Response time variability

Albert Corominas, Wieslaw Kubiak, Natalia Moreno Palli

IOC-DT-P-2004-08
Maig 2004

Response Time Variability [1]

Albert Corominas [a], Wieslaw Kubiak[b], Natalia Moreno Palli[b]

[a] *Institut d'Organització i Control de Sistemes Industrials*
Universitat Politècnica de Catalunya
Barcelona, Spain

[b]*Faculty of Business Administration*
Memorial University of Newfoundland
St. John's, Canada

April 29, 2004

Abstract: This paper presents a work-in-progress on the response time variability problem. This problem occurs whenever events, jobs, clients or products need to be sequenced so as to minimize the variability of time they wait for their next turn in obtaining the necessary resources. The problem has numerous real-life applications, which will be briefly reviewed. The problem has distinctive number theoretic flavor. We study its computational complexity, present efficient, polynomial time algorithms for some cases and the NP-completeness proof for a general problem. We then propose a position exchange heuristic and apply it to improve the response time variability of an initial sequence. The latter is obtained in various ways: the optimum bottleneck sequence, Webster and Jefferson sequences of the apportionment, or randomly. We report on computational experiments with the heuristic.

1. Introduction

Most modern systems shares its resources between different jobs. The jobs define a certain amount of work to be done, for instance the file size to be transmitted to or from a server or the number of cars of a particular model to be produced on a mixed-model assembly line. To ensure fair sharing of common resources between different jobs, this work is divisible in atomic tasks, for instance data blocks or

[1]This research has been supported by the Natural Sciences and Engineering Research Council of Canada grant OPG0105675 and by the Ministerio de Ciencia y Technologia of Spain project DPI2001-2176.

cars. These tasks, in turn, are required to be evenly distributed so that the distance between any two consecutive tasks of the same job is as regular as possible, in other words, ideally constant. The following are some real-live examples.
The Asynchronous Transfer Mode (ATM) networks divide each application (voice, large data file, video) into *cells* of fixed size so that the application can be preempted after each cell. Furthermore, *isochronous* applications, for instance voice and video, require that a inter-cell distance in a cell stream be as close to being constant as possible and in the worst case not exceeding some pre-specified value. The latter is to account for limited resources shared with other applications, Han, Lin, and Hou [9], and Altman, Gaujal and Hordijk [1]. In fact multimedia systems should avoid presenting video frames too early or too late which would result in jagged motion perceptions.
On a mixed-model, just-in-time assembly line a sequences of different models to produce is sought where each model is distributed as "evenly" as possible but appears a given number of times to satisfy demand for different models. Consequently, shortages, on one hand, and excessive inventories, on the other, are reduced, Monden [14].
The stride scheduling is a deterministic scheduling technique where each client is first issued a number of *tickets*. The resources are then allocated in discrete time slices called *quanta*. The client to be allocated resources in next quantum is calculated as a certain function of the number of allocations obtained in the past and the number of tickets issued, Waldspurger, and Weihl [17]. This paper considers the throughput error and the response time variability as two main metrics of the schedule obtained.

These problems are often considered as the distance-constrained scheduling problems, where the temporal distance between any two consecutive executions of a task is not longer that a pre-specified distance, Han, Lin, and Hou [9]. Sometimes even stronger condition is imposed that the temporal distance is *equal* to the pre-specified distance, constant gaps discussed by Altman, Gaujal and Hordijk [1], see also Anily, Glass and Hassin [2], where periodic machine maintenance problem is considered with equal distances between consecutive services of the same machine.

The distance-constrained model, however, suffers from a serious problem which is that there may not be a feasible solution that respects the distance constraints and at the same time makes sure that tasks are done at given rates. In this paper, we propose the response time variability metric instead to avoid the feasibility problem but at the same time preserve the main idea of having any two consecutive tasks at a distance which remains as constant as the existing resources and other competing tasks allow. We formulate the problem as follows.

Given are n positive integers $d_1 \leq \ldots \leq d_n$, define $D = \sum_{i=1}^n d_i$ and $r_i = \frac{d_i}{D}$. Consider a sequence $s = s_1 s_2 \ldots s_D$ of length D where i (client, product or task) occurs exactly d_i times. Such sequence will be called feasible. For $d_i \geq 2$, for any two consecutive occurrences of i define a distance t being the number of positions that separate the two plus 1. This distance will also be referred to as the end-to-end distance between the two occurrences. Since i occurs exactly d_i times in s, then there are exactly d_i distances $t_1^i, \ldots, t_{d_i}^i$ for i, where t_{d_i} is the distance between the last and the first occurrence of i. Since

$$t_1^i + \ldots + t_{d_i}^i = D,$$

then the average distance $\bar{t}_i$ between the i's in s equals

$$\frac{D}{d_i} = \frac{1}{r_i},$$

and it is the same for each feasible sequence s. We define the response time variability for i as follows

$$RTV_i = \sum_{1 \leq j \leq d_i} (t_j^i - \bar{t}_i)^2,$$

and the response time variability as

$$RTV = \sum_{i=1}^n RTV_i = \sum_{i=1}^n \sum_{1 \leq j \leq d_i} (t_j^i - \bar{t}_i)^2.$$

We observe that the response time variability is a weighted variance with weights being equal to demands, that is

$$RTV = \sum_{i=1}^n d_i Var_i,$$

where $Var_i = \frac{1}{d_i} \sum_{1 \leq j \leq d_i} (t_j^i - \bar{t}_i)^2$.

We also consider the maximum deviation just-in-time sequencing problem, or the bottleneck minimization problem. We shall use the term throughput error (TE) as well in the paper. Let $x_{i,k}$ be the number of occurrences of i in the k-prefix of s, that is in $s_1 \ldots s_k$.

$$B^* = \min \max_{i,k} |x_{i,k} - kr_i|$$

Subject to

$x_{i,k} \leq x_{i,k+1}$	for $i = 1, \ldots, n$ and $k = 1, \ldots, D-1$
$\sum_{i=1}^{n} x_{i,k} = k$	for $k = 1, \ldots, D$
$x_{i,k}$ non-negative integers	for $i = 1, \ldots, n$ and $k = 1, \ldots, D$.

The plan for the paper is as follows. Section 2 studies the optimization and the computational complexity of the response time variability problem. It first introduces the number decomposition graphs as a useful tool for the analysis of the response time variability. Second, it shows an optimization algorithm for two product case. The algorithm minimizes both the response time variability and the bottleneck at the same time. Third, the section shows a dynamic programming algorithm to prove polynomial solvability for a fixed number of products. Finally, it proves that the problem is NP-complete. Section 3 presents optimization algorithms for the response time variability. Section 4 presents a simple position exchange heuristic that takes a sequence exchanges positions of some products as long as the exchanges lead to improvements in the value of RTV. The sequences subjected to this exchange heuristic are generated by various procedures: the bottleneck that solves the bottleneck problem to optimality, insertion based on a solution to the two product case, Webster and Jefferson based on the well known methods of the apportionment. These are described in Section 4.2. Section 5 presents results of computational experiments with the exchange heuristic as well as very limited experiment with an optimization algorithm based on mathematical programming. Finally, Section 6 presents conclusions and remarks on further research.

2. Optimization and Complexity

2.1. **Number Decomposition Graphs vs RTV.**

For product i, consider a vector $\alpha = (\alpha_1, \ldots, \alpha_{d_i})$ of $d_i \geq 2$ *positive* integers that sum up to D. Without loss of generality we assume the coordinates of α ordered in descending order, that is $\alpha_1 \geq \ldots \geq \alpha_{d_i}$. Any vector α that meets the above conditions will be referred to as the *decomposition* vector of D into d_i components. Now let us define a *unit* exchange operation on the decomposition vector α as follows. Consider two components α_j and $\alpha_k > 1$, $j < k$, of α. Replace α_j by $\alpha_j + 1$ and α_k by $\alpha_k - 1$ and keep all other components of α unchanged. The new components after ordering them define another decomposition vector β. For instance, consider $\alpha = (6, 6, 5)$, then adding 1 to the second component and subtracting 1 from the third component leads to the vector $\beta = (7, 6, 4)$. Let us consider a *weighted* directed graph $\mathcal{D}_i$, we refer to this graph as the number decomposition graph for product i, with the set of nodes

$\mathcal{N}_i$ including all decomposition vectors of D into d_i components, and the set of arcs $\mathcal{A}_i$ including all pairs of decomposition vectors (α, β) such β is obtained from α by a unit exchange operation. The weight of the arc (α, β) is defined as

$$2(\alpha_j - \alpha_k + 1).$$

We have the following straightforward properties of $\mathcal{D}_i$.

