INTRODUCTION

For several decades now, scheduling has been an attractive research area within combinatorial optimisation. In general, scheduling is the process of assigning a timetable or plan of production to a set of tasks, commonly called *jobs*. Depending on the respective application, a variety of models have been developed in scheduling theory.

Figure 1: Images of the Potsdamer Platz in Berlin: The left image shows the construction site of the Potsdamer Platz in 1998, when it was Europe's biggest construction site, while the right image shows the Potsdamer Platz now, in 2003.

One of those standard models is captured by *project scheduling*. As the nomenclature already suggests, this model describes whatever one could call a project. A project might be a construction, a chemical process, a spacecraft launch or a transportation problem of heavy load. The left picture of Figure 1 shows an example of such a project: the construction site of the Potsdamer Platz in Berlin in 1998. This construction site was the largest construction site in Europe at that time. The right picture shows the result of the project, the Potsdamer Platz now, four years later. It should be clear that there is much to optimise and many activities to schedule in such a gigantic project.

The different activities that have to be undertaken in a project are represented by the jobs. We assume that we know in advance how long every activity in the project will take to be finished. This time requirement is represented by *processing times* on the jobs which are usually nonnegative. In most problems there arise interdependencies between certain jobs, that is some jobs can only be started after

some other jobs have been finished. On a construction site one can think of plastering the walls, which can certainly only be done after the walls have been built up. These dependencies are modeled by *precedence constraints*, commonly depicted in a *precedence graph*. In addition, jobs compete for common limited resources, such as commodities, machines, or labour, commonly abstracted as *resource constraints*. Probably the most relevant goal in project scheduling is to complete the project as early as possible. In scheduling theory the goal of the problem under consideration is called the *objective* or the *objective function* as in general, this goal can be expressed as a function of the completion times of the jobs. Besides the project completion time, the so-called *makespan*, the objective might be to meet one or several deadlines, or to minimise the sum of the completion times of the jobs.

In *machine scheduling* the focus is on the major restriction of the scheduling problem, the machine constraints. A set of jobs has to be scheduled on one or several machines in order to optimise the given objective function. The machines represent processors, workers or assembly lines. As such, the main applications of machine scheduling can be found in computer science and industrial production. Of course the classification between project and machine scheduling problems is floating. Although most parts of this thesis are in the field of project scheduling, Chapter 3 explicitly focuses on machine scheduling problems.

In this thesis we consider a special model of scheduling, namely scheduling with AND/OR-networks. Basically, an AND/OR-network is a directed graph or digraph for short. In contrast to standard digraphs, an AND/OR-network has two different sorts of nodes, AND- and OR-nodes, which represent the following constraints. An AND-node is just a normal node with the meaning that the job it represents can be processed as soon as every job preceding it in the graph has been completed. An OR-node on the other hand can be scheduled as soon as at least one of its predecessors has been finished. These generalised precedence constraints provide a good model for a variety of applications. We want to discuss two of them in more detail to motivate our effort on the subject.

The first example is *resource constrained project scheduling*. As already mentioned above, in this application the task is to schedule a set of jobs such that all precedence constraints between the jobs are respected. In addition, the jobs have individual demands for certain limited resources. The precedence constraints are given by the precedence graph. The *resource constraints* can be given as a set of resources, the amount that is available of each resource and the demand of each job for each of the resources. In this setting, we have to take care that at any point in time, the total demand for each of the resources of the jobs that are processed at that time does not exceed the availability of the resource. Again for an example, consider a construction site. If we have to paint the walls of five rooms in a flat, but we have only employed two painters, we cannot do it in par-

allel. These resource restrictions can also be represented in a more abstract way by *minimal forbidden sets*. A set of jobs without any precedence dependencies between them is called *forbidden*, if their total demand of some resource exceeds the available amount of that resource. Thus painting the five rooms, for example, is a forbidden set of jobs. A forbidden set is *minimal forbidden*, if any proper subset of it can be scheduled in parallel. In our case, painting any three of the five rooms constitute a minimal forbidden set, as our two painters can do any two rooms at the same time. A straightforward approach to resolve the resource conflicts given by a set of minimal forbidden sets would be to simply fix for every forbidden set a precedence relation between two of the jobs in the set. This technique reduces the resource constrained scheduling problem to a precedence-only constrained scheduling problem, which is easy to solve. Nevertheless, we might make a bad choice by fixing these relations. A less strict way to resolve the resource conflict, is to select one job in each minimal forbidden set, which then has to wait for the completion of at least one of the other jobs in its set. We call the job that is selected the *waiting job*. The pair consisting of a forbidden set without its waiting job and the waiting job is called a *waiting condition*. Of course we can still make a bad choice with the waiting job, but at least it does not have to wait for one previously selected job in the forbidden set, but can instead start as soon as an arbitrary one of the other jobs in the forbidden set has been completed. The precedence graph together with a set of waiting conditions can be represented by an AND/OR-network. For every waiting condition we introduce an OR-node which is a direct successor of each job in the forbidden set except the waiting job. The waiting job is itself a direct successor of the OR-node. In this model, the precedence constraints represented by the AND/OR-network capture the constraints of the waiting conditions and thus resolve the resource conflicts.

