Cyclic Machine Scheduling with Tool Transportation

Cindy Kuijpers

2001

Ph.D. thesis
University of Twente

Twente University **Press**

Also available in print:
www.tup.utwente.nl/uk/catalogue/technical/cyclic-scheduling

Cyclic Machine Scheduling with Tool Transportation

Center for Production, Logistics and Operations Management
P.O. Box 217, 7500 AE Enschede, the Netherlands

Publisher: Twente University Press, P.O. Box 217, 7500 AE Enschede, the Netherlands,
www.tup.utwente.nl

Cover design: Jo Molenaar [deel 4] ontwerpers, Enschede
Cover photo: Cindy Kuijpers, Enschede
Print: Grafisch Centrum Twente, Enschede

ISBN 9036515874

CYCLIC MACHINE SCHEDULING WITH TOOL TRANSPORTATION

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op donderdag 14 juni 2001 te 15.00 uur

door

Clasina Maria Hermina Kuijpers
geboren op 6 juli 1968
te Nijmegen

Dit proefschrift is goedgekeurd door de promotor

prof.dr. U. Faigle

en de assistent-promotor

dr.ir. W.M. Nawijn

To the memory of
Henny Kuijpers-Sanders

Preface

Why is there a picture of a train station on the cover of a thesis about cyclic scheduling?

A train station is just one of the many places where cyclic schedules are used: trains arrive and leave according to a timetable that is repeated every hour, train drivers and conductors may be allocated to trains according to a schedule that is repeated every week, etc.

Why is there a picture of a train station on the cover of a thesis about cyclic machine scheduling?

At first glance, a train station does not seem to have any relation with machine scheduling. However, scientists look at problems in such an abstract way that for them a scheduling problem at a train station and a machine scheduling problem can be equivalent.

Why is there a picture of the train station of Enschede on the cover of this thesis?

This thesis constitutes an overview of one of my main activities during the years that I have spent in Enschede.

Acknowledgments

My debts to all people that have contributed in one way or another to the realization of this thesis.

First of all, I want to thank my supervisor prof. Ulrich Faigle for having given me the opportunity to become a Ph.D. student.

For advice and encouragement I am very indebted to Wim Nawijn and Johann Hurink. Some years ago Wim came up with the problem that is the topic of this thesis. At that time we both expected that the problem for two machines could be solved in a few weeks. Who would have thought that I would write almost a complete thesis about it! In particular, I want to thank Wim for looking with me at the details of the proof of Theorem 5.1. I am especially thankful to Johann for his enthusiasm, proofreading the drafts of this thesis and of course for the two times daily knocks on my window indicating “coffee time”.

A word of thanks also goes to my colleagues of the department Discrete Mathematics, Operations Research and Stochastics, and in particular to my (former) office-mates Marcel Hunting, Daniël Paulusma and Petrica Pop, who all offered me a pleasant working environment. Especially in the last months Daniël and I spent many hours in office TWRC B131B, most of the time working seriously on our theses, but also discussing many other topics, especially music and movies (for which we did not have enough time).

I want to express my gratitude to prof. Peter Brucker, prof. Gerard Gaalman, prof. Aart van Harten, prof. Steef van de Velde and prof. Henk Zijm for participating in my graduation committee.

Moreover, thanks are due to Manita Lemmens, for her helpful comments, and to Ramon Clout and Wim Verhaegh for passing me their LaTeX style files.

Finally, very special thanks go to my family and friends. Although they did not see me very often in the last years, I could not have finished this thesis without their support.

Enschede, May 2001 Cindy Kuijpers

Contents

1

Introduction

1.1 Combinatorial optimization

In everyday life, one encounters many situations in which a decision has to be taken. Frequently, in these situations there are so many options to choose from that it is difficult to see what is the best to do. In mathematics, these kind of decision problems are often modeled as *combinatorial optimization problems.* A combinatorial optimization problem concerns selecting, from among a finite set of possible solutions, that solution that maximizes or minimizes a certain function, the so-called *objective function*. Commonly, the set of possible solutions is described by some inequality and equality constraints, and possibly integrality constraints on some of the variables.

If the objective function and the equality and inequality constraints are all linear, then such a problem can be written as

$$\min\{cx+hy : Ax+Gy \leq b, x \in \mathbb{Z}^n_+, y \in \mathbb{R}^p_+\}, \tag{1.1}$$

where $\mathbb{Z}^n_+$ is the set of nonnegative integral n-dimensional vectors, $\mathbb{R}^p_+$ is the set of nonnegative real p-dimensional vectors, A is an $m \times n$ matrix, G is an $m \times p$ matrix and b an m-vector. We call such a problem a *linear mixed-integer program* (MIP), where the word "mixed" refers to the presence of both integer and continuous variables.

A point that satisfies all constraints is called a *feasible solution*. A feasible solution (x^*, y^*) is an *optimal solution* if $cx^* + hy^* \leq cx + hy$ for all feasible solutions (x, y). A set of data that specifies the set of feasible solutions and the objective function is called a *problem instance*.

If in (1.1) $x \in \mathbb{B}^n$, where $\mathbb{B}^n$ is the set of n-dimensional binary variables, we speak about a *0-1 MIP* . In the special case in which there are no integral variables we have a *linear programming problem* (LP) :

$$\min\{hy : Gy \leq b, y \in \mathbb{R}^p_+\}.$$

The problem

$$\min\{cx + hy : Ax + Gy \leq b, x \in \mathbb{R}^n_+, y \in \mathbb{R}^p_+\},$$

is called the *LP-relaxation* of (1.1) and

$$\min\{cx + hy : Ax + Gy \leq b, x \in \mathbb{R}^n_+, 0 \leq x_1, x_2, \ldots, x_n \leq 1, y \in \mathbb{R}^p_+\}$$

is the LP-relaxation of the 0-1 MIP. Notice that these relaxations are linear programming problems.

For more information about discrete optimization and linear mixed-integer programs see, e.g., Papadimitriou & Steiglitz [1982], Schrijver [1998] and Nemhauser & Wolsey [1999].

A wide variety of problems can be represented by discrete optimization programs. An important area of application concerns the management and efficient use of scarce resources. This area includes a.o. distribution of goods, VLSI design and *scheduling*.

1.2 Scheduling

One intensively studied class of discrete optimization problems consists of the scheduling problems. Scheduling was defined by Baker [1974] as the allocation of resources over time to perform a collection of tasks. This sounds very abstract, but scheduling problems are part of daily life. Every day again one tries to use his scarce time as useful as possible. Further examples of scheduling problems are:

1. how do flight companies have to distribute pilots and cabin crew over their airplanes?
2. in a sports competition: which teams are going to play against each other and at what time?
3. at a school: which groups of students have which classes at what time?

Although scheduling problems may concern many different types of resources, many of them can be modeled as scheduling *jobs* on *machines*. Therefore, the theory concerning these problems, is referred to as *machine scheduling theory*. For

an extensive introduction into the theory of scheduling, see, e.g., Baker [1974], Coffman [1976], French [1982], Lawler, Lenstra, Rinnooy Kan & Shmoys [1993] Chrétienne, Coffman, Lenstra & Liu [1995] Pinedo [1995] Blazewicz, Ecker, Pesch, Schmidt & Weglarz [1996] and Brucker [1998].

In this thesis, we consider a *cyclic scheduling problem.* It is very difficult to give a clear definition of what a cyclic scheduling problem is, since in the literature, sometimes very different scheduling problems are called cyclic. In general, one can say that a scheduling problem is cyclic if it displays some kind of repetitive behavior. That it why cyclic scheduling problems are also often referred to as *periodic scheduling problems*.

Frequently, problems of the following form are studied (see, e.g., Liu & Layland [1973], Labetoulle [1974], Leung & Merrill [1980], Lawler & Martel [1981], Bertossi & Bonuccelli [1985]). Given is a set of jobs J, where each job $j \in J$ is characterized by a quadruple (r_j, p_j, d_j, π_j). Job j initially makes a request for processing at time r_j, and thereafter at the times $r_j + k\pi_j (k \in \mathbb{N})$. Each processing of task j takes p_j time units, and the deadline for this processing occurs d_j time units after its request. A schedule is called feasible if no deadline is ever missed. The objective is to find a feasible schedule on either a given, or a minimal, number of machines. In some studies, the values of $r_j (j \in J)$ are given; in others, they are part of the decision process.

Other cyclic scheduling problems concern problems of the following type. Given a set of machines, a set of jobs J, the processing times of the jobs and some constraints (e.g., on the order in which the jobs can be processed by the machines.) The target is to find a schedule that can be repeated every certain period of time and with which the processed number of jobs per unit of time is maximized. See for problems of this type, e.g., Roundy [1992], Papadimitriou, Serafini & Yannakakis [1993], Mahadev, Solot & De Werra [1993], McCormick & Rao [1994], Hanen & Munier [1995] and Agnetis, Pacciarelli & Rossi [1997].

More types of cyclic scheduling problems exist. Wegner [1997] attempts to make a classification of all the different types of problems that appear in the literature. Overviews of Korst [1992] and Hanen & Munier [1995] are more thorough, but concern more restricted classes of cyclic scheduling problems.

1.3 Branch-and-bound

One of the methods used for solving linear mixed-integer problems is *branch-and-bound*. Branch-and-bound is an *exact method*, i.e., it always results in an optimal solution. The disadvantage of the branch-and-bound method, however, is that it may be extremely slow.

In a branch-and-bound method for a linear mixed-integer program, a sequence of LP-problems is solved. While doing this, a tree of subproblems is built; each subproblem constitutes a node of the tree. In the root node, the LP-relaxation of the original MIP is solved. If the solution to the relaxation has fractional-valued integer variables, a fractional variable is chosen for *branching*, and two new subproblems are generated, each with more restrictive bounds for the branching variable. For a 0-1 MIP, mostly, one node has the variable fixed at zero and the other node has it fixed at one. The newly created subproblems can result in an all-integer solution, an infeasible solution or another fractional solution. If the solution is fractional, the process is repeated. If the subproblem is infeasible or if the objective function value of the corresponding LP-relaxation is worse than a certain value, the *cut-off value*, the node is considered to be of no interest and the tree is pruned at the node. At the start of the algorithm, the cut-off value is infinity. Every time an integer solution is found that is better than the cut-off value, the cut-off value is set to the objective function value that corresponds to this solution and all the nodes with objective functions that are not better that the new cut-off value are pruned from the tree.

Notice that if before the start of the algorithm an upper bound on the objective function value is known, then the algorithm can be speeded up by choosing the cut-off value at the start of the algorithm equal to the value of this upper bound.

For a more detailed description of the branch-and-bound method, see, e.g., Papadimitriou & Steiglitz [1982] or Nemhauser & Wolsey [1999].

1.4 Complexity

Some optimization problems are more difficult than others. Today there is a general agreement that a computational problem is considered to be ”easy” if there exists an algorithm for it, whose complexity grows polynomially with respect to the size of the input. In this case the algorithm is called a *polynomial time algorithm*.

The problems for which polynomial time algorithms exist constitute the class $\mathcal{P}$. It is well-known that *linear programming* is in $\mathcal{P}$. The class $\mathcal{P}$ is a subclass of the class $\mathcal{NP}$. A problem is in $\mathcal{NP}$ if the feasibility of any given solution of the problem can be verified in polynomial time. The most difficult problems of $\mathcal{NP}$ constitute the class of $\mathcal{NPC}$. Problems in $\mathcal{NPC}$ are called $\mathcal{NP}$*-complete*. $\mathcal{NP}$-complete problems have the following properties:

1. No $\mathcal{NP}$-complete problem can be solved by any known polynomial algorithm.
2. If there exists a polynomial algorithm for any $\mathcal{NP}$-complete problem, then there are polynomial algorithms for all $\mathcal{NP}$-complete problems.

To prove that a problem P is $\mathcal{NP}$-complete, one generally shows that problem P is in $\mathcal{NP}$, and that there exists a known $\mathcal{NP}$-complete problem that can be transformed in polynomial time to P.

It is interesting to know something about the complexity of a problem, because this gives an indication about what kind of methods can be used (best) for solving the problem. For a problem in $\mathcal{P}$, one will always try to find a low-order polynomial time algorithm for solving it. However, once a problem is known to be $\mathcal{NP}$-complete, one generally settles for less ambitious goals. This is, because it is widespread believed that for none of the $\mathcal{NP}$-complete problems a polynomial time algorithm will exist.

For more information about complexity theory, we refer to Papadimitriou & Steiglitz [1982],Garey & Johnson [1984] and Nemhauser & Wolsey [1999].

1.5 Problem description

We consider a production environment in which several identical machines all individually produce units of a certain product. During the production process of one unit of the product, a machine carries out a fixed sequence of operations, or *tasks*. For any of these tasks, the machine uses a task-specific *tool*. Let us, in the first place, assume that all the tools belonging to all the different tasks are available on each machine.

Suppose that the objective of the management of the factory is to maximize the average production per unit of time, the *throughput*. The question now is how to make optimal use of the machines.

Let us have a look at a small example. Suppose that the plant that we consider contains two machines, and that the manufacturing process of one unit of product consists of 3 tasks, which respectively take 2,1 and 3 time units. Then the production is maximized if the machines are used as shown in Figure 1.1. The rectangles

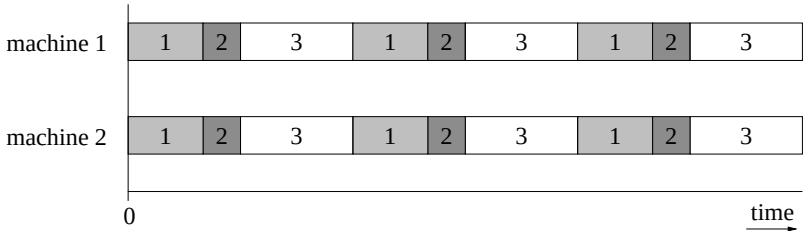


Figure 1.1. An optimal schedule if all tools are present at both machines.

in the figure indicate the processings of the tasks. We see that if all the different tools are present on all of the machines, then all the machines can be kept busy all the time.

However, tools may be very expensive. Therefore, machines frequently share tools. This makes the situation more complicated. We assume that for any task there is just one tool available. Thus, the machines can never carry out the same type of task simultaneously (see Figure 1.2).

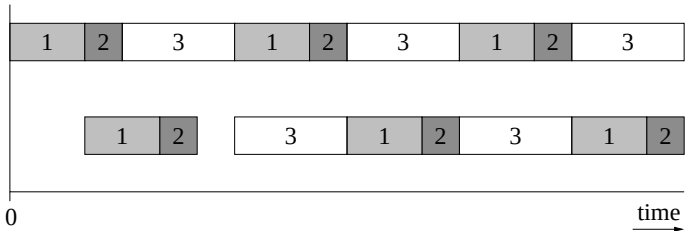


Figure 1.2. An optimal schedule if the machines share the tools.

In Figure 1.2, we see that at time zero machine 2 cannot start the execution of task 1, it has to wait until machine 1 has finished processing task 1. If a machine cannot continue with its next planned task, because the corresponding tool is still not present, we say that the machine is *idle*. We see that in Figure 1.2, machine 2 is idle twice, namely in the time intervals $[0,2]$ and $[5,6]$.

In the above example, it is still not taken into account that the tools have to be moved between the machines in some way. We consider a production system, where this is done by a robot. We assume that the robot can transport only one tool at a time. Obviously, every time the robot moves from one machine to another, this takes some positive transportation time. Assume that for the example of Figure 1.2, it takes one unit of time to move the robot from machine 1 to machine 2 or vice versa. Then, an optimal schedule is given in Figure 1.3.

The diagonal line segments in the figure represent the tool transports. The dashed line segment indicates a transport in which no tool is involved. We call such a transport an *empty transport*.

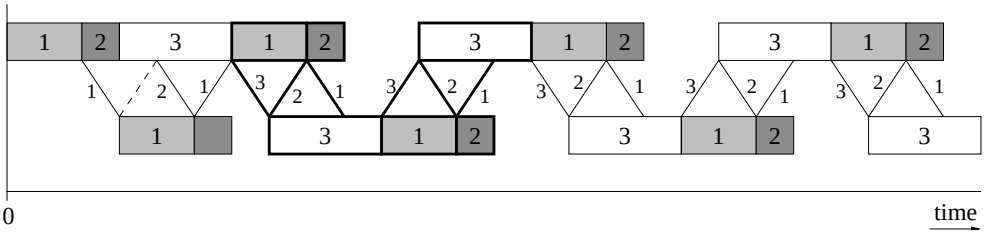


Figure 1.3. An optimal schedule if the tools are shared, with positive tool transportation times.

We see in Figure 1.3 that after some *start-up time* the solution gets a repetitive behavior. Moreover, we see that in the part of the solution that is repeated – the

bold part –, the job is executed exactly once on machine 1 and once on machine 2. In this thesis, we focus on schedules with such a specific repetitive behavior. The part of the schedule that is repeated and in which the job is processed once on each machine, we call a *cycle* and the time it takes to carry out this cycle, the *cycle time*.

Suppose that for the schedule in Figure 1.3 we ignore the start-up time. Then we can also represent the schedule by Figure 1.4. In this figure we see two circular

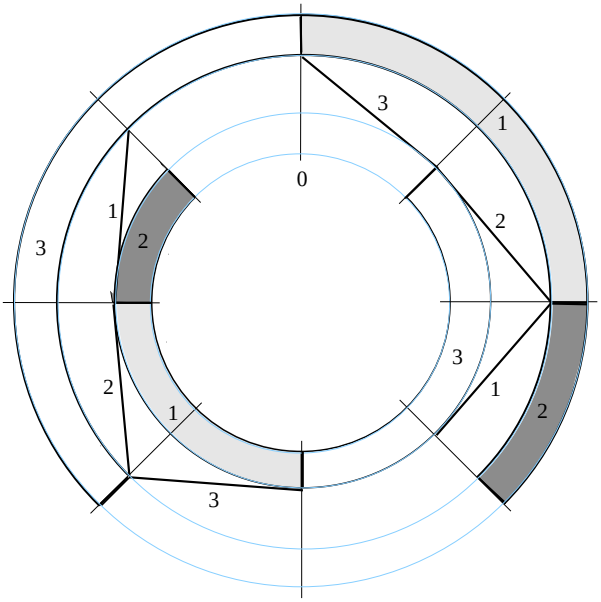


Figure 1.4. A graphical representation of a cyclic schedule.

strips, the outer one representing machine 1, the inner one representing machine 2. The line segments in the space between the two strips indicate the tool transports.

The problem for which a schedule is shown in Figure 1.3, we call the *linear version* of our problem; the problem belonging to the schedule in Figure 1.4 the *cyclic version* of our problem.

For convenience, however, we prefer to represent the schedule of Figure 1.4 in a similar way as for the linear version (see Figure 1.5). Notice that, although it seems that Figure 1.5 represents a linear time interval, it represents a cyclic schedule.

1.6 Background

Until recently, tool handling was not considered to be a critical issue in production planning. In the last years, however, the growing use of *C(omputer)N(umerical)C(ontrol) machines* in mechanical discrete part production has changed this. CNC is an advanced machine control system in which numeri-

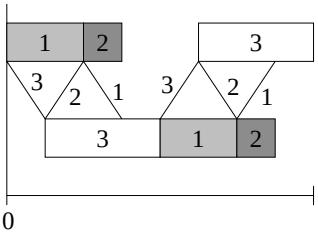


Figure 1.5. Another graphical representation of the schedule of Figure 1.4.

cal data is used for controlling machine tool motions. CNC machines are usually grouped into *flexible manufacturing systems (FMSs)*.

The versatile CNC machines may make use of extensive sets of, often very expensive, tools and fixtures. In order to decrease the costs, within an FMS, the machines often share the same tools. Tool handling is done by an automated device. The performance of an FMS highly depends on the interaction between the machines and this device.

That tool handling is important, is illustrated by the following numbers [Gaalman & Nawijn, 1996]. Viehweger [1988] indicates that in a single CNC machine center tools account for 29% and fixtures for 28% of the total invested capital. In flexible manufacturing systems tools and fixtures may reach up to 25% of the total FMS investment [Tomek, 1986], where tooling is, with 25-30% , the major component of the variable costs [Kuchinic & Seidmann, 1988].

Inefficient tool handling may cause a serious decrease in productivity. Mason [1986] and Hammer [1988] indicate that a foreman spends 40-80% of his time looking for and expediting materials and tools and that 16% of the scheduled production is not met because of the absence of tools.

A comprehensive survey of tool management issues is given by Veeramani, Upton & Barash [1992] and Gray, Seidmann & Stecke [1993].

1.7 Related research

Tool management in scheduling

Over the last years, a considerable amount of articles about tool management in scheduling has been published. The articles by Gray, Seidmann & Stecke [1993] and Veeramani, Upton & Barash [1992] contain comprehensive surveys of the literature on this topic. Also in Crama [1997] an overview of results in this field is given.

In these articles, it can be seen that most research in tool management concerns problems related with the limited capacity of the tool magazine of the machine(s) (see, a.o., Tang & Denardo [1988], Bard [1988] and De Werra & Widmer [1990].

As soon as a machine needs a tool that is not present in its magazine, the tool set in the magazine has to be changed. The objective is to minimize the number of these *tool switches*.

Scheduling problems with positive transportation times

In most of the research in which positive transportation times are taken into account, the transports do not concern tools but jobs.

A frequently studied problem in this context, concerns the scheduling of identical jobs (or *parts*) in a *robotic cell* (see, a.o., Sethi, Sriskandarajah, Sorger, Blazewicz & Kubiak [1992], Hall, Kamoun & Sriskandarajah [1997], Crama & Van de Klundert [1997] and Brauner & Finke [1998]). A robotic cell is a flow-shop (i.e., all jobs are first processed on machine 1, then on machine 2, etc.) in which there are some machines, an input device, an output device and a robot for transporting the parts between the mutual machines and between the machines and the devices. In the most general setting, a so-called, *Minimal Part Set (MPS)* is to be repeatedly produced, where the MPS consists of parts of different types in proportion to a certain target production mix. The objective is to determine the part input sequence and the corresponding sequence of robot moves that maximizes the throughput rate on the long term.

Also in Strusevich [1999] jobs are transported. Strusevich considers jobs that consist of a number of operations, each of which has to be processed on a specific machine. A job visits the machines in a certain order, that is not known in advance. The objective is to minimize the *makespan*, i.e., the time necessary for the processing of a certain number of jobs.

Hurink & Knust [2000] give complexity results for flow-shop problems in which a single robot moves the jobs with positive transportation times between the machines and in which the target is to minimize the finishing time of the last job.

The linear problem of Section 1.5

Knoop Pathuis [2000] considers the linear version of the problem described in Section 1.5 for two machines and for a finite set of jobs. (This problem will be modeled for an arbitrarily number of machines in Section 2.2.) Knoop Pathuis does not pronounce upon the complexity of the problem but gives a fast branch-and-bound algorithm for solving the problem. His algorithm is based on the observation that the only relevant decision moments are the moments at which the robot arrives at a machine where at that moment there is no tool available for picking up. At such a moment only two relevant decisions can be made: either the robot moves immediately empty to the other machine, or it waits at the machine until a tool becomes available.

Gaalman & Nawijn [1996] consider the linear version of the *m*-machine problem in which they allow more than one tool of the same type. They examine the consequences of three different tool transport policies with respect to the transport time and the transport frequency of the robot.

1.8 Outline of this thesis

In the next chapter, we describe the problem of Section 1.5 in more detail. After this, first the linear version of the problem is modeled as a linear mixed-integer program, after which this program is adapted for the cyclic version.

The remaining chapters are dedicated to the cyclic version of the problem for two machines. In Chapter 3 upper and lower bounds on the optimal solution are given and situations are considered in which the lower bounds can actually attained. Moreover, the two machine problem is solved exactly for the cases where the job contains at most three tasks. Furthermore, an upper bound on the total number of relevant schedules is given. Finally, the consequences of some model assumptions are discussed.

In Chapter 4 we introduce the concept of a *critical cycle*. We show that schedules can be characterized by such critical cycles and we look at some properties of these critical cycles.

Different types of critical cycles are described in Chapter 5. It turns out that only critical cycles of five different structures need to be considered.

Schedules with critical cycles of these five structures are examined in more detail in Chapter 6. We will see that for some of these structures optimal schedules can be obtained with a polynomial time algorithm.

In Chapter 7 three algorithms for solving the problem are given, that only take into account schedules with critical cycles of the five relevant structures. Moreover, experimental results are described.

Finally, in Chapter 8 some conclusions are given and areas of further research are mentioned.

2

Notation and modeling

2.1 Introduction and notation

In this chapter, the problem that was sketched in broad outlines in Section 1.5 is formalized and modeled as a linear mixed-integer program.

We consider a production environment with M machines. These machines are numbered and placed in line, such that the distance between any pair m_1, m_2 of machines is $|m_1 - m_2|d$, where d is a constant $(1 \leq m_1, m_2 \leq M)$. Each of the machines continuously repeats the processing of a job J. Job J consists of N different tasks, which have to be processed without preemption and in a fixed order. Let task j be the jth task of job J, the processing time of which is $p_j (p_j > 0, j = 1, 2, \ldots, N)$.

The machines can only execute one task at a time. Moreover, for the execution of a task on a machine, a task-specific tool is required at the machine. For any task there is just one such a tool available. Hence, the machines have to share the tools and consequently, the tools need to be transported between the machines. This is done by a robot. This robot can transport only one tool at a time. Moreover, every time the robot moves from one machine to another, this takes a positive transport time that only depends on the distance between the machines. Without loss of generality, we assume that the robot covers one unit of distance per unit of time. In case a tool does not arrive in time at a machine, that machine cannot continue with the next planned task and it becomes idle.

The objective is to find a *schedule*, i.e., a list of times at which the executions of the tasks and the transports of the tools occur. Notice that, since the processing times of tasks and the transportation times are known, only the start or end times of the processings and the transports are relevant. We choose to define the following decision variables:

$S_j^m[k] :=$ the time at which the kth processing of task j starts on machine m $(1 \leq j \leq N, 1 \leq m \leq M, k \geq 1)$ and

$T_j^m[k] :=$ the time at which tool j arrives at machine m for the kth processing of task j $(1 \leq j \leq N, 1 \leq m \leq M, k \geq 1)$.

For convenience, we define task $j_{\max}$ to be (one of) the task(s) with the largest processing time and we indicate the processing time of $j_{\max}$ by $p_{\max}$. Moreover, we denote a processing at machine m that starts at time r_1 by $(r_1^m,)_p$ and a transport to machine m that finishes at time r_2 by $(, r_2^m)_t$ $(m = 1,2)$. The kth processing of task j on machine m, we denote by $(S_j^m[k],)_p$ and the kth transport of tool j to machine m by $(, T_j^m[k])_t$ $(1 \leq j \leq N, 1 \leq m \leq M, k \geq 1)$.

Furthermore, we introduce some kind of cyclic addition and subtraction. If i_1 and i_2 are two elements of a the finite set $\{1, 2, \ldots, n\}$ where $n \in \mathbb{N}$, then

$$i_1 \oplus_n i_2 = \begin{cases} i_1 + i_2 & \text{if } i_1 + i_2 \leq n, \\ i_1 + i_2 - n & \text{otherwise} \end{cases}$$

and

$$i_1 \ominus_n i_2 = \begin{cases} i_1 - i_2 & \text{if } i_1 - i_2 \geq 1, \\ i_1 - i_2 + n & \text{otherwise.} \end{cases}$$

Lastly, let $\delta_{i_1 i_2}$ be the Kronecker-delta:

$$\delta_{i_1 i_2} = \begin{cases} 1 & \text{if } i_1 = i_2, \\ 0 & \text{otherwise.} \end{cases}$$

2.2 Model assumptions

Before we can present a modeling of our problem as a linear mixed-integer program, we need to make some model assumptions.

Assumption 2.1. *All machines are used.* □

This assumption may seem redundant, but problem instances exist for which the best result is obtained by using less than M machines. For example, if $N = 1$ or if the processing times are extremely small with respect to the transport time it is the best to use only one machine (see also Section 3.6).

Moreover, in order to balance the load on the machines, we make the following assumption.

Assumption 2.2. *The kth processing of task j on machine m occurs before the kth processing of task j on machine $m+1$ and the kth processing of task j on machine M occurs before the $k+1$th processing of task j on machine 1* ($1 \leq j \leq N$, $1 \leq m \leq M-1$, $k \geq 1$). □

Because of this assumption, a schedule for the linear problem for $N = 1$ and $M = 3$ looks like Figure 2.1.

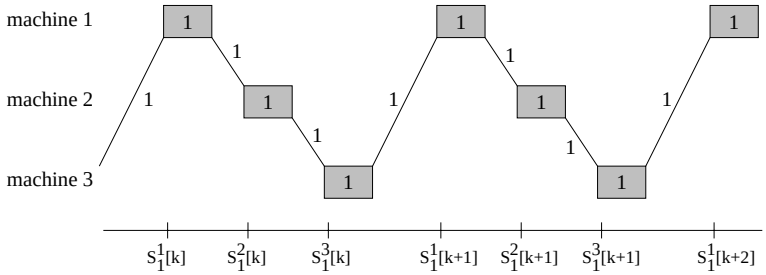


Figure 2.1. A linear schedule for which Assumption 2.2 holds ($N = 1, M = 3$).

Notice that, since no profit can be obtained by transporting tools to machines where they are not necessary, the above assumption implies that the tools are routed cyclicly on the machines: a tool is transported to, respectively, machine 1, machine 2, ..., machine M, machine 1, machine 2, ...

Assumption 2.3. *Transports and processings are not postponed, if there is no reason for it.* □

This means that a processing of a task starts as soon as the corresponding machine and the required tool are both available. A *nonempty transport* begins as soon as the robot and the tool to be transported are available for the transport. Finally, an empty transport starts immediately when the robot becomes available for that transport.

Assumption 2.4. *A robot movement is always useful; i.e., a transport from machine m_1 to machine m_2 is meant to transport a tool to machine m_2 or to pick up a tool at machine m_2.* □

Notice that this implies that the robot will never carry out two subsequent empty transports.

Moreover, for having a fixed reference time we assume the following.

Assumption 2.5. $T_1^1[1] = 0$. □

2.3 The linear problem

Using the problem description and the five assumptions given in the previous section, we now present a mathematical formulation for the linear version of the considered problem. Based on this formulation, in the next section we will derive a linear mixed-integer problem for the cyclic version (which is the main focus of this thesis).

First, we investigate which lists of start times of processings and tool arrival times constitute feasible schedules for our problem.

Lemma 2.1. *For each machine $m, 1 \leq m \leq M$, and for any $k \geq 1$, we may assume that a feasible schedule satisfies the condition*

$$T_1^m[k] < T_2^m[k] < \ldots < T_N^m[k] < T_1^m[k+1].$$

Proof. Consider a feasible schedule for which the above condition does not hold. Let m be the machine with the lowest index for which the condition does not hold and, let (j_1, j_2) be the pair of tasks with the lowest index j_1 such that $T_{j_1}^m[k] > T_{j_2}^m[k]$ for some $j_2 > j_1$.
Suppose first that $2 \leq m \leq M$. Notice that at time $T_{j_2}^m[k] - d$ the kth processing of the tasks $j_1, j_1+1, \ldots, j_2$ on machine $m-1$ must already be finished. Moreover, the kth processing of task j_1 on machine m can only take place after time $T_{j_1}^m[k]$. Suppose that the transports of tool j_1 and tool j_2 are interchanged, such that the new arrival moments $\bar{T}_{j_1}^m[k]$ and $\bar{T}_{j_2}^m[k]$ become $\bar{T}_{j_1}^m[k] = T_{j_2}^m[k]$ and $\bar{T}_{j_2}^m[k] = T_{j_1}^m[k]$. In view of the two observations it is clear that the resulting schedule is still feasible. For $m = 1$, the same reasoning, with some small adaptations, can be followed.
The above process of interchanging transports can be repeated until $T_1^m[k] < T_2^m[k] < \ldots < T_N^m[k]$ for all m. By a similar argument we may assume that $T_1^m[k+1] > T_N^m[k], 1 \leq m \leq M$. □

With the above lemma and the Assumptions 2.1-2.5, we can describe the properties of feasible schedules as follows:

1. Since a machine can handle only one task at a time, and since the tasks of the job have to be scheduled in order $1, 2, \ldots, N, 1, 2, \ldots$, we get:
$$\begin{aligned} &S_j^m[k] \geq S_{j-1}^m[k] + p_{j-1} \quad 2 \leq j \leq N, 1 \leq m \leq M, k \geq 1, \\ &S_1^m[k+1] \geq S_N^m[k] + p_N \quad 1 \leq m \leq M, k \geq 1. \end{aligned} \tag{2.1}$$
2. A processing of a task on a machine can only be started if its corresponding tool is available at that machine. Hence,
$$T_j^m[k] \leq S_j^m[k] \qquad 1 \leq j \leq N, 1 \leq m \leq M, k \geq 1. \tag{2.2}$$
3. A tool j cannot arrive at machine m before it has become available on the previous machine –which is machine $m \ominus_M 1$ – and it has been transported

to machine m. Hence,

$$\begin{array}{ll} T_j^m[k] \geq S_j^{m-1}[k] + p_j + d & 1 \leq j \leq N, 2 \leq m \leq M, k \geq 1, \\ T_j^1[k+1] \geq S_j^M[k] + p_j + (M-1)d & 1 \leq j \leq N, k \geq 1. \end{array} \tag{2.3}$$

4. The robot transports only one tool at a time. Moreover, as long as the robot is carrying out an empty transport, no tool can be picked up at a machine. Therefore, the intervals in which two subsequent – empty or nonempty – transports occur may not overlap. Observe that this implies that the time intervals in which two consecutive *non*empty transports take place may not overlap and, in case there exists an empty transport between them, the gap between the intervals must be large enough to carry out the empty transport. The above observation has to be translated into restrictions on our decision variables. Consider two consecutive nonempty transports.

   Let us first suppose that the transports both concern the same tool j $(1 \leq j \leq N)$. Because tool j is routed cyclicly on the machines, the two transports of tool j have to be of the form $(,T_j^M[k])_t$ and $(,T_j^1[k+1])_t$ if $m = 1$, and of the form $(,T_j^{m-1}[k])_t$ and $(,T_j^m[k])_t$ otherwise. Notice that no conflicting pair of consecutive transports of tool j exists if:

$$T_j^m[k_2] - T_j^{m \ominus_M 1}[k_1] \geq \begin{cases} d & \text{for } 2 \leq m \leq M, k_2 = k_1, k_1 \geq 1, \\ (M-1)d & \text{for } m = 1, k_2 = k_1 + 1, k_1 \geq 1. \end{cases}$$

   The above restrictions, however, are redundant. They can easily be derived from the Formulas (2.2) and (2.3) and the fact that $p_j > 0$.

   Next, suppose that the two consecutive nonempty transports are both to the same machine m. Then, there exists an empty transport in between them. Therefore, and because of Lemma 2.1, no conflicting pair of subsequent transports to machine m exists if:

$$T_j^m[k_2] - T_{j \ominus_N 1}^m[k_1] \geq \begin{cases} 2d & \text{for } 2 \leq j \leq N, 2 \leq m \leq M, k_1 \geq 1, k_2 = k_1, \\ 2d & \text{for } j = 1, 2 \leq m \leq M, k_1 \geq 1, k_2 = k_1 + 1, \\ 2(M-1)d & \text{for } 2 \leq j \leq N, m = 1, k_1 \geq 1, k_2 = k_1, \\ 2(M-1)d & \text{for } j = 1, m = 1, k_1 \geq 1, k_2 = k_1 + 1. \end{cases} \tag{2.4}$$

   Finally, suppose that we have two consecutive nonempty transports that concern different machines and different tools: the k_1th transport of tool j_1 to machine m_1 and the k_2th transport of tool j_2 to machine m_2, where $m_1 \neq m_2$ and $j_1 \neq j_2$. Without loss of generality, we assume that $m_1 < m_2$. We derive the required conditions to assure that there are no conflicts between the two consecutive nonempty transports by considering the cases $j_1 < j_2$ and $j_1 > j_2$ separately.

If $j_1 < j_2$, then by Lemma 2.1 and because the tools are routed cyclicly on the machines, we know that

$$T_{j_1}^{m_1}[k] < T_{j_1}^{m_2}[k] < T_{j_2}^{m_2}[k] < T_{j_1}^{m_2}[k+1] < T_{j_1}^{m_1}[k+2].$$

Hence, since $(,T_{j_1}^{m_1}[k])_t$ and $(,T_{j_2}^{m_2}[k])_t$ are different transports either $T_{j_1}^{m_1}[k] < T_{j_2}^{m_2}[k] < T_{j_1}^{m_1}[k+1]$ or $T_{j_1}^{m_1}[k+1] < T_{j_2}^{m_2}[k] < T_{j_1}^{m_1}[k+2]$.
Suppose first that $2 \leq m_1 \leq M-1$. If $T_{j_1}^{m_1}[k] < T_{j_2}^{m_2}[k] < T_{j_1}^{m_1}[k+1]$, then between $T_{j_2}^{m_2}[k]$ and $T_{j_1}^{m_1}[k+1]$ the robot moves at least from machine m_2 to $m_1 - 1$ and from $m_1 - 1$ to m_1. This implies that there are no conflicts between these transports if

$$T_{j_1}^{m_1}[k+1] - T_{j_2}^{m_2}[k] \geq (m_2 - m_1 + 2)d.$$

Between $T_{j_1}^{m_1}[k]$ and $T_{j_2}^{m_2}[k]$ the robot moves at least over a distance $(m_2 - m_1)d$. Hence, these transports do not overlap if

$$T_{j_2}^{m_2}[k] - T_{j_1}^{m_1}[k] \quad \geq \quad (m_2 - m_1)d.$$

If $T_{j_1}^{m_1}[k+1] < T_{j_2}^{m_2}[k] < T_{j_1}^{m_1}[k+2]$ it can be seen in the same way, that there do not exist any conflicts between the transports if

$$\begin{array}{rcl} T_{j_1}^{m_1}[k+2] - T_{j_2}^{m_2}[k] & \geq & (m_2 - m_1 + 2)d, \\ T_{j_2}^{m_2}[k] - T_{j_1}^{m_1}[k+1] & \geq & (m_2 - m_1)d. \end{array}$$

Hence, we see that in a feasible schedule it is necessary that either

$$\begin{array}{rcl} T_{j_1}^{m_1}[k] + (m_2 - m_1)d & \leq & T_{j_2}^{m_2}[k] \leq T_{j_1}^{m_1}[k+1] - (m_2 - m_1 + 2)d \\ T_{j_1}^{m_1}[k+1] + (m_2 - m_1)d & \leq & T_{j_2}^{m_2}[k] \leq T_{j_1}^{m_1}[k+2] - (m_2 - m_1 + 2)d \end{array} \quad (2.5)$$

for $1 \leq j_1 < j_2 \leq N, 2 \leq m_1 < m_2 \leq M, k \geq 1$.
Now let us consider the case $m_1 = 1$. Notice that between a transport to machine m_2 and machine 1, the robot moves from machine m_2 to machine M and from machine M to machine 1. Hence, we find that

$$\begin{array}{rcl} T_{j_1}^{1}[k] + (m_2 - 1)d & \leq & T_{j_2}^{m_2}[k] \leq T_{j_1}^{1}[k+1] - (2M - m_2 - 1)d \\ T_{j_1}^{1}[k+1] + (m_2 - 1)d & \leq & T_{j_2}^{m_2}[k] \leq T_{j_1}^{1}[k+2] - (2M - m_2 - 1)d \end{array} \quad (2.6)$$

for $1 \leq j_1 < j_2 \leq N, 2 \leq m_2 \leq M, k \geq 1$.

If $j_1 > j_2$, then in view of Lemma 2.1 and because the tools are routed cyclicly on the machines, we have that

$$T_{j_1}^{m_1}[k-1] < T_{j_1}^{m_2}[k-1] < T_{j_2}^{m_2}[k] < T_{j_1}^{m_2}[k] < T_{j_1}^{m_1}[k+1],$$

from which it follows, in the same way as before, that in a feasible schedule

either

$$\begin{aligned} T_{j_1}^{m_1}[k-1]+(m_2-m_1)d &\leq T_{j_2}^{m_2}[k] \leq T_{j_1}^{m_1}[k]-(m_2-m_1+2)d \\ T_{j_1}^{m_1}[k]+(m_2-m_1)d &\leq T_{j_2}^{m_2}[k] \leq T_{j_1}^{m_1}[k+1]-(m_2-m_1+2)d \end{aligned} \quad (2.7)$$

for $1 \leq j_2 < j_1 \leq N$ and $2 \leq m_1 < m_2 \leq M, k \geq 1$ and

$$\begin{aligned} T_{j_1}^{1}[k-1]+(m_2-1)d &\leq T_{j_2}^{m_2}[k] \leq T_{j_1}^{1}[k]-(2M-m_2-1)d \\ T_{j_1}^{1}[k]+(m_2-1)d &\leq T_{j_2}^{m_2}[k] \leq T_{j_1}^{1}[k+1]-(2M-m_2-1)d \end{aligned} \quad (2.8)$$

for $1 \leq j_2 < j_1 \leq N$ and $2 \leq m_2 \leq M, k \geq 1$.

The formulas (2.1)-(2.8) assure that the tasks of the job are processed in the right order and that no conflicts exist among the processings, among the transports, or among the processings and the transports. In other words, the formulas guarantee the feasibility of a schedule.

Having formulated the feasibility conditions, we still have to define the objective function. Frequently, one tries to minimize the makespan. However, since we are more interested in the long-term behavior of our production system, we are looking for a feasible solution that maximizes the throughput on the long term. Therefore, we formulate the following objective function:

$$\max \lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{M} R_m(t),$$

where $R_m(t) = \max\{r : S_N^m[r] + p_N \leq t\}$ $1 \leq m \leq M, t > 0$ denotes the number of processed jobs on machine m at time t.

The above objective function, however, is still not suitable for us, because it is not expressed in terms of our decision variables. Intuitively, it is clear that the objective is equivalent to the minimization of the average production time per job on the long term. For proving this formally, we need the following lemma.

Lemma 2.2. *If* $\lim_{k\to\infty} \frac{1}{k} S_1^{\bar{m}}[k] = c$ *exists for some* $\bar{m} \in \{1,2,\ldots,M\}$ *then* $\lim_{k\to\infty} \frac{1}{k} S_j^m[k] = \lim_{k\to\infty} \frac{1}{k} T_j^m[k] = c$ *independent of* $j \in \{1,2,\ldots,N\}$ *and* $m \in \{1,2,\ldots,M\}$.

Proof. Let $\lim_{k\to\infty} \frac{1}{k} S_1^{\bar{m}}[k] = c$. From (2.1) we see that for $2 \leq j \leq N$ and $k \geq 2$ we have that

$$\frac{1}{k} S_1^{\bar{m}}[k] \geq \frac{1}{k}(S_j^{\bar{m}}[k-1] + \sum_{i=j}^{N} p_i) \geq \frac{1}{k}(S_1^{\bar{m}}[k-1] + \sum_{i=1}^{N} p_i).$$

Hence, by the squeezing principle it follows that $\lim_{k\to\infty} \frac{1}{k} S_j^{\bar{m}}[k] = c$ for any j, $1 \leq j \leq N$. Moreover, from (2.2) and (2.3) it follows that for all m_1, m_2 for which

$1 \le m_1 \le \bar{m} \le m_2 \le M$, we have that

$$\begin{aligned} S_j^{\bar{m}}[k] &\ge S_j^{m_1}[k]+(\bar{m}-m_1)(p_j+d) \\ &\ge S_j^{m_2}[k-1]+(M-m_2+m_1-1)(p_j+d)+(p_j+(M-1)d)+(\bar{m}-m_1)(p_j+d) \\ &= S_j^{m_2}[k-1]+(M+\bar{m}-m_2)p_j+(2M+\bar{m}-m_2-2)d \\ &\ge S_j^{\bar{m}}[k-1]+(m_2-\bar{m})(p_j+d)+(M+\bar{m}-m_2)p_j+(2M+\bar{m}-m_2-2)d \\ &= S_j^{\bar{m}}[k-1]+Mp_j+2(M-1)d. \end{aligned}$$

Thus,

$$c = \lim_{k\to\infty}\frac{1}{k}S_j^{\bar{m}}[k] \ge \lim_{k\to\infty}\frac{1}{k}S_j^{m_1}[k] \ge \lim_{k\to\infty}\frac{1}{k}S_j^{m_2}[k] \ge \lim_{k\to\infty}\frac{1}{k}S_j^{\bar{m}}[k] = c$$

from which it follows that

$$\lim_{k\to\infty}\frac{1}{k}S_j^{m}[k] = c. \tag{2.9}$$

for any $m \in \{1,2,\dots,M\}$ and any $j \in \{1,2,\dots,N\}$.
Furthermore, by (2.2) and (2.3) we know that

$$\begin{cases} S_j^{m-1}[k]+p_j+d \le T_j^m[k] \le S_j^m[k] & 1 \le j \le N, 2 \le m \le M, \\ S_j^M[k-1]+p_j+(M-1)d \le T_j^1[k] \le S_j^1[k] & 1 \le j \le N. \end{cases}$$

Hence, by the squeezing principle and (2.9) we find that

$$\lim_{k\to\infty}\frac{1}{k}T_j^m[k] = c$$

independent of j and m. □

Lemma 2.3. *If* $c = \lim_{k\to\infty}\frac{1}{k}S_1^1[k]$, *then* c *is positive.*

Proof. The Formulas (2.2) and (2.3) imply that $S_1^1[k+1] \ge S_1^1[1]+kp_1$, where $p_1 > 0$ and $k \ge 1$. From this it follows that $c = \lim_{k\to\infty}\frac{1}{k+1}S_1^1[k+1] \ge p_1 > 0$. □

Better bounds on the value of c will be given in Section 3.2.

Theorem 2.1. *If* $c = \lim_{k\to\infty}\frac{1}{k}S_1^1[k]$, *then* $\lim_{t\to\infty}\frac{1}{t}\sum_{m=1}^{M}R_m(t) = \frac{M}{c}$.

Proof. From the definition of $R_m(t)$, it follows that

$$S_N^m(R_m(t)) \le t < S_N^m(R_m(t)+1),\ 1 \le m \le M.$$

Consequently, for $R_m(t) \ne 0$ we have

$$\frac{S_N^m(R_m(t))}{R_m(t)} \le \frac{t}{R_m(t)} < \frac{R_m(t)+1}{R_m(t)}\cdot\frac{S_N^m(R_m(t)+1)}{R_m(t)+1}.$$

Now, since $p_j > 0$,we have that $R_m(t) \to \infty$ if $t \to \infty$. Hence, by using Lemma 2.2:

$$\lim_{t\to\infty} \frac{t}{R_m(t)} = c$$

from which the assertion easily follows. □

Thus, we see that, since we assume the existence of $\lim_{k\to\infty} \frac{1}{k} S_1^1[k]$ and because M is a constant, we can replace the objective function

$$\max \lim_{t\to\infty} \frac{1}{t} \sum_{m=1}^{M} R_m(t)$$

by

$$\max \lim_{k\to\infty} \frac{1}{k} S_1^1[k].$$

With the Formulas (2.1)-(2.8) and the above objective function the general optimization problem can be formulated (see Figure 2.2). The problem that is described in Figure 2.2 we call P_{lin}^M, indicating that it is the linear version of the problem for M machines.

Knoop Pathuis [2000] studied problem P_{lin}^2. However, his objective was to minimize the makespan.

We remark that a feasible solution of the given problem formulation does not necessarily satisfy Assumption 2.3. However, a solution for which the assumption does not hold can easily be transformed into a solution for which the assumption holds, as can be seen in Figure 2.3. By moving the transports and the processings in Figure 2.3(a) as indicated by the arrows, the schedule of Figure 2.3(b) is created, which satisfies Assumption 2.3. Notice that since the processings and transports are only moved backwards in time, the newly created schedule is not worse than the original schedule.

2.4 The cyclic problem

As already explained in Section 1.5, in this thesis we assume that after some ignorable start-up time the production schedule becomes cyclic in such a way that in one cycle the job is processed exactly once on each of the machines. Let c denote the corresponding cycle time.

Hence, instead of considering P_{lin}^M in its generality, we focus on solutions satisfying the following condition

$$S_j^m[k+1] - S_j^m[k] = T_j^m[k+1] - T_j^m[k] = c \qquad 1 \le j \le N, 1 \le m \le M, k \ge 1, \tag{2.10}$$

$$\text{minimize} \lim_{k\to\infty} \frac{1}{k} S_1^1[k]$$

subject to

$$T_j^m[k] \le S_j^m[k] \qquad 1 \le j \le N, 1 \le m \le M, k \ge 1.$$

$$\begin{aligned} &S_j^m[k] \ge S_{j-1}^m[k] + p_{j-1} && 2 \le j \le N, 1 \le m \le M, k \ge 1, \\ &S_1^m[k+1] \ge S_N^m[k] + p_N && 1 \le m \le M, k \ge 1. \end{aligned}$$

$$\begin{aligned} &T_j^m[k] \ge S_j^{m-1}[k] + p_j + d && 1 \le j \le N, 2 \le m \le M, k \ge 1, \\ &T_j^1[k+1] \ge S_j^M[k] + p_j + (M-1)d && 1 \le j \le N, k \ge 1. \end{aligned}$$

$$\begin{aligned} &T_j^1[k] \ge T_{j-1}^1[k] + 2(M-1)d && 2 \le j \le N, k \ge 1, \\ &T_1^1[k+1] \ge T_N^1[k] + 2(M-1)d && k \ge 1. \end{aligned}$$

$$\begin{aligned} &T_j^m[k] \ge T_{j-1}^m[k] + 2d && 2 \le j \le N, 2 \le m \le M, k \ge 1, \\ &T_1^m[k+1] \ge T_N^m[k] + 2d && 2 \le m \le M, k \ge 1. \end{aligned}$$

$$\left.\begin{aligned} T_{j_2}^{m_2}[k] &\begin{cases} \le & T_{j_1}^{m_1}[k+1] - (m_2 - m_1 + 2)d \\ \ge & T_{j_1}^{m_1}[k] + (m_2 - m_1)d \end{cases} \\ \text{or} \\ T_{j_2}^{m_2}[k] &\begin{cases} \le & T_{j_1}^{m_1}[k+2] - (m_2 - m_1 + 2)d \\ \ge & T_{j_1}^{m_1}[k+1] + (m_2 - m_1)d \end{cases} \end{aligned}\right\} \quad \begin{aligned} &1 \le j_1 < j_2 \le N, \\ &2 \le m_1 < m_2 \le M, \\ &k \ge 1. \end{aligned}$$

$$\left.\begin{aligned} T_{j_2}^{m_2}[k] &\begin{cases} \le & T_{j_1}^{m_1}[k] - (m_2 - m_1 + 2)d \\ \ge & T_{j_1}^{m_1}[k-1] + (m_2 - m_1)d \end{cases} \\ \text{or} \\ T_{j_2}^{m_2}[k] &\begin{cases} \le & T_{j_1}^{m_1}[k+1] - (m_2 - m_1 + 2)d \\ \ge & T_{j_1}^{m_1}[k] + (m_2 - m_1)d \end{cases} \end{aligned}\right\} \quad \begin{aligned} &1 \le j_2 < j_1 \le N, \\ &2 \le m_1 < m_2 \le M, \\ &k \ge 1. \end{aligned}$$

$$\left.\begin{aligned} T_{j_2}^{m}[k] &\begin{cases} \le & T_{j_1}^{1}[k+1] - (2M - m - 1)d \\ \ge & T_{j_1}^{1}[k] + (m-1)d \end{cases} \\ \text{or} \\ T_{j_2}^{m}[k] &\begin{cases} \le & T_{j_1}^{1}[k+2] - (2M - m - 1)d \\ \ge & T_{j_1}^{1}[k+1] + (m-1)d \end{cases} \end{aligned}\right\} \quad \begin{aligned} &1 \le j_1 < j_2 \le N, \\ &2 \le m \le M, \\ &k \ge 1. \end{aligned}$$

$$\left.\begin{aligned} T_{j_2}^{m}[k] &\begin{cases} \le & T_{j_1}^{1}[k] - (2M - m - 1)d \\ \ge & T_{j_1}^{1}[k-1] + (m-1)d \end{cases} \\ \text{or} \\ T_{j_2}^{m}[k] &\begin{cases} \le & T_{j_1}^{1}[k+1] - (2M - m - 1)d \\ \ge & T_{j_1}^{1}[k] + (m-1)d \end{cases} \end{aligned}\right\} \quad \begin{aligned} &1 \le j_2 < j_1 \le N, \\ &2 \le m \le M, \\ &k \ge 1. \end{aligned}$$

$$T_1^1[1] = 0.$$

Figure 2.2. Problem formulation 1

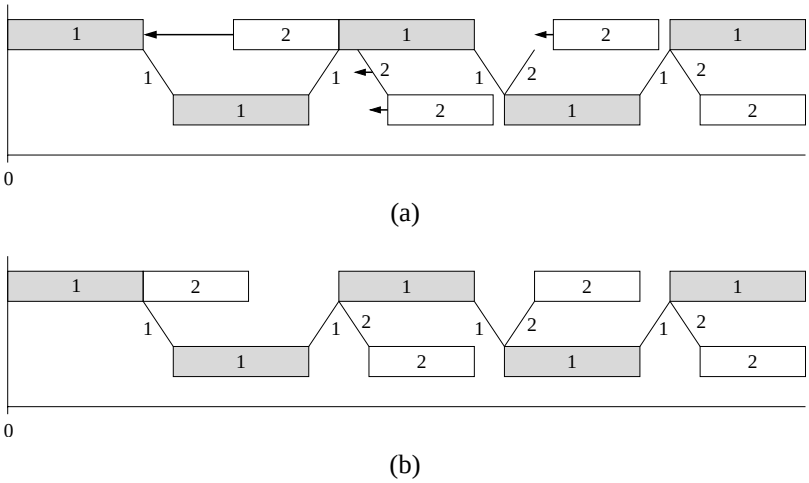


Figure 2.3. (a) Assumption 2.3 does not hold,(b) Assumption 2.3 holds.