Lemma 2.1. *The following properties hold for $\mathcal{D}_i$:*

- *The graph $\mathcal{D}_i$ is acyclic with a single maximal node, $\mathcal{N}_i$, and a single minimal node, $\mathcal{M}_i$.*
- $\mathcal{N}_i = (\underbrace{\lceil \frac{D}{d_i} \rceil, \ldots, \lceil \frac{D}{d_i} \rceil}_{D \mod d_i}, \underbrace{\lfloor \frac{D}{d_i} \rfloor, \ldots, \lfloor \frac{D}{d_i} \rfloor}_{d_i - D \mod d_i}).$
- $\mathcal{M}_i = (D - d_i + 1, \underbrace{1, \ldots, 1}_{d_i - 1}).$

Proof. If $(\alpha, \beta) \in \mathcal{A}_i$, then

$$\sum_{i=1}^{n} \alpha_i^2 < \sum_{i=1}^{n} \beta_i^2.$$

Therefore, $\mathcal{D}_i$ must be acyclic. The node $\mathcal{N}_i$ has in-degree 0. Otherwise, there would be a node α in $\mathcal{D}_i$ such that $(\alpha, \mathcal{N}_i) \in \mathcal{A}_i$. Then, there would be two components α_j and $\alpha_k > 1$, $j < k$, of α that would become some components $\alpha_j + 1$ and $\alpha_k - 1$ of $\mathcal{N}_i$. Then, $\alpha_j + 1$ must be either $\lfloor \frac{D}{d_i} \rfloor$ or $\lceil \frac{D}{d_i} \rceil$. Consequently, α_j must be either $\lfloor \frac{D}{d_i} \rfloor - 1$ or $\lceil \frac{D}{d_i} \rceil - 1$. Then, however, $\alpha_k - 1 \leq \lceil \frac{D}{d_i} \rceil - 2 \leq \lfloor \frac{D}{d_i} \rfloor - 1$, and thus $\alpha_k - 1$ cannot be a component of $\mathcal{N}_i$. No other node has in-degree 0 since it can be shown that there is a path from $\mathcal{N}_i$ to the node in $\mathcal{D}_i$. Finally, by definition the only node with out-degree 0 is the one that has the last d_i components all equal 1. Otherwise a unit exchange would be possible. However, since all components of any node must sum up to d_i, there is only one such a node, namely $\mathcal{M}_i$.

■

Figure 1 presents the number decomposition graph for $D = 16$ and $d_i = 3$.
Consider $\mathcal{N}_i$ and another node α in $\mathcal{D}_i$. The length of a directed path from $\mathcal{N}_i$ to α is the sum of all the weights along the path. We have the following lemma.

Lemma 2.2. *For any node α in $\mathcal{D}_i$, all directed paths from $\mathcal{N}_i$ do α have the same length.*

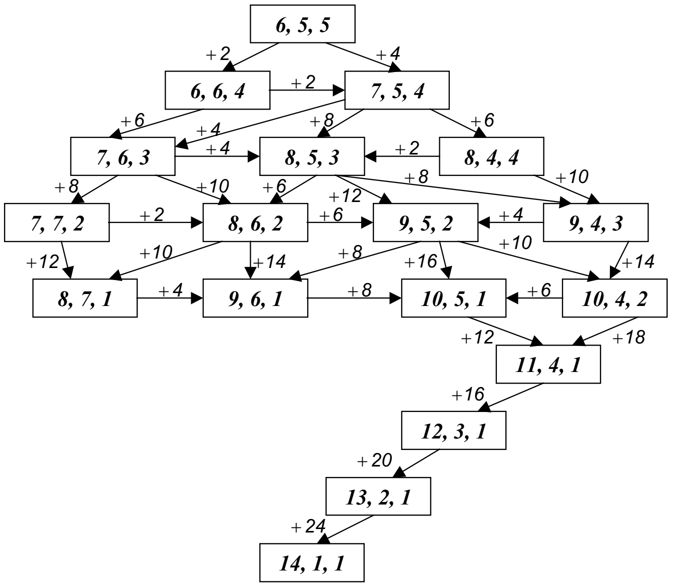

FIGURE 1. The number decomposition graph for D=16 and $d_i = 3$.

Proof. For a node $\alpha \in \mathcal{D}_i$ define

$$RTV(\alpha) = \sum_{j=1}^{n} (\alpha_j - \frac{D}{d_i})^2.$$

Let p be any path from $\mathcal{N}_i$ to α, and $w(p)$ its total weight. It can be shown that

$$RTV(\alpha) = RTV(\mathcal{N}_i) + w(p).$$

This, however, implies the lemma since both $RTV(\alpha)$ and $RTV(\mathcal{N}_i)$ are constants, that is path independent. ∎

The following lemma links the decomposition graphs and the response time variability problem.

Lemma 2.3. *Let S be a solution to the response time variability problem with value RTV. Then there are nodes $\alpha^1, \ldots, \alpha^n$ in the decomposition graphs $\mathcal{D}_1, \ldots, \mathcal{D}_n$ respectively, such that*

$$RTV = \sum_{i=1}^{n} w(\mathcal{N}_i, \alpha^i) + \sum_{i=1}^{n} RTV(\mathcal{N}_i),$$

where $w(\mathcal{N}_i, \alpha^i)$ is the length of a directed path from $\mathcal{N}_i$ to α^i in $\mathcal{D}_i$ and

$$RTV(\mathcal{N}_i) = (D \bmod d_i)(\lceil \frac{D}{d_i} \rceil - \frac{D}{d_i})^2 + (d_i - D \bmod d_i)(\lfloor \frac{D}{d_i} \rfloor - \frac{D}{d_i})^2$$

for $i = 1, \dots, n$.

Proof. Follows from Lemma 2.2. ■

2.2. **Two Product Case.** This section considers a two product case. It shows a solution that minimizes both the response time variability and the maximum deviation at the same time, which generally is impossible for more than two products. Prior to giving the details of the solution we need to point out that the sequence minimizing the response time variability for two products is quite straightforward to obtain as follows. Let $d_1 < d_2$. We omit the case $d_1 = d_2$ since it is trivial. It follows from our discussion in Subsection 2.1 that if one can find a solution with distances $\lceil \frac{D}{d_1} \rceil$ and $\lfloor \frac{D}{d_1} \rfloor$ for product 1, and $\lceil \frac{D}{d_2} \rceil$ and $\lfloor \frac{D}{d_2} \rfloor$ for product 2, then this solution will be optimal. We notice, however, that such solution is always possible since $\lfloor \frac{D}{d_1} \rfloor \geq 2$ and $2 > \frac{D}{d_2} > 1$. Therefore, starting a sequence with product 1 and then sequencing any consecutive copy of 1 at a distance either $\lceil \frac{D}{d_1} \rceil$ or $\lfloor \frac{D}{d_1} \rfloor$ (the number of times each distance is used is given in Lemma 2.3) from the last one will produce the sequence where empty positions are separated by at most a single copy of product 1. This allows us to fit in product 2 in the empty positions ensuring the desired distances for product 2.

We now discuss details of the solutions that minimizes both the response time variability and the maximum deviation. We assume that the greatest common divisor of d_1, d_2, and $D = d_1 + d_2$ is 1, that is $\gcd\{d_1, d_2\} = 1$. Otherwise, the optimal solution for $\frac{d_1}{g}$ and $\frac{d_2}{g}$ can be repeated $g = \gcd\{d_1, d_2\}$ times resulting into an optimal solution with respect to both metrics for the original instance with demands d_1 and d_2.

