47 USING REJECTION METHODS IN A DSS FOR PRODUCTION STRATEGIES

Massimiliano Caramia[1], Pasquale Carotenuto[2]
Stefano Giordani[3,4], Antonio Iovanella[4]
[1]*I.A.C. – C.N.R., Viale del Policlinico, 137, I-00161 Rome, Italy*
caramia@iac.rm.cnr.it
[2]*I.T.I.A. – C.N.R., Via del Politecnico 1, I-00133 Rome, Italy*
carotenuto@disp.uniroma2.it
[3]*Centro Interdip. "Vito Volterra" – Univ. of Rome "Tor Vergata", Via di Tor Vergata, I-00133 Rome, Italy, giordani@disp.uniroma2.it*
[4]*Dip. Informatica Sistemi e Produzione – Univ. of Rome "Tor Vergata", Via del Politecnico 1, I-00133 Rome, Italy, iovanella@disp.uniroma2.it*

In this paper we face the problem arising in an enterprise that must decide whether and when scheduling production orders in order to maximize the production efficiency. In particular we developed an on-line scheduling algorithm able to manage such decisions. Computational results are provided to show the performance of the algorithm.

1. INTRODUCTION

Today an increasing number of manufacturing enterprises must collaborate and communicate with a large number of suppliers spread in large areas to design and produce their products. The Information and Communication Technology gives an effectiveness support to this activity, but a well suited decision support system is also necessary to manage the supply chain with the final goal to increase the enterprise production efficiency [6].

The problem we study is the following. Suppose that an enterprise receives production orders continuously from the customers in an on-line fashion with the over time paradigm [5], that means that the enterprise does not know anything in advance about the requested orders until they arrive. Moreover, suppose that all the orders must be dispatched before a deadline and have a time length equal to their production time.

To produce what is ordered, the enterprise needs production resources (say machines) to be allocated to orders for certain times. In particular, we suppose that the machines are linearly ordered and each order requires a certain number of consecutive machines for a certain production time. Moreover, once an order is scheduled it can not be preempted.

V. Mařík et al. (eds.), *Knowledge and Technology Integration in Production and Services*

However, it is not sure that all the incoming orders can be scheduled. In fact, we are in a twofold scenario: either the order can be scheduled within the deadline, or it must be rejected as there is not enough space in the schedule.

In this paper we developed an on-line scheduling algorithm able to manage decisions maximizing the production efficiency. Computational results are provided to show the performance of the algorithm.

The remainder of the paper is organized as follows. Section 2 contains the model; in Section 3 we sketch the on-line algorithm and in Section 4 computational results are provided. Section 5 concludes the paper with some final remarks.

2. THE MODEL

We are given a set of H parallel identical machines arranged in a linear order, a time limit W and a set of orders (requests) J, with $n = |J|$. Each request $j \in J$ requires h_j consecutive machines for a certain time w_j , and can not be preempted. This is a special case of multiprocessor task scheduling problem with parallel identical processors (see [1-4]).

The requests are presented one by one, and the requirement of each request becomes available only when the request is presented. Each time a new request j is presented we have to decide to reject it or to accept it. In the latter case, we have to assign a subset of h_j consecutive machines for the entire request duration w_j to j; this is the same as assigning to j a free rectangular area X_j (of height h_j and width w_j) contained in the area A of height H and width W. Clearly, the areas assigned to accepted requests have to be mutually disjoint.

Our objective is the maximization of the production efficiency index ρ, measured as the ratio between the resources assigned to accepted orders and the maximum assignable production capacity. In terms of the proposed model ρ is the ratio between the size of the total assigned area and the minimum between the size of A and the total size of the area required by the set of requests. Of course, ρ is in between 0 and 1.

3. THE ON-LINE ALGORITHM

Without loss of generality, we consider the requests indexed according to the order in which they are presented. Given a list $L=\{1, 2,\ldots, n\}$ of such requests, an algorithm A considering L is said to be on-line if [2, 5]:

1. A considers requests in the order given by the list L;
2. A considers each request i without knowledge of any request j, with $j > i$;
3. A never reconsiders a request already considered.

The algorithm operates in $n = |J|$ iterations, and during iteration (j) the request j is considered (accepted or rejected) and a new (eventually empty) free sub-area A_{j+1} of A is defined.

