UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Heuristics for the Response Time Variability problem

Albert Corominas, Wieslaw Kubiak and Rafael Pastor.

EOLI – Enginyeria d'Organització i Logística Industrial

IOC-DT-P-2009-03
Desembre 2009

Institut d'Organització i Control
de Sistemes Industrials

Heuristics for the Response Time Variability problem[†]

Albert Corominas[1], Wieslaw Kubiak[2] and Rafael Pastor[1,*]
[1]Institute of Industrial and Control Engineering (IOC), Universitat Politècnica de Catalunya, Barcelona, Spain
[2]Faculty of Business Administration, Memorial University, St. John's, Canada
albert.corominas@upc.edu, wkubiak@mun.ca, rafael.pastor@upc.edu

Abstract

The recently introduced Response Time Variability Problem (RTVP) is a scheduling problem that has a broad range of real-life applications, for example, to sequence the models to be produced on a mixed-model assembly line. Previous studies include heuristic algorithms and mathematical programming models, whose practical limit for obtaining optimal solutions is around 40 units to be scheduled. In this paper, we propose and test new algorithms that combine heuristic procedures for obtaining initial sequences and several local optimization procedures.

Keywords: response time variability, heuristics, scheduling, fair sequences

1. Introduction

The Response Time Variability Problem (RTVP) can be formulated as follows. Let n be the number of products/jobs/messages (in this paper we will only use the term "product"). Let d_i be the number of units of product i $(i=1,...,n)$ and $D=\sum_{i=1}^{n} d_i$ the total number of units. These units have to be assigned to D consecutive time slots of equal length, thus constituting a sequence $s=s_1 s_2 ... s_D$ of length D, where s_j is the product that occupies the time slot j. For all the products with $d_i \geq 2$, let t_k^i be the difference between the numbers of the slots occupied by units $k+1$ and k of product i, which we call the distance between these units. Let us assume that s_1 immediately follows s_D (i.e., it is a circular sequence). Therefore, $t_{d_i}^i$ is the distance between the first unit of product i in a cycle and the last unit of the same product in the preceding cycle. Let $\overline{t_i}=D/d_i$ be the average distance between two consecutive units of product i. For all the products with $d_i=1$, t_1^i is equal to $\overline{t_i}$. The objective is to minimise $RTV=\sum_{i=1}^{n}\sum_{k=1}^{d_i}\left(t_k^i-\overline{t_i}\right)^2$, which is a weighted variance with weights equal to demands: $RTV=\sum_{i=1}^{n} d_i \cdot Var_i$, where $Var_i=\frac{1}{d_i}\cdot\sum_{k=1}^{d_i}\left(t_k^i-\overline{t_i}\right)^2$.

This problem has been recently introduced in the literature (Corominas *et al*., 2007 and

[†] Supported by the Spanish MCyT project DPI2007-61905, co-financed by ERDF, and by Natural Sciences and Engineering Research Council of Canada Grant OGP0105675.
[*] Corresponding author: Rafael Pastor, Institute of Industrial and Control Engineering (IOC), Av. Diagonal 647 (edif. ETSEIB), planta 11, 08028 Barcelona, Spain; Tel. + 34 93 401 17 01; Fax. + 34 93 401 66 05; e-mail: rafael.pastor@upc.edu

Corominas *et al.*, 2009), although the concept of Response Time Variability appears in Waldspurger and Weihl (1995) and may arise in many real situations (see Corominas *et al.*, 2007) such as *isochronous* applications or scheduling on a mixed-model, just-in-time assembly line production (Monden, 1983). Herrmann uses the response time variability problem in scheduling of waste collection (Herrmann, 2007). The scheduling of advertising slots for television (Bollapragada *et al.*, 2004 and Brusco. 2008) leads to a problem closely related to RTVP and also does the design of sales catalogues (Bollapragada *et al.*, 2004).

The response time variability is a convenient metric for most of these problems, which are often considered as distance-constrained scheduling problems (Han et al., 1996; Anily et al., 1998; Dong et al. 1998; Altman et al. 2000; Bar-Noy et al. 2002). The distance-constrained models, however, suffer from a serious practical disadvantage, which is that there may not be a feasible solution that respects the distance constraints and at the same time ensures that tasks are done at given rates. The total response time variability metric instead, avoids the feasibility problem but at the same time preserves the main idea of having any two consecutive tasks at a distance that remains as constant as the existing resources and other competing jobs permit.