Our solution relies on the *ideal* positions for copy j of product i, $i = 1, 2$ and $j = 1, \dots, d_i$ defined as $\lceil \frac{2j-1}{2r_i} \rceil$, see Kubiak and Sethi [10]. We consider two cases. One with one demand being odd and the other even is easier to deal with since then all the ideal positions are pairwise different. The other with both demands being odd is a bit more involved since then there may be two ideal positions with the same value creating a conflict for two copies.

Lemma 2.4. *Consider numbers $a_j = \frac{2j-1}{2r_1}$ for $j = 1, \dots, d_1$ and $b_k = \frac{2k-1}{2r_2}$ for $k = 1, \dots, d_2$. If $l-1 < a_j \leq l$ and $l-1 < b_k \leq l$ for some $l = 1, \dots, D$, then $a_j = b_k = l$.*

Proof. Since $a_j = \frac{2j-1}{2r_1} = (l-1) + f_1$ and $b_k = \frac{2k-1}{2r_2} = (l-1) + f_2$, with $0 < f_i \leq 1$ $(i = 1, 2)$, we have

$$2j - 1 = 2r_1(l-1) + 2r_1 f_1$$

$$2k - 1 = 2r_2(l-1) + 2r_2 f_2$$

and, since $r_1 + r_2 = 1$

$$j + k - 1 = (l-1) + r_1 f_1 + r_2 f_2$$

which, as the left-hand side of the equation and $(l-1)$are integers, can only be possible if $r_1 f_1 + r_2 f_2$ is also integer; however, $0 < r_1 f_1 + r_2 f_2 \leq 1$, since $r_1 f_1 + r_2 f_2 \leq (r_1 + r_2) \max(f_1, f_2) = \max(f_1, f_2) \leq 1$, and is equal to 1 if and only if $f_1 = f_2 = 1$.

∎

Lemma 2.5. *If one of demands, d_1 and d_2, is odd and the other even than none of the numbers $a_j = \frac{2j-1}{2r_1}$ for $j = 1, \dots, d_1$ and $b_k = \frac{2k-1}{2r_2}$ for $k = 1, \dots, d_2$ is an integer.*

Proof. If one of demands, d_1 and d_2, is odd and the other even, then D is odd and, consequently, $(2j-1)D$ is odd, $j = 1, \dots, d_i$ and $i = 1, 2$. Since $2d_i$ is even, then $\frac{(2j-1)D}{2d_i} = \frac{2j-1}{2r_i}$ is not an integer for $j = 1, \dots, d_i$ and $i = 1, 2$. ∎

Lemma 2.6. *If one of demands, d_1 and d_2, is odd and the other even, then $\alpha_j = \lceil a_j \rceil$ for $j = 1, \dots, d_1$ and $\beta_k = \lceil b_k \rceil$ for $k = 1, \dots, d_2$ are pairwise different.*

Proof. Follows immediately from Lemma 2.4 and Lemma 2.5. ∎

Lemma 2.7. *If both d_1 and d_2 are odd, than none of the numbers $a_j = \frac{2j-1}{2r_1}$ for $j = 1, \dots, \frac{d_1+1}{2} - 1, \frac{d_1+1}{2} + 1, \dots, d_1$ and $b_k = \frac{2k-1}{2r_2}$ for $k = 1, \dots, \frac{d_2+1}{2} - 1, \frac{d_2+1}{2} + 1, \dots, d_2$ is integer. However, both $a_{\frac{d_1+1}{2}}$ and $b_{\frac{d_2+1}{2}} = \frac{D}{2}$ are integers equal $\frac{D}{2}$.*

Proof. If d_1 and d_2 are odd, then $\frac{D}{2}$ is integer, however, $\frac{D}{2}$ and d_i are relatively prime for $i = 1, 2$. Furthermore, d_i divides $2j - 1$ for $j = 1, \dots, d_i$ and $i = 1, 2$ only if $j = \frac{d_i+1}{2}$. Therefore, $a_{\frac{d_1+1}{2}} = b_{\frac{d_2+1}{2}} = \frac{D}{2}$ and none of the numbers $\frac{(2j-1)D}{2d_i} = \frac{2j-1}{2r_i}$ for $j = 1, \dots, \frac{d_i+1}{2} - 1, \frac{d_i+1}{2} + 1, \dots, d_i$ and $i = 1, 2$ is an integer. ■

Lemma 2.8. *If both d_1 and d_2 are odd, then $\alpha_j = \lceil a_j \rceil$ for $j = 1, \dots, d_1$, and $\beta_j = \lceil b_k \rceil$ for $k = 1, \dots, \frac{d_2+1}{2} - 1, \frac{d_2+1}{2} + 1, \dots, d_2$, and $\beta_{\frac{d_2+1}{2}} = \lceil b_{\frac{d_2+1}{2}} \rceil + 1 = \frac{D}{2} + 1$ are pairwise different.*

Proof. Follows immediately from Lemma 2.4 and Lemma 2.7, and the fact that $\lceil a_j \rceil \neq \frac{D}{2} + 1$ for $j = 1, \dots, d_1$, and $\lceil b_k \rceil \neq \frac{D}{2} + 1$ for $k = 1, \dots, d_2$. The latter holds since $\lceil \frac{2j-1}{2r_i} \rceil \geq \frac{D}{2} + 2$ for $j = \frac{d_i+1}{2} + 1$. ■

We shall refer to the solutions defined in Lemma 2.6 and in Lemma 2.8 as $\alpha\beta$ solutions. We now show in Lemmas 2.9 to 2.11 that these solutions are optimal for the response variability problem. To that end we prove that the distance between any two consecutive copies of product 1 equals either $\lceil \frac{D}{d_1} \rceil$ or $\lfloor \frac{D}{d_1} \rfloor$ and for product 2 equals either $\lceil \frac{D}{d_2} \rceil$ or $\lfloor \frac{D}{d_2} \rfloor$.

Lemma 2.9. *We have $\lfloor \frac{D}{d_1} \rfloor \leq \lceil a_{j+1} \rceil - \lceil a_j \rceil \leq \lceil \frac{D}{d_1} \rceil$ for $j = 1, \dots, d_1 - 1$ and $\lfloor \frac{D}{d_2} \rfloor \leq \lceil b_{j+1} \rceil - \lceil b_j \rceil \leq \lceil \frac{D}{d_2} \rceil$ for $j = 1, \dots, d_2 - 1$.*

Proof. We have,

$$\frac{D}{d_1} - 1 = a_{j+1} - a_j - 1 \leq \lceil a_{j+1} \rceil - \lceil a_j \rceil \leq a_{j+1} - a_j + 1 = \frac{D}{d_1} + 1.$$

Since $\frac{D}{d_1}$ is not an integer and $\lceil a_{j+1} \rceil - \lceil a_j \rceil$ is an integer, then

$$\lfloor \frac{D}{d_1} \rfloor \leq \lceil a_{j+1} \rceil - \lceil a_j \rceil \leq \lceil \frac{D}{d_1} \rceil.$$

The proof for product 2 is similar and thus will be omitted. ■

Lemma 2.10. *We have $\lfloor \frac{D}{d_1} \rfloor \leq D - \lceil a_{d_1} \rceil + \lceil a_1 \rceil \leq \lceil \frac{D}{d_1} \rceil$ and $\lfloor \frac{D}{d_2} \rfloor \leq D - \lceil b_{d_2} \rceil + \lceil b_1 \rceil \leq \lceil \frac{D}{d_2} \rceil$.*

Proof. We have $a_{d_1} = D - \frac{D}{2d_1}$, $a_1 = \frac{D}{2d_1}$ and consequently

$$\frac{D}{d_1} - 1 = D - a_{d_1} + a_1 - 1 \leq D - \lceil a_{d_1} \rceil + \lceil a_1 \rceil \leq D - a_{d_1} + a_1 + 1 = \frac{D}{d_1} + 1.$$

Since $\frac{D}{d_1}$ is not an integer and $D + \lceil a_{d_1} \rceil - \lceil a_1 \rceil$ is an integer, then

$$\lfloor \frac{D}{d_1} \rfloor \leq D + \lceil a_{d_1} \rceil - \lceil a_1 \rceil \leq \lceil \frac{D}{d_1} \rceil.$$

The proof for product 2 is similar and thus will be omitted.