Let us consider iteration (j). Let $A^{(j-1)}$ be the non-assigned (free) area of A, and $\{A_1^{(j-1)}, \ldots, A_j^{(j-1)}\}$ a partition of $A^{(j-1)}$, that is $A_p^{(j-1)} \cap A_q^{(j-1)} = \varnothing$, for $p \neq q \in \{1, \ldots, j\}$ and $\cup_{s=1}^{j} A_s^{(j-1)} = A^{(j-1)}$, where each $A_s^{(j-1)}$ is a free rectangular area of A. See for example Figure 1. Clearly, at the beginning, we have $A_1^{(0)} = A^{(0)} = A$.

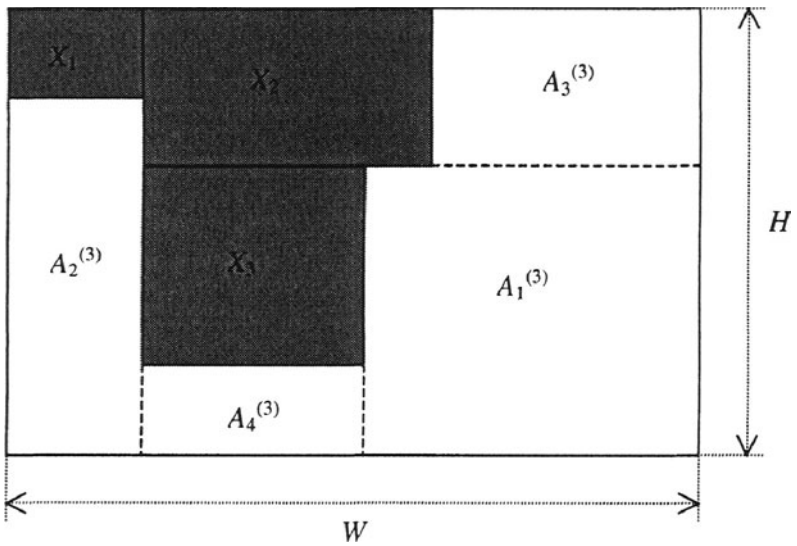

Figure 1 – Assigned and Free Rectangular Areas at the Beginning of Iteration (4)

The request j is accepted if there is a free area $A_k^{(j-1)} \in \{A_1^{(j-1)}, \ldots, A_j^{(j-1)}\}$ that may satisfy the requirement of j, that is both $W_k^{(j-1)} \geq w_j$ and $H_k^{(j-1)} \geq h_j$, with $W_k^{(j-1)}$ being the width and $H_k^{(j-1)}$ the height of $A_k^{(j-1)}$, respectively, otherwise j is rejected. In particular, if j is accepted let $A_k^{(j-1)}$ be the smaller (in terms of size) free area satisfying the requirement of j.

When j is accepted (see Figure 2), a sub-area X_j (of height h_j and width w_j) in the north-west corner of $A_k^{(j-1)}$ is assigned to j, leaving two free rectangular sub-areas, namely $A_k^{(j)}$ and $A_{j+1}^{(j)}$, of $A_k^{(j-1)}$.

In particular, let $A_k^{(j)} = A_k^{(j-1)} \setminus M_k^{(j)}$ and $A_{j+1}^{(j)} = M_k^{(j)} \setminus X_j$, with $M_k^{(j)}$ of size $m_k^{(j)} = \min\{w_j H_k^{(j-1)}, h_j W_k^{(j-1)}\}$ be the rectangular sub-area of $A_k^{(j-1)}$ located on the west side of $A_k^{(j-1)}$ if $m_k^{(j)} = w_j H_k^{(j-1)}$ otherwise in the north side; with this choice we have $A_k^{(j)} \geq A_{j+1}^{(j)}$, and the k-th free rectangular area A_k area is reduced by a minimal amount. If j is rejected, we consider $A_k^{(j)} = A_k^{(j-1)}$ and $A_{j+1}^{(j)} = \varnothing$.

