RUTCOR
RESEARCH
REPORT

A Method to Schedule Both Transportation and Production at the Same Time in a Special FMS

Navid Hashemian[a] Béla Vizvári[b]

RRR 12-08, September 2008

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]Dept. of Industrial Engineering, Eastern Mediterranean University, Mersin 10, Turkey

[b]RUTCOR, Dept. of Industrial Engineering, Eastern Mediterranean University, Mersin 10, Turkey and Department of Operations Research, Eötvös Loránd University of Budapest, H-1088 Budapest, Pázmány Péter sétány 1/C, H-1117 Hungary

Rutcor Research Report
RRR 12-08, September 2008

A Method to Schedule Both Transportation and Production at the Same Time in a Special FMS

Navid Hashemian Béla Vizvári

Abstract. A new method is developed to schedule both the production and transportation of a special FMS. The scheduling of the same FMS was the topic of earlier papers. But they scheduled either the production regardless the transportation system, or scheduled the transportation system only if the production has been already scheduled. Here the schedule of the two systems (production and transportation) is determined within a single algorithm at the same time.

1 Introduction

This paper is devoted to the schedule of a special FMS. This FMS was described first in [Blazewicz et al. 1991] and was characterized as producing parts of helicopters. The special properties of this FMS are these:

1. All machines are identical and are able to perform all operations.
2. The material handling system is a unidirectional AGV system, which transports the parts from the palette station/storing system to the machines.
3. There is another transportation system to supply the machines with tools.
4. The tool magazines of the machines have a large capacities.
5. The tools are stored in a central tool storing system having a very large capacity.
6. Each machine can process only one task at the same time.
7. Each part has only one, maybe complex, operation.
8. Preemption is not allowed.
9. All machines and parts are available at time zero.
10. The transportation may start earlier than zero. The earliest start of a transportation is given.
11. The AGV system has k vehicles.
12. Each AGV can carry a single part only.
13. The minimal time difference between the start of two vehicles is a.
14. The minimal time between two consecutive starts of the same vehicle, i.e. the time needed by an AGV to make a complete circuit, is A.
15. The loaing and unloading times are included in time A
16. The buffers of the machines have an infinite capacity.
17. The tasks have no due-dates.
18. Due-dates for transportations are determined according to the schedule of parts.

It follows from the conditions that the tool system is disregarded as it cannot restrict the systems activity in any sense.

[Blazewicz et al. 1991] discusses a scheduling method of the transportation system if the schedule of the production is fixed. They also describe a dynamic programming algorithm for the simultaneous scheduling of tasks and transportation based on the work of [Rothkopf 1966] but this method has no practical importance because of the extremely high memory and computational requirements. Even [Blazewicz et al. 1991] does not give any computational result. [Roh-Kim 1997] deals with a similar system having due-dates and tool constraints. It applies several heuristic methods, including list scheduling, to the scheduling of the production. On the other hand [Roh-Kim 1997] does not discusses the scheduling of the transportation system.

The main result of this paper is a new algorithm for the simultaneous schedule of the transportation system and the production. It is a modified version of the enumerative algorithm of [Demir-Vizvári 1995], which made possible to solve large scale instances of $P \mid\mid C_{\max}$.

An exact integer programming formulation of the problem is also provided.

2 Notations

Although all notations will be introduced in the discussion of the appropriate topic, here all of the are summarized.

m	the number of machines
n	the number of different processing times
k	the number of AGVs
i	the index of machines
j, β, γ	indices of tasks (jobs) and/or processing times
p_j	the j^{th} processing time
u_j	the number of taks having processing time p_j
N	the number of tasks (jobs) $= \sum_{j=1}^{n} u_j$
v_j	the number of taks, which have processing time p_j and still have not been loaded to machines
a	the safety time between the starts of two consecutive AGVs
A	the time needed to make a complete circuit by an AGV
e	the starting time of the transportations
e_l	the l^{th} possible earliest transportation starting time
τ_i	the transportation time from the palette station to the i^{th} machine
UB	upper bound of the load of the machines
LB	lower bound of the load of the machines
z_j	the number of jobs with processing time p_j in the load of the current machine
M	an appropriate large pozitive number

