Instances $n \times m$	Avg. #MRZ	Max. #MRZ	Avg. CPU time	Max. CPU time
15×30	4031	6683	0.274 s	0.484 s
16×32	6659.40	10058	0.685 s	1.390 s
17×34	9617	14496	1.730 s	4.749 s
18×36	16200.35	29593	6.972 s	22.478 s
19×38	25729	46072	27.676 s	99.691 s
20×40	33424.25	45814	51.126 s	106.565 s
21×42	55950.25	85555	140.927 s	267.348 s
22×44	93573.55	140500	438.947 s	1036.250 s
23×46	119337.35	208151	686.636 s	1492.820 s
24×48	192475.50	295362	2001.847 s	4911.350 s
25×50	-	-	> one hour	> one hour

Table 1: Maximum rectangles of zeros for low-density configurations ($d = 0.25$)

Instances $n \times m$	Avg. #MRZ	Max. #MRZ	Avg. CPU time	Max. CPU time
15×30	628.60	858	0.029 s	0.046 s
16×32	824.85	1051	0.050 s	0.078 s
17×34	1120.75	1723	0.064 s	0.109 s
18×36	1487.90	2044	0.097 s	0.156 s
19×38	2010.15	3271	0.158 s	0.297 s
20×40	2419.55	3658	0.212 s	0.422 s
21×42	3211.20	4325	0.337 s	0.517 s
22×44	3804.55	5491	0.453 s	0.877 s
23×46	4996.80	6692	0.726 s	1.284 s
24×48	6191.60	8531	1.047 s	2.098 s
25×50	8627.95	12335	2.096 s	3.801 s
26×52	9939.65	15525	3.027 s	8.747 s
27×54	12910.45	21516	5.630 s	18.255 s
28×56	14583.40	21009	7.982 s	19.569 s
29×58	18725.75	28465	15.454 s	40.599 s
30×60	23950.70	34265	28.252 s	69.967 s
31×62	26163.60	34319	36.464 s	64.189 s
32×64	32463.50	39123	62.021 s	96.086 s

Table 2: Maximum rectangles of zeros for mid-density configurations ($d = 0.50$)

n is greater than m without loss of generality because matrix Q and its transpose Q^T have the same rectangles of zeros ($I_k \times J_k$ has only to be replaced by $J_k \times I_k$). Computational results for instances with $n < 15$ are not reported because their solving time is very close to zero whatever configurations density. As expected, it can be observed that computation time increases significantly with instance size. Thus, Table 3.4.2 displays results only up to $n = 24$ because solving larger instances requires more than one hour of computation time. However, this only highlights a single aspect of the limits of the proposed approach as Tables 3.4.2 and 3.4.2 show that instance size is far from being the most important parameter impacting computation time. Density turns out to have an even more significant impact as the average computation time for instances of size 24×48 is around half and hour for $d = 0.25$ (sparse configurations) but only 0.03 s for $d = 0.75$ (dense configurations). This is due to the fact that there are far more maximum rectangles of zeros to be found in sparse configurations than in dense ones (about 192,000 for $d = 0.25$ and around 400 for $d = 0.75$ in the case of 24×48 instances).

Instances $n \times m$	Avg. #MRZ	Max. #MRZ	Avg. CPU time	Max. CPU time
15×30	101.30	122	0.00235 s	0.016 s
16×32	124.70	166	0.0008 s	0.016 s
17×34	136.45	171	0.00155 s	0.016 s
18×36	162.45	242	0.00715 s	0.016 s
19×38	204.05	270	0.00785 s	0.016 s
20×40	227.90	265	0.0102 s	0.016 s
21×42	278.05	371	0.015 s	0.031 s
22×44	311.65	417	0.01575 s	0.016 s
23×46	354	494	0.0205 s	0.032 s
24×48	406.15	501	0.0275 s	0.032 s
25×50	472	636	0.03145 s	0.048 s
26×52	505.35	657	0.0355 s	0.048 s
27×54	622.60	703	0.04895 s	0.063 s
28×56	689.40	904	0.0598 s	0.078 s
29×58	746.05	919	0.06385 s	0.094 s
30×60	879.60	1123	0.08525 s	0.110 s
31×62	1021.85	1340	0.10345 s	0.142 s
32×64	1071.80	1373	0.11595 s	0.174 s

Table 3: Maximum rectangles of zeros for high-density configurations ($d = 0.75$)

Finally, density and instance size (in a lesser extend) appear to be the most relevant parameters when considering running Algorithm 1 on a given configuration. The proposed approach is practicable regardless of density for instances having less than 20 product types, and up to 30 product types for dense and mid-dense configurations. This approach turned out to be applicable in the context of photolithography workshops, as the number of product types does not exceed 15 as far as the author knows.