The value of the solution found by the on-line algorithm is

$$\rho = \frac{W H - \sum_{s=1}^{n} (W_s^{(n)} H_s^{(n)})}{\min\{WH, \sum_{j=1}^{n} w_j h_j\}}$$

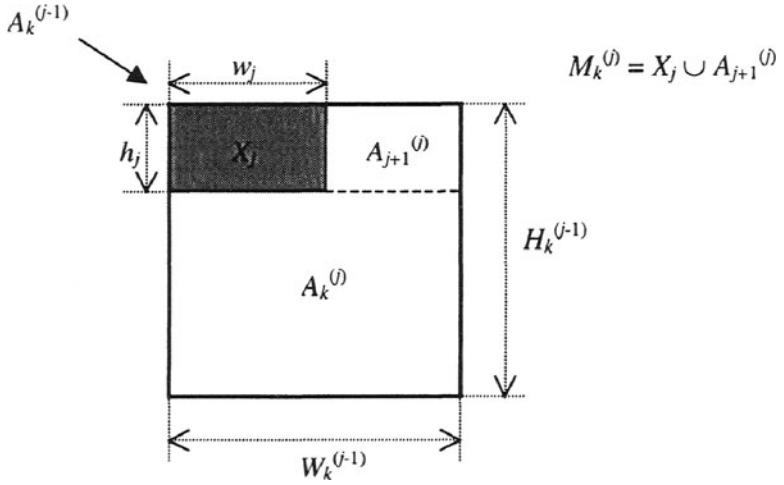

Figure 2 – Accepting Request *j*.

4. COMPUTATIONAL RESULTS

4.1 General

The algorithm was tested on randomly generated instances with n = 10, 20, 50 and 100 order requests getting a total number of 4 classes of test cases. For each class we have considered different test cases according to different choices of two parameters, say w_{max} and h_{max}, being the maximum time that can be associated with a request, and the maximum number of parallel contiguous machines for a request, respectively, and different areas A; in particular, we have considered h_{max} = 5, 10, 15 and w_{max} = 5, 10, 15, and 3 different areas A with the following value for the height H and the width W: (H, W) = {(15, 20), (20, 30), (25, 50)}, for a total number of 27 test cases for each class. For each one of the 108 different test cases we randomly generated ten instances where the request time w_j and number of machines h_j required by a request j are uniformly distributed in the intervals [1, w_{max}] and [1, h_{max}], respectively.

The algorithm and the instance generator have been implemented in the C language, compiled with the GNU CC 2.8.0 with the -o3 option and tested on a PC Pentium 600 MHz with Linux OS.

4.2 The Data Comparisons

In Figures 3-5 we summarize average results of the efficiency index ρ. It can be noted that if we have a small number of requests but with the longest duration and highest resource requirements the algorithm performs the lower values. As soon as the requests are small in the sense of the duration and/or number of resources required the efficiency reach a value almost one. This is due to the chance the algorithm has to allocate resources to "small" requests.