The second application are assembly and disassembly problems. As the name already suggests, those problems arise in manufacturing processes, for example. Consider the problem of disassembling a certain product on an assembly line. Due to the geometry of the product, there might be different possibilities how to actually disassemble the product or parts of it. A little example is given in Figure 2. The left figure represents the product and we want to remove part number 7, to repair it, for example. There exist four possible directions to remove a part of the product, indicated by the thick black arrows. The right figure presents the AND/OR-network that corresponds to the precedence constraints appearing in our problem. The circles are AND-nodes and the shaded squares are the OR-nodes of the network. To remove part 7, we can either remove part 5 and 6 to the south or part 4 to the west or part 3 to either the north or the east. To remove part three to the north, we need to remove part 1 to the north, and to remove part 3 to the east we have to remove part 2 to the east first. A schematic overview over different variants of assembly problems has been presented in Goldwasser and Motwani

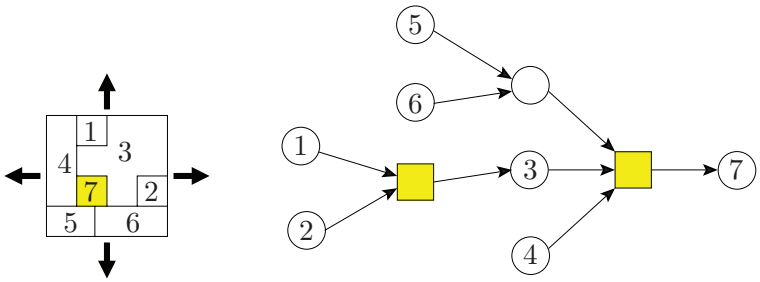

Figure 2: The left figure displays a product that has to be disassembled. The thick arrows mark the possible directions in which a part of the product may be removed. The right figure presents the AND/OR-network that corresponds to the precedence relations occurring in this disassembly instance.

(1999).

Chapters 3 and 4 are motivated by the problem of scheduling jobs restricted by AND/OR precedence constraints such that either the makespan or the sum of the weighted completion times of the jobs are minimised. The focus of Chapter 2 in contrast is slightly different. Consider again the construction site of the Potsdamer Platz displayed in Figure 1. At an early stage of the project, the project plan and the scheduling model representing this plan have to be designed. At this stage, the exact data is not known and has to be estimated. Even after thorough considerations, it is not possible to take everything into account. Especially in those long term undertakings, there is a high uncertainty in the processing times of the jobs, the availability of resources, and even the actual interdependencies between the jobs. This problem is on one hand attended to by *stochastic scheduling*. In stochastic scheduling, (parts of) the input data is assumed to be subject to random fluctuations. Thus the solution is not a schedule but a plan of procedure, a so-called *scheduling policy*, based on the past evolution of the instance. We will not pursue this approach and refer to the respective literature for this matter. Another possibility to attend to uncertainties in the input data is what could be called a *sensitivity analysis* of the network. We restrict our focus on changes in processing times of jobs. Some of the jobs in a project are not *critical* in the sense that, if they take longer, they do not increase the makespan. Whereas other jobs may directly influence the project completion time. Most of the time any delays in the project completion involve high costs so that it might be better to invest some money into the prolonged critical job, to keep its processing time fixed. In any case, a thorough analysis of the problem structure can provide the project planner with a useful tool to handle any external influences. Those questions have been discussed intensively for the case of standard precedences. Whereas for AND/OR-networks, to our knowledge, only the so-called *time-cost trade-off*

problem has been addressed, which can be attributed to this area of research. We will focus on a sensitivity analysis in AND/OR-networks in Chapter 2.