3 Preliminary Discussion of Tools of the Algorithm

3.1 Earliest Transportation Times and Transportation Due-Dates

In what follows it is assumed that the time necessary to make a complete cycle by an AGV, i.e. A, is at least as great as the total waiting time of the k vehicles after each other, i.e. ka, that is the inequality

$$A \geq ka \tag{1}$$

holds, otherwise the number of vehicles are to high and it is not possible to use the total capacity of the material handling system. Let $e < 0$ be the time of the beginning of transportation. Then the possible earliest starting times of the first k transportations are

$$e_1 = e,\ e_2 = e + a,\ \ldots\ e_k = e + (k-1)a.$$

Then it follows from (1) that the $k+1^{st}$ starting time is $e+A$. In general the l^{th} earliest possible starting time is

$$e_l \;=\; \left\lfloor \frac{l-1}{k} \right\rfloor A \;+\; \left(l - \left\lfloor \frac{l-1}{k} \right\rfloor k - 1 \right) a \;+\; e.$$

The transportation time from the palette station to the different machine is different. Assume that this time is τ_i to machine i. If a job j served by the l^{th} transportation, and the process of job j starts at time s_j, then its transportation is feasible if and only if

$$e_l \;+\; \tau_i \;\leq\; s_j. \tag{2}$$

In other words the transportation of job j must start not later than

$$s_j \;-\; \tau_i, \tag{3}$$

As it was determined in [Blazewicz et al. 1991]. A certain production schedule is served by the transportations if and only if (2) is true for all jobs. These constraints will be used in the algorithm as a feasibility test.

3.2 Lower and Upper Bounds of Load of Machines

The objective function value of any feasible solution, i.e. the current $C_{\max}$, is the load of at least one machine. Assume that the makespan of the best known feasible solution is C^{best}. If the optimal solution has not been explored yet then its value is not greater than $C^{best}-1$ in the case of integer processing and transportation times. Thus

$$UB \;=\; C^{best} - 1 \tag{4}$$

is an upper bound for the load of all machines in the feasible solutions still to be investigated. Hence

$$LB \;=\; \max \left\{ 0, \sum_{j=1}^{n} p_j - (m-1)UB \right\} \tag{5}$$

is a lower bound for all of the loads in the same feasible solutions. After finding a new and better feasible solution both bounds become tighter.

3.3 Lexicographic Order of the Loads of a Machine

The main algorithm is an implicit enumeration. It is very important to ensure that enumeration does not skip any potential feasible solution. The loads of the machines are determined after each other. Thus the potential loads of machine i with $1 < i \leq m$ are depending on the loads of the previous machines.

Any load of a machine must satisfy two constraints besides transportation feasibility:

(i) it cannot contain more tasks than what are still available,

(ii) the total load, i.e. the sum of the processing times, must be between the lower and upper bounds.

These conditions can be described by the following system of inequalities, where z_j is the number

of jobs with processing time p_j in the load of the current machine:

$$0 \ \leq \ z_j \ \leq \ v_j \quad j = 1, \ldots, n \tag{6}$$

$$LB \ \leq \ \sum_{j=1}^{n} p_j z_j \ \leq \ UB \tag{7}$$

$$z_j \ \text{is integer.} \ j = 1, \ldots, n, \tag{8}$$

where v_j is the number of tasks, which have processing time p_j and are still not loaded to any machine. E.g. in the case of the first machine $v_j = u_j$ for all j but the same statement is not true for machine 2 as some v_j's are strictly less than the appropriate u_j according to the load of the first machine.

There are many feasible solution of the system (6)-(8). For an implicit enumeration procedure they must be ordered somehow and enumerated in this order. One easy way to perform this task is to order them lexicographically. The lexicographically greatest solution is determined by the greedy method:

$$z_1 \ = \ \min \left\{ \left\lfloor \frac{UB}{p_1} \right\rfloor, \, v_1 \right\} \tag{9}$$

$$z_j \ = \ \min \left\{ \left\lfloor \frac{UB - \sum_{\beta=1}^{j-1} p_\beta z_\beta}{p_j} \right\rfloor, \, v_j \right\} \quad j = 2, \ldots, n. \tag{10}$$