■

Lemma 2.11. *For d_1 and d_2 odd, we have*

$$b_{\frac{d_2+1}{2}} + 1 - \lceil b_{\frac{d_2+1}{2}-1} \rceil = \lceil \frac{D}{d_2} \rceil$$

and

$$\lceil b_{\frac{d_2+1}{2}+1} \rceil - b_{\frac{d_2+1}{2}} - 1 = \lfloor \frac{D}{d_2} \rfloor.$$

Proof. For d_1 and d_2 odd, we have $b_{\frac{d_2+1}{2}} = \frac{D}{2}$, which is an integer. Consequently,

$$\frac{D}{d_2} = b_{\frac{d_2+1}{2}} - b_{\frac{d_2+1}{2}-1} \leq b_{\frac{d_2+1}{2}} + 1 - \lceil b_{\frac{d_2+1}{2}-1} \rceil \leq b_{\frac{d_2+1}{2}} + 1 - b_{\frac{d_2+1}{2}-1} = \frac{D}{d_2} + 1.$$

Since $\frac{D}{d_2}$ is not an integer and $b_{\frac{d_2+1}{2}} + 1 - \lceil b_{\frac{d_2+1}{2}-1} \rceil$ is an integer, then the first equality holds. Furthermore,

$$\frac{D}{d_2} - 1 = b_{\frac{d_2+1}{2}+1} - b_{\frac{d_2+1}{2}} - 1 \leq \lceil b_{\frac{d_2+1}{2}+1} \rceil - b_{\frac{d_2+1}{2}} - 1 \leq b_{\frac{d_2+1}{2}+1} - b_{\frac{d_2+1}{2}} = \frac{D}{d_2}.$$

Since $\lceil b_{\frac{d_2+1}{2}+1} \rceil - b_{\frac{d_2+1}{2}} - 1$ is an integer, then the second inequality also holds. This proves the lemma. ■

We can now conclude with the following theorem.

Theorem 2.12. *The $\alpha\beta$ solutions are optimal for the response time variability.*

Proof. Lemma 2.6, Lemmas 2.9 and 2.10 show that if one of demands, d_1 and d_2, is odd and the other even, then distances in the $\alpha\beta$ solutions are equal either $\lfloor \frac{D}{d_i} \rfloor$ or $\lceil \frac{D}{d_i} \rceil$ for $i = 1, 2$. Lemmas 2.8 to 2.11 show that if both demands, d_1 and d_2, are odd, then distances in the $\alpha\beta$ solutions are equal either $\lfloor \frac{D}{d_i} \rfloor$ or $\lceil \frac{D}{d_i} \rceil$ for $i = 1, 2$ as well.

■

We close this section by showing that the $\alpha\beta$ sequence minimizes maximum deviation as well.

Theorem 2.13. *The $\alpha\beta$ solutions minimize maximum deviation.*

Proof. The theorem holds for if one of the demands is even and the other odd since then all copies are sequenced in their ideal positions, which minimizes maximum deviation Kubiak [12]. If both demands are odd, then all copies are in their ideal positions, except copy $\frac{d_2+1}{2}$ of product 2 which is in position $\frac{D}{2}+1$ whereas its ideal position is $\frac{D}{2}$. This, however, does not change the deviation for the copy, which is $\frac{1}{2}$ in either position. Consequently, the maximum deviation equals $\frac{1}{2}$ which is optimal for both demands being odd, Kubiak [12]. Therefore, the $\alpha\beta$ solutions minimize maximum deviation.

■

2.3. **Fixed Number.** This section shows a straightforward dynamic programming for the response time variability problem. The program proves that the problem can be solved in polynomial time for any *fixed* number of products, and thus complements our other results concerning its complexity. However, it is not designed as a practical alternative for efficiently solving the problem.

The state of the dynamic programming is represented by a quadruple $\langle f,l,r,d\rangle$. The f is an n dimensional vector $f=(f_1,\ldots,f_n)$, where $f_i=0,1,\ldots,D-d_i+1$ for $i=1,\ldots,n$ represents the position of the first copy of product i. The l is an n dimensional vector $l=(l_1,\ldots,l_n)$, where $l_i=0,d_i+1,\ldots,D$ for $i=1,\ldots,n$ represents the position of the last copy of product i in the sequence of length d. The r is an n dimensional vector $r=(r_1,\ldots,r_n)$ represents the number of copies that remain to be sequenced, $r_i=0,...,d_i$. Finally, d is the length of the current sequence, $d=0,...,D$. Initially, $f=l=0$, $r=(d_1,...,d_n)$, and $d=0$. A final state is any state with $r=0$ and $d=D$. There is a weighted arc from a non-final state $\langle f,l,r,d\rangle$ to a state $\langle f',l',r',d'\rangle$ if and only if there is an i such that $r'_i=r_i-1\geq 0$, $d'=d+1$, $l'_i=d'$. Furthermore, if $f_i=0$, then $f'_i=d'$. The weight for the arc is calculated as follows for $d_i\geq 2$,

$$l_{\langle f,l,r,d\rangle\langle f',l',r',d'\rangle}=\begin{cases}0, & r_i=d_i;\\ (d'-l_i-\frac{D}{d_i})^2, & d_i-1\geq r_i>1;\\ (d'-l_i-\frac{D}{d_i})^2+(D-d'+f_i-\frac{D}{d_i})^2, & r_i=1\end{cases}$$

and it is equal to 0 for $d_i=1$. Finally, we connect all final states to a dummy state referred to as the destination. All arc to the destination have length 0. The shortest path between the initial state and the destination obviously defines an optimal solution to the response time variability. The shortest path can be found in time which is polynomial in the number of nodes. This is of $O(D^{3n+1})$, therefore the complexity is polynomial for a fixed number of products n.

2.4. **Complexity.** This section shows that the response time variability problem is NP-complete. The reduction is from the Periodic Maintenance Scheduling Problem studied by Anily, Glass and Hassin [2]. The latter is defined as follows. Given m machines and integer service intervals $l_1, l_2, \ldots, l_m$ such that $\sum \frac{1}{l_i} < 1$. Does there exist a servicing schedule $s_1, \ldots, s_L$, where $L = lcm(l_1, l_2, \ldots, l_m)$ is the least common multiplier of $l_1, l_2, \ldots, l_m$, of these machines in which consecutive servicing of machine i are exactly l_i time slots apart and no more than one machine is serviced in a single time slot ?
The Periodic Maintenance Scheduling Problem has been shown NP-complete by Bar-Noy et al. [4] who prove the following.

Theorem 2.14. *The Periodic Maintenance Scheduling Problem is NP-complete in the strong sense.*

We now show the following theorem.

Theorem 2.15. *The Response Time Variability Problem is NP-complete.*

Proof. For an instance $l_1, l_2, \ldots, l_m$ of the Periodic Maintenance Scheduling Problem, define demands $d_i = \frac{L}{l_i}$, $i = 1, \ldots, m$, and $d_{m+1} = 2L - \sum d_i$, where $L = lcm(l_1, l_2, \ldots, l_m)$. Thus, $D = 2L$ and $n = m + 1$. Finally, the upper bound on the variability $V = \beta(2 - \frac{D}{d_{m+1}})^2 + (d_{m+1} - \beta)(1 - \frac{D}{d_{m+1}})^2$, where $\beta \equiv D \mod d_{m+1}$. Assume that there is a solution $s_1, s_2, \ldots, s_L$ for the Periodic Maintenance Scheduling Problem. Then, consider the sequence $S = s_1^{'}, n, s_2^{'}, n, \ldots, s_L^{'}, n$ with every other time slot occupied by n, and $s_j^{'} = s_j$ if s_j is a machine or $s_j^{'} = n$ if s_j is empty in the Periodic Maintenance Scheduling Problem. Consequently, consecutive copies of i are exactly $2l_i$ time slots apart in S for $i = 1, \ldots, m$. Therefore, the variability for each of these i's is 0. Furthermore, any two consecutive copies on n are either next to each other or separated by a single copy of $i \neq n$, and thus, the variability of S equals V. Now, let us assume that there is a solution Q to the response time variability with variability $Var \leq V$. First, we observe from Lemma 2.1 that $Var_n \geq V$ in any solution to the response time variability, and consequently $Var = Var_n$ and $Var_i = 0$ for $i = 1, \ldots, m$ in Q. Second, again by Lemma 2.1, we observe that if any two copies of n were three or more slots apart in Q, that is, if they were separated by two or more copies of $i \neq n$, then $Var_n > V$

