Solving the Response Time Variability Problem by means of metaheuristics[†]

Alberto GARCÍA, Rafael PASTOR and Albert COROMINAS
Institut d'Organització i Control de Sistemes Industrials (IOC), Universitat Politècnica de Catalunya, Barcelona, Spain.

Abstract. The Response Time Variability Problem (RTVP) is a combinatorial NP-hard problem which has a wide range of real-life applications. It has recently appeared in the literature and has therefore not been widely discussed to date. The RTVP has been solved in other works by mixed integer linear programming (for small instances) and heuristics, but metaheuristic procedures have not previously been used. In this paper, a solution to the RTVP by means of multi-start, GRASP and PSO procedures is proposed. We report on our computational experiments and draw conclusions.

Keywords: response time variability, fair sequences, scheduling, metaheuristics.

Introduction

The Response Time Variability Problem (RTVP) consists in sequencing a list of products, events, clients and jobs in such a way that the variability in the time they spend waiting for their next turn to obtain the resources they need is minimized. This problem has recently been defined in the literature and to date very few papers have been published on the subject [1], [2], [3].

Corominas et al. [2] have proved that the RTVP is a combinatorial NP-hard problem and, with the exception of a few special cases, they have in fact found an optimum solution to the problem only for small instances. Therefore, solving the problem by means of heuristic and metaheuristic procedures is entirely justified. In this paper, a solution to the RTVP is put forward by applying the following three procedures: multi-start, GRASP (Greedy Randomized Adaptive Search Procedure) and PSO (Particle Swarm Optimization).

The multi-start method is based on generating initial random solutions and on improving each of them to find a local optimum, which is usually done through a local search procedure.

GRASP, designed by Feo and Resende [5] in 1989, can be considered to be a variant of the multi-start method in which the initial solutions are obtained using directed randomness. They are generated by means of a greedy strategy in which random steps are added and the choice of the elements to be included in the solution is adaptive.

[†] Sponsored by the Spanish Ministry of Education and Science's project DPI2004-03472; co-funded by the FEDER.

PSO is a metaheuristic procedure designed by Kennedy and Eberhart [6] in 1995. The original algorithm was designed for working with continuous functions of real variables and has obtained good results. Furthermore, it has recently been adapted for the purposes of working with combinatorial problems such as the travelling salesperson problem [7] or the flowshop problem [8]. In spite of these good results, there are not many PSO methods for solving combinatorial optimization problems.

The remainder of this paper is set out as follows: Section 1 presents a formal definition of the RTVP; Section 2 briefly describes the methods used and how they were adapted to solve the RTVP; Section 3 explains how the values for the metaheuristic parameters were established; the computational results are shown in Section 4; and finally, the conclusions are put forward in Section 5.

1. Response Time Variability Problem (RTVP)

The Response Time Variability Problem occurs whenever products, clients or jobs need to be sequenced so as to minimize variability in the time between the instants at which they receive the necessary resources.

The RTVP occurs in a wide range of real-life applications. For example, it is a common occurrence in the automobile industry in the sequencing of models [9] and in the Asynchronous Transfer Mode (ATM) when multimedia systems need to broadcast video or sound at a specific time [10].

These kinds of situations are often considered to be distance-constrained scheduling problems, in which the distance between any two given consecutive units of the same product is bounded. However, in the RTVP the aim is to minimize variability in the distances between any two consecutive units of the same product and to find a feasible solution that optimizes this objective.

The RTVP is formulated as follows. Let n be the number of products, d_i the number of units of product i and D the total number of units ($D = \sum_{i=1}^{n} d_i$). Let s be a solution of an instance in the RTVP that consists of a circular sequence of units ($s = s_1 s_2 \ldots s_D$), where s_j is the unit sequenced in position j of sequence s. For all products i in which $d_i \geq 2$, let t_k^i be the distance between the positions in which the units k+1 and k of product i are found (i.e., the number of positions between them). As the sequence is circular, position 1 comes immediately after position D; 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 $\bar{t}_i$ be the average distance between two consecutive units of product i ($\bar{t}_i = D / d_i$). For all products i in which $d_i = 1$, t_1^i is equal to $\bar{t}_i$. The objective is to minimize the $RTV = \sum_{i=1}^{n} \sum_{k=1}^{d_i} (t_k^i - \bar{t}_i)^2$.