Several other measures have been proposed for the regularity or fairness of a sequence of products on assembly lines, based either on the difference between ideal and actual productions (Miltenburg 1989; Kubiak 1993; Steiner and Yeomans 1993) or on the difference between ideal and actual production dates (Inman and Bulfin 1991; Bautista *et al*. 1997). The new measure of regularity is easier to understand by practitioners, since it use only a simple concept, the distance, and has the characteristic that the value of the measure does not depend on the position of those products of which an only unit should be sequenced.

Corominas *et al*. (2007) studied the computational complexity of the RTVP and proved that it is NP-hard. A simple optimization algorithm was proposed for the two-product case. In order to solve the RTVP to optimality, a special case of the quadratic assignment problem recast as a quadratic integer programming (QIP) problem was considered; the QIP is linearized, but the practical limit for obtaining optimal solutions is 25 units to be scheduled (i.e., $D = 25$). Finally, five heuristics and a local optimization exchange procedure were presented. The results showed that, on average, much lower RTV values were reached whenever the exchange procedure ran on the bottleneck, insertion and random sequences.

Corominas *et al*. (2009) solved the RTVP to optimality by means of mathematical programming. Some special features of the RTVP were analyzed and some new ideas for improving the MILP presented in Corominas *et al*. (2007) were proposed and tested, thus solving to optimality instances with up to 40 units to be scheduled. Nevertheless, heuristic procedures were still necessary to solve larger instances.

In this paper, we propose and test two new greedy algorithms and compare them with the previously published heuristic algorithms and with some variants of them that we propose in this paper. Having a good heuristic algorithm is useful either to obtain quickly good solutions to the RTVP or to provide an upper bound to exact algorithms.

The rest of this paper is set out as follows. First, Section 2 presents heuristic procedures

for obtaining initial sequences. Section 3 introduces the exchange procedures applied to obtain local optimums. Section 4 presents the results of the subsequent computational experiment. Finally, Section 5 is devoted to conclusions and possible lines for future research.

2. Heuristic procedures for obtaining initial sequences

In this section, we describe the heuristic procedures used to obtain initial sequences. We will call G the set of products such that $d_i = 1$.

2.1 The five heuristic procedures from Corominas et al. (2007) and four variants of them

The five heuristics presented in Corominas *et al.* (2007) are described in the Appendix. The two apportionment methods suffer from the fact that they place together, in the middle of the sequence, all the units of products belonging to G; therefore, when the number of such products is high the distance between a pair of units of each of the other products is obliged to take an excessively great value. This explains that, although the quality of the solutions provided by *We* and *Je* is often good when $|G| \leq 1$, it is very poor when $|G|$ is high and there are products with short average distances between its units. When this happens, apportionment algorithms yield better results if (such as is proposed in Waldspurger and Weihl, 1995) all the products in $G\left(\text{such that} |G| \geq 2\right)$ are combined in a single fictitious product g with $d_g = |G|$ which does not intervene in the objective function.

Applying this idea to *We* and *Je* we obtain two more heuristics that we call procedures We/d_g and Je/d_g. Obviously, algorithms $*/d_g$ are equivalent to * when $|G| \leq 1$.

In fact, out of the five heuristics in Corominas *et al.* (2007), only *In* relies on specific properties of the RTVP. It seemed to be worth to explore other variants of using the idea of insertion as in *In*, what gave raise to algorithms *In2* and *In3*.

In2 is identical to *In*, except that the order of the products is reversed. *In3* is identical to *In2*, except that, when solving the two-product problems, the first copy of the product with the less number of copies is assigned to the first position and the remaining copies are placed in the sequence $D \bmod d_i$ times with a distance $\left\lceil D/d_i \right\rceil$ to the last position assigned and $d_i - D \bmod d_i$ times with a distance $\left\lfloor D/d_i \right\rfloor$ to the last position assigned.

So far, nine heuristic procedures have been exposed, including the five heuristic previously published and four variants of them. Next two subsections describe three new heuristics conceived specifically to solve RTVP.

2.2. Adaptive Webster (AWe *and* AWe/d_g *)*

Parametric apportionment algorithms try to allocate the units without regard for the position occupied by the preceding unit of the same product. However, concerning RTVP, the best (locally) position for a unit of a product is the position that is at the most convenient distance (as near as possible to the average distance) from that corresponding to its preceding unit.

