Scheduling Unit-Length Parts On Identical Machines With A Constant Setup for Different Part Types

Alekos Triantafyllakis and Spyros G. Tzafestas

Intelligent Robotics and Control Unit
Department of Electrical and Computer Engineering
National Technical University of Athens
Zografou, Athens 15773
GREECE

Abstract: The problem of scheduling unit length tasks on identical machines in an industrial production unit is considered. The tasks belong to k part types, such that any task may and must belong to one part type. Every machine needs a setup time, say S, when changing from one part type to another. The problem is to obtain an assignment of tasks on machines such that the makespan is minimized. For the case of k=2 part types, a method that leads to the optimal solution (i.e.to the minimum makespan), as well as the associated algorithm are presented. For the case of k>2 part types, a method is presented that reduces the complexity of the optimal algorithm, and helps in the construction of approximate heuristic algorithms. A 2-part type example is provided that illustrates the application of the optimal algorithm, and shows the effect of the machines set-up time upon the optimal schedule.

1. Introduction

The problem under consideration in this paper belongs to the scheduling area, and is briefly as follows. There is a set of m identical machines and n unit length independent and nonpreemptive tasks. The tasks belong to k part types, such that any task may and must belong to one type. There is a setup time, say S, on any machine when changing from one part type to another, provided that both tasks belong to different part types. The setup is independent of the machines and the part types (i.e. it is a constant parameter). No more than one part can be processed by any machine at a time and machine capacity is not a restriction. For a given solution (i.e. the assignment of tasks on machines), the workload of each machine can be computed as the number of tasks that have been assigned on the machine plus S times the number of changeovers. The problem is to obtain an assignment (and scheduling) of tasks on machines such that the greatest workload (makespan) is minimized.

In the literature,many special and general cases of this problem have been studied and solved. In [1], a number of dispatching rules for such scheduling problems are exposed, and a comparison of their effectiveness features is made. In [2-4], several problems of this type are investigated, while in [5] a survey of solutions for single machine problems is made.

Particularly, in [5] it is shown that most problems of this type, (even for the single machine case: m=1) are NP-complete problems [6]. That is why many works were devoted to finding special models and subproblems, for which an optimal solution can be found in finite (polynomial) time. Two excellent surveys on the general scheduling problem can be found in [7,8] and two comprehensive textbooks are given in [9,10]. A study of the complexity of computer algorithms is presented in [11].

In this paper we will be concerned with the case of $m \geq 2$ identical machines, arbitrary numbers of unit length tasks, and an arbitrary set-up time. The optimization goal is to minimize the makespan.

The structure of the paper is as follows. In Section 2, the problem is precisely formulated, and in Section 3 the methodology for determining the optimal solution for the 2-part type case is presented, together with the proof of optimality. Section 4 contains the detailed steps of the optimal scheduling algorithm. In Section 5, we provide a representative illustrative example. Finally, in Section 6, we generalize the results to the case of more than two task part types, and in Section 7, we present some concluding remarks.

2. Optimal Scheduling Problem Formulation

In this section we formulate the problem of two part types. We are given:

- m identical machines, all available from time t=0, without any machine capacity restriction.
- n unit length independent nonpreemptive tasks (n_A tasks of part type A and n_B tasks of part type B where $n_A+n_B=n$)
- a setup time S needed by every machine for changing from one part type to the other.

At time t=0 no setup is needed by the machines to start execution of tsks of any part type.

We define the following parameters:

- i: machine index
- n_{Ai}: the number of tasks of part type A assigned on machine i (i = 1,...,m)
- n_{Bi}: the number of tasks of part type B assigned on machine i (i = 1,...,m)
- x_i: the number of setups on machine i
- W_i: the workload of machine i, $W_i=n_{Ai}+n_{Bi}+Sx_i$
- W_{max}: the makespan, $W_{max}=\max\{W_i\}$.

Problem:

Find n_{Ai}, n_{Bi} and x_i, (i=1,...,m) such that W_{max} is minimized.