In the lexicographic order next load is needed in the enumeration, whenever a branch is explored. It can be determined by the greedy method on the following way, where the current load is z, and the lexicographic next is $\bar{z}$

$$\gamma \ = \ \max \{ \, j \mid z_j \ > \ 0 \, \} \tag{11}$$

$$\bar{z}_j \ = \ z_j \quad j \ = \ 1, \ldots, \gamma \tag{12}$$

$$\bar{z}_\gamma \ = \ z_\gamma \ - \ 1 \tag{13}$$

$$\bar{z}_j \ = \ \min \left\{ \left\lfloor \frac{UB - \sum_{\beta=1}^{j-1} p_\beta z_\beta}{p_j} \right\rfloor, \, v_j \right\} \quad j = \gamma + 1, \ldots, n. \tag{14}$$

There is no further feasible load of the machine if

$$\forall \, j : \ z_j > 0 \ \text{implies that} \sum_{\beta=1}^{j-1} p_\beta z_\beta \ + \ p_j (z_j - 1) \ + \ \sum_{\beta=j+1}^{n} p_\beta v_\beta \ < \ LB. \tag{15}$$

4 The Main Algorithm

The main algorithm is an enumeration, which consists of two types of steps: construction and backtrack.

In the construction step the algorithm determines the load of the machines one by one. This is done by either the (9)-(10) greedy algorithm if the machine previously had no load, or by the formulae (11)-(14) if it had a load. Whenever a possible load is determined for machine i then two conditions are checked. The total processing time represented by the load must be at least as high as the lower bound, and all necessary transportations upto the load of the load of machine i must be feasible. If at least one of these constraints are violated then again the next possible load is determined by the formulae (11)-(14). If the construction is successful, i.e. all machines have load and all conditions are satisfied then the lower and upper bounds of are updated. The

enumeration is continued at that machine, which has maximal load and has minimal index among such machines. It lexicographically maximal load is determined according to the new upper bound by the formulae (9)-(10).

If the machine has no further feasible load then a backtrack must be made to the machine $i-1$ and machine i has no load anymore.

5 A New Integer Programming Model

As the dynamic programming problem formulation of [Blazewicz et al. 1991] has no practical importance, it was necessary to develop a new model. The aim is to compare the computing times and and the sizes of the solved problems if our above discussed method is applied or a general problem solver is applied on the model. Mathematical programming models for the simultaneous scheduling pf production and transportation in an FMS are known, see e.g. [Bilge-Ulusoy 1995], and [El Khayat et al. 2006]. But these models are more general. In the integer programming model discussed below all special properties of the FMS in question are used and therefore it is simpler than the previous models.

Based on the logic of the problem the following constraints must be formulated in a mathematical way:

1. Exactly one transportation must be assigned to each process and vice versa.

2. Each transportation must be feasible, i.e. it must start not later than the scheduled starting time of the process minus the transportation time from the palette station to the particular machine.

3. A *scheduling position* of a process is that on which machine in which position is the process. Each process must be assigned to a scheduling position. We assume that the number of tasks is higher than the number of machines, i.e. $n > m$, otherwise the problem is trivial. If the assumption holds then there is an optimal solution such that the number of the assigned tasks to each machine is t least 1. Hence no more than $n - m + 1$ task can be assigned to a machine.

4. At most one process can be assigned to each scheduling position.

5. The starting time of any process cannot be earlier then the completion time of the previous process on the same machine.

Furthermore to describe the objective function, i.e. the minimization of the makespan, some additional inequalities must be intoduced.

In the mathematical formalism four kinds of variables are used:

$$x_{ijk} = \begin{cases} 1 & \text{if process j is in the } k^{th} \text{ position on machine } i \\ 0 & \text{otherwise} \end{cases}$$

$$y_{jl} = \begin{cases} 1 & \text{if process j is served by transportation } l \\ 0 & \text{otherwise} \end{cases}$$

$$g_j = \text{the starting time of job } j.$$

$$h \;=\; \text{the value of the makespan.}$$

The mathematical forms of the constraints are the followings. The number of transportations is equal to the number of processes. This number is denoted by N. The relation of the transportations and the processes can be described by the usual constraints of the assignment problem:

$$\sum_{q=1}^{N} y_{jq} \;=\; 1 \quad j = 1, \ldots, N, \tag{16}$$

$$\sum_{j=1}^{N} y_{jq} \;=\; 1 \quad q = 1, \ldots, N. \tag{17}$$

Each tasks must be assigned to a machine and to a position on the machine. The number of the position, as it was mentioned above, cannot be greater that $n - m + 1$:

$$\sum_{i=1}^{m} \sum_{k=1}^{N-m+1} x_{ijk} \;=\; 1 \quad j = 1, \ldots, N. \tag{18}$$