for Q, which would lead to a contradiction. Therefore, no $i \le m$ and $j \le m$ are next to each other in Q. The first observation implies that all distances between consecutive copies of product i, $i = 1, \ldots, m$, are equal in Q. Furthermore, since there are d_i copies of i in the sequence of length D, then the distance between the consecutive copies of i is $\frac{D}{d_i} = 2l_i$ in Q. Now, let $c_i, c_i + 2l_i, \ldots, c_i + 2(d_i - 1)l_i$ be the ends of time slots in Q occupied by i, $i = 1, \ldots, m$. Of course all these numbers are different as no time slot is occupied by more than one i. Consequently, all numbers, $\frac{c_i}{2}, \frac{c_i}{2} + l_i, \ldots, \frac{c_i}{2} + (d_i - 1)l_i$, $i = 1, \ldots, m$, are different. Furthermore, by the second observation, if for some i, j, h and k, $\frac{c_k}{2} + hl_k > \frac{c_i}{2} + jl_i$, then $\frac{c_k}{2} + hl_k - (\frac{c_i}{2} + jl_i) > \frac{1}{2}$. Consequently, all numbers $\lceil \frac{c_i}{2} \rceil, \lceil \frac{c_i}{2} \rceil + l_i, \ldots, \lceil \frac{c_i}{2} \rceil + (d_i - 1)l_i$, $i = 1, \ldots, m$, are different. By servicing machine i in time slots $\lceil \frac{c_i}{2} \rceil, \lceil \frac{c_i}{2} \rceil + l_i, \ldots, \lceil \frac{c_i}{2} \rceil + (d_i - 1)l_i$, $i = 1, \ldots, m$ we obtain a solution to the Periodic Maintenance Scheduling Problem. This proves the theorem.

■

It remains an open question whether the response time variability problem is NP-complete in the strong sense.

3. Exact Algorithm

3.1. **Mathematical Programming Models to Obtain Optimal Solutions.** Optimal solutions to the response time variability problem (RTVP) can be obtained by means of the DP scheme described in Section 2.3. Another approach to get optimal solutions consists in using mathematical programming models.

From the outset, RTVP can be considered as a special case of Quadratic Assignment Problem (QAP) and therefore formalized as a quadratic integer programming (QIP). We will use the following notation:

$$G1 = \{i | d_i \ge 2\}$$

$$UB_i = D - d_i + 1$$

Upper bounds, $\forall i \in G1$, on the distance between two consecutive units of product i.

$$E_{ik} = k \;\; \forall i \in G1 \;\; k = 1, ..., d_i$$

Earliest position that can be occupied by unit k of product i.

$$L_{ik} = D - d_i + k \;\; \forall i \in G1 \;\; k = 1, ..., d_i$$

Latest position that can be occupied by unit k of product i.

$$y_{ikh} \in \{0, 1\}$$

$i = 1, ..., n;\ h \in H_{ik}$,where $H_{ik} = \{h | L_{ik} \leq h \leq E_{ik}\}$ $(i = 1, ..., n;\ k = 1, ..., d_i)$ and with $y_{ikh} = 1$ if and only if unit k of product i is placed in position h.

$$S_{ik} = \{(h, j) | h \in H_{ik},\ j \in H_{i,k+1}\}\ \ \forall i \in G1,\ \ k = 1, ..., d_i - 1$$

$$S_{id_i} = \{(h, j) |\ h \in H_{id_i},\ j \in H_{i1}\}\ \ \forall i \in G1$$

Sets of pairs of positions that can be occupied, respectively, by units k and $(k+1)$ mod d_i of product i. Define

$$RTV =$$

$$\sum_{i \in G1} \sum_{k=1}^{d_i - 1} \sum_{(h,j) \in S_{ik}} (j-h)^2 y_{ikh} y_{i,k+1,j} + \sum_{i \in G1} \sum_{(h,j) \in S_{id_i}} (D+j-h)^2 y_{id_ih} y_{i1j} - \sum_{i \in G1} \bar{t}_i^2$$

Then, the model can be formulated as follows:

$$minimise\ RTV$$

s. t.

$$\sum_{\forall (i,k) | h \in H_{ik}} y_{ikh} = 1 \quad h = 1, ..., D$$

$$\sum_{h \in H_{ik}} y_{ikh} = 1 \quad i = 1, ..., n;\ k = 1, ..., d_i$$

$$\sum_{h \in H_{i,k+1}} h y_{i,k+1,h} - \sum_{h \in H_{ik}} h y_{ikh} \geq 1\ \forall i \in G1 \quad k = 1, ..., d_i - 1$$

Where the first two groups of constraints are the assignment constraints and the third group the precedence constraints.

This QIP model can be solved by means of ILOG SOLVER. However, a preliminary computational experiment showed, as it was expected, that the algorithm was not able to find optimal solutions, within an acceptable computing time (fixed at 180 sec), except for very small instances.

Instead, the use of CPLEX with MILP models derived from the QIP turned out to be more efficient, as a computational experiment showed. In order to linearize the QIP model three techniques were applied:

(i) Substitute the binary variables $\delta_{ikhj}$for the products $y_{ikh} \cdot y_{i,k+1,j}$ and include the constraints:

$$1 + \delta_{ikhj} \geq y_{ikh} + y_{i,k+1,j}.$$

(ii) Substitute the binary variables $\delta_{ikhj}$for the products $y_{ikh} \cdot y_{i,k+1,j}$ and include the constraints:

$$\begin{aligned} 2 \cdot \delta_{ikhj} &\leq y_{ikh} + y_{i,k+1,j} \\ 1 + \delta_{ikhj} &\geq y_{ikh} + y_{i,k+1,j}. \end{aligned}$$

(iii) Introduce the binary variables δ_{ik}^{j} $(i \in G1;\ k = 1, ..., d_i;\ j = 1, ..., UB_i)$, substitute the objective function for $\sum\limits_{i,k,j} j^2 \cdot \delta_{ik}^{j}$ and include the constraints:

$$\sum_h hy_{i,k+1,h} - \sum_h hy_{ikh} = \delta_{ik}^{1} + ... + j\cdot \delta_{ik}^{j} + ... + UB_i \cdot \delta_{ik}^{UB_i}$$

$$D - \sum_{h \in H_{id_i}} hy_{i,d_i,h} + \sum_{h \in H_{i1}} hy_{i1h} = \delta_{ik}^{1} + ... + j \cdot \delta_{ik}^{j} + ... + UB_i \cdot \delta_{ik}^{UB_i}$$

$$\sum_{j=1}^{UB_i} \delta_{ik}^{j} = 1 \ \forall i, k$$

(which allows suppressing the precedence constraints, since they are embedded in the new ones).

The lower computing times were those corresponding to the model obtained by technique (iii). Therefore, this was the model that we used to compare the solutions obtained by heuristics with optimal solutions in the computational experiment described in Section 5. Notwithstanding, the experiments indicate that the practical limit to get optimal solutions applying CPLEX to this model is $D = 25$.

4. Heuristics

This section describes heuristics for the response time variability problem. Each of these heuristics uses the exchange procedure described in Subsection 4.1 to exchange products next to each other in a given sequence in order to reduce the sequence's response time variability. The exchange procedure can be applied to any feasible sequence, and as expected its result is rather sensitive to the selection of the initial sequence it is applied to. Therefore, in Subsection 4.2 we describe a number of different ways the initial sequence was obtained in our study.