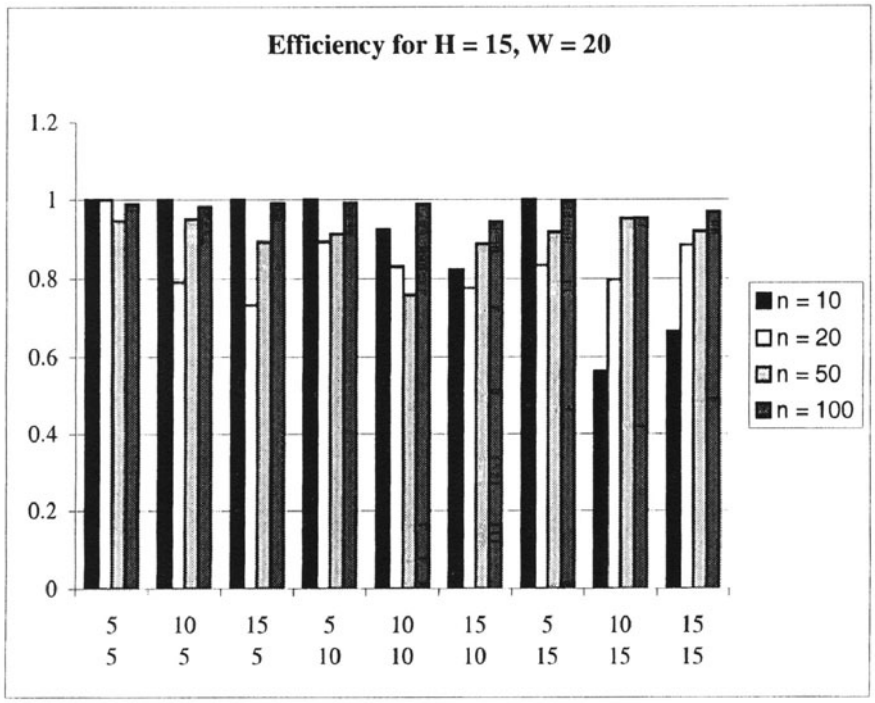

Figure 3 – Efficiency for $H = 15$ and $W = 20$.

For the sake of completeness, we report also in Tables 1 and 2 the complete results for the cases ($W = 15$, $H = 20$, $n = 10$) and ($W = 15$, $H = 20$, $n = 20$), respectively, which are the cases where we obtained the worst results. In those tables, the first column is the maximum number of resources required by a request, the second column is the maximum time, the third the number of the average rejected requests, and the last three columns are the minimum value of ρ, say ρ_{min}, the average values of ρ, say ρ_{ave}, and the maximum value of ρ, say ρ_{max}, respectively, taken over ten different instances.

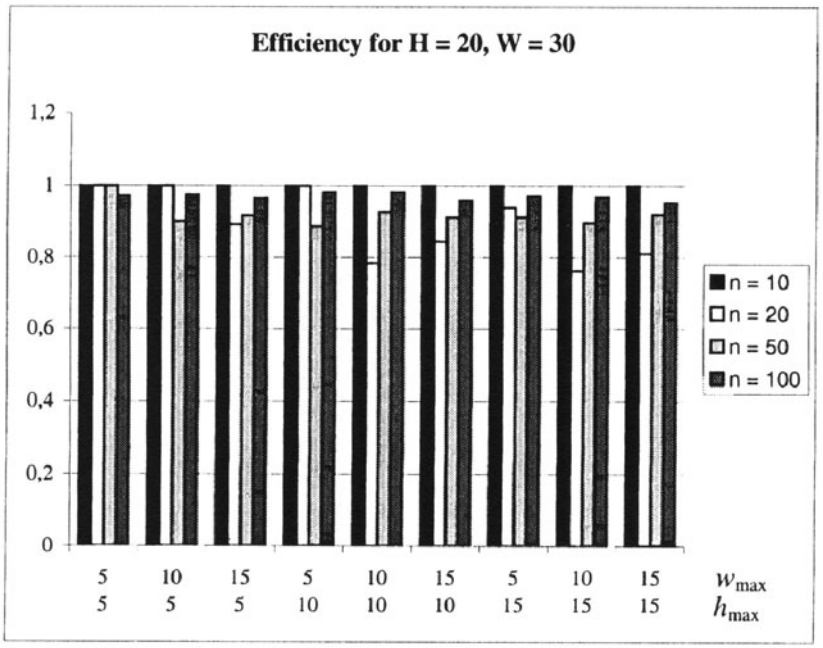

Figure 4 – Efficiency for $H = 20$ and $W = 30$.