4 Characterization of deadline meeting

In this section, the demands that can be processed by a deadline for a given configuration are shown to be characterized by a set of inequalities based on the maximum rectangles of zeros. This result provides an efficient way of assessing the robustness of a configuration as the minimum magnitude of disturbances on the forecast demand. In the rest of this paper, τ denotes a deadline that is common to the whole demand. Such an assumption on the deadline is realistic because the demand is often a set of products (i.e. a lot) that is expected to leave the workshop at a given date. In other words, the purpose of this section is to find the set of demands N such that the optimal value for C_{max} (returned by solving LP) is less than or equal to τ.

Theorem 4.1. *The three following propositions are equivalent:*

1. *The demand N can be processed in τ units of time or less*
2. *There exists a demand N' such that:*
   (a) $N' \geq N$
   (b) *N' can be processed in τ units of time or less*
   (c) *The production plan for N' is balanced*
3. *The demand N satisfies (S), the following set of inequalities:*

$$(S): \begin{cases} \sum_{i \in I} \frac{N_i}{VP_i} \leq \tau \sum_{j \in J} VM_j & (E_0) \\ \sum_{i \in I_k} \frac{N_i}{VP_i} \leq \tau \sum_{j \in \bar{J}_k} VM_j & (E_k) \quad (\forall k \in K) \end{cases}$$

Proof. This theorem is proved by showing that the three following implications (1) ⇒ (2), (2) ⇒ (3) and (3) ⇒ (1) hold.

First, it is shown that (1) ⇒ (2). It is assumed that R, a production plan for N meets the deadline τ. If R is balanced, then N' is set to N and the property (2) is true. If R is not balanced, then every machine having a load that is strictly less than τ can be loaded up to τ by increasing the quantity of products of any type for which the machine is qualified. By proceeding this way, a new demand N' satisfying $N' \geq N$ and the associated balanced production plan meeting the deadline τ are built.

Now, it is shown that (2) $\Rightarrow$ (3). Demand N' is supposed to be such that its corresponding production plan is balanced, and that it can be processed in τ units of time or less. According to the results presented in the previous section, it can be written that:

$$\frac{\sum_{i\in I}\frac{N'_i}{VP_i}}{\sum_{j\in \bar{J}} VM_j} \leq \tau$$

As $N \leq N'$ the above inequality also holds for N. Thus, the constraint denoted (E_0) in S is satisfied. Furthermore, as the production plan for N' is balanced, constraint (C_k) holds for all k:

$$(C_k) : \sum_{i\in I_k}\frac{N'_i}{VP_i}\sum_{j\in J_k} VM_j \leq \sum_{i\in \bar{I}_k}\frac{N'_i}{VP_i}\sum_{j\in \bar{J}_k} VM_j \quad (\forall k \in K)$$

Adding the quantity $\sum_{i\in I_k}\frac{N'_i}{VP_i}\sum_{j\in \bar{J}_k} VM_j$ to both sides of the above inequalities leads to:

$$\sum_{i\in I_k}\frac{N'_i}{VP_i}\sum_{j\in J} VM_j \leq \sum_{i\in I}\frac{N'_i}{VP_i}\sum_{j\in \bar{J}_k} VM_j \quad (\forall k \in K)$$

The right-hand side can be upper-bounded by $\tau\sum_{j\in J} VM_j \sum_{j\in \bar{J}_k} VM_j$ because it has been shown that $\sum_{i\in I}\frac{N'_i}{VP_i}$ is less than or equal to $\tau\sum_{j\in J} VM_j$, and then, by dividing the whole inequality by $\sum_{j\in J} VM_j$ (which is a strictly positive quantity), it yields to the new inequality:

$$\sum_{i\in I_k}\frac{N'_i}{VP_i} \leq \tau\sum_{j\in \bar{J}_k} VM_j \quad (\forall k \in K)$$

Then, as $N \leq N'$, the above inequality also holds for N, which is the expression of constraint (E_k) for all k in K.