For example, let $n = 3$, $d_A = 2$, $d_B = 2$ and $d_C = 4$; thus, $D = 8$, $\bar{t}_A = 4$, $\bar{t}_B = 4$ and $\bar{t}_C = 2$. A feasible solution is the sequence (C, A, C, B, C, B, A, C) where $RTV = \left[(5-4)^2 + (3-4)^2\right] + \left[(2-4)^2 + (6-4)^2\right] + \left[(2-2)^2 + (2-2)^2 + (3-2)^2 + (1-2)^2\right] = 2 + 8 + 2 = 12$

Corominas et al. [2] proved that the RTVP is NP-hard. The RTVP was optimally solved by means of mathematical programming, up to 40 units [3], and by means of heuristic procedures plus local optimization [2].

2. Multi-start, GRASP and PSO metaheuristic methods

2.1. Multi-start method

The multi-start method consists in generating random solutions, applying local optimization methods and preserving the best results.

The pseudocode of the adaptation of the multi-start method is

1. Let the value of the best solution found be $\overline{Z} = \infty$.
2. While (actual time < execution time), do:
3. Get a random initial solution X
4. Apply the local optimization to X and get X'
5. If value (X') < $\overline{Z}$, then $\overline{Z}$ = value (X')

Random solutions are generated as follows. For each position from 1 to *D* in the sequence, we randomly obtain which product will be sequenced with a probability equal to the number of units of that type of product that remain to be sequenced divided by the total number of units that remain to be sequenced.

The local optimization is applied as follows. A local search is performed iteratively in a neighbourhood that is generated by interchanging two consecutive units; the best solution in the neighbourhood is chosen; the optimization ends when no neighbouring solution remains that is better than the current solution.

2.2. Greedy Randomized Adaptive Search Procedure (GRASP) method

Like the multi-start method, GRASP consists in generating solutions, applying local optimizations and preserving the best results. However, the generation of solutions is performed by applying a heuristic with directed randomness, which is usually a random variation of a simple greedy heuristic. At each stage in the heuristic, the next product to be added to the solution is randomly selected from a list of candidates with a probability proportional to the value of an associated index.

The pseudocode of the GRASP adaptation is almost the same as that of the multi-start method: the only difference is the way in which the initial solutions are obtained, which is as follows. For each position from 1 to *D* in the sequence, the product to be sequenced is randomly selected from the candidate list with a probability proportional to the value of its Webster index. This index, defined in [2], is as follows: let $\delta = 1/2$ and let x_{ik} be the number of units of product *i* that have already been sequenced in the sequence of length *k*, *k* = 0, 1, …; the value of the Webster index of product *i* to be sequenced in position $k+1$ is $\frac{d_i}{x_{ik} + \delta}$.

The local optimization used is the same as the optimization used in the multi-start method.

The size of the candidate list was set to 5 candidates.

2.3. Particle Swarm Optimization (PSO) method

Kennedy and Eberhart designed the PSO metaheuristic by establishing an analogy to the social behaviour of flocks of birds when they search for food. Originally, this metaheuristic was designed to optimize continuous functions of real variables [6]. Due to its good performance, it has been adapted for the purposes of working with combinatorial problems [7], [8], [11].

In this kind of algorithm, the *particles*, which correspond to the birds, have a position (a feasible solution) and a velocity (the change in their position), and the set of particles form the *swarm*, which corresponds to the flock.