The problem that is defined in this way, we call P^M_{cycl}. Formula (2.10) implies that

$$\begin{array}{ll} S_j^m[k+1] = kc + S_j^m[1] & 1 \le j \le N, 1 \le m \le M, k \ge 1 \text{ and} \\ T_j^m[k+1] = kc + T_j^m[1] & 1 \le j \le N, 1 \le m \le M, k \ge 1. \end{array} \tag{2.11}$$

Let $S_j^m = S_j^m[1]$ and $T_j^m = T_j^m[1]$ for $1 \le j \le N, 1 \le m \le M$.

In order to remove the "or" conditions in the equations of Problem formulation 1, we define for any pair of transports $(,T_{j_1}^{m_1})_t$ and $(,T_{j_2}^{m_2})_t$, with $j_1 \ne j_2$ and $m_1 < m_2$, a binary variable $x_{j_1 j_2}^{m_1 m_2}$ as follows:

$$x_{j_1 j_2}^{m_1 m_2} = \begin{cases} 0 & \text{if } j_1 < j_2 \text{ and } (,T_{j_1}^{m_1} + c)_t \text{ is an later transport than } (,T_{j_2}^{m_2})_t, \\ 1 & \text{if } j_1 < j_2 \text{ and } (,T_{j_1}^{m_1} + c)_t \text{ is a earlier transport than } (,T_{j_2}^{m_2})_t, \\ 0 & \text{if } j_1 > j_2 \text{ and } (,T_{j_1}^{m_1})_t \text{ is an later transport than } (,T_{j_2}^{m_2})_t, \\ 1 & \text{if } j_1 > j_2 \text{ and } (,T_{j_1}^{m_1})_t \text{ is a earlier transport than } (,T_{j_2}^{m_2})_t, \end{cases}$$

where $1 \le j_1, j_2 \le N, j_1 \ne j_2$ and $1 \le m_1 < m_2 \le M$.

Now, Problem formulation 1, with additional condition (2.10), can be reformulated in such a way that Problem formulation 2 of Figure 2.4 is obtained. Also for a feasible solution of Problem formulation 2, Assumption 2.3 does not necessarily hold. Again, this can be solved by moving processings and transports backwards in time, as shown in Figure 2.5. To stress the cyclicity of the schedule, we prefer to say that we move the processings and transports *counterclockwise.*

Some of the inequalities of Problem formulation 2 are nonlinear, since they contain the product $x_{j_1 j_2}^{m_1 m_2} c$. In order to transform the model into a linear mixed-

minimize c

subject to

$$T_j^m \le S_j^m \qquad 1 \le j \le N, 1 \le m \le M.$$

$$\begin{aligned} &S_j^m \ge S_{j-1}^m + p_{j-1} && 2 \le j \le N, 1 \le m \le M, \\ &S_1^m + c \ge S_N^m + p_N && 1 \le m \le M. \end{aligned}$$

$$\begin{aligned} &T_j^m \ge S_j^{m-1} + p_j + d && 1 \le j \le N, 2 \le m \le M, \\ &T_j^1 + c \ge S_j^M + p_j + (M-1)d && 1 \le j \le N. \end{aligned}$$

$$\begin{aligned} &T_j^1 \ge T_{j-1}^1 + 2(M-1)d && 2 \le j \le N, 2 \le m \le M, \\ &T_1^1 + c \ge T_N^1 + 2(M-1)d && 2 \le m \le M. \end{aligned}$$

$$\begin{aligned} &T_j^m \ge T_{j-1}^m + 2d \\ &T_1^m + c \ge T_N^m + 2d \end{aligned} \qquad 2 \le j \le N,$$

$$T_{j_2}^{m_2} - T_{j_1}^{m_1} \quad \left\{ \begin{array}{rl} \le & (1 + x_{j_1 j_2}^{m_1 m_2})c - (m_2 - m_1 + 2)d \\ \ge & x_{j_1 j_2}^{m_1 m_2} c + (m_2 - m_1)d \end{array} \right\} \qquad \begin{aligned} &1 \le j_1 < j_2 \le N, \\ &2 \le m_1 < m_2 \le M. \end{aligned}$$

$$T_{j_2}^{m_2} - T_{j_1}^{m_1} \quad \left\{ \begin{array}{rl} \le & x_{j_1 j_2}^{m_1 m_2} c - (m_2 - m_1 + 2)d \\ \ge & (x_{j_1 j_2}^{m_1 m_2} - 1)c + (m_2 - m_1)d \end{array} \right\} \qquad \begin{aligned} &1 \le j_2 < j_1 \le N, \\ &2 \le m_1 < m_2 \le M. \end{aligned}$$

$$T_{j_2}^{m} - T_{j_1}^{1} \quad \left\{ \begin{array}{rl} \le & (1 + x_{j_1 j_2}^{1m})c - (2M - m - 1)d \\ \ge & x_{j_1 j_2}^{1m} c + (m-1)d \end{array} \right\} \qquad \begin{aligned} &1 \le j_1 < j_2 \le N, \\ &2 \le m \le M. \end{aligned}$$

$$T_{j_2}^{m} - T_{j_1}^{1} \quad \left\{ \begin{array}{rl} \le & x_{j_1 j_2}^{1m} c - (2M - m - 1)d \\ \ge & (x_{j_1 j_2}^{1m} - 1)c + (m-1)d \end{array} \right\} \qquad \begin{aligned} &1 \le j_2 < j_1 \le N, \\ &2 \le m \le M. \end{aligned}$$

$$T_1^1 = 0$$

$$x_{j_1 j_2}^{m_1 m_2} \in \{0,1\} \qquad \begin{aligned} &1 \le j_1, j_2 \le N, j_1 \ne j_2, \\ &1 \le m_1 < m_2 \le M. \end{aligned}$$

Figure 2.4. Problem formulation 2

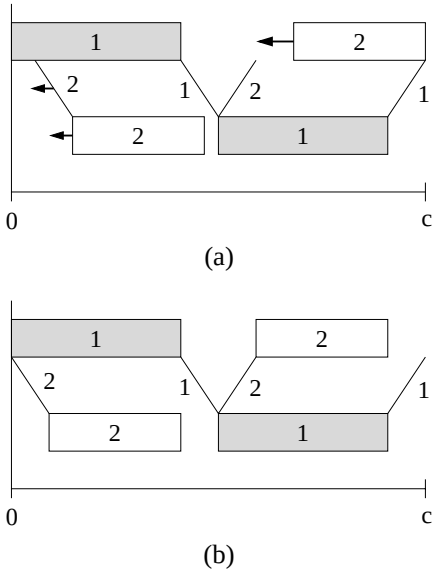


Figure 2.5. (a) Assumption 2.3 does not hold,(b) Assumption 2.3 holds.

integer program, we introduce the "relative" variables

$$t_j^m = T_j^m \cdot c^{-1}$$

and

$$s_j^m = S_j^m \cdot c^{-1}$$

and the production rate per machine

$$q = c^{-1},$$

which by Lemma 2.3 are well-defined. By dividing all inequalities in Problem formulation 2 by c, we finally obtain the linear mixed-integer program of Figure 2.6. We call this linear mixed-integer program MIP^M_{cycl}. (A similar problem formulation can be found in Hanen [1994].)

In contrast with Problem formulation 2, MIP^M_{cycl} is a linear mixed-integer program. However, since solutions of Problem formulation 2 can easily be transformed into solutions of MIP^M_{cycl} and vice versa, and we find the variables S_j^m and T_j^m easier to understand concepts than the variables s_j^m and t_j^m, we consider S_j^m and T_j^m to be our main decision variables.

The values of the variables S_j^m and T_j^m cannot be arbitrarily large. They can be bounded, not as one would expect by the cycle time c, but by $2c$.

$$\text{minimize } -q$$

subject to

$$t_j^m \le s_j^m \qquad 1 \le m \le M, 1 \le j \le N.$$

$$\begin{array}{ll} s_j^m \ge s_{j-1}^m + p_{j-1}q & 2 \le j \le N, 1 \le m \le M \\ s_1^m + 1 \ge s_N^m + p_N q & 1 \le m \le M. \end{array}$$

$$\begin{array}{ll} t_j^m \ge s_j^{m-1} + (p_j + d)q & 1 \le j \le N, 2 \le m \le M \\ t_j^1 + 1 \ge s_j^M + (p_j + (M-1)d)q & 1 \le j \le N, \end{array}$$

$$\begin{array}{ll} t_j^1 \ge t_{j-1}^1 + 2(M-1)dq & \\ t_1^1 + 1 \ge t_N^1 + 2(M-1)dq & 2 \le j \le N, \end{array}$$

$$\begin{array}{ll} t_j^m \ge t_{j-1}^m + 2dq & \\ t_1^m + 1 \ge t_N^m + 2dq & 2 \le j \le N, \end{array}$$

$$t_{j_2}^{m_2} - t_{j_1}^{m_1} \quad \left\{ \begin{array}{ll} \le & (1 + x_{j_1 j_2}^{m_1 m_2}) - (m_2 - m_1 + 2)dq \\ \ge & x_{j_1 j_2}^{m_1 m_2} + (m_2 - m_1)dq \end{array} \right\} \qquad \begin{array}{l} 1 \le j_1 < j_2 \le N, \\ 2 \le m_1 < m_2 \le M. \end{array}$$

$$t_{j_2}^{m_2} - t_{j_1}^{m_1} \quad \left\{ \begin{array}{ll} \le & x_{j_1 j_2}^{m_1 m_2} - (m_2 - m_1 + 2)dq \\ \ge & (x_{j_1 j_2}^{m_1 m_2} - 1) + (m_2 - m_1)dq \end{array} \right\} \qquad \begin{array}{l} 1 \le j_2 < j_1 \le N, \\ 2 \le m_1 < m_2 \le M. \end{array}$$

$$t_{j_2}^{m} - t_{j_1}^{1} \quad \left\{ \begin{array}{ll} \le & (1 + x_{j_1 j_2}^{1m}) - (2M - m - 1)dq \\ \ge & x_{j_1 j_2}^{1m} + (m-1)dq \end{array} \right\} \qquad \begin{array}{l} 1 \le j_1 < j_2 \le N, \\ 2 \le m \le M. \end{array}$$

$$t_{j_2}^{m} - t_{j_1}^{1} \quad \left\{ \begin{array}{ll} \le & x_{j_1 1 j_2}^{1m} - (2M - m - 1)dq \\ \ge & (x_{j_1 j_2}^{1m} - 1) + (m-1)dq \end{array} \right\} \qquad \begin{array}{l} 1 \le j_2 < j_1 \le N, \\ 2 \le m \le M. \end{array}$$

$$t_1^1 = 0$$

$$x_{j_1 j_2}^{m_1 m_2} \in \{0,1\} \qquad \begin{array}{l} 1 \le j_1, j_2 \le N, j_1 \ne j_2, \\ 1 \le m_1 < m_2 \le M. \end{array}$$

Figure 2.6. Problem formulation 3 (MIP_{cycl}^M).

Lemma 2.4.

(a) $0 \leq T_j^1 < c \quad (1 \leq j \leq N)$,
(b) $0 \leq T_j^m < 2c \quad (1 \leq j \leq N, 2 \leq m \leq M)$,
(c) $0 \leq S_1^1 < c$,
(d) $0 \leq S_j^m < 2c \quad (1 \leq j \leq N, 1 \leq m \leq M, j \neq 1 \text{ or } m \neq 1)$.

Proof. With the inequalities of Problem formulation 2 the following can easily be seen.
(a) For any task $j, 1 \leq j \leq N$:

$$0 = T_1^1 \leq T_j^1 \leq T_N^1 < T_1^1 + c = c.$$

(b) For any task $j, 1 \leq j \leq N$ and machine $m, 2 \leq m \leq M$:

$$0 = T_1^1 < T_1^m \leq T_j^m < T_1^m + c < T_1^1 + 2c = 2c.$$

(c)

$$0 = T_1^1 \leq S_1^1 < T_1^M \leq S_1^M < T_1^1 + c = c.$$

(d) For any task $j, 1 \leq j \leq N$ and machine $m, 1 \leq m \leq M$, with $j \neq 1$ or $m \neq 1$:

$$0 = T_1^1 \leq S_1^1 \leq S_j^1 \leq S_j^m \leq S_j^M < T_j^1 + c < T_1^1 + 2c = 2c.$$

□

Notice that Lemma 2.4 implies the following.

Corollary 2.1. *For any task j and any machine m we have that*

(a) either $S_j^m \in [0,c)$ or $S_j^m - c \in [0,c)$ and
(b) either $T_j^m \in [0,c)$ or $T_j^m - c \in [0,c)$.

□

Although the variables S_j^m and the $T_j^m (1 \leq j \leq N, 1 \leq m \leq M)$ are the decision variables, in fact the crucial decision to take is in which order to carry out the different tool transports. As soon as this order is known, an optimal schedule can easily be calculated. For the linear version of the problem this can be done directly: one starts at time zero and successively adds the transports and processings one by one, taking into account the conditions (2.1)-(2.8). In the cyclic version of the problem, there is no such an explicit time zero at which the production system is started up. However, as soon for the cyclic version of the problem the order of the tool transports is known, the variables $x_{j_1 j_2}^{m_1 m_2} (1 \leq j_1, j_2 \leq N, j_1 \neq j_2, 1 \leq m_1 < m_2 \leq M)$ are fixed and an optimal schedule can be found by solving a linear programming problem.

We see that the order in which the tools are transported to the machines is very important. We call this order the *transport strategy.*

2.5 The cyclic problem for two machines

The remaining part of this thesis is dedicated to problem P^2_{cycl}. Therefore, we give Problem formulation 2 explicitly for $M = 2$ (see Figure 2.7).

minimize c

subject to

$$T_j^m \le S_j^m \qquad 1 \le j \le N, m = 1,2. \tag{2.13}$$

$$\begin{aligned} &S_j^m \ge S_{j-1}^m + p_{j-1} \\ &S_1^m + c \ge S_N^m + p_N \end{aligned} \qquad \begin{aligned} &2 \le j \le N, m = 1,2 \\ &m = 1,2. \end{aligned} \tag{2.14}$$

$$\begin{aligned} &T_j^2 \ge S_j^1 + p_j + d \\ &T_j^1 + c \ge S_j^2 + p_j + d \end{aligned} \qquad \begin{aligned} &1 \le j \le N, \\ &1 \le j \le N. \end{aligned} \tag{2.15}$$

$$\begin{aligned} &T_j^m \ge T_{j-1}^m + 2d \\ &T_1^m + c \ge T_N^m + 2d \end{aligned} \qquad \begin{aligned} &2 \le j \le N, m = 1,2, \\ &m = 1,2. \end{aligned} \tag{2.16}$$

$$T_{j_2}^2 - T_{j_1}^1 \quad \left\{ \begin{array}{rc} \le & (1 + x_{j_1 j_2})c - d \\ \ge & x_{j_1 j_2} c + d \end{array} \right\} \qquad 1 \le j_1 < j_2 \le N. \tag{2.17}$$

$$T_{j_2}^2 - T_{j_1}^1 \quad \left\{ \begin{array}{rc} \le & x_{j_1 1 j_2} c - d \\ \ge & (x_{j_1 j_2} - 1)c + d \end{array} \right\} \qquad 1 \le j_2 < j_1 \le N. \tag{2.18}$$

$$T_1^1 = 0 \tag{2.19}$$

$$x_{j_1 j_2} \in \{0,1\} \qquad 1 \le j_1, j_2 \le N, j_1 \ne j_2.$$

Figure 2.7. Problem formulation 4.

Recall that the binary variables $x_{j_1 j_2}$ are necessary to describe the transport strategy. In Figure 2.8(a) the situation $x_{j_1 j_2} = 1$ is sketched for $j_1 < j_2$. Figure 2.8(b) shows the situation $x_{j_1 j_2} = 1$ for $j_1 > j_2$, or the situation $x_{j_1 j_2} = 0$ for $j_1 < j_2$. Finally, in Figure 2.8(c) the situation $x_{j_1 j_2} = 0$ for $j_1 > j_2$ is sketched.

The number of binary variables is, in general, large. However, implicit relations between the x_{ij} exist that may reduce the number of necessary decisions. More precisely, prescribing the value of one of the variables, may induce the fixation of the values of a lot of other variables. The following can easily be observed.

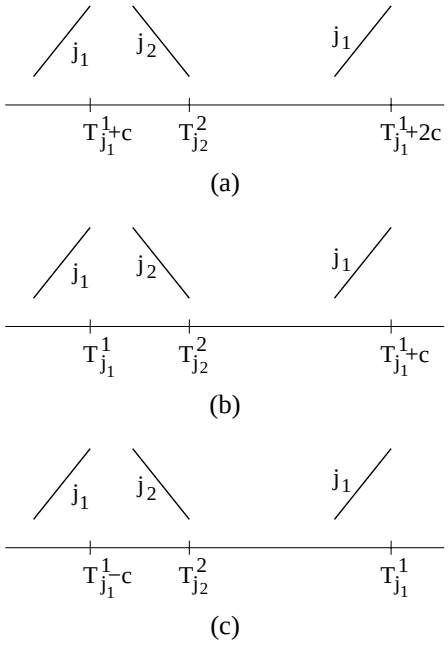


Figure 2.8. For $j_1 < j_2$: (a) $x_{j_1 j_2} = 1$, (b) $x_{j_1 j_2} = 0$. For $j_1 > j_2$: (b) $x_{j_1 j_2} = 1$, (c) $x_{j_1 j_2} = 0$.

Observation 2.1.

(a) $x_{j_1 j_2} = 1 \Rightarrow x_{j_1-1, j_2} = 1 \quad 2 \le j_1 \le N, 1 \le j_2 \le N, j_2 \notin \{j_1 - 1, j_1\}.$
(b) $x_{j_1 j_2} = 1 \Rightarrow x_{j_1, j_2+1} = 1 \quad 1 \le j_1 \le N, 1 \le j_2 \le N-1, j_1 \notin \{j_2, j_2 + 1\}.$
(c) $x_{j_1 j_2} = 0 \Rightarrow x_{j_1+1, j_2} = 0 \quad 1 \le j_1 \le N-1, 1 \le j_2 \le N, j_2 \notin \{j_1, j_1 + 1\}.$
(d) $x_{j_1 j_2} = 0 \Rightarrow x_{j_1, j_2-1} = 0 \quad 1 \le j_1 \le N, 2 \le j_2 \le N, j_1 \notin \{j_2 - 1, j_2\}.$

□

By combining these observations, the following linear inequalities can easily be derived:

Corollary 2.2.

(a) $x_{j_1 j_2} \le x_{j_1-1, j_2} \quad 2 \le j_1 \le N, 1 \le j_2 \le N, j_2 \ne \{j_1 - 1, j_1\}.$
(b) $x_{j_1 j_2} \le x_{j_1, j_2+1} \quad 1 \le j_1 \le N, 1 \le j_2 \le N-1, j_1 \ne \{j_2, j_2 + 1\}.$

□

3

Structural properties of schedules

3.1 Introduction

In this chapter we consider some properties of feasible and optimal schedules for the two-machine problem P^2_{cycl}.

First, we derive some bounds on the optimal cycle time that can easily be calculated, after which we describe optimal schedules for the situations in which $N \leq 3$. Next, we look at situations in which the described lower bounds can actually be attained. Also the relation between a schedule and the movements of the robot is studied. Lastly, we examine the consequences of the assumptions made in Section 2.2.

However, before doing all this, we first introduce some additional notation. An instance of problem P^2_{cycl} is denoted by a tuple $(\boldsymbol{p}, d)$, where $\boldsymbol{p} = (p_1, p_2, \ldots, p_N)$ is the vector of the processing times of the tasks. Furthermore, we denote feasible schedules by $S, \overline{S}$ or $\overline{\overline{S}}$, with, respectively, processing start times $S_j^m, \overline{S}_j^m, \overline{\overline{S}}_j^m$, tool arrival times $T_j^m, \overline{T}_j^m, \overline{\overline{T}}_j^m$ and cycle times $c, \overline{c}$ and $\overline{\overline{c}}$. Finally, c_{opt} denotes the optimal cycle time.

3.2 Lower and upper bounds

3.2.1 Lower bounds

Consider the problem formulation $\text{MIP}^2_{\text{cycl}}$. Obviously, solving a relaxation of $\text{MIP}^2_{\text{cycl}}$ gives an upper bound on the optimal value of q and therefore a lower bound

on c_{opt}. One relaxation that can easily be solved is the LP-relaxation of $\text{MIP}^2_{\text{cycl}}$, which we indicate by LP_{relax}.

Also other, more explicit, lower bounds on c_{opt} can easily be deduced. Notice, e.g., that in one cycle time the robot transports all tools exactly once to machine 1 and once to machine 2. Hence, in one cycle the robot carries out at least $2N$ transports which all take time d. This observation leads to the following result.

Lemma 3.1. $c_{\text{opt}} \geq 2Nd$.

Proof. With (2.16) and (2.19) we see that

$$c = T_1^1 + c \geq T_N^1 + 2d \geq T_{N-1}^1 + 4d \geq \cdots \geq T_1^1 + 2Nd = 2Nd.$$

This holds for any cycle time c and therefore also for c_{opt}. □

Furthermore, in one cycle each tool is used for a processing on machine 1, a processing on machine 2 and is transported twice. This implies the following lemma.

Lemma 3.2. $c_{\text{opt}} \geq 2p_{\max} + 2d$.

Proof. With (2.13) and (2.15) we see that

$$\begin{aligned} c &= T_j^1 + c - T_j^1 \\ &\geq S_j^2 + p_j + d - T_j^1 \\ &\geq T_j^2 + p_j + d - T_j^1 \\ &\geq S_j^1 + 2p_j + 2d - T_j^1 \\ &\geq 2p_j + 2d. \end{aligned}$$

This holds for any $j, 1 \leq j \leq N$ and for any cycle time c. Hence, $c_{\text{opt}} \geq 2p_{\max} + 2d$. □

We also know that in one cycle each machine processes all tasks exactly once. This gives the following lemma.

Lemma 3.3. $c_{\text{opt}} \geq \sum_{j=1}^{N} p_j$.

Proof. Using (2.14), we see that

$$\begin{aligned} c &= S_1^1 + c - S_1^1 \\ &\geq S_N^1 + p_N - S_1^1 \\ &\geq S_{N-1}^1 + p_N + p_{N-1} - S_1^1 \\ &\geq \cdots \\ &\geq \sum_{j=1}^{N} p_j. \end{aligned}$$

This holds for any cycle time c and therefore also for the optimal cycle time. □

Notice that in the proofs of the above three lemma's, we only used formulas in which the binary variables $x_{j_1 j_2} (1 \leq j_1, j_2 \leq N, j_1 \neq j_2)$ are absent. Therefore, if $c_{\text{opt}}(LP_{\text{relax}})$ denotes the optimal cycle time of LP_{relax}, we have

$$c_{\text{opt}}(LP_{\text{relax}}) \geq \max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N} p_j\}.$$

3.2.2 Upper bounds

Evidently, any feasible solution of the problem yields an upper bound on c_{opt} (or a lower bound on the optimal value of q). This implies that we can find upper bounds on c_{opt} by first fixing the values of the integral variables $x_{j_1 j_2} (1 \leq j_1, j_2 \leq N;$ $j_1 \neq j_2)$ and then solving the resulting linear program.

An explicit expression for an upper bound on c_{opt} for any problem instance can also be given.

Lemma 3.4. $c_{\text{opt}} \leq 2(Nd + \sum_{j=1}^{N} p_j)$.

Proof. It is not very difficult to see that the solution

$$\begin{array}{ll} S_j^1 = \sum_{i=1}^{j-1} 2(p_i + d) & 1 \leq j \leq N, \\ T_j^2 = S_j^2 = S_j^1 + p_j + d & 1 \leq j \leq N, \\ T_j^1 = S_j^2 + p_j + d - c & 1 \leq j \leq N, \\ c = 2(Nd + \sum_{j=1}^{N} p_j), & \end{array}$$

constitutes a feasible schedule. For $N = 3$ this schedule is sketched in Figure 3.1. □

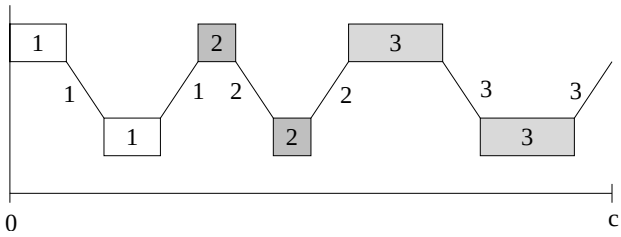


Figure 3.1. A feasible schedule for $N = 3$ with cycle time $2(3d + \sum_{j=1}^{3} p_j)$.

Notice that for the schedule in Figure 3.1 Assumption 2.3 does not hold. However, as shown in Section 2.3, the schedule can easily be transformed into a feasible schedule that is at least as good and for which the assumption holds.

Since according to the Lemma's 3.1 and 3.3 the values $2Nd$ and $\sum_{j=1}^{N} p_j$ are both lower bounds on the optimal cycle time, the following can be observed.

Observation 3.1. *The cycle time of the schedule of Lemma 3.4 is smaller than or equal to* $3c_{\text{opt}}$. □

3.3 A small number of tasks

3.3.1 Introduction

For very small N the problem can be solved explicitly. The case $N = 1$ is not very interesting: in this case the best result is obtained by using only one machine. We consider the cases $N = 2$ and $N = 3$.

3.3.2 N=2

Without loss of generality, we assume that $p_1 \geq p_2$.

If $p_1 + p_2 \leq 2d$, then it can easily be verified that

$$\begin{aligned} &T_j^1 = S_j^1 = (j-1)(p_1+2d) && j = 1,2, \\ &T_j^2 = S_j^2 = p_1 + d + (j-1)(p_2+2d) && j = 1,2, \\ &c = p_1 + p_2 + 4d \end{aligned}$$

is an optimal schedule. In Figure 3.2 the above schedule is drawn for problem instance $(\boldsymbol{p}, d) = ((3,2),3)$.

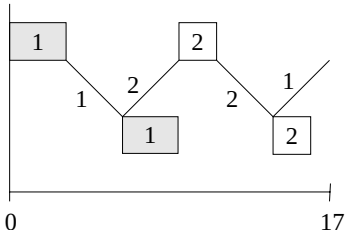


Figure 3.2. An optimal schedule for problem instance $(\boldsymbol{p}, d) = ((3,2),3)$.

If $p_1 + p_2 > 2d$ and $p_1 \leq 2d$, then the values

$$\begin{aligned} &T_j^1 = S_j^1 = 2(j-1)d && j = 1,2, \\ &T_j^2 = S_j^2 = 3d + 2(j-1)d && j = 1,2, \\ &c = 6d \end{aligned}$$

constitute an optimal solution. In Figure 3.3 this solution is shown for problem instance $(\boldsymbol{p}, d) = ((4,3),3)$.

Finally, if $p_1 > 2d$, then

$$\begin{aligned} &T_j^1 = S_j^1 = (j-1)p_1 && j = 1,2, \\ &T_j^2 = S_j^2 = p_1 + d + (j-1)p_1 && j = 1,2, \\ &c = 2p_1 + 2d \end{aligned}$$

is a feasible solution, which by Lemma 3.2, is also optimal. The solution is shown for problem instance $(\boldsymbol{p}, d) = ((8,3),3)$ in Figure 3.4. Notice that also for this schedule Assumption 2.3 does not hold.

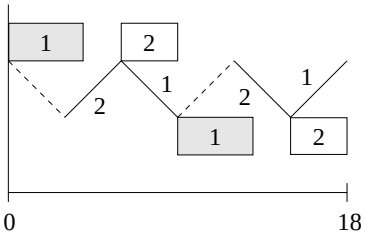


Figure 3.3. An optimal schedule for problem instance $(\boldsymbol{p},d) = ((4,3),3)$.

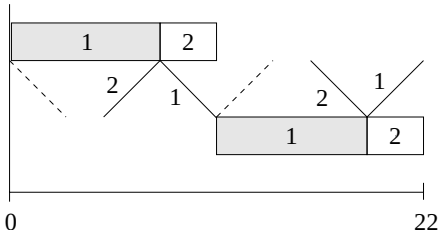


Figure 3.4. An optimal schedule for problem instance $(\boldsymbol{p},d) = ((8,3),3)$.

Observe that, for any of the cases, if $2 \cdot c^{-1} < (p_1 + p_2)^{-1}$, then the number of jobs processed per unit of time can be increased from $2 \cdot c^{-1}$ up to $(p_1 + p_2)^{-1}$ by releasing Assumption 2.1 and using only one machine.

3.3.3 N=3

For $N = 3$, the optimal value of the cycle time can directly be calculated.

Theorem 3.1. *For $N = 3$ we have that $c_{\text{opt}} = \max\{2p_{\max} + 2d, \sum_{j=1}^{3} p_j, 6d\}$.*

Proof. Without loss of generality, we assume that task $j_{\max}$ is task 1. We distinguish the following cases.

In case $p_1 \leq 2d$, the following values

$$\begin{aligned} T_j^1 &= S_j^1 = 2(j-1)d \quad j = 1,2, \\ T_j^2 &= S_j^2 = d + 2jd \qquad j = 1,2, \\ c &= 6d \end{aligned}$$

constitute a feasible schedule. By Lemma 3.1, they also constitute an optimal schedule. For problem instance $(\boldsymbol{p},d) = ((4,3,2),3)$, the above schedule is shown in Figure 3.5.

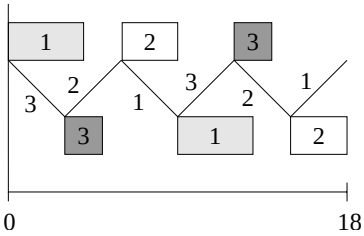


Figure 3.5. An optimal schedule for problem instance $(\boldsymbol{p},d)=((4,3,2),3)$.

If $p_1 > 2d$ and $p_1+2d \geq p_2+p_3$, then

$$\begin{aligned}
&T_j^1 = S_j^1 = (j-1)p_1, && j=1,2,\\
&T_3^1 = p_1+2d\\
&S_3^1 = \max\{p_1+p_2, p_1+2d\}\\
&S_j^2 = S_j^1+p_1+d && j=1,2,3,\\
&T_j^2 = T_j^1+p_1+d && j=1,2,3,\\
&c = 2p_1+2d,
\end{aligned}$$

is a feasible solution, which in view of Lemma 3.2 is also optimal. This solution

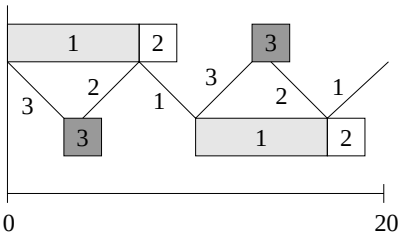


Figure 3.6. An optimal schedule for problem instance $(\boldsymbol{p},d)=((7,2,2),3)$.

is shown for problem instance $(\boldsymbol{p},d)=((7,2,2),3)$ in Figure 3.6. Also for this schedule Assumption 2.3 does not hold.

Lastly, if $p_1 > 2d$ and $p_1+2d \leq p_2+p_3$, then we have that $p_j \geq 2d$ for all tasks j and therefore it can easily be seen that

$$\begin{aligned}
&S_j^1 = T_j^1 = \textstyle\sum_{i=1}^{j-1} p_i && j=1,2,3,\\
&S_j^2 = S_j^1+p_1+d && j=1,2,3,\\
&T_j^2 = S_j^1+p_j+d && j=1,2,3,\\
&c = \textstyle\sum_{i=1}^{3} p_i,
\end{aligned}$$

is a feasible schedule, which by Lemma 3.3 is also optimal. In Figure 3.7 this solution is shown for problem instance $(\boldsymbol{p},d)=((7,6,6),2)$.

□

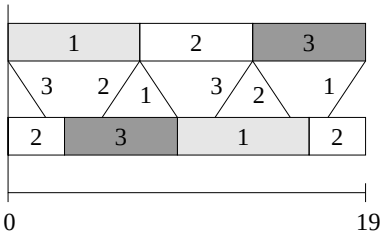


Figure 3.7. An optimal solution for problem instance $(\boldsymbol{p},d)=((7,6,6),2)$.

For any of the cases, if $2\cdot c^{-1}<(p_1+p_2+p_3)^{-1}$, then the number of jobs processed per unit of time can be increased from $2\cdot c^{-1}$ up to $(p_1+p_2+p_3)^{-1}$ by using only one machine.

3.4 Attaining the lower bounds

As can be seen in the examples of the previous section, problem instances exist for which the optimal cycle time value is equal to one of the lower bounds described in Section 3.2.1. In this section we give some necessary and sufficient conditions for attaining the lower bounds $2Nd, 2p_{\max}+2d$ and $\sum_{i=1}^{N} p_i$.

Obviously, lower bound $lb \in \{2Nd, \sum_{j=1}^{N} p_j, 2p_{\max}+2d\}$ can only be attained if $lb=\max\{2Nd, \sum_{j=1}^{N} p_j, 2p_{\max}+2d\}$.

In the following two lemma's, we consider the lower bound $2Nd$.

Lemma 3.5. *If lower bound $2Nd$ is attained, then there exists a task, the processing time of which is smaller than or equal to $2d$.*

Proof. Suppose that there is not such a task. This implies that $\sum_{j=1}^{N} p_j > 2Nd$ and consequently, lower bound $2Nd$ cannot be attained. □

Lemma 3.6. *If $N\geq 3$ and $p_j\leq 2d$ for every $j, 1\leq j\leq N$, then lower bound $2Nd$ is attained.*

Proof. If $N\geq 3$ and $p_j\leq 2d$ for every task j, then

$$\begin{aligned} T_j^1 = S_j^1 &= (j-1)2d \qquad 1\leq j\leq N,\\ T_j^2 = S_j^2 &= 2jd+d \qquad 1\leq j\leq N,\\ c &= 2Nd \end{aligned}$$

is a feasible schedule. For problem instance $(\boldsymbol{p},d)=((1,2,1,2,1),1)$, the schedule is shown in Figure 3.8. □

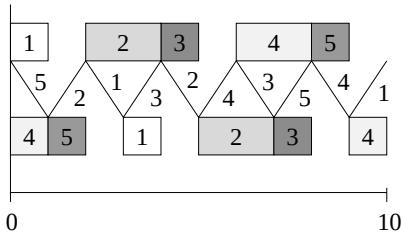


Figure 3.8. An optimal solution for problem instance $(\boldsymbol{p},d)=((1,2,1,2,1),1)$.

The following lemma gives sufficient conditions for attaining lower bound $2p_{\max}+2d$.

Lemma 3.7. *If $N\geq 2$ and $p_{\max}\geq \frac{1}{2}\sum_{i=1}^{N}p_j+(N-2)d$, then lower bound $2p_{\max}+2d$ is attained.*

Proof. Without loss of generality, we assume again that task $j_{\max}$ is task 1. If $N\geq 2$ and $p_{\max}\geq \frac{1}{2}\sum_{i=1}^{N}p_j+(N-2)d$ then the schedule

$$\begin{aligned} S_j^m &= \sum_{i=1}^{j-1}p_i+(m-1)(p_1+d), && 1\leq j\leq N, m=1,2,\\ T_j^m &= p_1-2(N-j)d+(m-1)(p_1+d) && 2\leq j\leq N, m=1,2,\\ T_1^m &= (m-1)(p_1+d) && m=1,2,\\ c &= 2p_1+2d \end{aligned}$$

is a feasible schedule. For problem instance $(\boldsymbol{p},d)=((10,1,2,1),1)$ the corresponding schedule is shown in Figure 3.9. □

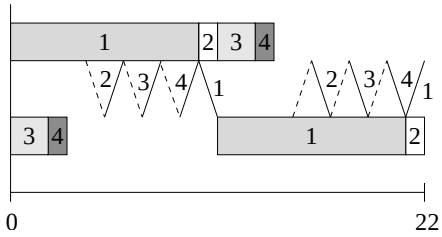


Figure 3.9. An optimal solution for problem instance $(\boldsymbol{p},d)=((10,1,2,1),1)$.

Lastly, we consider sufficient conditions for attaining lower bound $\sum_{j=1}^{N}p_j$.

Lemma 3.8. *If $N\geq 3$, $p_j\geq 2d$ for all $j, 1\leq j\leq N$ and $\sum_{j=1}^{N}p_j\geq 2p_{\max}+2d$ then lower bound $\sum_{j=1}^{N}p_j$ is attained.*

Proof. In this case, the schedule

$$\begin{aligned} T_j^1 = S_j^1 = \sum_{i=1}^{j-1} p_i, \qquad & 1 \le j \le N, \\ S_j^2 = S_j^1 + p_{\max} + d, \qquad & 1 \le j \le N, \\ T_j^2 = S_j^1 + p_j + d \qquad & 1 \le j \le N, \\ c = \sum_{j=1}^{N} p_j & \end{aligned}$$

is a feasible schedule. For problem instance $(\boldsymbol{p}, d) = ((7,6,6),2)$, the corresponding schedule is shown in Figure 3.7. □

3.5 Transport strategies

3.5.1 Introduction

In Section 2.4 we have already seen that the order in which the tool transports occur is of crucial importance. However, not every possible order of tool transports constitutes a feasible transport strategy. Consider, e.g., for $N = 2$, the order $(,T_1^1)_t,(,T_1^2)_t,(,T_2^2)_t,(,T_2^1)_t$. This is no feasible strategy, because in this order $T_2^2 < T_2^1$ and this is in contradiction with Assumption 2.2.

In this section, we derive an upper bound on the number of feasible transport strategies. Furthermore, we look at some strategies, from which we expect that they lead to good schedules.

3.5.2 The number of feasible transport strategies

As already said in the introduction, we first derive an upper bound on the number of feasible transport strategies.

Theorem 3.2. *The total number of feasible transport strategies is at most*

$$N \left(\begin{array}{c} 2N-1 \\ N \end{array} \right).$$

Proof. By Lemma 2.1, we know that $T_1^1 < T_2^1 < \ldots < T_N^1$ and that $T_1^2 < T_2^2 < \ldots < T_N^2$. However, we still have to combine these end times of the tool transports to machine 1 and machine 2 in some way. Suppose that the first tool that is transported to machine 2 after T_1^1 is tool j^*. This means, because of Assumption 2.2 and Lemma 2.1, that we have to merge the list $L_1 = [T_1^1, T_2^1, \ldots, T_N^1]$ with the list L_2, where $L_2 = [T_1^2, T_2^2, \ldots, T_N^2]$ if $j^* = 1$ and $L_2 = [T_{j^*}^2 - c, \ldots, T_N^2 - c, T_1^2, \ldots, T_{j^*-1}^2]$, otherwise. This merging has to happen in such a way that:

(a) T_1^1 is chosen before $T_{j^*}^2$.

(b) T_j^1 is inserted before T_j^2 $(1 \leq j \leq N)$.

(c) T_j^2 is inserted before $T_j^1 + c$ $(1 \leq j \leq N)$.

Suppose that we ignore the last two conditions. Then, we can use the following algorithm for merging:

Step 1 Make a list $L = [T_1^1, l_2, \ldots, l_{2N}]$.

Step 2 Select N numbers from the set $\{2, 3, \ldots, 2N\}$.

Step 3 If i is selected in Step 2, then l_i is chosen to be an element of L_2, otherwise it is chosen to be of L_1 $(i = 2, 3, \ldots, N)$.

Notice that in Step 3 there are $\binom{2N-1}{N}$ ways to select the N numbers, and that because of the order restrictions, Step 3 can only be carried out in one way. Moreover, there exist N different choices for j^*. Hence, there are at most $N\binom{2N-1}{N}$ feasible transport strategies. □

We can estimate this upper bound on the number of transport strategies as follows:

$$N\binom{2N-1}{N} = \frac{N}{2} \cdot \frac{(2N)!}{(N!)^2} \approx \frac{N}{2} \cdot \frac{(2N)^{2N}\sqrt{2\pi(2N)}e^{-2N}}{N^{2N}2\pi N e^{-2N}}$$

$$= \frac{N}{2} \cdot \frac{2^{2N}\sqrt{\pi N}}{\pi N} = 2^{2N-1} \cdot \frac{\sqrt{N}}{\sqrt{\pi}}.$$

3.5.3 Interesting transport strategies

As remarked in Section 3.2.2, we can find upper bounds on the optimal cycle time by first choosing a prescribed transport strategy and then solving the resulting linear programming problem. In this section we consider some strategies with which we expect to obtain good upper bounds.

N odd

Intuitively, it seems to be a good idea to transport a tool to machine m about $\frac{1}{2}c$ after it is transported to machine $m \ominus_M 1$. Hence, for N is odd and $N \geq 3$, the following transport strategy seems to be interesting:

$$(, T_1^1)_t, (, T_{\frac{N+3}{2}}^2 - c)_t, (, T_2^1)_t, (, T_{\frac{N+5}{2}}^2 - c)_t, \ldots, (, T_{\frac{N+1}{2}}^1)_t, (, T_1^2)_t, \ldots, (, T_N^1)_t, (, T_{\frac{N+1}{2}}^2)_t.$$

In Figure 3.10 this strategy is drawn for $N = 5$.

Figure 3.10. An interesting transport strategy for $N = 5$.

As we will see in Chapter 7, this strategy gives for many problem instances an optimal schedule. However, it is not difficult to find examples, where this is not the case. If we consider, e.g., problem instance $(\boldsymbol{p},d) = ((4,12,1,1,10),2)$, then the above strategy gives a schedule of length 30, while transport strategy $(,T_1^1)_t$, $(,T_3^2-c)_t$, $(,T_4^2-c)_t$, $(,T_5^2-c)_t$, $(,T_2^1)_t$, $(,T_1^2)_t$, $(,T_3^1)_t$ $(,T_4^1)_t$, $(,T_5^1)_t$, $(,T_2^2)_t$ gives a schedule of length 28 (see Figure 3.11).

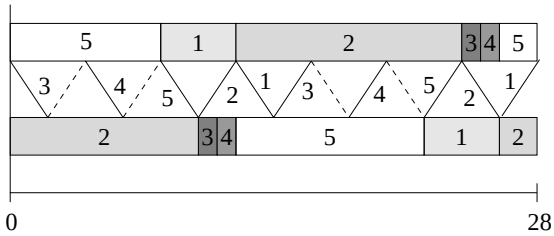


Figure 3.11. An optimal schedule for problem instance $(\boldsymbol{p},d) = ((4,12,1,1,10),2)$.

N even

For N is even, the above strategy cannot be used. However, following the same idea, the two following strategies seem to be interesting for N even and $N \geq 4$:

$$(,T_1^1)_t,(,T_{\frac{N+4}{2}}^2-c)_t,(,T_2^1)_t,(,T_{\frac{N+6}{2}}^2-c)_t,\ldots,(,T_{\frac{N}{2}}^1)_t,(,T_1^2)_t,\ldots,(,T_N^1)_t,(,T_{\frac{N+2}{2}}^2)_t$$

and

$$(,T_1^1)_t,(,T_{\frac{N+2}{2}}^2-c)_t,(,T_2^1)_t,(,T_{\frac{N+4}{2}}^2-c)_t,\ldots,(,T_{\frac{N+2}{2}}^1)_t,(,T_1^2)_t,\ldots,(,T_N^1)_t,(,T_{\frac{N}{2}}^2)_t.$$

In Figure 3.12 these strategies are drawn for $N = 4$. In Chapter 7 we will see that

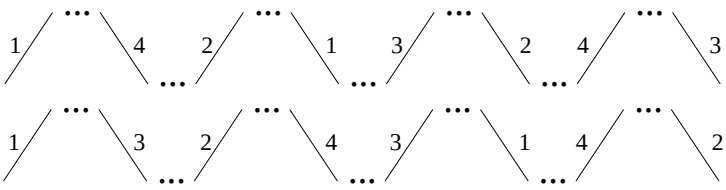


Figure 3.12. Some interesting transport strategies for $N = 4$.

these strategies will often give an optimal schedule. However, also for these strategies, it is possible to find examples in which they are not optimal. If we consider, e.g., problem instance $(\boldsymbol{p},d) = ((5,2,4,1),1)$, then the above strategies gives a best solution of 13 while transport strategy $(,T_1^1)_t$, $(,T_3^2-c)_t$, $(,T_2^1)_t$ $(,T_4^2-c)_t$, $(,T_1^2)_t$, $(,T_3^1)_t$, $(T_2^2)_t$, $(,T_4^1)_t$ gives an optimal solution of length 12 (see Figure 3.13).

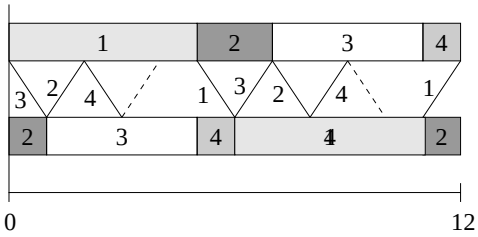


Figure 3.13. An optimal solution for problem instance $(\boldsymbol{p}, d) = ((5,2,4,1),1)$.

3.6 Consequences of the assumptions

In this section, we consider the consequences of the assumptions made in Section 2.2.

We first examine the consequences of using both machines (Assumption 2.1). Let $R(m)$ be the maximal average production per unit of time using m machines $(m = 1,2)$.

As we already remarked in Section 2.2, it is not always wise to use both machines. The next two lemma's give information about cases in which it is better to use one machine instead of two, and about how large the decrease of productivity may be if in these cases two machines are used.

Lemma 3.9. *If $\sum_{i=1}^{N} p_i < \max\{p_{\max} + d, Nd\}$ then it is better to use only one machine instead of two.*

Proof. This directly follows from $R(1) = (\sum_{i=1}^{N} p_i)^{-1}$ and $R(2) \leq 2 \cdot (\max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\})^{-1}$. □

Lemma 3.10. *For all $\alpha \in \mathbb{R}$ there exists a problem instance for which $\frac{R(1)}{R(2)} \geq \alpha$.*

Proof. Consider problem instance $(\boldsymbol{p}, d) = (\varepsilon, 1)$, where $0 < \varepsilon \ll 1$. For this instance $R(1) = \varepsilon^{-1}$. Figure 3.14 shows an optimal schedule using two machines. We see that $R(2) = 2 \cdot (2 + 2\varepsilon)^{-1}$.

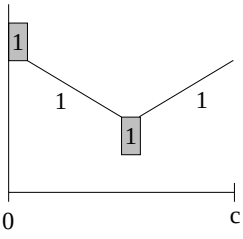


Figure 3.14. An optimal schedule (under Assumption 2.1).

Hence,

$$\lim_{\varepsilon\to 0}\frac{R(1)}{R(2)}=\lim_{\varepsilon\to 0}\frac{1+\varepsilon}{\varepsilon}=\infty.$$

□

Now let us look at the consequences of distributing the load equally over the machines (Assumption 2.2). Let $\bar{R}(m)$ be the maximal average production per unit of time if Assumption 2.2 is released ($m=1,2$).

Lemma 3.11. *For all $\alpha\in\mathbb{R}$ there exists a problem instance for which $\bar{R}(2)\cdot R(2)^{-1}\geq\alpha$, while $\bar{R}(2)>R(1)$.*

Proof. Consider problem instance $(\boldsymbol{p},d)$, where $p_i=N^{-1},1\leq i\leq N$, $d=(N-2)(2N)^{-1}$ and $N\geq 3$. For this instance, an optimal schedule on two machines is drawn in Figure 3.15. In Figure 3.16 an optimal schedule is shown if Assumption 2.2 is released. We see that

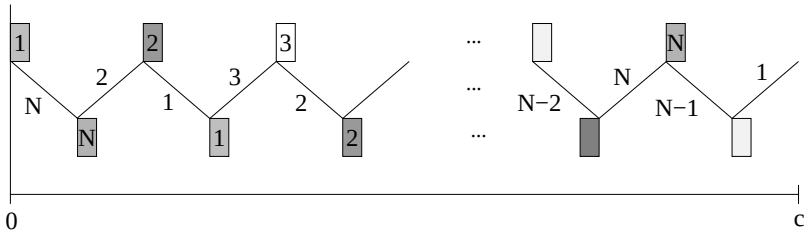


Figure 3.15. An optimal schedule (under Assumption 2.2).

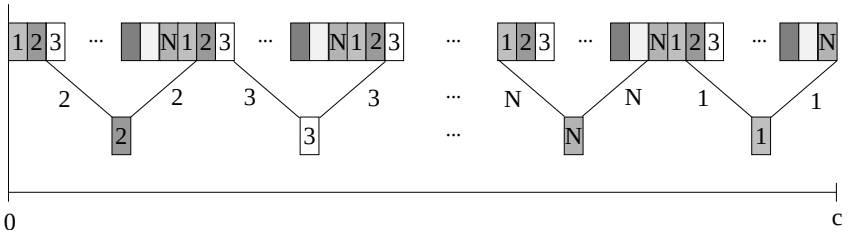


Figure 3.16. An optimal schedule (Assumption 2.2 released).

$$R(2)=\frac{2}{2Nd}=\frac{2}{N-2}$$

and

$$\bar{R}(2)=\frac{1+\frac{2}{N}}{1+\frac{1}{N}}=\frac{N+2}{N+1}.$$

Hence,

$$\lim_{N\to\infty} \bar{R}(2)/R(2) = \lim_{N\to\infty} \frac{N+2}{N+1} \cdot \frac{N-2}{2} = \infty.$$

Moreover, $R(1) = 1$ and therefore $\bar{R}(2)/R(1) > 1$. □

Notice that the remaining model assumptions (the Assumptions 2.3, 2.4 and 2.5 can be assumed without loss of generality.

4

Critical graphs and critical cycles

4.1 Introduction

In order to get more insight in the structure of feasible schedules, in this chapter we introduce two new notions: *critical graphs* and *critical cycles*. We show that, under Assumption 2.3, a feasible solution of P^2_{cyc} can be represented by a critical graph, and that the length of an optimal schedule can be characterized by a critical cycle.

Before doing this, we first recall some notions of basic graph theory that will be useful in this chapter. (For a more extensive introduction into graph theory, see, e.g., Bondy & Murty [1976]).

A *directed graph* is a pair $G = (V,A)$ where V is a finite set of *nodes* or *vertices* and A is a set of ordered pairs of vertices called *arcs*.

A *walk* in a directed graph $G = (V,A)$ is a sequence $[v_0, a_1, v_1, \ldots, a_k, v_k], k \geq 1$ of vertices from V and arcs from A such that a_i is the arc from vertex v_{i-1} to vertex v_i for $i = 1,2,\ldots,k$. If $v_{i_1} \neq v_{i_2}$ for all $1 \leq i_1, i_2 \leq k, i_1 \neq i_2$, then the walk is called a *path*. In case $v_0 = v_k$ and $v_{i_1} \neq v_{i_2}$ for all $1 \leq i_1, i_2 \leq k, i_1 \neq i_2$, we have a *cycle*. For convenience, we denote walk $[v_0, a_1, v_1, \ldots, a_k, v_k]$ by $[v_0, v_1, \ldots, v_k]$.

Frequently, vertices and/or arcs have values associated to them, these values are called *weights*. If for any vertex $v \in V$, w_v denotes the weight of v and for any

arc $a \in A$, w_a is the weight of a, then the *length of a walk* W is

$$\sum_{v \in W} w_v + \sum_{a \in W} w_a.$$

4.1.1 Critical graphs

In the first place, we introduce the concept of a *critical graph*. We do this as follows. For any feasible solution S of Problem formulation 4 for which Assumption 2.3 holds, we define a directed graph $G(S) = (V, A)$, where V is the set of all processings and all nonempty transports:

$$V = V_p \cup V_t,$$

with

$$V_p := \{(S_j^m,)_p | j = 1, \ldots, N; m = 1, 2\} \text{ and}$$

$$V_t := \{(, T_j^m)_t | j = 1, \ldots, N; m = 1, 2\}.$$

We call V the set of *events*.

We say that (v_1, v_2) is a pair of *possible subsequent events*, if in a feasible schedule event v_2 may occur later than event v_1 and there is no event v_3 that necessarily has to occur between v_1 and v_2.

For any pair of possible subsequent events (v_1, v_2) we define the *minimum time distance* from v_1 to v_2 as the minimum time that in a schedule has to occur between the end of v_1 and the beginning of v_2 in order to make the schedule feasible.

For almost all pairs of possible subsequent events this minimum time distance is equal to zero. E.g., the minimum time distances from $(S_1^1,)_p$ to $(S_2^1,)_p$ and from $(, T_1^1)_t$ to $(, T_2^2)_t$ are both equal to zero. However, if (v_1, v_2) is a pair of possible subsequent events, where v_1 and v_2 are two consecutive tool transports to the same machine, then in view of (2.16), the minimum time distance from v_1 to v_2 is d.