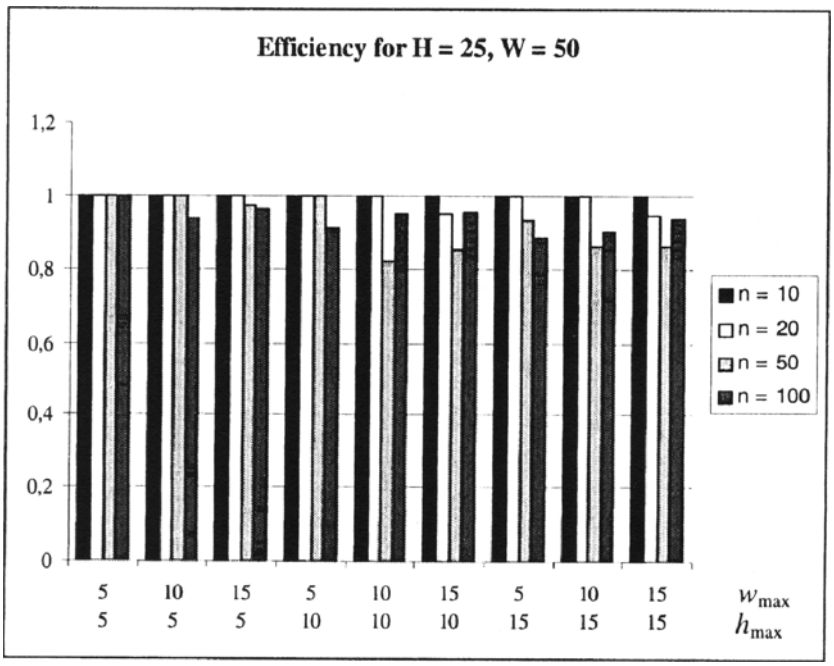

Figure 5 – Efficiency for $H = 20$ and $W = 30$

Table 1 – Results for $W = 15$, $H = 20$ and $n = 10$

h_{max}	w_{max}	*Rejected*	ρ_{min}	ρ_{ave}	ρ_{max}
5	5	0.0	1.000	1.000	1.000
5	10	0.0	1.000	1.000	1.000
5	15	1.0	0.911	1.000	1.000
10	5	0.0	1.000	1.000	1.000
10	10	1.5	0.623	0.923	1.000
10	15	3.0	0.731	0.822	0.913
15	5	0.5	0.935	1.000	1.000
15	10	3.4	0.547	0.561	0.570
15	15	6.4	0.637	0.665	0.780

Table 2 – Results for $W = 15$, $H = 20$ and $n = 20$

h_{max}	w_{max}	*Rejected*	ρ_{min}	ρ_{ave}	ρ_{max}
5	5	0.0	1.000	1.000	1.000
5	10	2.7	0.680	0.790	1.000
5	15	5.8	0.687	0.731	0.767
10	5	2.3	0.833	0.891	0.937
10	10	6.0	0.783	0.829	0.937
10	15	9.2	0.763	0.776	0.807
15	5	7.2	0.797	0.832	0.840
15	10	12.6	0.643	0.797	0.940
15	15	14.5	0.863	0.884	0.947

The worst results are obtained for the cases ($h_{max} = 15$, $w_{max} = 10$, $n = 10$) and ($h_{max} = 15$, $w_{max} = 15$, $n = 10$), both in terms of number of rejected requests and efficiency. This is due to the fact that in these cases we have to process requests requiring almost the whole production capacity. A similar situation occurs for the cases ($h_{max} = 10$, $w_{max} = 15$, $n = 20$) and ($h_{max} = 15$, $w_{max} = 15$, $n = 20$), even though the efficiency is higher with respect to the previous two cases with $n = 10$; this can be justified because with $n = 20$ we have more chances to efficiently use the whole production capacity. As one can expect the worst cases to deal with are those ones with h_{max} and/or w_{max} values close to H and W, respectively; indeed, in this cases we could have a very small chance to fit a request whose size is very close to the size of the whole (available) area A, especially if small requests have been accepted before implying a reduction on the size of the available areas. Nevertheless, the algorithm seems to perform well providing almost always solutions with efficiency value ρ greater than 0.8.

5. CONCLUSIONS

In this paper, we present a very simple on-line algorithm for scheduling production requests able to accept or reject incoming requests to maximize production

efficiency. We performed a wide computational analysis showing the behavior of the proposed algorithm. Performance results show that the on-line algorithm provides good solutions in almost all the tested cases.

6. REFERENCES

1. Brucker P. Scheduling Algorithms, Springer-Verlag, Berlin, 1995.
2. Caramia M., Dell'Olmo P., Iovanella A. On Line Algorithms for Multiprocessor Task Scheduling. Foundations of Computing and Decision Science, 2001, 26 (3), 197-214.
3. Dell'Olmo P., Giordani S., Speranza MG. An Approximation Result for a Duo-processor Task Scheduling Problem. Information Processing Letters, 1997, 61, 195-200.
4. Drozdowsky M. Scheduling Multiprocessor Task. An Overview. European Journal of Operations Research 1996, 94: 215-230.
5. Fiat A., Woeginger GJ. Online Algorithms. LNCS State of the Art Survey, Springer-Verlag, Berlin,1998.
6. Shapiro JF. Modeling the Supply Chain. Duxbury Press, 2000.