3. Scheduling Two Part Types of Unit Length Tasks

Here we shall present the following proposition, which will help us in constructing the optimal scheduling algorithm of complexity O (logn).

Proposition

At least one optimal solution of the above problem (Sec.2) has the property that

$$\sum_{i=1}^{m} x_i \leq 1.$$

Proof

By definition, the optimal solution is the solution that provides the minimum makespan among all feasible schedules. All feasible schedules can be searched through the following two search levels.

Level 1 (For each machine independently of the other machines)

(a) The machine does not make any changeover

(b) The machine one changeover

(c) The machine makes at least two changeovers

Level 2 (For the whole set of machines)

(a) No machine makes any changeover

(b) One machine makes at least one changeover

(c) At least two machines make at least one changeover

Clearly, any feasible schedule may involve combinations of cases belonging to both levels. For a machine to make *l* changeovers at level 1, there must occur *l* transitions (interchanges) among the part types under execution. With a procedure by which all tasks of part type A are put at the beginning, and those of part type B at the end, we can obtain a scheme that contains at most one changeover in each machine, and gives a makespan less than, or equal to those corresponding to case (c) of level 1. Thus, we have reduced the possible cases to (a) and (b).

This will now be made for the feasible schedules of level 2. Suppose that we have a feasible schedule in which p machines ($p \leq m$) are making changeover. It is obvious that each one of them makes one changeover according to the case (b) at level 1. Let us represent pictorially the partial schedules for two of these p machines (Figgure 1).

We can now apply to these two machines a cyclic procedure, which let us we give one task of part type A from machine i to machine j, and one task of part type B from machine j to machine i. In this way we have

$n_{Bi}'=n_{Bi}+1$

$n_{Bj}'=n_{Bj}-1$

$n_{Ai}'=n_{Ai}-1$

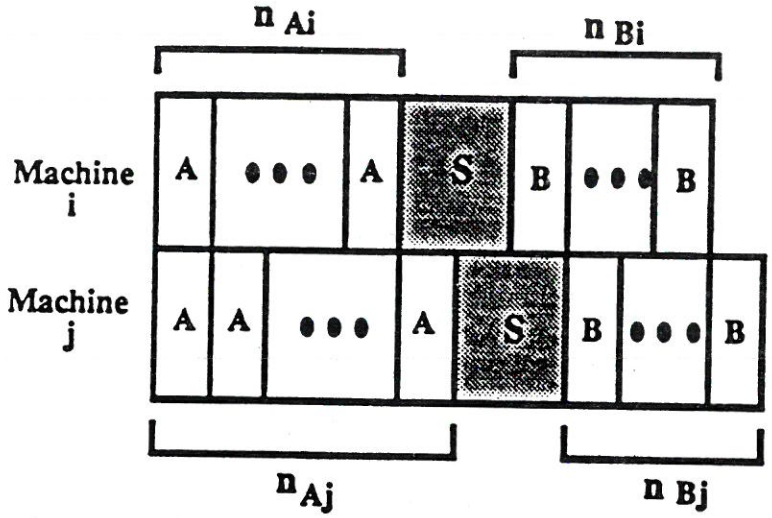

Figure 1. Partial Schedule with Two Machines

$n_{Aj}'=n_{Aj}+1$

from which it follows that

$n_{Bi}'+n_{Ai}'=n_{Bi}+n_{Ai}\Rightarrow n_{Bi}'+n_{Ai}'+S=$

$=n_{Bi}+n_{Ai}+S\Rightarrow W_i'=W_i$

$n_{Bj}'+n_{Aj}'=n_{Bj}+n_{Aj}\Rightarrow n_{Bj}'+n_{Aj}'+S=$

$=n_{Bj}+n_{Aj}+S\Rightarrow W_j'=W_j$

This means that during this cyclic procedure the two machines maintain their workloads unchanged. However, continuing this procedure takes us to one of the three cases below:

(a) Machine i does not make changeover and has only B-tasks ($n_{Ai}<n_{Bj}$).

(b) Machine j does not make changeover and has only A-tasks ($n_{Ai}>n_{Bj}$).