At each step in the PSO algorithm, the behaviour of a particle is the result of the combination of the following three factors: 1) to continue on the path that it is following, 2) to follow the best solution found and 3) to go to the best position found by the swarm. The formalization of this behaviour is expressed in the following two equations:

$$v_{t+1} = c_1 \cdot v_t \otimes c_2 \cdot (BP_t - X_t) \otimes c_3 \cdot (BSP_t - X_t) \quad (1)$$

$$X_{t+1} = X_t + v_{t+1} \quad (2)$$

where v_t is the velocity of the particle at time step t; X_t is the position of the particle at time step t; BP_t is the best position of the particle up to time step t; BSP_t is the best position of the swarm up to time step t; and c_1, c_2 and c_3 are the coefficients that weight the importance of the three types of decision.

The values of coefficients c_1, c_2 and c_3 are usually fixed in advance.

To apply the PSO algorithm to the RTVP, the elements and the operations of the equations (1) and (2) have to be defined.

2.3.1. Position of the particle

As mentioned above, a position represents a feasible solution. The position is represented by a D-length array that contains the sequence of D units.

2.3.2. Velocity of the particle

The expression ($X_2 - X_1$) represents the difference between two positions and it is the velocity needed to go from position X_1 to X_2. This velocity is an ordered list of transformations (called *movements*) that must be applied to the particle so that it changes from its current position to the other one. Two types of movements, each of which had two variations, were considered.

The first type of movement, called *M1*, is a pair of values (α / j). For each position s in the sequence X_1, a check is conducted to determine whether the unit in this position s is equal to the unit in position s of sequence X_2. If they are different, α is the unit in position s of X_2 and j is position s. Thus, this movement denotes that the unit in position j must be exchanged for the first unit that is equal to α and that is to the right of position s. This concept is used to solve the CONWIP problem [11].

The second type of movement, called *M2*, is a pair of positions (i, j). These values indicate that the units that are sequenced in positions *i* and *j* have been exchanged.

Two examples of the movements that are needed to move to position X_2 (A-B-C-A-B-C-A-B-C) from position X_1 (A-A-A-B-B-B-C-C-C) are shown below.

M1: movements (B/2), (C/3) and (C/6) are needed.
A-A-A-B-B-B-C-C-C → (B/2) → A-**B**-A-**A**-B-B-C-C-C → (C/3) →
A-B-**C**-A-B-**A**-C-C → (C/6) → A-B-C-A-B-**C**-A-**B**-C
M2: movements (2,4), (3,7) and (6,8) are needed.
A-A-A-B-B-B-C-C-C → (2,4) → A-**B**-A-**A**-B-B-C-C-C → (3,7) →
A-B-**C**-A-B-**A**-C-C → (6,8) → A-B-C-A-B-**C**-A-**B**-C

There would seem to be no difference between *M1* and *M2*, but when two velocities are added (see Section 2.3.4) then lists of movements that refute this may appear.

The two variations for each movement are: 1) if only the type of product is used to compare two units (this variation is called *T* and it is used in examples above), and 2) if the unit number is used to compare two units and therefore a unit is only equal to itself (this variation is called *F*). For example, in the case of variation *F*, position A1-A2-A3-B1-B2-B3-C1-C2-C3 (in which the number next to each letter is a unit identifier for each product) is different to position A2-A1-A3-B1-B3-B2-C1-C2-C3, but in variation *T* the two positions are equal (they appear as A-A-A-B-B-B-C-C-C).

The difference between two positions using variation *F* will always be greater than or equal to the difference when variation *T* is applied.

2.3.3. External multiplication of a coefficient by a velocity

The coefficients c_1, c_2 and c_3 yield values of between 0 and 1. When a coefficient is multiplied by a velocity, it indicates the probability of each movement that is to be applied. For example, if we multiply velocity [(B/2), (C/3), (C/6)] by coefficient 0.6, three random numbers between 0 and 1 are generated for comparison with coefficient 0.6; if the values are 0.3, 0.8 and 0.4, then movements (B/2) and (C/6) are applied, whereas movement (C/3) is not. The resulting velocity of the multiplication is therefore [(B/2), (C/6)].

2.3.4. Sum of velocities

The sum of two velocities is simply the concatenation of their own list of movements.

2.3.5. Sum of a velocity plus a position

The sum of a velocity plus a position gives the same result as applying each movement of the velocity to the position.