The algorithm that we call AWe derives from the previous remark. When a unit of a product is allocated, the ideal position for the next unit of the same product is updated. For each product $i\ (i=1,...,n)$, we calculate an index in_{ij} associated with the next unit j, $j=1,...,d_i$, to be sequenced, and the unit j of product i with the smallest in_{ij} value is sequenced. Let $in_{i1}=\dfrac{D}{2\cdot d_i}$ (whose ceiling is the ideal position of unit 1 of product i). Let k_{i1} be the position in which the first unit of product i is sequenced. Thus, $in_{i2}=k_{i1}+\dfrac{D}{d_i}$, and finally $in_{ij}=in_{i,j-1}+\dfrac{D}{d_i}$ $(j\geq 3)$. In the event of a tie between q products, such that $d_{(1)}\leq ...\leq d_{(q)}$: the first product to be sequenced is product (1), the last is product (2), the second is product (3), the second-to-last is product (4) and so on (that is: $d_{(1)},d_{(3)},d_{(5)},...d_{(6)},d_{(4)},d_{(2)}$). The objective of this tie-breaking rule is to prevent all the products in with $d_i=1$ from being sequenced together.

AWe/d_g is the combination of AWe with the use of the fictitious product g.

2.3. Opportunity cost (Oc *)*

Consider that a partial sequence has built up and including position k. Given a product, if we compare the cost of allocating a unit of it to position $k+1$ and the cost of allocating it to position $k+2$, the difference between the latter and the former can be called the opportunity cost of allocating the unit to position $k+2$ (instead of allocating it to position $k+1$). It is reasonable, then, to allocate to position $k+1$ the product with a greater opportunity cost. As this cost cannot be calculated exactly without an optimizing algorithm, the decision can be taken on the basis of an estimation of it, i.e., the value of parameter ψ_i, equal to $PLB(i,k+2)-PLB(i,k+1)$, where $PLB(i,k+2)$ and $PLB(i,k+1)$, are lower bounds on the objective function when a unit of product i is placed at positions $k+2$ (without being placed at position $k+1$) or at position $k+1$, respectively.

In the explanation, that follows, of the way to calculate $PLB(i,k+1)$, indexes $(i,k+1)$ are omitted for the sake of simplicity.

As introduced in Corominas *et al*. (2007), a decomposition vector of D into d_i components can be defined as follows: $\lambda_i=(\lambda_1,...,\lambda_{d_i})$ of d_i positive integers that add

up to D and $\lambda_1 \geq ... \geq \lambda_{d_i}$. The components of vector λ_i are the distances between the d_i units of product i. Thus, the minimum value of RTV for product i, RTV_i, can be obtained when $D \bmod d_i$ and $d_i - D \bmod d_i$ components of λ_i are equal to $\left\lceil \frac{D}{d_i} \right\rceil$ and $\left\lfloor \frac{D}{d_i} \right\rfloor$, respectively. For example, let $D = 24$, $n = 4$, $d = (9,7,5,3)$ and $\bar{t} = (2.67, 3.43, 4.8, 8)$. The decomposition vectors $\lambda_1 = (3,3,3,3,3,3,2,2,2)$, $\lambda_2 = (4,4,4,3,3,3,3)$, $\lambda_3 = (5,5,5,5,4)$ and $\lambda_4 = (8,8,8)$ provide the minimum values of RTV_i $(i = 1,...,4)$. A lower bound on the value of RTV_i, $RTVLB_i$, and a lower bound on the value of RTV, $RTVLB$, can be defined as follows: $RTVLB_i = (D \bmod d_i) \cdot \left(\left\lceil \frac{D}{d_i} \right\rceil - \bar{t}_i \right)^2 + (d_i - D \bmod d_i) \cdot \left(\left\lfloor \frac{D}{d_i} \right\rfloor - \bar{t}_i \right)^2$ and $RTVLB = \sum_{i=1}^{n} RTVLB_i$:

$$RTVLB = \left[6 \cdot (3 - 2.67)^2 + 3 \cdot (2 - 2.67)^2 \right] + \left[3 \cdot (4 - 3.43)^2 + 4 \cdot (3 - 3.43)^2 \right]$$
$$+ \left[4 \cdot (5 - 4.8)^2 + 1 \cdot (4 - 4.8)^2 \right] + \left[0 \cdot (8 - 8)^2 + 3 \cdot (8 - 8)^2 \right] = 4.51$$

In this case, however, a lower bound, PLB, is needed for a partial solution, that is, a solution in which one unit of product has been assigned to each of the first k positions.

A bound for a partial solution, PS, can be obtained by adding, for all the products with $d_i \geq 2$, the sum of RTV_{PS} (the value associated with the distances between the units of the product allocated in [1,…, k], if any) and RTV_{REM} (a bound corresponding to the assignment of the remaining units, if any, to the free positions).