On the other hand at most one job can be assigned to each position:

$$\sum_{j=1}^{N} x_{ijk} \;\leq\; 1 \quad i = 1, \ldots, m, \;\; k = 1, \ldots, N - m + 1. \tag{19}$$

If task j is assigned by machine i and served by transportation q then its starting time g_j must satisfy the inequality $e_q \leq g_j - \tau_i$. The assignments are described by the x and y variables. Hence the inequality

$$\sum_{q=1}^{N} e_q y_{jq} \;\leq\; g_j \;-\; \sum_{i=1}^{m} \tau_i \sum_{k=1}^{N-m+1} x_{ijk} \quad j = 1, \ldots, N \tag{20}$$

is obtained. No process can start earlier than the completion of all jobs scheduled before it on the same machine:

$$\sum_{s=1}^{N} \sum_{r=1}^{k-1} p_s x_{isr} \;\leq\; M(1 - x_{ijk}) g_j \quad i = 1, \ldots, m;\; j = 1, \ldots, N;\; k = 2, 3, \ldots, N - m + 1, \tag{21}$$

where M is an appropriate large pozitive number. The objective function constraints are these:

$$g_j \;+\; p_j \;\leq\; h \quad j = 1, \ldots, N. \tag{22}$$

Finally the technical constraints describing the type of the variables are as follows:

$$y_{jq} \;\in\; \{\, 0,\, 1 \,\} \quad j, q = 1, \ldots, N, \tag{23}$$

$$x_{ijk} \;\in\; \{\, 0,\, 1 \,\} \quad i = 1, \ldots, m;\; j = 1, \ldots, N;\; k = 1, \ldots, N - m + 1, \tag{24}$$

$$g_j \;\geq\; 0 \quad j = 1, \ldots, N, \tag{25}$$

$$h \;\geq\; 0. \tag{26}$$

6 Computational experiences

Three series of computational experiences have been carried out. All of them has the common property that the order of the process and transportation times are the same and even some τ_i's are greater than some p_j's. The reason is that if the process times are strictly larger than the transportation times then the problem becomes relatively easy and practically is equivalent to a $P \mid\mid C_{\max}$ problem.

In each of the three problem classes the following values are used:

$$\tau_i \;=\; i \quad i = 1, \ldots, m, \quad a \;=\; 1. \tag{27}$$

Our algorithm has been coded in Pascal.

6.1 Graham's example

Analysing the list scheduling of the $P \mid\mid C_{\max}$ problem [Graham 1969] gave an example that the performance ratio of the list scheduling is the worst possible on this example, i.e.

$$\frac{4}{3} \;-\; \frac{1}{3m}.$$

Later [Dosa 2004] proved that this example is the only one having this performance ratio and list scheduling gives a better solution on any other problem instances. As the FMS scheduling problem is related very closely to the $P \mid\mid C_{\max}$ problem it is natural to test the new tools on that problem class.

In the original example of Graham the number of parts is $2m+1$ in the case of m machines. The processing times are in decreasing order:

$$2m-1,\ 2m-1,\ 2m-2,\ 2m-2,\ \ldots,\ m+1,\ m+1,\ m,\ m,\ m.$$

Besides the data (27) the starting time of the transportation, i.e. e, was fixed to $-m-1$. The number of AGV's is $k = m$. Two different cycle times were used. In the first sequence of experiences $A = m+1$, in the other one $A = m+2$. The optimal solutions are the followings

The case $\mathbf{A = m+1}$. The optimal value of the appropriate $P \mid\mid C_{\max}$ problem is $3m$ and traditionally the optimal solution is written is the form:

$$\begin{aligned}
&\text{Machine 1: } 2m-1,\ m+1\\
&\text{Machine 2: } 2m-1,\ m+1\\
&\text{Machine 3: } 2m-2,\ m+2\\
&\qquad\ldots\\
&\text{Machine m: } m,\ m,\ m
\end{aligned} \tag{28}$$

The earliest transportation times are:

$$e_1 = -m-1,\ e_2 = -m, \ldots, e_m = -2,\ e_{m+1} = 0,\ \ldots,\ e_{2m} = m-1,\ e_{2m+1} = m+1. \tag{29}$$

Assume that the first m transportations are going in reversed index order to the machines. Then each AGV arrives at time -1. Thus each machine can start to work at time 0. If the same policy is applied to the second m transportations then each AGV arrives at time m with the second part to the machine. Thus Graham's example can be served if the machine having the three m-valued processing time is not the last one.