2.3.6. Pseudocode of the algorithm

1. Initiate the particles with random positions and empty velocities.
2. While (actual time < execution time), do:
3. Update the best swarm position.
4. For each particle:
5. update its best position and apply the two PSO equations.

The random positions are generated in the same way as the random solutions in the multi-start method.

3. Fine-tuning the PSO parameters

Adapting metaheuristics to a specific problem does not end with the definition of the space of solutions or the local search; moreover, it is required to set the parameters. The value of the parameters is vital because the results of the metaheuristic for each problem are very sensitive to them. To fine-tune is very expensive and it is usually done by intuitively testing several values.

For the purposes of this paper, we fine-tuned the parameters using a recent technique called CALIBRA [12]. CALIBRA is an automatic configuration procedure based on statistical analysis techniques (Taguchi's fractional factorial experimental designs) coupled with a local search procedure. A set of 60 representative instances was used to fine-tune the algorithms and a set of 740 units was used to test them. The four parameters to be fine-tuned were the number of particles in the swarm and coefficients c_1, c_2 and c_3. The range of the values used to fine-tune the algorithms was [5,30] for the number of particles and [0,1] for the coefficients. CALIBRA needed 35 hours to fine-tune each algorithm.

4. Computational results

As described in Section 2.3.2, depending on the type of movement (*M1* or *M2*) and the variation (*T* or *F*), we have four PSO algorithms (called *M1-F*, *M1-T*, *M2-F* and *M2-T*), as well as the multi-start algorithm and the GRASP algorithm.

The algorithms ran 740 instances, which were grouped into four classes (185 instances in each class) depending on their size. The instances in the first class (called *CAT1*) were generated using a random value of *D* (number of units) between 25 and 50, and a random value of *n* (number of products) between 3 and 15; for the second class (called *CAT2*), *D* was between 50 and 100, and *n* between 3 and 30; for the third class (called *CAT3*), *D* was between 100 and 200 and *n* between 3 and 65; and for the fourth class (called *CAT4*), *D* was between 200 and 500 and *n* between 3 and 150.

The algorithms were coded in Java and the computational experiments were carried out using a 3.4 GHz Pentium IV with 512 Mb of RAM.

Firstly, the six algorithms were run for 50 seconds for each instance. Table 1 shows the averages of the RTV values to be minimized for each class of instances.

Table 1. Averages of the RTV values to be minimized

	PSO				Multi-start	GRASP
	M1F	**M1T**	**M2F**	**M2T**		
CAT1	68.79	66.83	83.14	80.93	11.33	13.90
CAT2	445.55	509.89	604.27	517.05	48.10	91.64
CAT3	3050.38	4335.87	4488.44	3888.79	320.63	541.52
CAT4	28955.82	48917.80	37937.76	30029.34	79823.89	57041.74

In Table 1 it can be seen that the best results for the three first classes are given by the multi-start method, followed by the GRASP method, whereas the PSO algorithm yields the worst results. However, in the case of class *CAT4*, in which the instances are largest, the order is the reverse: the four PSO algorithms yield better results than the GRASP method, and the multi-start method gives the worst results. The reason for this is that the multi-start method does not have time to locally optimize a single solution for 87.57% of the instances in the *CAT4* class; this happens in the GRASP method for 84.32% of the instances.

The second computational experiment consisted in locally optimizing the solutions that were obtained with the PSO algorithms in the first computational experiment. The optimization used was the same as the multi-start optimization; it stops after 50 seconds if the optimization has not been completed. Table 2 shows the averages of the RTV values obtained for each class of instances.

Table 2. Averages of the RTV values of the PSO local optimized solutions

	M1F	**M1T**	**M2F**	**M2T**
CAT1	21.61	24.43	23.61	25.65
CAT2	67.42	89.75	77.56	95.14
CAT3	229.32	406.63	302.06	427.09
CAT4	15842.12	29604.35	20560.1	15537.62

The results obtained using *M1F* for the instances in class *CAT3* after local optimization are better than the results obtained using the multi-start method. Moreover, the optimization times for the first two classes are negligible and the average time for the third class is between 4.26 and 5.84 seconds (using *M1F* and *M1T*, respectively). The instances in the first three classes were all locally optimized. However, there was not enough time to optimize all the instances in class *CAT4*: only 60 instances (32.43%) were locally optimized based on the solutions that were obtained using *M1F*.