4.1. **The Exchange.** Consider a sequence $s = s_1 \dots s_D$ and a product i with $d_i \geq 2$ in position p of s, that is $s_p = i$. We define the closest to position p *clockwise* i as the first i in s encountered when moving clockwise away from p, similarly, we define the closest to p *counter-clockwise* i as the first i encountered when moving counter-clockwise away from p. Notice that if $d_i = 2$, then that the clockwise i is the same as the counter-clockwise i. The Exchange Heuristic does the exchange of two products that are next to each other whenever the exchange leads to the reduction of response time variability. More precisely, consider products i and j, $i \neq j$, that are next to each other in a given sequence s, assume the i is in position p and the j in position $p+1$. Let L_i and R_i be distances to the closest to p clockwise, respectively counter-clockwise, i in s. Similarly, let L_j and R_j be distances to the closest to $p+1$ clockwise, respectively counter-clockwise, j in s. Then, for $d_i, d_j \geq 2$,

$$B = L_i^2 + R_i^2 + L_j^2 + R_j^2$$

before the exchange and

$$A = (L_i + 1)^2 + (R_i - 1)^2 + (L_j - 1)^2 + (R_j + 1)^2$$

after the exchange. Since all the other distances remain unchanged, we have the following net change in the value of the response time variability,

$$\Delta = B - A = 2(L_i - R_i + 1) + 2(R_j - L_j + 1).$$

If $d_j = 1$ and $d_i \geq 2$, then $\Delta = 2(L_i - R_i + 1)$. By symmetry, if $d_i = 1$ and $d_j \geq 2$, then $\Delta = 2(L_j - R_j + 1)$. Finally, if $d_i = d_j = 1$, the $\Delta = 0$. The exchange takes place only if Δ is positive. The exchange algorithm passes counter-clockwise through the sequence and checks for each neighboring couple $s_p s_{p+1}$ if the exchange within the couple reduces the response time variability, if so, the exchange is made and the algorithm moves to position $p+1$. It is worth keeping in mind that position D is immediately followed by position 1. If position 1 is reached without any reduction in the response time variability, then the heuristic stops. Otherwise, the next pass through the sequence begins. We observe that the heuristic eventually stops since each pass either reduces the response time variability, an optimal value of which is obviously finite, or no improvement takes place. In fact the exchange heuristic goes a bit further, namely, whenever no improvement is possible in a given pass it tries to do the exchanges even if their net improvement is 0. However, these exchanges are only done if the maximum distance for either of the two products being exchanged does not increase and moreover at least one of the maxima actually decreases. This last condition ensures that the heuristic actually terminates. The exchange heuristic is applied to an initial sequence which is generated in a number of different ways. These will be detailed in the subsequent sections.

4.2. **The Initial Sequences.**

4.2.1. *Bottleneck (minimum throughput error) sequences.* Bottleneck sequences have been obtained by solving to optimality the bottleneck problem defined in the Introduction. We used the algorithm of Moreno, 2002 to solve the bottleneck problem. However, other approaches have been proposed in the literature and could be used to obtain, perhaps different, bottleneck sequences(see Kubiak [11], Steiner and Yeomans [16], and Bautista, companys and Corominas [6]).

4.2.2. *Random sequences.* The bottleneck sequence s has been randomized as follows. For each position x in $1..D$, get a random number ran in the range $1..D$. Then, swap $S[x]$ with $S[ran]$.

4.2.3. *Webster's sequences.* These sequences have been obtained by applying the parametric method of apportionment with parameter $\delta = \frac{1}{2}$, known as Webster's method (Balinski and Young [3], Bautista, Companys and Corominas [5]). The sequence is generated as follows. Consider x_{it} the number of product i copies in the sequence of length t, $t = 0, 1, \ldots$. Assume $x_{i0} = 0$, $i = 1, \ldots, n$. The product to be sequenced in position $t+1$ can be computed as follows $i^* = \arg\max_i \{\frac{d_i}{(x_{it}+\delta)}\}$.

4.2.4. *Jefferson's (stride scheduling) sequences.* These sequences have been generated by applying the parametric method of apportionment, described before, with $\delta = 1$, known as Jefferson's parametric method (Balinski and Young [3]). The stride scheduling technique produces the same sequences as shown by Kubiak [13].

4.2.5. *Insertion sequences.* Recall that $d_1 \leq \ldots \leq d_n$. Consider $n-1$, two-product problems $P_{n-1} = (d_{n-1}, d_n)$, $P_{n-2} = (d_{n-2}, \sum_{j=n-1}^{n} d_j)$, $\ldots$, $P_1 = (d_1, \sum_{j=2}^{n} d_j)$. In each of the problems $P_{n-1}, P_{n-2}, \ldots, P_1$, the first product is the original one, that is $n-1, n-2, \ldots, 1$ and the second product will be the same (fictitious product) for all problems, and denoted by $*$. Let sequences $S_{n-1}, S_{n-2}, \ldots, S_1$ be the optimal solution for problems $P_{n-1}, P_{n-2}, \ldots, P_1$ respectively. They can be obtained by the algorithm described in Section 2.2. Notice that the sequence S_j, $j = n-1, \ldots, 1$, is made up of the product j and $*$. Then the sequence for the original problem is built recursively by first replacing $*$ in S_1 by S_2 to obtain S_1'. Notice that the latter is made up of products 1, 2, and $*$. Next, $*$ are replaced by S_3 in S_1' to obtain a sequence S_1'' made up of products 1, 2, 3 and $*$. Finally, sequence S_{n-1} replaces all the remaining $*$ and thus we obtain a sequence, referred to as the *insertion* sequence, where product i occurs exactly d_i times.

5. Computational Experiments

The computational experiment has been carried out in order to study how the Exchange heuristic affects two metrics, throughput error and response time variability. The experiment consists of applying the Exchange heuristic to earlier described initial sequences and analyzing the main metrics. In this report we include only the main results.

All codes have been implemented in C++ and executed on a X86-based PC with 500 Mhz processor and 128 MB of RAM.

Instances for this experiment are generated by fixing the total number of units D and number of products n , and randomly selecting the number of copies of each product, d_i , which are uniformly distributed. A total of 6650 instances have been run for different (D, n) values, which are as follows:

$$D = 100, n : (3, 10k, k = 1, \ldots, 9)$$
$$D = 500, n : (3, 5, 10, 50k, k = 1, \ldots, 9)$$
$$D = 1000, n : (10, 50, 100k, k = 1, \ldots, 9)$$
$$D = 1500, n : (100k, k = 1, \ldots, 14)$$

5.1. **Computational Results.**

The Exchange heuristic has been designed for the problem of minimization of the response time variability (RTV). Therefore, RTV was considered as a main metric to examine, while the throughput error (TE) for the sequence was included as a secondary metric.

The computational results have shown that, on average, much lower values of final response time variability have been reached after the application of the Exchange heuristic to the initial bottleneck, random and insertion sequences. Moreover, the bottleneck and insertion sequences have produced final sequences with small final throughput errors, although, as expected, the TEs of bottleneck sequences' have increased with respect to the initial TE, while for the insertion sequences they have, on average, decreased.

The Webster's and Jefferson's sequences have resulted in the final sequences with values of RTV comparable to the bottleneck and insertion sequence only for very small values of n. For larger n, on average, their RTV was higher than for the bottleneck, random and insertion sequences. The same has been observed about the change in the values of TE. For small n, the TE has been in the same range as the values of TE for bottleneck and insertion sequences, while TE significantly grew when n increased. For these reasons we do not include the computational results with neither Webster's nor Jefferson's sequences in Tables 1-4.

On average, final TE for bottleneck sequences is lower than the final throughput error for random sequences. For small and large n values the bottleneck sequences result in lower RTV, while on average, random sequences perform slightly better with regard to RTV. To sum up, with respect to the reduced RTV value, the most promising initial sequences were bottleneck, random and insertion sequences.

The change in Response Time Variability (RTV) and Throughput Error (TE) for bottleneck, random and insertion initial sequences, for each value of D is presented in Tables 1-4. The first column contains values of n, the second column includes the averages of RTV for bottleneck sequences for each n (we call them initial RTV). The averages are rounded to integers for ease of understanding. The third column is composed of averages of reduced RTV values (we call them final RTV). The fourth column contains averages of optimum throughput error (initial TE) and the fifth column their final values (final TE). The same order is for the other two initial sequences, that is random and insertion.

Initially, we have compared only two sequences (bottleneck and random) and concluded that bottleneck sequences result in lower response time variability in 54% of cases; random sequences result in lower response time variability in 41% of cases; for 5% of cases the two coincide. The insertion method sequences seem to perform well with regard to both metrics, throughput error and response time variability, but on average, the random sequences still perform better with regard to response time variability. Finally, for very small values of n, the best sequences, resulting in lower RTV, are bottleneck sequences.