(c) Machines i and j do not make changeover and have only B-tasks or A-tasks respectively ($n_{Ai}=n_{Bj}$).

One observes that in each of these three cases the partial makespan has been reduced by $S(n_{Ai}=n_{Bj})$, or at most remained the same, since only the workload of only one machine has been reduced ($n_{Ai}\neq n_{Bj}$). Thus the new schedule has less than, or equal lmakespan to the original one, but one or two machines making changeover less (k-1 or k-2). On repeating the same procedure for this schedule the situation is reached when only one, or no machine makes changeover, and only a single changeover exists, if it does. Thus, for each feasible schedule of case (c) of level 2, there is a schedule of cases (a) and (b) of the same level that gives smaller, or at most equal makespan. The above implies that indeed the optimal solution has the property that an optimal schedule involves at most one changeover. This completes the proof of the proposition.

Therefore, in the algorithm that follows, we search the optimal solution between these two cases. More specifically, we determine the best solution for each one of these cases, and the best of them is the desired optimal solution of the problem.

4. The Optimal Scheduling Algorithm

Before presenting the steps of the optimal algorithm we have to give some definitions.

- $m_A(m_B)$is the number of machines that execute only part type A (B) tasks
- Round (x) is the closest integer to x. In

particular if x equals an integer, say y, plus 0.5 then Round (x) = y+1

- [x] is the closest integer to x, greater than or equal to x.
- $D(a,b)=\begin{cases} a-b & \text{if } a>b \\ 0 & \text{otherwise} \end{cases}$

Without loss of generality we assume that $n_A \geq n_B$. The steps of the algorithm in pseudocode mode are as follows:

Step 1:

$m_A=\text{Round}\left(\frac{n_A}{n}m\right)$

If $m_A=m$ then $m_A=m-1$

$m_B=m-m_A$

Step 2:

$n_{Ai}=0,\ i=m_A+1,...,m$

If $m_A=1$ then $W_1=n_{A1}=n_A$ else

Set $W_i=n_{Ai}=\left[\frac{n_A}{m_A}\right]$, $i=1,...,m_A-1$ and

$W_{m_A}=n_{Am_A}=n_A\left(m_A-1\right)$

$n_{Bi}=0,\ i=1,...,m_A$

If $m_B=1$ then set $W_m=n_{Bm}=n_B$ else

Set $W_i=n_{Bi}=\left[\frac{n_B}{m_B}\right]$, $i=m_A+1,...,m-1$ and

$W_m=n_{Bm}=n_B-n_{Bm-1}(m_B-1)$

Set $x_i=0,\ i=1,...,m$

Step 3:

$W_{max1}=\max\limits_i\left\{W_i\right\}$

Step 4:

Set $n_A=n_A-1,\ n_B=n_B-1,\ n=n-2$

Set $m_A=\text{Round}\left(\frac{n_A}{n}(m-1)\right)$, $m_B=m-1-m_A$

If $m_A<m-1$ then goto STEP 5 else continue

If $m_A=m-1$ then goto STEP 8 else continue

Step 5:

$n_{Ai}=0,\ i=m_A+1,...,m-1$

If $m_A=1$ then $W_1=n_{A1}=n_A$ else

Set $W_i=n_{Ai}=\left[\frac{n_A}{m_A}\right]$, $i=1,...,m_A-1$ and

$W_{m_A}=n_{Am_A}=n_A-n_{A1}\left(m_A-1\right)$

$n_{Bi}=0,\ i=1,...,m_A$

If $m_B=1$ then $W_{m-1}=m_{Bm-1}=n_B$ else

Set $W_i=n_{Bi}=\left[\frac{n_B}{m_B}\right]$, $i=m_A+1,...,m-2$ and

$W_{m-1}=n_{Bm-1}=n_B-n_{Bm-2}(m_B-1)$

Set $n_{Bm}=n_{Am}=1,\ W_m=S+2$

Step 6:

Find an integer T such that $\max\limits_i\left\{C_i\right\}$ is minimized, where