Finally, it is shown that (3) $\Rightarrow$ (1). This implication is shown by contraposition, i.e. it is proved that N cannot be processed in less than τ units of time which implies that either constraint (E_0) is violated, or there exists k in K such that constraint (E_k) is violated.

Thus, it is now assumed that any optimal production plan for N is such that $C_{max} > \tau$.

If N is such that its production plan is balanced, then it can be deduced that:

$$C_{max} = \frac{\sum_{i\in I}\frac{N_i}{VP_i}}{\sum_{j\in J} VM_j} > \tau$$

This clearly shows that constraint (E_0) is violated.

If N is such that its production plan is not balanced, then it has been shown in section 3.3 that there exists a maximum rectangle of zeros (i.e. an integer k in K such that $I_k \times J_k$ is a maximum rectangle of zeros) such that:

$$C_{max} = \frac{\sum_{i\in I_k}\frac{N_i}{VP_i}}{\sum_{j\in \bar{J}_k} VM_j} > \tau$$

Thus constraint (E_k) is violated.

The theorem is shown by transitivity of equivalence. □

5 Assessing the robustness of a configuration

Even in the field of scheduling, the term robustness may have different meanings as shown in [4] and in [11]. Whereas its general meaning often refers to the ability to face disturbances or uncertainties, the application-dependent character of this term may not be surprising as the so-called disturbances and uncertainties may be very different from one problem to another, as well as the available means to face them, and the solution features that are relevant for protection from disturbances.

In this paper, deadline satisfaction is assumed to be a key issue. This makes sense in the context of semiconductor manufacturing because the photolithography workshop is often a bottleneck. As a consequence, the robustness of a configuration is defined as its ability to maintain the deadline satisfaction despite the disturbances that may affect the forecast demand. The present section aims at assessing the minimum magnitude of disturbances that may lead to breaking the deadline. The higher this magnitude is, the more robust the configuration is.

The characterization of the demands that are such that the configuration can meet the deadline is a useful tool to assess the robustness because constraints (E_0) and (E_k) for all k in K define the *borders* that should be watched to check if disturbances may compromise the ability to meet the deadline. To achieve this goal, it is necessary to evaluate the distance from any demand meeting the deadline to the borders of the characterized set. The distance is evaluated using the L1-norm because $|N' - N|_1$ returns the additional load between demands N and N' when $N' \geq N$.

5.1 Distance from a demand to the constraints' frontiers

5.1.1 Distance to the frontier of (E_0)

Demand N is on the frontier of constraint (E_0) if and only if N satisfies:

$$\sum_{i \in I} \frac{N_i}{VP_i} = \tau \sum_{j \in J} VM_j$$

Where Q, VP, VM and τ are given.

The following lemma provides a handy formula for assessing the distance from any demand N to the frontier of constraint (E_0). This is not a distance definition since the classical L1-norm distance is used.

Lemma 2. *The distance from any demand N to the frontier of constraint (E_0) is denoted $d(N, E_0)$ and has the following expression:*

$$d(N, E_0) = \Big(\tau \sum_{j \in J} VM_j - \sum_{i \in I} \frac{N_i}{VP_i}\Big) VP_{min}$$

Where i_0 is defined by $VP_{min} = \min_{i \in I}\{VP_i\}$

The proof is not provided here for the sake of concision, but it is analogous to the proof used to show a very close lemma that can be found in [14].