The instances with ratio: $\frac{n}{D} \approx 0.1 - 0.5$ can be considered as the worst cases with regard to the reduced RTV value. It has been observed (see tables) that for these instances the values of reduced RTV are

TABLE 1. The exchange improvements on the Bottleneck, Insertion and Random initial sequences for D=100.

n:	Bottleneck				Random				Insertion			
	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE
3	43	26	0.639	0.738	1377	71	4.052	2.726	31	26	0.757	0.766
10	539	86	0.794	1.220	8868	143	4.114	2.454	306	101	1.232	1.188
20	2763	105	0.856	1.522	10424	142	4.178	2.608	498	104	1.120	1.059
30	3447	139	0.880	1.347	15598	120	3.562	1.828	1539	85	1.199	1.063
40	4908	172	0.923	1.337	32549	70	3.183	1.521	3307	71	1.263	1.033
50	8621	94	0.938	1.431	32280	51	2.908	1.546	2609	57	1.223	1.081
60	12100	35	0.946	1.128	31413	31	2.715	1.410	1840	31	0.995	0.990
70	15111	17	0.954	1.182	23834	25	2.616	1.510	687	11	0.990	0.990
80	20029	10	0.958	1.087	17576	15	2.338	1.194	187	4	0.990	0.990
90	20296	3	0.967	0.990	10381	3	1.876	0.995	17	2	0.990	0.990

TABLE 2. The exchange improvements on the Bottleneck, Insertion and Random initial sequences for D=500.

n:	Bottleneck				Random				Insertion			
	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE
3	82	73	0.651	0.677	2959	608	10.605	9.670	108	92	0.724	0.713
5	1081	177	0.748	1.170	27032	1224	9.379	7.010	285	183	1.218	1.170
10	3403	409	0.825	1.200	82248	2441	9.441	7.336	993	443	1.260	1.199
50	63498	1948	0.939	1.593	1000332	2384	6.607	3.534	29925	2256	1.549	1.323
100	153664	1404	0.960	2.036	1492918	1207	7.111	4.797	106108	1313	1.509	1.190
150	234034	4086	0.977	2.637	3372643	871	5.181	2.811	300787	1353	2.153	2.140
200	461694	1497	0.980	1.285	3730674	593	4.487	2.115	344989	1036	1.252	1.032
250	719887	496	0.985	1.433	4069915	345	5.008	3.431	380971	496	1.116	1.016
300	1245583	255	0.987	1.290	3800175	289	4.631	3.041	244113	236	1.031	1.003
350	1578706	86	0.987	1.365	2573962	306	5.136	3.731	88601	56	0.998	0.998
400	3079086	38	0.991	1.075	2788306	137	3.197	1.951	32116	20	0.998	0.998
450	1996168	41	0.991	1.089	1063597	149	3.300	1.646	1580	9	0.998	0.998

much higher than for instances corresponding to other n value. This is also shown by the averages of maximum differences between maximum and minimum distances. For these instances averages of maximum difference in improved sequences are higher than in other cases.

The time it takes to do the exchanges with the Exchange heuristic is almost negligible. It depends on both D and $\frac{n}{D}$. The longest times are reached for instances with ratio $\frac{n}{D} \approx 0.4 - 0.5$ and for instances with a very small difference between d_n and d_1. Average reduction times (clock time) have been: in range of $0 - 1$ sec for $D = 500$, range of $0 - 7$ sec for $D = 1000$, and of $0 - 20$ sec for $D = 1500$.

As a part of our computational experiment we also compared heuristic solutions with optimum obtained by solving the MILP described in

Table 3. The exchange improvements on the Bottleneck, Insertion and Random initial sequences for D=1000.

n:	Bottleneck				Random				Insertion			
	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE
10	17904	601	0.837	1.503	253685	5435	13.327	10.985	1546	625	1.461	1.406
50	180503	3208	0.936	2.004	2706679	7407	9.704	6.219	49101	4242	1.761	1.570
100	636545	4946	0.964	2.289	7130615	6375	8.812	5.564	253319	5866	2.207	1.825
200	1020847	8412	0.978	2.931	13476818	3517	7.662	4.596	873579	5032	2.039	1.977
300	1418033	6120	0.985	3.555	16271934	2266	8.605	6.400	1376361	2717	2.166	2.153
400	3574364	3913	0.991	4.754	35047637	1195	6.339	4.203	3584645	3395	3.512	3.409
500	6232671	1162	0.992	2.327	35083775	862	5.883	4.006	3290617	1313	1.856	1.574
600	7681994	496	0.992	1.648	21386442	962	7.549	5.924	1321738	375	1.049	1.014
700	12206931	179	0.993	1.407	20496095	952	6.716	5.093	706786	130	1.007	1.001
800	18083157	334	0.994	1.466	15952258	697	5.431	3.676	184422	84	1.163	1.071
900	15549203	82	0.996	1.181	8299817	472	4.421	2.685	14832	18	1.004	1.001

Table 4. The exchange improvements on the Bottleneck, Insertion and Random initial sequences for D=1500.

n:	Bottleneck				Random				Insertion			
	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE	Init RTV	Final RTV	Init TE	Final TE
100	885866	8780	0.967	2.005	13310978	13475	10.325	6.394	318918	11125	1.752	1.462
200	3206865	10985	0.977	2.209	28313115	10532	10.292	6.582	1554419	12273	1.732	1.320
300	3243331	13577	0.986	1.833	51059006	5475	8.309	5.019	3611346	9623	1.696	1.166
400	4996223	23905	0.989	1.874	73166821	4048	8.200	5.157	6624499	8586	1.608	1.190
500	6458663	10564	0.991	1.808	82218609	3517	6.980	4.284	8396917	7558	1.446	1.140
600	11085096	4866	0.992	1.534	98681397	2208	7.225	4.847	9990472	4804	1.425	1.070
700	14593169	2051	0.994	1.646	116870944	1628	7.574	5.671	10742017	3627	1.236	1.049
800	24081386	1309	0.994	1.472	110641048	1369	6.394	4.217	8495253	1501	1.147	1.038
900	36781821	1007	0.995	1.316	112399627	1086	5.888	3.884	7107165	1068	1.076	1.016
1000	49367053	298	0.995	1.215	100131267	990	5.423	3.641	4335813	442	1.022	1.003
1100	67535885	150	0.996	1.179	90861641	893	4.934	3.286	2475907	259	1.003	0.999
1200	78982366	116	0.996	1.123	71570932	620	4.542	2.964	966875	70	0.999	0.999
1300	83546651	82	0.997	1.067	52228958	632	4.213	2.737	201881	29	0.999	0.999
1400	60688751	65	0.997	1.038	27742011	360	3.179	1.697	14809	8	0.999	0.999

Section 3. The MILP has been applied to 389 instances, for D=10, 15, 20 and 25 and different values of n. The results are presented in Table 5. For most instances the optimal solutions have been reached within 300 sec. Nevertheless, for some instances, 39 out of 389, the number of nodes of the branch-and-bound tree was increasing dramatically after the first 180 sec. and the execution was aborted, for these instances the best objective function value has been kept, however, these instances

Table 5. The comparison of the exchange heuristic for different initial sequences with optimal solutions.