For the graph $G(S)$, we define the following set of arcs:

$A := \{(v_1, v_2) | v_1, v_2 \in V;$ (v_1, v_2) is a pair of possible subsequent events and in the schedule S the distance from v_1 to v_2 is equal to the minimum time distance from v_1 to $v_2\}$.

Notice that if $(v_1, v_2) \in A$, then the times at which the events v_1 and v_2 in S occur cause an equality in one or more of the constraints of Problem formulation 4. Hence, we can say that the arcs of $G(S)$, in some way, indicate "critical values" of the decision variables. Because of this, we call $G(S)$ the *critical graph* corresponding to schedule S. Observe that the graph $G(S)$ contains enough information to recalculate the schedule S, since by Assumption 2.3 each event is involved in at least one arc of $G(S)$.

4.2 Critical cycles

The critical graph focuses in some way on the critical relations within a schedule. Therefore, it somehow has to contain information, that characterizes the cycle time of the schedule. This idea leads to the concept of a *critical cycle*.

Eventually, we want a cycle in the graph $G(S)$ to represent the length of schedule S. Therefore, we give all the vertices of V a weight equal to the time it takes to carry out the corresponding event:

$$w_{(S_j^m,)_p} := p_j \quad (j = 1,\ldots,N; m = 1,2),$$
$$w_{(,T_j^m)_t} := d \quad (j = 1,\ldots,N; m = 1,2)$$

and we give each arc $(v_1, v_2) \in A$ a weight that is equal to the minimum time distance from v_1 to v_2:

$$w_{(v_1,v_2)} := \begin{cases} d & \text{if } v_1 \text{ and } v_2 \text{ are transports to the same machine,} \\ 0 & \text{otherwise.} \end{cases}$$

In this way, the length of any cycle in $G(S)$ becomes equal to the cycle time of schedule S. Therefore, we call cycles in $G(S)$ *critical cycles*. Notice that, because of Assumption 2.3, a critical graph always contains at least one critical cycle.

The critical graph belonging to the schedule of Figure 4.1 is shown in Figure 4.2. In Figure 4.2 processing $(S_j^m,)_p$ and transport $(,T_j^m)_t$ are, respectively, indicated by $p(j,m)$ and $t(j,m)$, where $1 \le j \le 4$ and $m = 1,2$. The bold lines in both figures indicate the critical cycle of the schedule. The numbers in Figure 4.2 are the weights on the vertices and the arcs. It can easily be seen that the critical cycle is of length 10.

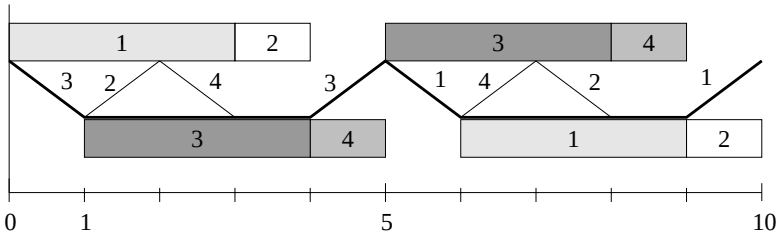


Figure 4.1. An optimal schedule S for problem instance $(\boldsymbol{p},\boldsymbol{d}) = ((\boldsymbol{3},\boldsymbol{1},\boldsymbol{3},\boldsymbol{1}),\boldsymbol{1})$.

If Assumption 2.3 is relaxed, then a feasible schedule does not necessarily have to contain a critical cycle any more (see Figure 4.3). However, notice that a schedule that does not have a critical cycle can always be improved. Therefore, an optimal schedule also contains a critical cycle if Assumption 2.3 is relaxed.

Now the following important observation can be made.

Observation 4.1. *The problem of finding an optimal schedule is equivalent with the problem of finding a critical graph with a critical cycle of minimum length.* □

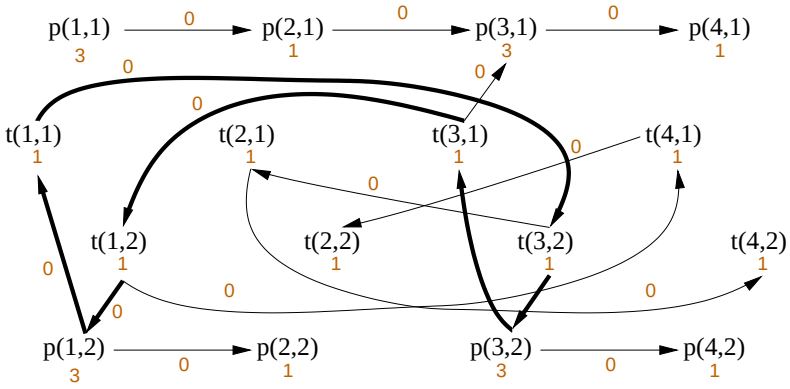


Figure 4.2. The critical graph $G(S)$.

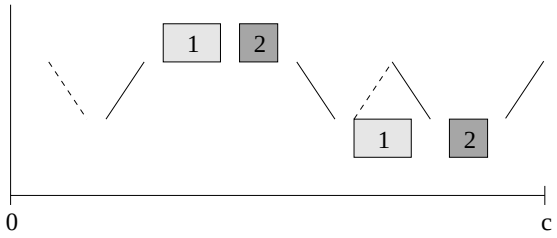


Figure 4.3. A schedule without a critical cycle.

Notice that a critical cycle can be seen as a sequence of processings and transports. A list of consecutive processings on a critical cycle with no transport in between them, we call a *block*. E.g., the critical cycle shown in the Figures 4.1 and 4.2 contains two blocks, namely a block consisting of $(S_1^2,)_p$ only and another block consisting of $(S_3^2,)_p$ only.

We make the following assumption.

Assumption 4.1. *Consider a schedule S. We only consider critical cycles C of S for which holds that there does not exist a critical cycle $\bar{C}$ of S for which holds that*

$$\{x | x \text{ is a processing of a block of } C\} \subsetneq \{x | x \text{ is a processing of a block of } \bar{C}\}.$$

□

This means that in Figure 4.4 only the solid critical cycle is considered, not the dashed one.

The sum of the processing times belonging to all the processings in a block B, we call the *length of block B*. Let $|B|$ denote be the number of processings in block B.

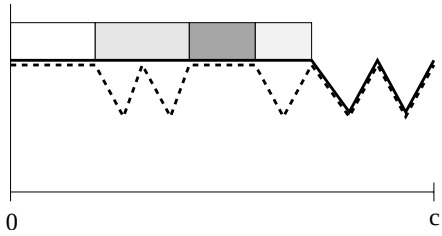


Figure 4.4. Only the solid critical cycle is considered, not the dashed one.

4.3 Some properties of critical cycles and blocks

In this section we look at some properties of critical cycles and their blocks. These properties will be used in the next chapter for proving that for any problem instance there always exists an optimal schedule with a critical cycle that contains at most two blocks.

In the first place, the following can be observed.

Observation 4.2. *Consider a block B on machine m. Let j_1 and j_2 respectively be the first and the last task of B (see the Figures 4.5 and 4.6). Then, $T^m_{j_1} = S^m_{j_1}$, $S^m_{j_2} + p_{j_2} + d = T^{m \oplus_2 1}_{j_2} + c(m-1)$ and $S^m_1 + c = S^m_N + p_N$. Furthermore, $S^m_j = S^m_{j-1} + p_{j-1}$ for $j = j_1+1, \ldots, j_2$, if $j_1 < j_2$ and $S^m_j = S^m_{j-1} + p_{j-1}$ for $j = j_1+1, \ldots, N, 2, 3, \ldots, j_2$), otherwise.*

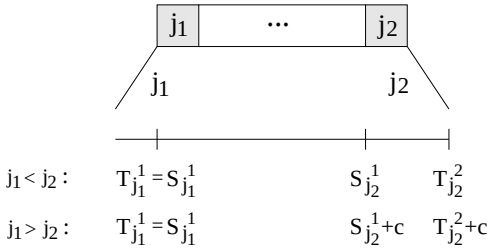


Figure 4.5. A block on machine 1.

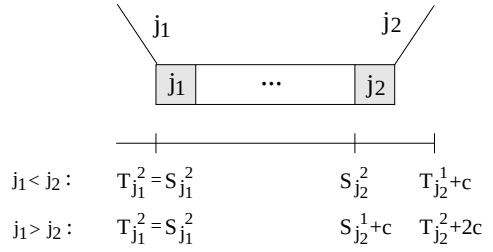


Figure 4.6. A block on machine 2.

□

Secondly, we have the following result.

Lemma 4.1. *The sum of the processing times of the first k tasks in a block B is greater than or equal to* $2kd (k = 1, \ldots, |B| - 1)$.

Proof. Consider a block B on machine m. Let $k < |B|$. Without loss of generality, we assume that the first task of B is task 1. With Formula (2.13) and Observation 4.2 we see that

$$\sum_{j=1}^{k} p_j = S_{k+1}^m - S_1^m \geq T_{k+1}^m - T_1^m \geq 2kd.$$

□

Due to symmetry considerations, also the following holds.

Lemma 4.2. *The sum of the processing times of the last k tasks in a block B is greater than or equal to* $2kd (k = 1, \ldots, |B| - 1)$. □

Since the previous two lemma's imply that for a block B with $|B| > 1$ we have that $\sum_{j=1}^{|B|-1} p_j \geq 2d(|B| - 1)$ and that $p_{|B|} \geq 2d$, we have the following result.

Corollary 4.1. *The length of a block B, with* $|B| > 1$*, is larger than or equal to* $2d|B|$. □

As a consequence, we have the following result.

Corollary 4.2. *A block of processing times on a critical cycle that is shorter than* $2d$ *contains only one task.* □

For convenience, we call blocks of length smaller than $2d$ *short blocks* and blocks of length at least $2d$ *long blocks*. The last corollary tells us that no critical cycles exist that contain a short block consisting of more than one processing. Below, we show that we do not need to consider critical cycles with short blocks at all.

Theorem 4.1. *If* $N \geq 3$*, then there always exists an optimal schedule with a critical cycle in which all blocks, if any, are long blocks.*

Proof. Consider a problem instance $(\boldsymbol{p}, d)$ where $N \geq 3$. If $p_j < 2d$ for all tasks $j, 1 \leq j \leq N$ then, in view of Lemma 3.6, there exists an optimal schedule with a critical cycle that consists of nonempty transports only. Hence, from now on we may assume that there exists at least one task that has a processing time larger than or equal to $2d$.
Consider for problem instance $(\boldsymbol{p}, d)$ an optimal schedule S with a critical cycle C that contains at least one short block. Without loss of generality, we assume that C contains a short block on machine 1.
Notice that since at least one of the tasks has a processing time of at least $2d$, there

exists a task j_1, such that $(S^1_{j_1},)_p$ is a short block of C and $(S^1_{j_1\oplus_N 1},)_p$ is no block of C or a long block of C. Without loss of generality, we assume that task j_1 is task 1. In other words, $(S^1_2,)_p$ is no short block of C.
In view of Observation 4.2, the path $P = [(\ ,T^1_1)_t,(S^1_1,\ )_p,(,T^2_1)_t]$ is part of C. We consider the following cases:

Case 1 In C, the path P is followed by a block B. In this case, $(S^2_1,)_p$ is the first processing of B. Suppose that the last processing of B is $(S^2_{j_2},)_p$ (see Figure 4.7). Consider the interval $[0,c]$. The processing of task 1 on machine

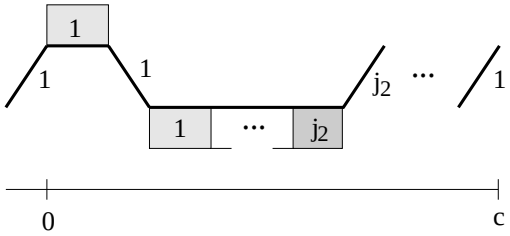


Figure 4.7. P is followed by a block.

1 that starts at time 0 is $(S^1_1,)_p$. This processing is followed by $(,T^2_1)_t$ and $(S^2_1,)_p$. Therefore, the processing of task j_2 on machine 2 in $[0,c]$ is $(S^2_{j_2},)_p$. In view of Observation 4.2, this processing is followed by $(,T^1_{j_2}+c)_t$. However, this implies that $T^1_{j_2}+c \le T^1_1+c$, which by Lemma 2.1 is only possible if $j_2 = 1$ and therefore $N = 1$. This, however, is in contradiction with the assumptions.

Case 2 In C, the path P is followed by an empty transport (see Figure 4.8). In this case, the robot arrives empty at machine 1 at time T^2_1+d. This implies that at that time machine 1 cannot start the processing of task 2. Therefore, and since at time T^2_1+d all tools are on machine 2, C must contain two consecutive empty transports. This, however, is in contradiction with Assumption 2.4.

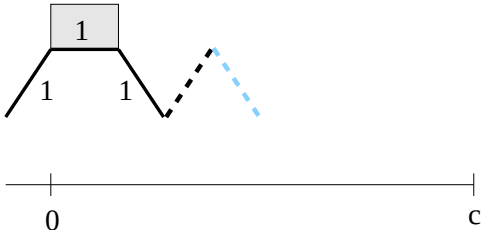


Figure 4.8. P is followed by an empty transport.

Case 3 In C, the path P is followed by a nonempty transport. In this case, C contains the path $[(,T^1_1)_t,(S^1_1,)_p,(,T^2_1)_t,(,T^1_2)_t]$ (see Figure 4.9).

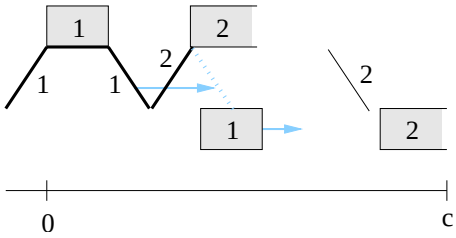


Figure 4.9. P is followed by a nonempty transport.

The processing $(S_2^1,)_p$ is no short block. In other words, either $(S_2^1,)_p$ is not contained in a block or it is the first processing of a long block. Notice that in any case $T_2^2 \geq S_1^1 + p_1 + 5d$. (If $(S_2^1,)_p$ is the first task of a long block, then, because of the definition of a long block and because of Lemma 4.1, we have that $p_2 \geq 2d$.) Hence, we can move T_1^2 and S_1^2 such that we obtain a new schedule $\bar{S}$ in which $\bar{T}_1^2 = S_1^1 + p_1 + 3d$ and $\bar{S}_1^2 = \max\{\bar{T}_1^2, S_N^2 + p_N - c\}$ (see the arrows in Figure 4.9). Notice that $\bar{S}$ is still optimal and that therefore it contains a critical cycle $\bar{C}$.

Let $G(S)$ and $G(\bar{S})$ respectively be the critical graphs belonging to the schedule S and $\bar{S}$. Observe that since $\bar{T}_1^2 > S_1^1 + p_1 + d$, there is no path $(,T_1^1)_t, (S_1^1,)_p, (,\bar{T}_1^2)_t$ in $G(\bar{S})$. Moreover, because $p_1 < 2d$, we have that either $\bar{T}_1^2 < \bar{S}_1^2$ or $\bar{S}_1^2 + p_1 + d < T_1^1 + c$. (If $\bar{T}_1^2 = \bar{S}_1^2$ and $\bar{S}_1^2 + p_1 + d = T_1^1 + c$, then this would imply that $c = 2p_1 + 4d < 6d$, which is impossible because of Lemma 3.1 and the fact that $N \geq 3$.) Hence, $(,T_1^2)_t, (\bar{S}_1^2,)_p, (,T_1^1 + c)_t$ is no path in $G(\bar{S})$ either. Consequently, $(S_1^1,)_p$ and $(\bar{S}_1^2,)_p$ cannot be short blocks of $\bar{C}$.

Recall that for obtaining $\bar{S}$ we did not move the processing and the transport belonging to tasks $j, j \neq 1$. Hence, if $G(S)$ did not contain the path $[(,T_j^m)_t, (S_j^m,)_p, (,T_j^{m \oplus_M 1} + c(m-1))_t$, then $G(\bar{S})$ does not contain this path either $(m = 1,2)$.

If $\bar{C}$ still contains a short block B, then we can renumber the tasks such that B concerns task 1 and repeat the above procedure. (Notice that if B is a block on machine 2 then a small adaptation of the procedure is necessary. Then, we have to change the transport of tool 1 to machine 1 and the processing of tool 1 on machine 1: $\bar{T}_1^1 = S_1^2 + p_1 + 3d - c$ and $\bar{S}_1^1 = \max\{\bar{T}_1^1, S_N^1 + p_N - c\}$.) Every time that the procedure is repeated, the number of tasks j for which $(,T_j^1)_t, (S_j^1,)_p, (,T_j^2)_t$ and $(,T_j^2)_t, (S_j^2,)_p, (,T_j^1 + c)_t$ are no paths in the critical graph is increased by one. Hence, the above procedure can be repeated until a schedule is obtained with a critical cycle with no short blocks.

□

5

Different structures of critical cycles

5.1 Introduction

In this chapter we examine the structure of critical cycles, in particular of critical cycles that belong to optimal schedules. We do this in order to get more insight in the structure of optimal schedules.
In the previous chapter, we saw that a critical cycle can be seen as a sequence of processings and transports, and that a critical cycle may contain blocks. In this chapter we address the following questions:

(a) Does a critical cycle belonging to an optimal schedule always contain both processings as well as transports?

(b) Does a critical cycle of an optimal schedule always contain blocks?

(c) How many blocks can a critical cycle belonging to an optimal schedule actually have?

We will see that a critical cycle of an optimal schedule may consist of processings only, and also of transports only. Hence, such a critical cycle does not necessarily contain any blocks. We do not answer the last question completely, but we show that it is not necessary to consider critical cycles with more than two blocks, since for any problem instance there always exists an optimal schedule with a critical cycle that contains at most two blocks.

5.2 Critical cycles consisting of transports only

At first, we show that optimal schedules exist that contain a critical cycle that consists of transports only and, consequently, does not contain any blocks. An example of such an optimal schedule is shown in Figure 5.1. The bold line in the figure indicates the critical cycle.

Notice that, since all the transports in Figure 5.1 are nonempty, for this schedule lower bound $2Nd$ is attained. In Figure 5.2 an optimal schedule is drawn with a critical cycle that also consists of transports only, but that also contains some empty ones.

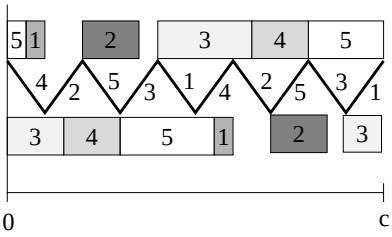


Figure 5.1. A critical cycle that consists of transports only.

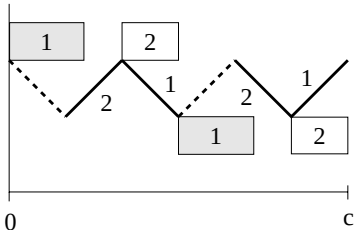


Figure 5.2. A critical cycle that consists of empty and nonempty transports.

5.3 Critical cycles consisting of processings only

A critical cycle of a schedule may consist of processings only. In this case, the length of the cycle is equal to the lower bound $\sum_{i=1}^{N} p_i$ and, consequently, the corresponding solution is optimal. An example of such a critical cycle is shown in Figure 5.3.

5.4 Critical cycles that consist of one block and some transports

It is also possible that the critical cycle of an optimal schedule consists of one block and some transports. Examples of such critical cycles are shown in the Figures 5.4 and 5.5.

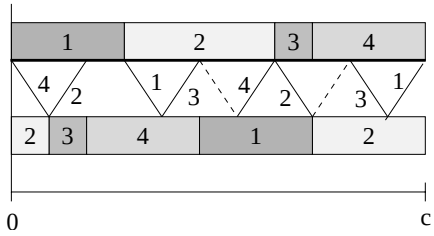


Figure 5.3. A critical cycle that consists of one block and no transports.

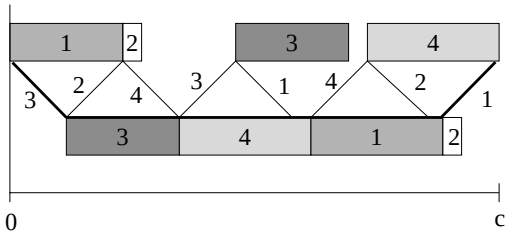


Figure 5.4. A critical cycle of an optimal schedule that consists of one block and two transports.

5.5 Critical cycles with two blocks

For the case that the critical cycle of an optimal schedule contains exactly two blocks, we distinguish two different cases.

5.5.1 Critical cycles with two blocks on the same machine

Figure 5.6 shows an example of an optimal schedule that has a critical cycle that contains two blocks on the same machine.

5.5.2 Critical cycles with two blocks on different machines

Another possibility is that the blocks are distributed over both machines. An example of such a critical cycle is shown in Figure 5.7.

In this case, more information is known about the structure of the critical cycle.

Lemma 5.1. *If a critical cycle contains exactly one block on machine 1 and exactly one block on machine 2, then:*

(a) both of the blocks only contain one task,

(b) both blocks contain the same task,

(c) both blocks have length $p_{\max}$*,*

(d) between the block on machine m and the block on machine $m \oplus_2 1$ *exactly one transport occurs* $(m = 1, 2)$ *and, consequently,*

(e) lower bound $2p_{\max} + 2d$ *is attained.*

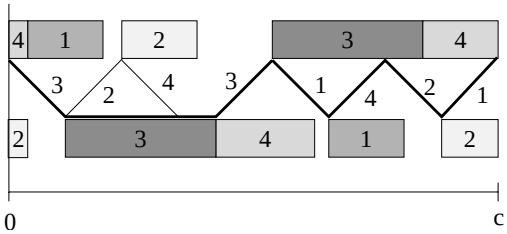


Figure 5.5. A critical cycle of an optimal schedule that consists of one block and more than two transports.

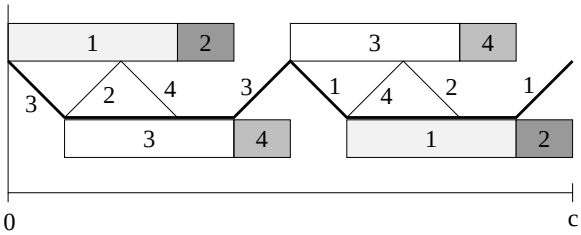


Figure 5.6. The critical cycle consists of two blocks on the same machine.

Proof. Consider a critical cycle that contains two blocks on different machines. Let j_1 and j_2, respectively, be the tasks that constitute the first and the last task of the block on machine 1 and let j_3 and j_4, respectively, be the first and the last task of the block on machine 2 (see Figure 5.8). Without loss of generality, we assume that $j_1 = 1$.
Consider the interval $[T_1^1, T_1^1 + c) = [0, c)$. Obviously, $(S_1^1,)_p$ is the processing of task 1 on machine 1 that occurs in this interval (and not the one that occurs in [c,2c)). Moreover, since $j_2 \geq 1$, the processing of task j_2 on machine 1 in $[0, c)$ is $(S_{j_2}^1,)_p$. Therefore, the transport of task j_2 to machine 2 in $[0, c)$ is $(, T_{j_2}^2)_t$. Notice that all tools $j, 1 \leq j \leq j_2$, are transported to machine 2 in the interval $[T_1^2, T_{j_2}^2]$.

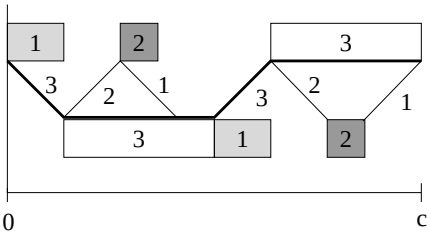


Figure 5.7. A critical cycle that contains two blocks on different machines.

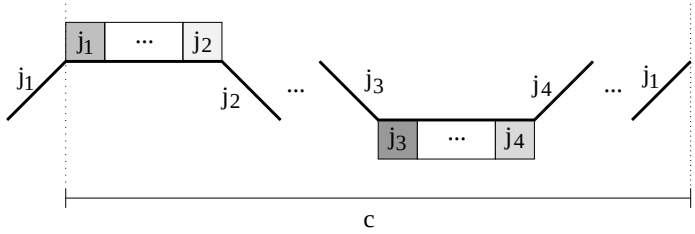


Figure 5.8. The structure of a critical cycle with two blocks on different machines.

Hence, since the transport of tool j_3 to machine 2 do not occur in this interval, we have that $j_3 > j_2$. This implies that the transport of tool j_3 to machine 2 is $(,T^2_{j_3})_t$. The transport of tool j_4 to machine 2 in $[0,c)$ occurs between $(,T^2_{j_3})_t$ and $(,T^2_1+c)_t$. Therefore, it is $(,T^2_{j_4})_t$ and, consequently, $j_4 \geq j_3$. This implies that the processing of task j_4 on machine 2 in $[0,c)$ is $(S^2_{j_4},)_p$ and that the transport of task j_4 to machine 1 in $[0,c)$ is $(,T^1_{j_4}+c)_t$. This, however, implies that $T^1_{j_4}+c \leq T^1_1+c$. Notice that this is only possible, if $j_4 = 1$. Moreover, since $1 \leq j_2 \leq j_3 \leq j_4$, this implies that the blocks in the critical cycle both consist of a processing of task 1 only and that the critical cycle is of length $2p_1+2d$. The latter, in view of Lemma 3.2, implies that $p_1 = p_{\max}$. □

5.6 Critical cycles with three or more blocks

A critical cycle belonging to an optimal schedule may contain more than two blocks. In this case, however, all blocks need to be on the same machine.

Lemma 5.2. *If a critical cycle contains more than two blocks, then these blocks are all on the same machine.*

Proof. Consider a critical cycle C that contains at least three blocks and suppose that these blocks are not all on the same machine. Then C contains a path P with two consecutive blocks, the first of which is on machine 1 and the second of which is on machine 2. In the same way as in the proof of Theorem 5.1 we can show that both blocks of P consist of one task only and that they both contain the same task. But this would imply that C contains only two blocks, which is a contradiction. We conclude that the blocks all have to be on the same machine. □

Hence, optimal schedules with critical cycles that contain more than two blocks distributed over both machines do not exist. Moreover, it is shown below, that critical cycles of optimal schedules that contain more than two blocks on the same machine are not relevant.

Theorem 5.1. *For any problem instance, there exists an optimal schedule with a critical cycle with at most two blocks.*

Proof. Consider an optimal schedule S of length c with a critical cycle C that contains $b, b \geq 3$ blocks. In view of Lemma 5.2, we know that all these blocks have to be on the same machine. Without loss of generality, we assume that the blocks are on machine 1. We will show that it is always possible to change the schedule S into another optimal schedule $\bar{S}$ that has a critical cycle with at most two blocks.

We number the blocks of C in order $1, 2, 3, \ldots, b$ in such a way that either $(S^1_{j\max},)_p$ is in block 1 or $(S^1_{j\max},)_p$ is situated between the last block b and block 1 (see the Figures 5.9 and 5.10). Without loss of generality, we assume that task 1 is the first task of block 1.

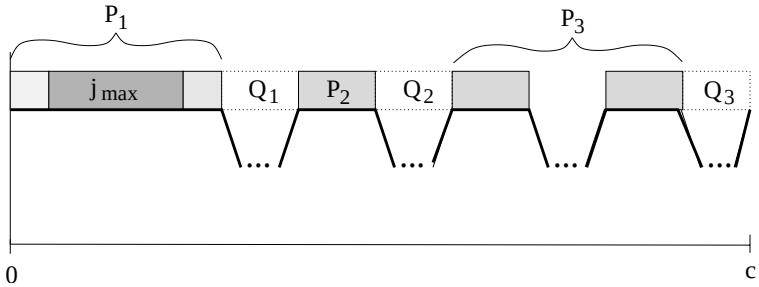


Figure 5.9. Processing $(S^1_{j\max},)_p$ is in a block.

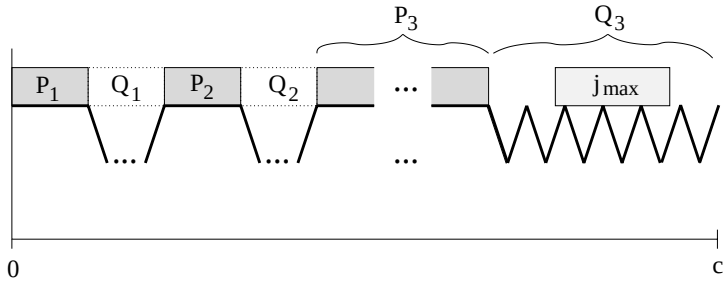


Figure 5.10. Processing $(S^1_{j\max},)_p$ is not in a block.

Consider the ordered set of tasks and suppose that we divide this set into the following ordered subsets (see the Figures 5.9 and 5.10):

P_a : the tasks whose processing is in block $a (a = 1, 2)$.
Q_a : the tasks whose processing is between the blocks a and $a \oplus_3 1 (a = 1, 2, 3)$.
P_3 : all remaining tasks.

Hence, we see that

$$j_{\max} \in P_1 \cup Q_3. \tag{5.1}$$

Let $b(P_a)^1$ and $e(P_a)^1$, respectively, denote the times at which in schedule S the processing of the first task of P_a on machine 1 begins and the processing of the last task of P_a on machine 1 ends $(a = 1,2,3)$. We define

$$\begin{array}{lll} l(P_a) := e(P_a)^1 - b(P_a)^1, & \textit{the length of } P_a & (a = 1,2,3), \\ l(Q_a) := b(P_{a\oplus_3 1})^1 + \delta_{a3}c - e(P_a)^1 & \textit{the length of } Q_a & (a = 1,2,3), \end{array}$$

In view of Theorem 4.1, we assume that all blocks are long blocks: we have that

$$l(P_a) \geq 2d \qquad (a = 1,2,3). \tag{5.2}$$

Furthermore, since in a critical cycle two blocks on the same machine are always connected by two or more transports, we have

$$l(Q_a) \geq 2d \qquad (a = 1,2,3). \tag{5.3}$$

Obviously,

$$\sum_{a=1}^{3} l(P_a) + \sum_{a=1}^{3} l(Q_a) = c. \tag{5.4}$$

Depending on the problem instance, we distinguish the following cases:

Case 1 There is a task that has a very large processing time ($p_{\max} \geq \frac{1}{2}c - 3d$).

(a) The processing of task $j_{\max}$ on machine 1 is in a block.

(b) The processing of task $j_{\max}$ on machine 1 is not in a block.

Case 2 There is no task with a very large processing time ($p_{\max} < \frac{1}{2}c - 3d$).

We consider the above cases one by one. For all of the cases, we first give an algorithm for changing the schedule S, after which we proof that the algorithm indeed results in an optimal schedule with the stated property.

Proof for Case 1(a)

Let

$$p_{\max} \geq \frac{1}{2}c - 3d \tag{5.5}$$

and let $(S^1_{j_{\max}},)_p$ be contained in a block.

We first explain how schedule $\bar{S}$ is obtained, after which we show that $\bar{S}$ is feasible and optimal, and that it contains a critical cycle with at most two blocks.

Let U and V, respectively, be the sets of tasks of P_1 that are processed before and after task $j_{\max}$(see Figure 5.11).

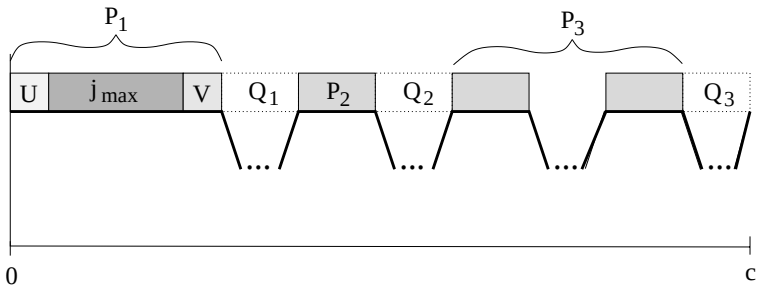


Figure 5.11. The structure of critical cycle C.

Sketch of the algorithm

We first describe in broad outlines how schedule S is transformed into schedule $\bar{S}$.

Step 1 We start by changing in S the schedule of the tasks on machine 2: we copy the schedule of the tasks on machine 1 onto machine 2, after which we move the resulting schedule *clockwise* – to the right – as much as possible without creating any conflicts with the transports to machine 1. (See Figure 5.12(a) and (b) for an example.)

(Remark: the schedule in Figure 5.12(a) is not an optimal schedule. It is chosen to explain the different Steps of the algorithm clearly.)

Step 2 On machine 2, we move the tasks of Q_3 clockwise and the tasks of Q_1 counterclockwise until machine 2 is not idle any more between the processing of the first task of Q_3 and the processing of the last task of Q_1 (see Figure 5.12(c)).

Notice that after the Steps 1 and 2 no conflicts among processings on the same machine exist. Furthermore, no transports overlap. The only infeasibilities that may occur, result from transports that arrive too late at machine 2 or start before the corresponding tool becomes available on machine 2.

Step 3 We change the transports of the tools $j \in V \cup Q_1$ to machine 1 in the following way. If j is transported to machine 2 a number of t time units after $S^1_{j_{\max}}$, then we transport the tool to machine 1 a number of t time units after $S^2_{j_{\max}} - c$ (see Figure 5.12(d)).

Later on we will show that now all transports of tools of $V \cup Q_1$ to machine 1 finish in the interval $I^* = [T^1_{j_{\max}} + d, T^2_{j_{\max}} - d]$. Notice that now overlapping transports may exist and that all transports to machine 1 that finish in I^* belong to tools of tasks in $V \cup Q_1$.

We now create some transports to machine 2 that finish in I^*.

Step 4 In a similar way as in Step 3, we change the transports of the tools j of $Q_3 \cup U$ to machine 2. If j arrives at machine 1 a number of t time units after $S^1_{j_{\max}}$, then we transport the tool to machine 2 a number of $t - 4d$ time units after $S^2_{j_{\max}}$ (see Figure 5.12(e)).

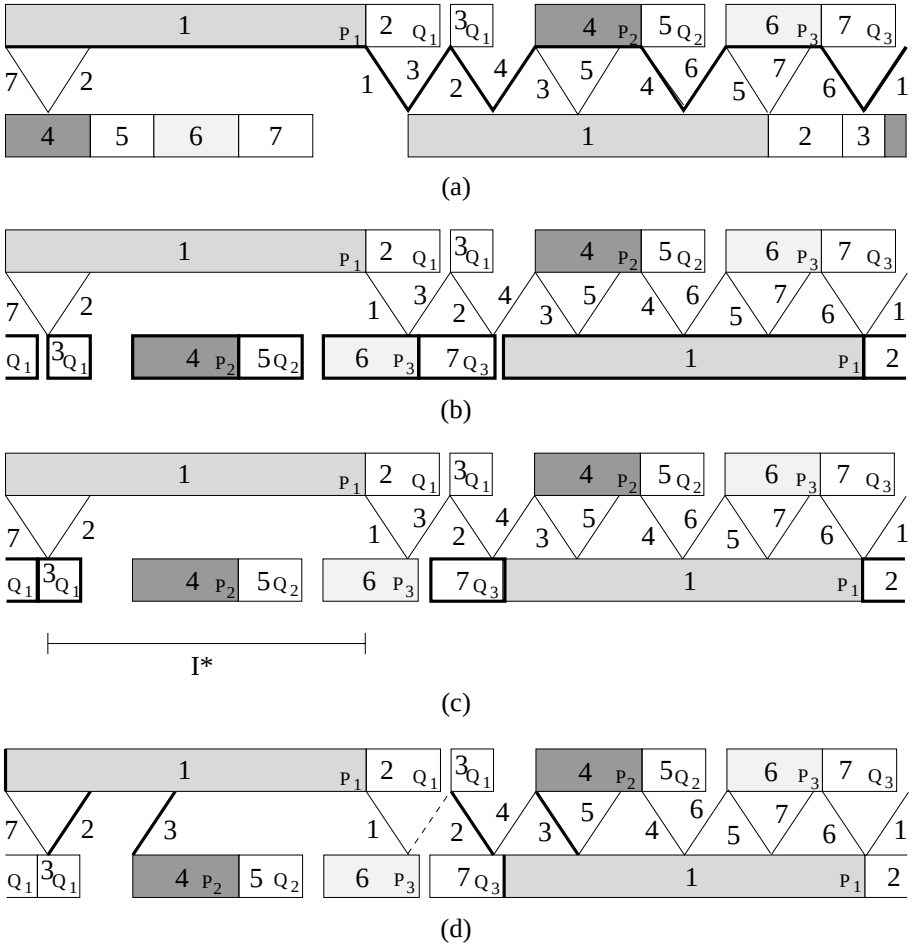


Figure 5.12. (Part 1.) An example of a schedule (a) before Step 1, (b) after Step 1, (c) after Step 2, (d) after Step 3.

Later, we will prove that now all transports of $U \cup Q_3$ to machine 2 finish in I^*. (More precisely, $T^2_j \in I^*$ for $j \in U$ and $T^2_j - c \in I^*$ for $j \in Q_3$.) Notice that all transports to machine 2 that finish in I^* belong to tools from $Q_3 \cup U$. Consequently,

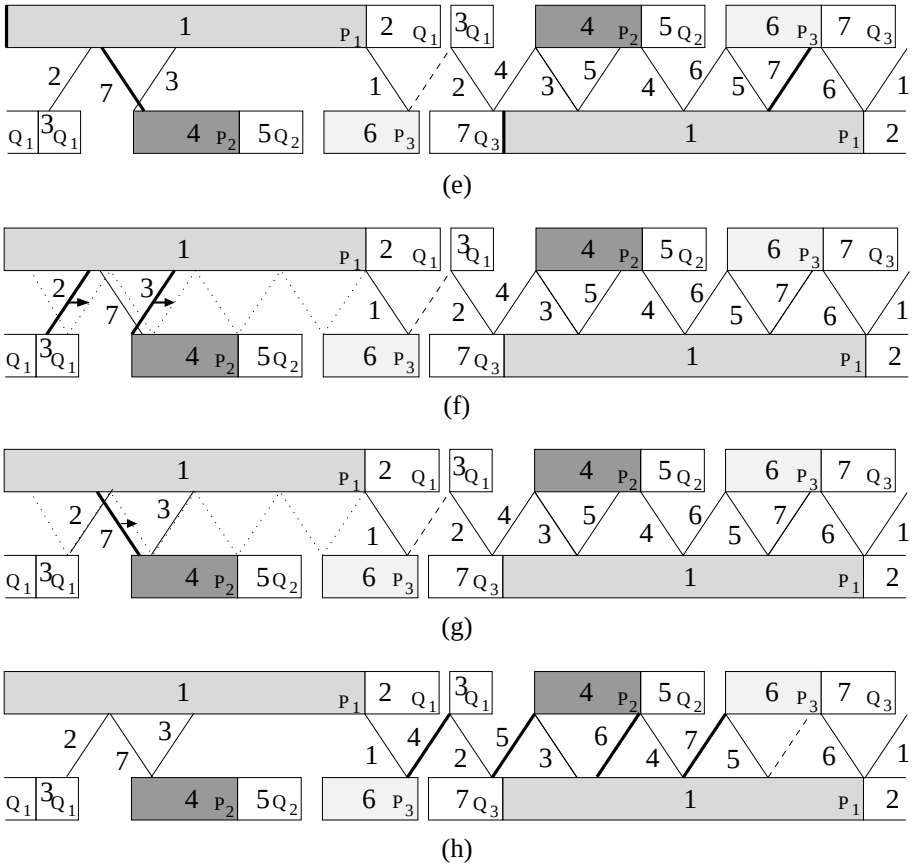


Figure 5.12. (Part 2.) (e) after Step 4, (f) Step 5, (g) Step 6, (h) After Step 7.

transports that we did not change in the Steps 3 and 4 all finish completely outside the interval I^*. Hence, at this moment there do not exist any conflicts among the transports that finish outside the interval I^*.

However, as can be seen in Figure 5.12(e), it is possible that conflicts exist between the transports to machine 1 that finish in interval I^* and the transports to machine 2 that end in the interval I^*. Therefore, we carry out the following step.

Step 5 We fill the interval I^* with as many empty "candidate" transports as possible and we move these transports as much clockwise as possible (see Figure 5.12(f)). Then, we move each transport of a tool of $Q_1 \cup V$ to machine 1 clockwise until it coincides for the first time with one of the "candidate" transports to machine 1 (see Figure 5.12(f)).

Step 6 We do the same with the tools of $U \cup Q_3$ to machine 2: we move them one by one clockwise until they coincide for the first time with one of the "candidate" transports to machine 2 (see Figure 5.12(g)).

Although it may not be clear at first sight, all the transports that before the Steps 5 and 6 finished completely in I^*, still end in I^* after these steps. This will be proved later.

Our aim is to create a schedule with a critical cycle with less than three blocks. In view of Observation 4.2, we can reduce the number of processings that could be first processings of a block by doing the following.

Step 7 We move each transport to machine 1 that does not concern a tool of a task of $\{j_{\max}\} \cup V \cup Q_1$ counterclockwise by $2d$(see Figure 5.12(h)). If this causes a conflict with a transport to machine 2 then we move it counterclockwise until the conflict disappears.

The last step can be carried out because before Step 7 the transport of the first tool of P_2 to machine 1 finishes at least $2d$ after the end of block P_1 and the transport to machine 1 of the tool corresponding to the last task of Q_1 finishes at least $2d$ before the end of block P_1. (The former is obviously true and the latter will be shown later.)

Call the schedule that is created in this way $\bar{S}$. As will be proved later, $\bar{S}$ is an optimal schedule. Therefore, it contains a critical cycle. Because of Step 7, only tasks of $\{j_{\max}\} \cup V \cup Q_1$ can be first processings of a block on machine 1. Later on we will see that only task $j_{\max}$ can be the first processing of a block on machine 1. Therefore a critical cycle of $\bar{S}$ contains at most one block on machine 1. We will also show that a critical cycle of $\bar{S}$ can contain at most one block on machine 2. Hence, any critical cycle of $\bar{S}$ contains at most two blocks.

Notice that for $\bar{S}$ Assumption 2.3 and 2.4 are not necessarily valid. However, we can change $\bar{S}$ into a schedule $\bar{\bar{S}}$ for which the assumptions hold by:

1. iteratively removing pairs of subsequent empty transports, until no such pair exists any more,
2. moving the processings and transports counterclockwise as explained in Section 2.4.

$\bar{\bar{S}}$ is still an optimal schedule and therefore it still contains a critical cycle $\bar{\bar{C}}$. Notice that $\bar{\bar{C}}$ also has to be a critical cycle of $\bar{S}$ and that therefore it contains at most two blocks. ($\bar{S}$ may contain more critical cycles than $\bar{\bar{C}}$. In this case these cycles contain a pair of subsequent empty transports.)

Intermezzo

Before we look at the more technical details of the algorithm, we first give some definitions and a lemma that will be helpful in the remaining part of this proof.

Definition 5.1. Consider an interval $I = [I_b, I_e]$. The set of transports $\{(,r^1)_t | r = I_e - 2kd, r \geq I_b + 2d, k \in \mathbb{N} \cup \{0\}\} \cup \{(,r^2)_t | r = I_e - 2kd + d, r \geq I_b + d, k \in \mathbb{N}\}$ we call a *counterclockwise zigzag robot movement* on I (ccwZZ(I)) (see Figure 5.13 for an example). □

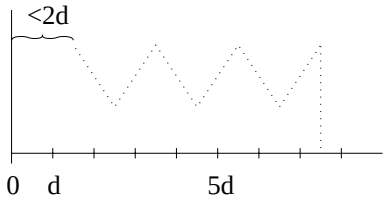


Figure 5.13. ccwZZ$[0, 7.5d]$.

Definition 5.2. Consider an interval $I = [I_b, I_e]$ and a transport $(,r^m)_t$. We say that we *project* $(,r^m)_t$ *clockwise on ccwZZ(I)* if we shift it clockwise until it coincides for the first time with a robot movement to machine m of the ccwZZ(I) (see Figure 5.14 for some examples). In order to assure the existence of this projection, we assume that $r \leq I_e$ if $m = 1$ and $r \leq I_e - d$ if $m = 2$. The resulting transport we call the *clockwise projection of* $(,r^m)_t$ *on ccwZZ(I)*. □

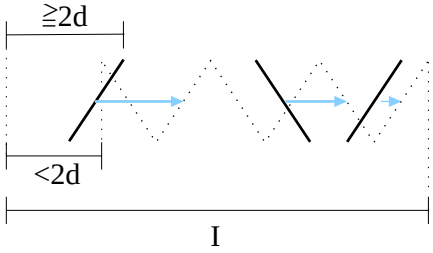


Figure 5.14. The solid transports are projected clockwise on the dotted ccwZZ(I).

Definition 5.3. Consider an interval $I = [I_b, I_e]$ and a set of transports $TP = \{(,r_i^{m_i})_t | m_i \in \{1,2\}; i = 1,2,\ldots,k\}$ where $r_i \geq I_b + 2d$ if $m_i = 1$ and $r_i \geq I_b + d$ if $m_i = 2$. The set of projections of the transports of TP on ccwZZ(I) is called the *clockwise projection of TP on ccwZZ(I)* (see Figure 5.14 for an example). □

In the above definition, the restrictions on the transports of TP ($r_i \geq I_b + 2d$ or $r_i \geq I_b + d$) are necessary to assure that all projections of a feasible set of transports are different. This can be seen by the following example. Consider

ccwZZ($[0,3.5d]$) and suppose that $TP = \{(,0.5d^2)_t,(,2.5d^2)_t\}$. Notice that both transports of TP have the same projection on ccwZZ($[0,3.5d]$) (see Figure 5.15).

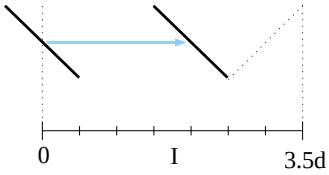


Figure 5.15. Two transports with the same projection.

The following important observation can be made.

Observation 5.1. *During a projection a transport is shifted by less than* $2d$. □

The following lemma gives more information about the result of projecting a transport on a ccwZZ.

Lemma 5.3. *Consider a ccwZZ($[I_b,I_e]$) and a transport* $(,r^m)_t$ *such that* $r^m \in [J_b,J_e]$, $m \in \{1,2\}$. *Let* $(,\bar{r}^m)_t$ *be the clockwise projection of* $(,r^m)_t$ *on the ccwZZ.*

(a) *If* $m = 1, I_b \leq J_b - 2d$ *and* $I_e \geq J_e$, *then* $\bar{r} \in [J_b, J_e + 2d)$.

(b) *If* $m = 2, I_b \leq J_b - d$ *and* $I_e \geq J_e + d$, *then* $\bar{r} \in [J_b, J_e + 2d)$.

(c) *If* $m = 1, I_b \leq J_b - 2d$ *and* $I_e = J_e + 2kd (k \in \mathbb{N} \cup \{0\})$, *then* $\bar{r} \in [J_b, J_e]$ *(see Figure 5.16 for some examples).*

(d) *If* $m = 2, I_b \leq J_b - d$ *and* $I_e = J_e + 2kd + d (k \in \mathbb{N} \cup \{0\})$, *then* $\bar{r} \in [J_b, J_e]$ *(see Figure 5.17 for some examples).*

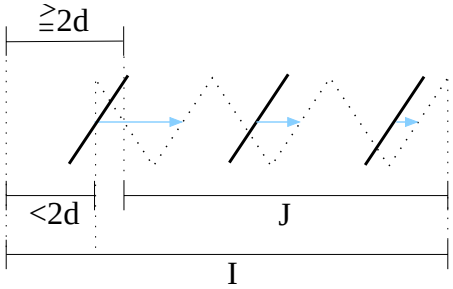


Figure 5.16. $I_e = J_e + 2kd (k \in \mathbb{N} \cup 0)$.

Proof.

(a) Follows directly from Observation 5.1.

(b) Idem.

(c) Obviously, $\bar{r} \geq r \geq J_b$. Moreover, there exists an $l \in \mathbb{N} \cup \{0\}, l \geq k$ such that $I_e - 2(l+1)d < r \leq I_e - 2ld$ and $\bar{r} = I_e - 2ld = J_e + 2d(k-l) \leq J_e$.

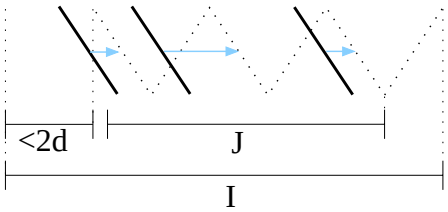


Figure 5.17. $I_e = J_e + 2kd + d(k \in \mathbb{N} \cup \{0\})$.

(d) Obviously, $\bar{r} \geq r \geq J_b$. Moreover, there exists an $l \in \mathbb{N} \cup \{0\}, l \geq k$ such that $I_e - 2(l+1)d - d < r \leq I_e - 2ld - d$ and $\bar{r} = I_e - 2ld - d = J_e + 2d(k-l) \leq J_e$.

□

Technical description of the algorithm

For describing the different steps of the algorithm more precisely, we introduce the variables $S_j^m(s)$ and $T_j^m(s)$ which, respectively, denote the start times of the processings and the tool arrival times just after step s of the algorithm $(1 \leq j \leq N; m = 1,2; 1 \leq s \leq 7)$.

We define

$$l(U) := \sum_{j \in U} p_j \quad \textit{the length of } U \quad \text{and}$$

$$l(V) := \sum_{j \in V} p_j \quad \textit{the length of } V.$$

Clearly,

$$l(U) + p_{\max} + l(V) + \sum_{a=2}^{3} l(P_a) + \sum_{a=1}^{3} l(Q_a) = c. \tag{5.6}$$

Now let us have a more detailed look at the algorithm. Below, we describe the different steps one by one. We indicate how the variables S_j^m and T_j^m change during the application of the algorithm. We remark that if no explicit formula is given for a variable $S_j^m(i)$ or $T_j^m(i)$ then this means that the corresponding processing or transport is not changed in Step $i (1 \leq i \leq 7, 1 \leq j \leq N; m = 1,2)$.

Step 1 Copy the schedule of machine 1 on machine 2 and move the resulting schedule on machine 2 clockwise as much as possible without creating any conflicts with the transports to machine 1.

Because $p_{\max}$ is so large with respect to the cycle time, we can move the schedule

clockwise until

$$S^2_{j_{\max}}(1) + p_{\max} + d = T^1_{j_{\max}} + c. \tag{5.7}$$

Later on in this proof, it will be shown that this is possible without creating conflicts between the processings on machine 2 and the transports to machine 1.

Shifting the schedule on machine 2 clockwise any further would cause the processing of task $j_{\max}$ on machine 2 to finish too late with respect to the transport of tool $j_{\max}$ to machine 1.

Clearly,

$$S^2_j(1) \geq S^2_j \quad j \in \{j_{\max}\} \cup U. \tag{5.8}$$

Moreover, since now both machines carry out the same schedule, apart from a shift, we have that

$$S^2_j(1) = S^1_j + (S^2_{j_{\max}}(1) - S^1_{j_{\max}}) \text{ for all tasks } j. \tag{5.9}$$

Let $b(P_a)^2$ and $e(P_a)^2$ denote the times at which now, respectively, the processing of the first task of P_a on machine 2 begins and the processing of the last task of P_a on machine 2 ends.

Step 2 Shift the processings of the tasks of Q_3 clockwise and of the tasks of Q_1 counterclockwise until:

$$S^2_j(2) = \begin{cases} S^2_{j_{\max}}(1) + \sum_{i=j_{\max}}^{j-1} p_i & \text{for } j \in V \cup Q_1, \\ S^2_{j_{\max}}(1) - \sum_{i=j}^{j_{\max}-1} p_i & \text{for } j \in U, \\ S^2_{j_{\max}}(1) - \sum_{i=j}^{j_{\max}-1} p_i + c & \text{for } j \in Q_3. \end{cases} \tag{5.10}$$

Obviously,

$$S^2_j(2) \quad \geq \quad S^2_j(1) \quad j \in U \cup Q_3. \tag{5.11}$$

Step 3 For all $j \in V \cup Q_1$, we define

$$\begin{aligned} T^1_j(3) \quad &:= \quad T^2_j(2) + (S^2_{j_{\max}}(2) - S^1_{j_{\max}}(2)) - c \\ &= \quad T^2_j + S^2_{j_{\max}}(1) - S^1_{j_{\max}} - c. \end{aligned} \tag{5.12}$$

Claim 1 *After Step 3 we have that* $T_j^1 \in I^* = [T_{j_{\max}}^1 + d, T_{j_{\max}}^2 - d]$ *for all tasks* $j \in V \cup Q_1$.