The case $\mathbf{A = m + \mu}$, where μ is a pozitive integer such that

$$2 \leq \mu \leq \left\lfloor \frac{m}{2} \right\rfloor .$$

Then the same loads are optimal, but they must be in different order. More precisely the loads of the first and last machines are interchanged. The first m transportations can be carried out as in the case $A = m + 1$. Then the first transportation of the second ones goes to the first machine and the further $m - 1$'s are assigned in decreasing index order to the machines. Finally the last transportation serves to bring the third job with processing time m to the first machine. Then the arrival times, denoted by AT_l, are these:

$$\begin{aligned} AT_{m+1} &= -m - 1 + m + \mu + 1 = \mu, \\ AT_{2m} &= -m - 1 + m + \mu + 1 + m = \mu + m, \\ AT_{2m-1} &= -m - 1 + m + \mu + 2 + m - 1 = \mu + m, \\ &\cdots \\ AT_{m+2} &= -m - 1 + m + \mu + m - 1 + 2 = \mu + m. \end{aligned}$$

The first starting time of a second job on a machine is m. This is the second job having processing time m on the first machine. The second starting time of a second job is

$$2m - \left\lfloor \frac{m}{2} \right\rfloor .$$

Thus a necessary condition to the feasibility of this assignment of jobs that the inequality

$$m + \mu \leq 2m - \left\lfloor \frac{m}{2} \right\rfloor$$

holds, what follows from the definition of μ. Finally the last transportation goes to the first machine, hence

$$AT_{2m+1} = -m - 1 + 2(m + \mu) + 1 = m + 2\mu.$$

Hence another feasibility condition is that

$$m + 2\mu \leq 2m, \tag{30}$$

what again follows from the definition of μ. Constraint (30) also shows that if

$$\mu > \left\lfloor \frac{m}{2} \right\rfloor$$

then the system has no feasible solution. Finally it must be mentioned that if $\mu < \left\lfloor \frac{m}{2} \right\rfloor$ the three m processing time can be assigned to other machines than the first one, i.e. alternative optimal solutions exist.

In the computational experiences μ was fixed to 2 in the second case. In all cases an optimal solution was found and its optimality has been proven. The CPU times are reported in the following table:

m	CPU time (sec)	
	$\mu = 1$	$\mu = 2$
10	0.61	0.60
20	2.25	4.06
30	3.63	6.65
40	9.34	27.90
50	24.50	11.04
60	18.94	29.71
70	42.12	51.08
80	58.25	31.42
90	37.79	39.00
100	82.34	119.52
110	37.24	89.25
120	79.70	107.05
130	193.06	136.60
140	229.04	160.32
150	170.65	171.59
160	234.14	243.76

Table 1. Computational experiences with Graham's example

6.2 Random problems with $U(1, 99)$ processing times

In two series of experiments the processing times were random pozitive integer numbers drawn from the uniform distribution $U(1, 99)$. Although many experiences have been carried out in both series here only the results of the largest problems are reported. In each problem size 10 problems have been generated. The minimal, maximal and average CPU times concern to these 10 instances. The same is true for the series of experiences discussed in the next subsection.

Seies No. 1: $m = 3$. In this series the number of machines is fixed. The other parameters are as follows: $e = -4$, $A = 4$, $k = 3$. Our program was able to solve problems having not more than 760 jobs. Above this limit it had memory problems although the used CPU time is still very small. The short CPU time can be explained by the fact that the depth of the enumeration, i.e. the depth of the stack, is never more than 3.

# of jobs	**CPU time (sec)**		
	min	**max**	**avg**
500	0.05	0.11	0.06
600	0.05	0.06	0.05
700	0.05	0.06	0.06
760	0.05	0.11	0.06

Table 2. Computational experiences with random problems having $U(1, 99)$ processing times and $m = 3$.

Seies No. 2: $m = N/5$. In this series the ratio of number of machines and the number of jobs is fixed. The other parameters are as follows: $e = -m - 2$, $A = m + 1$, $k = m$. Here the depth of the enumeration is much greater than in the previous case. This is the reason that our program was able to solve problems with significantly less number of jobs.

