Ősz Olivér

Methods for scheduling industrial processes

Módszerek ipari folyamatok ütemezésére

Doctoral dissertation

Doktori értekezés

Témavezető:

Dr. Hegyháti Máté

Egyetemi docens

Multidiszciplináris Műszaki Tudományi Doktori Iskola

Széchenyi István Egyetem

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scheduling is a research field under operations research that deals with optimization problems containing timing-related decisions. As time is an important aspect in our life, scheduling problems arise in lots of vastly different areas. Some examples are timetabling, CPU scheduling, supply-chain management, project planning, and production scheduling. My research is focused on the latter, the scheduling of industrial processes in manufacturing systems, with possible applications of the methods in logistics and project planning as well.

As in all optimization problems, the goal is to find the best solution among a lot of possible alternatives. In scheduling, the solution is a schedule, which contains decisions about the timing of events, and other related values, such as resource allocations, transportation paths, and material quantities. The quality of the solution is defined by the objective function, which assigns a numerical value to each solution. The goal is to find the solution with the lowest or largest such value, depending on whether it is a minimization or maximization problem. The most often occurring objective is to minimize the maximum completion time over a set of processes. Other examples for objectives are minimization of operation costs, investment costs, delay, or maximization of hourly profit.

An optimization problem is defined by the set of available decisions, the constraints they must satisfy, and the previously mentioned objective function. The problem definition consists of two parts: model and data. The types of decisions, their parameterized constraints, objective function, and the types of required parameters are described in the model, which defines a whole class of problems. For example, this determines whether the lengths of processes are constant, or dependent on other decisions such as machine allocation, or variable among bounds with associated cost parameters. For an instance of

this problem class, the problem data define the actual parameters, such as the number of processes, their operational parameters, resource requirements, etc.

The research problems in scheduling research are to create efficient solution algorithms for finding optimal, or near-optimal solutions for scheduling problems. As even fairly simple scheduling problems have NP-hard complexity, exact solution approaches guaranteeing optimality have exponentially increasing execution times in relation to problem sizes. For this reason, heuristic approaches are also actively researched, which are not guaranteed to find the globally optimal solution but can quickly obtain good solutions even for large problems. My research is concentrated on exact solution methods because as computational power is increasing and the algorithms are improving, more and more practical problems can be solved to optimality in reasonable execution times. Furthermore, exact and heuristic methods are often used in combination, so it is important to research improvements to both approaches. Also, most exact algorithms can be stopped to provide the best found solution even before proven optimality is reached, and an upper bound on the deviation from the optimal solution is known.

During my research, I investigated several problem classes and different solution techniques. Some of the problems come from theoretical classifications that apply to a wide range of scheduling problems, while others come from case studies of specific application areas. This work contains both types of problems.

Scheduling has a vast literature, and there are already numerous known problems and solution approaches, although there is still a lot of room for further research. My research has multiple motivations:

- Identify new problem classes in practice that have not been investigated in the literature, and propose solution methods for them.
- Extend the capabilities of existing solution methods, so they can be applied to a more general problem class.
- Improve the performance of existing solution approaches.

Chapter 2 gives an introduction to the existing solution approaches that I used and improved upon. Here, modeling techniques of Mixed-Integer Linear Programming (MILP) and the concepts of the S-graph methodology are presented.

The following chapters show the theses about the results of my research for different scheduling problems. Chapter 3 presents the problem of scheduling automated wet-etch stations used in semiconductor manufacturing, and MILP solution approaches. Chap-

ter 4 presents an extension of the S-graph framework for Resource-Constrained Project Scheduling Problems (RCPSP). In Chapter 5, a new type of scheduling problem in the steel-processing industry is defined and solved with an MILP model. Chapter 6 presents an S-graph based approach for considering water reuse during scheduling to minimize freshwater usage. Finally, Chapter 7 summarizes the conclusions of this research.

Chapter 2

Literature overview of modelling methods for scheduling

In this chapter, the related literature is reviewed, and the concepts which my research is based on, are introduced. First, problem characteristics that are frequently used to distinguish different scheduling problems are explained in Section 2.1. Then, various modeling methods used by MILP approaches are presented in Section 2.2. The methods of Constraint Programming are presented in Section 2.3. Then, Section 2.4 introduces the basics of the S-graph approach, which I used in several parts of my research.

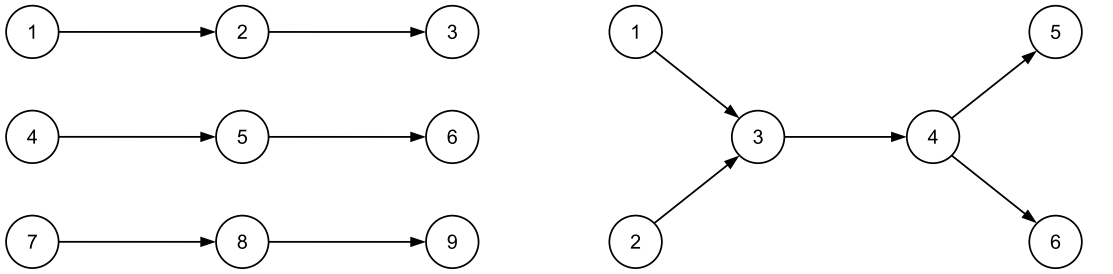
2.1 Classification of scheduling problems

In every scheduling problem, there are processes that take time, and usually require some resources to execute them. In production systems, there are continuous and batch processes. The difference between them is how their input/output is consumed/produced. Continuous processes consume and produce resources continuously during their execution, with either a fixed or variable rate. Batch processes consume all their input when they are started, and produce all their output when finished. These two types require different modeling techniques, and both have a vast literature. In this research, only batch processes are considered.

A process may have several subprocesses with different characteristics, and there are multiple names for them in the literature; so to avoid confusion, I will refer to atomic processes as tasks, and sets of related tasks as jobs. A typical job is to produce a given amount from a certain product, and the required production steps are the tasks of this

job.

In a production process, some steps must be carried out in a fixed order. These orderings can be represented by a directed acyclic graph, which is called a precedence graph. Its nodes are the tasks and its arcs are the fixed orders between them. Most jobs have a linear task sequence, with no branching in their precedence graph. However, a task may have multiple prerequisites or dependent tasks in some production systems. This happens most typically when the task requires materials produced by separate tasks, or produces materials that are used for multiple other tasks. Figure 2.1 shows examples for both types of jobs.



(a) Sequential jobs (b) Tasks with multiple inputs or outputs

Figure 2.1: Precedence graphs

Scheduling may include batching decisions, which determine the sizes (material quantities) and number of batches for each job. Sizes are constrained by capacities of the production equipment, and they can affect the processing times of batches. This highly complicates scheduling, so these decisions are usually made separately, before scheduling. In this research, fixed batch sizes and numbers are assumed in each problem.

Production equipment is an important resource in production planning. Single equipment units of a production system will be referred to as machines. A machine can only execute one task at a time, which imposes a constraint on the scheduling of tasks. It is also usually assumed that a task is fully executed on one machine, not split between multiple machines, and its execution cannot be interrupted (non-preemptive).

There can be multiple identical machines, or machines that differ in capabilities or at least in processing speed. Therefore, in the general case, the set of suitable machines and their corresponding processing times must be given for each task. However, some special cases of scheduling sequential jobs have been investigated in more detail in the literature. In a flow-shop scheduling problem, each job goes through the same machine sequence. A specific subclass of flow-shop problems, the so-called permutational flow-shop, also requires the that the job order is the same on each machine. Job-shop scheduling is a

more general problem, where jobs can have different machine sequences. Flexible variants of these problems allow multiple identical machines to be present at each stage. Reentrant variants allow jobs to visit the same machine (or set of machines) multiple times during their execution.

Apart from the actual production tasks, there can be several additional time-consuming operations constraining the schedule. Materials may need transportation between the machines, requiring time and equipment (see Chapter 3). Machines may require cleaning, maintenance or changeover operations between different tasks (see Chapter 5).

Besides the machines, tasks may require additional resources (see Chapter 4) for execution, for example, freshwater (see Chapter 6), human workers, or raw materials.

Between two consecutive tasks of a job, the intermediate material may need storage if the later task cannot start immediately. If storage is abundant, Unlimited Intermediate Storage (UIS) is assumed. However, in some production systems (especially in chemical plants), storing these materials is not trivial, and must be considered during scheduling. Materials may have Finite (FIS) dedicated, Common (CIS) shared, or No Intermediate Storage (NIS). Machines can store intermediates but cannot execute tasks in the meantime. Some intermediates may be unstable, and they cannot be stored indefinitely between tasks (see Chapter 3). Based on the limit on this waiting time, the constraint on an intermediate may be Zero-Wait (ZW), Limited Wait (LW), or Unlimited Wait (UW).

While mathematical methods can provide theoretically optimal solutions, many things can go wrong in practice. There are different ways to deal with this problem. Stochastic scheduling can utilize statistical data and probabilistic simulations to provide a good solution even in uncertain circumstances [31, 52]. Robust optimization can give solutions that can better withstand certain disruptions [14]. Machine learning techniques can help in robust planning and scheduling of production systems [70, 93]. Reactive scheduling can be used to recover from an unexpected situation or to adapt to changing demands [99]. In the problems investigated in the theses, I assumed deterministic behavior, although the developed methods can be used in a reactive or robust fashion as well, with minor modifications. I also collaborated on a project addressing uncertainty during scheduling [22].

2.2 Modeling scheduling problems with MILP

Mathematical programming is the most commonly used solution method for scheduling problems. The underlying algorithms for Linear Programming and Integer Programming are well established, efficient techniques. Leading commercial general-purpose solvers like Gurobi[1] or CPLEX[2] can quickly solve most complex MILP models to optimality. Therefore, scheduling research is concentrated on formulating these models to be efficiently solvable, rather than developing solution algorithms. However, there are some exceptions, as problem-specific algorithms can greatly increase solution performance, for example: defining custom branching strategies, cutting planes, facet-lifting procedures, decomposition techniques, and utilizing heuristic methods. From research perspective, using open-source solvers, such as from the COIN-OR project[3], also has merits, as the solution techniques and the aiding heuristics used in commercial solvers are not publicly available, which is against transparent research.

An extensive review of MILP modeling techniques for batch process scheduling was composed by Méndez et al. [67], and more recently by Harjunkoski et al. [36]. Here, a short introduction is presented to help in understanding the theses.

An MILP model is a set of linear equalities, inequalities, and a linear objective function to be minimized or maximized. They consist of continuous (real-valued) and integer variables, and constant parameters.

A crucial difference between MILP scheduling models is how they represent timing decisions. Precedence-based models [68] use binary sequencing variables for task pairs to decide their execution order in case they require the same machine. For example, a variable $x_{i,i'}$ can represent the order between task i and i', so that if $x_{i,i'} = 1$, then i must be finished before i' is started. Starting times of each task are represented by continuous variables, and constrained by these sequencing variables. An example is shown in Constraint (2.1), where t_i^s is the starting time of task i, d_i is its duration, and M is a sufficiently large number that the inequality always holds when $x_{i,i'} = 0$. This type of constraint is called a big-M constraint.

$$t_{i'}^s \geq t_i^s + d_i - M(1 - x_{i,i'}) \quad \forall i, i' \in I : i \neq i' \tag{2.1}$$

[1] https://www.gurobi.com
[2] https://www.ibm.com/analytics/cplex-optimizer
[3] https://www.coin-or.org

As precedence only needs to be set when the tasks are executed on the same machine, these variables must be connected to the assignment variables. For example, this can be achieved with binary variables of the form $y_{i,j}$, denoting whether task i is assigned to machine j. An example for connecting the two types of decision variables is shown in Constraint (2.2).

$$1 \geq x_{i,i'} + x_{i',i} \geq y_{i,j} + y_{i',j} - 1 \quad \forall i, i' \in I, j \in J : i \neq i' \tag{2.2}$$

Another way to represent time is to define time points and assign events to them to create the schedule. In this case, scheduling decisions are binary assignment variables indexed by time points, for example, $s_{i,t}$ represents whether task i is started at time point t.

The time points can be separators of a predefined interval partitioning of the considered time horizon [53]. This so-called discrete time point model creates an equidistant, fixed global time grid. While this simplifies modeling of resource availability, the number of required time points can be too large, which can lead to too many binary variables and high computational needs. Task durations must be rounded up to multiples of the interval length, so decreasing the number of time points leads to wasted time by moving some events to the next time point.

To decrease the number of time points, their position can be set by continuous variables. The time points can be either global for all events [90], or unit-specific [16], where each machine has a separate set of time points (also called time events in this case). Both approaches lead to fewer time points but since their synchronization is more complex, the number of time points still needs to be set to the minimum for good solution performance. Unfortunately, it is difficult to determine the minimum number of time points, so practice is to solve the model iteratively, with increasing number of time points, and stop when the objective value is not improving anymore, which does not guarantee optimality.