Proof (Claim 1). Notice that for $j \in V$ we have:

$$T_j^2 \in [T_{j_{\max}}^2 + 2d, e(P_1)^1 + d] \overset{(2.15)}{\subseteq} [S_{j_{\max}}^1 + p_{\max} + 3d, S_{j_{\max}}^1 + p_{\max} + l(V) + d] \tag{5.13}$$

and for $j \in Q_1$:

$$\begin{aligned} T_j^2 \quad &\in \quad [T_{j_{\max}}^2 + 2d, e(P_2)^1 - d] \\ &\overset{(2.15)}{\subseteq} \quad [S_{j_{\max}}^1 + p_{\max} + 3d, S_{j_{\max}}^1 + p_{\max} + l(V) + l(Q_1) + l(P_2) - d] \end{aligned} \tag{5.14}$$

Hence, for $j \in V \cup Q_1$ we get

$$\begin{aligned} T_j^1(3) \quad &\overset{(5.12)}{=} \quad T_j^2 + S_{j_{\max}}^2(1) - S_{j_{\max}}^1 - c \\ &\overset{(5.13)(5.14)}{\geq} \quad S_{j_{\max}}^1 + p_{\max} + 3d + S_{j_{\max}}^2(1) - S_{j_{\max}}^1 - c \\ &\overset{(5.7)}{=} \quad T_{j_{\max}}^1 + 2d. \end{aligned}$$

Moreover, if $j \in V[Q_1]$ then

$$\begin{aligned} T_j^1(3) \quad &\overset{(5.12)}{=} \quad T_j^2 + S_{j_{\max}}^2(1) - S_{j_{\max}}^1 - c \\ &\overset{(5.13)(5.14)}{\leq} \quad S_{j_{\max}}^1 + p_{\max} + l(V) + d + S_{j_{\max}}^2(1) - S_{j_{\max}}^1 - c\,[+l(Q_1) + l(P_2) - 2d] \\ &\overset{(5.7)}{=} \quad l(V) + T_{j_{\max}}^1\,[+l(Q_1) + l(P_2) - 2d] \\ &\overset{(2.13)}{\leq} \quad S_{j_{\max}}^1 + l(V)\,[+l(Q_1) + l(P_2) - 2d] \\ &\overset{(5.6)}{=} \quad S_{j_{\max}}^1 + c - l(U) - p_{\max} - \sum_{a=2}^{3} l(P_a) - \sum_{a=1}^{3} l(Q_a)\,[+l(Q_1) + l(P_2) - 2d] \\ &\overset{l(U)\geq 0,(5.2)(5.3)}{\leq} \quad S_{j_{\max}}^1 + c - p_{\max} - 10d\,[+2d] \\ &\overset{(5.5)}{\leq} \quad S_{j_{\max}}^1 + p_{\max} - 4d\,[+2d]. \end{aligned}$$

Hence,

$$T_j^1(3) \in \begin{cases} [T_{j_{\max}}^1 + 2d, S_{j_{\max}}^1 + p_{\max} - 4d] & \text{for } j \in V, \\ [T_{j_{\max}}^1 + 2d, S_{j_{\max}}^1 + p_{\max} - 2d] & \text{for } j \in Q_1. \end{cases} \tag{5.15}$$

from which the claim directly follows. □(Claim 1)

Step 4 For all $j \in Q_3 \cup U$ define

$$\begin{aligned} T_j^2(4) &:= T_j^1(3) + (S^2_{j_{\max}}(3) - S^1_{j_{\max}}(3)) - 4d. \\ &= T_j^1 + S^2_{j_{\max}}(1) - S^1_{j_{\max}} - 4d. \end{aligned} \tag{5.16}$$

Claim 2 *After Step 4 we have that $T_j^2 \in I^*$ for all tasks $j \in U$ and $T_j^2 - c \in I^*$ for all tasks $j \in Q_3$.*

Proof (Claim 2). Observe that for $j \in Q_3$ we have:

$$T_j^1 \in [b(P_3)^1 + 2d, b(P_1)^1 + c - 2d] \subseteq [b(P_3)^1 + 2d, T^1_{j_{\max}} + c - 2d] \tag{5.17}$$

and for $j \in U$:

$$T_j^1 \in [b(P_1)^1, T^1_{j_{\max}} - 2d] \overset{(5.2),(5.3)}{\subseteq} [b(P_3)^1 - c + 4d, T^1_{j_{\max}} - 2d]. \tag{5.18}$$

For $j \in U[Q_3]$ we get:

$$\begin{aligned} T_j^2(4) &\overset{(5.16)}{=} T_j^1 + S^2_{j_{\max}}(1) - S^1_{j_{\max}} - 4d \\ &\overset{(5.17),(5.18)}{\leq} T^1_{j_{\max}} - 2d + S^2_{j_{\max}}(1) - S^1_{j_{\max}} - 4d\,[+c] \\ &\overset{(2.13)}{\leq} S^1_{j_{\max}} + S^2_{j_{\max}}(1) - S^1_{j_{\max}} - 6d\,[+c] \\ &\overset{(5.7)}{=} T^1_{j_{\max}} + c - p_{\max} - 7d\,[+c] \\ &\overset{(5.5)}{\leq} T^1_{j_{\max}} + p_{\max} - d\,[+c] \\ &\overset{(2.13)}{\leq} S^1_{j_{\max}} + p_{\max} - d\,[+c] \\ &\overset{(2.15)}{\leq} T^2_{j_{\max}} - 2d\,[+c] \end{aligned}$$

and

$$\begin{aligned} T_j^2(4) &\overset{(5.16)}{=} T_j^1 + S^2_{j_{\max}}(1) - S^1_{j_{\max}} - 4d \\ &\overset{(5.17),(5.18)}{\geq} b(P_3)^1 - c + S^2_{j_{\max}}(1) - S^1_{j_{\max}}\,[+c - 2d] \\ &= S^1_{j_{\max}} + p_{\max} + l(V) + l(Q_1) + l(P_2) + l(Q_2) - c \\ &\quad + S^2_{j_{\max}}(1) - S^1_{j_{\max}}\,[+c - 2d] \\ &\overset{(5.7)}{=} T^1_{j_{\max}} + l(V) + l(Q_1) + l(P_2) + l(Q_2) - d\,[+c - 2d] \\ &\overset{l(V)\geq 0,(5.2)(5.3)}{\geq} T^1_{j_{\max}} + 5d\,[+c - 2d] \end{aligned}$$

Hence,

$$T_j^2(4) \in \begin{cases} [T_{j_{\max}}^1 + c + 3d, T_{j_{\max}}^2 + c - 2d] & \text{for } j \in Q_3, \\ [T_{j_{\max}}^1 + 5d, T_{j_{\max}}^2 - 2d] & \text{for } j \in U, \end{cases} \tag{5.19}$$

from which the claim directly follows. □(Claim 2)

Step 5 Project the transports of $\{(\,, T_j^1)_t | j \in V \cup Q_1\}$ clockwise on ccwZZ($[T_{j_{\max}}^1, T_{j_{\max}}^2 - d]$).

Obviously, for all $j \in V \cup Q_1$ we have that

$$T_j^1(5) \geq T_j^1(3). \tag{5.20}$$

Claim 3 *After Step 5 we still have that $T_j^1 \in I^*$ for all tasks $j \in V \cup Q_1$.*

Proof (Claim 3). Consider the transports of tools of $V[Q_1]$ to machine 1. Because of (5.15) we can take $I_b = T_{j_{\max}}^1, I_e = T_{j_{\max}}^2 - d, J_b = T_{j_{\max}}^1 + 2d$ and $J_e = S_{j_{\max}}^1 + p_{\max} - 4d\,[+2d]$ in Lemma 5.3(a). Then $I_b \leq J_b - 2d$ and $J_e = S_{j_{\max}}^1 + p_{\max} - 4d\,[+2d] \overset{(2.15)}{\leq} T_{j_{\max}}^2 - 5d\,[+2d] \leq I_e$. Hence, after the projection we have for $j \in V[Q_1]$ that

$$T_j^1(5) \quad \in \quad [T_{j_{\max}}^1 + 2d, S_{j_{\max}}^1 + p_{\max} - 2d\,[+2d]), \tag{5.21}$$

from which it easily follows by (2.15) that

$$T_j^1(5) \in [T_{j_{\max}}^1 + 2d, T_{j_{\max}}^2 - 3d\,[+2d]) \subseteq I^*.$$

□(Claim 3)

Step 6 Project the transports of $\{T_j^2 | j \in U\} \cup \{T_j^2 - c | j \in Q_3\}$ clockwise on ccwZZ($[T_{j_{\max}}^1, T_{j_{\max}}^2 - d]$)

Claim 4 *After Step 6 we still have that $T_j^2 \in I^*$ for all tasks $j \in U$ and $T_j^2 - c \in I^*$ for all tasks $j \in Q_3$.*

Proof (Claim 4). Consider the transports of tools of U $[Q_3]$ to machine 2. We can take $I_b = T_{j_{\max}}^1, I_e = T_{j_{\max}}^2 - d, J_b = T_{j_{\max}}^1 + 3d$ and $J_e = T_{j_{\max}}^2 - 2d$ in Lemma 5.3(d) and thus after the projection for any task $j \in U$ $[Q_3]$ we have:

$$T_j^2(6)[-c] \quad \in \quad [T_{j_{\max}}^1 + 3d, T_{j_{\max}}^2 - 2d]. \tag{5.22}$$

□(Claim 4)

Moreover, since according to Observation 5.1 during the projection the transports are shifted by less than $2d$, we know that for $j \in U \cup Q_3$:

$$T_j^2(4) \leq T_j^2(6) < T_j^2(4) + 2d. \tag{5.23}$$

Claim 5 *The transport of the last task of Q_1 to machine 1 finishes at least $2d$ before the end of block P_1.*

Proof (Claim 5). Let $j \in Q_1$. If $l(V) > 0$, then

$$T_j^1(6) = T_j^1(5) \overset{(5.21)}{<} S_{j_{\max}}^1 + p_{\max} = e(P_1)^1 - l(V) \overset{\text{Lemma 4.2}}{\leq} e(P_a)^1 - 2d.$$

If $l(V) = 0$ then, since $T_j^1(6) = T_j^1(3)$ is on the ccwZZ(I^*), we have that $T_j^1(6) = e(P_1)^1 - 2kd$ for a certain $k \geq 0$. Since

$$T_j^1(6) = T_j^1(5) \overset{(5.21)}{<} S_{j_{\max}}^1 + p_{\max} = e(P_1)^1,$$

this k must at least equal to one. Consequently, $T_j^1(6) \leq e(P_1)^1 - 2d$. □(Claim 5)

Based on this claim, the following step can be made.

Step 7 Move all transports of $P_2 \cup Q_2 \cup P_3 \cup Q_3 \cup U$ to machine 1 counterclockwise by $2d$. If one of the resulting transports overlaps with a transport to machine 2, we move it further counterclockwise until the overlapping disappears.

Notice that after Step 7 the tool belonging to the first task of $P_a(a = 2,3)$ arrives at machine 1 at time $b(P_a)^1 - 2d$. Moreover, if $U \neq 0$, then the tool belonging to the first task of U arrives at machine 1 at time $b(P_1)^1 - 2d$. Consequently,

$$T_j^1(7) \geq b(P_1)^1 - 2d \qquad \text{for } j \in U, \tag{5.24}$$

$$T_j^1(7) \geq b(P_a)^1 - 2d \qquad \text{for } j \in P_a(a = 2,3). \tag{5.25}$$

Moreover, it can be seen easily that

$$T_j^1(7) \geq b(P_a)^1 \qquad \text{for } j \in Q_a(a = 2,3). \tag{5.26}$$

Feasibility of $\bar{S}$

We still have to show that the schedule $\bar{S}$ that we have obtained is a feasible schedule.

Notice that we have created $\bar{S}$ in such a way that there are no conflicts among the processings on machine $m(m = 1,2)$ and there do not exist conflicts among the transports. Hence, for showing that $\bar{S}$ is feasible we only have to examine whether the tools arrive in time at the machines and whether they do not leave from the

machines before they become available. In other words, for any task $j = 1, \ldots, N$ we need to check the following conditions:

1. $T_j^1(7) \leq S_j^1(7)$,
2. $S_j^1(7) + p_j + d \leq T_j^2(7)$,
3. $T_j^2(7) \leq S_j^2(7)$,
4. $S_j^2(7) + p_j + d \leq T_j^1(7) + c$.

Below, we check these conditions one by one.

Condition 1 Notice that the schedule on machine 1 has never been changed. Hence, $S_j^1(7) = S_j^1$ for any task j. The transport of tool $j_{\max}$ to machine 1 is not changed either. This implies that $T_{j_{\max}}^1(7) = T_{j_{\max}}^1$ and that Condition 1 still holds for task $j_{\max}$.

The transports to machine 1 of tools belonging to tasks of $P_2 \cup Q_2 \cup P_3 \cup Q_3 \cup U$ are only changed in Step 7. There they are moved counterclockwise. Hence, for a tool $j \in P_2 \cup Q_2 \cup P_3 \cup Q_3 \cup U$ we have that $T_j^1(7) < T_j^1$, which implies that also for these tasks Condition 1 is still valid.

We still have to consider the transports to machine 1 of tools of tasks $j \in V \cup Q_1$. These are only moved in the Steps 3 and 5. For any task $j \in V[Q_1]$ we have that

$$T_j^1(7) = T_j^1(5) \overset{(5.21)}{<} S_{j_{\max}}^1 + p_{\max} - 2d\,[+2d] \overset{j \in V[Q_1](2.14)}{\leq} S_j^1 = S_j^1(7).$$

Summarizing, we get

$$T_j^1(7) \begin{cases} < & S_j^1(7) \quad \text{for } j \neq j_{\max}, \\ \leq & S_j^1(7) \quad \text{for } j = j_{\max}. \end{cases} \tag{5.27}$$

Condition 2 We have already mentioned that $S_j^1(7) = S_j^1$ for all tasks j. Moreover, from the transports to machine 2, we have only moved the ones concerning tools belonging to tasks of $Q_3 \cup U$ (in the Steps 4 and 6). From this we can conclude immediately that for all tasks that are not in $Q_3 \cup U$ Condition 2 still holds.

For tasks $j \in U$ we have

$$\begin{aligned} S_j^1(7) + p_j + d &= S_j^1 + p_j + d \\ &\overset{(2.14), j \in U}{\leq} S_{j_{\max}}^1 + d \\ &\overset{l(V) \geq 0, (5.2)(5.3)}{\leq} S_{j_{\max}}^1 + l(V) + \sum_{a=2}^{3} l(P_a) + \sum_{a=1}^{3} l(Q_a) - 9d \\ &\overset{(5.6)}{=} S_{j_{\max}}^1 + c - l(U) - p_{\max} - 9d \end{aligned}$$

$$\begin{aligned}
&= b(P_1)^1 + c - p_{\max} - 9d \\
&\overset{(5.5)}{\leq} b(P_1)^1 + p_{\max} - 3d \\
&\overset{j\in U}{\leq} T_j^1 + p_{\max} - 3d \\
&\overset{(2.15)}{\leq} T_j^1 + T_{j_{\max}}^2 - S_{j_{\max}}^1 - 4d \\
&\overset{(2.13)}{\leq} T_j^1 + S_{j_{\max}}^2 - S_{j_{\max}}^1 - 4d \\
&\overset{(5.8)}{\leq} T_j^1 + S_{j_{\max}}^2(1) - S_{j_{\max}}^1 - 4d \\
&\overset{(5.16)}{=} T_j^2(4) \\
&\overset{(5.23)}{\leq} T_j^2(6) \\
&= T_j^2(7)
\end{aligned}$$

and for tasks $j \in Q_3$ we get

$$\begin{aligned}
S_j^1(7) + p_j + d &= S_j^1 + p_j + d \\
&\overset{(2.14),\, j\in Q_3}{\leq} b(P_1)^1 + c + d \\
&\leq T_{j_{\max}}^1 + c + d \\
&\overset{(5.22)}{<} T_j^2(6) \\
&= T_j^2(7).
\end{aligned}$$

Thus, Condition 2 is valid for any task j.

Condition 3. For checking Condition 3, we distinguish five cases.

(i) $j = j_{\max}$:

$$T_{j_{\max}}^2(7) = T_{j_{\max}}^2 \overset{(2.13)}{\leq} S_{j_{\max}}^2 \overset{(5.8)}{\leq} S_{j_{\max}}^2(1) = S_{j_{\max}}^2(7).$$

(ii) $j \in U \cup Q_3$:

$$\begin{aligned}
T_j^2(7) &= T_j^2(6) \\
&\overset{(5.23)}{<} T_j^2(4) + 2d \\
&\overset{(5.16)}{=} T_j^1 + S_{j_{\max}}^2(1) - S_{j_{\max}}^1 - 2d \\
&\overset{(2.13)}{<} S_j^1 + S_{j_{\max}}^2(1) - S_{j_{\max}}^1 \\
&\overset{(5.9)}{=} S_j^2(1) \\
&\overset{(5.11)}{\leq} S_j^2(2) \\
&= S_j^2(7).
\end{aligned}$$

(iii) $j \in V \cup Q_1$:

$$\begin{aligned}
T_j^2(7) &= T_j^2 \\
&\overset{(5.13)(5.14)(5.2)(5.3)}{\leq} S^1_{j_{\max}} + p_{\max} + l(V) + l(Q_1) + l(P_2) - d \\
&\overset{(2.13)(2.15)}{\leq} S^2_{j_{\max}} + l(V) + l(Q_1) + l(P_2) - 2d \\
&\overset{(5.8)}{\leq} S^2_{j_{\max}}(1) + l(V) + l(Q_1) + l(P_2) - 2d \\
&\overset{(5.6)}{=} S^2_{j_{\max}}(1) + c - l(U) - p_{\max} - l(Q_2) - l(P_3) - l(Q_3) - 2d \\
&\overset{l(U)\geq 0,(5.2)(5.3)}{\leq} S^2_{j_{\max}}(1) + c - p_{\max} - 8d \\
&\overset{(5.5)}{\leq} S^2_{j_{\max}}(1) + p_{\max} - 2d \\
&\overset{(5.10)}{<} S_j^2(2) \\
&= S_j^2(7).
\end{aligned}$$

(iv) $j \in P_a\,(a = 2,3)$:

$$\begin{aligned}
T_j^2(7) &= T_j^2 \\
&\overset{j\in P_a}{\leq} e(P_a)^1 + d \\
&= S^1_{j_{\max}} + p_{\max} + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i) + d \\
&\overset{(2.13)(2.15)}{\leq} S^2_{j_{\max}} + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i) \\
&\overset{(5.8)}{\leq} S^2_{j_{\max}}(1) + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i) \\
&\overset{(5.5)}{=} S^2_{j_{\max}}(1) + 2p_{\max} + 6d - c + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i) \\
&= b(P_a)^2 + p_{\max} + 6d - c + l(P_a) \\
&\overset{(5.6)}{=} b(P_a)^2 + 6d - l(U) - l(V) - \sum_{i=2}^{3} l(P_i) + l(P_a) - \sum_{a=1}^{3} l(Q_a) \\
&\overset{l(U),l(V)\geq 0,(5.2)(5.3)}{\leq} b(P_a)^2 - 2d \\
&\overset{j\in P_a}{<} S_j^2(1) \\
&= S_j^2(7).
\end{aligned}$$

(v) $j \in Q_2$:

$$\begin{aligned}
T_j^2(7) &= T_j^2 \\
&\overset{j\in Q_2}{\leq} e(P_3)^1 - d \\
&= S^1_{j_{\max}} + p_{\max} + l(V) + \sum_{i=1}^{2} l(Q_i) + \sum_{i=2}^{3} l(P_i) - d \\
&\overset{(2.13)(2.15)}{\leq} S^2_{j_{\max}} + l(V) + \sum_{i=1}^{2} l(Q_i) + \sum_{i=2}^{3} l(P_i) - 2d \\
&\overset{(5.8)}{\leq} S^2_{j_{\max}}(1) + l(V) + \sum_{i=1}^{2} l(Q_i) + \sum_{i=2}^{3} l(P_i) - 2d \\
&\overset{(5.5)}{\leq} S^2_{j_{\max}}(1) + 2p_{\max} + 4d - c + l(V) + \sum_{i=1}^{2} l(Q_i) + \sum_{i=2}^{3} l(P_i) \\
&= e(P_2)^2 + p_{\max} + 4d - c + l(Q_2) + l(P_3) \\
&\overset{(5.6)}{=} e(P_2)^2 + 4d - l(U) - l(V) - l(P_2) - l(Q_1) - l(Q_3) \\
&\overset{l(U),l(V)\geq 0,(5.2)(5.3)}{\leq} e(P_2)^2 - 2d \\
&\overset{j\in Q_2}{<} S_j^2(1) \\
&= S_j^2(7).
\end{aligned}$$

Summarizing,

$$T_j^2(7) \begin{cases} < S_j^2(7) & \text{for } j \neq j_{\max}, \\ \leq S_j^2(7) & \text{for } j = j_{\max}. \end{cases} \tag{5.28}$$

and, thus, Condition 3 is valid for any task j.

Condition 4. Notice that after Step 1 Condition 4 still holds for all tasks. In the Steps 2-7 $(S^2_{j_{\max}},)_p$ and $(,T^1_{j_{\max}})_t$ are not changed. Hence, in $\bar{S}$ Condition 4 still holds for task $j_{\max}$.

To check Condition 4 for the remaining tasks, we distinguish five cases.

(i) $j \in U$:

$$\begin{aligned}
S_j^2(7) + p_j + d &= S_j^2(1) + p_j + d \\
&\overset{j\in U}{\leq} S^2_{j_{\max}}(1) + d \\
&\overset{(5.7)}{=} T^1_{j_{\max}} + c - p_{\max} \\
&\overset{(2.13)}{\leq} S^1_{j_{\max}} + c - p_{\max}
\end{aligned}$$

$$\begin{aligned}
&= && b(P_1)^1 + l(U) + c - p_{\max} \\
&\overset{(5.5)}{\leq} && b(P_1)^1 + l(U) + p_{\max} + 6d \\
&\overset{(5.6)}{=} && b(P_1)^1 + c - l(V) - \sum_{a=2}^{3} l(P_a) - \sum_{a=1}^{3} l(Q_a) + 6d \\
&\overset{l(V)\geq 0,(5.2)(5.3)}{\leq} && b(P_1)^1 + c - 4d \\
&\overset{(5.24)}{<} && T_j^1(7) + c.
\end{aligned}$$

(ii) $j \in V \cup Q_1$:

Since S was a feasible schedule, we have that

$$T_j^2 \overset{(2.15)}{\geq} S_j^1 + p_j + d \overset{(2.14)}{\geq} S_{j_{\max}}^1 + \sum_{i=j_{\max}}^{j} p_i + d. \tag{5.29}$$

Hence,

$$\begin{aligned}
S_j^2(7) + p_j + d &= S_j^2(2) + p_j + d \\
&\overset{(5.10)}{=} S_{j_{\max}}^2(1) + \sum_{i=j_{\max}}^{j} p_i + d \\
&\leq S_{j_{\max}}^2(1) + T_j^2 - S_{j_{\max}}^1 \\
&\overset{(5.12)}{=} T_j^1(3) + c \\
&\overset{(5.20)}{\leq} T_j^1(5) + c \\
&= T_j^1(7) + c.
\end{aligned}$$

(iii) $j \in P_a (a = 2,3)$:

$$\begin{aligned}
S_j^2(7) + p_j + d &= S_j^2(1) + p_j + d \\
&\overset{j\in P_a}{\leq} e(P_a)^2 + d \\
&= S_{j_{\max}}^2(1) + p_{\max} + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i) + d \\
&\overset{(5.7)}{=} T_{j_{\max}}^1 + c + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i) \\
&\overset{(2.13)}{\leq} S_{j_{\max}}^1 + c + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i) \\
&\overset{(5.5)}{\leq} S_{j_{\max}}^1 + 2p_{\max} + 6d + l(V) + \sum_{i=1}^{a-1} l(Q_i) + \sum_{i=2}^{a} l(P_i)
\end{aligned}$$

$$\begin{aligned}
&= b(P_a)^1 + p_{\max} + 6d + l(P_a)\\
&\overset{(5.6)}{=} b(P_a)^1 + c - l(U) - l(V) - \sum_{i=2}^{3} l(P_i) + l(P_a) - \sum_{a=1}^{3} l(Q_a) + 6d\\
&\overset{l(U),l(V)\geq 0,(5.2)(5.3)}{\leq} b(P_a)^1 + c - 2d\\
&\overset{(5.25)}{\leq} T_j^1(7) + c.
\end{aligned}$$

(iv) $j \in Q_2$:

$$\begin{aligned}
S_j^2(7) + p_j + d &= S_j^2(1) + p_j + d\\
&\overset{j\in Q_2}{\leq} b(P_3)^2 + d\\
&= S_{j_{\max}}^2(1) + p_{\max} + l(V) + \sum_{a=1}^{2} l(Q_a) + l(P_2) + d\\
&\overset{(5.7)}{=} T_{j_{\max}}^1 + c + l(V) + \sum_{a=1}^{2} l(Q_a) + l(P_2)\\
&\overset{(2.13)}{\leq} S_{j_{\max}}^1 + c + l(V) + \sum_{a=1}^{2} l(Q_a) + l(P_2)\\
&\overset{(5.5)}{\leq} S_{j_{\max}}^1 + 2p_{\max} + 6d + l(V) + \sum_{a=1}^{2} l(Q_a) + l(P_2)\\
&= b(P_2)^1 + p_{\max} + 6d + l(P_2) + l(Q_2)\\
&\overset{(5.6)}{=} b(P_2)^1 + c - l(U) - l(V) - l(Q_1) - l(P_3) - l(Q_3) + 6d\\
&\overset{l(U),l(V)\geq 0,(5.2)(5.3)}{\leq} b(P_2)^1 + c\\
&\overset{(5.26)}{\leq} T_j^1(7) + c.
\end{aligned}$$

(v) $j \in Q_3$:

$$\begin{aligned}
S_j^2(7) + p_j + d &= S_j^2(2) + p_j + d\\
&\overset{(5.10)}{\leq} S_{j_{\max}}^2(1) - l(U) + c + d\\
&\overset{(5.7)}{=} T_{j_{\max}}^1 + 2c - p_{\max} - l(U)\\
&\overset{(2.13)}{=} S_{j_{\max}}^1 + 2c - p_{\max} - l(U)\\
&= b(P_3)^1 + c - p_{\max} + l(P_3) + l(Q_3)\\
&\overset{(5.5)}{\leq} b(P_3)^1 + p_{\max} + 6d + l(P_3) + l(Q_3)
\end{aligned}$$

$$\begin{aligned} & \overset{l(U),l(V)\geq 0,(5.2)(5.3)}{\leq} && b(P_3)^1+p_{\max}+l(U)+l(V)+\sum_{a=2}^{3} l(P_a)+\sum_{a=1}^{3} l(Q_a) \\ & \overset{(5.6)}{=} && b(P_3)^1+c \\ & \overset{(5.26)}{\leq} && T_j^1(7)+c. \end{aligned}$$

We see that Condition 4 holds for any task j.

Since the four conditions hold for any of the tasks j, we conclude that $\bar{S}$ is a feasible schedule.

Optimality of $\bar{S}$

Since S was an optimal schedule and $\bar{S}$ is a feasible schedule of the same length as S, $\bar{S}$ is also an optimal schedule.

A critical cycle of $\bar{S}$ contains at most two blocks

Schedule $\bar{S}$ is optimal and therefore, it contains a critical cycle.

For any task $j \neq j_{\max}$ we have that $T_j^1(7) \overset{(5.27)}{<} S_j^1(7)$. Hence, a block on machine 1 can only start with $(S^1_{j_{\max}},)_p$. This implies that any critical cycle of $\bar{S}$ can have at most one block on machine 1.

Moreover, we have seen that for any task $j \neq j_{\max}$ we have that $T_j^2(7) \overset{(5.28)}{<} S_j^2(7)$. Therefore, if a critical cycle of $\bar{S}$ contains a block on machine 2 then this must be a block that starts with $(S^2_{j_{\max}},)_p$. Hence, a critical cycle of $\bar{S}$ can have at most one block on machine 2. This implies that a critical cycle of $\bar{S}$ always contains at most two blocks.

(If for schedule $\bar{S}$ the Assumptions 2.3 and 2.4 do not hold, then, as explained before, $\bar{S}$ can be transformed in an optimal schedule $\bar{\bar{S}}$, which contains a critical cycle with at most two blocks and for which the assumptions hold.)

□(of Case 1(a))

Proof for Case 1(b)

Let

$$p_{\max} \geq \frac{1}{2}c - 3d \tag{5.30}$$

and let $(S^1_{j_{\max}},)_p$ **not** be contained in a block. We prove that this situation is not possible.

Let U and V, respectively, be the sets of tasks of Q_3 that are processed before and after task $j_{\max}$(see Figure 5.18).

Notice that since the robot is never idle between $e(P_3)^1$ and $b(P_1)^1+c$ there are at least $2\lceil \frac{p_{\max}}{2d} \rceil$ transports that finish in the interval $I^* = [T^1_{j_{\max}}+d, T^2_{j_{\max}}-d]$. Notice

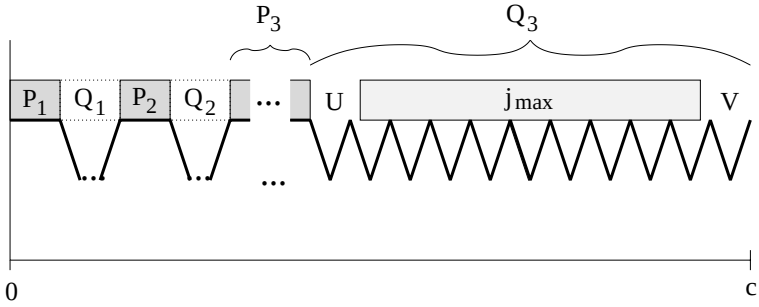


Figure 5.18. The tasks of U are processed before task $j_{\max}$ and the tasks of V after task $j_{\max}$.

that, in view of Assumption 2.4, there are no subsequent empty transports. This implies that in I^* there finish at least $\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil$ nonempty transports to machine 1 and $\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil$ nonempty transports to machine 2.

The $\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil$ nonempty transports to machine 1 that end in I^* concern tools of $V \cup P_1$. Hence, $V \cup P_1$ contains at least $\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil$ tasks. The transports of these tools to machine 2 all finish in the interval $[T^2_{j_{\max}} + 2d, e(P_1)^1 + c + d]$. Hence, the length of the interval $[T^2_{j_{\max}} + 2d, e(P_1)^1 + c + d]$ is at least equal to $2d(\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil - 1)$, which implies that

$$\text{the length of the interval } [T^2_{j_{\max}}, e(P_1)^1 + c] \text{ is at least } 2d(\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil) - d. \tag{5.31}$$

The $\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil$ nonempty transports to machine 2 that finish in I^* concern tools of $P_3 \cup U$. Hence, $P_3 \cup U$ contains at least $\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil$ tasks. The transports of these tools to machine 1 all finish in the interval $[b(P_3)^1, T^1_{j_{\max}} - 2d]$. Hence, the length of the interval $[b(P_3)^1, T^1_{j_{\max}} - 2d]$ is at least equal to $2d(\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil - 1)$, which implies that

$$\text{the length of the interval } [b(P_3)^1, T^1_{j_{\max}}] \text{ is at least } 2d(\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil). \tag{5.32}$$

From this it follows that

$$\begin{aligned} c &= (e(P_1)^1 + c - T^2_{j_{\max}}) + (T^2_{j_{\max}} - T^1_{j_{\max}}) + (T^1_{j_{\max}} - b(P_3)^1) + \\ &\quad (b(P_3)^1 - e(P_1)^1) \\ &\overset{(5.31)(5.32)}{\geq} 4d(\lceil \frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil + \frac{1}{2} \rceil) - d + (T^2_{j_{\max}} - T^1_{j_{\max}}) + (b(P_3)^1 - e(P_1)^1) \\ &> 4d(\frac{1}{2} \lceil \frac{p_{\max}}{2d} \rceil) - d + (T^2_{j_{\max}} - T^1_{j_{\max}}) + (b(P_3)^1 - e(P_1)^1) \end{aligned}$$

$$\begin{aligned} &\geq && p_{\max} - d + (T^2_{j_{\max}} - T^1_{j_{\max}}) + l(Q_1) + l(P_2) + l(Q_2) \\ &\overset{(5.2)(5.3)}{\geq} && p_{\max} + (T^2_{j_{\max}} - T^1_{j_{\max}}) + 5d \\ &\overset{(2.13)(2.15)}{\geq} && 2p_{\max} + 6d, \end{aligned}$$

which is in contradiction with (5.30).

□(of Case 1(b))

Proof of Case 2

Let

$$p_{\max} < \frac{1}{2}c - 3d. \tag{5.33}$$

Again, we first explain how schedule $\bar{S}$ is obtained, after which we prove that $\bar{S}$ is feasible and optimal, and that it contains a critical cycle with less than two blocks. In this case, it even turns out that $\bar{S}$ contains a critical cycle with no blocks at all.

Sketch of the algorithm

Step 1 Fill the interval $[b(P_3)^1, e(P_1)^1 + c]$ with as many "candidate" transports as possible and move all these candidate transports clockwise as much as possible (fill the interval with a ccwZZ, see Figure 5.19.). Move any transport that occurs completely in the interval $[b(P_3)^1, e(P_1)^1 + c]$ clockwise until it coincides for the first time with one of the candidate transports.

Do the same for interval $[b(P_2)^1, e(P_2)^1]$: fill the interval with candidate transports and move the transports in the interval clockwise until they coincide with a candidate transport.

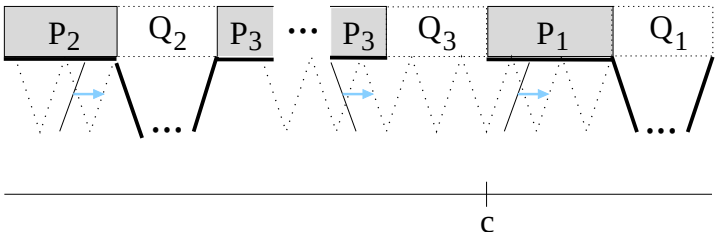


Figure 5.19. Step 1.

Notice that this may cause some tools to arrive a little bit, but less than $2d$, too late on machine 1 (see Figure 5.20). We remedy this in Step 3. In Step 4, we will create a completely new schedule on machine 2. In order to be able to do this

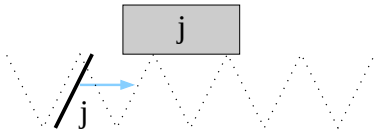


Figure 5.20. Tools may arrive, less than 2*d*, too late at machine 1.

without causing any infeasibilities, we try to make the time intervals between the time at which a tool arrives at machine 1 and the time at which the tool leaves from machine 1 again, as small as possible.

Step 2 Move the transports to machine 1 one by one clockwise by 2*d* until none of them can be moved any more without causing a transport to arrive 2*d* or more too late at machine 1 or causing a conflict with another transport to machine 1 (see Figure 5.21).
Move the transports to machine 2 one by one counterclockwise by 2*d* until none of them can be moved any more without causing a transport to leave too early from machine 1 or causing a conflict with another transport to machine 2 (see Figure 5.21).

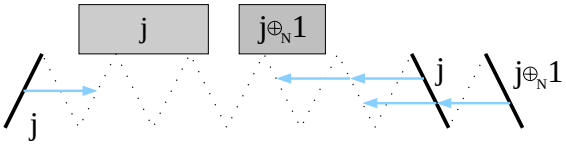


Figure 5.21. Step 2.

After Step 2 even more tools may arrive too late at machine 1. However, tools still arrive less than 2*d* too late.

For solving any conflicts between the transports to machine 1 and the processings on machine 1, we do the following.

Step 3 Each transport to machine 1 is advanced by 2*d* (see Figure 5.22).

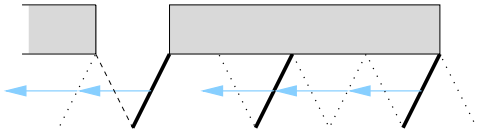


Figure 5.22. Step 3.

The above change has another nice consequence: after Step 3 all tools arrive at machine 1 strictly before the time at which they are necessary at this machine. (Hence, any existing critical cycle of the current schedule does not contain any block on machine 1.)

In the current schedule no conflicts between processings and transports exist at machine 1. However, at machine 2 tools may arrive too late or leave to early. Therefore, we create a new schedule on machine 2, as follows.

Step 4 Copy the schedule on machine 1 onto machine 2 and shift it clockwise as much as possible without creating conflicts with the tool transports to machine 1.

Before we prove that the schedule $\bar{S}$ that is created in this way is a feasible schedule, we first look at the algorithm in more detail.

Technical description of the algorithm

Let $S_j^m(s), T_j^m(s)$, respectively, denote the start times of the processings and the tool arrival times just after step s of the algorithm $(1 \leq j \leq N; m = 1,2; 1 \leq s \leq 4)$.

Step 1 Project all the transports that occur completely in $[b(P_3)^1, e(P_1)^1 + c]$ clockwise on ccwZZ$[b(P_3)^1, e(P_1)^1 + c]$) and and project all transports that occur completely in $[b(P_2)^1, e(P_2)^1]$ clockwise on ccwZZ($[b(P_2)^1, e(P_2)^1]$).

Obviously,

$$T_j^m(1) \geq T_j^m \qquad 1 \leq j \leq N, m = 1,2. \tag{5.34}$$

Furthermore, in view of Observation 5.1, in the projections the transports are moved by less than $2d$. Hence,

$$T_j^m(1) < T_j^m + 2d \qquad 1 \leq j \leq N, m = 1,2. \tag{5.35}$$

Claim 6 *The transports that before Step 1 occurred in* $[b(P_3)^1, e(P_1)^1 + c]$*, after Step 1 still occur in this interval.*

Proof (Claim 6). Consider the transports to machine 1 that in S occur completely in $[b(P_3)^1, e(P_1)^1 + c]$. Notice that these transports finish in $[b(P_3)^1 + 2d, e(P_1)^1 + c]$. Therefore, we can take $I_b = b(P_3)^1, I_e = e(P_1)^1 + c, J_b = b(P_3)^1 + 2d$ and $J_e = e(P_1)^1 + c$ in Lemma 5.3(c) and we see that after Step 1 these transports finish in $[b(P_3)^1 + 2d, e(P_1)^1 + c]$.

Now consider the transports to machine 2 that before Step 1 were completely in $[b(P_3)^1, e(P_1)^1 + c]$. In S these transports end in $[b(P_3)^1 + d, e(P_1)^1 + c - d]$. Hence,

we can take $I_b = b(P_3)^1, I_e = e(P_1)^1 + c, J_b = (P_3)^1 + d$ and $J_e = e(P_1)^1 + c - d$ in Lemma 5.3(d) and we see that after Step 1 the transports finish in $[b(P_3)^1 + d, e(P_1)^1 + c - d]$. □(Claim 6)

In an analogous way, the following can be proved.

Claim 7 *The transports that in before Step 1 occurred in* $[b(P_2)^1, e(P_2)^1 + c]$*, after Step 1 still occur in this interval.* □(Claim 7)

We see that in Step 1 we have not changed anything outside the intervals $[b(P_3)^1, e(P_1)^1 + c]$ and $[b(P_2)^1, e(P_2)^1 + c]$. Moreover, in these intervals, we have projected all transports on the ccwZZ's. Hence, in Step 1 we did not create any conflicts among the tool transports.

However, since we have moved transports clockwise, we may have caused some problems between the tool transport to machine 1 and the processing on machine 1, or between the transports to machine 2 and the processings on machine 2. Anyway, because during the projections the transports are moved by strictly less than $2d$ the transports arrive less than $2d$ too late.

For convenience, we relax Assumption 2.4 and we assume that the robot carries out all the movements of the added ccwZZ's.

Step 2 If there exists a tool j

(a) that arrives at machine 1 strictly before it is necessary at machine 1 and

(b) whose transport to machine 1 is followed by an empty transport to machine 1,

then the transport of tool j to machine 1 is delayed by $2d$. This is repeated until no such tool j can be found any more.

Moreover, if there exists a tool j

(a) whose transport to machine 2 starts at least $2d$ after the end of its processing on machine 1 and

(b) whose transport to machine 2 is proceeded by an empty transport to machine 2,

then the transport of tool j to machine 2 is advanced by $2d$.
This is repeated until no such tool can be found any more.

Clearly,

$$T_j^1(2) \geq T_j^1(1) \quad j = 1,2,\ldots,N. \tag{5.36}$$

$$T_j^2(2) \leq T_j^2(1) \quad j = 1,2,\ldots,N. \tag{5.37}$$

However, we still have for all tasks j that

$$T_j^1(2) < S_j^1(2) + 2d = S_j^1 + 2d \text{ and} \tag{5.38}$$

$$T_j^2(2) \geq S_j^1(2) + p_j + d = S_j^1 + p_j + d. \tag{5.39}$$

Since transports may still arrive too late at machine 1, we do the following.

Step 3 Shift each tool transport to machine 1 counterclockwise by $2d$.

Notice that this does not cause any conflicts among the transports. Obviously,

$$T_j^1(3) = T_j^1(2) - 2d \qquad j = 1,2,\ldots,N, \tag{5.40}$$

Moreover, for any task j we have that

$$T_j^1(3) \overset{(5.40)}{=} T_j^1(2) - 2d \overset{(5.38)}{<} S_j^1 = S_j^1(3) \tag{5.41}$$

and

$$T_j^2(3) = T_j^2(2) \overset{(5.39)}{\geq} S_j^1 + p_j + d = S_j^1(3) + p_j + d. \tag{5.42}$$

Hence, there do not exist any conflicts any more between the transports and the processings on machine 1. However, it is still an open question if there exist any conflicts between the transports and the processings on machine 2. Therefore, we create a new schedule on machine 2.

Step 4 Copy the schedule of machine 1 onto machine 2 and shift it clockwise as long as for all tasks j:

$$S_j^2(4) + p_j + d \leq T_j^1(3) + c. \tag{5.43}$$

Notice that this implies that there do not exist any conflicts between the processings on machine 2 and the transports to machine 1. Moreover, shifting the schedule on machine 2 clockwise any further would cause at least one processing on machine 2 to finish too late. Since now both machines carry out the same schedule, apart from a shift, we have for any pair j_1, j_2 of tasks:

$$S_{j_1}^2(4) - S_{j_1}^1(4) = S_{j_1}^2(4) - S_{j_1}^1 = S_{j_2}^2(4) - S_{j_2}^1 = S_{j_2}^2(4) - S_{j_2}^1(4). \tag{5.44}$$

The new schedule created by Steps 1-4 is denoted by $\bar{S}$.

Feasibility of $\bar{S}$

Notice that in $\bar{S}$ there do not exist any conflicts:

1. among the processings on machine $m (m = 1,2)$.
2. among the transports.
3. between the transports to machine 1 and the processings on machine 1 and
4. between the processings on machine m and the transports to machine $m \oplus_2 1 (m = 1,2)$.

Hence, for checking the feasibility of schedule $\bar{S}$ it is only necessary to check whether all tools arrive in time at machine 2, i.e., whether $T_j^2(4) \leq S_j^2(4)$ for any task j. Since this is an extensive proof, we give it in Appendix A. In Appendix A we will even prove that

$$T_j^2(4) < S_j^2(4) \qquad \text{for any task } j, \tag{5.45}$$

from which we conclude that $\bar{S}$ is a feasible schedule.

Optimality of $\bar{S}$

Since $\bar{S}$ is a feasible schedule of the same length as S, $\bar{S}$ is also optimal.

A critical cycle of $\bar{S}$ contains no blocks

Since $\bar{S}$ is an optimal schedule, it contains a critical cycle C. Notice that for any task j we have that $T_j^1(4) = T_j^1(3) \overset{(5.41)}{<} S_j^1(3) = S_j^1(4)$. Hence, C does not contain a block on machine 1. Moreover, since $T_j^2(4) \overset{(5.45)}{<} S_j^2(4)$, C does not contain a block on machine 2 either.

□(of Case 2)

□

5.7 Summary

In view of Theorem 5.1, we need only consider critical cycles with at most two blocks. Therefore, we only look at schedules with critical cycles that contain:

1. *no blocks*. If in this case the critical cycle does not contain any empty transports, then $c_{\text{opt}} = 2Nd$.
2. *one block and no transports*, in this case: $c_{\text{opt}} = \sum_{j=1}^{N} p_j$.
3. *one block and several transports*.
4. *two blocks on the same machine*.
5. *two blocks on different machines*. In this case, both blocks contain only one task and they both contain the same task. Moreover, $c_{\text{opt}} = 2p_{\max} + 2d$.

6

Relevant structures of critical cycles

6.1 Introduction

In the previous chapter we have seen that for obtaining an optimal schedule, we need only consider schedules with critical cycles of five different structures.

For convenience, we say that a critical cycle is of

structure 1 if it consists of the transports of tool $j_{\max}$ and the processings of task $j_{\max}$.

structure 2 if it consists of processing times only.

structure 3 if it consists of transport times only. If the critical cycle contains $2E$ empty transports, we speak about structure 3^E $(E \geq 0)$.

structure 4 if it contains one block and some transport times. Without loss of generality, we assume that the block is on machine 1.

structure 5 if it contains two blocks on the same machine. Without loss of generality, we assume that the blocks are on machine 1.

If a schedule contains a critical cycle of structure i, then we say that it is *a schedule of structure* $i (i = 1, 2, \ldots, 5)$. Notice that a schedule can be of more than one structure.

Let $P^2_{\text{cycl,i}}$ be the problem of determining an optimal schedule of structure i for problem $P^2_{\text{cycl}}(i = 1,2,\ldots,5)$. Notice that if we can solve $P^2_{\text{cycl,i}}$ in polynomial time for all $i \in \{1,2,\ldots,5\}$, then we can also solve P^2_{cycl} in polynomial time, since there are only 5 different choices for i. On the other hand, if we can prove that there exists an $i \in \{1,2,\ldots,5\}$ for which $P^2_{\text{cycl,i}}$ is $\mathcal{NP}$-complete, then this could be an indication that P^2_{cycl} is also $\mathcal{NP}$-complete.

In this chapter we have a closer look at schedules and critical cycles of the structures 1-5. For every structure i, we first describe how a linear mixed-integer program for solving $P^2_{\text{cycl,i}}$ can be obtained. Secondly, we investigate if we can restrict ourselves to *symmetric schedules*, i.e., we try to answer the question whether there exists an optimal schedule S of structure i for which $S^2_j = S^1_j + \frac{1}{2}c$ and $T^2_j = T^1_j + \frac{1}{2}c$ for all tasks j.

For a symmetric schedule, we have the following nice result.

Lemma 6.1. *In a symmetric schedule, we have* $x_{j_2 j_1} = 1 - x_{j_1 j_2} (1 \leq j_1 < j_2 \leq N)$.

Proof. Let S be a symmetric schedule and let $j_1, j_2 (j_1 < j_2)$ be two tasks. For this schedule we have (see (2.17) and (2.18)) that

$$\begin{cases} x_{j_1 j_2} c + d & \leq T^2_{j_2} - T^1_{j_1} \leq (1 + x_{j_1 j_2})c - d \quad \text{and} \\ (x_{j_2 j_1} - 1)c + d & \leq T^2_{j_1} - T^1_{j_2} \leq x_{j_2 j_1} c - d. \end{cases} \tag{6.1}$$

Since $T^2_{j_1} = T^1_{j_1} + \frac{1}{2}c$ and $T^2_{j_2} = T^1_{j_2} + \frac{1}{2}c$ this corresponds to

$$\begin{cases} x_{j_1 j_2} c + d & \leq T^1_{j_2} - T^2_{j_1} + c \leq (1 + x_{j_1 j_2})c - d \quad \text{and} \\ (x_{j_2 j_1} - 1)c + d & \leq T^1_{j_1} - T^2_{j_2} + c \leq x_{j_2 j_1} c - d, \end{cases}$$

which is equivalent with

$$\begin{cases} -x_{j_1 j_2} c + d & \leq T^2_{j_1} - T^1_{j_2} \leq (-x_{j_1 j_2} + 1)c - d \quad \text{and} \\ (-x_{j_2 j_1} + 1)c + d & \leq T^2_{j_2} - T^1_{j_1} \leq (-x_{j_2 j_1} + 2)c - d. \end{cases} \tag{6.2}$$

By comparing (6.1) and (6.2), and by using that $x_{j_1 j_2}$ and $x_{j_2 j_1}$ are 0-1 variables, we see that $x_{j_2 j_1}$ needs to be equal to $1 - x_{j_1 j_2}$. □

In some cases in which no symmetric optimal schedule exists, an optimal schedule S exists for which $S^2_j = S^1_j + \frac{1}{2}c - d$ and $T^2_j = T^1_j + \frac{1}{2}c - d$ for any task j. We call such a schedule a *near-symmetric schedule*.

In this chapter we also give, for any structure $i (1 \leq i \leq 5)$, some necessary conditions that must hold for a problem instance to make schedules of structure i possible. Moreover, we try to answer the question whether the problem $P^2_{\text{cycl,i}}$ can be solved in polynomial time. Here the existence of a symmetric schedule turns out to be very useful.

6.2 Structure 1

We first look at schedules of structure 1, i.e., schedules with a critical cycle that consists of the transports of tool $j_{\max}$ and the processings of task $j_{\max}$.

6.2.1 The linear mixed-integer program

Consider a schedule S of structure 1. For this schedule the following holds (see Figure 6.1):

$$\begin{aligned} T^m_{j_{\max}} &= S^m_{j_{\max}} && (m = 1,2), \\ T^2_{j_{\max}} &= S^1_{j_{\max}} + p_{\max} + d, \\ T^1_{j_{\max}} + c &= S^2_{j_{\max}} + p_{\max} + d, && \text{where} \\ c &= 2p_{\max} + 2d. \end{aligned}$$

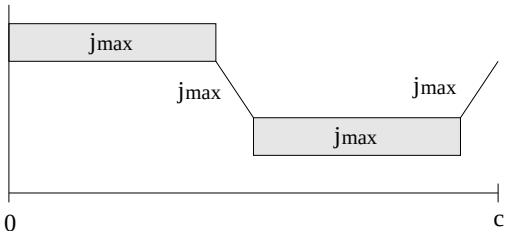


Figure 6.1. A critical cycle of structure 1.

The above information leads to some equality signs and a prescribed objective function value in Problem formulation 4 of Chapter 2. The corresponding linear mixed-integer problem, we call $MIP^2_{\text{cycl},1}$. $MIP^2_{\text{cycl},1}$ is a feasibility problem, since the objective value $q = c^{-1}$ has been fixed by the above constraints. Notice that a feasible schedule of structure 1 is of length $2p_{\max} + 2d$ and therefore, due to Lemma 3.2, it is also an optimal schedule for MIP^2_{cycl}.

6.2.2 A symmetric schedule

Not surprisingly, the existence of an optimal schedule of structure 1 implies the existence of an optimal symmetric schedule of structure 1.

Lemma 6.2. *If there exists an optimal schedule of structure 1, then there exists an optimal symmetric schedule of structure 1.*

Proof. Consider an optimal schedule S of structure 1. In S there exist $2N$ nonempty transports of which either at least N finish in the interval $I_1 := [T^1_{j_{\max}}, T^2_{j_{\max}})$ or at least N finish in the interval $I_2 := [T^2_{j_{\max}}, T^1_{j_{\max}} + c)$. Due to symmetry considerations, we may assume without loss of generality that the former is the case. Suppose that there are k nonempty transports to machine 1 in I_1. (Since, in total there are N nonempty transports to machine 1, $k \leq N$.) This implies that the tools belonging to

the tasks $j_{\max}, j_{\max} \oplus_N 1, \ldots, j_{\max} \oplus_N (k-1)$ arrive at machine 1 in I_1. Since, the total number of nonempty transports that finish in I_1 is larger than or equal to N, there are at least $N-k$ nonempty transports to machine 2 in I_1. Hence, the tools $j_{\max} \ominus_N 1, \ldots, j_{\max} \ominus_N (N-k)$ arrive at machine 2 in I_1. We see that each tool is transported at least once in I_1. This suggests the creation of a new schedule by copying the schedule of I_1 "upside down" in I_2. We create the following schedule $\bar{S}$:

$$\bar{S}_j^m = \begin{cases} S_j^m & \text{for } j = j_{\max}; m = 1,2, \\ S_j^2 - \frac{1}{2}c\delta_{m1} & \text{otherwise.} \end{cases}$$

$$\bar{T}_j^m = \begin{cases} T_j^1 + \frac{1}{2}c\delta_{m2} & \text{if } j \neq j_{\max}; m = 1,2; T_j^1 \in I_1, \\ T_j^2 - \frac{1}{2}c\delta_{m1} & \text{if } j \neq j_{\max}; m = 1,2; T_j^2 \in I_1, \\ T_j^m & \text{otherwise.} \end{cases}$$

In Figure 6.2(a) and (b) an example is shown of how schedule $\bar{S}$ can be obtained.

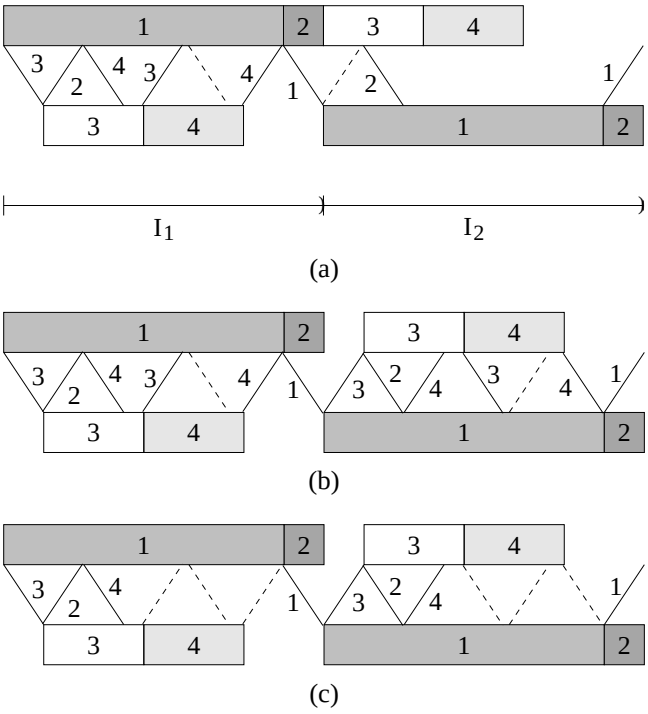


Figure 6.2. (a) Schedule S. (b) Schedule $\bar{S}$. (c) Schedule $\bar{S}$ after correction.