$C_i=W_i-D(W_i-T),\ i=1,...,m-1$

$C_m=S+2+\sum\limits_{i=1}^{m-1}D\left(W_i-T\right)$

Step 7:

Set $n_{Am}=1+\sum\limits_{i=1}^{m_A}D\left(W_i-T\right)$,

$n_{Bm}=1+\sum\limits_{i=m_A+1}^{m-1}D\left(W_i-T\right)$

Set $n_{Ai}=W_i-D(W_i-T),\ i=1,...,m_A,\ n_{Ai}=0,\ i=m_A+1,...,m-1$

Set $n_{Bi}=W_i-D(W_i-T),\ i=m_A+1,...,m-1,\ n_{Bi}=0,\ i=1,...,m_A$

$x_i=0,\ i=1,...,m-1,\ x_m=1$

Set $W_i=n_{Ai},\ i=1,...,m_A$ and

$W_i=n_{Bi},\ i=m_A+1,...,m-1$

Set $W_m=2+S+n_{Am}+n_{Bm}$

Set $W_{max2}=\max\limits_i\left\{W_i\right\}$

goto STEP 13

Step 8:

If $m=2$ then goto STEP 10

$n_{Ai}=0$, i=1,...,m-1 and $n_{Am}=1$

$n_{Bi}=0$, i=1,...,m-2 and

$W_{m-1}=n_{Bm-1}=n_B$ and $n_{Bm}=1$

If m=3 then $n_{A1}=n_A$

if m>3 then $W_i=n_{Ai}=\left[\frac{n_A}{m-2}\right]$, i=1,...,m−3 and

$W_{m-2}=n_{Am-2}=n_A-n_{A1}(m-3)$

Step 9:

Find an integer T such that $\max_i\{C_i\}$ is minimized, where

$C_i=W_i-D(W_i-T)$, i=1,...,m-1 and

$C_m=S+2+\sum_{i=1}^{m-1} D\left(W_i-T\right)$.

Set $W_{max2}=\max_i\{C_i\}$

Set $n_{Ai}=W_i-D(W_i-T)$, i=1,...,m-2, $n_{Am-1}=0$

Set $n_{Bi}=0$, i=1,...,m-2 and $n_{Bm-1}=W_{m-1}-D(W_{m-1}-T)$

Set $n_{Am}=1+\sum_{i=1}^{m-2} D\left(W_i-T\right)$ and

$n_{Bm}=1+D(W_{m-1}-T)$

Set $x_i=0$, i=1,...,m-1, $x_m=1$.

Step 10:

Set $n_{Bi}=0$, i=1,...,m-1 and $n_{Bm}=n_B+1$

Set $W_i=n_{Ai}=\left[\frac{n_A}{m-1}\right]$, i=1,...,m−2

Set $W_{m-1}=n_{Am-1}=n_A-n_{A1}(m-2)$ and $n_{Am}=1$

Step 11:

Find an integer T such that $\max_i\{C_i\}$ is minimized, where

$C_i=W_i-D(W_i-T)$, i=1,...,m-1

$C_m=S+2+n_B+\sum_{i=1}^{m-1} D\left(W_i-T\right)$.

Set $W''_{max2}=\max_i\{C_i\}$

Set $n_{Ai}=W_i-D(W_i-T)$, i=1,...,m-1 and

$n_{Bi}=0$, i=1,...,m-1

Set $n_{Am}=1+\sum_{i=1}^{m-1} D\left(W_i-T\right)$ and $n_{Bm}=1+n_B$

Set $x_i=0$, i=1,...,m-1 and $x_m=1$

Step 12:

If m≥3 then set $W_{max2}=\min\left\{W'_{max2},W''_{max2}\right\}$

else set $W_{max2}=W''_{max2}$

Step 13:

$W_{max}=\min\left\{W_{max1},W_{max2}\right\}$

Step 14:

If the optimal makespan W_{max} is W_{max1} or W_{max2} or W'_{max2} or W''_{max2} then the optimal values for n_{Ai}, n_{Bi}, x_i(i = 1,...,m) are given by steps 2,7,9 or 11 respectively.