Finally, the six procedures were re-run for 200 seconds using the instances in class *CAT4*, which are the most difficult to solve. In the case of PSO algorithms, 100 seconds were spent on obtaining a solution and a further 100 seconds, at the most, were spent on locally optimizing the previous solution. Table 3 shows the average of the RTV values obtained for class *CAT4* (the values in parenthesis were obtained using the PSO algorithms before local optimization was applied).

Table 3. Average of the RTV values of the *CAT4* instances

M1F	**M1T**	**M2F**	**M2T**	**multi-start**	**GRASP**
(*24022.52*)	(44697.30)	(36445.60)	(29838.01)		
8782.07	21432.13	14892.35	11984.25	39719.71	30020.35

The results show that all the PSO algorithms give better results than the multi-start and GRASP algorithms. In this last experiment, 97 instances (52.43%) were locally optimized after applying the *M1F* algorithm.

5. Conclusions and future lines of research

In this paper we have presented our solution to the RTVP (a problem that has not been widely researched to date), to which six algorithms were applied: one multi-start, one GRASP and four PSO.

The results show that the best procedure is the multi-start for small instances (between 25 and 100 units and between 3 and 30 products). However, for bigger instances (between 100 and 500 units and between 3 and 150 products) the search should be more specific as the four PSO algorithms are much better than the multi-start and GRASP methods and the latter are better than the multi-start methods. Moreover, as was to be expected, there is a significant improvement in the solutions that were obtained using the PSO algorithm to which local optimization had been applied.

Future research will consist in adapting new metaheuristic procedures, such as for example simulated annealing and tabu search.

References

[1] D. León, A. Corominas, A. Lusa, *Resolución del problema PRV min-var*, Working paper IOC-DT-I-2003-03, UPC, Barcelona, Spain, 2003.

[2] A. Corominas, W. Kubiak, N. Moreno, *Response time variability*, Working paper IOC-DT-P-2004-08. UPC, Barcelona, Spain, 2004.

[3] A. Corominas, W. Kubiak, R. Pastor, *Solving the Response Time Variability Problem (RTVP) by means of mathematical programming*, Working paper IOC-DT, UPC, Barcelona, Spain, 2006.

[4] R Martí, *Multi-start methods*, Handbook of Metaheuristics, Glover and Kochenberger (eds.), Kluwer Academic Publishers, pp. 355-368, 2003.

[5] T.A. Feo, M.G.C. Resende, *A probabilistic heuristic for a computationally difficult set covering problem*, Operations Research Letters, vol. 8, pp. 67-81, 1989.

[6] J. Kennedy, R.C. Eberhart, *Particle swarm optimization*, IEEE International Conference on Neural Networks, Australia, 1995.

[7] B. Secrest, *Travelling salesman problem for surveillance mission using PSO*, PhD thesis, Air Force Institute of Technology, Ohio, USA, 2001.

[8] C.J. Liao, C.T. Tseng, P. Luarn, *A discrete version of PSO for flowshop scheduling problems*, Computers & Operations Research, in press, corrected proof available online, 5 December 2005.

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

[10] L. Dong, R. Melhem, D. Mossel, *Time slot allocation for real-time messages with negotiable distance constraint requirements*, Real-time Technology and Application Symposium, RTAS, Denver, 1998

[11] C. Andrés, R. Pastor, J.M. Framiñán, *Optimización mediante cúmulos de partículas del problema de secuenciación CONWIP*, Eighteenth Conference on Statistics and Operations Research SEIO'04, Cádiz, Spain, 2004.

[12] B. Adenso-Díaz, M. Laguna, *Fine-tuning of algorithms using fractional experimental designs and local search*, Operations Research, vol. 54, no. 1, pp. 99-114, 2006.