Notice that in $\bar{S}$ the schedules of processings on both machines are feasible. Furthermore, the tools arrive in time at the machines and they do not leave too early from the machines. However, it is possible that there exist tools that are transported four times, two times to each machine (see Figure 6.2(b)). This can be corrected as

follows. If tool j is transported four times then we convert the transport of tool j to machine 1 in I_1 and the transport of tool j to machine 2 in I_2 into empty transports ($j = 1,2,\ldots,N$). An example is shown in Figure 6.2(c). It is not difficult to see that the schedule that is created in this way is feasible, symmetric and of structure 1. □

The Lemma's 6.1 and 6.2 directly imply the following corollary.

Corollary 6.1. *If there exists an optimal schedule of structure 1, then there exists an optimal symmetric schedule of structure 1 in which* $x_{j_2 j_1} = 1 - x_{j_1 j_2}$ $(1 \leq j_1 < j_2 \leq N)$. □

Hence, with respect to the schedules of structure 1, we may restrict ourselves to the symmetric ones and therefore, in $MIP^2_{\text{cycl},1}$ we can replace S^2_j by $S^1_j + \frac{1}{2}c$, T^2_j by $T^1_j + \frac{1}{2}c (1 \leq j \leq N, m = 1,2)$ and $x_{j_2 j_1}$ by $1 - x_{j_1 j_2} (1 \leq j_1 < j_2 \leq N)$. This halves the number of decision variables. After also eliminating redundant equations, the resulting linear mixed-integer problem contains $2N + \frac{1}{2}N(N-1)$ decision variables and $4N + N(N-1) + 1$ equations. (MIP^2_{cycl} has $4N + N(N-1)$ decision variables and $8N + 2N(N-1) + 1$ equations.)

6.2.3 Necessary conditions for the problem instance

In this section we derive some conditions that must hold for a problem instance in order to make the existence of a feasible schedule of structure 1 possible. These conditions will be useful in the next chapter.

Consider a schedule S of structure 1. Let $m \in \{1,2\}$. In S there are at most $\lfloor \frac{p_{\max}}{2d} \rfloor + 1$ transports to machine m that finish in $I_1 := [T^1_{j_{\max}}, T^2_{j_{\max}}]$. Therefore, there exist at least $N - \lfloor \frac{p_{\max}}{2d} \rfloor - 1$ nonempty transports to machine m in $I_2 := (T^2_{j_{\max}}, T^1_{j_{\max}} + c)$. However, there are at most $\lfloor \frac{p_{\max}}{2d} \rfloor$ transports to machine m that finish in I_2. Hence, there are at most $2\lfloor \frac{p_{\max}}{2d} \rfloor - N + 1$ empty transports to machine m in I_2. Notice that most space for the processings on machine 1, with respect to the transports to machine 2, is created by placing the nonempty transports to machine 2 as late as possible in I_2 as shown in Figure 6.3).

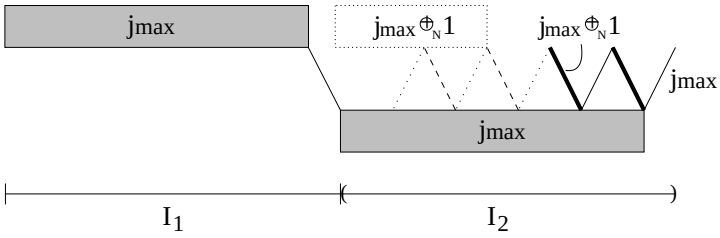


Figure 6.3. Most space is available for the processings on machine 1 if the nonempty transports to machine 2 in I_2 are placed as late as possible.

Hence, the processings on machine 1 can only fit if for $k = 1,2,\ldots,N-\lfloor\frac{p_{\max}}{2d}\rfloor-1$ we have that

$$\sum_{i=1}^{k} p_{j_{\max}\oplus_N i} \leq p_{\max} - 2d(N - \lfloor\frac{p_{\max}}{2d}\rfloor - 1 - k).$$

Moreover most space for the processings on machine 1, with respect to the transports to machine 1, is created by placing the nonempty transports to machine 1 as early as possible in I_2 (see Figure 6.4).

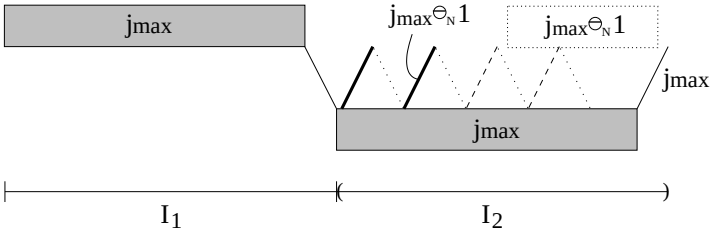


Figure 6.4. Most space is available for the processings on machine 1 if the nonempty transports to machine 1 in I_2 occur as early as possible.

There can only be enough space for the processings on machine 1 if for $k = 1,2,\ldots,N-\lfloor\frac{p_{\max}}{2d}\rfloor-1$ we have that

$$\sum_{i=1}^{k} p_{j_{\max}\ominus_N i} \leq p_{\max} - 2d(N - \lfloor\frac{p_{\max}}{2d}\rfloor - 1 - k).$$

The above relations do not contain any of the decision variables. However, they will be useful in the next chapter, where we develop algorithms for solving P^2_{cycl}.

6.2.4 Complexity status of the problem

Although $MIP^2_{\text{cycl},1}$ is a linear mixed-integer program, the existence of a schedule of structure 1, can be verified in polynomial time.

Theorem 6.1. *If there exists an optimal schedule of structure 1, then such a schedule can be found in polynomial time.*

Proof. If there exists an optimal schedule, then, according to Lemma 6.2, there also exists a symmetric optimal schedule of structure 1. Moreover, by Lemma 3.2 any feasible schedule of structure 1 is optimal. This implies that for proving the theorem it is sufficient to show how, if it exists, a feasible symmetric schedule of structure 1 can be created in polynomial time. We show that, for any problem instance, we can

(a) verify the existence of a feasible symmetric schedule of structure 1 and

(b) create a feasible symmetric schedule of structure 1, if it exists,

by solving at most N different linear programming problems. As it is well-known, linear programming problems can be solved in polynomial time.
Without loss of generality, we assume that task $j_{\max}$ is task 1. We consider N different situations. Let in situation k, k be the highest number for which tool k is present at machine 1 at time p_1. Whether under this assumption a feasible symmetric schedule S of structure 1 exists can be verified by the algorithm described in Figure 6.5.
If in situation k the linear program in Step 8 has a feasible solution, then this solution constitutes a feasible symmetric schedule of structure 1, in which k is the highest number for which tool k is present at machine 1 at time p_1. If no feasible solution is found, then, obviously, no feasible symmetric schedule of structure 1 exists, in which k is the highest number for which tool k is present at machine 1 at time p_1. If in one of the situations $S_1, S_2, \ldots, S_k$, a feasible schedule S is found, then it is an optimal symmetric schedule of structure 1. Otherwise, no optimal symmetric schedule of structure 1 exists. □

6.3 Structure 2

The second type of schedules that we consider, are schedules with a critical cycle that consists of processing times only.

6.3.1 The linear mixed-integer program

For a schedule S of structure 2 the following equations are valid:

$$\begin{aligned} S_j^m &= S_{j-1}^m + p_{j-1} \quad (2 \le j \le N; m = 1,2), \\ S_1^m + c &= S_N^m + p_N \quad (m = 1,2), \text{where} \\ c &= \textstyle\sum_{j=1}^{N} p_j. \end{aligned}$$

Hence, we get equality signs in (2.14) and a prescribed objective function value in Problem formulation 4. The corresponding linear mixed-integer program, we call $MIP^2_{\text{cycl},2}$. It contains $4N + N(N-1)$ decision variables and $8N + 2N(N-1) + 1$ equations. $MIP^2_{\text{cycl},2}$ is a feasibility problem, since its objective value $q = c^{-1}$ is fixed. In view of Lemma 3.3, any feasible schedule of $MIP^2_{\text{cycl},2}$ constitutes an optimal schedule for MIP^2_{cycl}.

6.3.2 A symmetric schedule

If an optimal schedule of structure 2 exists, then there does not have to exist a symmetric optimal schedule of structure 2. This can be seen in the Figure 6.6(a) and (b). Figure 6.6(a) shows an optimal schedule, while in Figure 6.6(b) for the same problem instance a symmetric schedule of minimum length is given.

Step 1 Let $T_1^1 = S_1^1 = 0$ and $T_1^2 = S_1^2 = p_1 + d$. Place at time p_1 the tools $1, 2, \ldots, k$ on machine 1 and all other tools on machine 2. Moreover, due to the symmetry, let at time $2p_1 + d$ the tools $1, 2, \ldots, k$ be on machine 2 and all other tools on machine 1 (see Figure 6.5(a) for an example where $k = 5$).

Step 2 Notice that since at time p_1 all tools $1, 2, \ldots, k$ are present at machine 1, we can place the processings of the tasks $2, \ldots, k$ on machine 1 directly – without introducing idle time – after the processing of task 1 on machine 1 (see Figure 6.5(b)).

Step 3 Since, at time $T_1^1 + c$ the tools $k+1, k+2 \ldots, N$ are still at machine 1, the processings of the tasks $k+1, \ldots, N$ on machine 1 can be placed immediately before $(S_1^1 + c,)_p$(see Figure 6.5(c)).

Step 4 Now we have created a schedule on machine 1. Obviously, for creating a symmetric schedule we have to take $S_j^2 = S_j^1 + \frac{1}{2}c$ for all tasks j (see Figure 6.5(d)).

Step 5 Now suppose that we fill the interval $[0, p_1]$ with as many feasible transports as possible (without paying attention to the exact position of the transports in the interval). (See Figure 6.5(e).)

Step 6 Notice that we can use the first $|N-k|$ of these added transports to machine 2 for transporting the tools $k+1, \ldots, N$ to machine 2 and the last $k-1$ added transports to machine 1 for transporting the tools $2, \ldots, k$ to machine 1 (see Figure 6.5(f)).

Step 7 Now, since S has to become symmetric $T_j^2 = T_j^1 + \frac{1}{2}c$ for $j = 2, 3, \ldots, k$ and $T_j^1 = T_j^2 - \frac{1}{2}c$ for $j = k+1, \ldots, N$ (see Figure 6.5(g)).

Step 8 In this way, although we do not know the exact position of all the tool transports, the transport strategy of the robot is completely determined. By adding this transport to $MIP^2_{\text{cycl},1}$, all integer variables become fixed and therefore the resulting program is a linear programming problem and can be solved in polynomial time. Hence, if a feasible schedule exists then this is found by solving the linear program that is obtained from $MIP^2_{\text{cycl},1}$ by adding the transport strategy and the already known values of S_j^m and the relations $T_j^2 = T_j^1 + \frac{1}{2}c\,(1 \leq j \leq N; m = 1, 2)$.

Figure 6.5. Algorithm for verifying the existence of a schedule of structure 1 in situation k.

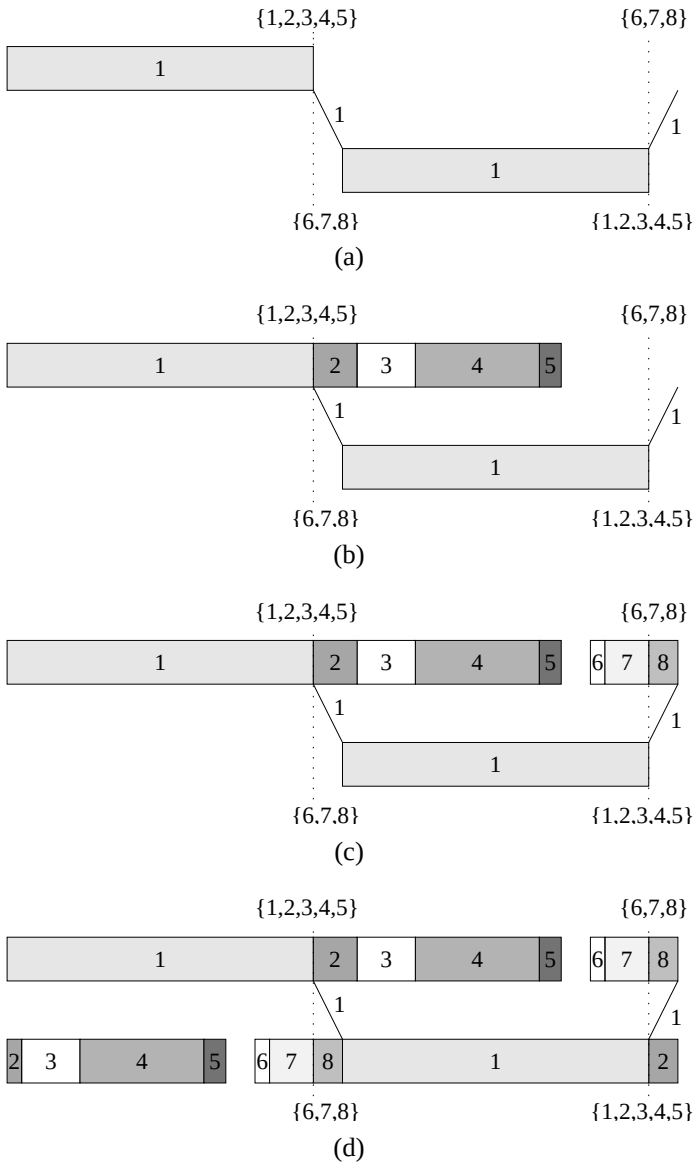


Figure 6.6. (Part 1.) The creation of S for problem instance $(\boldsymbol{p},d) = ((42,6,8,17,3,2,6,4),4)$ and $k = 5$: (a) After Step 1, (b) After Step 2, (c) After Step 3 and (d) After Step 4.

6.3.3 Necessary conditions for the problem instance

Consider an optimal schedule S of structure 2. Notice that we can always move the schedule of processings on machine 1 counterclockwise until there exists at least

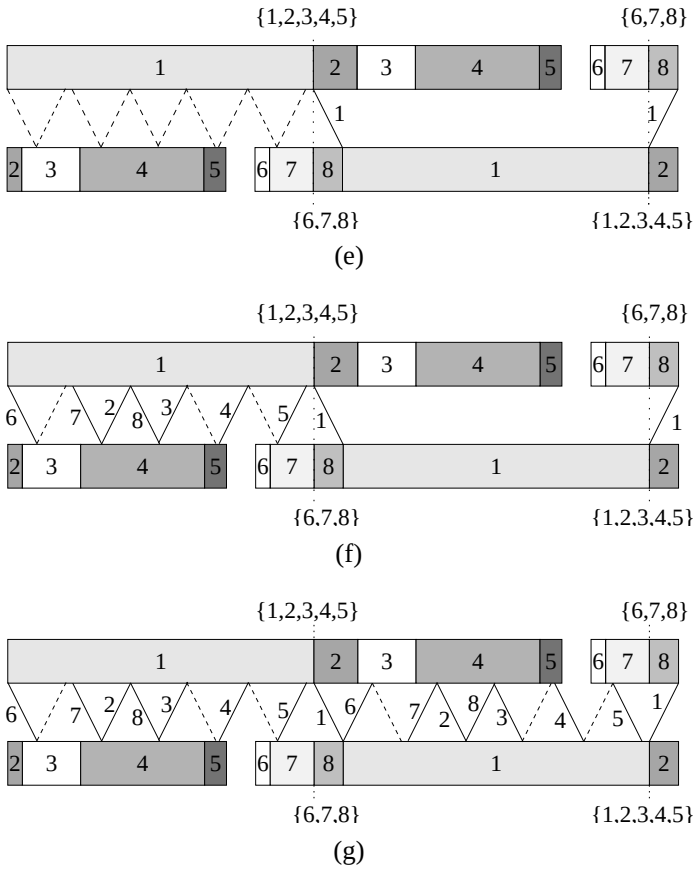


Figure 6.5. (Part 2.) The creation of S for problem instance $(\boldsymbol{p},d) = ((42,6,8,17,3,2,6,4),4)$ and $k = 5$: (e) After Step 5, (f) After Step 6 and (g) After Step 7.

one task j_1 for which $T^1_{j_1} = S^1_{j_1}$. Notice that tool $j_1 \oplus_N 1$ cannot arrive at machine 1 before time $T^1_{j_1} + 2d$ (see Figure 6.7). This implies that

$$p_{j_1} = S^1_{j_1 \oplus_N 1} - S^1_{j_1} \overset{(2.13)}{\geq} T^1_{j_1 \oplus_N 1} - S^1_{j_1} = T^1_{j_1 \oplus_N 1} - T^1_{j_1} \overset{(2.16)}{\geq} 2d.$$

By following this reasoning also for tool $j_1 \oplus_N 2, j_1 \oplus_N 3, \ldots, j_1 \oplus_N N$ we see that

$$\sum_{i=1}^{k} p_{j_1 \oplus_N (i-1)} \geq 2dk \qquad (k = 1,2,\ldots,N).$$

Instead of moving the schedule on machine 1 counterclockwise, we can also move it clockwise. Suppose that we move it clockwise until for the first time there ex-

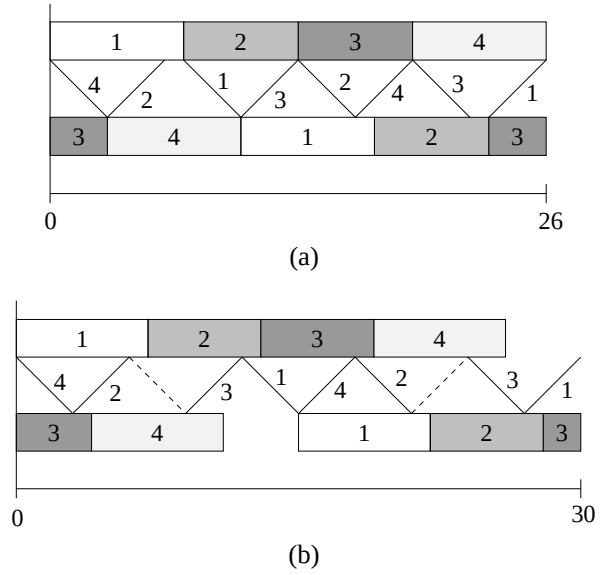


Figure 6.6. For problem instance $(\boldsymbol{p},d)=((7,6,6,7),3)$: (a) An optimal schedule, (b) An optimal symmetric schedule.

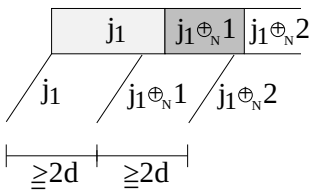


Figure 6.7. $p_{j_1}\geq 2d, p_{j_1}+p_{j_1\oplus_N 1}\geq 4d$.

ists at least one task j_2 for which $S^1_{j_2}+p_{j_2}+d=T^2_{j_2}$. Tool $j_2\ominus_N 1$ cannot arrive at machine 2 later than at time $T^2_{j_2}-2d$. In other words, $T^2_{j_2\ominus_N 1}\leq T^2_{j_2}-2d$ and therefore,

$$p_{j_2}=T^2_{j_2}-S^1_{j_2}-d\overset{(2.16)}{\geq}T^2_{j_2\ominus_N 1}-S^1_{j_2}+d=T^2_{j_2\ominus_N 1}-S^1_{j_2\ominus_N 1}-p_{j_2\ominus_N 1}+d\overset{(2.15)}{\geq}2d.$$

Following this reasoning also for tool $j_2\ominus_N 2, j_2\ominus_N 3,\ldots,j_2\ominus_N N$ gives

$$\sum_{i=1}^{k}p_{j_2\ominus_N(i-1)}\geq 2dk\qquad(k=1,2,\ldots,N).$$

Notice that for obtaining the tasks j_1 and j_2 we need to have an explicit schedule S. Therefore, the derived information cannot be expressed in relations on the decision

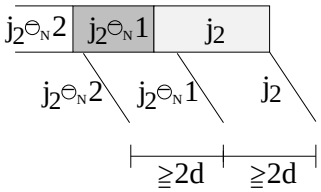


Figure 6.8. $p_{j_2} \geq 2d, p_{j_2} + p_{j_2 \ominus_N 1} \geq 4d$.

variables S_j^m and $T_j^m (1 \leq j \leq N, m = 1,2)$. Hence, the above information cannot be incorporated in $MIP^2_{\text{cycl},2}$. However, for a problem instance an optimal schedule of structure 2 can only exist if there are two tasks j_1 and j_2 for which the equations derived in this section are valid. This information will be used in the next chapter.

6.3.4 Complexity status of the problem

If the problem $P^2_{\text{cycl},2}$ is solvable in polynomial time is still an open question.

6.4 Structure 3

The next schedules to consider are schedules with critical cycles that consist of transports only.

6.4.1 The linear mixed-integer program

We first look at schedules with critical cycles that consist of nonempty transports only, after which we consider schedules with critical cycles that also contain empty transports.

Structure 3^0

Consider a schedule S of structure 3^0. In this schedule the following equations are valid:

$$\begin{aligned} T_j^m &= T_{j-1}^m + 2d \quad (j = 2, \ldots, N; m = 1,2), \\ T_1^m + c &= T_N^m + 2d \quad (m = 1,2), \text{where} \\ c &= 2Nd. \end{aligned}$$

This corresponds to equality signs in (2.16) and a prescribed objective function value in Problem formulation 4. Let $MIP^2_{\text{cycl},3^0}$ be the corresponding linear mixed-integer program. Also $MIP^2_{\text{cycl},3^0}$ is a feasibility problem. Furthermore, according to Lemma 3.1, a feasible schedule of structure 3^0 is an optimal schedule for MIP^2_{cycl}.

Structure $3^E(E>0)$

An optimal schedule of structure $3^E(E>0)$ is of length

$$c = 2(N+E)d.$$

The linear mixed-integer program that is obtained with this information we call $MIP^2_{\text{cycl},3^E}$. For a fixed value of E, $MIP^2_{\text{cycl},3^E}$ is a feasibility problem. Notice that a feasible schedule of $MIP^2_{\text{cycl},3^E}$ is not necessarily an optimal schedule of MIP^2_{cycl}.

6.4.2 A (near-)symmetric schedule

For structure 3, an optimal symmetric schedule exists if $N+E$ is odd.

Lemma 6.3. *If there exists an optimal schedule of structure* $3^E(E\geq 0)$ *and* $N+E$ *is odd, then there exists an optimal symmetric schedule of structure* 3^E.

Proof. Consider an optimal schedule S of structure $3^E(E\geq 0)$. Notice that in S there exists a machine m for which $T_1^{m\oplus_2 1}+\delta_{m2}c-T_1^m\geq\frac{1}{2}c$. Renumber the machines such that this holds for machine 1. Hence, since $T_1^1=0$ we have that $T_1^2\geq\frac{1}{2}c$, which implies that $T_j^2\geq\frac{1}{2}c$ and $S_j^2\geq\frac{1}{2}c$ for every task j.
We create a schedule $\bar{S}$ as follows. If in S the processing of a task j on machine 1 is started t_1 time units after time 0 and the processing of task j on machine 2 begins t_2 time units after time $\frac{1}{2}c$, then in $\bar{S}$ we let the processing of task j on machine m begin $\min\{t_1,t_2\}$ time units after time $\frac{1}{2}\delta_{m2}c(m=1,2)$. For the transports, we do the same: if in S the transport of tool j to machine 1 finishes at time t_3 and the transport of tool j to machine 2 ends at time $\frac{1}{2}c+t_4$, then in $\bar{S}$ we let the transport of task j on machine m finish at time $\frac{1}{2}\delta_{m2}c+\min\{t_3,t_4\}(m=1,2)$.
Hence, $\bar{S}$ is the following schedule:

$$\begin{array}{lcll}
\bar{S}_j^1 & = & \min\{S_j^1, S_j^2-\frac{1}{2}c\} & j=1,2,\ldots,N,\\
\bar{T}_j^1 & = & \min\{T_j^1, T_j^2-\frac{1}{2}c\} & j=1,2,\ldots,N,\\
\bar{S}_j^2 & = & \bar{S}_j^1+\frac{1}{2}c & j=1,2,\ldots,N,\\
\bar{T}_j^2 & = & T_j^1+\frac{1}{2}c & j=1,2,\ldots,N, \text{where}\\
c & = & 2(N+E)d. &
\end{array}$$

We first prove that $\bar{S}$ is an feasible schedule by showing that for the given variable assignments the restrictions of Problem formulation 4 are valid. For task j and machine m, we have that

$$\begin{array}{rcll}
\bar{S}_j^m & = & \min\{S_j^1, S_j^2-\frac{1}{2}c\}+\frac{1}{2}c\delta_{m2} & \\
& \overset{(2.13)}{\geq} & \min\{T_j^1, T_j^2-\frac{1}{2}c\}+\frac{1}{2}c\delta_{m2} & \\
& = & \bar{T}_j^m & \text{(validity of (2.13)).}
\end{array}$$

$$\begin{aligned}
\bar{S}^m_j + \delta_{j1}c &= \min\{S^1_j + \delta_{j1}c, S^2_j + \delta_{j1}c - \tfrac{1}{2}c\} + \tfrac{1}{2}c\delta_{m2} \\
&\overset{(2.14)}{\geq} \min\{S^1_{j\ominus_N 1} + p_{j\ominus_N 1}, S^2_{j\ominus_N 1} + p_{j\ominus_N 1} - \tfrac{1}{2}c\} + \tfrac{1}{2}c\delta_{m2} \\
&= \bar{S}^m_{j\ominus_N 1} + p_{j\ominus_N 1} \qquad \text{(validity of (2.14))}.
\end{aligned}$$

$$\begin{aligned}
\bar{T}^2_j &= \min\{T^1_j, T^2_j - \tfrac{1}{2}c\} + \tfrac{1}{2}c \\
&\overset{(2.15)}{\geq} \min\{S^2_j + p_j + d - \tfrac{1}{2}c, S^1_j + p_j + d\} \\
&= \bar{S}^1_j + p_j + d \qquad \text{(validity of (2.15) part 1)}.
\end{aligned}$$

$$\begin{aligned}
\bar{T}^1_j + c &= \bar{T}^2_j + \tfrac{1}{2}c \\
&\geq \bar{S}^1_j + p_j + d + \tfrac{1}{2}c \\
&= \bar{S}^2_j + p_j + d \qquad \text{(validity of (2.15) part 2)}.
\end{aligned}$$

$$\begin{aligned}
\bar{T}^m_j + \delta_{j1}c &= \min\{T^1_j + \delta_{j1}c, T^2_j - \tfrac{1}{2}c + \delta_{j1}c\} + \tfrac{1}{2}c\delta_{m2} \\
&\overset{(2.16)}{\geq} \min\{T^1_{j\ominus_N 1} + 2d, T^2_{j\ominus_N 1} + 2d - \tfrac{1}{2}c\} + \tfrac{1}{2}c\delta_{m2} \\
&= \bar{T}^m_{j\ominus_N 1} + 2d \qquad \text{(validity of (2.16))}.
\end{aligned}$$

Moreover, notice that since S is of structure 3^E we have $T^1_j = 2k_1 d$ with $k_1 \in \mathbb{N}\cup\{0\}$ and $T^2_j = 2k_2 d + d$ with $k_2 \in \mathbb{N}\cup\{0\}$. Hence, $\bar{T}^1_j = \min\{T^1_j, T^2_j - \frac{1}{2}c\} = \min\{2k_1 d, 2k_2 d + d - (N+E)d\} \overset{N+E \text{ odd}}{=} 2k_3 d$ for a certain $k_3 \in \mathbb{N}\cup\{0\}$ and $\bar{T}^2_j = \bar{T}^1_j + \frac{1}{2}c = 2k_3 d + (N+E)d \overset{N+E \text{ odd}}{=} 2k_4 d + d$ for a certain $k_4 \in \mathbb{N}\cup\{0\}$. From this, we conclude that a transport to machine 1 and a transport to machine 2 can never overlap and, thus, (2.17) and (2.18) are valid.
The above implies that $\bar{S}$ is feasible. Obviously, $\bar{S}$ is also symmetric and of schedule 3^E. □

The Lemma's 6.1 and 6.3 imply the following corollary.

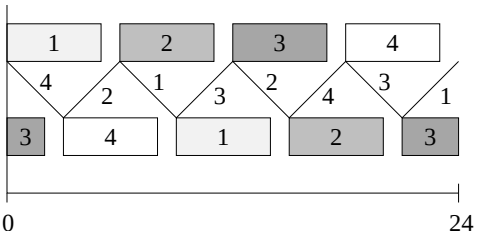
Corollary 6.2. *If there exists an optimal schedule of structure* $3^E (E \geq 0)$ *and* $N+E$ *is odd, then there exists an optimal symmetric schedule of structure* 3^E *in which* $x_{j_2 j_1} = 1 - x_{j_1 j_2} (1 \leq j_1 < j_2 \leq N)$. □

Hence, if $N+E$ is odd then in $MIP^2_{\text{cycl},3^E}$ we can replace S^2_j by $S^1_j + \frac{1}{2}c, T^2_j$ by $T^1_j + \frac{1}{2}c (1 \leq j \leq N)$ and $x_{j_2 j_1}$ by $1 - x_{j_1 j_2} (1 \leq j_1 < j_2 \leq N)$. After eliminating redundant equations as well, we get a linear mixed-integer program with $2N + \frac{1}{2}N(N-1)$ decision variables and $4N + N(N-1) + 1$ equations.

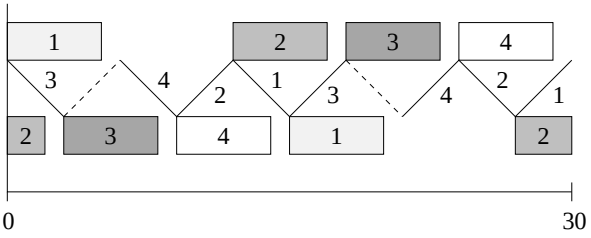
If in addition $E = 0$, then we have that $T^1_j = 2(j-1)d$ and, consequently, $T^2_j = (2j + N - 2)d$ $(1 \leq j \leq N)$. This implies that we can eliminate all equations that contain the variables $x_{ij} (1 \leq j_1, j_2, \leq N, j_1 \neq j_2)$ from the linear mixed-integer program. In this way we get a linear program with $2N$ decision variables and

$4N+1$ equations. We remark that by using the equations $T_j^1 = 2(j-1)d$ and $T_j^2 = (2j-2+N)d$, also the decision variables $T_j^m (1 \leq j \leq N; m = 1,2)$ can be removed from the linear mixed-integer program. However, in practice it turns out that it takes longer to remove these variables and to solve the resulting linear mixed-integer problem than to solve the problem directly.

If $N+E$ is even then there does not have to exist a symmetric optimal schedule of structure 3. This can be seen in the Figures 6.9(a) and (b). Figure 6.9(a) shows an optimal schedule, while in 6.9(b) for the same problem instance a symmetric schedule of minimum length is given.



(a)



(b)

Figure 6.9. For problem instance $(\boldsymbol{p}, d) = ((5,5,5,5),3)$: (a) An optimal schedule of structure 3, (b) An optimal symmetric schedule of structure 3.

Lemma 6.4. *If there exists an optimal schedule of structure* 3^0 *and N is even, then there exists an optimal near-symmetric schedule of structure* 3^0.

Proof. Let S be an optimal schedule of structure 3^0 and let N be even. Suppose that we start the creation of a new schedule $\bar{S}$ as follows.

$$\bar{T}_j^1 = 2(j-1)d \qquad j = 1,2,\ldots,N,$$

$$\bar{T}_j^2 = 2(j-1)d + \frac{1}{2}c - d \qquad j = 1,2,\ldots,N.$$

Suppose that we cannot complete $\bar{S}$ in such a way that a feasible schedule is obtained. Notice that, because of Assumption 2.3, this implies that there exist tasks

j_1 and j_2 such that

$$\sum_{j=j_1}^{j_2} p_j + d > \overline{T}^2_{j_2} - \overline{T}^1_{j_1} = 2(j_2 - j_1) + \frac{1}{2}c - d. \tag{6.3}$$

(If $j_1 > j_2$, then by $\sum_{j=j_1}^{j_2} p_j$ is meant $\sum_{j=j_1}^{N} p_j + \sum_{j=1}^{j_2} p_j$.)

Since schedule S is of structure 3^0, we have that $T^m_{j_2} - T^m_{j_1} = 2d(j_2 - j_1)$. From this, it follows that

$$T^1_{j_2} - T^1_{j_1} + T^2_{j_2} - T^2_{j_1} + c = 4d(j_2 - j_1) + c = 4(j_2 - j_1)d + 2Nd \overset{N\text{even}}{=} 4k_1 d$$

for a certain $k_1 \in \mathbb{N}$. Since $T^2_{j_2} - T^1_{j_1} = 2k_2 d + d$ and $T^1_{j_2} + c - T^2_{j_1} = 2k_3 d + d$, for certain $k_2, k_3 \in \mathbb{N}$, this implies that

$$\min\{T^2_{j_2} - T^1_{j_1}, T^1_{j_2} + c - T^2_{j_1}\} \leq 2(j_2 - j_1) + \frac{1}{2}c - d.$$

However, since S is a feasible schedule, we have that

$$T^2_{j_2} - T^1_{j_1} \geq \sum_{j=j_1}^{j_2} p_j + d \text{ and } T^1_{j_2} + c - T^2_{j_1} \geq \sum_{j=j_1}^{j_2} p_j + d.$$

Therefore, we get

$$\sum_{j=j_1}^{j_2} p_j + d \leq 2(j_2 - j_1) + \frac{1}{2}c - d,$$

which is in contradiction with (6.3).
Hence, $\overline{S}$ can be completed in such a way that a feasible schedule is obtained. Suppose that we complete $\overline{S}$ such that we get a feasible schedule. Then $\overline{S}$ is an optimal schedule but it does not necessarily have to be near-symmetric. However, if $\overline{S}$ is not near-symmetric, then we can make it near-symmetric by changing the schedule on machine 2 as follows. Make $\overline{S}^2_j$ equal to $\overline{S}^1_j + \frac{1}{2}c - d (1 \leq j \leq N)$. Obviously, now the schedule $\overline{S}$ is near-symmetric. Moreover, notice that now $\overline{T}^2_j = \overline{T}^1_j + \frac{1}{2}c - d \leq \overline{S}^1_j + \frac{1}{2}c - d = \overline{S}^2_j$ and that $\overline{S}^2_j + p_j + d = \overline{S}^1_j + \frac{1}{2}c - d + p_j + d \leq \overline{T}^2_j + \frac{1}{2}c - d = \overline{T}^1_j + c - 2d$, from which we conclude that $\overline{S}$ is still feasible. □

According to the above lemma, for N even and $E = 0$, we can restrict ourselves to near-symmetric schedules. Thus, in $MIP^2_{\text{cycl},3^0}$ we can replace the variables S^2_j and T^2_j by, respectively, $S^1_j + \frac{1}{2}c - d$ and $T^1_j + \frac{1}{2}c - d$. This results in a linear mixed-integer program with $2N + N(N-1)$ decision variables and $4N + 2N(N-1) + 1$ equations.

Moreover, since $E = 0$ we have $T^1_j = 2(j-1)d$ and therefore $T^2_j = (2j + N - 3)d$ $(1 \leq j \leq N)$. This implies that we can eliminate all equations that

contain the variables $x_{ij}(1 \leq j_1, j_2, \leq N, j_1 \neq j_2)$ from the linear mixed-integer program. Hence, also for N is even $MIP^2_{\text{cycl},3^0}$ is a linear program with $2N$ decision variables and $4N+1$ equations.

We see that, for both N is even as well as for N is odd, $MIP^2_{\text{cycl},3^0}$ is a linear programming problem. To stress this, we also write $LP^2_{\text{cycl},3^0}$.

6.4.3 Necessary conditions for the problem instance

Consider an optimal schedule S of structure $3^E(E \geq 0)$. Suppose that we move the schedule on machine 1 counterclockwise until there exists at least one task j for which $T^1_j = S^1_j$. Let task j_1 be one of these tasks. Now, tool $j_1 \ominus_N 1$ does not arrive at machine 1 before time $T^1_{j_1} - 2d - 2dE$ (see Figure 6.10). This implies

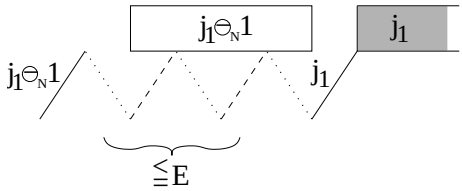


Figure 6.10. $p_{j_1 \ominus_N 1} \leq 2d + 2dE$.

that $p_{j_1 \ominus_N 1} \leq 2d + 2dE$. Following this reasoning also for the tools $j_1 \ominus_N 2$, $j_1 \ominus_N 3, \ldots, j_1 \ominus_N N$ gives

$$\sum_{i=1}^{k} p_{j_1 \ominus_N i} \leq 2d(k+E) \qquad (k = 1,2,\ldots,N).$$

We can also move the schedule on machine 1 clockwise until there exists a task j_2 for which $S^1_{j_2} + p_{j_2} + d = T^2_{j_2}$. Notice that tool $j_2 \oplus_N 1$ cannot arrive at machine 2 later than $T^2_{j_2} + 2d + 2dE$ (see Figure 6.11). This implies that $p_{j_2 \oplus_N 1} \leq 2d + 2dE$. By following the same kind of reasoning also for tool $j_2 \oplus_N 1, j_2 \oplus_N 2, \ldots, j_2 \oplus_N N$

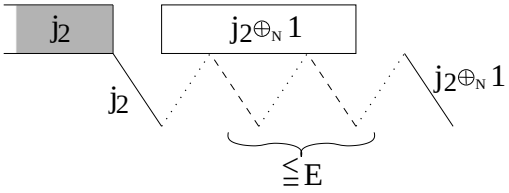


Figure 6.11. $p_{j_2 \oplus_N 1} \leq 2d + 2dE$.

we see that

$$\sum_{i=1}^{k} p_{j_2 \oplus_N i} \leq 2d(k+E) \qquad (k = 1,2,\ldots,N).$$

Also the above information cannot be expressed in equations with respect to the decision variables of $MIP^2_{\text{cycl},2}$. However, it will be useful in the next chapter.

6.4.4 Complexity status of the problem

If $N+E$ is odd and $E>0$ then $MIP^2_{\text{cycl},3^E}$ is still a linear mixed-integer program, but also in this case, it can be verified in polynomial time if an optimal schedule exists.

Theorem 6.2. *If there exists an optimal schedule of structure 3^E then this can be found in polynomial time if $N+E$ is odd, or if N is even and $E=0$.*

Proof. Let $N+E$ be odd or let N be even and $E=0$. Suppose that there exists an optimal schedule of structure 3^E. By the Lemma's 6.3 and 6.4 there exists a schedule $\bar{S}$ of structure 3^E, that is symmetric, if $N+E$ is odd, and near-symmetric, if N is even and $E=0$. Suppose that in $\bar{S}$ we move the processings, one by one, counterclockwise until no processing can be moved any more without making the schedule infeasible. Now in $\bar{S}$ there exists at least one task j for which $\bar{T}^1_j=\bar{S}^1_j$ and $\bar{T}^2_j=\bar{S}^2_j$. For simplicity of notation, we renumber the tasks such that this task is task 1. Notice that $\bar{S}$ is still symmetric if $N+E$ is odd and near-symmetric if N is even and $E=0$.
Based on the above, a feasible schedule S of structure 3^E that is symmetric if $N+E$ is odd and near-symmetric if N is even and $E=0$, can be found as follows (see Figure 6.12 for an example). We first place the processings and the transports concerning task 1 in S: let $T^1_1=S^1_1=0$ and let $T^2_1=S^2_1=\frac{1}{2}c$, if $N+E$ is odd and $T^2_1=S^2_1=\frac{1}{2}c-d$, if N is even and $E=0$ (see Figure 6.12(a)). After this, we iteratively add the processings and transports concerning the tasks $2,3,\ldots,N$ to schedule S: the processings and transports are placed as early as possible in S, taking into account that S has to become a feasible schedule of structure 3 (see Figure 6.12(b)-(d)). In other words, schedule S is found by applying the algorithm of Figure 6.13. Notice that for obtaining schedule S at most $N+E-1$ iterations have to be executed and that in these iterations no other loops occur. Thus, the algorithm of Figure 6.13 is a polynomial time algorithm. Since there are N different ways for choosing task 1, this implies that a schedule of structure 3^E can be found in polynomial time, if $N+E$ is odd or if N is even and $E=0$. □

6.5 Structure 4

We now consider schedules with critical cycles that consist of one block on machine 1 and some transports.

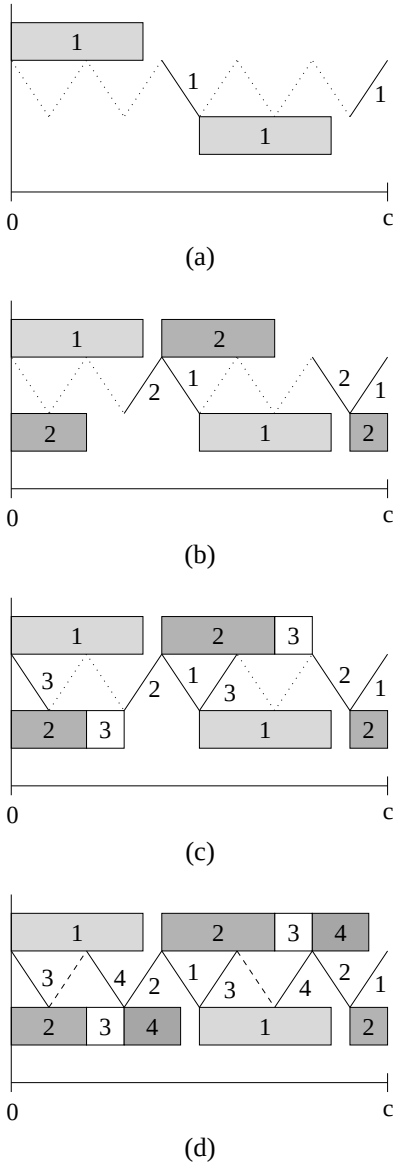


Figure 6.12. Executing the algorithm for $(\boldsymbol{p},d)=(7,6,2,3),2)$ and $E=1$: (a) After initialization, (b) After 1 iteration, (c) After 2 iterations (d) Final schedule.

6.5.1 The linear mixed-integer program

Consider a schedule S of structure 4. Let C be a critical cycle of S that consists of one block on machine 1 and $2k$ transports. Suppose that the block of C starts

Step 1 *Initialization*

$$\lambda = \begin{cases} 0 & \text{if } N+E \text{ is odd,} \\ 1 & \text{if } N \text{ is even and } E = 0, \end{cases}$$

$c = 2(N+E)d$,
$T_1^1 = S_1^1 = 0$,
$T_1^2 = S_1^2 = \frac{1}{2}c - \lambda d$,
$j = 1$,
$k = 1$.

Step 2 *Iteration*

WHILE $j \leq N$ AND $k < N+E$
{ $T_{j+1}^1 = 2kd$,
$S_{j+1}^1 = \max\{S_j^1 + p_j, T_{j+1}^1\}$,
$T_{j+1}^2 = T_{j+1}^1 + \frac{1}{2}c - \lambda d$,
$S_{j+1}^2 = S_{j+1}^1 + \frac{1}{2}c - \lambda d$.
Check if $S_{j+1}^1 + p_{j+1} + d \leq T_{j+1}^2$ and $S_{j+1}^2 + p_{j+1} + d \leq T_{j+1}^1 + c$.
IF yes THEN j=j+1,
k=k+1.
}

Step 3 *End*

IF $j > N$ THEN
A feasible schedule of structure 3^E is obtained, that is
symmetric if $N+E$ is odd and near-symmetric if $N+E$ is even.
ELSE
No feasible schedule of structure 3^E exists.

Figure 6.13. Polynomial time algorithm for verifying the existence of a schedule of structure 3^E ($N+E$ odd or $E = 0$).

with the processing of task j_1 on machine 1 and that it ends with the processing of task j_2 on machine 1 (see Figure 6.14). Without loss of generality, we assume that $j_1 \leq j_2$. Now for S the following relations hold:

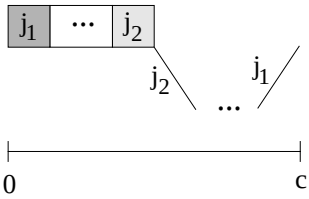


Figure 6.14. A critical cycle of structure 4.

$$\begin{aligned} T^1_{j_1} &= S^1_{j_1}, \\ T^2_{j_2} &= S^1_{j_2} + p_{j_2} + d, \\ S^1_j &= S^1_{j-1} + p_{j-1} \qquad (j = j_1+1, \ldots, j_2), \\ c &= \textstyle\sum_{j=j_1}^{j_2} p_j + 2kd, \end{aligned}$$

This corresponds to several equality signs and, given j_1, j_2 and k, a prescribed objective function value in Problem formulation 4 of Chapter 2. The corresponding linear mixed-integer program, we call $MIP^2_{\text{cycl},4}(\boldsymbol{j},k)$, where $\boldsymbol{j} = (j_1, j_2)$. Given $\boldsymbol{j}$ and k, $MIP^2_{\text{cycl},4}(\boldsymbol{j},k)$ is a feasibility problem. It contains $4N + N(N-1)$ decision variables and $8N + 2N(N-1) + 1$ equations. Notice that a feasible schedule of $MIP^2_{\text{cycl},4}(\boldsymbol{j},k)$ does not necessarily constitute an optimal schedule for MIP^2_{cycl}.

There are N choices for both j_1 and j_2. Furthermore, because of Assumption 2.4, the number of empty transports in a schedule is never more than $2(N-1)$. This implies that the total number of transports in a schedule is at most $4N-2$ and therefore, $k \leq 2N-1$.

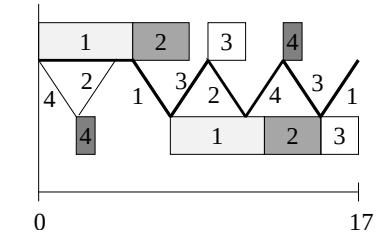
Hence, we have a polynomial number of parameter combinations (j_1, j_2, k). Thus, if for any of these parameter combinations $MIP^2_{\text{cycl},4}(\boldsymbol{j},k)$ can be solved in polynomial time, then the existence of an optimal schedule of structure 4 can also be verified in polynomial time.

6.5.2 Symmetry

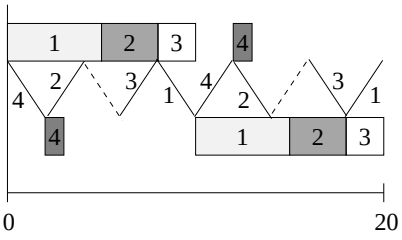
The existence of an optimal schedule of structure 4 does not imply the existence of a symmetric or a near-symmetric optimal schedule (see Figure 6.15). Figure 6.15(a) shows an optimal schedule for problem instance $(\boldsymbol{p},d) = ((5,3,2,1),2)$. The Figures 6.15(b) and (c), respectively, show an optimal symmetric and an optimal near-symmetric schedule for this problem instance.

6.5.3 Necessary conditions for the problem instance

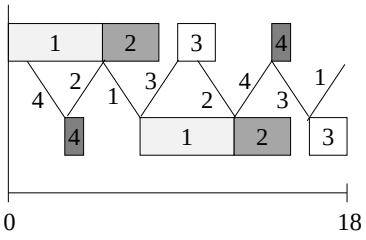
Consider again the schedule S of Section 6.5.1. Since the processings that are no part of the block of C all have to fit on machine 1 in the available time, we have



(a)



(b)



(c)

Figure 6.15. For $(\boldsymbol{p},d) = (2,(5,3,2,1))$, (a) an optimal schedule, (b) an optimal symmetric schedule and (c) an optimal near-symmetric schedule.

that

$$\sum_{j=1}^{N} p_j - \sum_{j=j_1}^{j_2} p_j \leq 2kd.$$

Moreover, in view of the Lemma's 4.1 and 4.2 we have

$$\sum_{i=0}^{m} p_{j_1 \oplus_N i} \geq 2d(m+1) \qquad (m = 0,1,\ldots,j_2 - j_1)$$

and

$$\sum_{i=0}^{m} p_{j_2 \ominus_N i} \geq 2d(m+1) \qquad (m = 0,1,\ldots,j_2 - j_1).$$

In the interval $(T^1_{j_1}, T^2_{j_2})$ there can finish at most $x := \lfloor \frac{\sum_{j=j_1}^{j_2} p_j}{2d} \rfloor$ transports to machine m $(m = 1,2)$. This implies that at least $N - x$ transports to machine m finish in $[T^2_{j_2}, T^1_{j_1} + c](m = 1,2)$. This is an interval of length $2kd$. Therefore, we must have that

$$k \geq N - x.$$

Furthermore, in $[T^2_{j_2}, T^1_{j_1} + c]$ there are at most $k - (N - x)$ empty transports to machine 1. This implies that the transport of the tool belonging to task $j_2 \oplus_N 1$ to machine 2 has to finish before $T^2_{j_2} + 2(k - (N - x)) + 2d$ (see Figure 6.16). Therefore,

$$p_{j_2 \oplus_N 1} \leq 2d + 2d(k - (N - x)).$$

Following this reasoning also for the tasks $j_2 \oplus_N 2, \ldots, j_2 \oplus_N (N - x - 1)$ shows

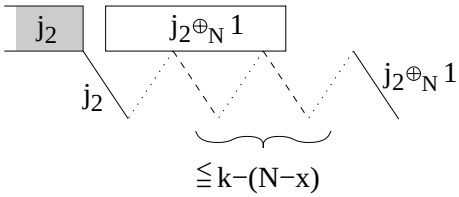


Figure 6.16. $p_{j_2 \oplus_N 1} \leq 2d + 2d(k - (N - x))$.

that

$$\sum_{i=1}^{m} p_{j_2 \oplus_N i} \leq 2d(m + k - (N - x)) \qquad (m = 1, 2, \ldots, N - x - 1).$$

Moreover, because in $[T^2_{j_2}, T^1_{j_1} + c]$ there are also at most $k - (N - x)$ empty transports to machine 2, we have, as can be seen in Figure 6.17, that

$$\sum_{i=1}^{m} p_{j_1 \ominus_N i} \leq 2d(m + k - (N - x)) \qquad (m = 1, 2, \ldots, N - x - 1).$$

Again, the above information cannot be incorporated directly in $MIP^2_{\text{cycl},4}(\boldsymbol{j}, k)$, but will be useful in the next chapter.

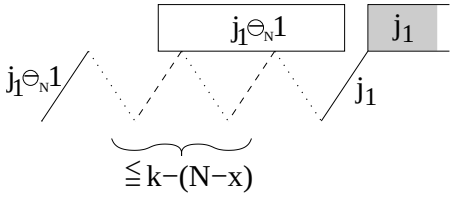


Figure 6.17. $p_{j_1 \ominus_N 1} \leq 2d + (k - (N - x))$.

6.5.4 Complexity status of the problem

The complexity of the problem $P^2_{\text{cycl},4}$ is still not known.

6.6 Structure 5

The last schedules to be considered are the ones that contain a critical cycle with two blocks on machine 1.

6.6.1 The linear mixed-integer program

Consider a schedule S of structure 5. Let C be a critical cycle of S that contains two blocks on machine 1. Let the first block of C start with a processing of task j_1 and end with a processing of task j_2 and let the second block begin with a processing of task j_3 and finish with a processing of a task j_4 (see Figure 6.18). Without

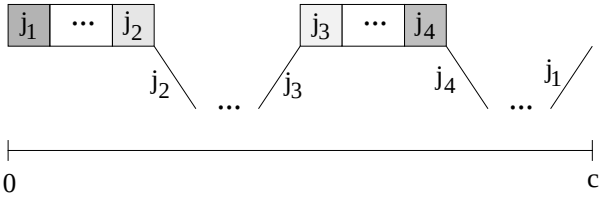


Figure 6.18. A critical cycle of structure 5.

loss of generality, we assume that $j_1 \leq j_2 \leq j_3 \leq j_4$. Suppose that C contains $2k_1$ transports that finish in the interval $[S^1_{j_2} + p_{j_2} + d, S^1_{j_3}]$ and $2k_2$ that finish in the interval $[S^1_{j_4} + p_{j_4} + d, S^1_{j_1} + c]$.