5.1.2 Distance to the frontier of (E_k)

For all k in K, demand N is on the frontier of constraint (E_k) if and only if N satisfies:

$$\sum_{i \in I_k} \frac{N_i}{VP_i} = \tau \sum_{j \in \bar{J}_k} VM_j$$

It is assumed that Q, VP, VM, k and τ are given. The distance from any demand N to the frontier of the constraint (E_k) is denoted $d(N, E_k)$ and has the following expression:

$$d(N, E_k) = \Big(\tau \sum_{j \in \bar{J}_k} VM_j - \sum_{i \in I_k} \frac{N_i}{VP_i}\Big) VP_{i_k} \quad (\forall k \in K)$$

Where i_k is defined by $VP_{i_k} = \min_{i \in I_k}\{VP_i\}$

The proof is not provided, but it is also derived from [14]. It is recalled that as the maximum rectangles of zeros cannot always be computed in a reasonable amount of time, assessing the distance from N to the frontier of all (E_k) may not always be practicable as shown in section 3.4.

5.2 Robustness measures

Because most workshops evolve in an uncertain environment, it is now considered that the actual demand $\tilde{N}$ that the workshop must process is not exactly the forecast demand N^*. As this change may affect the workshop ability to meet the deadline τ, it is relevant to assess the robustness of the configuration, i.e. its ability to guaranty that the workshop can meet the deadline even when $\tilde{N}$ is different from N^*.

More formally, the actual demand $\tilde{N}$ can be written as $\tilde{N} = N^* + \Delta N^+ - \Delta N^-$ where ΔN^+ and ΔN^- are nonnegative vectors satisfying $\Delta N_i^+ \times \Delta N_i^- = 0$ for all i in I. This means that ΔN_i^+ and ΔN_i^- cannot both be strictly positive. ΔN_i^+ is the additional quantity of product type i that is found in the actual demand $\tilde{N}$ and ΔN_i^- is the missing quantity of product type i in $\tilde{N}$. As an example, $N_1^* = 60$ and $\tilde{N}_1 = 72$ lead to $\Delta N_1^- = 0$ and $\Delta N_1^+ = 12$ because there are 12 more products of type 1 in $\tilde{N}$ than expected in N^*. Thus the equality $N_1^* = \tilde{N}_1 + \Delta N_1^+ - \Delta N_1^-$ holds. It would also hold with $\Delta N_1^- = 1$ and $\Delta N_1^+ = 13$ but this set of values is not valid as $\Delta N_1^+ \times \Delta N_1^- \neq 0$.

In the rest of this paper, it is assumed that the workshop can meet the deadline τ for the forecast demand N^*. In other words, τ is supposed to be greater than or equal to the optimal C_{max} value returned after solving LP for N^*. It can easily be shown that if $\tilde{N} \leq N^*$, the workshop ability to meet the deadline is not put in question. As a consequence, additional quantities of products are only taken into consideration when examining the differences between N^* and $\tilde{N}$. Thus, the robustness $r(Q, N^*, \tau)$ of a configuration Q is the minimum quantity of products added to the forecast demand N^* (regardless of product types) that *may* lead to missing the deadline τ. More formally, $r(Q, N^*, \tau)$ is the minimum distance from N^* to the frontier of (E_0) and of (E_k) for all k:

$$r(Q, N^*, \tau) = \min\left\{d(N^*, E_0), \min_{k \in K}\left\{d(N^*, E_k)\right\}\right\}$$

Thus, the robustness is defined as a distance. In that sense, this definition is very close to the Sotskov stability radius [16], however robustness is focused on additive perturbations only, as fewer products than expected cannot put in question the configuration ability to meet the deadline. Since the robustness measure is based on the maximum rectangles of zeros, its use may be unpracticable for some instances as shown in section 3.4.

Since Q is assumed to be such that a production plan meeting the deadline can be found for N^*, $r(Q, N^*, \tau)$ is nonnegative. If a configuration does not satisfies this assumption, it makes no sense to assess its robustness because it is not even suitable for the forecast demand. The above expression clearly shows that the robustness value is either set by (E_0), or by one or more constraints (E_k). These two cases have very different significations, as constraint (E_0) does not depend on the configuration, while (E_k) is derived from the maximum rectangles of zeros in Q.