Another important aspect of the MILP models is the modeling of resource balances. In precedence-based formulations, the number of executions and batch sizes for each task must be predetermined. However, time point models are able to decide the numbers and sizes of tasks during optimization, based on the given material flows and demands. For this purpose, two network-based techniques became popular: STN and RTN.

In the STN (State-Task Network) representation proposed by Kondili et al. [53], tasks and machines are assigned to start and finish at certain time points, and material quan-

tities are set by continuous variables. Material availability at each state of processing, and at each time point, is represented by continuous variables, and updated by material balance constraints based on the assignments. This makes it simple to set both lower and upper limits for storage quantities.

The RTN (Resource-Task Network) representation by Pantelides [83] uses a similar approach, the key difference being that machines and materials are treated in the same way, as resources consumed and produced by tasks. This simplifies the model description, and inherently eliminates redundant schedules only differing in the assignments of identical machines. However, considering machines as resources makes it difficult to model certain unit-specific constraints, such as unit-specific transfer times, cleaning, and changeover times.

Both STN and RTN representation are still used in the literature for various scheduling problems [14, 92, 107]. To keep up with the recent trends in research, they have been extended for energy-efficient approaches too [48, 98].

When jobs are sequential and batch sizes are predetermined, material balance constraints are not necessary. In these models, instead of time points, the time intervals between them are in focus, which are often referred to as time slots [85].

There is no best method among these formulations, as each one has advantages and disadvantages when modeling problem-specific constraints. Therefore, each modeling technique is still widely used in scheduling.

2.3 Constraint Programming

Constraint Programming (CP) has only become widespread for scheduling problems in the last 20 years, so it is not as widely used as mathematical programming. CP is a technique for solving Constraint Satisfaction Problems, in which constraints are written as equations and logical statements, consisting of decision variables and parameters. A distinguishing feature of CP is the use of inference during search, called constraint propagation. This technique improves upon the classic backtracking search by using the available information from previous decisions to forbid values or combinations of values for some variables that would make it impossible to satisfy a subset of the constraints [10].

Constraint modeling for CP is similar to formulating mathematical programming models. CP can deal with nonlinear constraints more efficiently than mathematical program-

ming, so unlike in MILP models, even polynomial terms can be present, and more types of constraints are allowed, such as non-equality, logical implication, conjunctions, and disjunction. However, just as mathematical programming loses a great amount of its efficiency when nonlinearity is present, problems with continuous variables pose a difficult challenge for CP methods. While there are special techniques for these problems, CP is the most powerful when working with discrete variables. For scheduling problems, this means that CP methods mostly use discrete time models like global uniform time grids.

CP is a solution method for satisfaction problems but it can be used for optimization by iteratively solving the model while using the objective of the previous solution as a constraint to find a strictly better solution. When a solution is found, the search does not need to be restarted, it can be continued with the updated constraint on the objective value. When the problem becomes unsatisfiable, the last solution found (if any) is the optimal solution.

There are several CP solvers available. As CP is often used for parts of MILP solution techniques, commercial MILP solvers like Gurobi and CPLEX can also handle CP models. But there are solvers specifically for CP as well. One of the best of such solvers is an open-source project by Google, OR-Tools[4]. It has won the MiniZinc Challenge solver competition in most categories from 2018 to 2021[5].

2.4 S-graph methodology

The S-graph model and the associated solution procedure was introduced by Sanmartí et al. [88, 89], a joint work from Universitat Politècnica de Catalunya (Barcelona, Spain) and University of Pannonia (Veszprém, Hungary). I started studying at the latter in 2011, and the next year I joined the scheduling research group involved in the development of the S-graph solver framework. The motivation behind the S-graph approach was to solve batch process scheduling problems with an efficient, scheduling-specific algorithm, and to correctly model both UIS and NIS tasks, from which the latter is problematic in MILP models [24, 40]. Since its introduction, it was extended to different problem classes with success, achieving promising solution performances.

The two main pillars of the S-graph approach are a graph-based mathematical model to represent schedules, and a B&B (branch-and-bound) algorithm.

[4] https://opensource.google/projects/or-tools
[5] https://www.minizinc.org/challenge2021/results2021.html

2.4.1 Modeling schedules with S-graphs

Formally, an S-graph is a directed graph $G(N, A)$. There are two types of nodes: task nodes (N_t) representing the start times of tasks, and product nodes (N_p) representing the completion times of products ($N_t \cup N_p = N, N_t \cap N_p = \emptyset$). Arcs too have two classes ($A_1 \cup A_2 = A, A_1 \cap A_2 = \emptyset$): recipe-arcs ($A_1 \subset N_t \times N$) and schedule-arcs ($A_2 \subset N \times N_t$). A nonnegative weight is defined for each arc ($c(i, i')\ \forall (i, i') \in A$), representing the minimal required time difference between the events associated with the two nodes. This means that task i' cannot start (or product i' cannot be completed) earlier than $c(i, i')$ time units later than the start time of task i (or completion time of product i). A machine set is defined for each task node ($S_i\ \forall i \in N_t$) to represent the set of available machines for the task.

The solution procedure starts with a special S-graph, the recipe graph ($G(N, A_1)$), which is generated from the input parameters. A recipe-arc $(i, i') \in A_1$ is present if there is a mandatory precedence relation between tasks i and i', or if i is the final task of product i'. The recipe-graph contains no schedule-arcs ($A_2 = \emptyset$). Arc weights are the minimum possible processing times: $c(i, i') = \min_{j \in S_i}\{pt_{i,j}\}\ \forall (i, i') \in A_1$, where $pt_{i,j}$ denotes the processing time of task i on machine j.

As an illustrational example, the S-graph representation of a problem by Ferrer-Nadal et al. [24] is presented. Figure 2.2 shows the production paths and processing times (h) of the 4 products (A, B, C, D) among the 4 machines (U1, U2, U3, U4). The corresponding recipe-graph is shown in Figure 2.3. In this problem, each task has only one suitable machine ($|S_i| = 1\ \forall i \in N_t$), which simplifies the calculation of minimal processing times needed to determine the arc weights.

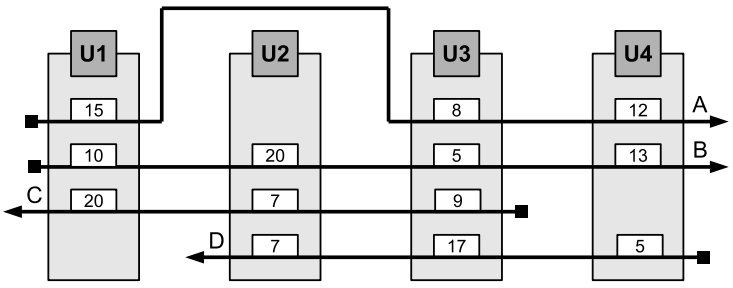


Figure 2.2: Process flow of the illustrative example

The time differences represented by arc weights are transitive, so directed paths also impose a timing constraint with the sum of the arc weights. The weights are lower bounds

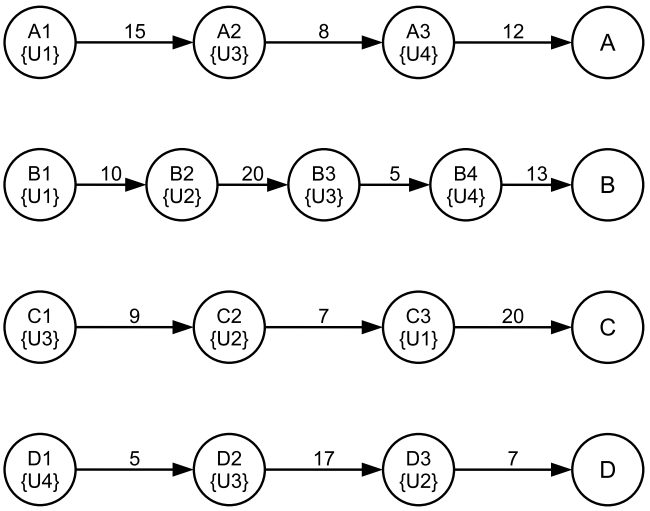


Figure 2.3: Recipe-graph of the illustrative example

on the time difference, so if there are multiple paths between two nodes, the one with the highest sum of weights is dominant. The longest path in the graph the one with the maximal such sum, which represents the makespan (total completion time) of the schedule.

The recipe-graph represents a partial schedule without any sequencing decisions, so its makespan does not account for machine availability but gives a lower bound on the possible makespan. In the example recipe-graph, the longest path is B1-B with a weight of 48 h. This is the theoretical limit: the optimal solution cannot be below 48 h, even if all products were processed in parallel, without any waiting between stages, as shown by the Gantt chart in Figure 2.4. However, some waiting is often inevitable when machines have different workloads.

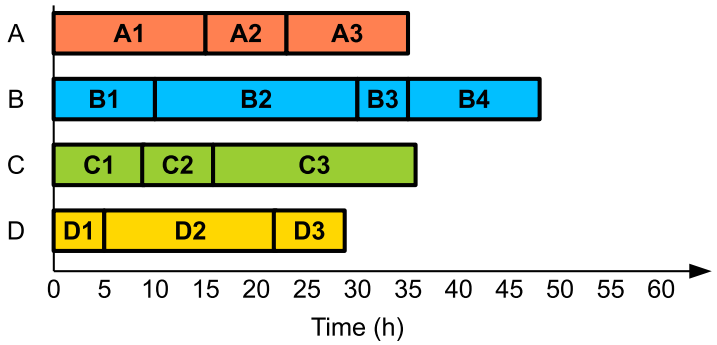


Figure 2.4: Gantt chart of the initial schedule

When multiple tasks are executed by the same machine, their order needs to be decided, as a machine can only execute one task at a time, and must finish it before starting

a new one. These sequencing decisions can be represented by adding schedule-arcs to the graph. There are two main ways to do this, based on the storage policy of the intermediate materials involved: UIS or NIS.

If UIS policy is used, a machine can start its next task anytime after the current task is finished. This is represented by adding a schedule-arc starting at the node of the earlier task and ending at the node of the later task, with a weight equal to the processing time of the earlier task. For example, if U1 has the task sequence A1-B1-C3, and U2 has C2-B2-D3, the resulting S-graph can be seen on Figure 2.5, with the schedule-arcs highlighted in blue. The corresponding Gantt chart is shown in Figure 2.6. The top part contains all tasks grouped by products, while the bottom part only contains the assigned tasks grouped by the machines they are assigned to. Red arrows indicate precedence relations that cause wait times before some of the tasks.

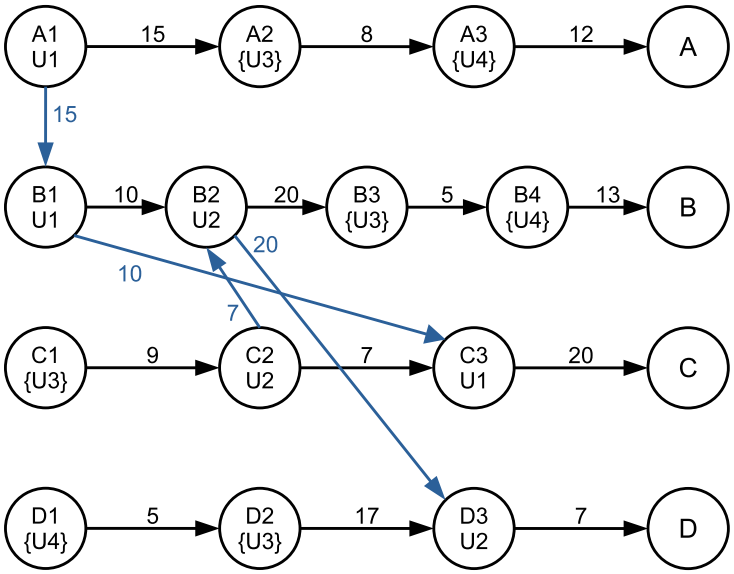


Figure 2.5: S-graph of a partial schedule with UIS schedule-arcs

If NIS policy is used, the machine acts as a storage until output materials are removed and transferred to the their next production step[6]. In this case, schedule-arcs start from the successor task(s) of the earlier task, and have 0 weights (these weights are often omitted in graphical representations for a clearer figure). If NIS is used in the previous example, the same sequencing decisions lead to an infeasible schedule. This is indicated by a directed cycle in the S-graph, shown in red in Figure 2.7. If the cycle had a positive

[6]Here, it is assumed that all input materials of a task are transferred to the machine at the same time, so its execution can start immediately, without the machine acting as storage before execution. To handle the more general case, a model transformation is necessary [45].