For schedule S the following equations hold:

$$\begin{aligned}
T^1_{j_1} &= S^1_{j_1}, \\
T^2_{j_2} &= S^1_{j_2} + p_{j_2} + d, \\
T^1_{j_3} &= S^1_{j_3}, \\
T^2_{j_4} &= S^1_{j_4} + p_{j_4} + d, \\
S^1_j &= S^1_{j-1} + p_{j-1} \qquad (j = j_1+1, \ldots, j_2, j_3+1, \ldots, j_4)
\end{aligned}$$

and

$$c = \sum_{j=j_1}^{j_2} p_j + \sum_{j=j_3}^{j_4} p_j + 2(k_1+k_2)d.$$

This information corresponds to several equality signs and, given the values of j_1, j_2, j_3, j_4, k_1 and k_2, a prescribed objective function value in Problem formulation 4. Moreover, for all $i_1 \in \{j_1, \ldots, j_2\}$ and $i_2 \in \{j_3, \ldots, j_4\}$ we have that $T^1_{i_2} - c < T^2_{i_1} < T^1_{i_2}$ and $T^1_{i_1} < T^2_{i_2} < T^1_{i_1} + c$, and consequently, $x_{i_1 i_2} = x_{i_2 i_1} = 0$.

The linear mixed-integer program that we obtain by adding this information to MIP^2_{cycl}, we call $MIP^2_{\text{cycl},5}(\boldsymbol{j}, \boldsymbol{k})$, where $\boldsymbol{j} = (j_1, j_2, j_3, j_4)$ and $\boldsymbol{k} =$

(k_1,k_2). Also $MIP^5_{\text{cycl}}(\boldsymbol{j},\boldsymbol{k})$ is a feasibility problem. It contains $4N+N(N-1)-2(j_2-j_1+1)(j_4-j_3+1)$ decision variables and $8N+2N(N-1)+1$ equations. A feasible schedule of $MIP^2_{\text{cycl},5}(\boldsymbol{j},\boldsymbol{k})$ does not necessarily have to be an optimal schedule for MIP^2_{cycl}.

Again, there exist a polynomial number of parameter combinations $(\boldsymbol{j},\boldsymbol{k})$. Thus, if for any of these parameter combinations $MIP^2_{\text{cycl},5}(\boldsymbol{j},\boldsymbol{k})$ can be solved in polynomial time, the existence of an optimal schedule of structure 5 can also be verified in polynomial time.

6.6.2 Symmetry

As can be seen in Figure 6.19 the existence of an optimal schedule of structure 5 does not have to imply the existence of a symmetric or near-symmetric optimal schedule. For problem instance $(\boldsymbol{p},d)=((3,2),3)$, Figure 6.19(a) shows an optimal schedule, which is of structure 5. Figure 6.19(b) shows an optimal symmetric schedule and Figure 6.19(c) an optimal near-symmetric schedule.

6.6.3 Necessary conditions for the problem instance

Consider the schedule S of Section 6.6.1. In a similar way as in Section 6.5.3, it can be proved that

$$\sum_{j=j_2+1}^{j_3-1} p_j \leq 2k_1d,$$

$$\sum_{j=1}^{N} p_j - \sum_{j=j_1}^{j_4} p_j \leq 2k_2d,$$

$$\sum_{i=0}^{m} p_{j_1+i} \geq 2d(m+1) \qquad (m=0,1,\ldots,j_2-j_1),$$

$$\sum_{i=0}^{m} p_{j_2-i} \geq 2d(m+1) \qquad (m=0,1,\ldots,j_2-j_1),$$

$$\sum_{i=0}^{m} p_{j_3+i} \geq 2d(m+1) \qquad (m=0,1,\ldots,j_4-j_3) \text{ and that}$$

$$\sum_{i=0}^{m} p_{j_4-i} \geq 2d(m+1) \qquad (m=0,1,\ldots,j_4-j_3).$$

Consider the j_3-j_1 tools that arrive at machine 1 in the interval $(T^1_{j_1},T^1_{j_3}]$. At most $x_1 := \lfloor \frac{\sum_{j=j_1}^{j_2} p_j}{2d} \rfloor$ of these tools arrive at machine 1 in $(T^1_{j_1},T^2_{j_2}-d]$. Hence, in the interval $(T^2_{j_2}-d,T^1_{j_3}]$ there are at least $j_3-j_1-x_1$ nonempty tool transports to

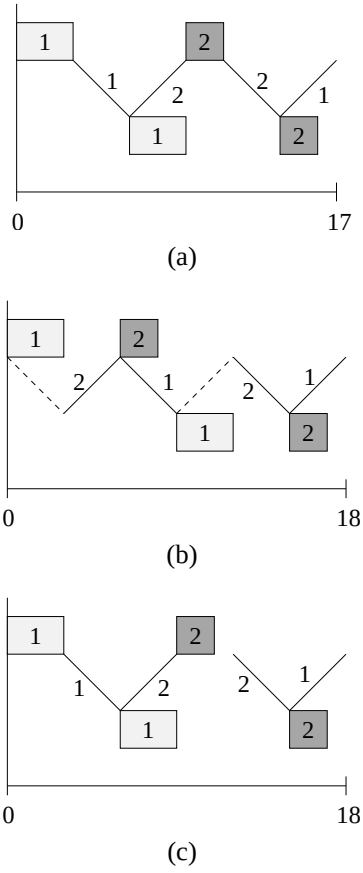


Figure 6.19. For $(\boldsymbol{p},d)=((3,2),3)$: (a) an optimal schedule, (b) an optimal symmetric schedule and (c) an optimal near-symmetric schedule.

machine 1. Since the interval $[T^2_{j_2}-d,T^1_{j_3}]$ is of length $2k_1d$ we have that

$$k_1 \geq j_3 - j_1 - x_1.$$

Furthermore, there are at most $k_1-(j_3-j_1-x_1)$ empty transports to machine 1 that finish in $(T^2_{j_2}-d,T^1_{j_3}]$. Therefore,

$$\sum_{i=1}^{m} p_{j_3-i} \leq 2d(k_1-(j_3-j_1-x_1)+m) \qquad (m=1,2,\ldots,j_3-j_1-x_1-1).$$

Let $x_2 := \lfloor \frac{\sum_{j=j_3}^{j_4} p_j}{2d} \rfloor$. Following the same reasoning for the tools that arrive at machine 1 in $(T^1_{j_3},T^1_{j_1}+c]$, we get

$$k_2 \geq j_1 + N - j_3 - x_2.$$

and

$$\sum_{i=1}^{m} p_{j_1 \oplus_N i} \leq 2d(k_2 - (j_1 + N - j_3 - x_2) + m) \qquad (m = 1, 2, \ldots, j_1 + N - j_3 - x_2 - 1).$$

Now consider the $j_4 - j_2$ tools that arrive at machine 2 in the interval $[T^2_{j_2}, T^2_{j_4})$. At most x_2 of these tools arrive at machine 2 in $[T^1_{j_3}, T^2_{j_4})$. Hence, in the interval $[T^2_{j_2}, T^1_{j_3})$ there are at least $j_4 - j_2 - x_2$ nonempty tool transports to machine 2. Since the interval $[T^2_{j_2}, T^1_{j_3}]$ is of length $2k_1 d$, we have that

$$k_1 \geq j_4 - j_2 - x_2.$$

There are at most $k_1 - (j_4 - j_2 - x_2)$ empty transports to machine 2 in $[T^2_{j_2}, T^1_{j_3})$. Therefore,

$$\sum_{i=1}^{m} p_{j_2 + i} \leq 2d(k_1 - (j_4 - j_2 - x_2) + m) \qquad (m = 1, 2, \ldots, j_4 - j_2 - x_2 - 1).$$

Analogously, by considering the tools that arrive at machine 2 in the interval $[T^2_{j_4}, T^2_{j_2} + c)$ we find that

$$k_2 \geq j_2 + N - j_4 - x_1.$$

and that

$$\sum_{i=1}^{m} p_{j_4 \oplus_N i} \leq 2d(k_2 - (j_2 + N - j_4 - x_1) + m) \qquad (m = 1, 2, \ldots, j_2 + N - j_4 - x_1 - 1).$$

The above information cannot be directly incorporated in $MIP^2_{\text{cycl},5}(\boldsymbol{j}, \boldsymbol{k})$, but will be used in the next chapter.

6.6.4 Complexity status of the problem

The complexity of the problem $P^2_{\text{cycl},5}$ is still unknown.

6.6.5 Summary

In this chapter we have defined the following linear-mixed integer programs:

(a) $MIP^2_{\text{cycl},1}$, a linear mixed-integer program with $2N + \frac{1}{2}N(N-1)$ decision variables and $4N + N(N-1) + 1$ equations, for verifying the existence of an optimal schedule of structure 1.

(b) $MIP^2_{\text{cycl},2}$, a linear mixed-integer program with $4N + N(N-1)$ decision variables and $8N + 2N(N-1) + 1$ equations which can be used for checking whether there exists an optimal schedule of structure 2.

(c) $MIP^2_{\text{cycl},3^0} = LP^2_{\text{cycl},3^0}$, a linear programming problem with $2N$ decision variables and $4N + 1$ equations. It can be used for verifying the existence of an optimal schedule of structure 3^0.

(d) $MIP^2_{\text{cycl},3^E}$, a linear mixed-integer program with $2N+\frac{1}{2}N(N-1)$ decision variables and $4N+N(N-1)+1$ equations, if $N+E$ is odd and a linear mixed-integer program with $4N+N(N-1)$ decision variables and $8N+2N(N-1)+1$ equations, if $N+E$ is even. $MIP^2_{\text{cycl},3^E}$ can be used, given E, for verifying the existence of an optimal schedule of structure 3^E.

(e) $MIP^2_{\text{cycl},4}(\boldsymbol{j},k)$, a linear mixed-integer program with $4N+N(N-1)$ decision variables and $8N+2N(N-1)+1$ equations, which can be used for answering the question whether, given $\boldsymbol{j}$ and k, there exists an optimal schedule of structure 4.

(f) $MIP^2_{\text{cycl},5}(\boldsymbol{j},\boldsymbol{k})$, a linear mixed-integer program with $4N+N(N-1)-2(j_2-j_1+1)(j_4-j_3+1)$ decision variables and $8N+2N(N-1)+1$ equations that can verify, given $\boldsymbol{j}$ and $\boldsymbol{k}$, the existence of an optimal schedule of structure 5.

We have seen that an optimal schedule of structure 1 or structure 3^0 can be found in polynomial time and that an optimal schedule of structure $3^E, E>0$ can be found in polynomial time if $N+E$ is odd. However, whether or not an optimal schedule of structure 2, 3^E where $E>0$ and $N+E$ is even, 4 or 5 can be found in polynomial time is still an open question.

In other words, we are still not able to solve all the problems $P^2_{\text{cycl,i}}$ in polynomial time and, thus, we still do not know if problem P^2_{cycl} is solvable in polynomial time.

However, based on the considerations in this and the previous chapter, it is not very difficult to use *dynamic programming* techniques for developing an algorithm for P^2_{cycl}, the running time of which is bounded by a polynomial in N and $p_{\max}$. This is a so-called *pseudo-polynomial* algorithm. The states in the dynamic program can, e.g., be chosen in such a way that they correspond to the times at which the robot finishes a nonempty transport. They can be characterized by the distribution of the tools over the machines, the tasks in process on the machines, the remaining processing times of these tasks, and the index of the machine at which the robot arrives.

In the next chapter, we will develop algorithms for tackling problem P^2_{cycl}. We will not use the above described method based on dynamic programming, but we will use the information that we have obtained about schedules of the five different structures.

7

Experimental results

7.1 Introduction

In Chapter 5 we have seen that for finding an optimal schedule we need only consider schedules of five different structures, and in the previous chapter we have looked at these structures in more detail. In this chapter, we use the knowledge that we have obtained about the structures for the development of some algorithms for tackling problem P^2_{cycl}.

We start by describing three different algorithms. The first of these algorithms simply consists of solving the linear mixed-integer problem of Chapter 2 and does not use any of the information obtained in the Chapters 3-6. It is meant as a reference algorithm. The other two algorithms use the upper and lower bounds of Chapter 3 and only take into account schedules of the structures 1-5.

With the three algorithms, several experiments are carried out. It turns out that the more intelligent algorithms are often much faster than the reference algorithm, but that they are much slower in other cases. Therefore, a fourth algorithm is developed that contains even more of the information given in the previous chapters. In general, this algorithm turns out to be much faster than the reference algorithm.

7.2 Algorithms

The first algorithm that we consider simply consists of solving the linear mixed-integer program of Chapter 2 (see Figure 7.1). For doing this, we use the Mixed Integer Solver of CPLEX. CPLEX is a commercial software package that solves linear mixed-integer programs by means of branch-and-bound techniques. Algorithm

---

Algorithm 1

Step 1 Solve MIP^2_{cycl}.

Figure 7.1. Description of Algorithm 1.

1 does not contain any of the information obtained in the Chapters 3-6. Nevertheless, Algorithm 1 is of interest, since the results that we obtain with this algorithm can be used as a reference for evaluating the results that we find with other algorithms.

The first algorithm based on the information of the Chapters 3-6 is described in Figure 7.2. In this figure $c_{\text{opt}}(MIP^2_{\text{cycl,i}})$ denotes the optimal cycle time of $MIP^2_{\text{cycl,i}}$ $(i = 1,2,3^E,4,5)$ and $c_{\text{opt}}(LP^2_{\text{cycl},3^0})$ denotes the optimal cycle time of $LP^2_{\text{cycl},3^0}$.

The algorithm makes use of the lower bounds $2Nd, 2p_{\max}+2d, \sum_{j=1}^{N} p_j$ and the optimal solution of the LP-relaxation of MIP^2_{cycl}, and of the upper bound $2(Nd + \sum_{j=1}^{N} p_j)$. Moreover, it only considers schedules of the structures 1-5.

It roughly works as follows. It first examines, if the maximum of the lower bounds is equal to the upper bound (see Step 1). If this is the case, then an optimal solution is already known and the algorithm stops. If there is a gap between the best lower bound and the upper bound then the algorithm examines whether or not there exists a feasible schedule with a cycle time equal to the maximum of the lower bounds $2Nd, 2p_{\max}+2d$ and $\sum_{j=1}^{N} p_j$ (see Step 2). Depending on the value of the maximum of the lower bounds, the algorithm solves $MIP^2_{\text{cycl},1}$, $MIP^2_{\text{cycl},2}$, $LP^2_{\text{cycl},3^0}$ or no linear mixed-integer program. The last occurs when the value of lower bound $c_{\text{opt}}(LP_{\text{relax}})$ is greater than $\max\{2Nd, 2p_{\max}+2d, \sum_{j=1}^{N} p_j\}$. If it is possible that there exists an optimal schedule of structure 3^0, then $LP^2_{\text{cycl},3^0}$ is solved. If there cannot be an optimal schedule of structure 3^0 but there may exist an optimal schedule of structure 1, then $MIP^2_{\text{cycl},1}$ is solved. If no optimal schedule of the structures 1 and 3^0 can exist and there may exist an optimal schedule of structure 2, then $MIP^2_{\text{cycl},2}$ is solved. (This order is not coincidental, it is chosen like this because from the three linear mixed-integer programs $LP_{\text{cycl},3^0}$ is expected

to be the easiest to solve and $MIP_{\text{cycl},2}$ the hardest.) If a feasible schedule is found, then it is also an optimal schedule of MIP^2_{cycl} and the algorithm stops.

If no feasible schedule with a cycle time equal to $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N} p_j\}$ is found, the Steps 3,4 and 5 of the algorithm are carried out. In these steps, respectively, schedules of the structures 3^E,4 and 5 are considered. In Step 3 the best schedule of structure $3^E (E > 0)$ is searched for, at least if this schedule has an interesting cycle time, i.e., if its corresponding cycle time is in between the current values of the lower and upper bound. First the smallest value of E is determined for which the value of the current lower bound is exceeded. Then $MIP^2_{\text{cycl},3^E}$ is solved. If this results in a feasible solution, then this is the best solution of structure 3^E and the algorithm goes on with Step 4. If no feasible solution is found, then E is increased by 1. This process continues until either a feasible schedule is found or E becomes too large to be of interest.

In Step 4 the best schedule of structure 4 is searched for. In this step, all possible combinations of j_1 and j_2 are considered. For every combination, first the smallest value of k is determined for which the corresponding objective function value is at least equal to the current value of the lower bound but below the current upper bound. Then, $MIP^2_{\text{cycl},4}(\boldsymbol{j}, k)$ is solved. If this leads to a feasible schedule, then this is the best schedule that can be obtained for the corresponding combination of j_1 and j_2 and the cycle time belonging to the schedule constitutes a new upper bound. If no feasible solution is found, then k is increased by 1. This process continues until either a feasible schedule is found or k becomes too large to be interesting. Step 4 stops when all different combinations of j_1 and j_2 have been examined.

Step 5 is somehow similar to Step 4. Here, all different combinations of j_1, j_2, j_3 and j_4 for which $j_1 \leq j_2 < j_3 \leq j_4$ or $j_4 < j_1 \leq j_2 < j_3$ are examined. For every combination, a feasible schedule with a minimal cycle time between the current values of the lower and the upper bound is searched for. Every time such a feasible schedule is found, its corresponding cycle time constitutes a better upper bound on the optimal cycle time length. When all the combinations of j_1, j_2, j_3 and j_4 have been examined, the algorithm finishes.

For solving the different linear-mixed integer problems in the Steps 2-5 of the algorithm CPLEX is used.

The next algorithm that we consider is almost equal to Algorithm 2 (see Figure 7.3). The only difference between the algorithms is found in Step 1. Here, Algorithm 3 also calculates the upper bound that is found by solving the linear programming problem that results from using the transport strategies described in Section 3.5.3. More precisely, for N odd, we use transport strategy

$$(,T_1^1)_t, (,T^2_{\frac{N+3}{2}} - c)_t, (,T_2^1)_t, (,T^2_{\frac{N+5}{2}} - c)_t, \ldots, (,T^1_{\frac{N+1}{2}})_t, (,T_1^2)_t, \ldots, (,T_N^1)_t, (,T^2_{\frac{N+1}{2}})_t.$$

Algorithm 2

Step 1 *Initialization*
LowerBound=max$\{2Nd, 2p_{\max}+2d, \sum_{j=1}^{N} p_j, c_{\text{opt}}(LP_{\text{relax}})\}$.
UpperBound= $2(Nd+\sum_{j=1}^{N} p_j)$.
BestSolution=∞.
OptimalSolution=∞.
IF UpperBound=LowerBound THEN
{ BestSolution=LowerBound
GOTO Step 6
}

Step 2 *Searching for a schedule of length* $\max\{2Nd, 2p_{\max}+2d, \sum_{j=1}^{N} p_j\}$
IF LowerBound= $2Nd$ THEN
{ Solve $LP^2_{\text{cycl},3^0}$.
IF feasible THEN
{ BestSolution= $[c_{\text{opt}}(LP^2_{\text{cycl},3^0})]^{-1}$,
GOTO Step 6.
}
}
ELSE IF LowerBound= $2p_{\max}+2d$ THEN
{ Solve $MIP^2_{\text{cycl},1}$.
IF feasible THEN
{ BestSolution= $[c_{\text{opt}}(MIP^2_{\text{cycl},1})]^{-1}$,
GOTO Step 6.
}
}
ELSE IF LowerBound= $\sum_{j=1}^{N} p_j$ THEN
{ Solve $MIP^2_{\text{cycl},2}$.
IF $MIP^2_{\text{cycl},2}$ = feasible THEN
{ BestSolution= $[c_{\text{opt}}(MIP^2_{\text{cycl},2})]^{-1}$,
GOTO Step 6.
}
}

Step 3 *Searching for the best schedule of structure* $3^E\,(E>0)$
$E=\min\{\bar{E}|2(N+\bar{E})d \geq \text{LowerBound}\}$.
WHILE $2(N+E) \leq$ UpperBound
{ Solve $MIP^2_{\text{cycl},3^E}$.
IF feasible THEN
{ BestSolution= $[c_{\text{opt}}(MIP^2_{\text{cycl},3^E})]^{-1}$,

Figure 7.2. Description of Algorithm 2 (Part 1).

UpperBound=BestSolution,
$E = \infty$.
}
ELSE E=E+1.
}

Step 4 *Searching for the best schedule of structure 4*
FOR every $\boldsymbol{j} = (\boldsymbol{j_1}, \boldsymbol{j_2})$
{ $k = \min\{\bar{k} | \sum_{j=j_1}^{j_2} p_j + 2\bar{k}d \geq \text{LowerBound}\}$.
WHILE $\sum_{j=j_1}^{j_2} p_j + 2kd \leq \text{UpperBound}$
{ Solve $MIP^2_{\text{cycl},4}(\boldsymbol{j}, k)$.
IF feasible THEN
{ BestSolution= $[c_{\text{opt}}(MIP^2_{\text{cycl},4}(\boldsymbol{j}, k))]^{-1}$,
UpperBound=BestSolution,
$k = \infty$.
}
ELSE $k = k + 1$.
}
}

Step 5 *Searching for the best schedule of structure 5*
FOR every $\boldsymbol{j} = (\boldsymbol{j_1}, \boldsymbol{j_2}, \boldsymbol{j_3}, \boldsymbol{j_4})$ for which $j_1 \leq j_2 < j_3 \leq j_4$ or $j_4 < j_1 \leq j_2 < j_3$
{ $k = \min\{\bar{k} | \sum_{j=j_1}^{j_2} p_j + \sum_{j=j_3}^{j_4} p_j + 2\bar{k}d \geq \text{LowerBound}\}$.
WHILE $\sum_{j=j_1}^{j_2} p_j + \sum_{j=j_3}^{j_4} p_j + 2kd \leq \text{UpperBound}$
{ $k_1 = 1$,
WHILE $k_1 <= k - 1 < \infty$
{ Solve $MIP^2_{\text{cycl},5}(\boldsymbol{j}, \boldsymbol{k_1}, \boldsymbol{k} - \boldsymbol{k_1})$.
IF feasible THEN
{ BestSolution= $[c_{\text{opt}}(MIP^2_{\text{cycl},5}(\boldsymbol{j}, \boldsymbol{k_1}, \boldsymbol{k} - \boldsymbol{k_1}))]^{-1}$,
UpperBound=BestSolution,
$k_1 = k = \infty$,
}
ELSE IF $k_1 < k - 1$ THEN
$k_1 = k_1 + 1$.
ELSE { $k = k + 1$,
$k_1 = \infty$.
}
}
}
}

Step 6 *End*
OptimalSolution=BestSolution.

Figure 7.2. Description of Algorithm 2 (Part 2).

and for N even, we use

$$(,T_1^1)_t,(,T_{\frac{N+4}{2}}^2-c)_t,(,T_2^1)_t,(,T_{\frac{N+6}{2}}^2-c)_t,\ldots,(,T_{\frac{N}{2}}^1)_t,(,T_1^2)_t,\ldots,(,T_N^1)_t,(,T_{\frac{N+2}{2}}^2)_t.$$

We call the linear programming problem that results from adding these prescribed transport strategy LP_{trans}. The optimal solution of LP_{trans} we denote by $c_{\text{opt}}(LP_{\text{trans}})$.

Algorithm 3
Replace in Algorithm 2 "UpperBound= $2(Nd+\sum_{j=1}^{N}p_j)$" by:
UpperBound= $\min\{2(Nd+\sum_{j=1}^{N}p_j),c_{\text{opt}}(LP_{\text{trans}})\}$.

Figure 7.3. Description of Algorithm 3.

7.3 Comparison of the Algorithms 1,2 and 3

In this section we present numerical results obtained with the three algorithms described in the previous section.

For obtaining these results we devised several experiments. In every experiment the algorithms are used for solving problem P_{cycl}^2 for a certain test set of problem instances. Such a test set is characterized by a tuple $(\mathcal{N},UB_p,n_{\text{inst}})$ and some additional characteristics that depend on the experiment. Here,

$\mathcal{N}$ is a set of natural numbers from which the number of tasks N is chosen,

UB_p is a set of real values that represent upper bounds on the values of the processing times of the tasks. If for a certain problem instance upper bound $ub_p \in UB_p$ is selected, this means that the processing times of the tasks are chosen randomly in the interval $[1,ub_p]$. (In some experiments all processing times but one are chosen in this interval.) The transport time d of the robot is also chosen in the interval $[1,ub_p]$.

n_{inst} is the number of problem instances that are generated for every combination of N,ub_p and d.

7.3.1 Experiment 1

Description

We take $\mathcal{N}=\{10,25,50,75\},UB_p=\{5,10,25\}$ and $n_{\text{inst}}=5$. For transport time d several values in the interval $[1,ub_p]$ are chosen.

Results

The results of Experiment 1 are shown in the Table B.1 of Appendix B. The columns of the table respectively indicate:

1. the number of tasks N,

2. the interval from which the processing times $p_i (1 \leq j \leq N)$ are chosen,
3. the value of the transport time d.
4. the average execution time (in seconds of CPU time) and the corresponding standard deviation for Algorithm 1,
5. the average execution time (in seconds of CPU time) and the corresponding standard deviation for Algorithm 2,
6. the average execution time (in seconds of CPU time) and the corresponding standard deviation for Algorithm 3,
7. the number of executions in which the optimal cycle time of LP_{relax} is equal to $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N} p_j\}$,
8. the number of executions in which the optimal cycle time of LP_{relax} is equal to the optimal cycle time of LP_{trans},
9. the number of executions in which the optimal cycle time of LP_{trans} is optimal.
10. the structure of the generated optimal schedules.

A number between round brackets indicates the number of experiments in which CPLEX returns "solution status 102". This means that an optimal solution within a certain tolerance is found. More precisely, when the value of

$$\frac{|\text{objective value best node - objective value best integer node}|}{1 + |\text{objective value best node}|}$$

falls below 10^{-4}, the algorithm stops.

A number between square brackets ($[\,]$) indicates the number of executions in which CPLEX does not manage to give an optimal solution because of shortage of memory. In this case the average and the standard deviation are calculated over the executions in which no memory problems occur.

The results of Table B.1 can be summarized as follows:

1. Experiments that are carried out with the same values for N, ub_p and d may have very different execution times. The standard deviations can sometimes even be larger than the corresponding average execution times. However, the standard deviation can also be much smaller than the average.
2. We see that for all instances an optimal solution of structure 2 or structure 3^0 is found. If $2Nd$ is larger than $\sum_{j=1}^{N} p_j$ then there exists an optimal schedule of structure 3^0, i.e., an optimal schedule length $2Nd$. If $2Nd$ is smaller than $\sum_{j=1}^{N} p_j$ then there exists an optimal schedule of structure 2, which is of length $\sum_{j=1}^{N} p_j$.

3. Algorithm 2 is almost always faster than Algorithm 1, in general much faster. This is especially true if there exists an optimal schedule of structure 3^0. This was to be expected, because for finding an optimal schedule of 3^0 Algorithm 1 needs to solve a linear mixed-integer problem with $4N + N(N-1)$ decision variables and $8N + 2N(N-1) + 1$ equations, and Algorithm 2 a linear programming problem with $2N$ decision variables and $4N + 1$ equations. For finding an optimal schedule of structure 2, both algorithms solve a linear mixed-integer program with $4N + N(N-1)$ variables and $8N + 2N(N-1) + 1$ equations.
4. In all the cases in which the average execution time of Algorithm 1 is smaller than the average execution time of Algorithm 2, Algorithm 1 ended in some of the executions with "solution status 102". This may be an indication that in these executions Algorithm 1 stopped too early.
5. If there exists an optimal schedule of structure 2, then this is found much faster with Algorithm 3 than with Algorithm 2, especially for larger values of N. This is, because the LP_{trans} that is solved in the beginning of Algorithm 3, always results in a feasible and, therefore, optimal schedule. Thus, for finding an optimal schedule of structure 2, Algorithm 3 solves a linear programming problem with $4N$ decision variables and $8N + 2N(N-1) + 1$ equations, while Algorithm 2 solves a linear mixed-integer program with $4N + N(N-1)$ variables and $8N + 2N(N-1) + 1$ equations.
6. In case there exists an optimal schedules of structure 3^0, Algorithm 2 is faster than Algorithm 3. This is explained by the fact that for finding an optimal schedule of structure 3^0 Algorithm 2 solves a linear program with $2N$ decision variables and $4N + 1$ equations while Algorithm 3 solves a linear program with $4N$ decision variables and $8N + 2N(N-1) + 1$ equations.
7. The results obtained with the Algorithms 2 and 3 are more reliable than the ones obtained with Algorithm 1. If Algorithm 1 is used for larger N, then CPLEX often returns "solution status 102".
8. Algorithm 1 sometimes fails to calculate an optimal schedule because of shortage of memory.
9. We always find that the value of lower bound $c_{\text{opt}}(LP_{\text{relax}})$ is equal to $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N} p_j\}$ and to the value of upper bound $c_{\text{opt}}(LP_{\text{trans}})$. Consequently, this value is also equal to the optimal cycle time value.
10. Notice that the Algorithms 2 and 3 for all instances first solve LP_{relax}, and then only one additional linear mixed-integer program ($MIP^2_{\text{cycl},2}, MIP^2_{\text{cycl},3^0}$ or LP_{trans}).

In the above described experiment, the parameters $p_j (1 \leq j \leq N)$ and d have real values. We have also carried out the above experiment with integral values for these parameters. However, we did not observe any significant differences between the results for integer values and the results for real values, apart from that the use of real values seems to lead to more experiments that end with "solution status 102" than the use of integral values. A reason why hardly differences between the results for real and integral values exist can be that the decision variables s_j^m and $t_j^m (1 \leq j \leq N; m = 1,2)$ have real values anyway.

Since there is no need to consider both integral and real values of $p_j (1 \leq j \leq N)$ and d, from now on we assume that these parameters have real values.

In Experiment 1 we find for all problem instances an optimal solution that is equal to the lower bound $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N} p_j\}$. Due to the way in which we devised the experiment, this lower bound is always $2Nd$ or $\sum_{j=1}^{N} p_j$. Depending on which of these two bounds is larger, the optimal structure is structure 2 or 3^0. In the next experiment we examine what happens if $2Nd \approx \sum_{j=1}^{N} p_j$.

7.3.2 Experiment 2

Description

We take $\mathcal{N} = \{10, 25, 50\}, UB_p = \{5, 10, 25\}$ and $n_{\text{inst}} = 10$. Let d^* be the value of d for which $2Nd = \sum_{j=1}^{N} p_j$, that is, $d^* = \frac{1}{2N} \sum_{j=1}^{N} p_j$. We take d equal to $d^* + \varepsilon_d$ where, $\varepsilon_d \in \{-0.10, -0.05, 0.00, 0.05, 0.10\}$.

Results

The results of Experiment 2 are shown in the Table B.2 of Appendix B. We do not observe any relevant differences with the results of Experiment 1. We still find that the optimal schedule is of length $\max\{2Nd, \sum_{j=1}^{N} p_j\}$.

Thus, it seems to be hard to generate problem instances for which the optimal cycle time is larger than the lower bounds without the introduction of some "large" tasks, i.e., increasing the value of the third lower bound $2p_{\max} + 2d$. Next, we examine what happens if we choose all three lower bounds $2Nd, 2p_{\max} + 2d$ and $\sum_{j=1}^{N} p_j$ almost equal to each other.

7.3.3 Experiment 3

Description

We take $\mathcal{N} = \{10, 24, 25, 49, 50\}, UB_p = \{5, 10, 25\}$ and $n_{\text{inst}} = 10$, and we generate $p_1, p_2, \ldots, p_{N-1}$ at random in the same way as in the first experiments. Let $p_{\max}^{**}$ and d^{**}, respectively, be the values of $p_{\max}$ and d for which $2Nd = \sum_{j=1}^{N} p_j = 2p_{\max} + 2d$, i.e., $d^{**} = \frac{1}{N+1} \sum_{j=1}^{N-1} p_j$ and $p_{\max}^{**} = \frac{N-1}{N+1} \sum_{j=1}^{N-1} p_j$. We take d equal to

d^{**} and p_N equal to $p^{**}_{\max} + \varepsilon_p$ where, $\varepsilon_p \in \{-10.0, -5.0, 0.0, 5.0, 10.0\}$.

Results

The results of Experiment 3 are shown in the Table B.3 of Appendix B. If in the table a number is given between angle brackets $(<>)$, then this number indicates the number of executions in which an optimal schedule cannot be calculated in two hours.

We see that now the results become more interesting.

1. Optimal schedules of the structures 1,3^0,3^1 and 4 are found.
2. For N odd, the optimal schedules are of structure 3^0 if $\varepsilon_p < 0$, of the structures 1,2 and 3^0 if $\varepsilon_p = 0$ and of structure 1 if $\varepsilon_p > 0$.
3. For N is even the optimal schedules are of structure 3^0 if $\varepsilon_p \ll 0$, of structure 3^1 or 4 if $\varepsilon_p \approx 0$ and of structure 1 if $\varepsilon_p \gg 0$.
4. Sometimes the optimal cycle time of LP_{relax} is strictly larger than $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N}\}$. This only occurs for N even.
5. For N odd, the optimal cycle time of LP_{relax} is always equal to the optimal cycle time of LP_{trans}, for N even this is not the case.
6. If there exists an optimal schedule of structure 1, then this is found faster by the Algorithms 2 and 3 than by Algorithm 1. Algorithm 2 is faster than Algorithm 3 for small N and slower for large N.
7. In case an optimal schedule of structure 3^0 is found, then this is found faster by Algorithm 2 than by Algorithm 3.
8. If the optimal schedule is of structure 3^1 or 4 then, in general, it is only found by Algorithm 1. The Algorithms 2 and 3 are only able to find these schedules in the available time of two hours if N is very small.
9. Algorithm 1 finishes very often with "solution status 102".

In the above experiment first we have chosen $p_{\max}$ and d such that the three lower bounds all became equal to each other, after which we varied the value of $p_{\max}$. Obviously, we can also decide to vary the value of d. This is what we do in the next experiment.

7.3.4 Experiment 4

Description

We take $\mathcal{N} = \{10, 24, 25, 49, 50\}, UB_p = \{5, 10, 25\}$ and $n_{\text{inst}} = 10$, and we generate $p_1, p_2, \ldots, p_{N-1}$ in the same way as in the previous experiment. Let $p^{**}_{\max}$ and d^{**} again be the values of $p_{\max}$ and d for which $2Nd = \sum_{j=1}^{N} p_j = 2p_{\max} + 2d$. We take d equal to $d^{**} + \varepsilon_d$ and p_N equal to $p^{**}_{\max}$ where, $\varepsilon_d \in \{-0.50, -0.25, 0.00, 0.25, 0.50\}$.

Results

The results of Experiment 4 are shown in the Table B.4 of Appendix B. Similar results as for Experiment 3 are found. We observe the following.

1. Now optimal schedules of the structures 2,3^0, 3^1 and 4 are found.
2. For N is odd, the optimal schedules are of structure 2 if $\varepsilon_d < 0$, of the structures 1,2 and 3^0 if $\varepsilon_d = 0$ and of structure 3^0 if $\varepsilon_d > 0$.
3. For N is even, the optimal schedules are of structure 2 if $\varepsilon_d \ll 0$, of the structures 3^1 and 4 if $\varepsilon_d \approx 0$ and of structure 3^0 if $\varepsilon_d \gg 0$.
4. For N odd, the optimal cycle time of LP_{relax} is equal to $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N}\}$. For N even, the optimal cycle time of LP_{relax} is sometimes larger than $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N}\}$.
5. For N even, the optimal cycle time of LP_{relax} is not always equal to the optimal cycle time of LP_{trans}.
6. An optimal schedule of structure 2 is, in general, found much faster by Algorithm 2 than by Algorithm 1 and much faster by Algorithm 3 than by Algorithm 2.
7. If the optimal schedule is of structure 3^0, then it is found faster by Algorithm 2 than by Algorithm 3.
8. If the optimal schedule is of structure 3^1 or 4, it is, in general, only found by Algorithm 1. The Algorithms 2 and 3 are only able to find these schedules in the available time of two hours if N is very small.
9. Algorithm 1 finishes very often with "solution status 102".

We conclude that although the Algorithms 2 and 3 give nice results in case an optimal schedule of structure $1,2$ or 3^0 exists, they are not suitable for finding optimal schedules of the remaining structures. This is because when they find a parameter combination for which the corresponding critical cycle is of an interesting length, they always solve the corresponding linear mixed-integer program to see if this leads to a feasible solution. In general, many of such parameter combinations exist. Thus, if in the Steps 1-2 of the algorithm no feasible schedule is found with a cycle time equal to one of the lower bounds, then the Algorithms 2 and 3 start to consider many parameter combinations and therefore to solve a large number of linear mixed-integer programs.

7.4 Another algorithm

The Algorithms 2 and 3 contain a lot of the information that is given in the Chapters 3-6, but they still do not contain the information given in the Sections 6.2.3, 6.3.3, 6.4.3, 6.5.3 and 6.6.3. In these sections, we respectively derived necessary

conditions for obtaining an optimal schedule of structure $1,2,3,4$ and 5. We create another algorithm, Algorithm 4, that does make use of this information (see Figure 7.4). It works in a similar way as Algorithm 3, but before it starts to solve a linear mixed-integer program $MIP_{\text{cycl,i}}$ it first checks whether the corresponding necessary conditions hold $(1 \leq i \leq 5)$. If the conditions do not hold, then the corresponding linear mixed-integer program does not have to be solved any more.

Algorithm 4
For any $i = 1,2,\ldots,5$, replace in Algorithm 3: "Solve $MIP^2_{\text{cycl,i}}$" by:

{ Check necessary conditions of Section 6.(i+1).3
  IF conditions hold THEN
    Solve $MIP^2_{\text{cycl,i}}$
}.

(Remark: the above explains the basic idea. In the algorithm a few more small changes have to be made.)

Figure 7.4. Description of Algorithm 4.

Notice that in the cases where the lower bound LP_{relax} is equal to the upper bound LP_{trans} the Algorithms 3 and 4 both do exactly the same. Hence, Algorithm 4 would give the same results in the Experiments 1 and 2 as Algorithm 3. Also in the Experiments 3 and 4 both algorithms would obtain the same results, in case N is odd.

7.5 The performance of Algorithm 4

We repeat the Experiments 3 and 4 for Algorithm 4. The results are shown in the Tables B.5 and B.6. We only show the cases in which Algorithm 4 does not give the same results as Algorithm 3.

We observe the following:

1. Algorithm 4 can also find optimal schedules of the structures 3^1 and 4 within the available time of two hours.
2. For small instances, Algorithm 4 is slower than Algorithm 1. However, for larger N, Algorithm 4 is faster then Algorithm 1, in general much faster.
3. Executions of Algorithm 4 do not end with "solution status 102" and do not give memory problems.

Although it is clear that Algorithm 4 is the best of the four algorithms, optimal schedules of structure 3^0 are found faster by Algorithm 2 than by Algorithm 4. However, notice that Algorithm 4 can easily be adapted such that it finds optimal

schedules of structure 3^0 as fast as Algorithm 2 by replacing the first part of Step 1 of Algorithm 4 by:

Step 1 *Initialization*

LowerBound= $\max\{2Nd, 2p_{\max}+2d, \sum_{j=1}^{N} p_j, c_{\text{opt}}(LP_{\text{relax}})\}$.

IF LowerBound $= 2Nd$ THEN

UpperBound= $2(Nd+\sum_{j=1}^{N} p_j)$,

ELSE

UpperBound=min$\{2(Nd+\sum_{j=1}^{N} p_j), c_{\text{opt}}(LP_{\text{trans}})\}$.

7.6 Large problem instances

From the results with Experiment 1, we conclude that Algorithm 1 can be used for problem instances with up to more or less 75 tasks. For N=75, the algorithm finishes very often with "solution status 102" and sometimes there is even not enough memory to carry out the experiment. In order to see if Algorithm 4 can handle larger problem instances than Algorithm 1, we carry out Experiment 1 for $\mathcal{N} = \{100,150,200,250\}, UB_p = \{10\}, d = 2.75$ and Experiment 3/4 for $\mathcal{N} = \{74,75,99,100,149,150,199\}, UB_p = \{10\}, \varepsilon_d = \varepsilon_p = 0.00$. The results are shown in the Tables B.7 and B.8. We observe the following:

1. If the processing times are all chosen randomly in $[1, ub_p]$, then Algorithm 4 can solve problem instances with 200 tasks without any problems. If $N = 250$ then not enough memory is available.
2. If $N-1$ of the processing times are chosen randomly and p_N and d are chosen such that $2Nd = 2p_N + 2d = \sum_{j=1}^{N} p_j$ then instances up to 199 tasks can still be solved without memory problems and without messages "solution status 102". However, there is a large difference in execution time between odd and even values of N. If $N = 149$ then we find an average running time of 21 minutes, $N = 150$ gives an average running time of about 20 hours.

7.7 Some disproofs

The results presented in this chapter give the impression that for any problem instance there always exists an optimal schedule of the structures 1-4. However, this is not true. For problem instance $(\boldsymbol{p}, d) = ((3,1,3,1),1)$, no optimal schedule of the structures 1-4 exists. An optimal schedule for this instance is shown in Figure 7.5. It is of structure 5.

If we look at the results of the experiments it seems that for N odd there always exists an optimal schedule of length $\max\{2Nd, 2p_{\max}+2d, \sum_{j=1}^{N} p_j\}$. However, this is not true. For problem instance $(\boldsymbol{p}, d) = ((11.98,1,1,0.98,4),2)$, an optimal

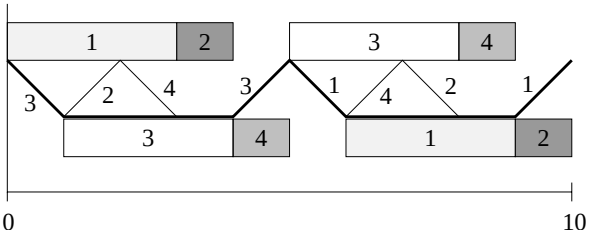


Figure 7.5. An optimal schedule of structure 5 (for problem instance $(\boldsymbol{p},d) = (3,1,3,1),1)$.)

schedule is given in Figure 7.6. It is of structure 3^2. Its cycle time is strictly larger than $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N} p_j\}$.

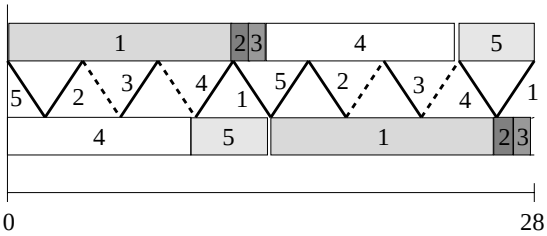


Figure 7.6. For problem instance $(\boldsymbol{p},d) = ((11.98,1,1,0.98,4),2)$ the optimal cycle time is strictly larger than $\max\{2Nd, 2p_{\max} + 2d, \sum_{j=1}^{N} p_j\}$.

7.8 Concluding remarks

The numerical results presented in this chapter show that we have developed an algorithm for tackling P^2_{cycl} that is much more accurate and, in general, much faster than solving the linear mixed-integer program MIP^2_{cycl}.

The results also demonstrate that it is difficult to generate difficult problem instances, randomly generated instances are easy to solve.

Furthermore, solving LP_{trans} turns out to be an excellent heuristic for solving P^2_{cycl}. For 2243 of the 2250 generated problem instances this resulted in an optimal solution. For the 7 other problem instances the optimal solution of LP_{trans} was within 0.5% of the optimum value for 4 of the instances, within 1% of the optimum for 6 of the instances and within 1.5% of the optimum for all of the instances.

In the cases in which solving LP_{trans} did not result in an optimal solution, optimal schedules of the structures 1,2 and 3^1 are found.

8

Conclusions and further research

In this final chapter we summarize the main results of this thesis and give suggestions for further research.

8.1 Conclusions

We introduced two versions of an m-machine scheduling problem with positive tool transportation times. First, we considered the linear version of the problem (P^m_{lin}), which then was used for introducing the cyclic version of the problem (P^m_{cycl}). The latter problem we modeled as a linear mixed-integer problem.

Even for only two machines, the complexity of the problem P^m_{cycl} turned out to be very difficult to determine. A thorough study of the cyclic two-machine problem provided us with a lot of knowledge, e.g., lower and upper bounds on the optimal cycle time length, symmetric schedules and interesting transport strategies. However, it did not bring the answer to the question whether the two-machine problem is solvable in polynomial time, or whether it is $\mathcal{NP}$-complete.

We showed that the relevant characteristics of optimal schedules for P^2_{cycl} can be represented by a cycle in a certain graph and that these cycles can be of different structures. Only five of these structures proved to be relevant, since for any problem instance there always exists an optimal schedule that corresponds to a cycle of one of these five structures. For some of the five structures an optimal schedule can

be found in polynomial time, for other structures, however, this is not clear. If for all five structures an optimal schedule can be determined in polynomial time, then problem P^2_{cycl} is also solvable in polynomial time.

Although the theoretical results presented in this thesis did not give us the complexity of the cyclic two-machine problem, they were used for developing an efficient algorithm, that is more accurate and, in general, much faster than solving the linear mixed-integer problem MIP^2_{cycl}. Furthermore, it can handle much larger problem instances.

In practice, most problem instances can be solved to optimality by using a simple heuristic. It is difficult to generate problem instances for which an optimal solution cannot easily be given.

8.2 Further research

Many interesting questions are still unanswered. In the first place, there is the question about the complexity of problem P^m_{cycl}. Even for two machines, we were not able to decide upon whether the problem is solvable in polynomial time, or whether it is NP-complete.

We have seen that if it can be proved that an optimal schedule of structure i can be found in polynomial time for $i = 2, 3^E (E > 0, E \text{ even}), 4$ and 5, then problem P^2_{cycl} can also be solved in polynomial time.

This thesis mainly concerned the two-machine problem P^2_{cycl}. Many different variations of this problem could be examined. Some directions for further research are suggested below.

1. We mainly looked at the two-machine problem. The problem becomes of more practical relevance if more than two machines are concerned.
2. We have only considered schedules in which in one cycle the job is executed exactly once on machine 1 and once on machine 2. Also other types of schedules could be examined.
3. In problem P^m_{cycl} for any task there is just one tool available. It would be interesting to allow for every task j a certain number n_j of tools. Furthermore, it would be useful to develop an approach for examining if the cost of buying an extra tool are compensated by the resulting increase in productivity of the system.
4. In this thesis we neglect the times that the robot needs for loading or unloading the tools. If the robot would only carry out nonempty transports, then these (un)loading times could be seen as part of the transport time. However, we also allow the robot to move without transporting a tool. In our

model such a transport does not take less time than a transport in which a tool is involved.

5. In our model the transport time of the robot linearly depends on the distance to be covered. However, in practice the robot probably carries out the following motion: it starts at velocity zero, then it respectively accelerates, maybe it moves some time at a constant velocity, it decelerates and it finishes at velocity zero. For the two-machine problem our assumption does not have consequences, since the robot always moves over the same distance. This implies that for the two-machine problem it is sufficient to know the time that the robot needs to cover the distance between the machines. However, for more machines it is likely that the time that the robot needs to cover the distance from a machine m_1 to a machine m_2 is strictly less than the time necessary to move from machine m_1 to a machine m_3 plus the time needed to move from machine m_3 to machine m_2.
6. It is interesting to see what happens if more robots are available for transporting the tools.

Since the ”basic” problem P^2_{cycl} already turned out to be so difficult to tackle, it is to be expected that for the above suggested problems heuristics have to be used.

A

Appendix to Theorem 5.1 (Case 2)

A.1 Introduction

In this appendix we consider the schedule $\bar{S}$ from the proof of Theorem 5.1 (Case 2) and prove that in this schedule:

$$T_j^2(4) < S_j^2(4) \text{ for any task } j.$$

We start by introducing some useful variables. For any task j, let

$$I(j) := S_j^1(4) + c - S_j^2(4) = S_j^1 + c - S_j^2(4) \tag{A.1}$$

and

$$II(j) := T_j^2(4) - S_j^1(4) = T_j^2(2) - S_j^1, \tag{A.2}$$

(see Figure A.1). Using these variables, $T_j^2(4) < S_j^2(4)$ is equivalent with $I(j) + II(j) < c$.

Notice that in $\bar{S}$, because of Step 4 of the algorithm, there exists at least one task j for which $S_j^2(4) + p_j + d = T_j^1(3) + c$. Let task j^* be the task of the job with the highest index for which this is true. Hence,

$$S_j^2(4) + p_j + d \begin{cases} \leq T_j^1(3) + c & \text{for } j = 1, 2, \ldots, j^* - 1, \\ = T_j^1(3) + c & \text{for } j = j^*, \\ < T_j^1(3) + c & \text{for } j = j^* + 1, \ldots, N. \end{cases} \tag{A.3}$$

Since in $\bar{S}$ both machines carry out the same schedule, apart from a shift, we have

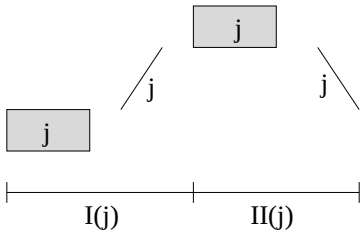


Figure A.1. $I(j)$ and $II(j)$.

for any task j that

$$I(j) \overset{(A.1)}{=} S_j^1 + c - S_j^2(4) \overset{(5.44)}{=} S_{j^*}^1 + c - S_{j^*}^2(4) \overset{(A.1)}{=} I(j^*).$$

Hence, for proving that $T_j^2(4) < S_j^2(4)$ for task j, it is sufficient to show that $I(j^*) + II(j) < c$ for task j. We do this by distinguishing the following cases:

$$\textbf{Case 2(a)} \quad I(j^*) \leq p_{j^*} + 3d \text{ and } II(j) \leq p_j + 3d + \delta(j), \tag{A.4}$$

$$\textbf{Case 2(b)} \quad I(j^*) > p_{j^*} + 3d \text{ and } II(j) \leq p_j + 3d + \delta(j), \tag{A.5}$$

$$\textbf{Case 2(c)} \quad I(j^*) \leq p_{j^*} + 3d \text{ and } II(j) > p_j + 3d + \delta(j), \tag{A.6}$$

$$\textbf{Case 2(d)} \quad I(j^*) > p_{j^*} + 3d \text{ and } II(j) > p_j + 3d + \delta(j), \tag{A.7}$$

where $\delta(j)$ is the idle time of the robot in the interval $[\overline{S}_j^1(4) + p_j + d, \overline{T}_j^2(4)]$.

In the rest of this appendix, we first introduce some new variables and notation and we derive some useful relations. After this, we give the proof for Case 2(a). Next, we give some sufficient conditions that imply that $I(j^*) \leq p_{j^*} + 3d$ or that $II(j) \leq p_j + 3d + \delta(j)$, and we derive some relations that hold for specific cases. Finally, the obtained relations are used for proving that $I(j^*) + I(j) < c$ for the Cases 2(b), 2(c) and 2(d).

A.2 Useful definitions, notation and relations

As announced, we start by introducing some variables and notation that will be useful in the rest of this appendix. We define for any task $j \in P_a (a = 1,2,3)$ (see Figure A.2):

$$\begin{array}{lcll} l(P_a| < j) & := & S_j^1 - b(P_a)^1 & = \sum_{i \in P_a, i < j} p_i, \\ l(P_a| > j) & := & e(P_a)^1 - (S_j^1 + p_j) & = \sum_{i \in P_a, i > j} p_i. \end{array}$$

and for any task $j \in Q_a (a = 1,2,3)$ (see Figure A.2):

$$\begin{array}{lcl} l(Q_a| < j) & := & S_j^1 - e(P_a)^1, \\ l(Q_a| > j) & := & b(P_{a \oplus_3 1})^1 + \delta_{a3} c - (S_j^1 + p_j). \end{array}$$

Consider schedule $\overline{S}$. Let $b(P_a)^2$ and $e(P_a)^2$, respectively, denote the times at

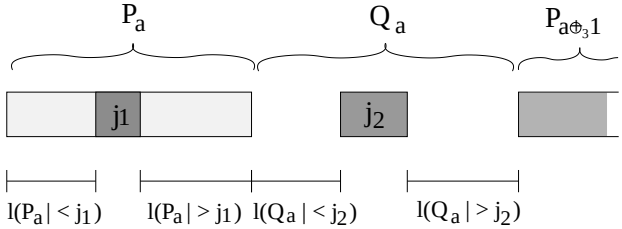


Figure A.2. $l(P_a) = l(P_a| < j_1) + p_{j_1} + l(P_a| > j_1)$ and $l(Q_a) = l(Q_a| < j_2) + p_{j_2} + l(Q_a| > j_2)$.

which in this schedule on machine 2 the processing of the first task of P_a starts and the time at which the processing of the last task of P_a ends $(a = 1,2,3)$.