This distinction is meaningful as it provides the decision-maker with a key piece of information. If $r(Q, N^*, \tau) = d(N^*, E_0)$, then modifying the configuration cannot lead to increase robustness. The full workshop potential for facing demand uncertainty is already used. However, if there exists k in K such that $d(N^*, E_k) < d(N^*, E_0)$, then the configuration limits the robustness that the machines may offer. As a matter of fact, the rectangle of zeros $I_k \times J_k$ prevents the configuration from offering more robustness. Thus, the decision-maker can consider qualifying a machine in J_k for a product type in I_k for increasing the configuration robustness. It may also be relevant to let him or her know the current configuration robustness potential $\rho(Q, N^*, \tau)$ defined as follows:

$$\rho(Q, N^*, \tau) = \frac{r(Q, N^*, \tau)}{d(N^*, E_0)}$$

It must be stressed that if $d(N^*, E_0)$ is zero, then $\rho(Q, N^*, \tau)$ is set to one by convention. Indeed, $d(N^*, E_0) = 0$ means that N^* is in the frontier of (E_0). The robustness potential is zero, but no other configuration could improve that result.

The robustness potential $\rho(Q, N^*, \tau)$ is in $[0, 1]$, and can be viewed as a fitness indicator. The higher it is, the more suitable the configuration is for facing uncertainties in the neighborhood of the forecast demand. $\rho(Q, N^*, \tau) = 1$ means that the configuration takes advantage of the full workshop potential for facing disturbances.

6 Application example

All the theoretical framework presented in this paper is illustrated in this section through a small instance. The considered workshop is made of $m = 4$ machines for processing $n = 5$ product types. The technological matrix is supposed to be the n-by-m all-ones matrix (i.e. machines are not subject to technological constraints). The workshop configuration Q, the speed matrix $V = VP \times VM$ and the forecast demand N^* are given hereafter.

$$Q = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \quad VP = \begin{bmatrix} 1 \\ 3 \\ 3 \\ 5 \\ 2 \end{bmatrix}, \quad VM = \begin{bmatrix} 2 & 1 & 1 & 3 \end{bmatrix}, \quad N^* = \begin{bmatrix} 135 \\ 22 \\ 79 \\ 86 \\ 43 \end{bmatrix}$$

6.1 Solving the underlying scheduling problem

Solving LP on the instance above leads to the following computational results for the production plan R and the completion time C_{max}, but it must be stressed that this solution is not unique:

$$R = \begin{bmatrix} 47.542 & 39.917 & 0 & 0 \\ 0 & 0.271 & 7.063 & 0 \\ 0 & 0 & 26.333 & 0 \\ 0 & 0 & 0 & 5.733 \\ 0 & 7.354 & 14.146 & 0 \end{bmatrix}, \qquad C_{max} = 47.542$$

These results have been computed using Matlab® with the `linprog` command, but without the `Simplex` option. Thus, Matlab® uses an interior point based method [10], that is expected to run faster than the Simplex method on large instances. However, since solvers have a finite precision, numerical errors may affect the results. It is the case here, where it has been deliberately decided to display the results with a decimal precision limited to three digits. Such an imprecision is not a problem because determining critical machines allows to compute the C_{max} value symbolically (with an infinite precision). It can be observed that the production plan is not balanced because machine 4 is underloaded whereas the other machines appear to be critical. Then, according to the results shown in section 3.3, the subproblem involving the product types that are not processed by machine 4 (i.e. $I_0 = \{1, 2, 3, 5\}$) and all the machines but machine 4 (i.e. $\bar{J}_0 = \{1, 2, 3\}$) has a balanced production plan, and a completion time having a value identical to those of the original problem.

$$C_{max} = \frac{\sum_{i \in I_0} \frac{N_i^*}{VP_i}}{\sum_{j \in \bar{J}_0} VM_j} = \frac{\frac{135}{1} + \frac{22}{3} + \frac{79}{3} + \frac{43}{2}}{2 + 1 + 1} = \frac{1141}{24} \approx 47.542$$

The C_{max} value returned by the solver cannot be exact as 1141 is not divisible by 3. However, computational approximations do not prevent one from accessing the symbolical value for C_{max}, provided that critical machines can be identified.