Let i be a product, with $d_i \geq 2$, whose units have not all been assigned in the partial solution PS. Three cases must be distinguished:

- Case 1. No unit of product i has been assigned in the k time slots: We must distribute D time slots among d_i distances between two units of product i, guaranteeing that one distance be greater than or equal to $k+1$.
- Case 2. Only one unit of product i has been assigned to position h $(\leq k)$: We must distribute D time slots among d_i distances, guaranteeing that one distance be greater than or equal to $k - h + 1$ and another be greater than or equal to h.
- Case 3. p units of product i have been assigned in the k time slots, the first in the sequence in position h_f and the last one in position h_l: We must distribute $D - h_l + h_f$ time slots among $d_i - p + 1$ distances, but guarantee that one distance be greater than or equal to $k - h_l + 1$ and another be greater than or equal to h_f. Case 2 can be reduced to Case 3 taking into account that $h_f = h_l = h$ and $p = 1$. Case 1 can

be reduced to Case 3 taking into account that $h_f = h_l = h = 0$ and $p = 1$.

Thus, the problem consists in distributing $D - h_l + h_f$ "units of distance" among $d_i - p + 1$ distances t_j^i $(j = 1,..., d_i - p + 1)$, taking into account that two distances are lower bounded by $k - h_l + 1$ and h_f, respectively, and the others are lower bounded by 1, with the objective of minimizing a function of the discrepancy between the distances and the average distance $\overline{t_i}$. Thus, it is the apportionment problem with lower bounds. Bautista *et al*. (2001) propose a general optimization procedure for a convex, nonnegative (symmetric or not) discrepancy function and such that $f(0) = 0$. For the discrepancy function considered here (the quadratic discrepancy), the resulting procedure is as follows:

$t_1^i = k - h_l + 1$
$t_2^i = h_f$
for $j = 3$ to $d_i - p + 1$
    $t_j^i = 1$
next j
for $j = 1$ to $D - k + p - d_i$ $\quad \left(= D - \left(h_f + k - h_l + 1\right) - \left(h_l - h_f\right) - \left(d_i - p - 1\right)\right)$
    find $s^* \left| t_{s^*}^i = \min_{1 \le s \le \max(2, d_i - p + 1)} \left(t_s^i\right) \right.$
    $t_{s^*}^i = t_{s^*}^i + 1$
next j

For the instance defined by $n = 4$ and $d = (9,7,5,3)$ and the partial solution $PS = (1,3,2,1,1,3,3,3,1,,,,,,,,,,,,)$, we have:

$$RTV_{PS} = \left[1 \cdot (4 - 2.67)^2 + 1 \cdot (3 - 2.67)^2 + 1 \cdot (1 - 2.67)^2\right] + \left[1 \cdot (4 - 4.8)^2 + 2 \cdot (1 - 4.8)^2\right] = 34.187$$

And, applying the procedure described above, the distances (3,3,3,3,2,2), (7,3,3,3,3,3,2), (9,9) and (10,7,7) are obtained for products 1, 2, 3 and 4, respectively. The value corresponding to these distances, RTV_{REM}, is:

$$RTV_{REM} = \left[4 \cdot (3 - 2.67)^2 + 2 \cdot (2 - 2.67)^2\right] + \left[1 \cdot (7 - 3.43)^2 + 5 \cdot (3 - 3.43)^2 + 1 \cdot (2 - 3.43)^2\right]$$
$$+ \left[2 \cdot (9 - 4.8)^2\right] + \left[1 \cdot (10 - 8)^2 + 2 \cdot (7 - 8)^2\right] = 58.327$$

And, finally, $PLB = RTV_{PS} + RTV_{REM} = 34.187 + 58.327 = 92.51$

When breaking ties, products are selected in descending order of i.

3. Local optimization procedures

This section describes twelve position-exchange procedures for local optimisation that have been applied to the heuristic initial sequences introduced in Section 2.

The twelve procedures result from combining three neighbourhoods (*N1*: swapping two consecutive units; *N2*: swapping any pair of units; *N3*: a unit of a product i is removed from the position it occupies and inserted between a pair of consecutive positions provided that there is no another unit of i between the initial position of the unit and the position in which is inserted); two rules for replacing the current solution with a new one (*R1*: is replaced with the first neighbour that is better that current solution; *R2*: is replaced with the best neighbour, provided it is better than the current solution) and two stopping rules:

F1: When there is not a neighbour better than the current solution, the algorithm considers, as a candidates for replacing the current solution, the neighbours for which the net improvement is 0 and, to avoid cycling, such that the maximum distance is not increased for either of two products being exchanged and at least one of the maximum distances actually decreases. If there are not such neighbours, the algorithm stops.

F2: This stopping rule differs from the preceding one only in that considers also as candidates the neighbours for which the net improvement is 0 and such that the minimum distance does not decrease for either of the two products being exchanged and, moreover, at least one of the minimum distances actually increases.