Outline of the Thesis

This thesis is divided into four chapters that will be described in detail in the following. The first chapter is supposed to introduce the reader into the basic concepts that will be used in all other chapters. Chapters 2 to 4 are essentially independent of each other. Throughout the thesis we consider AND/OR precedence constrained jobs. First we examine structural issues of AND/OR-networks and the behaviour of schedules in AND/OR-networks subject to changes in processing times. The next chapter will tighten the restrictions imposed among the jobs by additionally taking machine constraints into account. Chapter 4 then goes into the other direction and considers a relaxation of the AND/OR constraints, in which we are allowed to exchange the waiting job.

Chapter 1 is dedicated to give an introduction into scheduling with AND/OR-networks. First we specify the data of the scheduling model used and review basic graph theoretic notation needed to describe our model. In Remark 1.1.2 important assumptions about the structure of the AND/OR-networks considered in this thesis are listed and justified. A standard classification scheme will enable us to describe efficiently the respective model under consideration. Section 1.2 concentrates on AND/OR-networks and previous work on scheduling in AND/OR-networks, including the presentation of an algorithm to check for *feasibility* of AND/OR-networks and a Dijkstra-like algorithm to compute an *earliest start schedule* for the jobs. Important features of the earliest start schedule are that it is optimal for several objectives such as the makespan and that it is monotone as a function on the processing times. We close the chapter by a short presentation of *directed hypergraphs* and publications on directed hypergraphs related to our work. A directed hypergraph is a graph theoretic concept equivalent to an AND/OR-network.

Chapter 2 presents a thorough qualitative and quantitative analysis of AND/OR-networks, respectively their earliest start schedule with respect to a strictly positive processing time vector. We start with a review of the concept of criticality in standard precedence graphs. Inspired by the different notions of criticality in standard networks, we define four different types of critical jobs and sets in AND/OR-networks, namely *cut*-, *path*-, *delay*-, and *bulk-critical* jobs and sets. A cut-critical set is an inclusion-minimal set of jobs with the property that an arbitrarily small decrease of the processing times of all jobs in the set results in a

decrease of the makespan. A path-critical set in contrast is inclusion-minimal with the property that it preserves the makespan, even if the processing times of all jobs not in the path-critical set are decreased. Delay- and bulk-critical sets are defined similarly but with respect to an increase in processing times. A job is critical, if it is included in a critical set. We prove that in each of the four cases, the system of all critical sets of one type is a clutter. In particular, the cut-critical sets are the blocking clutter of the path-critical sets and the delay-critical sets are the blocking clutter of the bulk-critical sets. Thus, a cut-critical job is also path-critical and a delay-critical job is also bulk-critical, but in general, for a job being cut-critical does not imply being delay-critical and being delay-critical does not imply being cut-critical. Both pairs of blocking clutters do not have the *Max-Flow-Min-Cut property* and we prove that the gap between a maximum flow on the path-critical sets and the minimal costs of a cut-critical set, for example, is at least as big as the integrality gap of SET COVER. This fact already indicates that the time-cost trade-off problem, which we will treat later, might be hard. Next we extend the notion of criticality to OR-nodes as well by defining the *closure* of path- and bulk-critical sets. For every critical set we prove that the change in processing times does not affect the start times of the jobs in the set. This property is the main ingredient for obtaining a number of later results. A purely structural representation of path- and bulk-critical sets closes the basic studies of critical sets and jobs.

The next section is dominated by complexity matters. We notice that it is easy to find a critical set in each of the four cases. In contrast to this, it is NP-complete to decide whether a certain job is cut-, path-, delay-, or bulk-critical. This result is established by a reduction from the satisfiability problem (SAT), where it is remarkable that two different reductions are needed for the four cases, one for cut- and path-critical jobs and one for delay- and bulk-critical jobs. For all other results about critical jobs or sets, the proofs for the four cases can be accomplished in a way similar to each other. In an excursion into logic, we ascertain a close relation between critical sets in AND/OR-networks and minimal fulfilling assignments for monotone Boolean functions. We establish a complexity result for minimal fulfilling assignments of monotone Boolean functions which is new to our knowledge: It is NP-complete to decide whether a variable is true in some inclusion-minimal truth assignment.

In Section 2.5 we briefly discuss structurally restricted AND/OR-networks. In AND/OR-networks forming an *in-tree*, the four notions of criticality are equivalent and it is easy to decide whether a job is critical. For *N-free* AND/OR-networks it turns out that a path-critical (and thus also cut-critical) job is bulk-critical (and delay-critical), but the other implication is not true in general. Considering the AND/OR-network as a hypergraph, we establish a slightly different notion of N-freeness. By giving appropriate counter-examples we show that if an AND/OR-network is N-free in the hypergraph notation, again neither of the implications

hold.