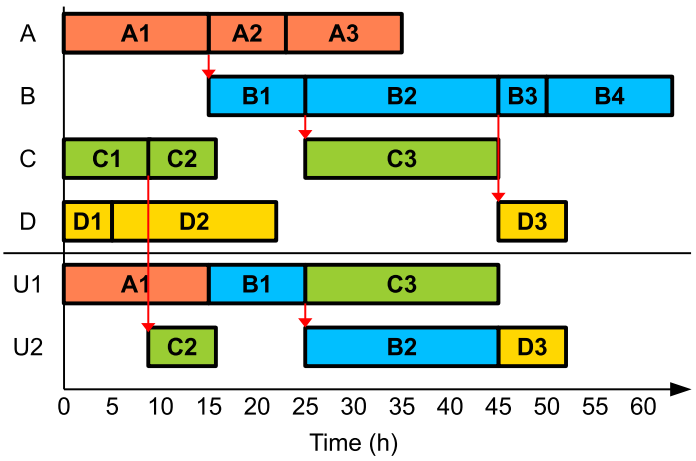


Figure 2.6: Gantt chart of the partial schedule

weight, it would indicate that a task should start later than its start time. The B2-C3-B2 cycle has 0 weight, which indicates a cross-transfer: the materials in 2 or more machines (U1 and U2 in this case) have to switch places with each other, which is not possible without intermediate storage.

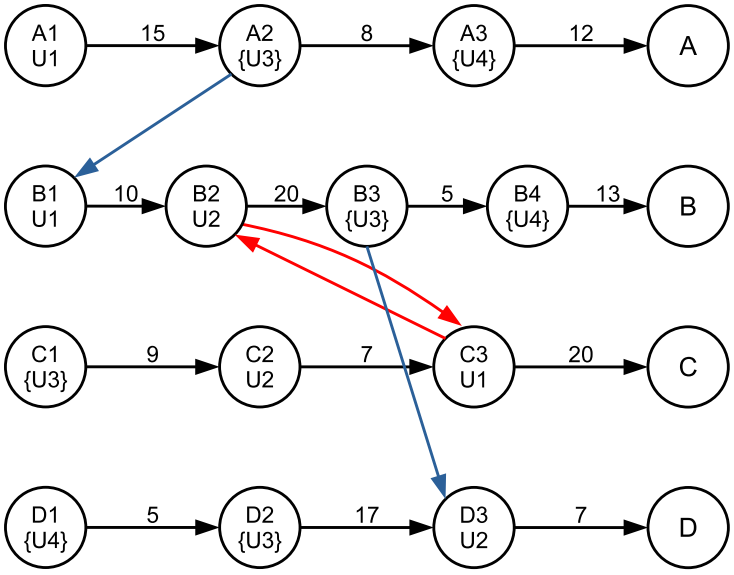


Figure 2.7: S-graph of a partial schedule with NIS schedule-arcs

Usually, all materials have the same storage policy but it is possible to model a mixed policy, where each task is classified as UIS or NIS ($N_{UIS} \cup N_{NIS} = N_t, N_{UIS} \cap N_{NIS} = \emptyset$), and their schedule-arcs are added accordingly.

2.4.2 Algorithms for finding the optimal schedule

The scheduling decisions are made by a B&B algorithm which enumerates the search space to find the optimal schedule. The high-level solution procedure is an ordinary B&B algorithm, as it is shown in Algorithm 2.1. The root node contains the recipe-graph and any variables needed by the branching procedure.

Algorithm 2.1 Structure of the B&B algorithm

```
Open := {root_node}
best := ∅
while Open ≠ ∅ do
  current := selectAndRemove(Open)
  if best = ∅ ∨ bound(current) < bound(best) then
    Children := branching(current)
    if Children = ∅ then
      best := current
    else
      Open := Open ∪ Children
    end if
  end if
end while
return best
```

The bound can be calculated thanks to the way how decisions are modeled by the S-graphs. Sequencing decisions introduce new arcs, and assignments may increase arc weights, as the actual processing time can be higher than the minimum that was used in the recipe-graph. By applying these operations, the weight of the longest path cannot decrease between any two nodes. Therefore, the longest path in any S-graph is a lower bound on the makespan of schedules reachable from it [88]. This value is used as the bound in the B&B procedure. For efficient calculation, the implementation does not perform a longest path search at each node, instead, a matrix stores the longest paths between each node pair, and it is updated when an arc or weight is modified. If there is a directed cycle in the graph, the bound function returns ∞ to signal infeasibility.

Scheduling decisions are made in the branching step of the B&B algorithm to partition the search space. This can be done in different ways. The algorithm published by Sanmartí et al. [88, 89] was later named as the equipment-based branching. It selects a machine, and for each unassigned task that the machine can perform, it creates a partition where this task is performed on this machine after the previously assigned tasks. The formal branching procedure is presented in the following.

The algorithm requires storing the following data for each node:

$G(N, A_1 \cup A_2)$ contains the nodes and arcs of the S-graph.

c defines the values of each arc weight, which can be stored in the graph adjacency matrix (the maximum weight is stored for duplicate arcs).

$S_i \; \forall i \in N_t$ are the sets of available machines.

SOUN is the set of tasks that are not scheduled yet, initially containing all tasks.

last_node is an array indexed by the set of machines (M), where last_node[j] is the last task scheduled to machine j, or $\varnothing$ if no task is assigned to the machine yet, which is the initial value of each element.

Child nodes are generated from the current node in the following way:

1. If every task is already scheduled (SOUN= $\emptyset$), the current node is a solution, no further branching is possible, return $\emptyset$.
2. Select a machine j, which has at least one task that can be assigned to it: $\{i \in \text{SOUN} \mid j \in S_i\} \neq \emptyset$.
3. For each task i that can be assigned to j, create a branch where i is scheduled as the next task on j. That is, create a child node ($G'(N, A_1 \cup A'_2)$, c', $\{S'_i \; \forall i \in N_t\}$, SOUN$'$, last_node$'$), such that:
   - Weights on the recipe-arcs starting from i are updated to the processing time on j: $c'(i, i') = pt_{i,j} \; \forall (i, i') \in A_1$.
   - Weights of other recipe-arcs remain unchanged: $c'(k, k') = c(k, k') \; \forall (k, k') \in A_1 : k \neq i$.
   - Schedule-arcs are added between $l :=$ last_node[j] and i:
     - If $l \in N_{UIS}$, $A'_2 = A_2 \cup \{(l, i)\}, c'(l, i) = \min_{j \in S_l}\{pt_{l,j}\}$.
     - If $l \in N_{NIS}$, $A'_2 = A_2 \cup \{(k, i) \; \forall (l, k) \in A_1\}, c'(k, i) = 0 \; \forall (l, k) \in A_1$.
   - If $i \in N_{UIS}$, weights of existing schedule-arcs starting from its node are updated to the processing time (just like with the recipe-arcs): $c'(i, i') = pt_{i,j} \; \forall (i, i') \in A_2$.
   - Weights of other schedule-arcs remain unchanged: $c'(k, k') = c(k, k') \; \forall (k, k') \in A_2$.
   - Other machines are removed from the available set of i: $S'_i = \{j\}$.
   - i becomes scheduled: SOUN$'$ = SOUN $\setminus \{i\}$.
   - i becomes the last task of j: last_node$'[j] := i$, last_node$'[k] =$ last_node$[k] \; \forall k \in M : k \neq j$.

4. If every task that can be assigned to j has at least one other available machine, also create a child node $(G(N, A_1 \cup A_2), c', \{S'_i \; \forall i \in N_t\}, \text{SOUN}, \text{last\_node})$ where no other tasks will be assigned to j:
   - Remove j from the sets of available machines: $S'_i = S_i \setminus \{j\} \; \forall i \in N_t$.
   - Update weights of the recipe-arcs: $c'(i, i') = \min_{k \in S'_i}\{pt_{i,k}\} \; \forall (i, i') \in A_1$.
   - Update weights of the schedule-arcs starting from UIS tasks: $c'(i, i') = \min_{k \in S'_i}\{pt_{i,k}\} \; \forall i \in N_{UIS}, (i, i') \in A_2$.
5. Return the set of child nodes generated in steps 3 and 4.

The optimal solution of the illustrative example has 59 h makespan if all tasks have UIS, and 87 h if all tasks have NIS. Figure 2.8 shows the schedule-graph (S-graph of a solution node) of the UIS case. The longest path is highlighted with thicker lines. The Gantt chart of the corresponding schedule is shown in Figure 2.9. Similarly, the schedule-graph of the NIS case is shown in Figure 2.10. And its Gantt chart is shown in Figure 2.11, where the lighter colored blocks illustrate the periods when a machine acts as intermediate storage.

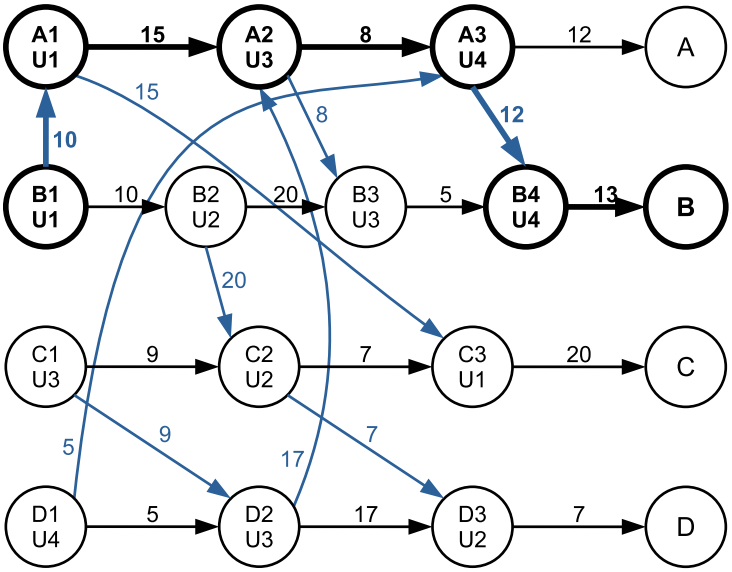


Figure 2.8: Schedule-graph with UIS tasks

The method explained above can solve makespan minimization problems. To solve other types of scheduling problems, modifications of the model and/or the algorithm are necessary. This can be regarded as a disadvantage to MILP models, where a general-purpose solver can be used for any model, however, it also offers opportunity to develop algorithms which can take advantage of problem-specific characteristics.

Several extensions and improvements have been proposed for the S-graph frame-

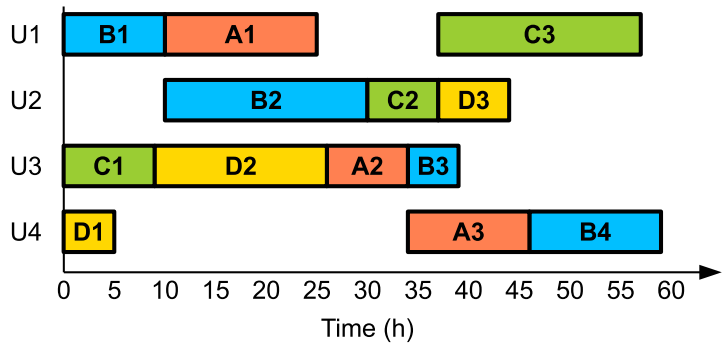


Figure 2.9: Gantt chart with UIS tasks

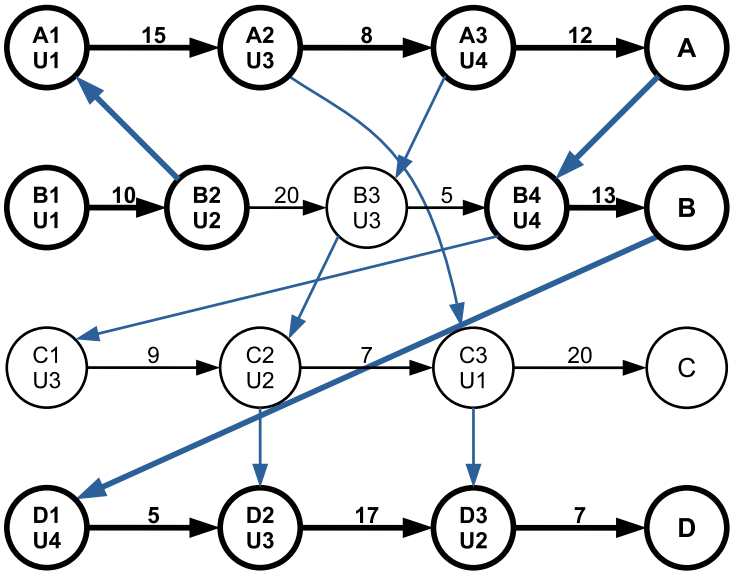


Figure 2.10: Schedule-graph with NIS tasks

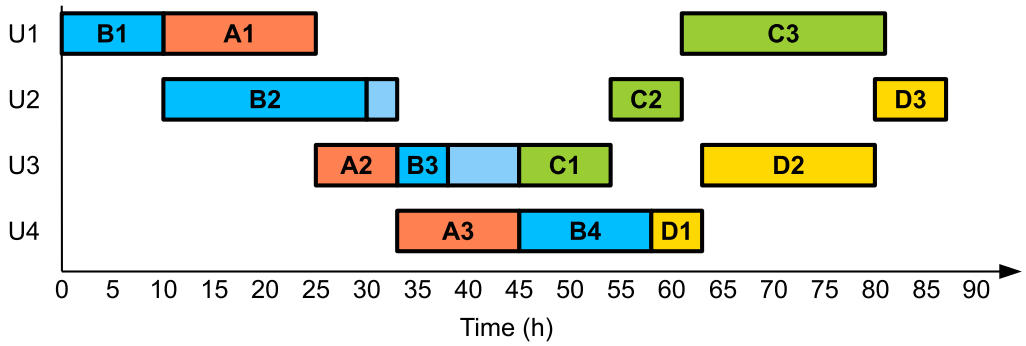


Figure 2.11: Gantt chart with NIS tasks

work. A throughput maximization method was developed [66, 46, 47], which utilizes the makespan minimization method as a feasibility checker. The S-graph method was adapted for scheduling problems arising in paint production [2], dairy manufacturing [1], and routing of railway networks [3]. The approach was also extended with handling of uncertainty [58] and limited waiting times [43]. My contributions to the S-graph framework are discussed in Chapters 4 and 6.