Moreover, let δ_a be equal to the idle time of the robot in the interval $[b(P_a)^1 - d, e(P_a)^1]$ in $\bar{S}$. Notice that, since we have assumed that the robot carries out all the movements of the added ccwZZ's, $\delta_1 = 0, \delta_2 < 2d$ and $\delta_3 < 2d$ (see Figure A.3). Notice that now

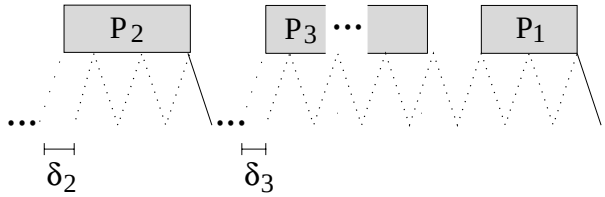


Figure A.3. $\delta_1 = 0, \delta_2 < 2d$ and $\delta_3 < 2d$.

$$\delta(j) = \begin{cases} \delta_2 & \text{if } j \in Q_1 \text{ and } T_j^2(4) \geq b(P_2)^1 + d, \\ \delta_3 & \text{if } j \in Q_2 \text{ and } T_j^2(4) \geq b(P_3)^1 + d, \\ 0 & \text{otherwise.} \end{cases} \tag{A.8}$$

Hence, for any task j we have

$$\delta(j) < 2d. \tag{A.9}$$

Let k be the number of transports to machine 1 that in $\bar{S}$ finish in the interval $[T_{j^*}^1(4) + 2d, S_{j^*}^1(4)]$. It is not very difficult to see that if $k > 0$, then the robot is not idle in the interval $[T_{j^*}^1(4), S_{j^*}^1(4)]$ and therefore (see Figure A.4):

$$I(j^*) \in [p_{j^*} + 2kd + d, p_{j^*} + 2(k+1)d + d) \tag{A.10}$$

and if $k = 0$, then (see Figure A.5 for an example):

$$I(j^*) \leq p_{j^*} + 3d. \tag{A.11}$$

If $k > 0$ and j^* is the first task of $Q_a (a = 1,2)$, we have that $S_{j^*}^1 = e(P_a)^1$, because in the original schedule S Assumption 2.3 holds. This implies that $S_{j^*}^1(4) = e(P_a)^1$

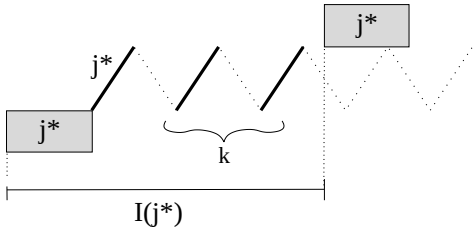


Figure A.4. There are k transports to machine 1 that finish in $[T^1_{j^*}(4)+2d, S^1_{j^*}(4)]$.

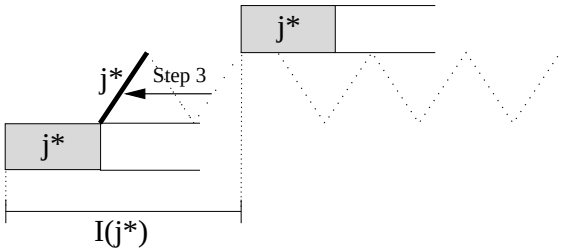


Figure A.5. If j^* is the first block of P_2 or P_3, then $k=0$ and $I(j^*)=p_{j^*}+3d$.

and, therefore, if j^* is the first task of $Q_a(a=1,2)$ and $k>0$, we get (see Figure A.6):

$$I(j^*)=p_{j^*}+2kd+d. \tag{A.12}$$

For any j, let $l(j)$ be the number of transports to machine 2 that in $\bar{S}$ finish in

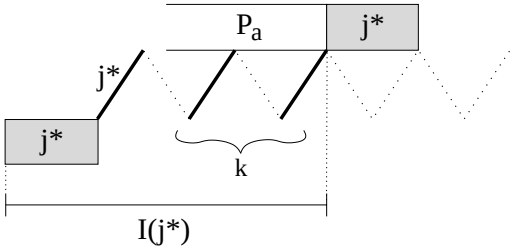


Figure A.6. Task j^* is the first block of $Q_a(a=1,2)$ and $k>0$.

$[S^1_j(4)+p_j+d, T^2_j(4)-2d]$ (see the Figures A.7 and A.8). Notice that

$$II(j)\in[p_j+2l_jd+d+\delta(j), p_j+2(l_j+1)d+d+\delta(j)). \tag{A.13}$$

Finally, notice that, obviously, for any task j we have

$$2d(\lfloor\frac{p_j}{2d}\rfloor+1)>p_j. \tag{A.14}$$

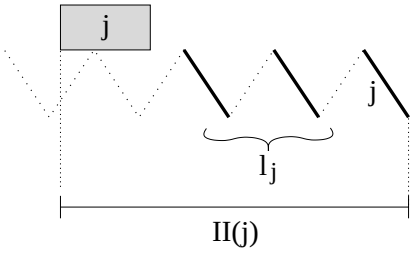


Figure A.7. The situation when $\delta(j) = 0$.

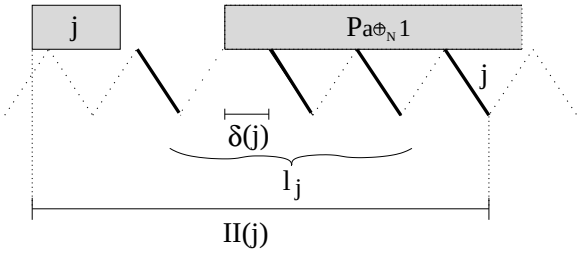


Figure A.8. The situation when $j \in Q_a (a = 1,2)$ and $\delta(j) > 0$.

A.3 The proof for Case 2(a)

Using the above notation and relations, we prove that $I(j^*) + I(j) < c$ for Case 2(a). Thus, assume that $I(j^*) \leq p_{j^*} + 3d$ and $II(j) \leq p_j + 3d + \delta(j)$. We distinguish two cases.

(i) If $\delta(j) = 0$ then

$$I(j^*) + II(j) \overset{(A.4)}{\leq} p_{j^*} + p_j + 6d + \delta(j) \leq 2p_{\max} + 6d \overset{(5.33)}{<} c.$$

(ii) If $\delta(j) > 0$ then relation (A.8) implies that $j \in Q_1 \cup Q_2$ and therefore,

$$\begin{aligned}
I(j^*) + II(j) &\overset{(A.4)}{\leq} && p_{j^*} + p_j + 6d + \delta(j) \\
&\overset{(A.9)}{<} && p_{j^*} + p_j + 8d \\
&\overset{j \in Q_1 \cup Q_2, (5.3)}{\leq} && p_{j^*} + l(Q_1) + l(Q_2) + 6d \\
&\leq && p_{\max} + l(Q_1) + l(Q_2) + 6d \\
&\overset{(5.1)(5.2)(5.3)}{\leq} && l(P_1) + \sum_{a=1}^{3} l(Q_a) + 4d \\
&\overset{(5.2)}{\leq} && \sum_{a=1}^{3} l(P_a) + \sum_{a=1}^{3} l(Q_a) \\
&\overset{(5.4)}{=} && c.
\end{aligned}$$

Thus, in both cases we have that $I(j^*)+II(j)<c$.

A.4 Sufficient conditions for $I(j^*)\leq p_{j^*}+3d$

In this section we give some conditions that imply that $I(j^*)\leq p_{j^*}+3d$. This information will be used later on in the proof.

(a) If $k=0$, then $I(j^*) \overset{(A.11)}{\leq} p_{j^*}+3d$.

(b) If j^* **is the first task of** $P_a(a=1,2,3)$, then $T^1_{j^*}=S^1_{j^*}$ and therefore,

$$\begin{aligned} I(j^*) &\overset{(A.1)}{=} S^1_{j^*}+c-S^2_{j^*}(4)\\ &\overset{(A.3)}{=} S^1_{j^*}-T^1_{j^*}(3)+p_{j^*}+d\\ &\overset{(5.40)}{=} S^1_{j^*}-T^1_{j^*}(2)+p_{j^*}+3d\\ &\overset{(5.34),(5.36)}{\leq} S^1_{j^*}-T^1_{j^*}+p_{j^*}+3d\\ &= p_{j^*}+3d. \end{aligned}$$

(c) Suppose that $p_{j^*}<2d$. Because of Assumption 2.3, we know that $S^1_{j^*}=S^1_{j^*\ominus_N 1}+p_{j^*\ominus_N 1}-\delta_{j^*1}c$ or $S^1_{j^*}=T^1_{j^*}$. If the latter is the case, then in exactly the same way as in (b) it can be seen that $I(j^*)\leq p_{j^*}+3d$.
Now suppose that $S^1_{j^*}=S^1_{j^*\ominus_N 1}+p_{j^*\ominus_N 1}-\delta_{j^*1}c$. Recall that after Step 3 of the algorithm no conflicts among the transports exist. Hence, $T^1_{j^*}(3) \overset{(2.16)}{\geq} T^1_{j^*\ominus_N 1}(3)+2d-\delta_{j^*1}c$ and therefore,

$$\begin{aligned} p_{j^*} &= p_{j^*}+S^1_{j^*}-(S^1_{j^*\ominus_N 1}+p_{j^*\ominus_N 1}-\delta_{j^*1}c)\\ &\overset{(5.44)}{=} p_{j^*}+S^2_{j^*}(4)-S^2_{j^*\ominus_N 1}(4)-p_{j^*\ominus_N 1}+\delta_{j^*1}c\\ &\overset{(A.3)}{=} T^1_{j^*}(3)+c-d-S^2_{j^*\ominus_N 1}(4)-p_{j^*\ominus_N 1}+\delta_{j^*1}c\\ &\geq T^1_{j^*\ominus_N 1}(3)+c+d-S^2_{j^*\ominus_N 1}(4)-p_{j^*\ominus_N 1}\\ &\overset{(5.43)}{\geq} 2d, \end{aligned}$$

which is a contradiction.

A.5 Sufficient conditions for $II(j)\leq p_j+3d+\delta(j)$

We give some conditions that imply that $II(j)\leq p_j+3d+\delta(j)$.

(a) If $l_j=0$ then $II(j) \overset{(A.13)}{<} p_j+3d+\delta(j)$.

(b) If $\boldsymbol{j}$ **is the last task of** $\boldsymbol{P_a(a=1,2,3)}$ then $T_j^2 = S_j^1 + p_j + d$ and therefore

$$\begin{aligned} II(j) &\overset{(A.2)}{=} T_j^2(2) - S_j^1 \\ &\overset{(5.35)(5.37)}{<} T_j^2 - S_j^1 + 2d \\ &= p_j + 3d. \end{aligned}$$

(c) Suppose that $\boldsymbol{j}$ **is the first task of** $\boldsymbol{Q_a(a=1,2)}$.
In the original schedule S, the tool of the last task of P_a arrives at machine 2 at time $e(P_a)^1 + d \leq S_j^1 + d$. Since $a \neq 3$, this is still the same in the schedule $\overline{S}$. Hence, in $\overline{S}$, the earliest transport to machine 2 that starts at or after $S_j^1 + p_j$ is not used for transporting the last tool of P_a. Notice that this transport cannot be an empty transport either because this would imply that in Step 2 of the algorithm still a transport could have been moved counterclockwise. Hence, the earliest transport to machine 2 that starts at or after $S_j^1 + p_j$ is used for transporting tool j and consequently $l_j = 0$ and $II(j) < p_j + 3d + \delta(j)$.

A.6 Implications from $I(j^*) > p_{j^*} + 3d$

Assume that $I(j^*) > p_{j^*} + 3d$. From Section A.4 we conclude that this implies

$$k > 0 \tag{A.15}$$

$$j^* \text{ is not the first task of } P_a (a = 1,2,3) \tag{A.16}$$

$$p_{j^*} \geq 2d. \tag{A.17}$$

We distinguish two cases:

(i) $\boldsymbol{j^* \in P_a(a=1,2,3)}$
If $j^* \in P_a$ then, in view of (A.16), j^* is not the first task of P_a. In the original schedule S, the tool of the first task of P_a arrives at machine 1 at time $b(P_a)^1$. Therefore, (5.34)(5.36) and (5.40) imply that in $\overline{S}$ this tool does not arrive at machine 1 before $b(P_a)^1 - 2d$. (If $a \in \{2,3\}$, then the tool arrives at machine 1 at $b(P_a)^1 - 2d$.) Moreover, the robot is idle from time $b(P_a)^1 - d$ until $b(P_a)^1 - d + \delta_a$ (see Figure A.9). Hence, $T_{j^*}^1(4) \geq b(P_a)^1 + \delta_a$.

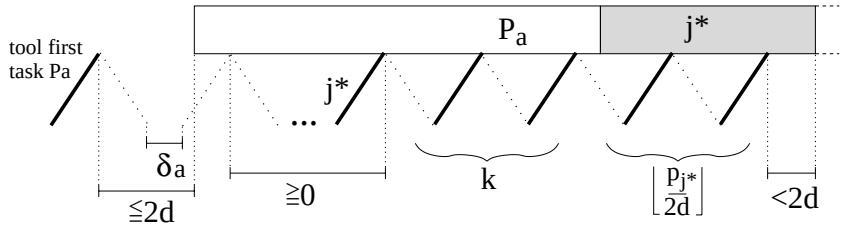


Figure A.9. Task $j^* \in P_a$.

In $\bar{S}$ there are k transports to machine 1 that finish in $[T^1_{j^*}(4)+2d, S^1_{j^*}(4)] \subseteq [b(P_a)^1+2d+\delta_a, S^1_{j^*}]$. This is an interval of length $l(P_a| < j^*) - 2d - \delta_a$. Hence,

$$\boxed{l(P_a| < j^*) \geq 2kd + \delta_a.} \tag{A.18}$$

There are $k + \lfloor \frac{p_{j^*}}{2d} \rfloor$ transports to machine 1 that arrive in $[T^1_{j^*}(4)+2d, S^1_{j^*}(4)+p_{j^*}]$. Therefore, (5.40) implies that just after Step 2 of the algorithm there exist $k+\lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ transports that arrive at machine 1 in $[T^1_{j^*}(2)+2d, S^1_{j^*}(4)+p_{j^*}] = [T^1_{j^*}(2)+2d, S^1_{j^*}(2)+p_{j^*}]$. Suppose first that one of these $k+\lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ transports to machine 1 is an empty transport. However, this means that Step 2 of the algorithm has not been carried out

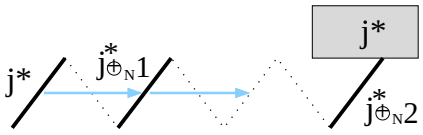


Figure A.10. Step 2 of the algorithm has not been carried out correctly.

correctly, since in this step still some transports to machine 1 could have been moved clockwise (see Figure A.10 for an example). Hence, all the $k+\lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ transports to machine 1 are nonempty transports. Since just after Step 2 of the algorithm all tools of $P_{a\oplus 3}1$ arrive at machine 1 after $e(P_a)^1$, this is only possible if there exist at least $k+\lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ tasks after j^* in $P_a \cup Q_a$.

Suppose first that there exist $x > 0$ tasks after j^* in P_a and suppose that $x \leq k+\lfloor \frac{p_{j^*}}{2d} \rfloor - 1$. Let j_1 be the last tool of P_a. Then, see Figure A.11,

$$T^1_{j_1}(3) = T^1_{j_1}(4) = T^1_{j^*}(4) + 2dx = T^1_{j^*}(3) + 2dx. \tag{A.19}$$

In the original schedule S the last x tasks of P_a arrive at machine 2 in $[S^1_{j^*} + p_{j^*} + 3d, e(P_a)^1 + d]$. This is an interval of length $l(P_a| > j^*) - 2d$. Hence, $l(P_a| > j^*) \geq 2dx$. (Remark: if the processings of the tasks of P_a form a single block, and not a sequence of blocks with some transports in between, then this follows directly from Lemma 4.2.) Therefore,

$$\begin{aligned} S^1_{j_1}(4) + p_{j_1} + d &= S^1_{j^*}(4) + p_{j^*} + l(P_a| > j^*) + d \\ &\geq S^1_{j^*}(4) + p_j + 2dx + d \\ &\overset{(A.3)}{=} T^1_{j^*}(3) + c + 2dx \\ &\overset{(A.19)}{=} T^1_{j_1}(3) + c. \end{aligned}$$

However, since task 1 is the first task of P_1, we know that $j_1 > j^*$. Hence, we get a contradiction with (A.3). As a consequence, for the number x of tasks

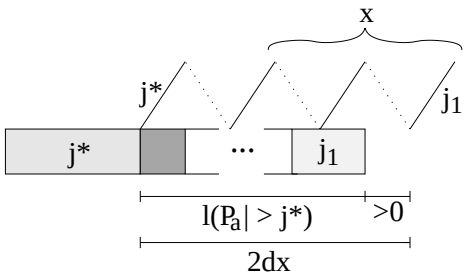


Figure A.11. The processing of task j_1 on machine 2 finishes strictly before $S^2_{j^*}(4)+p_{j^*}+2dx$.

after j^* in P_a we get $x \geq k+\lfloor \frac{p_{j^*}}{2d} \rfloor$.
In S all the tools of the last x tasks of P_a arrive at machine 2 in $[S^1_{j^*}+p_{j^*}+3d, e(P_a)^1+d]$. This is an interval of length $l(P_a| > j^*)-2d$. Hence, if j^* is not the last task of P_a we get

$$\boxed{l(P_a| > j^*) \geq 2d(k+\lfloor \tfrac{p_{j^*}}{2d} \rfloor).} \tag{A.20}$$

Now suppose that j^* is the last task of P_a. This implies that the $k+\lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ nonempty transports all are transports of tools of Q_a. Therefore, Q_a contains at least $k+\lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ tasks. In S the tools of Q_a arrive at machine 2 in the interval $[e(P_a)^1+3d, e(P_{a\oplus_3 1})^1+\delta_{a3}c-d]$. This is an interval of length $l(Q_a)+l(P_{a\oplus_3 1})-4d$. Hence, if j^* is the last task of P_a, this implies

$$l(Q_a)+l(P_{a\oplus_3 1}) \geq 2d(k+\lfloor \frac{p_{j^*}}{2d} \rfloor). \tag{A.21}$$

From (A.20), (A.21),(5.2) and (5.3) we now can conclude that for task $j^* \in P_a$ we have

$$\boxed{l(P_a| > j^*)+l(Q_a)+l(P_{a\oplus_3 1}) \geq 2d(k+\lfloor \tfrac{p_{j^*}}{2d} \rfloor).} \tag{A.22}$$

Hence, combining (A.18) and (A.22) gives:

$$\begin{aligned} l(P_a)+l(Q_a)+l(P_{a\oplus_3 1}) &= l(P_a|<j^*)+p_{j^*}+l(P_a|>j^*)+l(Q_a)+l(P_{a\oplus_3 1}) \\ &\geq 2kd+\delta_a+p_{j^*}+2d(k+\lfloor \frac{p_{j^*}}{2d} \rfloor) \\ &\overset{(A.14)}{>} 2p_{j^*}+4kd-2d+\delta_a. \end{aligned}$$

and, thus,

$$\boxed{p_{j^*} < \tfrac{1}{2}l(P_a)+\tfrac{1}{2}l(Q_a)+\tfrac{1}{2}l(P_{a\oplus_3 1})-2kd+d-\tfrac{1}{2}\delta_a.} \tag{A.23}$$

(ii) $\boldsymbol{j^* \in Q_a (a = 1,2,3)}$
If task $j^* \in Q_a$, then

$$S^1_{j^*} + p_{j^*} < b(P_{a\oplus_3 1})^1 + \delta_{a3} c. \tag{A.24}$$

This can be seen as follows. Because of Assumption 2.3, we have that $S^1_{j^*} + p_{j^*} = T^1_{j_1} + \sum_{i=j_1}^{j^*} p_i$ for a certain $j_1 \in \{\text{first task of } P_a, \ldots, j^*\}$. Now suppose that $S^1_{j^*} + p_{j^*} = b(P_{a\oplus_3 1})^1 + \delta_{a3} c$. This implies that $b(P_{a\oplus_3 1})^1 + \delta_{a3} c = T^1_{j_1} + \sum_{i=j_1}^{j^*} p_i$ (see Figure A.12 for an example). This, however, is in contradiction with Assumption 4.1. Thus, $S^1_{j^*} + p_{j^*}$ is strictly less than $b(P_{a\oplus_3 1})^1 + \delta_{a3} c$.

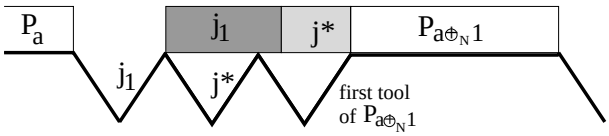


Figure A.12. Because of Assumption 4.1, $P_{a\oplus_3 1}$ is not a block.

In the same way as for $j^* \in P_a$, we can show that $T^1_{j^*}(4) \geq b(P_a)^1 + \delta_a$. In $\bar{S}$ there are k transports to machine 1 that finish in $[T^1_{j^*}(4) + 2d, S^1_{j^*}(4)] \subseteq [b(P_a)^1 + 2d + \delta_a, S^1_{j^*}]$ (see Figure A.13). This is an interval of length $l(P_a) + l(Q_a| < j^*) - 2d - \delta_a$. Hence,

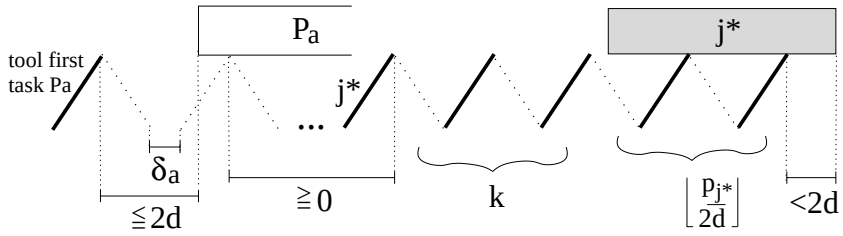


Figure A.13. Schedule $\bar{S}$: $j^* \in Q_a (a = 1,2,3)$.

$$\boxed{l(P_a) + l(Q_a| < j^*) \geq 2kd + \delta_a.} \tag{A.25}$$

It is not very difficult to see that if j^* is not the first task of Q_a then $T^1_{j^*}(4) \geq b(P_a)^1 + 2d + \delta_a$ and consequently,

$$\boxed{l(P_a) + l(Q_a| < j^*) \geq 2kd + 2d + \delta_a.} \tag{A.26}$$

In $\bar{S}$ there are $k + \lfloor \frac{p_{j^*}}{2d} \rfloor$ transports to machine 1 that finish in $[T^1_{j^*}(4) + 2d, S^1_{j^*}(4) + p_{j^*}]$. Hence, just after Step 2 of the algorithm there are $k + \lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ transports to machine 1 that finish in $[T^1_{j^*}(2) + 2d, S^1_{j^*}(2) + p_{j^*}]$. Following the same reasoning as for $j^* \in P_a$, these $k + \lfloor \frac{p_{j^*}}{2d} \rfloor - 1$ transports

to machine 1 cannot be empty transports. Notice that in $\bar{S}$ these $k+\lfloor\frac{p_{j^*}}{2d}\rfloor-1$ nonempty transports arrive at machine 1 before or at $S^1_{j^*}(4)+p_{j^*}-2d \overset{(A.24)}{<} b(P_{a\oplus_3 1})^1+\delta_{a3}c-2d$.

In the original schedule S the tool of the first task of $P_{a\oplus_3 1}$ arrives at machine 1 at time $b(P_{a\oplus_3 1})^1$. Hence, (5.34),(5.36) and (5.40) imply that in $\bar{S}$, the first tool of $P_{a\oplus_3 1}$ does not arrive at machine 1 before $b(P_{a\oplus_3 1})^1-2d$. Hence, all the $k+\lfloor\frac{p_{j^*}}{2d}\rfloor-1$ nonempty transports concern tools of Q_a. Consequently, there exist at least $k+\lfloor\frac{p_{j^*}}{2d}\rfloor-1$ tasks after task j^* in Q_a. In S the tools corresponding to these tasks arrive at machine 2 at a time in $[S^1_{j^*}+p_{j^*}+3d, e(P_{a\oplus_3 1})^1+\delta_{a3}c-d]$. This is an interval of length $l(Q_a|>j^*)+l(P_{a\oplus_3 1})-4d$. Hence, we get

$$\boxed{l(Q_a|>j^*)+l(P_{a\oplus_3 1})\geq 2d(k+\lfloor\tfrac{p_{j^*}}{2d}\rfloor).} \tag{A.27}$$

Combining (A.25) and (A.27) gives:

$$\begin{aligned} l(P_a)+l(Q_a)+l(P_{a\oplus_3 1}) &= l(P_a)+l(Q_a|<j^*)+p_{j^*}+l(Q_a|>j^*)+l(P_{a\oplus_3 1}) \\ &\geq 2kd+\delta_a+p_{j^*}+2d(k+\lfloor\frac{p_{j^*}}{2d}\rfloor) \\ &\overset{(A.14)}{>} 2p_{j^*}+4kd-2d+\delta_a. \end{aligned}$$

Therefore, we have

$$\boxed{p_{j^*}<\tfrac{1}{2}l(P_a)+\tfrac{1}{2}l(Q_a)+\tfrac{1}{2}l(P_{a\oplus_3 1})-2kd+d-\tfrac{1}{2}\delta_a.} \tag{A.28}$$

Combining (A.26) and (A.27) now implies that if j^* is not the first task of Q_a, then

$$\boxed{p_{j^*}<\tfrac{1}{2}l(P_a)+\tfrac{1}{2}l(Q_a)+\tfrac{1}{2}l(P_{a\oplus_3 1})-2kd-\tfrac{1}{2}\delta_a.} \tag{A.29}$$

A.7 Implications from $II(j)>p_j+3d+\delta(j)$

Assume that $II(j)>p_j+3d+\delta(j)$. From Section A.5 it follows that

$$l_j>0 \tag{A.30}$$

$$j \text{ is not the last task of } P_a (a=1,2,3) \tag{A.31}$$

$$j \text{ is not the first task of } Q_a (a=1,2) \tag{A.32}$$

We distinguish three cases.

(i) $j\in P_a (a=1,2,3)$

In the original schedule S the tool belonging to the last task of P_a arrives at machine 2 at time $e(P_a)^1+d$. Hence, (5.35) and (5.37) imply that in $\bar{S}$ this tool arrives at machine 2 strictly before $e(P_a)^1+3d$. (If $a\in\{1,2\}$, this tool arrives at machine 2 at $e(P_a)^1+d$.) Since due to (A.31) j is not the last task

of P_a and since in $\bar{S}$ there are no conflicts among the transports to machine 2, this implies that $T_j^2(4) < e(P_a)^1 + d$.

There are l_j robot transports that finish in $[S_j^1(4) + p_j + d, T_j^2(4) - 2d] \subseteq [S_j^1 + p_j + d, e(P_a)^1 - d)$ (see Figure A.14). This is an interval of length shorter than $l(P_a| > j) - 2d$. Hence,

$$\boxed{l(P_a| > j) > 2l_j d.} \tag{A.33}$$

There are $l_j + \lfloor \frac{p_j}{2d} \rfloor$ transports that arrive at machine 2 in $[S_j^1(4) + d, T_j^2(4) - 2d]$. Since in the Steps 3 and 4 the schedule on machine 1 and the transports to machine 2 are not changed, this implies that just after Step 2 there exist $l_j + \lfloor \frac{p_j}{2d} \rfloor$ transports to machine 2 that finish in $[S_j^1(2) + d, T_j^2(2) - 2d]$. Suppose that one of these $l_j + \lfloor \frac{p_j}{2d} \rfloor$ transports to machine 2 is empty. Again, this

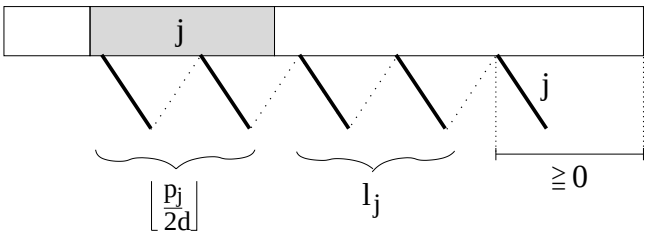


Figure A.14. Task $j \in P_a$.

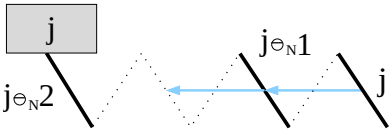


Figure A.15. Step 2 of the algorithm has not been carried out correctly.

means that Step 2 of the algorithm has not been carried out correctly, since in this step still some transports could have been moved counterclockwise (see Figure A.15 for an example). Hence, all the $l_j + \lfloor \frac{p_j}{2d} \rfloor$ transports to machine 2 are nonempty transports. Notice that, since after Step 2 the last tool of $P_{a \oplus_N 2}$ arrives at machine 2 strictly before

$$e(P_{a \oplus_N 2})^1 + 3d \overset{(5.3)}{\leq} b(P_a)^1 + d,$$

this is only possible if there exist at least $l_j + \lfloor \frac{p_j}{2d} \rfloor$ tasks before j in $Q_{a \oplus_3 2} \cup P_a$. In S the tools belonging to these tasks arrive at machine 1 in $[b(P_{a \oplus_3 2})^1 + 2d, S_j^1 - 2d]$. This is an interval of length $l(P_{a \oplus_3 2}) + l(Q_{a \oplus_3 2}) + l(P_a| < j) - 4d$. Therefore, we get

$$\boxed{l(P_{a \oplus_3 2}) + l(Q_{a \oplus_3 2}) + l(P_a| < j) \geq 2d(l_j + \lfloor \tfrac{p_j}{2d} \rfloor + 1).} \tag{A.34}$$

Combining this with (A.33) gives

$$\begin{aligned} l(P_{a\oplus_3 2})+l(Q_{a\oplus_3 2})+l(P_a) &= l(P_{a\oplus_3 2})+l(Q_{a\oplus_3 2})+l(P_a|<j)+p_j+l(P_a|>j) \\ &> 2d(l_j+\lfloor\frac{p_j}{2d}\rfloor+1)+p_j+2l_jd \\ &\overset{(A.14)}{>} 2p_j+4l_jd \end{aligned}$$

and, thus,

$$\boxed{p_j < \tfrac{1}{2}l(P_{a\oplus_3 2})+\tfrac{1}{2}l(Q_{a\oplus_3 2})+\tfrac{1}{2}l(P_a)-2l_jd.} \qquad \text{(A.35)}$$

(ii) $\boldsymbol{j \in Q_a(a=1,2)}$

In the same way as for $j \in P_a$, it can be seen that $T_j^2(4) < e(P_a)^1 + d$ and $T_j^2(4) < e(P_{a\oplus_3 1})^1 + d$. In $\bar{S}$ there are l_j transports that finish in $[S_j^1(4) + p_j + d, T_j^2(4) - 2d] \subseteq [S_j^1 + p_j + d, e(P_{a\oplus_3 1}^1) - d)$ (see the Figures A.16 and A.17). This is an interval of length shorter than $l(Q_a| > j) + l(P_{a\oplus_3 1}) - 2d$.

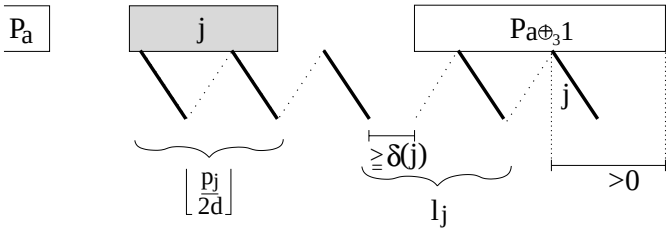


Figure A.16. $j \in Q_a(a = 1,2)$ when $\delta(j) > 0$.

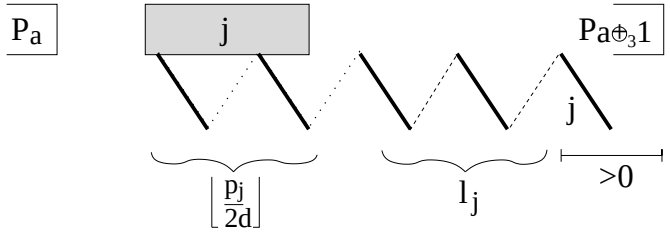


Figure A.17. $j \in Q_a(a = 1,2,3)$ when $\delta(j) = 0$.

Moreover, notice that in $\bar{S}$ the robot is also idle for $\delta(j)$ time units in the interval $[S_j^1(4) + p_j + d, e(P_{a\oplus_3 1}^1) - d)$. Hence,

$$\boxed{l(Q_a| > j) + l(P_{a\oplus_3 1}) > 2l_jd + \delta(j).} \qquad \text{(A.36)}$$

There are $l_j + \lfloor\frac{p_j}{2d}\rfloor$ transports that finish in $[S_j^1(4) + d, T_j^2(4) - 2d]$. Following the same reasoning as for $j \in P_a$, we find that none of these $l_j + \lfloor\frac{p_j}{2d}\rfloor$ transports can be an empty transport. Notice that, since $a \in \{1,2\}$, the last tool of P_a arrives at machine 2 at time $e(P_a)^1 + d < S_j^1(4) + d$ (use (A.32)

and $p_{j\ominus_N 1} > 0$). Therefore, all $l_j + \lfloor \frac{p_j}{2d} \rfloor$ nonempty transports that finish in $[S_j^1(4)+d, T_j^2(4)-2d]$ concern tools of Q_a. Hence, there exist at least $l_j + \lfloor \frac{p_j}{2d} \rfloor$ tasks before j in Q_a. In S the tools corresponding to these tasks arrive at machine 1 in $[b(P_a)^1 + 2d, S_j^1 - 2d]$. This is an interval of length $l(P_a) + l(Q_a| < j) - 4d$. Hence,

$$\boxed{l(P_a) + l(Q_a| < j) \geq 2d(l_j + \lfloor \tfrac{p_j}{2d} \rfloor + 1).} \tag{A.37}$$

Combining (A.36) with (A.37) gives

$$\begin{aligned} l(P_a)+l(Q_a)+l(P_{a\oplus_3 1}) &= l(P_a)+l(Q_a|<j)+p_j+l(Q_a|>j)+l(P_{a\oplus_3 1}) \\ &> 2d(l_j + \lfloor \frac{p_j}{2d} \rfloor + 1) + p_j + 2l_j d + \delta(j) \\ &\overset{(A.14)}{>} 2p_j + 4l_j d + \delta(j). \end{aligned}$$

Therefore, we have

$$\boxed{p_j < \tfrac{1}{2} l(P_a) + \tfrac{1}{2} l(Q_a) + \tfrac{1}{2} l(P_{a\oplus_3 1}) - 2l_j d - \tfrac{1}{2}\delta(j).} \tag{A.38}$$

(iii) $\boldsymbol{j \in Q_3}$

In $\bar{S}$ the tool belonging to the last task of P_1 arrives at machine 2 at $e(P_1)^1 + d$. Since in $\bar{S}$ there are no conflicts among the transports, this implies that $T_j^2(4) \leq e(P_1)^1 - d$. There are l_j transports that finish in $[S_j^1(4) + p_j + d, T_j^2(4) - 2d] \subseteq [S_j^1 + p_j + d, e(P_1)^1 - 3d]$. This is an interval of length $l(Q_3| > j) + l(P_1) - 4d$. Hence,

$$l(Q_3| > j) + l(P_1) \geq 2l_j d + 2d$$

and since for $j \in Q_3$ we have $\delta(j) = 0$ this implies that

$$\boxed{l(Q_3| > j) + l(P_1) \geq 2l_j d + 2d + \delta(j).} \tag{A.39}$$

There are $l_j + \lfloor \frac{p_j}{2d} \rfloor$ transports that finish in $[S_j^1(4)+d, T_j^2(4)-2d]$. In the same way as for the case $a = 1,2$ we can see that none of these $l_j + \lfloor \frac{p_j}{2d} \rfloor$ robot transports can be empty. Consider the transport of the last tool of P_3 to machine 2. In S this transport finishes at time $e(P_3)^1 + d$. During the execution of the algorithm the transport is only moved in Step 1. There, it is moved clockwise by less than $2d$. Hence in $\bar{S}$ the transport of the last tool of P_3 to machine 2 finishes strictly before $e(P_3)^1 + 3d < S_j^1 + 3d = S_j^1(4) + 3d$ (use (A.32) and $p_{j\ominus_N 1} > 0$). Therefore, the last $l_j + \lfloor \frac{p_j}{2d} \rfloor - 1$ nonempty transports that finish in $[S_j^1(4)+d, T_j^2(4)-2d]$ concern tools of Q_3. Hence, there exist at least $l_j + \lfloor \frac{p_j}{2d} \rfloor - 1$ tasks before j in Q_3. In S the tools corresponding to these tasks arrive at machine 1 in $[b(P_3)^1 + 2d, S_j^1 - 2d]$. This is an interval

of length $l(P_3)+l(Q_3|<j)-4d$. Hence, we get

$$\boxed{l(P_3)+l(Q_3|<j) \geq 2d(l_j+\lfloor\tfrac{p_j}{2d}\rfloor).} \tag{A.40}$$

Combining (A.39) with (A.40) results in

$$\begin{array}{lcl} l(P_3)+l(Q_3)+l(P_1) & = & l(P_3)+l(Q_3|<j)+p_j+l(Q_3|>j)+l(P_1) \\ & \geq & 2d(l_j+\lfloor\frac{p_j}{2d}\rfloor)+p_j+2l_jd+2d+\delta(j) \\ & \overset{(A.14)}{>} & 2p_j+4l_jd+\delta(j) \end{array}$$

and, therefore,

$$\boxed{p_j < \tfrac{1}{2}l(P_3)+\tfrac{1}{2}l(Q_3)+\tfrac{1}{2}l(P_1)-2l_jd-\tfrac{1}{2}\delta(j).} \tag{A.41}$$

A.8 The proof for the Cases 2(b), 2(c) and 2(d)

We still have to prove that if $I(j^*)>p_{j^*}+3d$ or $II(j)>p_j+3d+\delta(j)$ then $I(j^*)+II(j)<c$. This can be done with the help of the relations derived in the previous sections of this appendix.

We distinguish many different cases. For all these cases, it can be proved that $I(j^*)+II(j)<c$ by combining an adequate subset of the given relations in an appropriate way.

Suppose for example that $j^*\in P_a$ and $j\in Q_b(a\in\{1,2,3\},b\in\{a,a\oplus_3 1\})$ and that we have Case 2(c). In this situation we have that:

$$\begin{array}{lcl} I(j^*)+II(j) & \overset{(A.6)(A.13)}{<} & p_{j^*}+p_j+2l_jd+6d+\delta(j) \\ & \overset{(A.36)/(A.39)}{<} & p_{j^*}+p_j+l(Q_b|>j)+l(P_{b\oplus_3 1})+6d \\ & \overset{j^*\in P_a, j\in Q_b}{\leq} & l(P_a)+l(Q_b)+l(P_{b\oplus_3 1})+6d \\ & \overset{(5.4),b\in\{a,a\oplus_3 1\}}{=} & c-\sum_{i\neq a,b\oplus_3 1} l(P_i)-l(Q_{b\oplus_3 1})-l(Q_{b\oplus_3 2})+6d \\ & \overset{(5.2)(5.3)}{\leq} & c. \end{array}$$

For all the different cases, it can be proved in a similar way that $I(j^*)+II(j)<c$, by using the formulas given on the next page. (A complete elaboration of the required calculations can be found in Kuijpers [2001].)

A.9 Conclusion

In Section A.3 we have seen that in Case 2(a) we have that $I(j^*)+II(j)<c$ for any task j. Moreover, in the previous section we have explained how it can be proved that also in the Cases 2(b), 2(c) and 2(d) we have that $I(j^*)+II(j)<c$ for any task j. From this we conclude that $I(j^*)+II(j)<c$ for any task j, which is equivalent with $T_j^2(4)<S_j^2(4)$ for any task j.

$j^* \in P_a, j \in P_a$ ($a = 1,2,3$)

2(b)	$j^*=j$	(A.5),(A.10),(A.8),$j^*=j$,(A.23),$\delta_a \geq 0$,(A.15),(5.4),(5.2),(5.3)
	$j^*<j$	(A.5),(A.10),(A.8),(A.18),$\delta_a \geq 0$; $j^*, j \in P_a$; $j^*<j$,(5.4),(5.2),(5.3)
	$j^*>j$	(A.5),(A.10),(A.8),(A.22),$j^*, j \in P_a$; $j^*>j$,(A.17),(5.4),(5.2),(5.3)
2(c)	$j^*=j$	(A.6),(A.13),(A.8),$j^*=j$,(A.35),(A.30),(5.4),(5.2),(5.3)
	$j^*<j$	(A.6),(A.13),(A.8),(A.33);$j^*, j \in P_a$; $j^*<j$,(5.4),(5.2),(5.3)
	$j^*>j$	(A.6),(A.13),(A.8),(A.34);$j^*, j \in P_a$; $j^*>j, p_j>0$,(5.4),(5.2),(5.3)
2(d)	$j^*=j$	(A.10),(A.13),(A.20),(A.32);$j^*, j \in P_a$; $j^*=j$,(A.14),(5.4),(5.2),(5.3)
	$j^*<j$	(A.10),(A.13),(A.8),(A.18),$\delta_a \geq 0$,(A.33);$j^*, j \in P_a$; $j^*<j$,(5.4),(5.2),(5.3)
	$j^*>j$	(A.10),(A.13),(A.22),(A.34); $j^*, j \in P_a$; $j^*>j$,(A.17),$p_j>0$,(5.4),(5.3)

$j^* \in P_a, j \in P_b (a, b \in \{1,2,3\}, a \neq b)$

2(b)		(A.5),(A.10),(A.8),(A.18),$\delta_a \geq 0$, $j^* \in P_a, j \in P_b, a \neq b$,(5.4),(5.2),(5.3)
2(c)		(A.6),(A.13),(A.8),(A.33),$j^* \in P_a, j \in P_b, a \neq b$,(5.4),(5.2),(5.3)
2(d)		(A.10),(A.13),(A.8),(A.18),$\delta_a \geq 0$,(A.33),$j^* \in P_a, j \in P_b, a \neq b$,(5.4),(5.2),(5.3)

$j^* \in P_a, j \in Q_b$ ($a = 1,2,3; b \in \{a, a \oplus_3 1\}$)

2(b)		(A.5),(A.10),(A.9),(A.18),$\delta_a \geq 0$, $j^* \in P_a, j \in Q_b$,(5.4),$b \in \{a, a \oplus_3 1\}$,(5.2),(5.3)
2(c)		(A.6),(A.13),(A.36)/(A.39), $j^* \in P_a, j \in Q_b$,(5.4),$b \in \{a, a \oplus_3 1\}$,(5.2),(5.3)
2(d)		(A.10),(A.13),(A.18),$\delta_a \geq 0$,(A.36)/(A.39),$j^* \in P_a, j \in Q_b$,(5.4),$b \in \{a, a \oplus_3 1\}$,(5.2),(5.3)

$j^* \in P_a, j \in Q_{a \oplus_3 2}$ ($a = 1,2,3$)

2(b)		(A.5),(A.10),(A.9),(A.18),$\delta_a \geq 0$, $j^* \in P_a, j \in Q_{a \oplus_3 2}$,(5.4),(5.2),(5.3)
2(c)	$a=1$	(A.6),(A.13),(A.8),(A.40),$j^* \in P_1, j \in Q_3, p_j>0$,(5.4),(5.2),(5.3)
	$a \neq 1$	(A.6),(A.13),(A.9),(A.37),$j^* \in P_a, j \in Q_{a \oplus_3 2}, p_j>0$,(5.4),(5.2),(5.3)
2(d)	$a=1$	(A.10),(A.13),(A.8),(A.18),$\delta_a \geq 0$,(A.40), $j^* \in P_1, j \in Q_3, p_j>0$,(5.4),(5.2),(5.3)
	$a \neq 1$	(A.10),(A.13),(A.9),(A.18),$\delta_a \geq 0$,(A.37),$j^* \in P_a, j \in Q_{a \oplus_3 2}, p_j>0$,(5.4),(5.2),(5.3)

$j^* \in Q_a, j \in P_a$ ($a = 1,2,3$)

2(b)		(A.5),(A.10),(A.8),(A.27),$j^* \in Q_a, j \in P_a$,(A.17),(5.4),(5.2),(5.3)
2(c)		(A.6),(A.13),(A.8),(A.33),$j^* \in Q_a, j \in P_a$,(5.4),(5.2),(5.3)
2(d)		(A.10),(A.13),(A.8),(A.27),(A.33),$j^* \in Q_a, j \in P_a$,(A.17),(5.4),(5.2),(5.3)

$j^* \in Q_a, j \in P_{a \oplus_3 1}$ ($a = 1,2,3$)

2(b)		(A.5),(A.10),(A.8),(A.25),$\delta_a \geq 0$, $j^* \in Q_a, j \in P_{a \oplus_3 1}$,(5.4),(5.2),(5.3)
2(c)		(A.6),(A.13),(A.8),(A.33),$j^* \in Q_a, j \in P_{a \oplus_3 1}$,(5.4),(5.2),(5.3)
2(d)		(A.10),(A.13),(A.8),(A.25),$\delta_a \geq 0$,(A.33),$j^* \in Q_a, j \in P_{a \oplus_3 1}$,(5.4),(5.2),(5.3)

$j^* \in Q_a, j \in P_{a \oplus_3 2}$ ($a = 1,2,3$)

2(b)		(A.5),(A.10),(A.8),(A.25),$\delta_a \geq 0$, $j^* \in Q_a, j \in P_{a \oplus_3 2}$,(5.4),(5.2),(5.3)
2(c)		(A.6),(A.13),(A.8),(A.33),$j^* \in Q_a, j \in P_{a \oplus_3 2}$,(5.4),(5.2),(5.3)
2(d)		(A.10),(A.13),(A.8),(A.27),(A.33),$j^* \in Q_a, j \in P_{a \oplus_3 2}$,(A.17),(5.4),(5.2),(5.3)

$j^* \in Q_a, j \in Q_a$ ($a = 1,2,3$)

2(b)	$j^*=j$	$a=3$	(A.5),(A.10),(A.8),$j^*=j$,(A.28),$\delta_a \geq 0$,(A.15),(5.4),(5.2),(5.3)
		$a \neq 3$[1]	(A.5),(A.12),(A.9),$j^*=j$,(A.28),$\delta_a \geq 0$,(A.15),(5.4),(5.2),(5.3)
		$a \neq 3$[2]	(A.5),(A.10),(A.9),$j^*=j$,(A.29),$\delta_a \geq 0$,(A.15),(5.4),(5.2),(5.3)
	$j^*<j$		(A.5),(A.10),(A.9),(A.25),$\delta_a \geq 0$; $j^*, j \in Q_a$; $j^*<j$,(5.4),(5.2),(5.3)
	$j^*>j$		(A.5),(A.10),(A.9),(A.27),$j^*, j \in Q_a$; $j^*>j$,(A.17),(5.4),(5.2),(5.3)
2(c)	$j^*=j$		(A.6),(A.13),$j^*=j$,(A.38)/(A.41),(A.30),(5.4),(5.2),(5.3)
	$j^*<j$		(A.6),(A.13),(A.36)/(A.39),$j^*, j \in Q_a$; $j^*<j$,(5.4),(5.2),(5.3)
	$j^*>j$		(A.6),(A.13),(A.9),(A.37)/(A.40),$j^*, j \in Q_a$; $j^*>j, p_j>0$,(5.4),(5.2),(5.3)
2(d)	$j^*=j$	$a \neq 3$	(A.10),(A.13),(A.9),$j^*=j$,(A.27),(A.37); $j^*, j \in Q_a$;$j^*=j$,(A.14),(5.4),(5.2),(5.3)
		$a=3$	(A.10),(A.13),(A.8),$j^*=j$,(A.27),(A.40),$j^*, j \in Q_3$;$j^*=j$,(A.14),(A.32),(5.4),(5.2),(5.3)
	$j^*<j$		(A.10),(A.13),(A.25),$\delta_a \geq 0$,(A.36)/(A.39), $j^*, j \in Q_a$;$j^*<j$,(5.4),(5.2),(5.3)
	$j^*>j$		(A.10),(A.13),(A.9),(A.27),(A.37)/(A.40),$j^*, j \in Q_a$;$j^*>j$,(A.17),(5.4),(5.2),(5.3)

$j^* \in Q_a, j \in Q_{a \oplus_3 1}$ ($a = 1,2,3$)

2(b)		(A.5),(A.10),(A.9),(A.27),$j^* \in Q_a, j \in Q_{a \oplus_3 1}$,(A.17),(5.4),(5.2),(5.3)
2(c)		(A.6),(A.13),(A.36)/(A.39),$j^* \in Q_a, j \in Q_{a \oplus_3 1}$,(5.4),(5.2),(5.3)
2(d)		(A.10),(A.13),(A.27),(A.36)/(A.39),$j^* \in Q_a, j \in Q_{a \oplus_3 1}$,(A.17),(5.4),(5.2),(5.3)

$j^* \in Q_a, j \in Q_{a \oplus_3 2}$ ($a = 1,2,3$)

2(b)		(A.5),(A.10),(A.9),(A.27), $j^* \in Q_a, j \in Q_{a \oplus_3 2}$,(A.17),(5.4),(5.2),(5.3)
2(c)	$a=1$	(A.6),(A.13),(A.8),(A.40),$j^* \in Q_1, j \in Q_3, p_j>0$,(5.4),(5.2),(5.3)
	$a \neq 1$	(A.6),(A.13),(A.9),(A.37),$j^* \in Q_a, j \in Q_{a \oplus_3 2}, p_j>0$,(5.4),(5.2),(5.3)
2(d)	$a=1$	(A.10),(A.13),(A.8),(A.27),(A.40), $j^* \in Q_1, j \in Q_3$,(A.17),$p_j>0$,(5.4),(5.2),(5.3)
	$a \neq 1$	(A.10),(A.13),(A.9),(A.27),(A.37),$j^* \in Q_a, j \in Q_{a \oplus_3 2}$,(A.17),(5.4),(5.2),(5.3)

1. j^* is the first task of Q_a 2. j^* is not the first task Q_a

B

Numerical results

Table B.1. Results Experiment 1 (Part 1).

N	p_i $(1 \leq i \leq N)$	d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	c_{relax} = $\max_{lb}$ [23]	c_{relax} = c_{trans} [24]	c_{trans}[4] optimal	optimal structure
10	1-5	1.00	0.318	0.048 (1)[5]	0.152	0.034	0.090	0.000	5x	5x	5x	2 (5x)
		1.50	0.276	0.096	0.144	0.036	0.096	0.009	5x	5x	5x	2 (4x), 3^0(1x)
		2.00	0.274	0.074	0.078	0.004	0.088	0.008	5x	5x	5x	3^0(5x)
		3.00	0.170	0.010	0.074	0.005	0.078	0.008	5x	5x	5x	3^0(5x)
		4.00	0.174	0.005	0.070	0.000	0.080	0.007	5x	5x	5x	3^0(5x)
	1-10	1.00	0.370	0.084	0.182	0.054	0.106	0.015	5x	5x	5x	2 (5x)
		2.00	0.244	0.063	0.176	0.021	0.126	0.005	5x	5x	5x	2 (5x)
		2.75	0.296	0.097 (1)	0.228	0.091	0.132	0.004	5x	5x	5x	2 (4x), 3^0(1x)
		5.00	0.194	0.029	0.092	0.004	0.128	0.004	5x	5x	5x	3^0(5x)
		9.00	0.140	0.023	0.092	0.004	0.124	0.011	5x	5x	5x	3^0(5x)
	1-25	2.00	0.170	0.046	0.146	0.015	0.120	0.007	5x	5x	5x	2 (5x)
		5.00	0.320	0.050	0.172	0.028	0.134	0.005	5x	5x	5x	2 (5x)
		6.50	0.328	0.059	0.214	0.090	0.138	0.008	5x	5x	5x	2 (4x), 3^0(1x)
		12.00	0.136	0.039 (2)	0.094	0.005	0.128	0.011	5x	5x	5x	3^0(5x)
		20.00	0.126	0.009 (1)	0.094	0.005	0.136	0.005	5x	5x	5x	3^0(5x)
25	1-5	1.00	9.120	5.274	2.098	1.515	0.672	0.022	5x	5x	5x	2 (5x)
		1.50	14.544	8.142	1.972	1.552	0.766	0.034	5x	5x	5x	2 (3x), 3^0(2x)
		2.00	9.508	10.386	0.360	0.010	0.614	0.021	5x	5x	5x	3^0(5x)
		3.00	7.842	3.082	0.358	0.034	0.576	0.055	5x	5x	5x	3^0(5x)
		4.00	14.198	6.502	0.338	0.016	0.546	0.046	5x	5x	5x	3^0(5x)
	1-10	1.00	13.832	5.008 (2)	9.934	4.085	1.520	0.039	5x	5x	5x	2 (5x)
		2.00	12.328	7.272	5.236	1.234	1.350	0.199	5x	5x	5x	2 (5x)
		2.75	10.122	7.386	4.044	3.726	1.430	0.205	5x	5x	5x	2 (3x), 3^0(2x)
		5.00	4.884	2.506	0.640	0.171	1.084	0.228	5x	5x	5x	3^0(5x)
		9.00	2.154	0.762 (3)	0.500	0.048	0.918	0.098	5x	5x	5x	3^0(5x)
	1-25	2.00	6.524	0.559 (2)	3.984	1.116	1.356	0.119	5x	5x	5x	2 (5x)
		5.00	14.570	5.033	7.502	3.530	1.386	0.155	5x	5x	5x	2 (5x)
		6.50	15.680	10.886	4.278	3.409	1.496	0.144	5x	5x	5x	2 (3x), 3^0(2x)
		12.00	5.944	1.242	0.672	0.068	1.188	0.111	5x	5x	5x	3^0(5x)
		20.00	3.468	1.574 (5)	0.540	0.084	1.000	0.116	5x	5x	5x	3^0(5x)

1. calculated over 5 experiments, in seconds (CPU time).
2. $c_{relax} = c_{opt}(LP_{relax})$.
3. $\max_{lb} = \max\{2p_{max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{trans} = c_{opt}(LP_{trans})$.
5. number of experiments that finished with "solution status 102".