The local optimisation procedure proposed in Corominas *et al*. (2007) is *N1/R1/F1*.

4. Computational experiment

A computational experiment was carried out to evaluate the effectiveness of the exposed in Section 2 and 3.

600 instances were used. They are generated as follows: 200 instances for three combinations of D and n. The combinations of D and n are as follows: a) for instances 001 to 200, D is randomly selected from a uniform distribution between 25 and 50 and n is randomly selected from a uniform distribution between 3 and 15; b) for instances 201 to 400, D is randomly selected from a uniform distribution between 50 and 100 and n is randomly selected from a uniform distribution between 3 and 30; and c) for instances 401 to 600, D is randomly selected from a uniform distribution between 100 and 200 and n is randomly selected from a uniform distribution between 3 and 65. For all the instances, d_i values are randomly selected from a uniform distribution between 1 and $\left\lceil \frac{D-n+1}{2.5} \right\rceil$; since the d_i values have to fulfil $\sum_{i=1}^{n} d_i = D$, if $\sum_{i=1}^{n} d_i < D$ a product i is randomly selected and $d_i = d_i + 1$ (until $\sum_{i=1}^{n} d_i = D$) and if $\sum_{i=1}^{n} d_i > D$ a product i with $d_i > 1$ is randomly selected and $d_i = d_i - 1$ (until $\sum_{i=1}^{n} d_i = D$).

The data were ordered as follows: $d_1 \geq ... \geq d_n$. Moreover, when various products with $d_i = 1$ are combined in a single fictitious product g, it is placed in the last position of the list for procedures xxx / d_g.

The heuristic algorithms were implemented in JAVA and the computational experiment was carried out on a PC Pentium IV, CPU 3.40 GHz with 512 Mb of RAM.

For each of the 600 instances, we had 12 algorithms for the initial sequence and 12 local optimisation procedures. Therefore, 86,400 tests were carried out.

The results were analyzed by considering the set of 600 instances (which we will call TS), among them the set of 462 instances with $|G| \geq 2$ (which we will call SG) and the set of 138 instances with $|G| \leq 1$ (which we will call $S\overline{G}$).

First, we analyzed the results obtained by calculating the initial sequences using the proposed procedures (that is, without using an exchange procedure). Table 1 shows the results obtained for the 600 instances of TS and Table 2 shows the results obtained for sets SG and $S\overline{G}$. The notation used in Tables 1 and 2 is as follows: $\overline{RTV}$, the average of the RTV values for the instances considered; NBS, the number of best solutions obtained; $\Delta_{\max}$, Δ_{ave}, $\Delta_{\min}$, maximal, average and minimal deviation from the best value, respectively (the deviation of a $RTV_{a\lg}$ value is calculated as $100 \cdot \left(RTV_{a\lg} - RTV_{\min}\right) \Big/ RTV_{\min}$); and CT, average computing time (in seconds).

	$\overline{RTV}$	NBS	$\Delta_{\max}$	Δ_{ave}	$\Delta_{\min}$	CT
Bo	1702.7	22	47788	1609	0	0.0301
Ra	10113.8	0	174837	9166	653	0.0005
We	3170.1	29	141308	3223	0	0.0043
We / d_g	185.6	51	2400	121	0	0.0024
Je	3425.2	19	157809	3560	0	0.0023
Je / d_g	240.5	21	2400	181	0	0.0021
In	362.6	32	9499	336	0	0.0013
$In2$	3484.6	25	146775	3533	0	0.0006
$In3$	3872.8	18	168428	3974	0	0.0007
AWe	3106.9	31	133872	3134	0	0.0034
AWe / d_g	172.2	55	2500	106	0	0.0028
Oc	100.6	499	160	4	0	0.3279

Table 1. Results, without using a local optimisation procedure, for the set TS.

	SG						$S\bar{G}$					
	$\overline{RTV}$	NBS	$\Delta_{\max}$	Δ_{ave}	$\Delta_{\min}$	CT	$\overline{RTV}$	NBS	$\Delta_{\max}$	Δ_{ave}	$\Delta_{\min}$	CT
Bo	2188.9	0	47788	2060	16	0.0301	75.0	22	1551	100	0	0.0123
Ra	12620.3	0	174837	10806	793	0.0005	1722.6	0	21067	3678	653	0.0002
We	4101.0	0	141308	4171	10	0.0053	53.7	29	454	51	0	0.0009
We/d_g	225.0	22	2400	142	0	0.0028	53.7	29	454	51	0	0.0009
Je	4430.2	0	157809	4602	16	0.0028	60.8	19	551	70	0	0.0003
Je/d_g	294.2	2	2400	215	0	0.0024	60.8	19	551	70	0	0.0008
In	399.9	24	4312	298	0	0.0017	237.8	8	9499	464	0	0.0001
In2	4503.7	0	146775	4558	19	0.0007	72.7	25	914	100	0	0.0001
In3	5009.1	0	168428	5135	38	0.0009	68.7	18	746	84	0	0.0001
AWe	4019.6	0	133872	4058	18	0.0040	51.1	31	373	40	0	0.0015
AWe/d_g	208.3	24	2500	126	0	0.0035	51.1	31	373	40	0	0.0006
Oc	119.1	398	160	3	0	0.4155	38.6	101	116	7	0	0.0346