D	n	[1]	[2]	Insertion			Webster			Jefferson			Bottleneck			Random			[6]
				[3]	[4]	[5]	[3]	[4]	[5]	[3]	[4]	[5]	[3]	[4]	[5]	[3]	[4]	[5]	
10	3	8	0	7	0.500	4	8	0.000	0	8	0.000	0	8	0.000	0	8	0.000	0	8
	4	7	0	5	0.857	4	6	0.286	2	5	0.857	4	5	0.857	4	5	0.857	4	7
	5	7	0	3	1.429	4	2	2.286	6	6	0.571	4	2	2.286	6	4	0.857	2	7
	6	5	0	4	0.400	2	3	0.800	2	4	0.400	2	5	0.000	0	5	0.000	0	5
	7	3	0	3	0.000	0	3	0.000	0	3	0.000	0	2	0.667	2	3	0.000	0	3
	8	2	0	2	0.000	0	2	0.000	0	2	0.000	0	2	0.000	0	2	0.000	0	2
	9	1	0	1	0.000	0	1	0.000	0	1	0.000	0	1	0.000	0	1	0.000	0	1
for D=10		33	0	25			25			29			25			28			33
15	3	12	0	10	1.333	10	10	0.667	6	11	0.500	6	11	0.167	2	9	0.833	4	12
	4	16	0	12	1.375	8	13	1.000	8	11	1.375	6	14	0.875	8	14	0.375	4	16
	5	14	0	6	3.571	16	8	1.286	8	8	1.429	4	8	1.000	4	10	0.571	2	12
	7	13	0	5	3.231	16	1	7.385	16	2	4.308	10	1	5.846	16	5	2.769	10	7
	9	9	0	7	0.667	4	3	3.778	12	3	2.444	6	3	4.000	14	7	0.444	2	9
	11	4	0	4	0.000	0	3	0.500	2	3	0.500	2	3	0.500	2	2	2.000	6	4
	13	2	0	2	0.000	0	2	0.000	0	2	0.000	0	2	0.000	0	2	0.000	0	2
for D=15		70	0	46			40			40			42			49			62
20	3	19	0	15	0.947	8	15	0.632	4	14	0.842	4	16	0.421	4	12	2.105	16	18
	4	17	0	11	1.765	14	11	1.176	8	9	1.529	8	13	0.471	2	11	0.824	4	14
	5	17	3	6	3.429	14	7	2.286	14	3	3.714	12	7	2.000	14	6	3.286	12	10
	6	20	3	3	6.471	20	1	6.000	18	2	4.471	10	4	4.941	14	10	2.000	8	12
	7	18	3	5	4.400	12	1	7.067	14	0	6.400	10	2	6.267	14	4	4.267	10	7
	8	18	0	5	3.333	8	2	8.333	18	2	5.556	16	4	7.000	18	5	3.444	10	8
	9	17	0	5	3.765	16	0	11.760	34	1	5.882	16	1	10.940	34	4	4.940	24	8
	11	13	0	7	1.846	8	2	6.769	18	2	5.231	24	2	4.615	8	2	3.692	8	7
	13	9	0	7	0.444	2	3	2.667	8	1	3.556	6	4	1.556	4	3	1.778	4	8
	15	6	0	5	0.333	2	4	0.667	2	4	0.667	2	3	1.000	2	4	1.000	4	5
	18	2	0	2	0.000	0	2	0.000	0	2	0.000	0	2	0.000	0	2	0.000	0	2
for D=20		156	9	71			48			40			58			63			99
25	3	18	2	13	1.625	12	15	0.625	10	15	0.625	10	16	0.000	0	9	2.250	14	16
	5	17	8	3	4.444	14	3	4.667	14	4	4.000	10	4	4.000	14	3	4.444	16	7
	7	20	14	2	4.000	14	1	5.000	8	0	10.000	16	1	5.333	10	0	5.000	8	2
	9	18	6	2	9.167	30	0	14.670	22	0	11.500	20	0	13.500	22	1	6.667	18	3
	11	18	0	0	10.110	38	0	23.890	52	1	10.890	20	0	19.560	52	2	7.000	16	3
	13	15	0	3	5.067	18	1	16.000	36	1	11.200	28	0	14.530	36	2	4.933	16	5
	16	13	0	6	1.538	6	1	7.385	20	3	4.769	12	3	4.615	12	5	2.923	8	9
	19	8	0	6	0.500	2	5	0.750	2	4	1.250	4	4	1.250	4	5	0.750	2	7
	22	3	0	3	0.000	0	3	0.000	0	3	0.000	0	3	0.000	0	3	0.000	0	3
for D=25		130	30	38			29			31			31			30			55
Total		389	39	180			142			140			156			170			249

(1) Number of instances
(2) Number of instances excluded from examination
(3) Number of instances for which the heuristic found an optimal solution
(4) Average difference between the heuristic and optimal values
(5) Maximum difference between the heuristic and optimal values
(6) Number of instances for which some heuristic found an optimal solution

have been excluded from the comparison statistics. The highest average and maximum differences between heuristic and optimal solutions have been achieved whenever $0.3 \leq \frac{D}{n} \leq 0.6$. The sequences obtained by the Insertion method have been reduced to optimal ones most often, closely followed by Random sequences and bottleneck sequences. The heuristics found optimum for 249 instances out of the 350 for which the optimum could be found.

6. Conclusions and Further Research

There a number of possible directions to proceed in order to search for improvement in response time variability. We think the following are most natural for the problem and promising.

- Exploit the properties of the number decomposition graphs introduced in Section 2 and the sufficient conditions for local optima defined by the exchange procedure from Section 4 in the Constraint Logic programming approach.
- Use the randomized greedy algorithm, in the form of GRASP, see Feo and Resende [7]. This would build a sequence, one product at a time, by taking into account the increase in the response time variability, first, and the throughput error, second, associated with the candidate product. The choice could be randomized by randomly choosing one of the best candidates, but not necessarily the best one. The complete sequence would then by subjected to the exchange procedure described in Section 4.
- Use multi-start.
- Find lower bounds on the response time variability for partial solutions and use them along with the upper bounds obtained by the heuristics of Section 4 to prune the states of the Dynamic Programming algorithm defined in Section 2.

References

[1] Altman, E., B. Gaujal and A. Hordijk(2000), "Multimodularity, Convexity and Optimization Properties", *Math. of Oper. Research*, Vol. 25, pp. 324-347.

[2] Anily, S., C.A. Glass and R. Hassin (1998), The scheduling of maintenance service, *Discrete Applied Mathematics* **82**, 27 42.

[3] Balinski, M. L., Young, H.P. (1982): Fair Representation. Yale University Press, New Haven, CT.

[4] Bar-Noy, A., R. Bhatia, J. Naor, and B. Schieber (1998) Minimizing service and operation costs of periodic scheduling. In *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms SODA, 1998*, 11-20.

[5] Bautista, J., R. Companys and A. Corominas (1996) A note on the relation between the product rate variation (PRV) and the apportionment problem. *Journal of Operational Research Society* **47** 1410-1414.

[6] Bautista, J., Companys, R., Corominas, A. (1997): Modelling and solving the production rate variation problem, *TOP* , vol. 5, 2, 221-239.

[7] Feo, T. and M. Resende (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters* 8, 6771.

[8] Dong, L., R. Melhem and D. Mosse(1998), "Time Slot Allocation for Real-time Messages with Negotiable Distance Constrains Requirements", The Real-time Technology and Application Symposium, RTAS, Denver, CO.

[9] Han, C.C., K.J. Lin, and C.J. Hou(1996), "Distance-Constrained Scheduling and Its Applications in Real-Time Systems," *IEEE Trans. on Computers*, Vol. 45, No. 7, pp. 814-826.

[10] Kubiak, W., and S.P. Sethi (1991) A Note on "Level schedules for mixed-model assembly lines in just-in-time production systems," *Management Science* **37** 121-122.

[11] Kubiak, W. (1993): Minimizing variations of productions rates in just-in-time systems: A survey, *Eur. J. Opl. Res.*, 66, 259-271.

[12] Kubiak, W. (2003) On small deviations conjecture, *Bulletin of the Polish Academy of Sciences* **51** 189-203.

[13] Kubiak, W. (2004) Fair Sequences, In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Joseph Y-T. Leung (Edt.) Chapman and Hall/CRC, to appear.

[14] Monden, Y. (1983) *Toyota Production Systems* Industrial Engineering and Management Press, Norcross, GA.

[15] Moreno, N. (2002): Solving the Product Rate Variation Problem (PRVP) of large dimensions as an assignment problem, Doctoral Thesis, DOE, ETSEIB-UPC.

[16] Steiner, G., Yeomans S. (1993): Level Schedules for Mixed-Model, Just-in-Time Processes, *Management Science*, vol. 39, no. 6, 728-735.

[17] Waldspurger, C.A., Weihl, W.E. (1995) Stride Scheduling: Deterministic Proportional-Share Resource Management. Technical Memorandum MIT/LCS/TM-528 MIT Laboratory for Computer Science Cambridge, MA 02139