N	p_i $(1 \le i \le N)$	d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	c_{relax} = max_{lb} [3]	c_{relax} = c_{trans} [2]	c_{trans} optimal [4]	optimal structure
50	1-5	1.00	1575.642	1525.095	156.808	131.765	5.600	0.685	5x	5x	5x	2 (5x)
		1.50	1063.652	690.859	27.106	21.769	5.974	0.401	5x	5x	5x	2 (3x), 3^0(2x)
		2.00	867.252	975.991	2.540	0.112	4.596	0.180	5x	5x	5x	3^0(5x)
		3.00	161.900	172.573	2.298	0.129	4.162	0.227	5x	5x	5x	3^0(5x)
		4.00	154.512	159.145 (2)[5]	2.360	0.240	4.212	0.285	5x	5x	5x	3^0(5x)
	1-10	1.00	361.096	206.415 (3)	239.212	89.784	16.740	1.373	5x	5x	5x	2 (5x)
		2.00	398.284	212.045	196.520	74.623	19.014	0.944	5x	5x	5x	2 (5x)
		2.75	459.748	170.274	111.542	108.832	20.852	1.044	5x	5x	5x	2 (3x), 3^0(2x)
		5.00	341.772	403.406	12.060	1.405	18.838	2.419	5x	5x	5x	3^0(5x)
		9.00	200.016	24.320 (5)	4.604	0.429	8.342	0.406	5x	5x	5x	3^0(5x)
	1-25	2.00	137.684	91.943 (3)	98.254	28.158	17.902	0.798	5x	5x	5x	2 (5x)
		5.00	486.414	159.578 (2)	202.452	127.130	21.234	1.104	5x	5x	5x	2 (5x)
		6.50	185.626	115.020 (2)	131.396	126.331	21.382	0.702	5x	5x	5x	2 (3x), 3^0(2x)
		12.00	133.332	62.930 (2)	12.298	0.721	19.480	1.113	5x	5x	5x	3^0(5x)
		20.00	177.972	41.710 (2)	5.522	1.395 (1)	9.974	2.056	5x	5x	5x	3^0(5x)
75	1-5	1.00	1849.816	583.629 (5)	2539.420	1210.605 [1]	22.146	9.895	5x	5x	5x	2 (5x)
		1.50	2229.881	376.980	778.723	1369.790	21.522	1.146	5x	5x	5x	2 (3x), 3^0(2x)
		2.00	1497.953	729.436 (4)[1][6]	8.224	0.366	15.496	1.449	5x	5x	5x	3^0(5x)
		3.00	83.753	71.163 (5)	6.884	0.656	12.596	0.778	5x	5x	5x	3^0(5x)
		4.00	535.060	783.455 (5)	6.884	0.777	12.870	1.366	5x	5x	5x	3^0(5x)
	1-10	1.00	2474.895	1588.451 (3)	1955.372	501.348	75.044	5.021	5x	5x	5x	2 (5x)
		2.00	1633.778	938.355 (5)	1226.844	274.149	89.504	1.484	5x	5x	5x	2 (5x)
		2.75	2348.185	1239.497 (4)	466.286	406.719	91.992	4.017	5x	5x	5x	2 (3x), 3^0(2x)
		5.00	1129.585	902.313 (2)	55.100	13.790	80.982	18.008	5x	5x	5x	3^0(5x)
		9.00	1598.073	556.278 (2)	17.976	2.187	31.330	3.106	5x	5x	5x	3^0(5x)
	1-25	2.00	1989.786	692.582 (4)	1942.640	1282.960	90.358	10.423	5x	5x	5x	2 (5x)
		5.00	1621.143	1046.644 (3)	1738.414	963.983	90.298	4.254	5x	5x	5x	2 (5x)
		6.50	968.214	1063.514 (2)[2]	748.690	9.597	101.318	3.494	5x	5x	5x	2 (3x), 3^0(2x)
		12.00	1654.776	1340.105 (2)	62.096	7.863	91.676	9.597	5x	5x	5x	3^0(5x)
		20.00	1785.432	325.788 (5)	26.232	9.530	45.174	12.533	5x	5x	5x	3^0(5x)

Table B.1. Results Experiment 1 (Part 2).

1. calculated over 5 experiments, in seconds (CPU time).
2. $c_{relax} = c_{opt}(LP_{relax})$.
3. $max_{lb} = \max\{2p_{max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{trans} = c_{opt}(LP_{trans})$.
5. number of experiments that finished with "solution status 102".
6. number of experiments that stopped because of memory problems.

Table B.2. Results Experiment 2 (Part 1).

N	p_i $(1 \leq i \leq N)$	ε_d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	$c_{\text{relax}} = \max_{\text{lb}}$ [2] [3]	$c_{\text{relax}} = c_{\text{trans}}$ [2] [4]	c_{trans} [4] optimal	optimal structure
10	1-5	-0.10	0.319	0.034	0.196	0.035	0.099	0.012	10x	10x	10x	2 (10x)
		-0.05	0.283	0.075	0.173	0.013	0.098	0.006	10x	10x	10x	2 (10x)
		0.00	0.279	0.051	0.132	0.051	0.102	0.009	10x	10x	10x	3^0 (10x)
		0.05	0.272	0.069	0.084	0.005	0.094	0.005	10x	10x	10x	3^0 (10x)
		0.10	0.248	0.076	0.083	0.005	0.096	0.005	10x	10x	10x	3^0 (10x)
	1-10	-0.10	0.330	0.057	0.224	0.066	0.133	0.007	10x	10x	10x	2 (10x)
		-0.05	0.269	0.080	0.187	0.036	0.129	0.006	10x	10x	10x	2 (10x)
		0.00	0.270	0.056	0.150	0.040	0.131	0.003	10x	10x	10x	3^0 (10x)
		0.05	0.304	0.070	0.098	0.004	0.135	0.008	10x	10x	10x	3^0 (10x)
		0.10	0.234	0.069	0.106	0.005	0.134	0.005	10x	10x	10x	3^0 (10x)
	1-25	-0.10	0.327	0.133 (1)[5]	0.189	0.077	0.136	0.007	10x	10x	10x	2 (10x)
		-0.05	0.393	0.150 (2)	0.176	0.048	0.136	0.007	10x	10x	10x	2 (10x)
		0.00	0.280	0.076 (1)	0.119	0.028	0.139	0.012	10x	10x	10x	3^0 (10x)
		0.05	0.289	0.087	0.101	0.003	0.135	0.005	10x	10x	10x	3^0 (10x)
		0.10	0.322	0.121	0.101	0.006	0.134	0.005	10x	10x	10x	3^0 (10x)
25	1-5	-0.10	10.511	3.927 (4)	3.626	1.327	0.755	0.053	10x	10x	10x	2 (10x)
		-0.05	10.051	4.206 (7)	4.090	1.588	0.781	0.030	10x	10x	10x	2 (10x)
		0.00	5.990	2.631	2.609	2.035	0.801	0.050	10x	10x	10x	3^0 (10x)
		0.05	15.409	10.735	0.434	0.016	0.759	0.040	10x	10x	10x	3^0 (10x)
		0.10	10.439	10.105	0.436	0.012	0.743	0.025	10x	10x	10x	3^0 (10x)
	1-10	-0.10	18.688	17.321 (7)	8.507	6.929	1.401	0.251	10x	10x	10x	2 (10x)
		-0.05	11.815	6.556	4.453	0.533	1.396	0.257	10x	10x	10x	2 (10x)
		0.00	12.567	10.660	2.155	2.172	1.476	0.166	10x	10x	10x	3^0 (10x)
		0.05	11.368	5.351	0.839	0.067	1.472	0.099	10x	10x	10x	3^0 (10x)
		0.10	9.423	8.248	0.834	0.077	1.440	0.131	10x	10x	10x	3^0 (10x)
	1-25	-0.10	14.369	10.474 (3)	5.256	1.145	1.547	0.106	10x	10x	10x	2 (10x)
		-0.05	15.314	8.744 (1)	5.174	1.615	1.549	0.146	10x	10x	10x	2 (10x)
		0.00	11.351	3.425 (1)	1.654	1.690	1.487	0.111	10x	10x	10x	3^0 (10x)
		0.05	9.534	1.806	0.863	0.078	1.504	0.126	10x	10x	10x	3^0 (10x)
		0.10	7.701	4.675	0.857	0.086	1.549	0.122	10x	10x	10x	3^0 (10x)

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".

Table B.2. Results Experiment 2 (Part 2).

N	p_i $(1 \le i \le N)$	ε_d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	$c_{\text{relax}} = \max_{\text{lb}}$[2][3]	$c_{\text{relax}} = c_{\text{trans}}$[2]	c_{trans}[4] optimal	optimal structure
50	1-5	-0.10	691.269	663.565 (10)[5]	220.785	181.902	5.694	0.205	10x	10x	10x	2 (10x)
		-0.05	947.133	788.217 (10)	202.217	201.380	5.883	0.327	10x	10x	10x	2 (10x)
		0.00	325.294	199.956 (4)	72.986	116.569	6.570	0.308	10x	10x	10x	3^0 (10x)
		0.05	929.698	761.222	3.431	0.194	6.099	0.375	10x	10x	10x	3^0 (10x)
		0.10	1074.590	746.393 (1)	3.066	0.167	5.798	0.253	10x	10x	10x	3^0 (10x)
	1-10	-0.10	746.446	373.149 (5)	210.280	162.687	21.848	1.342	10x	10x	10x	2 (10x)
		-0.05	437.709	225.877 (5)[1][6]	191.514	132.541	22.155	1.868	10x	10x	10x	2 (10x)
		0.00	330.343	205.301 (5)	39.048	41.599 (2)	21.604	1.542	10x	10x	10x	3^0 (10x)
		0.05	555.855	691.078 (4)	14.253	0.869	21.996	1.671	10x	10x	10x	3^0 (10x)
		0.10	304.587	219.360 (3)	13.999	1.271	21.411	1.680	10x	10x	10x	3^0 (10x)
	1-25	-0.10	924.385	838.577 (6)	302.624	305.140	22.400	0.977	10x	10x	10x	2 (10x)
		-0.05	577.483	520.244 (4)	122.571	27.901	22.369	1.465	10x	10x	10x	2 (10x)
		0.00	586.317	645.154 (5)	64.934	43.618 (2)	22.803	0.964	10x	10x	10x	3^0 (10x)
		0.05	226.821	114.322 (3)	14.218	0.604	22.262	0.765	10x	10x	10x	3^0 (10x)
		0.10	385.995	425.826 (2)	14.580	1.383	22.518	1.499	10x	10x	10x	3^0 (10x)

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".
6. number of experiments that stopped because of memory problems.

Table B.3. Results Experiment 3 (Part 1).

N	p_i $(1 \leq i \leq N)$	ε_p	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	$c_{\text{relax}} = \max_{\text{lb}}$ [2] [3]	$c_{\text{relax}} = c_{\text{trans}}$ [2] [4]	c_{trans} [4] optimal	optimal structure
10	1-5	-10.0	0.208	0.047 (2)[5]	0.100	0.015	0.132	0.011	10x	10x	10x	3^0(10x)
		-5.0	0.205	0.053	0.094	0.005	0.134	0.007	10x	10x	10x	3^0(10x)
		0.0	0.305	0.033	39.994	1.895	62.477	2.964	10x	0x	10x	4 (10x)
		5.0	0.202	0.034	0.117	0.005	0.135	0.008	10x	10x	10x	1 (10x)
		10.0	0.200	0.038 (1)	0.117	0.005	0.128	0.010	10x	10x	10x	1 (10x)
	1-10	-10.0	0.239	0.052	0.094	0.005	0.133	0.008	10x	10x	10x	3^0(10x)
		-5.0	0.237	0.053 (3)	1.225	1.833	0.488	0.689	10x	7x	10x	3^0(7x), 4 (3x)
		0.0	0.303	0.053 (1)	39.926	3.396	62.543	5.883	10x	0x	10x	4 (10x)
		5.0	0.233	0.061 (3)	0.527	0.783	0.784	1.248	10x	7x	10x	1 (7x), 4 (3x)
		10.0	0.193	0.057 (1)	0.117	0.007	0.129	0.010	10x	10x	10x	1 (10x)
	1-25	-10.0	0.243	0.061 (6)	14.275	23.774	8.174	5.765	10x	2x	10x	3^0(2x), 4 (8x)
		-5.0	0.319	0.031 (4)	25.306	5.152	37.213	8.666	10x	0x	10x	4 (10x)
		0.0	0.290	0.047 (3)	41.282	4.873	65.351	7.986	8x	0x	10x	4 (10x)
		5.0	0.347	0.027 (2)	24.745	3.182	40.000	5.503	10x	0x	9x	3^1(1x), 4 (9x)
		10.0	0.223	0.054 (8)	4.837	4.476	7.776	7.251	10x	2x	9x	1 (3x), 4 (7x)
24	1-5	-10.0	6.327	6.367	0.623	0.038	1.116	0.092	10x	10x	10x	3^0(10x)
		-5.0	7.545	8.331	0.670	0.041	1.157	0.068	10x	10x	10x	3^0(10x)
		0.0	21.851	10.469 (3)	—	— <10>[6]	—	— <10>	10x	0x	10x	4 (10x)
		5.0	6.354	4.072	0.907	0.077	1.148	0.084	10x	10x	10x	1 (10x)
		10.0	5.448	2.268	0.992	0.145	1.112	0.076	10x	10x	10x	1 (10x)
	1-10	-10.0	8.035	6.987	0.701	0.053	1.243	0.083	10x	10x	10x	3^0(10x)
		-5.0	12.744	11.169 (6)	238.467	582.446 <4>	283.855	692.233 <4>	10x	5x	10x	3^0(5x), 4 (5x)
		0.0	20.380	14.794 (6)	—	— <10>	—	— <10>	9x	0x	10x	4 (10x)
		5.0	9.233	7.668 (6)	47.763	114.233 <4>	74.528	179.514 <4>	10x	5x	10x	1 (5x), 4 (5x)
		10.0	10.637	6.778 (9)	1.040	0.241	1.257	0.098	10x	10x	10x	1 (10x)
	1-25	-10.0	12.403	9.834 (10)	—	— <10>	4249.340	0.000 <9>	10x	0x	10x	4 (10x)
		-5.0	14.753	10.264 (10)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		0.0	23.022	9.310 (8)	—	— <10>	—	— <10>	9x	0x	10x	4 (10x)
		5.0	17.385	8.397 (9)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		10.0	12.022	7.476	5530.920	1891.751 <8>	5482.936	2106.862 <8>	10x	0x	9x	3^1(1x), 4 (9x)

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".
6. number of experiments that do not finish in 2 hours.

Table B.3. Results Experiment 3 (Part 2).

N	p_i $(1 \le i \le N)$	ε_p	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	$c_{\text{relax}} = \max_{\text{lb}}$[2][3]	$c_{\text{relax}} = c_{\text{trans}}$[2]	c_{trans}[4] optimal	optimal structure
25	1-5	-10.0	4.435	4.268 (3)[5]	0.735	0.063	1.265	0.102	10x	10x	10x	3^0(10x)
		-5.0	7.905	10.330 (1)	0.778	0.070	1.305	0.086	10x	10x	10x	3^0(10x)
		0.0	9.986	11.233	1.621	2.600	1.392	0.094	10x	10x	10x	1,2,3^0(10x)
		5.0	8.872	5.984 (10)	1.073	0.185	1.302	0.086	10x	10x	10x	1 (10x)
		10.0	6.308	3.454 (10)	1.103	0.140	1.221	0.090	10x	10x	10x	1 (10x)
	1-10	-10.0	11.566	11.170 (1)	0.798	0.088	1.407	0.167	10x	10x	10x	3^0(10x)
		-5.0	11.547	8.110	0.822	0.084	1.399	0.121	10x	10x	10x	3^0(10x)
		0.0	15.919	11.771	1.067	0.678	1.469	0.142	10x	10x	10x	1,2,3^0(10x)
		5.0	8.863	7.703 (10)	1.141	0.153	1.393	0.130	10x	10x	10x	1 (10x)
		10.0	9.454	4.850 (6)	1.152	0.174	1.380	0.159	10x	10x	10x	1 (10x)
	1-25	-10.0	20.540	11.324	0.829	0.063	1.487	0.089	10x	10x	10x	3^0(10x)
		-5.0	16.549	10.751	0.832	0.054	1.532	0.128	10x	10x	10x	3^0(10x)
		0.0	15.880	9.332	1.661	2.126	1.577	0.109	10x	10x	10x	1,2,3^0(10x)
		5.0	15.838	7.855 (4)	1.170	0.171	1.515	0.103	10x	10x	10x	1 (10x)
		10.0	17.081	5.044 (10)	1.167	0.133	1.548	0.142	10x	10x	10x	1 (10x)
49	1-5	-10.0	832.903	833.635 (2)[1][6]	9.118	0.746	14.855	1.219	10x	10x	10x	3^0(10x)
		-5.0	876.313	498.286 (2)	9.729	0.888	15.573	1.562	10x	10x	10x	3^0(10x)
		0.0	1217.850	431.723 (2)	10.013	0.735	16.292	1.434	10x	10x	10x	1,2,3^0(10x)
		5.0	1212.052	493.658 (9)	23.557	8.073	14.884	1.076	10x	10x	10x	1 (10x)
		10.0	1642.802	377.151 (10)	21.286	4.266	14.764	1.436	10x	10x	10x	1 (10x)
	1-10	-10.0	1105.675	351.584 (6)	10.649	0.770	17.384	1.223	10x	10x	10x	3^0(10x)
		-5.0	1296.073	319.374 (2)	11.474	1.338	18.096	1.859	10x	10x	10x	3^0(10x)
		0.0	1427.364	363.628 (7)	11.237	0.732	17.984	1.294	10x	10x	10x	1,2,3^0(10x)
		5.0	1363.886	463.857 (9)	34.846	11.830	17.538	1.350	10x	10x	10x	1 (10x)
		10.0	1593.524	287.775 (8)	30.095	8.105	17.190	1.013	10x	10x	10x	1 (10x)
	1-25	-10.0	1060.969	408.761 (8)	11.365	0.952	18.723	1.273	10x	10x	10x	3^0(10x)
		-5.0	1377.096	315.404 (6)	11.847	1.144	18.826	1.258	10x	10x	10x	3^0(10x)
		0.0	1938.588	494.410 (10)	11.753	0.772	19.252	1.174	10x	10x	10x	1,2,3^0(10x)
		5.0	1785.583	364.100 (10)	53.803	17.885	19.044	1.441	10x	10x	10x	1 (10x)
		10.0	1563.616	398.310 (9)	44.127	8.041	18.687	1.266	10x	10x	10x	1 (10x)

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".
6. number of experiments that stopped because of memory problems.

N	p_i $(1 \leq i \leq N)$	ε_p	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	$c_{\text{relax}} = \max_{\text{lb}}$[2] [3]	$c_{\text{relax}} = c_{\text{trans}}$[2] [4]	c_{trans}[4] optimal	optimal structure
50	1-5	-10.0	1773.106	1018.510 (5)[5]	10.862	0.985	16.968	1.252	10x	10x	10x	3^0(10x)
		-5.0	1347.749	514.385 (1)	11.484	1.013	17.771	1.087	10x	10x	10x	3^0(10x)
		0.0	2106.746	576.560 (9)	—	— <10>[6]	—	— <10>	10x	0x	10x	4 (10x)
		5.0	2223.384	760.196 (8)	25.337	5.209	17.327	1.165	10x	10x	10x	1 (10x)
		10.0	1942.521	909.495 (6)	24.775	6.358	16.544	1.231	10x	10x	10x	1 (10x)
	1-10	-10.0	1507.268	547.142 (4)	13.199	1.538	20.469	2.420	10x	10x	10x	3^0(10x)
		-5.0	1622.415	733.069 (6)	13.693	1.776 <6>	21.560	1.984 <6>	10x	4x	10x	3^0(4x), 4 (6x)
		0.0	2253.008	584.856 (9)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		5.0	1586.608	638.793 (10)	32.000	0.000 <9>	19.7000	0.000 <9>	10x	1x	10x	1 (1x), 4 (9x)
		10.0	1991.484	514.456 (9)	39.543	14.556	20.670	2.862	10x	10x	10x	1 (10x)
	1-25	-10.0	1981.637	304.489 (9)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		-5.0	1826.475	579.720 (9)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		0.0	2173.636	983.518 (9)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		5.0	2182.035	230.323 (10)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		10.0	1900.991	219.002 (10)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)

Table B.3. Results Experiment 3 (Part 3).

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".
6. number of experiments that do not finish in 2 hours.

N	p_i $(1 \le i \le N)$	ε_d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	$c_{\text{relax}} = \max_{\text{lb}}$[2,3]	$c_{\text{relax}} = c_{\text{trans}}$[2,4]	c_{trans}[4] optimal	optimal structure
10	1-5	-0.50	0.442	0.263 (1)[5]	0.201	0.058	0.132	0.006	10x	10x	10x	2 (10x)
		-0.25	0.266	0.059	0.172	0.019	0.137	0.007	10x	10x	10x	2 (10x)
		0.00	0.305	0.033	36.978	13.056	55.843	19.773	10x	0x	10x	4 (10x)
		0.25	0.370	0.082 (3)	10.668	3.985	12.853	6.456	10x	0x	10x	4 (10x)
		0.50	0.300	0.073	0.098	0.004	0.136	0.007	10x	10x	10x	3^0 (10x)
	1-10	-0.50	0.453	0.122 (1)	0.184	0.030	0.156	0.075	10x	9x	9x	2 (10x)
		-0.25	0.451	0.052	16.443	2.369	25.040	3.871	10x	0x	9x	3^1 (1x), 4 (9x)
		0.00	0.303	0.053 (1)	32.828	17.591	49.998	26.828	10x	0x	10x	4 (10x)
		0.25	0.390	0.047	23.470	3.797	33.924	6.285	10x	0x	10x	4 (10x)
		0.50	0.297	0.080 (6)	7.518	4.552	8.640	6.004	10x	2x	10x	3^0 (2x), 4 (8x)
	1-25	-0.50	0.513	0.078 (2)	22.486	3.815	34.375	6.364	10x	0x	9x	3^1 (1x), 4 (9x)
		-0.25	0.427	0.044	32.042	7.788	50.666	13.325	10x	0x	10x	4 (10x)
		0.00	0.290	0.047 (3)	34.563	18.520	54.153	29.076	8x	0x	10x	4 (10x)
		0.25	0.409	0.073 (3)	35.053	5.582	53.546	9.513	9x	0x	10x	4 (10x)
		0.50	0.393	0.028 (2)	26.870	4.826	39.666	8.063	10x	0x	10x	4 (10x)
24	1-5	-0.50	8.608	6.334 (4)	7.580	6.642	1.072	0.100	10x	10x	10x	2 (10x)
		-0.25	12.848	8.940 (8)	15.649	9.995	1.118	0.080	10x	10x	10x	2 (10x)
		0.00	21.851	10.469 (3)	—	— <10>[6]	—	— <10>	10x	0x	10x	4 (10x)
		0.25	3.139	0.314	0.671	0.026	1.190	0.058	10x	10x	10x	3^0 (10x)
		0.50	3.151	0.303	0.627	0.025	1.114	0.044	10x	10x	10x	3^0 (10x)
	1-10	-0.50	13.141	5.404 (9)	11.891	12.579	1.231	0.103	10x	10x	10x	2 (10x)
		-0.25	11.893	8.423	3.445	2.231	1.257	0.075	10x	10x	10x	2 (10x)
		0.00	20.380	14.794 (6)	—	— <10>	—	— <9>	9x	0x	10x	4 (10x)
		0.25	8.393	6.745 (4)	1189.367	1634.211	1098.713	1554.613	10x	6x	10x	3^0 (6x), 4 (4x)
		0.50	7.942	7.642	0.693	0.042	1.213	0.081	10x	10x	10x	3^0 (10x)
	1-25	-0.50	11.607	9.231 (6)	581.739	1075.427 <1>	590.177	1093.573 <1>	10x	5x	9x	2 (6x), 4 (4x)
		-0.25	19.281	11.259 (10)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		0.00	23.022	9.310 (8)	—	— <10>	—	— <10>	9x	0x	10x	4 (10x)
		0.25	10.892	9.195 (10)	—	— <10>	—	— <10>	10x	0x	10x	4 (10x)
		0.50	18.301	8.250 (8)	2489.197	2734.050 <4>	2180.215	2692.670 <4>	10x	3x	10x	3^0 (3x), 4 (7x)

Table B.4. Results Experiment 4 (Part 1).

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".
6. number of experiments that do not finish in 2 hours.

Table B.4. Results Experiment 4 (Part 2).

N	p_i $(1 \le i \le N)$	ε_d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 2 average	Algorithm 2 st. dev.	Algorithm 3 average	Algorithm 3 st. dev.	$c_{\text{relax}} = \max_{\text{lb}}$[2][3]	$c_{\text{relax}} = c_{\text{trans}}$[2][4]	c_{trans}[4] optimal	optimal structure
25	1-5	-0.50	10.623	3.769 (4)[5]	5.895	3.713	1.225	0.162	10x	10x	10x	2 (10x)
		-0.25	9.028	4.868	3.345	0.906	1.268	0.146	10x	10x	10x	2 (10x)
		0.00	9.986	11.233	1.621	2.600	1.392	0.094	10x	10x	10x	1,2,3(10x)
		0.25	6.425	5.606 (2)	0.740	0.029	1.281	0.094	10x	10x	10x	3^0 (10x)
		0.50	4.096	0.855 (2)	0.719	0.035	1.243	0.061	10x	10x	10x	3^0 (10x)
	1-10	-0.50	10.388	5.390 (4)	6.081	3.034	1.336	0.103	10x	10x	10x	2 (10x)
		-0.25	17.534	11.561 (10)	11.355	6.686	1.359	0.119	10x	10x	10x	2 (10x)
		0.00	15.919	11.771	1.067	0.678	1.469	0.142	10x	10x	10x	1,2,3^0(10x)
		0.25	11.683	9.168	0.809	0.090	1.387	0.138	10x	10x	10x	3^0 (10x)
		0.50	10.504	7.731 (1)	0.765	0.032	1.320	0.058	10x	10x	10x	3^0 (10x)
	1-25	-0.50	14.340	8.923 (10)	10.168	4.847	1.523	0.192	10x	10x	10x	2 (10x)
		-0.25	15.212	10.236 (3)	2.704	0.696	1.516	0.150	10x	10x	10x	2 (10x)
		0.00	15.880	9.332	1.661	2.126	1.577	0.109	10x	10x	10x	1,2,3^0(10x)
		0.25	15.952	11.355 (1)	0.860	0.075	1.521	0.120	10x	10x	10x	3^0 (10x)
		0.50	16.023	10.869 (1)	0.813	0.049	1.494	0.106	10x	10x	10x	3^0 (10x)
49	1-5	-0.50	446.891	409.844 (7)	143.918	16.206	14.602	1.845	10x	10x	10x	2 (10x)
		-0.25	888.662	535.451 (8)	123.304	44.327	14.944	1.852	10x	10x	10x	2 (10x)
		0.00	1217.850	431.723 (2)	10.013	0.735	16.292	1.434	10x	10x	10x	1,2,3^0(10x)
		0.25	605.249	344.647 (3)	8.857	0.640	13.901	0.872	10x	10x	10x	3^0 (10x)
		0.50	439.889	394.175 (2)	7.917	0.254	13.074	0.400	10x	10x	10x	3^0 (10x)
	1-10	-0.50	1211.965	525.920 (8)	296.375	244.072	16.437	1.277	10x	10x	10x	2 (10x)
		-0.25	1472.543	566.162 (8)	238.795	109.844	17.337	1.906	10x	10x	10x	2 (10x)
		0.00	1427.364	363.628 (7)	11.237	0.732	17.984	1.294	10x	10x	10x	1,2,3^0(10x)
		0.25	1109.450	430.519 (6)	10.890	1.223	17.278	1.895	10x	10x	10x	3^0 (10x)
		0.50	1508.416	779.853 (8)	10.072	1.205	16.321	1.671	10x	10x	10x	3^0 (10x)
	1-25	-0.50	1806.802	472.327 (10)	298.861	92.004	18.264	1.437	10x	10x	10x	2 (10x)
		-0.25	1458.436	623.531 (10)	468.216	269.113	18.534	1.450	10x	10x	10x	2 (10x)
		0.00	1938.588	494.410 (10)	11.753	0.772	19.252	1.174	10x	10x	10x	1,2,3^0(10x)
		0.25	1290.055	373.703 (8)	11.854	0.696	19.401	1.416	10x	10x	10x	3^0 (10x)
		0.50	1262.741	378.343 (8)	11.947	0.834	19.307	1.487	10x	10x	10x	3^0 (10x)

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".

N	p_i $(1 \le i \le N)$	ε_d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]		Algorithm 2 average	Algorithm 2 st. dev.		Algorithm 3 average	Algorithm 3 st. dev.		$c_{\text{relax}} = \max_{\text{lb}}$[2,3]	$c_{\text{relax}} = c_{\text{trans}}$[2,4]	c_{trans}[4] optimal	optimal structure
50	1-5	-0.50	697.053	421.149	(8)[5]	206.419	105.012		17.257	2.463		10x	10x	10x	2 (10x)
		-0.25	1404.381	755.635	(9)	1131.056	1424.334		17.072	1.914		10x	10x	10x	2 (10x)
		0.00	2106.746	576.560	(9)	—	—	<10>[7]	—	—	<10>	10x	0x	10x	4 (10x)
		0.25	1559.855	822.774	(4)	10.371	0.612		16.480	0.804		10x	10x	10x	3^0(10x)
		0.50	274.561	210.333	(2)	9.667	0.762		15.353	0.889		10x	10x	10x	3^0(10x)
	1-10	-0.50	1261.584	851.042	(7)	528.330	1014.582		19.834	2.349		10x	10x	10x	2 (10x)
		-0.25	1413.007	561.330	(10)	1249.610	721.496		20.237	1.611		10x	10x	10x	2 (10x)
		0.00	2253.008	584.856	(9)	—	—	<10>	—	—	<10>	10x	0x	10x	4 (10x)
		0.25	1831.104	876.932	(3)	13.001	1.478		20.154	2.148		10x	10x	10x	3^0(10x)
		0.50	1247.488	798.721	(5)[1][6]	12.668	1.878		19.860	2.699		10x	10x	10x	3^0(10x)
	1-25	-0.50	2321.525	784.202	(10)	732.124	331.802		21.409	1.317		10x	10x	10x	2 (10x)
		-0.25	2320.323	804.784	(9)	399.443	173.094		21.862	1.448		10x	10x	10x	2 (10x)
		0.00	2173.636	983.518	(9)	—	—	<10>	—	—	<10>	10x	0x	10x	4 (10x)
		0.25	1618.019	351.811	(7)	13.868	1.342	<3>	21.694	2.181	<3>	10x	7x	10x	3^0(7x), 4 (3x)
		0.50	1370.961	442.241	(7)	13.996	1.265		21.415	1.339		10x	10x	10x	3^0(10x)

1. calculated over 10 experiments, in seconds (CPU time).
2. $c_{\text{relax}} = c_{\text{opt}}(LP_{\text{relax}})$.
3. $\max_{\text{lb}} = \max\{2p_{\max} + 2d, 2Nd, \sum_{j=1}^{N} p_j\}$.
4. $c_{\text{trans}} = c_{\text{opt}}(LP_{\text{trans}})$.
5. number of experiments that finished with "solution status 102".
6. number of experiments that stopped because of memory problems.
7. number of experiments that do not finish in 2 hours.

Table B.4. Results Experiment 4 (Part 3).

N	p_i $(1 \leq i \leq N)$	d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 4 average	Algorithm 4 st. dev.
10	1-5	0.0	0.305	0.033	0.412	0.151
	1-10	-5.0	0.237	0.053 (3)[2]	0.150	0.086
	1-10	0.0	0.303	0.053 (1)	0.327	0.196
	1-25	-10.0	0.243	0.061 (6)	0.220	0.083
	1-25	-5.0	0.319	0.031 (4)	0.347	0.090
	1-25	0.0	0.290	0.047 (3)	0.339	0.177
24	1-5	0.0	21.851	10.469 (3)	9.650	8.849
	1-10	-5.0	12.744	11.169 (6)	4.964	5.625
	1-10	0.0	20.380	14.794 (6)	14.034	9.927
	1-25	-10.0	12.403	9.834 (10)	8.877	4.972
	1-25	-5.0	14.753	10.264 (10)	10.047	5.023
	1-25	0.0	23.022	9.310 (8)	10.710	9.482
50	1-5	0.0	2106.746	576.560 (9)	578.603	624.928
	1-10	-5.0	1622.415	733.069 (6)	743.179	649.392
	1-10	0.0	2253.008	584.856 (9)	890.882	508.027
	1-25	-10.0	1981.637	304.489 (9)	1260.353	523.637
	1-25	-5.0	1826.475	579.720 (9)	1200.093	470.400
	1-25	0.0	2173.636	983.518 (9)	1294.820	571.219

1. over 10 experiments, in seconds (CPU time).
2. number of experiments that finished with “solution status 102”.

Table B.5. Results Experiment 3 (Algorithm 4).

N	p_i $(1 \leq i \leq N)$	d	Algorithm 1 average[1]	Algorithm 1 st. dev.[1]	Algorithm 4 average	Algorithm 4 st. dev.
10	1-5	0.00	0.305	0.033	0.412	0.151
	1-5	0.25	0.370	0.082 (3)[2]	0.405	0.085
	1-10	-0.25	0.451	0.052	0.591	0.171
	1-10	0.00	0.303	0.053 (1)	0.327	0.196
	1-10	0.25	0.390	0.047	0.356	0.073
	1-10	0.50	0.297	0.080 (6)	0.285	0.117
	1-25	-0.50	0.513	0.078 (2)	0.696	0.202
	1-25	-0.25	0.427	0.044	0.539	0.164
	1-25	0.00	0.290	0.047 (3)	0.339	0.177
	1-25	0.25	0.409	0.073 (3)	0.398	0.147
	1-25	0.50	0.393	0.028 (2)	0.353	0.172
24	1-5	0.00	21.851	10.469 (3)	9.650	8.849
	1-10	0.00	20.380	14.794 (6)	14.034	9.927
	1-10	0.25	8.393	6.745 (4)	4.485	5.647
	1-25	-0.50	11.607	9.231 (6)	8.074	7.701
	1-25	-0.25	19.281	11.259 (10)	19.694	4.680
	1-25	0.00	23.022	9.310 (8)	10.710	9.482
	1-25	0.25	10.892	9.195 (10)	6.992	5.206
	1-25	0.50	18.301	8.250 (8)	4.767	4.626
50	1-5	0.00	2106.746	576.560 (9)	578.603	624.928
	1-10	0.00	2253.008	584.856 (9)	890.882	508.027
	1-25	0.00	2173.636	983.518 (9)	1294.820	571.219
	1-25	0.25	1618.019	351.811 (7)	504.83	803.201

1. over 10 experiments, in seconds (CPU time).
2. number of experiments that finished with “solution status 102”.

Table B.6. Results Experiment 4 (Algorithm 4).

N	p_j $(1 \leq j \leq N)$	d	Algorithm 4 average[1]
100	1-10	2.75	5.6 min.
150	1-10	2.75	24.0 min.
200	1-10	2.75	78.4 min.
250	1-10	2.75	—— [5][2]

1. calculated over 5 experiments, in seconds (CPU time).
2. number of experiments that stopped because of memory problems.

Table B.7. Experiment 1, Algorithm 4 for large instances.

N	p_j $(1 \leq j \leq N)$	ε_p ε_d	Algorithm 4 average[1]
74	1-10	0.00	16.9 min.
75	1-10	0.00	1.2 min
99	1-10	0.00	4.3 min.
100	1-10	0.00	ca. 11 hours
149	1-10	0.00	21.0 min.
150	1-10	0.00	ca. 20 hours
199	1-10	0.00	106 min.

1. calculated over 5 experiments, in seconds (CPU time).

Table B.8. Experiment 3/4, Algorithm 4 for large instances.

Bibliography

AGNETIS, A., D. PACCIARELLI, AND F. ROSSI [1997], Batch scheduling in a two-machine flow shop with limited buffer, *Discrete Applied Mathematics* **72**, 243–260.

BAKER, K.R. [1974], *Introduction to Sequencing and Scheduling*, John Wiley & Sons, New York.

BARD, J.F. [1988], A heuristic for minimizing the number of switches on a flexible machine, *IIE Transactions* **20**, 382–391.

BERTOSSI, A.A., AND M.A. BONUCCELLI [1985], A polynomial feasibility test for preemptive periodic scheduling of unrelated processors, *Discrete Applied Mathematics* **12**, 195–201.

BLAZEWICZ, J., K.H. ECKER, E. PESCH, G. SCHMIDT, AND J. WEGLARZ [1996], *Scheduling Computer and Manufacturing Processes*, Springer-Verlag, Berlin.

BONDY, J.A., AND U.S.R. MURTY [1976], *Graph Theory with Applications*, Macmillan, London and Elsevier, New York.

BRAUNER, N., AND G. FINKE [1998], *On Cycles and Permutations in Robotic Cells*, Research report RR10-10-I-, Laboratory Leibniz-IMAG, Grenoble Cedex, France.

BRUCKER, P. [1998], *Scheduling Algorithms* (Second ed.), Springer-Verlag, Berlin.

CHRÉTIENNE, P., E.G. COFFMAN, JR., J.K. LENSTRA, AND Z. LIU (eds.) [1995], *Scheduling Theory and its Applications*, John Wiley & Sons, Chichester.

COFFMAN, JR., E.G. (ed.) [1976], *Computer and Job-Shop Scheduling Theory*, John Wiley and Sons, New York.

CRAMA, Y. [1997], Combinatorial optimization models for production scheduling in automated manufacturing systems, *European Journal of Operational Research* **99**, 136–153.

CRAMA, Y., AND J. VAN DE KLUNDERT [1997], Cyclic scheduling of identical parts in a robotic cell, *Operations Research* **45**, 952–965.

FRENCH, S. [1982], *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, Chichester and Halsted Press [John

Wiley & Sons], New York.

GAALMAN, G.J.C., AND W.M. NAWIJN [1996], Tool sharing in parallel part production, *International Journal of Production Economics* **46–47**, 521–533.

GAREY, M.R., AND D.S. JOHNSON [1984], *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Fourth ed.), Freeman, New York.

GRAY, A.E., A. SEIDMANN, AND K.E. STECKE [1993], A synthesis of decision models for tool management in automated manufacturing, *Management Science* **39**, 549–567.

HALL, N.G., H. KAMOUN, AND C. SRISKANDARAJAH [1997], Scheduling in robotic cells: Classification, two and three machine cells, *Operations Research* **45**, 421–439.

HAMMER, H. [1988], Neuartiges Werkzeugschnellwechelverfahren für Bearbeitungszentren, *Zeitschrift für wirtschaftlichen Fabrikbetrieb* **83**, 191–196.

HANEN, C. [1994], Study of a NP-hard cyclic scheduling problem: The recurrent job-shop, *European Journal of Operational Research* **72**, 82–101.

HANEN, C., AND A. MUNIER [1995], Cyclic scheduling on parallel processors: an overview, *Scheduling Theory and its Applications*, Wiley, Chichester, 193–226.

HURINK, J.L., AND S. KNUST [2000], Flow-shop problems with transportation times and a single robot, *Discrete Applied Mathematics*, To appear.

KNOOP PATHUIS, V.R.J. [2000], Scheduling voor twee machines met toolsharing en één transport-robot, Master's thesis, University of Twente, Enschede, The Netherlands, (In Dutch).

KORST, J.H.M. [1992], *Periodic Multiprocessor Scheduling*, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, the Netherlands.

KUCHINIC, A.E., AND A. SEIDMANN [1988], Tool management in automated manufacturing: Operational issues and mathematical models, *Proceedings of the International Industrial Engineering Conference*, 379–382.

KUIJPERS, C.M.H. [2001], *Cyclic Machine Scheduling with Tool Transportation – Additional Calculations*, Memorandum 1580, University of Twente, Enschede, The Netherlands.

LABETOULLE, J. [1974], Some theorems on real time scheduling, in: E. Gelenbe and R. Mahl (eds.), *Computer Architectures and Networks*, North-Holland, Amsterdam, 285–298.

LAWLER, E.L., J.K. LENSTRA, A.H.G. RINNOOY KAN, AND D.B. SHMOYS [1993], Sequencing and scheduling: Algorithms and complexity., in: S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.), *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 445–522.

LAWLER, E.L., AND C.U. MARTEL [1981], Scheduling periodically occurring tasks on multiple processors, *Information Processing Letters* **12**, 9–12.

LEUNG, J.Y.-T., AND M.L. MERRILL [1980], A note on preemptive scheduling of periodic, real-time tasks, *Information Processing Letters* **11**, 115–118.

LIU, C.L., AND J.W. LAYLAND [1973], Scheduling algorithms for multiprogramming in a hard-real-time environment, *Journal of the Association for Computing Machinery* **20**, 46–61.

MAHADEV, N.V.R, PH. SOLOT, AND D. DE WERRA [1993], The cyclic compact open-shop scheduling problem, *Discrete Mathematics* **111**, 361–366.

MASON, F. [1986], Computerized cutting tool management, *American Machinist & Automated Manufacturing* **130**, 105–132.

MCCORMICK, S.T., AND U.S. RAO [1994], Some complexity results in cyclic scheduling, *Mathematical and Computer Modelling* **20**, 107–122.

NEMHAUSER, G.L., AND L.A. WOLSEY [1999], *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, (Reprint of the 1988 original).

PAPADIMITRIOU, C.H., P. SERAFINI, AND M. YANNAKAKIS [1993], Computing the throughput of a network with dedicated lines, *Discrete Applied Mathematics* **42**, 271–278.

PAPADIMITRIOU, C.H., AND K. STEIGLITZ [1982], *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J.

PINEDO, M. [1995], *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, Englewood Cliffs, NJ.

ROUNDY, R. [1992], Cyclic schedules for job-shops with identical jobs, *Mathematics of Operations Research* **17**, 842–865.

SCHRIJVER, A. [1998], *Theory of Linear and Integer Programming* (New ed.), John Wiley & Sons, Chichester.

SETHI, S.P., C. SRISKANDARAJAH, G. SORGER, J. BLAZEWICZ, AND W. KUBIAK [1992], Sequencing of parts and robot moves in a robotic cell, *International Journal of Flexible Manufacturing Systems* **4**, 331–358.

STRUSEVICH, V.A. [1999], A heuristic for the two-machine open-shop scheduling problem with transportation times, *Discrete Applied Mathematics* **93**, 287–304.

TANG, C.S., AND E.V. DENARDO [1988], Models arising from a flexible manufacturing machine. Part 1: Minimization of the number of tool switches, *Operations Research* **36**, 767–777.

TOMEK, P. [1986], Tooling strategies related to FMS management, *Flexible Manufacturing Systems Management* **4**, 102–107.

VEERAMANI, D., D.M. UPTON, AND M.M. BARASH [1992], Cutting-tool management in computer-integrated manufacturing, *International Journal of Flexible Manufacturing Systems* **3–4**, 237–265.

VIEHWEGER, B. [1988], Für Bearbeitungszentren und Flexibele Fertigungssysteme: Tool-management in Zukunft immer Bedeutender, *Industrie-Anzeiger* **110**, 24–26.

WEGNER, R.R. [1997], A Provisional Overview of Periodic Scheduling, Second draft.

WERRA, D. DE, AND M. WIDMER [1990], Loading problems with tool management in flexible manufacturing systems: A few integer programming models, *International Journal of Flexible Manufacturing Systems* **3**, 71–82.

Symbol Index

Linear mixed-integer programs

(Decision) variables

Revenue

$R_m(t)$	number of processed jobs on machine m at time t	17
$R(m)$	maximal average production using m machines	40
$\overline{R}(m)$	maximal average production using m machines (Assumption 2.2 relaxed)	41

Graphs

$G=(V,A)$	graph	43
$G(S)$	critical graph of schedule S	44
V	vertex set	43
V_p	vertices representing processings	44
V_t	vertices representing transports	44
A	arc set	43
w_v	weight of vertex	43
w_a	weight of arc	44

Experiments

$\mathcal{N}$	set of values of N	118
UB_p	set of upper bounds on the processing times p_j	118
n_{inst}	number of instances processed	118
d^*	d such that $2Nd=\sum_{j=1}^{N} p_j$	121
d^{**}	d such that $2Nd=\sum_{j=1}^{N} p_j=2p_{\max}+2d$	121
$p_{\max}^{**}$	$p_{\max}$ such that $2Nd=\sum_{j=1}^{N} p_j=2p_{\max}+2d$	121
ε_d	deviation of transport time d	122
ε_p	deviation of processing time $p_{\max}$	122
$[.]$	experiments with memory problems	149
$<.>$	experiments with run time exceeded	152
$(.)$	experiments with solution status 102	148

Proof of Theorem 5.1

P_a, Q_a	subsets of tasks	56
U,V	subsets of tasks	57,76
$l(X)$	length of X	57
$l(X\|<j)$	length of X before j	132
$l(X\|>j)$	length of X after j	132
$b(P_a)^1$	begin of P_a on machine 1	57
$e(P_a)^1$	end of P_a on machine 1	57

Author Index

Subject Index

Summary

This thesis concerns a cyclic machine scheduling problem with characteristic feature that also tools have to be transported. We speak about a *machine scheduling* problem, because it deals with the question how to make optimal use of a set of machines for the processing of a set of tasks. It is a *cyclic* scheduling problem because the objective is to find a schedule that does not have a real starting or finishing point, but that can be continuously repeated. An example of a cyclic schedule is, e.g., the time table that can be found at a Dutch train station. This time table is a cyclic schedule that is repeated every hour.

For the processing of tasks, machines frequently need certain tools. These tools may be very expensive. If this is the case, then it can be lucrative that the machines share the tools. This, however, has the consequence that the tools have to be transported between the machines. This can, e.g, be done by a person, a conveyer belt or a robot. In the problem that is considered in this thesis, a robot is used. This robot can only transport one tool at a time. Depending on the velocity of the robot and the times that the machines need to process the tasks, *the processing times of the tasks*, it may happen that a machine cannot start with a next task, because the required tool has not arrived yet at that machine. This implies that the machine becomes idle for some time. This, obviously, is not very efficient. In general, one will try to keep the machines busy as much as possible, in order to maximize the average production per unit of time. In this thesis, we try to answer the question with what cyclic schedule, given the velocity of the robot and the processing times of the tasks, the average production per unit of time can be maximized.

Before a model of the cyclic problem can be given, first the *linear version* of the problem is considered, in which there is still a clear starting point. By adapting the mathematical formulation of the linear problem, the cyclic problem can be formulated as a *linear mixed-integer program.*

In this thesis we mainly consider the cyclic problem for the case that there are two machines. For this case a variety of properties of optimal schedules are derived. We find, a.o., upper and lower bounds on the optimal length of a schedule and we see that there may actually exist schedules the length of which is equal to one of these lower bounds. If there are only a very small number of different

tasks, then an optimal schedule can directly be given. We also see that the order in which the robot carries out the different tool transports, the *transport strategy*, is very important.

Optimal schedules can be characterized by a certain type of graph, which we call a *critical graph*. A critical graph always contains a cycle, which we call a *critical cycle*. Such a critical cycle represents a list of processings and transports, of which the total duration determines the length of the schedule. Depending on how such a list looks like, we distinguish different types of critical cycles. Interesting is, that only five different types of critical cycles are relevant, since there always exists an optimal schedule with a critical cycle of one of the five types.

We tried to use the above result for determining the *complexity* of the cyclic problem for two machines. However, we did not manage to answer the question whether the problem is solvable in polynomial time. We also do not know if the problem belongs to the class of $\mathcal{NP}$*-complete problems*.

Nevertheless, we did use the above result for developing an algorithm for solving the cyclic problem for two machines, with which an optimal schedule is found much faster than by just solving the original linear mixed-integer program. Furthermore, it turns out that there exists a simple heuristic that in almost all cases results in an optimal schedule.

Samenvatting

Dit proefschrift behandelt een cyclisch machine scheduling probleem, dat als opvallend kenmerk heeft dat er gereedschappen bij worden vervoerd. We spreken van een *machine scheduling* probleem, omdat het gaat over de vraag op welke wijze we optimaal gebruik kunnen maken van een aantal machines voor het uitvoeren van een aantal bewerkingen. Het is een *cyclisch* scheduling probleem omdat we uiteindelijk een schedule (schema, plan) willen vinden dat geen echt begin en eind heeft, maar continu kan worden herhaald. Een voorbeeld van een cyclisch schedule is bijvoorbeeld de dienstregeling van de Nederlandse Spoorwegen. Dat is een cyclisch schedule dat elk uur wordt herhaald.

Machines hebben voor het uitvoeren van bewerkingen vaak bepaalde gereedschappen nodig. Deze gereedschappen kunnen erg kostbaar zijn. Als dit het geval is, dan kan het voordelig zijn om de machines gereedschappen te laten delen. Dat betekent dan wel dat de gereedschappen tussen de machines op en neer moeten worden vervoerd. Dit kan worden gedaan door bijvoorbeeld een persoon, een lopende band of een robot. Bij het probleem dat wordt bekeken in dit proefschrift wordt een robot gebruikt. Deze robot kan maar één gereedschap tegelijkertijd vervoeren. Afhankelijk van de snelheid van de robot en de tijden die de machines nodig hebben voor het uitvoeren van de bewerkingen, *de bewerkingstijden*, kan het voorkomen dat een machine niet met een volgende bewerking kan beginnen omdat het benodigde gereedschap nog niet op de machine aanwezig is. Dit houdt dan in dat de machine een tijdje stil komt te staan. Dit is natuurlijk niet erg efficiënt. Over het algemeen zal men proberen om de machines zo veel mogelijk continu door te laten werken, om in totaal een zo hoog mogelijke productie per tijdseenheid te behalen. In dit proefschrift proberen we een antwoord te geven op de vraag met welk cyclisch schedule, gegeven de snelheid van de robot en de bewerkingstijden, de productie per tijdseenheid kan worden gemaximaliseerd.

Voordat er een model van het cyclische probleem kan worden opgesteld, wordt er eerst gekeken naar de *lineaire versie* van het probleem, waarbij er nog wel een duidelijk begintijdstip is. Door aanpassing van de wiskundige omschrijving van het lineaire probleem kan uiteindelijk het cyclische probleem als een *geheeltallig lineair programmeringsprobleem* geformuleerd worden.

In dit proefschrift bekijken we voornamelijk het cyclische probleem in het

geval dat er precies twee machines zijn. Voor dit geval worden allerlei eigenschappen van optimale schedules afgeleid. Zo vinden we, o.a., onder- en bovengrenzen op de optimale lengte van een schedule en zien we dat er vaak optimale schedules bestaan waarvan de lengte gelijk is aan de waarde van een van die ondergrenzen. In het geval dat er maar een heel klein aantal verschillende bewerkingen bestaan kunnen we direct een optimaal schedule geven. Ook zien we dat de volgorde waarin de robot de gereedschappen vervoert, *de transportstrategie*, erg belangrijk is.

Optimale schedules blijken te kunnen worden gerepresenteerd door een speciaal soort graaf, die we een *kritieke graaf* noemen. Een kritieke graaf bevat altijd een cykel, die we *kritieke cykel* noemen. Zo een kritieke cykel representeert een lijst van bewerkingen en/of transporten, waarvan de gezamenlijke duur de lengte van het schedule bepaalt. Afhankelijk van hoe zo een lijst eruit ziet, onderscheiden we verschillende soorten kritieke cykels. Interessant is dat er maar vijf verschillende soorten kritieke cykels van belang te zijn, omdat er, ongeacht wat de snelheid van de robot en de bewerkingstijden zijn, altijd een optimaal schedule bestaat met een kritieke cykel van één van de vijf soorten.

Het bovenstaande resultaat hebben we geprobeerd te gebruiken om iets te zeggen over de *complexiteit* van het cyclische probleem voor twee machines. Het is echter niet gelukt om een antwoord te vinden op de vraag of het probleem oplosbaar is in polynomiale tijd. Ook weten we niet of het probleem behoort tot de klasse van $\mathcal{NP}$*-complete* problemen.

Wel hebben we met behulp van het bovenstaande resultaat een algoritme kunnen ontwikkelen dat voor het cyclische probleem met twee machines veel sneller een optimaal schedule oplevert dan het standaard oplossen van het oorspronkelijke geheeltallige lineaire programmeringsprobleem. Ook blijkt er een eenvoudige heuristiek te bestaan die in vrijwel alle gevallen een optimaal schedule oplevert.

About the author

Cindy Kuijpers was born on July 6, 1968, in Nijmegen, the Netherlands. In 1986, after completion of her secondary school education (pre-university education) at the Merletcollege in Cuijk, she started to study technical mathematics at the Eindhoven University of Technology. She carried out her final project, which concerned the determination of optimal scan strategies for a wafer stepper (device for IC production), at the Philips Research Laboratories in Eindhoven. Her Master's thesis, which was written under the supervision of prof.dr. J.K. Lenstra, was awarded the Thesis Award 1994 of the Netherlands Society for Statistics and Operational Research (VVS). In 1994, she joined the Department of Computer Science and Artificial Intelligence of the University of the Basque Country in San Sebastián, Spain, where she worked on the use of genetic algorithms for some combinatorial optimization problems with a statistical background. In 1996, she became a Ph.D. student in the Operations Research and Mathematical Programming group of prof.dr. U. Faigle at the Faculty of Mathematical Sciences of the University of Twente. Her research concerned a cyclic scheduling problem, and resulted in this thesis.