If the new configuration Q' defined below is used instead of Q, solving LP leads to the following results for N^*:

$$Q' = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \qquad R' = \begin{bmatrix} 0 & 80.667 & 54.333 & 0 \\ 1.128 & 0 & 0 & 1.692 \\ 0 & 0 & 26.333 & 0 \\ 2.646 & 0 & 0 & 3.969 \\ 0 & 0 & 0 & 7.167 \end{bmatrix}, \qquad C'_{max} = 80.667$$

It can be observed in R' that the machines 2 and 3 are critical. This is due to the fact that in the configuration Q', the product types 1 and 3 are only processable on these machines. Consequently, it can be deduced that $C'_{max} = \frac{242}{3} \approx 80.667$.

This clearly shows that the configuration Q performs better than Q' on the forecast demand N^* as $C_{max} < C'_{max}$. Since Q' has 11 qualifications (i.e. one-entries) and Q has only 10, this result also shows that temporal performance is not directly related to the number of qualifications in the configuration matrix.

6.2 Robustness assessment

The set of demands that can be processed in less than τ units of time with configuration Q can now be determined. To do so, theorem 4.1 is applied. As it requires the computation of the maximum rectangles of zeros in Q, Algorithm 1 is run, returning the seven following maximum rectangles of zeros:

$$\begin{aligned} r_1 &= \{2, 3\} \times \{1, 4\} \\ r_2 &= \{3, 4\} \times \{2\} \\ r_3 &= \{1, 4\} \times \{3\} \\ r_4 &= \{1, 2, 3, 5\} \times \{4\} \\ r_5 &= \{1\} \times \{3, 4\} \\ r_6 &= \{3\} \times \{1, 2, 4\} \\ r_7 &= \{4\} \times \{2, 3\} \end{aligned}$$

Thus, $K = [1..7]$, and the constraints (E_k) are given below for all k in K:

$$
\begin{array}{lrcl}
(E_1): & \frac{N_2}{VP_2}+\frac{N_3}{VP_3} & \leq & \tau\big(VM_2+VM_3\big)\\
(E_2): & \frac{N_3}{VP_3}+\frac{N_4}{VP_4} & \leq & \tau\big(VM_1+VM_3+VM_4\big)\\
(E_3): & \frac{N_1}{VP_1}+\frac{N_4}{VP_4} & \leq & \tau\big(VM_1+VM_2+VM_4\big)\\
(E_4): & \frac{N_1}{VP_1}+\frac{N_2}{VP_2}+\frac{N_3}{VP_3}+\frac{N_5}{VP_5} & \leq & \tau\big(VM_1+VM_2+VM_3\big)\\
(E_5): & \frac{N_1}{VP_1} & \leq & \tau\big(VM_1+VM_2\big)\\
(E_6): & \frac{N_3}{VP_3} & \leq & \tau VM_3\\
(E_7): & \frac{N_4}{VP_4} & \leq & \tau\big(VM_1+VM_4\big)
\end{array}
$$

Besides constraints (E_k) for all k in K depending on the configuration Q, constraint (E_0) must also be enforced:

$$(E_0): \frac{N_1}{VP_1}+\frac{N_2}{VP_2}+\frac{N_3}{VP_3}+\frac{N_4}{VP_4}+\frac{N_5}{VP_5} \leq \tau\big(VM_1+VM_2+VM_3+VM_4\big)$$

The set of demands that can be processed by τ units of time is denoted S in the statement of theorem 4.1 (it is recalled that S is made of constraints (E_0) and (E_k) for all k in K). By construction, S is a bounded intersection of a finite set of half spaces. Consequently, S is a polyhedron. As inequality (E_2) is equal to the sum of inequalities (E_6) and (E_7), it can be deduced that the proposed description for S is not minimal because some constraints may be redundant (where *redundant* should be understood as *not facet-defining*). Some techniques to get a minimal description of a polyhedron can be found in [17]. Adapting these techniques to the present work may lead to saving memory space when storing the maximum rectangles of zeros in a configuration. However, it would be far more desirable to embed these techniques into the generation of maximum rectangles of zeros algorithm, as many constraints appear to be redundant in some particular cases. Such an improvement would be both time and space saving.

In the present example, it can be checked that the forecast demand N^* does not satisfy all these constraints for $\tau = 45$, as it cannot be processed in less than 47.542 units of time: (E_4) is not satisfied for $\tau = 45$.