# of jobs	# of machines	CPU time (sec)		
		min	max	avg
100	20	0.50	1.97	1.09
110	22	0.93	2.63	1.56
120	24	0.55	1.04	1.10

Table 3. Computational experiences with random problems having $U(1, 99)$ processing times and $m = N/5$.

6.3 Random problems with $U(5, 15)$ processing times

Here two similar series of experiences have been carried out.

Seies No. 1: $m = 3$. Here problems with very high number of jobs have been solved. On the other hand this is the only case when not all of the problems having a large size were solved until optimality within a time limit of 3600 seconds. In all of the four such cases the objective function value of the best known feasible solution was greater than the lower bound only by 1. The objective function value of these four cases was always between 10,690 and 11,420. The table below gives the number of solved problems, too. The CPU times concern only to the exactly solved cases. If the number of jobs exceeded 3650 the Pascal program had again memory problems. The parameters are as follows are the same as in the case of $U(1, 99)$, i.e. $e = -4$, $A = 4$, $k = 3$.

# of jobs	# of solved problems	CPU time (sec)		
		min	max	avg
3000	10	0.10	0.17	0.14
3200	8	0.22	0.28	0.24
3400	8	0.22	0.44	0.37
3600	10	0.16	0.22	0.17
3650	10	0.22	0.39	0.31

Table 4. Computational experiences with random problems having $U(5, 15)$ processing times and $m = 3$.

Seies No. 2: $m = N/5$. In this series the ratio of number of machines and the number of jobs is fixed in the same way as in the previous case. The other parameters are as follows: $e = -m - 5$, $A = m + 1$, $k = m$. If the number of jobs was greater than 50 then the solution time exceded the time limit of 3600 seconds. Among the largest problems there is one problem, which is still solved exactly, but its running time is extremely longer than that of the other ones.

# of jobs	# of machines	CPU time (sec)		
		min	max	avg
30	6	0.00	0.22	0.06
40	8	0.00	0.16	0.07
50	10	0.06	2386.62	238.98

Table 5. Computational experiences with random problems having $U(5, 15)$ processing times and $m = N/5$.

6.4 Computational Experiences with Integer Programming Model

To test the effectiveness the new algorithm the integer programming model was solved by commercial solver.

The integer programming model has been solved by both LINGO 9 (LINDO 2005) and CPLEX professional solvers in the case of Graham's example with $A = m + 1$. LINGO found the optimal solution in five cases but it was unable to prove its optimality within the time limit of two hours. CPLEX solved the problem for $m = 3$ in 10.56 seconds and found the optimal solution of nine further problems but was unable to prove its optimality within the same time limit. Here are the computational results in in details:

m	CPU time until finding the optimal solution	
	LINGO	CPLEX
3	3	0.5
4	14	1.33
5	300	4.00
6	550	7.60
7	–	39.92
8	–	49.03
9	–	263.64
10	–	736.71
11	–	1403.27

7 Conclusions

A new algorithm has been developed for the simultenous scheduling of production and transportation is a specially structured FMS.

References

[Bilge-Ulusoy 1995] Bilge, Ü., Ulusoy, G., A Time Window Approach to Simultaneous Scheduling of Machine and Material Handling, Operations Research, 43(1995), 1058-1070.

[Blazewicz et al. 1991] Blazewicz, J., Eiselt, H.A., Finke G., Laporte, G., Weglarz, J., Scheduling Tasks and Vehicles in a Flexible Manufacturing System, The Internatinal Journal of Flexible Manufacturing Systems, 4(1991), 5-16.

[Demir-Vizvári 1995] Demir, R., Vizvári, B., A column generation algorithm to schedule identical parallel machines, PU.M.A. (PUre Mathematics and Application), 6(1995), 287-299.

[Dosa 2004] Dósa, Gy., Graham's example is the only tight one for $P \mid\mid C_{\max}$, Annales Univ. Sci. Budapest., 47 (2004), 207-210.

[El Khayat et al. 2006] El Khayat, G., Langevin, A., Riopel, D., Integrated production and material handling scheduling using mathematical programming and constraint programming, European Journal of operations Research, 175(2006), 1818-1832.