The algorithm examines the cases with one changeover (on machine m) and without changeover in the schedule, and selects among them the best one. Steps 1,2 and 3 provide the best (minimum) makespan for the case where there is no setup on any machine. In this case the machines are split into two groups (m_A, m_B) with respect to n_A and n_B. Then the n_A and n_B tasks are distributed among the m_A and m_B machines respectively on an equal basis.

Steps 4 through 12 examine the case where there is exactly one setup only on machine m, providing the best solution with a setup. Having defined m_A and m_B Step 6 or 9 or 11 in this case distributes the tasks among the respective machines on an equal basis to have the most balanced machine workloads. Some extreme cases e.g. m=2 or $m_B=0$ are also considered by the algorithm. Finally Step 13 provides the optimal W_{max}, and Step 14 the final optimal values of n_{Ai}, n_{Bi} and x_i (i=1,...,n). Having now the n_{Ai}'s, n_{Bi}'s and x_i's all tasks of the same part type on a given machine are scheduled consecutively. Therefore the final optimal schedule can easily be constructed.

From the proposition and the construction of the algorithm it follows that the derived solution is optimal.

The complexity of the algorithm is dominated by the complexity of Steps 6,9 and 11 (i.e. O (logn)) since a binary search can be applied between integers proportional to n.

5. Example

In this section we present a simple but illustrative example for the problem at hand.

Consider a two part type scheduling problem with the following data:

$$m=5,\ n_A=79,\ n_B=63,\ s=8.3$$

Applying the presented optimal algorithm we get as intermediate results the following:

After Step 3: Wmax1 = 32 (case without setup on machine).

After Step 7: Wmax2 = 30.3 (case with one setup on machine m).

Solving this example the algorithm did not execute steps 8 to 12. Finally Step 13 finds the optimal $W_{max}=30.3$ and Step 14 provides the optimal values of n_{Ai}, n_{Bi} and x_i (i=1,...,m) shown in Table 1.

Table 1. Optimal Solution

i	1	2	3	4	5
n_{Ai}	30	30	0	0	19
n_{Bi}	0	0	30	30	3
x_i	0	0	0	0	1
W_i	30	30	30	30	30.3
Wmax = 30.3					

As we have already mentioned, all tasks of the same part type on a given machine are scheduled consecutively. On machine m we first schedule all 19 tasks of part type A, then there is the setup S and finally the three tasks of part type B consecutively. Figure 2 shows the Gantt chart of this optimal solution.

At this point it is very useful to examine the same problem from another point of view. We would like to find how the setup S affects the optimal makespan, as well as the decision on whether to have a setup or not. Applying the algorithm for different values of S we get the results shown in Figure 3.

From this Figure we conclude that given m, and there is a threshold of S if and only if S is less than this threshold then the optimal solution always includes one setup.

6. The Case of Scheduling Many Task Part Types

In this section, a useful property will be established for the case of many (more than two) task part types when the set-up time is the same for any switching among them.

Consider k part types of unit length tasks, namely $A_1, A_2, \ldots, A_k$. The different changeovers that might be in a schedule are A_1-A_2, A_1-A_3, ...,A_1-A_k, A_2-A_3,... Their number is equal t

$$\binom{k}{2}=\frac{k(k-1)}{2}$$

Suppose we have an initial schedule. Using the methodology (theorem) presented in the paper, we can transform this schedule into some other schedule of smaller, or at most equal schedule length, which will not contain the same changeover more than two times. By generalizing this theorem, and by applying it to the resulting schedule, we further reduce the number of changeovers.

On the basis of this schedule, we can draw a graph containing k nodes, one for each part type. If the schedule involves a changeover A_i-A_j between the task part types A_i and A_j, then the graph contains the corresponding arc, that connects node i with node j. For example suppose that we have k=5 part types and that the schedule contains the changeovers A_1-A_2, A_2-A_3, A_3-A_4, A_2-A_4 and A_3-A_5. Then the part type graph is the following (Figure 4.)