Table 2. Results, without using a local optimisation procedure, for sets SG and $S\bar{G}$.

For the 600 instances of set TS, the procedure that yields better solutions is *Oc* and this is true, too, for sets SG and $S\bar{G}$. *Oc* gives the best values of $\overline{RTV}$, NBS, $\Delta_{\max}$, Δ_{ave} and $\Delta_{\min}$. However, the average computing time is many times greater for *Oc* than for the other procedures, especially for the instances belonging to SG. However, the solutions obtained with this heuristic require, when the exchange procedures are applied, less time than solutions obtained with others.

Next, we analyze the overall results (see Table 3) of the local optimisation procedures (that is, the average results obtained by applying them to the sequences obtained with all the heuristics described in Section 2) for set TS.

	$\overline{RTV}$	NBS	$\Delta_{\max}$	Δ_{ave}	$\Delta_{\min}$	CT
N1/R1/F1	120.4	625	1676	238	0	1.35
N1/R1/F2	119.7	629	1672	236	0	1.36
N1/R2/F1	148.7	623	2094	315	0	1.86
N1/R2/F2	146.8	634	2091	311	0	1.88
N2/R1/F1	114.5	640	1722	222	0	25.32
N2/R1/F2	114.1	643	1724	222	0	25.43
N2/R2/F1	113.1	640	1800	216	0	11.70
N2/R2/F2	112.3	638	1800	215	0	11.91
N3/R1/F1	53.01	1162	603	56	0	72.98
N3/R1/F2	53.00	1127	602	57	0	73.05
N3/R2/F1	63.5	925	738	81	0	18.66
N3/R2/F2	63.1	924	738	82	0	18.92

Table 3. Overall results for the 12 exchange procedures for set ST.

Although *F2* obtains slightly better $\overline{RTV}$ values than *F1* does, the number of best solutions obtained with *F1* was slightly greater than that obtained with *F2*. The calculation times for the two stopping rules were very similar. Exchange algorithm *R1* provided usually better results than *R2* and required generally a longer calculation time. The use of neighbourhood *N3* provides the most significant improvements, although it requires a longer calculation time.

Finally, we analyze the obtained results for the 144 combinations that result from applying the 12 local optimization procedures to each of the 12 initial sequences. For simplicity, we only show the best results. Concerning $\overline{RTV}$, NBS and Δ_{ave} of set TS, the best local optimization was always $N3/R1/F1$, followed closely by $N3/R1/F2$. However, as shown in Table 4, none of the initial sequences consistently provided the best value for these parameters.

	$N3/R1/F1$				$N3/R1/F2$			
	$\overline{RTV}$	NBS	Δ_{ave}	CT	$\overline{RTV}$	NBS	Δ_{ave}	CT
AWe/d_g	42,46	**155**	**27,24**	12,06	42,40	**154**	**27,29**	12,17
Oc	**42,14**	134	27,26	5,18	**42,26**	122	28,54	5,25

Table 4. Results for set TS.

For set SG, the best local optimization procedure was again $N3/R1/F1$, followed closely by $N3/R1/F2$. In this case, the initial sequence provided by Oc gave the best value for these parameters (see Table 5).

	$N3/R1/F1$				$N3/R1/F2$			
	$\overline{RTV}$	NBS	Δ_{ave}	CT	$\overline{RTV}$	NBS	Δ_{ave}	CT
AWe/d_g	45,08	**92**	30,17	15,60	45,06	**91**	**30,36**	15,74
Oc	**44,54**	**92**	**29,96**	6,69	**44,69**	79	31,56	6,78

Table 5. Results for set SG.

For set $S\overline{G}$, all of the exchange procedures yielded very similar results and none of them was consistently the best for all analyzed values. $N1/R2/F2$ and Oc provided the best value for $\overline{RTV}$ (33,25); local optimization procedures $N2/R1/F2$ and $N3/R2/F2$ and heuristics AWe and AWe/d_g (which, obviously, are equivalent for the instances belonging to $S\overline{G}$) provided the best value for NBS (64). Finally, $N1/R2/F1$ and Oc provided the best Δ_{ave} value (14,07).