However, constraints (E_0) and (E_k) for all k in K are satisfied for $\tau = 50$. It that case, the distances from N^* to the frontier of these constraints are given below:

$$
\begin{array}{ll}
d(N^*, E_0) = 142.6 & d(N^*, E_1) = 199.0\\
d(N^*, E_2) = 769.4 & d(N^*, E_3) = 147.8\\
d(N^*, E_4) = 9.8 & d(N^*, E_5) = 15.0\\
d(N^*, E_6) = 71.0 & d(N^*, E_7) = 1164.0
\end{array}
$$

It can be observed that constraint (E_4) is the closest one to N^*. For that reason, the robustness of configuration Q is $r(Q, N^*, 50) = d(N^*, E_4) = 9.8$. This indicates that any demand N defined by $N = N^* + \Delta N^+ + \Delta N^-$ can be processed in less than 50 units of time, provided that $|\Delta N^+|_1 \leq 9.8$. The decision-maker is provided with a quantitative insight in the demand disturbances that the configuration can face while meeting the deadline. Besides, it should be stressed that this robustness measure relies on a worst-case scenario: some demands N defined by $N = N^* + \Delta N^+ + \Delta N^-$ with $|\Delta N^+|_1 > r(Q, N^*, \tau)$ that can be processed in less than τ units of time may exist. In the present example, it can easily be checked that $N^0 = N^* + \Delta N^+ + \Delta N^-$ with ΔN^+ and ΔN^- defined below is processable in less than 50 units of time, while $|\Delta N^+|_1 = 20$.

$$
\Delta N^+ = \begin{bmatrix} 0\\0\\0\\20\\0 \end{bmatrix}, \quad \Delta N^- = \begin{bmatrix} 0\\0\\0\\0\\0 \end{bmatrix}, \quad R^0 = \begin{bmatrix} 47.542 & 39.917 & 0 & 0\\ 0 & 0.27083 & 7.0625 & 0\\ 0 & 0 & 26.333 & 0\\ 0 & 0 & 0 & 7.0667\\ 0 & 7.3542 & 14.146 & 0 \end{bmatrix}, \quad C^0_{max} = 47.542
$$

R^0 and C^0_{max} have been computed by solving LP with N^0. In other words, $|\Delta N^+|_1 \leq 9.8$ is a sufficient (but not necessary) condition for ensuring that the actual demand can be processed in less than 50 units of time.

It has been shown in section 5.2 that assessing the robustness of a configuration that does not allow to meet the deadline for the forecast demand is not relevant. This is clearly the case for Q' since $C'_{max} = 80.667 > \tau$. Consequently, this configuration is not suitable for N^* (but Q is) and assessing its robustness is useless. However, this conclusion would be the opposite with the forecast demand $\bar{N}$ defined below.

$$\bar{N} = \begin{bmatrix} 58 \\ 297 \\ 24 \\ 66 \\ 90 \end{bmatrix}, \qquad C_{max}(Q, \bar{N}) = 53.3 > \tau, \qquad r(Q', \bar{N}, 50) = 34$$

Where $C_{max}(Q, \bar{N})$ is the optimal objective value for LP with configuration Q and demand $\bar{N}$. Thus, configuration Q' fits the forecast demand $\bar{N}$ whereas it is not the case for Q. This shows that robustness is not an intrinsic property of a configuration since it also depends on the forecast demand.

If the robustness offered by the current configuration is considered too low, the decision-maker may wish to add qualifications (i.e. replace some zero entries by one entries in Q). However, this can be a very costly operation (at least in the field of photolithography), and may not always be efficient for increasing robustness. In the present example, the configuration robustness potential value is $\rho(Q, N^*, \tau) = 6.9\%$. This means that Q is not really suitable for N^*, as only 6.9% of its robustness potential is currently used. Then, the decision-maker can expect far better results by introducing new qualifications in the current configuration (this work does not address the effective choice of such additional qualifications). More precisely, by choosing the configuration Q^* defined by $Q^*_{ij} = 1$ for all i and j, the robustness would be $r(Q^*, N^*, \tau) = d(N^*, E_0) = 142.6$. Thus, the robustness potential would be 100%. The comparison with Q^* is rather extreme, as Q^* is the most expensive configuration. This shows that the tools presented here provide useful information to the decision-maker that may wish to find the best possible tradeoff between configuration cost, performance and robustness.