[Graham 1969] Graham, R.L., Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17(1969) 416-429.

[Roh-Kim 1997] Roh, H.K., Kim, D., Due-date based loading and scheduling methods for a flexible manufacturing system with an automatic tool transporter, International Journal of Production Research, 35(1997), 2989-3003.

[Rothkopf 1966] Rothkopf, M.H., Scheduling independent tasks on parallel machines, Management Science, 12(1966), 437-447.

Appendix A

An optimal solution of a problem with $m = 3$, $N = 760$. The processing times are drawn from the distribution$U(1, 99)$. If there is a number behind the processing time then it means that how many jobs are on the machine having that processing time. If there is no number behind the processing time then there is only one such job.

$M1$	99, 98/11, 97/10, 96/9, 95/9, 94/11, 93/7, 92/5, 91/4, 90/8, 89/6, 88/3, 87/8, 86/7, 85/2, 84/3, 83/7, 82/7, 81/9, 80/8, 79/2, 44
$M2$	79/8, 78/12, 77/7, 76/5, 75/7, 74/5, 73/11, 72/6, 71/9, 70/5, 69/6, 68/6, 67/4, 66/8, 65/11, 64/6, 63/11, 62/8, 61/5, 60/7, 59/11, 58/8, 57/8, 56/7, 55/2, 31
$M3$	55/8, 54/6, 53/6, 52/10, 51/6, 50/4, 49/12, 48/4, 47/8, 46/5, 45/6, 44/5, 43/8, 42/5, 41/6, 40/9, 39/12, 38/12, 37/13, 36/13, 35/13, 34/12, 33/8, 32/12, 31/8, 30/5 29/4, 28/4, 27/9, 26/8, 25/7, 24/10, 23/3, 22/8, 21/13, 20/10, 19/10, 18/15, 17/6, 16/10, 15/7, 14/10, 13/5, 12/6, 11/5, 10/7, 9/7, 8/9, 7/8, 6/12, 5/12, 4/6, 3, 2/10

Appendix B

An optimal solution of a problem with $m = 24$, $N = 120$. The processing times are drawn from the distribution$U(1, 99)$. If there is a number behind the processing time then it means that how many jobs are on the machine having that processing time. If there is no number behind the processing time then there is only one such job.

$M1$	99, 98, 75
$M2$	96, 95, 81
$M3$	95/2, 80
$M4$	94, 93, 85
$M5$	93/2, 86
$M6$	92/2, 87
$M7$	91, 90, 86, 5
$M8$	84, 83/2, 22
$M9$	80, 79, 77, 36
$M10$	76/3, 44
$M11$	75, 74, 73, 50
$M12$	71, 70/2, 61
$M13$	68/4
$M14$	67, 66/3, 7
$M15$	65/3, 64, 13
$M16$	63/2, 62/2, 21
$M17$	62, 59/2, 34
$M18$	57/3, 56, 45
$M19$	56, 55/3, 49
$M20$	54, 52/3, 48, 14
$M21$	48, 47, 46, 44, 43, 42
$M22$	41/2, 40/4, 29
$M23$	37, 32, 31/2, 28, 27/4
$M24$	26, 24, 23, 21, 20/3, 17, 16/3, 15, 12, 11, 7, 6

Appendix C

An optimal solution of a problem with $m = 3$, $N = 3650$. The processing times are drawn from the distribution $U(5, 15)$. If there is a number behind the processing time then it means that how many jobs are on the machine having that processing time. If there is no number behind the processing time then there is only one such job.

$M1$	15/172, 14/379, 13/334
$M2$	13/28, 12/398, 11/364, 10/308, 9
$M3$	10/54, 9/362, 8/361, 7/373, 6/361, 5/154

Appendix D

An optimal solution of a problem with $m = 10$, $N = 50$. The processing times are drawn from the distribution $U(5, 15)$. If there is a number behind the processing time then it means that how many jobs are on the machine having that processing time. If there is no number behind the processing time then there is only one such job.

$M1$	15/3
$M2$	15, 14, 13, 5
$M3$	14, 13/2, 7
$M4$	13/2, 12, 9
$M5$	11/3, 9, 5
$M6$	11/3, 9, 5
$M7$	11, 10/3, 6
$M8$	10, 9, 8/2, 7, 5
$M9$	8/2, 7, 6/4
$M10$	7/4, 6/3