When $|G| \geq 2$, we recommend using the heuristic procedure obtained by applying the local optimization procedure $N3/R1/F1$ to the initial sequence obtained by Oc. When there are less than two products with $d_i = 1$ $(|G| \leq 1)$, we recommend applying $N1/R2/F2$ to the initial sequence obtained by Oc in order to obtain the best value of $\overline{RTV}$, applying $N2/R1/F2$ to the initial sequence given by AWe in order to obtain the best value of NBS and applying $N1/R2/F1$ to the initial sequence provided by Oc in order to obtain the best Δ_{ave} value. When the previous recommendations are applied, an overall $\overline{RTV}$ value of 41,94 is obtained, as opposed to $\overline{RTV}$ value of 42,14 obtained when all 600 instances are considered together (Table 4).

5. Conclusions

This paper discusses ways of solving the Response Time Variability Problem (RTVP) using heuristic algorithms. The RTVP, recently defined in the literature, is a scheduling

problem with a broad range of real-life applications that is very difficult to solve to optimality. A previous study developed a RTVP position-exchange heuristic to improving greedy initial sequences. Mathematical programming models for solving it to optimally have also been tested (although the practical limit for obtaining optimal solutions is around 40 units to be scheduled).

This paper proposes and tests new heuristic algorithms that combine heuristic procedures for obtaining initial sequences with several exchange procedures.

The results of a computational experiment show that the proposed heuristic procedures for obtaining initial sequences and exchange procedures are superior to the heuristic algorithms published in the literature. We recommend the following: first, the instances to be solved should be classified based on whether the number of products with $d_i = 1$ is ≥ 2. Next, if $|G| \geq 2$, use $Oc + N3/R1/F1$. If $|G| \leq 1$, use $Oc + N1/R2/F2$ in order to obtain the best value of $\overline{RTV}$, $AWe + N2/R1/F2$ to obtain the best value of NBS and $Oc + N1/R2/F1$ to obtain the best Δ_{ave} value.

Future research may focus on designing a branch and bound procedure for the RTVP.

References

[1] Altman E, Gaujal B, Hordijk A. (2000). Multimodularity, convexity and optimization properties. *Mathematics of Operations Research* 25; 324-347.

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

[3] Balinski ML, Young HP. (1982). *Fair representation*. Yale University Press, New Haven, CT.

[4] Bar-Noy A, Bhatia R, Naor J, Schieber B. (2002). Minimizing service and operation costs of periodic scheduling. *Mathematics of Operations Research* 27, 518-544.

[5] Bautista J, Companys R, Corominas A. (1997). Modelling and solving the production rate variation problem. *TOP* 5; 221–239.

[6] Bautista J, Companys R, Corominas A. (2001). Solving the generalized apportionment problem through the optimization of discrepancy functions. *European Journal of Operational Research* 131; 676-684.

[7] Brusco MJ. (2008). Scheduling advertising slots for television. *Journal of the Operational Research Society* 59; 1363-1372.

[8] Bollapragada S, Bussieck MR, Mallik S. (2004). Scheduling commercial videotapes in broadcast television. *Operations Research* 52(5); 679-689.

[9] Corominas A, Kubiak W, Moreno N. (2007). Response time variability. *Journal of Scheduling* 10; 97-110.

[10] Corominas A, Kubiak W, Pastor R. (2009). Mathematical programming modelling of the response time variability problem. European Journal of Operational Research, doi: 10.1016/j.ejor.2009.01.014.

[11] Dong L, Melhem R, Mosse D. (1998). Time slot allocation for real-time messages with negotiable distance constrains requirements. Fourth IEEE Real-Time Technology and Applications Symposium (RTAS'98), Denver, CO., 131-136.

[12] Han CC, Lin KJ, Hou CJ. (1996). Distance-constrained scheduling and its

applications in real-time systems. *IEEE Trans. on Computers* 45; 814-826.
[13] Herrmann JW. (2007). Generating Cyclic Fair Sequences using Aggregation and Stride Scheduling. *Technical Report*, University of Maryland; USA.
[14] Inman RR, Bulfin RL. (1991). Sequencing JIT mixed-model assembly lines. *Management Science* 37; 901–904.
[15] Kubiak W. (1993). Minimizing variation of production rates in just-in-time systems: A survey. *European Journal of Operational Research* 66; 259-271.
[16] Miltenburg J. (1989). Level schedule for mixed-model assembly lines in Just-in-time production system. *Management Science* 35; 192-207.
[17] Monden Y. (1983). *Toyota Production Systems*. Industrial Engineering and Management Press, Norcross, GA.
[18] Moreno N, Corominas A. (2006). Solving the minmax product rate variation problem (PRVP) as a bottleneck assignment problem. *Computers & Operations Research* 33; 928-939.
[19] Steiner G, Yeomans S. (1993). Level schedules for mixed-model, just-in-time processes. *Management Science* 39; 728-735.
[20] 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.