Chapter 3

Improved MILP models for scheduling wet-etch stations

In the age of Industry 4.0, automated manufacturing systems are used in more and more places. A well-studied scheduling problem is involved with such a system used in semiconductor manufacturing, the Automated Wet-etch Station (AWS).

The first, most important, and most complex stage of semiconductor manufacturing is the fabrication process [104]. Wafers made of silicon or gallium arsenide enter this process, and several layers of integrated circuits are created on its surface through metal deposition, photolithography, and etching. Wet-etching is a technique to chemically remove material from the wafer surface through a series of chemical and de-ionizing baths.

Etching operations require strict scheduling [84], as exposing the material to chemicals for too much or not enough time results in damaged products. This constraint is one reason why AWS scheduling a challenging problem. Another reason is the usage of transportation devices, robots, in the system, and their scarce availability. They are needed to ensure precise and consistent operation times, and to avoid contamination from human operators. Transport operations have to be considered when scheduling the processes of an AWS.

My research on this topic was motivated by the aim of the S-graph research group to solve this scheduling problem with an extension of the S-graph framework. While studying the literature MILP methods, I found possible improvements to these models, which I will present in this chapter.

An S-graph-based approach has also been developed as a joint work with Balázs Kovács, supervised by Máté Hegyháti and Ferenc Friedler. As it is not entirely my

own accomplishment, this approach is not detailed in this thesis. My improved MILP formulations and the S-graph approach were both presented in the paper by Hegyháti et al. [44].

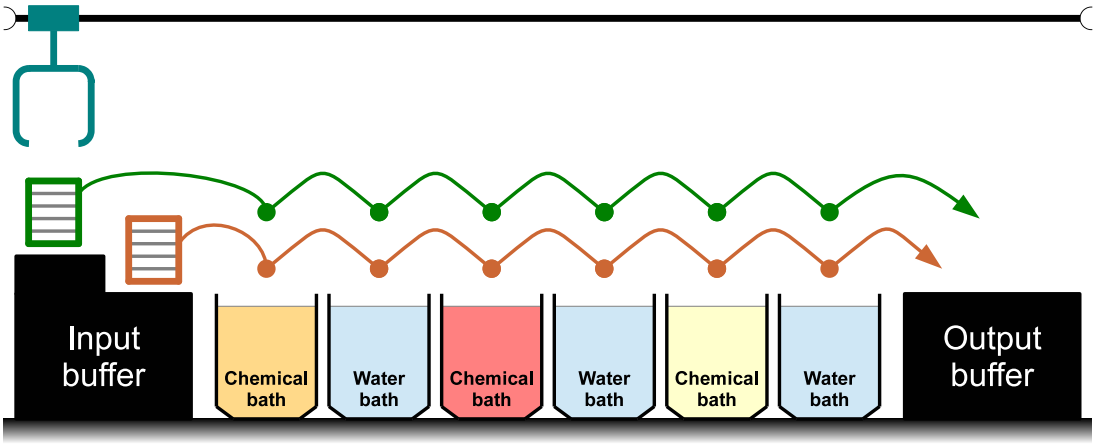
A formal definition of the problem is given in Section 3.1. A summary of the literature methods is presented in Section 3.2. My proposed improvements to these models are shown in Section 3.3. Empirical analysis of the proposed formulations is presented in Section 3.4.

3.1 Problem definition

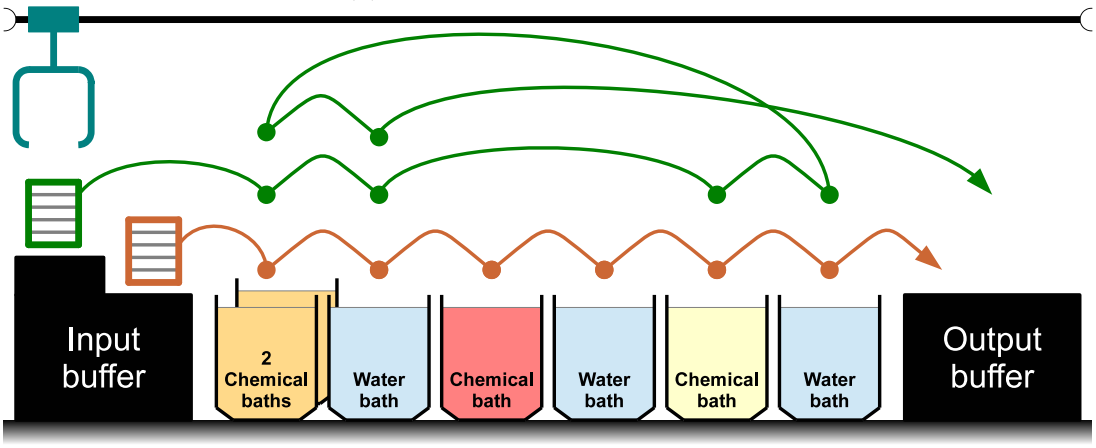
Wafers are processed as batches of several wafers held together, called as wafer lots. At the start, wafer lots are waiting in an input buffer for processing, and they are transferred to an output buffer at the end of their processing. During their etching process, wafer lots go through alternating stages of chemical and water (de-ionizing) baths. Chemical stages have a ZW policy, meaning that lots have to be transferred immediately from the chemical bath to a water bath after the required etching time has passed. Water stages have required treatment times, and a less restrictive waiting policy. In some literature methods, unlimited waiting times are allowed at these stages, while other approaches consider limited wait times. Wafer lots may differ in the required chemical and water treatment times, and waiting time limits.

In most literature methods, the stages are considered to operate in a flow-shop manner: there is on bath on each stage, and every wafer lot goes through the same bath sequence. With the combination of the lack of intermediate storage, this results in a permutation flow-shop problem, where the order of the wafer lots is the same on every stage. Other works in the literature consider more general problem classes, similar to reentrant flexible job-shops: multiple baths may be present on a stage, wafer lots can differ in the number and order of stages, and lots may visit the same stage multiple times. Figure 3.1 shows the differences between the two problem classes.

In addition to constraints on the processing and waiting times, and on the stage sequence, a robot must be available to transfer the wafer lot from one stage to the next. Scheduling the robot movements makes even the relatively simple permutation flow-shop variant a challenging problem. For this reason, early approaches only considered systems having a single robot. The generalization to multiple robots allows a more efficient oper-



(a) Permutation flow-shop AWS



(b) Reentrant flexible job-shop AWS

Figure 3.1: AWS recipe structures

ation of the AWS. In systems where the robots (also called hoists) travel on shared rails, they may interfere with each other during their movements. Taking this into account in scheduling would be a very difficult problem, so when multiple robots are present, either it is assumed that they each have their separate rail allowing independent movement from other robots, or non-overlapping sections (robot zones) of a shared rail are determined and assigned to separate robots.

Just as etching times, transfer times are also subject to limited wait policy. In single-robot systems, there is no advantage for performing a transfer slower than possible, however with multiple robots, if one robot brings a lot to a bath, and another robot is taking out a lot from this bath, the first robot may need to wait for the other. This can be modeled with flexible transfer times, which can vary in the given interval, opposed to using constant transfer times.

The objective is to minimize the makespan, which includes the transfer of the last wafer lot to the output buffer. A related problem called the Hoist Scheduling Problem [94] also

deals with the scheduling of transport equipment but with the objective of maximizing cyclic throughput. Here, only the makespan minimization problem is considered.

Scheduling an AWS is often done in an online fashion: the next set of lots must be scheduled so that the schedule of the currently processed set is adhered to. The most simple solution for this is to include tasks of the previous set of lots in the model, and fix variables regarding past events.

To summarize, the list of input parameters follows, using the notation of [5]:

I is the set of jobs (wafer lots).

S/S_i is the ordered set of stages, whose cardinality depends on job (i) in the flexible job-shop variant.

$J_{i,s}$ is the set of baths available for stage (i,s) in the flexible job-shop variant.

R is the set of robots.

$R_{i,s}$ is the set of robots that can transfer to stage (i,s), if robot zones are used.

$t_{i,s}^{min}$ is the minimum required processing time of stage (i,s).

$t_{i,s}^{max}$ is the maximum allowed processing time of stage (i,s).

$\pi_{i,s}^{min}$ is the minimum possible transfer time to stage (i,s) from the previous stage of the job.

$\pi_{i,s}^{max}$ is the maximum allowed transfer time to stage (i,s) from the previous stage of the job.

$\pi_{j,j'}^{abs}$ is the travel time between baths j and j'.

H_r is the starting position of robot r.

Figure 3.2 illustrates the input data of a reentrant flexible job-shop problem by Aguirre et al. [5]. There are 3 types of products with 2 batches from each. Their paths through the stages are shown with minimum and maximum processing times $(t_{i,s}^{min}, t_{i,s}^{max})$ above the stages, and loaded transfer times $(\pi_{i,s}^{min}, \pi_{i,s}^{max})$ between the stages. Unloaded travel times $(\pi_{j,j'}^{abs})$ between consecutive baths are shown at the top (for non-consecutive baths, the distance is the sum of the distances in-between). All times are given in minutes. There are 2 robots, $r1$ has the zone $j0 - j7$, and $r2$ covers $j7 - j37$.

The optimal solution of the problem is shown by the Gantt chart in Figure 3.3. The makespan value is 160.05 minutes.