We conclude the chapter by a quantitative analysis of AND/OR-networks with respect to changes in processing times of critical sets. Cut- and delay-critical sets have a direct unbiased impact on the makespan. If the processing times of the jobs in a cut-critical set are decreased by some small amount, the makespan decreases by exactly the same amount. The equivalent holds for an increase in the processing times of the jobs in a delay-critical set. Another issue that belongs to the quantitative analysis of AND/OR-networks is the *time-cost trade-off* problem. We introduce the linear time-cost trade-off problem and discuss the properties of the time-cost trade-off curve on a significant example. The time-cost trade-off curve is piecewise linear and nonincreasing, but neither monotone nor continuous, like in the case of standard precedence constraints. In addition, the DEADLINE PROBLEM is NP-complete, which was proved by Stork (2001).

Chapter 3 discusses the problems of minimising the makespan and the total weighted completion time of a set of AND/OR precedence constrained jobs on one or several identical parallel machines. First we present Graham's well known *list scheduling* algorithm, which will be the major ingredient of the subsequent approaches to the stated problems. We consider the two objective functions in one section each, again divided into the one machine and the several machine case. In each section we review known results for the case of no and standard precedence constraints among the jobs and then present known and new results for AND/OR-precedence constraints.

The makespan objective is trivial to solve on one machine if the jobs are not restricted at all, but also if standard or AND/OR precedence constraints are imposed among the jobs. The problem on identical parallel machines is NP-complete even without precedence constraints and just two machines. Nevertheless, Graham's list scheduling provides a 2-approximation for scheduling precedence constrained jobs. Gillies and Liu (1995) present a 2-approximation for scheduling jobs on an acyclic AND/OR-network. This algorithm first transforms the AND/OR-network into a standard precedence graph and then applies list scheduling. We extend the algorithm of Gillies and Liu to general feasible AND/OR-networks.

Minimising the total weighted completion time on one machine is less trivial for precedence constrained jobs than the makespan. It is well known that the basic problem without any additional constraints can be solved optimally by *Smith's rule*. The total weighted completion time of standard precedence constrained jobs has been approximated with guarantee 2 with various techniques, which resist any application to AND/OR precedence constraints. For minimising the total completion time, that is the special case of unit weights, we prove that list scheduling with *shortest processing time rule* is a $\sqrt{n}$-approximation, where n is the number

of jobs. This bound is tight. The case with arbitrary weights seems to be even harder. By a reduction of LABEL COVER proposed by Goldwasser and Motwani (1999) together with an improved inapproximability result for LABEL COVER by Dinur and Safra (1999) we show that minimising the total weighted completion time to within a factor of $2^{\log^{1-1/\log\log^c n} n} n$ of the optimum is NP-hard for any $c < 1/2$. We prove that list scheduling with shortest processing time rule is a simple n-approximation for the problem and that this bound is again tight.

Chapter 4 considers a relaxation of the AND/OR precedence constraints, where it is allowed to exchange the direct successor of an OR-node against one of its direct predecessors. The idea is to use neighbourhood search to find an AND/OR-network that minimises the makespan. We first give a short introduction into local search and then propose a canonical neighbourhood on AND/OR-networks. The main result in this chapter is that the feasible networks are connected under the proposed neighbourhood. We briefly introduce the job shop problem, where local search has been applied quite successfully. The basic neighbourhoods used for the job shop problem resemble our canonical neighbourhood for AND/OR-networks. We close the chapter and the thesis by a short discussion about possible strategies of neighbourhood search in AND/OR-networks.

The thesis is essentially self-contained and we will introduce most of the notation needed. Nevertheless we expect the reader to be familiar with the basic concepts of graph theory, combinatorial optimisation, complexity theory, logic, and local search. For an introduction to graph theory we refer to Bollobás (1990), Jungnickel (1991), Diestel (2000), and West (2001). A thorough overview over combinatorial optimisation is presented in Schrijver (2003), but we also refer the reader to Korte and Vygen (2000), Nemhauser and Wolsey (1988), and Trotter (1992). The textbooks of Garey and Johnson (1979), Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela, and Protasi (1999), and Hochbaum (1997) cover our needs for complexity theory. For an introduction to logic and Boolean functions, we refer to Wegener (1987) and Prestel (1992). An introduction into and an overview of application of local search in combinatorial optimisation is presented in Aarts and Lenstra (1997).