If the graph possesses a closed path (cycle), such as A_2-A_3-A_4-A_2, then we make a cyclic exchange

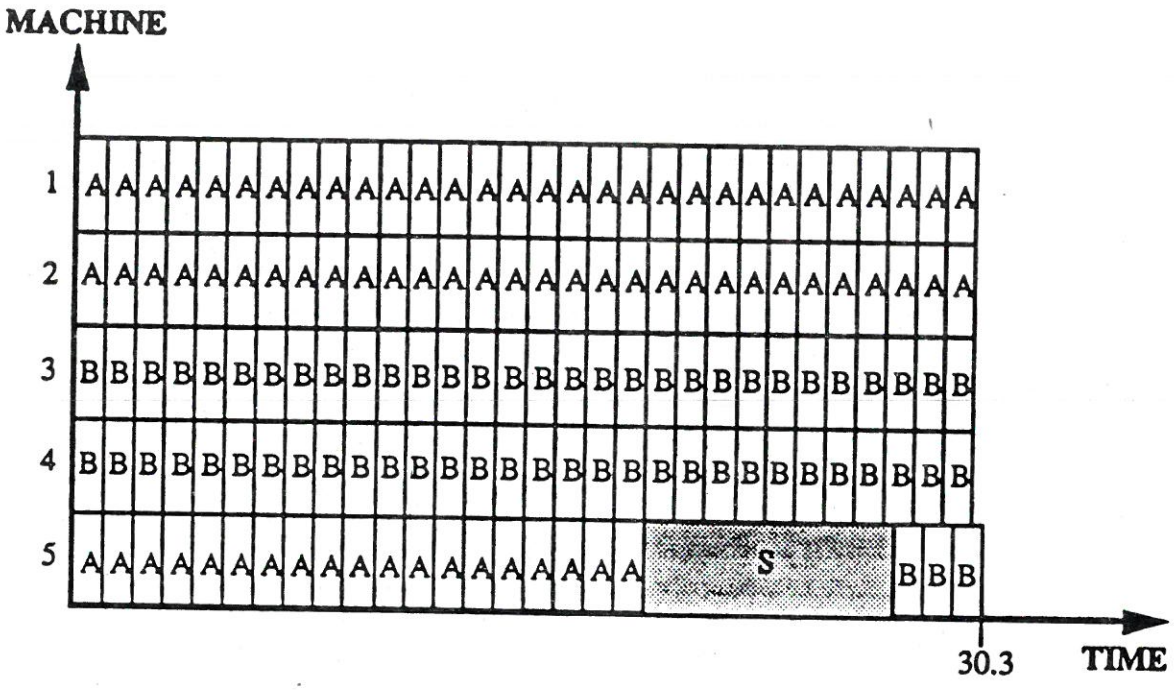

Figure 2. Gantt Chart of Example

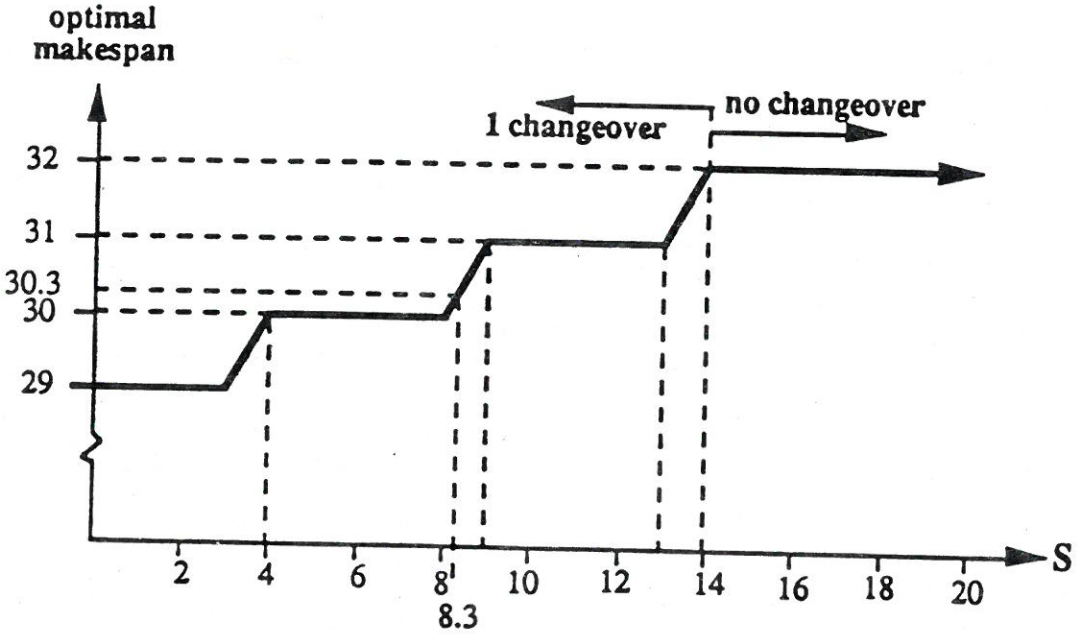

Figure 3. Effect of Setup Time on Optimal Makespan

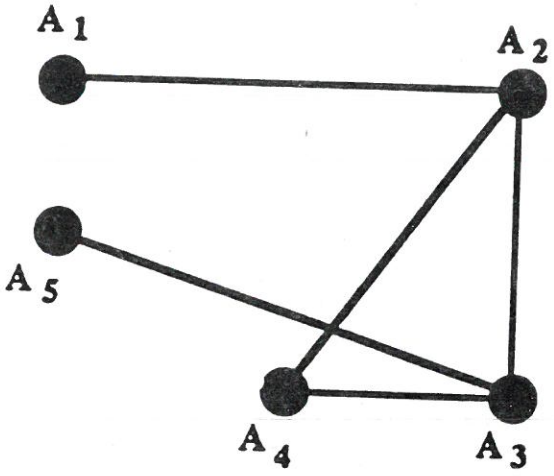

Figure 4. Part Type Graph

of tasks in the schedule, as follows. We remove an A_2-task from the section that contains the changeover A_4-A_2, and put it into the schedule that contains the changeover A_2-A_3. Then we move an A_3 -task from A_2-A_3 to A_3-A_4 and finally an A_4-task from A_3-A_4 to A_4-A_2. In this way the schedule lengths of all sections remain unchanged. If we continue this cyclic process in the same exchange direction, we will arrive at a point where one or more changeovers do not exist anymore in the schedule, and in the corresponding graph at least one arc of the closed path is removed, while some arcs are replaced by others. The important point is that, certainly, with this cyclic permutation, the number of arcs in the graph is finally reduced. The question now arises. Can we repeat the above cyclic process on the new graph, and if yes,up to what point? The answer to the first question is obvious. Yes, we can repeat this procedure as long as the new graph involves at least one closed path. To answer the second question, the following property of geometry must be involved. "**If on a plane we have n points, not belonging to a straight line every three, and l straight segments that connect l pairs (l≥n) of points, then there exists at least one closed line (path)**". From this property it follows that we can repeat the cyclic procedure at least until there remain k-1 arcs in the graph, or equivalently, k-1 changeovers in the corresponding schedule.

In other words, we have shown that having any schedule, we can get a better (or in the worst case an equivalent) schedule, which has at most k-1 changeovers, as it happens with an optimal schedule. The above method does not of course solve the problem which remains NP-complete (k, m, n_i: arbitrary), but we reduce considerably the complexity of the optimal algorithm. and contribute to developing efficient heuristic algorithms.

7. Concluding Remarks

In this paper, an effort was made to provide an optimal solution to a scheduling problem not studied in the past (according to the authors'knowledge). This problem is based on the idea of batch processing. The approach made, guarantees a fast and optimal solution to the problem of two part types, and simplifies substantially the solution to the problem of more than two part types (i.e. k>2), that at most k-1 changeovers are possible in each schedule. For k=2 there is at most only one changeover.