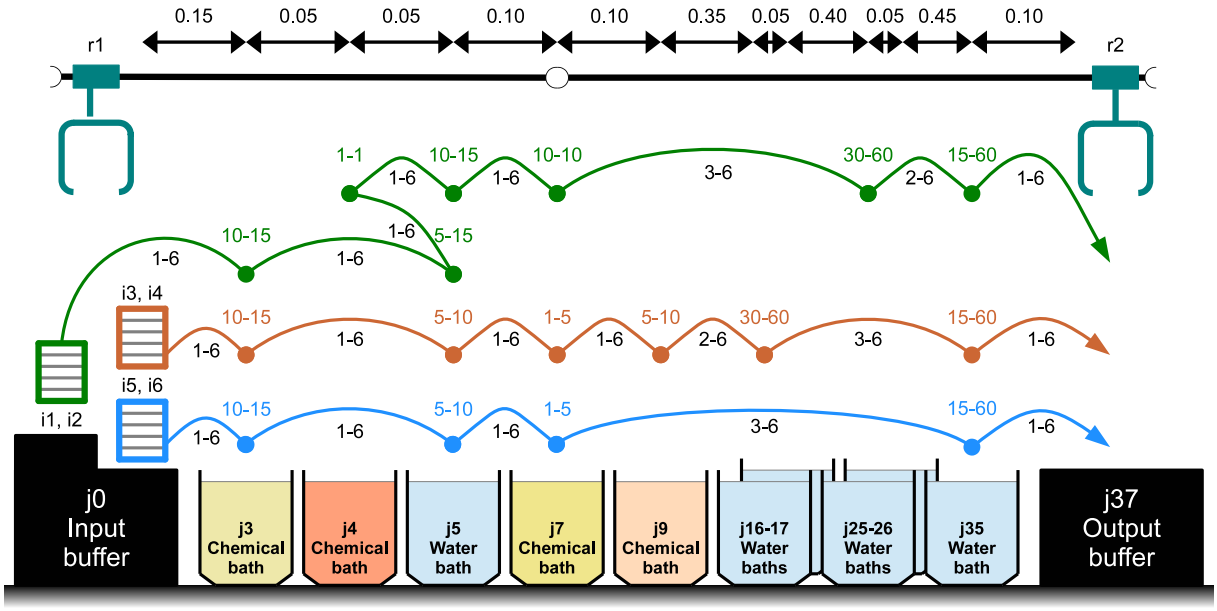


Figure 3.2: Problem data of the case study by Aguirre et al. [5]

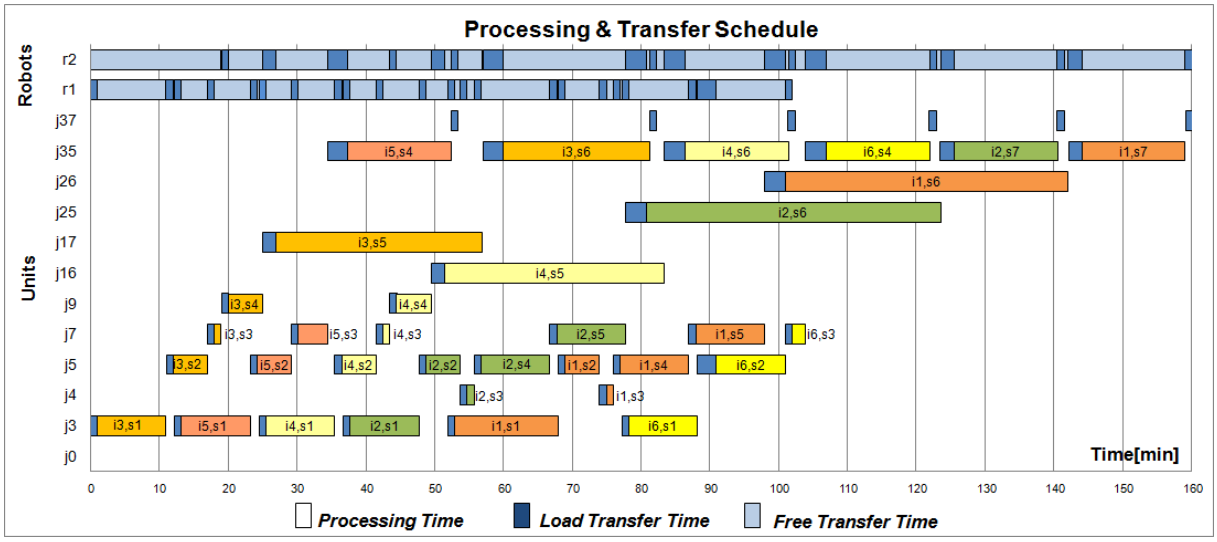


Figure 3.3: Optimal schedule of the case study by Aguirre et al. [5]

3.2 Literature approaches

The first solution approach was A Tabu Search based heuristic by Geiger et al. [34]. It only considered a single robot and the permutation flow-shop structure. Later, Bhushan and Karimi [12] proposed better heuristic algorithms for this problem class, using Simulated Annealing.

The first exact approach was a time slot MILP model by Bhushan and Karimi [11]. The authors also proposed a two-step heuristic, RCURM (Robot-Constrained Unlimited Robot Model), where a model with unlimited robots is solved first, then the job permutation is fixed and robot movements are scheduled.

A hybrid model was proposed by Castro et al. [15] which combined slot-based and precedence-based techniques to achieve better performance. The model and its extension is presented in Section 3.3.2.

Zeballos et al. [106] presented a solution approach based on Constraint Programming and a search strategy tailored for the investigated problem.

The previously mentioned approaches did not account for the empty movements of the robots. They assumed that only transfer movements take time and neglected the time necessary for the robot to go from one transfer to another. Novas et al. [72] pointed out this problem and proposed a Constraint Programming approach which correctly models empty robot movements. However, their model is only suitable for scheduling a single robot.

A precedence-based MILP model was presented by Aguirre et al. [5] which addresses the empty robot movement problem of earlier MILP models and can also solve the more general job-shop AWS scheduling problem. The model and its improvement is presented in Section 3.3.1.

An S-graph-based solution approach was proposed [44] for the flexible job-shop variant. Transfer times were modeled as changeover times of the robots between their transfer tasks. Product changeover times of machines had been modeled with S-graphs earlier but in this case, sequence-dependent changeover was required. This needed an extension to the branching algorithm to update the arc weights of schedule-arcs based on the baths assigned to the tasks before and after a transfer. Modeling limited wait times were achieved with negative-weighted arcs which was also a novel extension to the S-graph framework. As limited waiting constraints are present in many practical scheduling problems, alternative modeling techniques for it were also investigated [42] but using negative arcs proved to

be the best approach.

3.3 Proposed model improvements

In this section, two literature models and my proposed improvements for them are presented in detail. The first model is the precedence-based approach by Aguirre et al. [5], for which I proposed a more efficient formulation, as presented in Section 3.3.1. The second model is a hybrid method by Castro et al. [15], which I extended to a more general problem class, shown in Section 3.3.2.

3.3.1 Improving the model by Aguirre et al. (2013)

First, the original model [5] is presented, then my proposed modifications follow. The model considers the most general, reentrant flexible job-shop problem class.

The tasks are indexed by (i, s) $\forall i \in I, s \in S_i$ lot-stage pairs. The first ($s = 1$) and last ($s = |S_i|$) stages are the input and output buffers for each lot, with 0 processing time.

The model contains the following variables:

$Ts_{i,s}, Tf_{i,s} \geq 0$ are the start and finish times of tasks

$t_{i,s}$ is the processing time of a task

$\pi^{load}_{i,s}$ is the transfer time into a task

$\pi^{free}_{i,s}$ is the empty movement time before transferring into a task

$\pi^{seq-dep}_{i,i',s,s'}$ is the sequence-dependent transfer time from (i', s') to (i, s)

$X_{i,i',s,s'}, Y_{i,i',s,s'}$ are the binary precedence variables for tasks and transfers

$K_{i,i',s,s',r}$ is the binary immediate precedence variable for transfers

$w_{i,s,j}, q_{i,s,r}$ are the binary task-bath and transfer-robot assignment variables

$Pos_{i,s,r}$ is variable denoting the position of a transfer in the transfer sequence of a robot

MK is the makespan

$X_{i,i',s,s'}, Y_{i,i',s,s'}$ are so-called general precedence variables, representing the order of two tasks. The immediate precedence variable $K_{i,i',s,s',r}$ represents a stricter relationship, that the two tasks are executed in succession by the same robot.

The objective is to minimize the makespan:

$$\text{minimize} MK \tag{A:1}$$

$$MK \geq Ts_{i,|S_i|} \quad \forall i \in I \tag{A:2}$$

Every task is assigned to a single bath, and every transfer to a single robot:

$$\sum_{j \in J_{i,s}} w_{i,s,j} = 1 \quad \forall i \in I, s \in S_i \tag{A:3}$$

$$\sum_{r \in R_{i,s}} q_{i,s,r} = 1 \quad \forall i \in I, s \in S_i \tag{A:4}$$

The difference between finish and start times of a task is set by the processing time, which is bounded by the given parameters:

$$Tf_{i,s} = Ts_{i,s} + t_{i,s} \quad \forall i \in I, s \in S_i \tag{A:5}$$

$$t_{i,s}^{min} \leq t_{i,s} \leq t_{i,s}^{max} \quad \forall i \in I, s \in S_i \tag{A:6}$$

The transfer times between subsequent stages are set by the $\pi_{i,s}^{load}$ variables:

$$Ts_{i,s} = Tf_{i,s-1} + \pi_{i,s}^{load} \quad \forall i \in I, s \in S_i : s > 1 \tag{A:7}$$

$$\pi_{i,s}^{min} \geq \pi_{i,s}^{load} \geq \pi_{i,s}^{max} \quad \forall i \in I, s \in S_i \tag{A:8}$$

Task sequencing is done by the $X_{i,i',s,s'}$ binary precedence variables:

$$X_{i,i',s,s'} = \begin{cases} 1 & \text{if task } (i,s) \text{ is processed after } (i',s') \text{ in the same bath} \\ 0 & \text{otherwise} \end{cases} \tag{A:9}$$

$$\forall i, i' \in I, s \in S_i, s' \in S_{i'} : i > i'$$

$$Ts_{i,s} \geq Tf_{i',s'} + \pi_{i,s}^{load} + \pi_{i',s'+1}^{load} - M \cdot (3 - X_{i,i',s,s'} - w_{i,s,j} - w_{i',s',j}) \tag{A:10}$$

$$\forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s} \cap J_{i',s'} : i > i', s' \neq |S_{i'}|$$

$$Ts_{i',s'} \geq Tf_{i,s} + \pi_{i',s'}^{load} + \pi_{i,s+1}^{load} - M \cdot (2 + X_{i,i',s,s'} - w_{i,s,j} - w_{i',s',j}) \tag{A:11}$$

$$\forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s} \cap J_{i',s'} : i > i', s \neq |S_i|$$

Note that in Constraints (A:10) and (A:11), the transfer times of removing the material from the bath and the delivery of new material are added together when calculating the time difference between the 2 tasks. However, if there are multiple robots, these 2 transfers may be assigned to different robots, which can execute them in parallel. So as Castro et al. [15] proposed based on an earlier precedence based model [4], for the multiple robot model, Constraints (A:10) and (A:11) should be replaced by Constraints (Am:12)-(Am:15).

$$Ts_{i,s} \geq Tf_{i',s'} - M \cdot (3 - X_{i,i',s,s'} - w_{i,s,j} - w_{i',s',j}) \quad \forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s} \cap J_{i',s'} : i > i', s' \neq |S_{i'}| \tag{Am:12}$$

$$Ts_{i',s'} \geq Tf_{i,s} - M \cdot (2 + X_{i,i',s,s'} - w_{i,s,j} - w_{i',s',j}) \quad \forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s} \cap J_{i',s'} : i > i', s \neq |S_i| \tag{Am:13}$$

$$Ts_{i,s} \geq Tf_{i',s'} + \pi^{load}_{i,s} + \pi^{load}_{i',s'+1} - M \cdot (3 - X_{i,i',s,s'} - w_{i,s,j} - w_{i',s',j}) - M \cdot (2 - q_{i,s,r} - q_{i',s'+1,r}) \quad \forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s} \cap J_{i',s'}, r \in R_{i,s} \cap R_{i',s'+1} : i > i', s' \neq |S_{i'}| \tag{Am:14}$$

$$Ts_{i',s'} \geq Tf_{i,s} + \pi^{load}_{i',s'} + \pi^{load}_{i,s+1} - M \cdot (2 + X_{i,i',s,s'} - w_{i,s,j} - w_{i',s',j}) - M \cdot (2 - q_{i,s+1,r} - q_{i',s',r}) \quad \forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s} \cap J_{i',s'}, r \in R_{i,s+1} \cap R_{i',s'} : i > i', s \neq |S_i| \tag{Am:15}$$

To schedule the movements of the robots, a sequencing of their transfers needs to be determined. A robot cannot execute more than one transfer at a time, and needs to travel from the end of a transfer to the start of the next transfer. The latter time is represented by the $\pi^{free}_{i,s}$ variable, and the sequencing of transfers is set by the $Y_{i,i',s,s'}$ binary precedence variables.

$$Y_{i,i',s,s'} = \begin{cases} 1 & \text{if } (i,s) \text{ is transferred after } (i',s') \text{ by the same robot} \\ 0 & \text{otherwise} \end{cases} \tag{A:16}$$

$$Ts_{i,s} \geq Ts_{i',s'} + \pi^{load}_{i,s} + \pi^{free}_{i,s} - M \cdot (3 - Y_{i,i',s,s'} - q_{i,s,r} - q_{i',s',r}) \quad \forall i, i' \in I, s \in S_i, s' \in S_{i'}, r \in R_{i,s} \cap R_{i',s'} : i \geq i', (i,s) \neq (i',s') \tag{A:17}$$

$$Ts_{i',s'} \geq Ts_{i,s} + \pi^{load}_{i',s'} + \pi^{free}_{i',s'} - M \cdot (2 + Y_{i,i',s,s'} - q_{i,s,r} - q_{i',s',r}) \quad \forall i, i' \in I, s \in S_i, s' \in S_{i'}, r \in R_{i,s} \cap R_{i',s'} : i \geq i', (i,s) \neq (i',s') \tag{A:18}$$

In [5], the domain of Constraints (A:17) and (A:18) is constricted to $i > i'$, which is sufficient, as different stages of a lot are already sequenced by the stage order. However, broadening the domain of these constraints to the domain of $Y_{i,i',s,s'}$ leads to a more efficient formulation. Without this change, the model performed significantly worse in the tests than the performance reported in [5]. As later constraints in [5] have the condition $i \geq i', (i,s) \neq (i',s')$, I assumed the above constraints should have the same, and were

just written mistakenly in the paper.