Appendix: The five heuristics presented in Corominas *et al*. (2007)

a) Bottleneck sequences (Bo)

Bo is a procedure that obtains an optimal sequence for the bottleneck problem. If we call x_{it} the number of units of product i allocated in slots $[1,\ldots,t]$ the bottleneck problem consists in *minimize* $\max_{i,t}\left|x_{it} - t\cdot d_i / D\right|$ (see Steiner and Yeomans 1993 and Moreno and Corominas 2006).

b) Random sequences (Ra)

Procedure *Ra* obtains a sequence by randomizing the bottleneck sequence as follows: for each position k_1 in $1...D$, take a random number k_2 in the range $1...D$ and swap $S[k_1]$ with $S[k_2]$, where $S[k]$ indicates the unit scheduled in position k of the sequence.

c) Two parametric methods for the apportionment problem: Webster's and Jefferson's sequences (We and Je)

The parametric method of apportionment (to apportion the seats of a House among the states, Balinski and Young 1982) is as follows. Let x_{ik} be the number of copies of product i in the sequence of length k, $k = 0,1,\ldots$; assume $x_{i0} = 0$, $i = 1,\ldots,n$; the product to be sequenced in position $k+1$ can be computed as follows: $i^* = \arg\max_i \left\{\frac{d_i}{x_{ik} + \delta}\right\}$, where $0 \le \delta \le 1$. In the event of a tie, products are selected in descending order of i.

Webster's and a Jefferson's sequences are obtained, respectively with $\delta = 0,5$ and $\delta = 1$.

d) Insertion sequences (In)

Assume that $d_1 \le \ldots \le d_n$ and consider $n-1$ two-product problems $P_2 = (d_2, d_1)$, $P_3 = \left(d_3, \sum_{j=1}^{2} d_j\right)$, …, $P_n = \left(d_n, \sum_{j=1}^{n-1} d_j\right)$. In each of the problems P_n, P_{n-1}, …, P_3, the first product is the original one, that is n, $n-1$, …, 3, respectively, and the second product is the same fictitious product for all the problems, denoted by *. Let sequences S_n, S_{n-1}, …, S_2 be the optimal solution to problems P_n, P_{n-1}, …, P_2, respectively. They can be obtained by the optimal algorithm proposed for the two-product case described in Corominas *et al*. (2007), which is introduced in the next paragraph. Notice that sequence S_i, $i = n,\ldots,3$, is made up of the product i and the fictitious product *. The sequence for the original problem is then built recursively by first replacing * in S_n with S_{n-1} to obtain S_n' (notice that S_n' is made up of products n, $n-1$ and *). Next, * are replaced by S_{n-2} in S_n' to obtain a sequence S_n'' made up of products n, $n-1$,

$n-2$ and *. Finally, sequence S_2 replaces all of the remaining instances of *.

A simple algorithm for obtaining an optimal solution for the two-product case ($n=2$) is presented in Corominas *et al*. (2007). Let $d_1 < d_2$ (the case $d_1 = d_2$ is trivial). Consider a sequence that begins with product 1 assigned to any position in the sequence and has each subsequent copy of product 1 at either a distance $\left\lceil D/d_1 \right\rceil$ or a distance $\left\lfloor D/d_1 \right\rfloor$ from the last one; the number of times the distances $\left\lceil D/d_1 \right\rceil$ and $\left\lfloor D/d_1 \right\rfloor$ need to be used in this sequence are $D \bmod d_1$ and $d_1 - D \bmod d_1$, respectively. The empty positions are then filled with the copies of product 2 (and distances equal to 2 and 1 must occur exactly $D \bmod d_2$ and $d_2 - D \bmod d_2$ times, respectively, in the sequence). The result is a sequence that minimizes the value of RTV. Obviously, there are several possibilities for selecting the position for units of both products, fulfilling the previous conditions. For the insertion sequence In, the positions for unit j of product 1 $\left(j = 1,...,d_1\right)$ are equal to their ideal positions, which are defined as $\left\lceil \frac{2 \cdot j - 1}{2 \cdot r_1} \right\rceil$ where $r_1 = d_1/D$.