Using the methodology presented in this paper (k=2), we can also solve a different, practical and more difficult problem than the original one: Given n_A, n_B m, find the maximum set-up time S (if any exists), such that the schedule length does not exceed a given deadline T. This problem can be solved using the corresponding Figure 3, that relates the optimal schedule length, and the set-up time S. The results of the paper can find important applications since most manufacturing processes involve more than one part type. The authors are currently in contact with a local discrete manufacturing company to identify particular practical problems that can be solved by the present algorithm.

Other results derived by the authors for various scheduling models and problems can be found in [12-15]. Although many and important results are available in the industrial scheduling area, much research is needed for identifying and solving practical scheduling problems, that can be solved efficiently in polynomial time.

REFERENCES

1. WILBRECHT, J.K. and PRESCOTT, W.B., **The Influence of Setup Time in Job Shop Performance,** MANAGEMENT SCIENCE, Vol.16, No.4, 1969, pp. B- 274-B-280.
2. HU, T.C., KUO, Y.S. and RUSKEY, F., **Some Opimum Algorithms for Scheduling Problems with Changeover Costs,** OPERATIONS RESEARCh, Vol.35, No.1, 1987, pp.94-99.
3. BITRAN, R, HAAS, E.A. and MATSUO, H., **PRoduction Planning of Style Goods with High Setup Costs and Forecast Revisions,** OPERATIONS RESEARCH, Vol.34, No.2, 1986, pp.226-236.
4. LOVE, Jr, R.R. and VEMUEANTI, R.R., **The Single Plant Mold Allocation Problem with Capacity and Changeover Restrictions,** OPERATIONS RESEARCH, Vol.26, No.1, 1978, pp.159-165.
5. BRUNO, J. and DOWNEY, P., **Complexity of Task Sequencing with Deadlines, Set-up Times and Changeover Costs,** SIAM. J. COMPUT., Vol.7, No.4, 1978, pp.393-404.
6. GAREY, M. R. and JOHNSON, .S., **Computers and Intractability: A Guide to the Theory of NP-completeness,** W.H. FREEMAN, San Francisco, CA, 1979.
7. BLAZEWICZ, J., **Selected Topics in Scheduling Theory,** ANNALS DISCRETE MATH., Vol.31, 1987, pp.1-60.
8. RODAMMER, F. R. and PRESTON WHITE Jr, K., **A Recent Survey of Protection Scheduling,** IEEE TRANS. SYST. MAN AND CYBERN., Vol.18, No.6, 1989, pp.841-851.
9. BAKER, K. R., **Introduction to Sequencing and Scheduling,** J. WILEY & SONS, New York, 1974.
10. LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN A. H. G. and SHMOYDS, D. B., **Sequencing and Scheduling: Algorithms and Complexity,** NORTH HOLLAND, Amsterdam, 1989.
11. AHO, A., HOCROFT, J. and ULLMAN, J., **The Design and Analysis of Computer Algorithms,** ADDISON-WESLEY, New York, 1974.
12. TZAFESTAS, S. and TRIANTAFYLLAKIS, A., **Optimal Task Scheduling via the LPT and MULTIFIT Heuristics Under a Geometric Progression Condition,** FOUND. COMPUTING & DECISION SCI. Vol.17, No.3, 1992.
13. TZAFESTAS, S. and TRIANTAFYLLAKIS, A., **A New Adaptively Weighted Combinatorial Dispatching Rule for Complex Scheduling Problems,** COMPUTER INTEGRATED MANUF. SYST., Vol.7, No.1, 1994, pp.7-15.
14. TRIANTAFYLLAKIS, A. and TZAFESTAS, S., **Parallel Scheduling of Tasks with AND/OR Dependencies,** SYSTEMS SCIENCE, Vol.21, No.2, 1995, pp.45-54.
15. TRIANTAFYLLAKIS, A. and TZAFESTAS, S., **Optimal Grouping of Dependent Tasks for Parallel Scheduling,** Proc. IMACS Intl.Symp. on Parallel and Distributed Computing, Corfu, Greece, June 1991.