To set the value of $\pi_{i,s}^{free}$ exactly to the time that the robot requires for traveling from the end of its previous transfer to the start of transfer (i,s), the authors converted the general precedence relations into immediate precedence variables. This is done with the help of $Pos_{i,s,r}$ variables, which represent the position of the transfer i,s among all the transfers done by robot r. Their values are set by Constraints (A:19)-(A:22). If (i,s) is not transferred by r, the value of $Pos_{i,s,r}$ is 0.

$$\begin{aligned} Pos_{i,s,r} \geq Pos_{i',s',r} + 1 - M \cdot (3 - Y_{i,i',s,s'} - q_{i,s,r} - q_{i',s',r}) \\ \forall i,i' \in I, s \in S_i, s' \in S_{i'}, r \in R_{i,s} \cap R_{i',s'} : i \geq i', (i,s) \neq (i',s') \end{aligned} \tag{A:19}$$

$$\begin{aligned} Pos_{i',s',r} \geq Pos_{i,s,r} + 1 - M \cdot (2 + Y_{i,i',s,s'} - q_{i,s,r} - q_{i',s',r}) \\ \forall i,i' \in I, s \in S_i, s' \in S_{i'}, r \in R_{i,s} \cap R_{i',s'} : i \geq i', (i,s) \neq (i',s') \end{aligned} \tag{A:20}$$

$$Pos_{i,s,r} \leq \sum_{i' \in I} \sum_{\substack{s' \in S_{i'}: \\ r \in R_{i',s'}}} q_{i',s',r} \quad \forall i \in I, s \in S_i, r \in R_{i,s} \tag{A:21}$$

$$q_{i,s,r} \leq Pos_{i,s,r} \leq M \cdot q_{i,s,r} \quad \forall i \in I, s \in S_i, r \in R_{i,s} \tag{A:22}$$

Then, the auxiliary variable, $K_{i,i',s,s',r}$ is set to 0 if (i,s) is the next transfer of r after (i',s'):

$$\begin{aligned} K_{i,i',s,s',r} = Pos_{i,s,r} - Pos_{i',s',r} - 1 + M \cdot (3 - Y_{i,i',s,s'} - q_{i,s,r} - q_{i',s',r}) \\ \forall i,i' \in I, s \in S_i, s' \in S_{i'}, r \in R_{i,s} \cap R_{i',s'} : i \geq i', (i,s) \neq (i',s') \end{aligned} \tag{A:23}$$

$$\begin{aligned} K_{i',i,s',s,r} = Pos_{i',s',r} - Pos_{i,s,r} - 1 + M \cdot (2 + Y_{i,i',s,s'} - q_{i,s,r} - q_{i',s',r}) \\ \forall i,i' \in I, s \in S_i, s' \in S_{i'}, r \in R_{i,s} \cap R_{i',s'} : i \geq i', (i,s) \neq (i',s') \end{aligned} \tag{A:24}$$

The sequence-dependent unloaded travel time ($\pi_{i,i',s,s'}^{seq-dep}$) is set by Constraint (A:25), based on the baths assigned to the respective tasks. In [5] this constraint is not defined for $s = 1$, the case where the robot moves to the input buffer. My tests showed that this would lead to infeasible solutions by allowing the robot to travel to the input buffer in an instant. Without setting the distance as a lower bound for $\pi_{i,i',1,s'}^{seq-dep}$, it becomes 0, and so does $\pi_{i,1}^{free}$, the unloaded travel time before $(i,1)$. While stage $s = 1$ does not have an actual transfer from an earlier stage (the transfer time is 0), the robot must travel back to the input buffer. As the order of transfers is (i',s'), $(i,1)$, $(i,2)$, there is no immediate precedence between (i',s') and $(i,2)$, so the distance is not set there either. I assume this was an editorial mistake in the article [5], as the reported solutions are correct, so the

model must have included a constraint for the $s = 1$ case as well. As $s - 1$ would be out of bounds in this case, I extended the Constraint (A:25) to $s = 1$ in Constraint (A:26) by using $w_{i,1,j}$ in place of $w_{i,s-1,j}$, as there is no previous stage for the first stage.

$$\begin{aligned} \pi^{seq-dep}_{i,i',s,s'} &\geq \pi^{abs}_{j',j} - M \cdot (2 - w_{i,s-1,j} - w_{i',s',j'}) \\ \forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s}, j' \in J_{i',s'} &: s > 1, (i,s) \neq (i',s') \end{aligned} \tag{A:25}$$

$$\begin{aligned} \pi^{seq-dep}_{i,i',1,s'} &\geq \pi^{abs}_{j',j} - M \cdot (2 - w_{i,1,j} - w_{i',s',j'}) \\ \forall i, i' \in I, s' \in S_{i'}, j \in J_{i,1}, j' \in J_{i',s'} &: (i,1) \neq (i',s') \end{aligned} \tag{A:26}$$

In Constraint (A:27), the value of $\pi^{seq-dep}_{i,i',s,s'}$ is used as a lower bound for $\pi^{free}_{i,s}$ when (i', s') is the previous transfer (when $K_{i,i',s,s',r} = 0$).

$$\begin{aligned} \pi^{free}_{i,s} &\geq \pi^{seq-dep}_{i,i',s,s'} - M \cdot K_{i,i',s,s',r} - M \cdot (2 - q_{i,s,r} - q_{i',s',r}) \\ \forall i, i' \in I, s \in S_i, s' \in S_{i'}, r \in R_{i,s} \cap R_{i',s'} &: (i,s) \neq (i',s') \end{aligned} \tag{A:27}$$

And finally, to handle the starting position (H_r) of each robot, Constraint (A:28) is used to set the unloaded travel time before the first transfer of a robot:

$$\begin{aligned} \pi^{free}_{i,s} &\geq \pi^{abs}_{H_r,j} - M \cdot (Pos_{i,s,r} - 1) - M \cdot (2 - q_{i,s,r} - w_{i,s-1,j}) \\ \forall i \in I, s \in S_i, j \in J_{i,s-1}, r \in R_{i,s} &: s > 1 \end{aligned} \tag{A:28}$$

However, like with the Constraints (A:10)-(A:11), there is also a problem with Constraint (A:28) if multiple robots are allowed for the same transfer, i.e., robot zones are not disjoint. In this case, $Pos_{i,s,r}$ and $q_{i,s,r}$ both equal to 0 for the unassigned robot, activating the constraint for not just the first transfers of the robots but for other transfers as well. There is no simple solution for this, as the constraint should be activated when $Pos_{i,s,r} = 1$ but not when it is lower or greater than 1. It needs additional binary variables indicating whether a transfer is the first transfer of a robot. My proposed formulation resolves this issue too.

Improved formulation

The idea behind my improved model is the following: if the transfer to (i, s) is preceded by the transfer to (i', s'), and both are assigned to the same robot, their time difference must be at least the unloaded travel time between the baths assigned to (i', s') and $(i, s - 1)$. Even if the robot carries out other transfers between these two, that cannot decrease the travel time. This statement holds assuming that travel times between baths

satisfy the triangle inequality, and loaded transfer times are not lower than unloaded travel times, which are reasonable assumptions in these manufacturing systems. The motivating example given in [5] more than satisfies these assumptions: the unloaded robot travel speeds are much higher than loaded speeds.

In the improved model, Constraints (A:17)-(A:28) are replaced with (A*:29)-(A*:31). Also, all $\pi^{free}_{i,s}$, $Pos_{i,s,r}$, $K_{i,i',s,s',r}$, and $\pi^{seq-dep}_{i,i',s,s'}$ variables are removed, and the domain of $Y_{i,i',s,s'}$ is reduced to $\forall i, i' \in I, s \in S_i, s' \in S_{i'} : i > i'$.

$$\begin{aligned} Ts_{i,s} \geq\ & Ts_{i',s'} + \pi^{load}_{i,s} + \pi^{abs}_{j',j} - M \cdot (1 - Y_{i,i',s,s'}) \\ & - M \cdot (2 - w_{i,s-1,j} - w_{i',s',j'}) - M \cdot (2 - q_{i,s,r} - q_{i',s',r}) \end{aligned} \tag{A*:29}$$

$$\forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s-1}, j' \in J_{i',s'}, r \in R_{i,s} \cap R_{i',s'} : i > i', s > 1$$

$$\begin{aligned} Ts_{i',s'} \geq\ & Ts_{i,s} + \pi^{load}_{i',s'} + \pi^{abs}_{j,j'} - M \cdot Y_{i,i',s,s'} \\ & - M \cdot (2 - w_{i,s,j} - w_{i',s'-1,j'}) - M \cdot (2 - q_{i,s,r} - q_{i',s',r}) \end{aligned} \tag{A*:30}$$

$$\forall i, i' \in I, s \in S_i, s' \in S_{i'}, j \in J_{i,s}, j' \in J_{i',s'-1}, r \in R_{i,s} \cap R_{i',s'} : i > i', s' > 1$$

$$\begin{aligned} Ts_{i,s} \geq\ & \pi^{load}_{i,s} + \pi^{abs}_{H_r,j} - M \cdot (2 - q_{i,s,r} - w_{i,s-1,j}) \\ & \forall i \in I, s \in S_i, j \in J_{i,s-1}, r \in R_{i,s} : s > 1 \end{aligned} \tag{A*:31}$$

4 model variants have been presented above. Table 3.1 shows which constraints make up which models. The original model presented by [5] will be referred to as model A. The version which allows parallel input-output transfers in multi-robot systems, as suggested in [15], will be called Am. My improved variants of these models will be denoted by A* and Am*.

Table 3.1: Constraints contained in the 4 models

Constraints	A	Am	A*	Am*
(A:1)-(A:8)	X	X	X	X
(A:10)-(A:11)	X		X	
(Am:12)-(Am:15)		X		X
(A:17)-(A:28)	X	X		
(A*:29)-(A*:31)			X	X

3.3.2 Extending the model by Castro et al. (2012)

This model [15] addresses the less general permutation flow-shop variant but takes advantage of this to achieve a better performance. This motivated me to extend this model

to consider empty robot movements, robot zones, limited wait times, and flexible transfer times that were introduced in [5]. Here, the original model (with some notational changes) and my extended model is presented.

In the considered problem class, there is only a single bath at each stage, so $S = J$[1]. In the original model, the set of stages (S) is partitioned into chemical ($ZW = 1, 3, \ldots, |S| - 2$) and water ($LS = 2, 4, \ldots, |S| - 1$) stages, and the output buffer, which is the last stage ($|S|$) with 0 processing time. The former has no wait time ($t_{i,s}^{min} = t_{i,s}^{max}$), and the latter has unlimited wait time ($t_{i,s}^{max} = \infty$), so $p_{i,s}$ is used to denote the processing time on all stages. Constant, lot-independent transfer times are assumed: $\pi_{i,s}^{min} = \pi_{i',s}^{max}\ \forall i, i' \in I$, so they are denoted by π_s.

The permutation flow-shop variant is considered, where every wafer lot goes through the same bath sequence. With no multiple identical baths, no intermediate storage, and no reentrant flow, the order of the wafer lots will be the same at every stage. This order is represented using time slots. The set of time slots, T, has the same size as the set of lots ($|T| = |I|$). A pairing of lots and time slots are set by binary variables $N_{i,t} \in \{0, 1\}\ \forall i \in I, t \in T$. If $N_{i,t} = 1$, lot i will be the t-th lot in the processing order.

The start time of the t-th lot at stage s is set by non-negative continuous variables $T_{t,s}$, and if it is a water stage ($s \in LS$), $Te_{t,s} \geq 0$ sets the end time of its processing. Another non-negative continuous variable, MS, represents the makespan, which is minimized:

$$\text{minimize} MS \tag{C:1}$$

This technique is called the unit-specific time slot formulation, where time slots have a different start time in each equipment unit. Normally, this would require task-slot assignments, not job-slot assignments, but because the job order is the same at each stage, the assignment variable does not need a stage index.