7 Conclusion

The quantitative assessment of temporal performance along with robustness provides useful information to the decision-maker who has to take into account many parameters and criteria to update a configuration for multi-purpose machines, or to design a new one. The proposed approach, while not polynomial, turns out to be practicable for real-life instances, provided they are dense enough and not too large. This work must be completed by a sensitivity analysis in order to assess the completion time deviation for any given value for the magnitude disturbance on the forecast demand. This would provide a valuable piece of information to the decision-maker, allowing him or her to estimate the trend for the completion time that the workshop may offer under unexpected overload. Future works should also address the problem of choosing the new qualifications to add to a configuration (if necessary) for increasing temporal performance and/or robustness. This could be achieved through a decision-making process mixing these tools to a configuration generator using a branch-and bound approach or an integer programming model. Then, the model could be extended to account for the qualification cost, that may depend on the machines as well as on the product types.

8 Acknowledgments

The author would like to thank the anonymous referees whose suggestions greatly improved the quality of this manuscript, and Joanna Ropers for correcting English grammar and spelling.

References

[1] Akçali, E., Nemoto, K. and Uzsoy, R., 2001. Cycle-Time Improvements for Photolithography Process in Semiconductor Manufacturing, *IEEE Transaction on Semiconductor Manufacturing*, Vol. 14, No.1.

[2] Aubry, A., Rossi, A. and Jacomino, M., 2006. Minimizing setup costs for parallel multi-purpose machines under load-balancing constraint, *European Journal of Operational Research*. doi:10.1016/j.ejor.2006.05.050.

[3] Aubry, A., 2007 *Optimisation pour la configuration robuste de systèmes de production de biens et de services*, Grenoble, PhD Thesis.

[4] Billaut,J-C., Moukrim, A. and Sanlaville, E., 2007. *Flexibility and Robustness in Scheduling*. Paris: ISTE.

[5] Brucker, P., Jurisch, B. and Kramer, A., 1997. Complexity of scheduling problems with multi-purpose machines. *Annals of Operational Research* 70,5773.

[6] Brucker, P., 2004. *Scheduling Algorithms*. Springer.

[7] Chvátal, V., 1980. *Linear Programming*. New York: W. H. Freeman and Company.

[8] Garey, M.R., Johnson, D.S., 1979. *Computers And Intractability, A Guide To The Theory Of NP-Completness*. New York: W.H. Freeman and Company.

[9] Graham, R., Lawler, E., Lenstra, J. and Kan, A.R., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, vol. 5, 287-326.

[10] Karmarkar, N.,1984. A polynomial-time algorithm for linear programming. *Combinatorica*, vol. 4, 373-395.

[11] Kouvelis, P. and Yu, G., 1997. *Robust Discrete Optimisation and its Applications*. Dordrecht: Kluwer Academic Publisher.

[12] Nourine, L., and Raynaud, O., 1999. A fast algorithm for building lattices. *Information Processing Letters*, vol. 71, 199-204.

[13] Pinedo, M., 2005. *Planning and Scheduling in Manufacturing and Services*. New York: Springer Science.

[14] Rossi, A., 2003 *Ordonnancement en milieu incertain, mise en œuvre d'une démarche robuste*, Grenoble, PhD Thesis.

[15] Roy, B., 1994. On operational research and decision aid. *European Journal of Operational Research*, vol. 73, 23-26.

[16] Sotskov, Y., Wagelmans, A. and Werner F., 1998. On the Calculation of the Stability Radius of an Optimal or an Approximate Schedule. *Annals of Operations Research*, vol. 83, 231-252.

[17] Wolsey, L.A., 1998. *Integer Programming*. New York: John Wiley and Sons.

[18] Yoon, H.J. and Lee, D.Y., 2004. Deadlock-Free Scheduling of Photolithography Equipment in Semiconductor Fabrication. *IEEE Transaction on Semiconductor Manufacturing*, Vol. 17, No.1.