There are separate constraints for the timing of chemical and water stages. For chemical stages, Constraint (C:2) ensures that the required processing time is passed before the next lot is started to be processed. And Constraint (C:3) ensures that the lot is immediately transferred to its next stage after it was processed. Constraint (C:4) sets the

[1] As stages and machines can be used interchangeably, stage index is used here instead of bath index, to make the notation more similar to the previous model.

first stage of the first lot to start after it is transferred from the input buffer.

$$T_{t+1,s} \geq T_{t,s} + \sum_{i \in I} N_{i,t} p_{i,s} \quad \forall t \in T, s \in ZW : t \neq |T| \tag{C:2}$$

$$T_{t,s+1} = T_{t,s} + \sum_{i \in I} N_{i,t} p_{i,s} + \pi_{s+1} \quad \forall t \in T, s \in ZW : t \neq |T| \tag{C:3}$$

$$T_{1,1} \geq \pi_1 \tag{C:4}$$

For water stages, Constraint (C:5) ensures that there is no overlap between the processing of two subsequent lots. The required processing time is ensured by Constraint (C:6), and the transfer time is set by Constraint (C:7).

$$T_{t+1,s} \geq Te_{t,s} \quad \forall t \in T, s \in LS : t \neq |T| \tag{C:5}$$

$$Te_{t,s} \geq T_{t,s} + \sum_{i \in I} N_{i,t} p_{i,s} \quad \forall t \in T, s \in LS \tag{C:6}$$

$$T_{t,s+1} = Te_{t,s} + \pi_{s+1} \quad \forall t \in T, s \in LS \tag{C:7}$$

To consider flexible transfer times and LW policy, I replaced the Constraints (C:2)-(C:7)) with (C:8)-(C:13). The domain of $Te_{t,s}$ is extended to all stages to handle the LW policy.

$$Te_{t,s} \geq T_{t,s} + \sum_{i \in I} N_{i,t} t^{min}_{i,s} \quad \forall t \in T, s \in S \tag{C:8}$$

$$Te_{t,s} \leq T_{t,s} + \sum_{i \in I} N_{i,t} t^{max}_{i,s} \quad \forall t \in T, s \in S \tag{C:9}$$

$$T_{t+1,s} \geq Te_{t,s} \quad \forall t \in T, s \in S : t \neq |T| \tag{C:10}$$

$$T_{t,s+1} \geq Te_{t,s} + \pi^{min}_{s+1} \quad \forall t \in T, s \in S : s \neq |S| \tag{C:11}$$

$$T_{t,s+1} \leq Te_{t,s} + \pi^{max}_{s+1} \quad \forall t \in T, s \in S : s \neq |S| \tag{C:12}$$

$$T_{1,1} \geq \pi^{min}_1 \tag{C:13}$$

The pairing between lots and time slots is set by Constraints (C:14) and (C:15). Makespan is lower bounded in Constraint (C:16) by the last lot arriving at the output buffer, and Constraint (C:17) is added for more efficiency by connecting the other slot start time variables with the makespan. In my extended model, Constraint (C:17) is replaced by (C*:18).

$$\sum_{t \in T} N_{i,t} = 1 \quad \forall i \in I \tag{C:14}$$

$$\sum_{i \in I} N_{i,t} = 1 \quad \forall t \in T \tag{C:15}$$

$$T_{|T|,|S|} \leq MS \tag{C:16}$$

$$T_{t,s} + \sum_{i \in I} N_{i,t} \cdot \sum_{\substack{s' \in S: \\ s \leq s' \leq |S|}} (p_{i,s'} + \pi_{i,s'}) \leq MS \quad \forall t \in T, s \in S : s \neq |S| \tag{C:17}$$

$$T_{t,s} + \sum_{i \in I} N_{i,t} \cdot \sum_{\substack{s' \in S: \\ s \leq s' \leq |S|}} (t^{min}_{i,s'} + \pi^{min}_{i,s'}) \leq MS \quad \forall t \in T, s \in S : s \neq |S| \tag{C*:18}$$

The above constraints make up the URM (Unlimited Robot Model) which is used in the first step of the RCURM heuristic. To account for the limited availability of the robots, additional variables and constraints are needed.

The sequencing of transfer tasks are done with general precedence variables:

$$\bar{Y}_{t,s,t',s'} = \begin{cases} 1 & \text{if slot } t \text{ at stage } s \text{ starts later than slot } t' \text{ at } s' \\ 0 & \text{otherwise} \end{cases} \tag{C:19}$$

$$\forall t, t' \in T, s, s' \in S : t < t', s \neq s'$$

When there are multiple robots (MRM), an assignment variable, $\bar{W}_{t,s,r}$, is also needed to select which robot performs each transfer. Constraint (C:20) ensures that a single robot is selected for each. If the same robot that removes a lot from a bath transports the next lot into it, the big-M Constraints (C:21) and (C:22) ensure that there is enough time for both transfers. For transfers of different stages assigned to the same robot, Constraints (C:23) and (C:24) ensure that the one of the two transfers has precedence, and the other is executed after it.

$$\sum_{r \in R} \bar{W}_{t,s,r} = 1 \quad \forall t \in T, s \in S \tag{C:20}$$

$$T_{t+1,s} \geq T_{t,s} + \sum_{i \in I} N_{i,t} p_{i,s} + \pi_{s+1} + \pi_s - M \cdot (2 - \bar{W}_{t+1,s,r} - \bar{W}_{t,s+1,r}) \tag{C:21}$$

$$\forall t \in T, s \in ZW, r \in R : t \neq |T|$$

$$T_{t+1,s} \geq Te_{t,s} + \pi_{s+1} + \pi_s - M \cdot (2 - \bar{W}_{t+1,s,r} - \bar{W}_{t,s+1,r}) \quad \forall t \in T, s \in LS, r \in R : t \neq |T| \tag{C:22}$$

$$T_{t,s} \geq T_{t',s'} + \pi_s - M \cdot (3 - \bar{Y}_{t,s,t',s'} - \bar{W}_{t,s,r} - \bar{W}_{t',s',r}) \quad \forall t, t' \in T, s, s' \in S, r \in R : t < t', s \neq s' \tag{C:23}$$

$$T_{t',s'} \geq T_{t,s} + \pi_{s'} - M \cdot (2 + \bar{Y}_{t,s,t',s'} - \bar{W}_{t,s,r} - \bar{W}_{t',s',r}) \quad \forall t, t' \in T, s, s' \in S, r \in R : t < t', s \neq s' \tag{C:24}$$

The problem with Constraints (C:21)-(C:24) is that they do not ensure that the robot has enough time to travel from the endpoint of a transfer to the starting point of the next transfer. To fix this, I replaced these constraints with (C*:25)-(C*:27). To handle robot zones, only robots that are allowed for both transfers, are considered: R_s is the set of robots that can transfer to stage S (similarly to $R_{i,s}$ used before, but here the stages are the same for each lot).

$$T_{t+1,s} \geq Te_{t,s} + \pi^{min}_{s+1} + \pi^{min}_{s} + \pi^{abs}_{s-1,s+1} - M \cdot (2 - \bar{W}_{t+1,s,r} - \bar{W}_{t,s+1,r}) \quad \forall t \in T, s \in S, r \in R_s \cup R_{s+1} : s \neq |S|, t \neq |T| \tag{C*:25}$$

$$T_{t,s} \geq T_{t',s'} + \pi^{min}_{s} + \pi^{abs}_{s',s-1} - M \cdot (3 - \bar{Y}_{t,s,t',s'} - \bar{W}_{t,s,r} - \bar{W}_{t',s',r}) \quad \forall t, t' \in T, s, s' \in S, r \in R_s \cup R_{s'} : t < t', s \neq s' \tag{C*:26}$$

$$T_{t',s'} \geq T_{t,s} + \pi^{min}_{s'} + \pi^{abs}_{s,s'-1} - M \cdot (2 + \bar{Y}_{t,s,t',s'} - \bar{W}_{t,s,r} - \bar{W}_{t',s',r}) \quad \forall t, t' \in T, s, s' \in S, \in R_s \cup R_{s'} : t < t', s \neq s' \tag{C*:27}$$

With these additional constraints, the model is capable to schedule a finite number of robots. While that includes the single-robot case, the model can be formulated more efficiently for this special case (ORM). Not only the big-M parts associated with robot assignment can be removed, the domain of $\bar{Y}$ can be reduced [15]. This is presented here only on the extended model.

Equation (C:28) computes the lower bound of the position of transfer t, s, based on the fact that earlier lots will precede it at earlier stages, and they need to be transferred to subsequent stages. The upper bound is calculated similarly from the end of the sequence by Equation (C:29). Using these bounds, the value of $\bar{Y}_{t,s,t',s'}$ can be fixed for certain pairs of transfers by Constraints (C:30) and (C:31). Constraints (C*:26)-(C*:27) are replaced in the ORM by Constraints (C*:32)-(C*:33), which are only defined for pairs where $\bar{Y}_{t,s,t',s'}$ is not fixed to a value which would deactivate the constraint through the big-M term.

And by removing the robot assignment, Constraint (C*:25) is replaced by (C*:34).

$$LB_{t,s} = \sum_{\substack{t' \in T: \\ t' \leq t}} \sum_{\substack{s' \in S: \\ s' \leq s+t-t'}} 1 \quad \forall s \in S, t \in T \tag{C:28}$$

$$UB_{t,s} = |I| \cdot |S| + 1 - \sum_{\substack{t' \in T: \\ t' \geq t}} \sum_{\substack{s' \in S: \\ s' \geq s+t-t'}} 1 \quad \forall s \in S, t \in T \tag{C:29}$$

$$\bar{Y}_{t,s,t',s'} = 0 \quad \forall t, t' \in T, s, s' \in S : t < t', s \neq s', UB_{t,s} \leq LB_{t',s'} \tag{C:30}$$

$$\bar{Y}_{t,s,t',s'} = 1 \quad \forall t, t' \in T, s, s' \in S : t < t', s \neq s', UB_{t',s'} \leq LB_{t,s} \tag{C:31}$$

$$\begin{gathered} T_{t,s} \geq T_{t',s'} + \pi^{min}_{s} + \pi^{abs}_{s',s-1} - M \cdot \bar{Y}_{t,s,t',s'} \\ \forall t, t' \in T, s, s' \in S : t < t', s \neq s', UB_{t,s} > LB_{t',s'} \end{gathered} \tag{C*:32}$$

$$\begin{gathered} T_{t',s'} \geq T_{t,s} + \pi^{min}_{s'} + \pi^{abs}_{s,s'-1} - M \cdot (2 + \bar{Y}_{t,s,t',s'}) \\ \forall t, t' \in T, s, s' \in S : t < t', s \neq s', UB_{t',s'} > LB_{t,s} \end{gathered} \tag{C*:33}$$

$$\begin{gathered} T_{t+1,s} \geq Te_{t,s} + \pi^{min}_{s+1} + \pi^{min}_{s} + \pi^{abs}_{s-1,s+1} \\ \forall t \in T, s \in S : s \neq |S|, t \neq |T| \end{gathered} \tag{C*:34}$$

3.4 Computational tests

To measure the performance improvement of the precedence-based model, and validate the correctness of the extension for the hybrid time slot model, I implemented the models, and compared them on problems taken from the literature. The tests ran on a laptop with an Intel i7-8750H 6-core 2.20 GHz CPU, 16 GB RAM, using Gurobi 9.1, with a 1000 s solution time limit.

For the reentrant flexible job-shop problem class, I used the case study by Aguirre et al. [5], shown in Figure 3.2. I tested both the single-robot and 2-robot scenarios, using robot zones for the latter, as defined in [5]. The results are shown in Table 3.2.

In these tests, my proposed models (A*, Am*) highly outperformed the literature models. The obtained makespan values are the same as reported by Aguirre et al. [5]. All model variants obtained the same makespan for both instances. Because of the robot zones, the robots may only meet at one stage, so multi-robot models do not have much advantage even if 2 robots are present.

Table 3.2: Test results for the case study by Aguirre et al. [5]

$\|R\|$	CPU time (s) A	Am	A*	Am*	Makespan
1	453.75	796.27	10	5.47	161.2
2	66.06	73.12	2.63	3.04	160.05

For the permutation flow-shop problem-class, I used instances based on the 25-lot, 12-bath problem from [12]. This example was used for benchmark by many in the literature [4, 15, 72] by taking the first M stages of the first N jobs to get instances of different sizes.

Table 3.3 shows the solution times of the different models. The original model by Castro et al. [15] is denoted by C, and my extended version of the model by C*. Similarly to the results for the more general problem, my improved versions (A*, Am*) of the precedence-based models performed much better than the originals. However, for this special problem class, the hybrid time slot model performed even better which supports its extension for further problem features.

Table 3.3: Solution times (s) for permutation flow-shop problems with 1 robot

MxN	A	Am	A*	Am*	C	C*
4x4	1.03	2.28	0.13	0.12	0.03[1]	0.02
4x5	7.52	9.55	0.23	0.25	0.08[1]	0.04
4x6	26.32	29.64	0.56	0.56	0.15[1]	0.12
4x7	101.21	152.37	1.26	1.26	0.25[1]	0.21
4x9	–	–	15.10	15.05	1.05[1]	0.74
4x11	–	–	728.37	715.39	5.36[1]	4.74
6x4	9.54	9.95	0.27	0.27	0.04[1]	0.06
6x5	35.93	29.20	0.75	0.78	0.06[1]	0.10
6x7	–	–	5.95	5.94	0.53[1]	0.44
6x9	–	–	81.54	81.51	2.09[1]	1.75
8x5	83.40	165.33	1.83	1.85	0.21[1]	0.15
8x7	–	–	11.42	9.56	0.93[1]	1.02
10x7	5.35	–	20.82	20.53	1.79[1]	1.53
10x9	0.40	–	319.31	318.80	8.14[1]	183.12
12x5	–	–	5.83	5.85	0.50[1]	0.46
12x7	–	–	32.91	32.85	2.44[1]	8.19

[1] : Empty robot movement not considered.
– : Terminated after 1000 s.

As it can be seen in Table 3.4, the original model (C) provided practically infeasible

schedules by neglecting the empty robot movements. Considering the empty movements led to higher solution times, especially when the number of stages (M) was higher (10, 12), the C* model still has a better performance than the precedence-based models.

Table 3.4: Makespan for permutation flow-shop problems with 1 robot

MxN	A	Am	A*	Am*	C	C*
4x4	51.76	51.76	51.76	51.76	50.67[1]	51.76
4x5	63.42	63.42	63.42	63.42	62.00[1]	63.42
4x6	70.92	70.92	70.92	70.92	69.15[1]	70.92
4x7	78.38	78.38	78.38	78.38	76.18[1]	78.38
4x9	97.78[2]	94.08[2]	93.54	93.54	90.68[1]	93.54
4x11	–	133.40[2]	112.30	112.30	108.81[1]	112.30
6x4	62.03	62.03	62.03	62.03	61.08[1]	62.03
6x5	73.69	73.69	73.69	73.69	72.41[1]	73.69
6x7	93.47[2]	–	93.47	93.47	90.83[1]	93.47
6x9	–	–	108.15	108.15	104.65[1]	108.15
8x5	93.04	93.04	93.04	93.04	91.92[1]	93.04
8x7	–	–	109.38	109.38	106.55[1]	109.38
10x7	–	–	125.61	125.61	122.45[1]	125.61
10x9	–	–	146.67	146.67	141.57[1]	146.67
12x5	130.16[2]	–	130.16	130.16	128.90[1]	130.16
12x7	–	–	144.58	144.58	142.66[1]	144.58

[1] : Empty robot movement not considered.
[2] : Incumbent solution value after 1000 s.
– : No solution found in 1000 s.

Solution times for the 2 robot scenario are shown in Table 3.5. Robot zones were not used, which makes the problem more difficult, as both robots are available for all transfers. The performance ranking is similar to the single robot scenario. The obtained makespan values shown in Table 3.6 give some additional insight about the models. It can be seen that models A and A* provide suboptimal solutions for the multi-robot problems, so the fixed constraints proposed by [15] are necessary. However, the fixed version of the original precedence-based model (Am) still provided suboptimal solutions in some cases (4x4 and 6x4) because of the issue with Constraint (A:28) discussed earlier, and also provided an infeasible solution in one case (4x7) due to numerical instability.

I investigated the cause of the numerical instability. The model definition of [5] did not specified if the variables K and Pos should be defined as integer or continuous variables. I used continuous variables in the tests reported here which caused numerical instability for instance 4x7 with big-M values of 256, 512, or 1024. Here, $K_{7,1,1,3,1} = 0.00029296875$

was obtained instead of 0. This variable is multiplied by M in Constraint (A:27) which resulted in $pi_{7,1}^{free}$ being lower-bounded by 0 instead of $pi_{7,1,1,3}^{abs}$, whose value is 0.3. If integer variables are used, the solver cannot find a solution in 1 hour. My improved model (Am*) does not have that issue, as it does not contain K and Pos variables.

Table 3.5: Solution times (s) for permutation flow-shop problems with 2 robots

MxN	A	Am	A*	Am*	C	C*
4x4	16.11	12.71	0.07	0.09	0.11[1]	0.05
4x5	31.12	32.47	0.35	0.48	0.21[1]	0.15
4x6	76.86	190.02	1.26	1.40	0.55[1]	0.37
4x7	315.36	830.64	4.77	8.17	1.05[1]	1.15
4x9	–	–	–	41.22	3.00[1]	2.87
4x11	–	–	–	–	25.82[1]	40.41
6x4	17.42	41.08	0.20	0.25	0.10[1]	0.15
6x5	68.21	95.42	1.03	0.83	0.36[1]	0.38
6x7	–	–	8.47	7.01	2.77[1]	3.05
6x9	–	–	160.52	64.22	6.73[1]	8.06
8x5	325.46	–	2.45	2.60	0.68[1]	0.68
8x7	–	–	16.25	15.91	6.26[1]	5.59
10x7	–	–	23.07	22.56	3.40[1]	7.52
10x9	–	–	238.72	176.36	26.88[1]	35.87
12x5	–	–	9.92	7.73	0.97[1]	2.23
12x7	–	–	45.32	44.75	16.13[1]	15.45

[1] : Empty robot movement not considered.
– : Terminated after 1000 s.

Table 3.6: Solution times (s) for permutation flow-shop problems with 2 robots

MxN	A	Am	A*	Am*	C	C*
4x4	50.67[3]	49.78[3]	50.67[3]	49.58	49.58[1]	49.58
4x5	62.00[3]	60.58	62.00[3]	60.58	60.58[1]	60.58
4x6	69.15[3]	67.38	69.15[3]	67.38	67.38[1]	67.38
4x7	76.18[3]	74.31[4]	76.36[3]	74.84	73.98[1]	74.84
4x9	–	–	91.05[2]	87.98	87.78[1]	87.98
4x11	–	–	109.26[2]	106.08	105.28[1]	106.08
6x4	61.08[3]	60.43[3]	61.08[3]	60.15	60.15[1]	60.15
6x5	72.41[3]	71.15	72.41[3]	71.15	71.15[1]	71.15
6x7	91.07[2,3]	–	91.07[3]	88.75	88.75[1]	88.75
6x9	–	–	104.89[3]	102.25	101.98[1]	102.25
8x5	91.92[3]	91.17[2]	91.92[2]	90.82	90.82[1]	90.82
8x7	–	–	106.90[3]	105.22	104.52[1]	105.22
10x7	–	–	122.45[3]	120.61	120.61[1]	120.61
10x9	–	–	141.84[3]	139.01	139.01[1]	139.01
12x5	–	–	128.90[3]	127.64	127.64[1]	127.64
12x7	–	–	142.66[3]	140.74	140.74[1]	140.74

[1] : Empty robot movement not considered.
[2] : Incumbent solution value after 1000 s.
[3] : Suboptimal solution.
[4] : Infeasible due to numerical instability.
– : No solution found in 1000 s.

3.5 Summarizing statements

Thesis statement 1 *I have developed improvements to existing MILP models for scheduling automated wet-etch stations and other automated manufacturing systems, accelerating the more general model by orders of magnitude, and extending the solvable problem class of the more specialized model.*

Thesis statement 1/a *I have improved the performance of the model by Aguirre et al. (2013) by developing new constraints for modeling empty robot movement without requiring immediate precedence variables.*

Thesis statement 1/b *I have extended the model by Castro et al. (2012) to be able to handle empty robot movements, robot zones, limited wait times, and flexible transfer times, and validated its correctness on literature problem instances.*

My publications related to the statement: [44] and presentations [41, 77]

Chapter 4

S-graph approach for RCPSP and its variants

The Resource-Constrained Project Scheduling Problem (RCPSP) originated from project management sciences, however, it has also been used for modeling several industrial processes, such as CNC machining [87], plastic injection molding [82], and complex construction projects [108]. It is a more general problem class than most machine scheduling problems, as it allows a task to require varying amounts from multiple types of resources for its execution. These resources have limited but renewable capacities. Tasks consume them at their start, and return them at their completion. Therefore, tasks must be scheduled such that the cumulative resource needs of tasks executed in parallel do not exceed the capacity of any resource at any moment. Renewable resources can be anything from machines, workers, and tools, to limited physical locations at a construction site.

The RCPSP has several variants, and different approaches have been proposed for solving them. I extended the S-graph framework with new algorithms to solve some RCPSP variants. The problem classes are defined in Section 4.1. A summary of the literature approaches are presented in Section 4.2. My proposed solution method is explained in Section 4.3, and the results of computational tests are discussed in Section 4.4.

4.1 Problem definitions

There are many variants of the RCPSP [38, 39], here, only those are introduced which I investigated for potential extensions of the S-graph framework.

The input data of the classic RCPSP are the following:

T is the set of tasks (also called as activities).

H is the set of precedence relations. ($H \subset T \times T$)

R is the set of resources.

d_i are the task durations. ($\forall i \in T$)

b_k are the resource capacities. ($\forall k \in R$)

$r_{i,k}$ are the resource usages of tasks. ($\forall i \in T, k \in R$)

The goal is to provide a schedule with minimal completion time, that satisfies the precedence and resource constraints. That means determining the start time of each task, $s_i \geq 0$, such that:

$$\text{minimize} \max_{i \in T}(s_i + d_i) \tag{4.1}$$

Any task can only start after all of its predecessors are finished:

$$s_{i'} \geq s_i + d_i \quad \forall (i, i') \in H \tag{4.2}$$

At any moment, the total resource usages of the tasks in progress do not exceed their respective capacities:

$$\sum_{\forall i \in T:\, s_i \leq t < s_i + d_i} r_{i,k} \leq b_k \quad \forall k \in R, t \in \mathbb{R} \tag{4.3}$$

The multi-mode variant (MRCPSP) is an important extension of the classic problem. It allows tasks to have multiple alternative operation modes, that can differ in resource usage and duration. For example, a task may be done faster using more resources, or conversely, extended over a longer time period while saving resources.

While in the single-mode problem, the total resource usage of all the tasks is independent of the schedule, in the multi-mode problem, it depends on the chosen operation modes of the tasks. This makes it possible to define resource capacities not just for momentary usages but for the entire project. This is useful for modeling the limited availability of non-renewable resources, such as raw materials. While renewable resources are consumed at the start of a task and released at its end, non-renewable resources are only consumed and never get regenerated. This means that the capacities of non-renewable resources limit the total consumption, and render some operation mode assignments infeasible, regardless of the timing decisions.

The input data of the MRCPSP are the following:

T is the set of tasks (also called activities).

H is the set of precedence relations. ($H \subset T \times T$)

R_R is the set of renewable resources.

R_N is the set of non-renewable resources.

M_i are the sets of possible operation modes of tasks. $(\forall i \in T)$

$d_{i,m}$ are the task durations. $(\forall i \in T, m \in M_i)$

b_k are the resource capacities. $(\forall k \in R_R \cup R_N)$

$r_{i,m,k}$ are the resource usages of tasks. $(\forall i \in T, m \in M_i, k \in R_R \cup R_N)$

For the MRCPSP, in addition to the start times, a single operation mode, $m_i \in M_i$, must be chosen for each task $(\forall i \in T)$. Resource usages are dependent on the mode selection, so renewable resource constraints are modified this way:

$$\sum_{\forall i \in T:\, s_i \leq t < s_i + d_{i,m_i}} r_{i,m_i,k} \leq b_k \quad \forall k \in R_R, t \in \mathbb{R} \tag{4.4}$$

And non-renewable resource constraints must be satisfied too:

$$b_k \geq \sum_{i \in T} r_{i,m_i,k} \quad \forall k \in R_N \tag{4.5}$$

Another generalization of the problem introduces time-varying resource capacities. While the classic problems assume that every resource is available all the time, this variant allows the modeling of planned changes in resource availability. For example, external specialists may only be present for a limited time during a project, or different work shifts have different number of workers. In this variant, renewable resource capacities are given as piecewise constant functions over time. This extension can be applied to both single- and multi-mode problems.

4.2 Literature approaches

A discrete time point MILP model for the RCPSP was first introduced by Pritsker et al. [86], then it was extended to the multi-mode case by Talbot [96]. The discrete time formulation provides a simple way for constraining the cumulative resource usages of all active tasks at a time point from exceeding the resource capacities. However, as it was discussed in Section 2.2, the high number of binary variables makes this type of model computationally inefficient, and can also lead to suboptimal schedules if durations are not multiples of the interval length.

Olaguíbel and Goerlich [73] proposed a continuous time model for the single-mode problem, where the start time of each task is set by a continuous variable. This method

makes resource balance constraints impossible, so resource constraints are ensured by defining incompatible sets of tasks, whose parallel execution is infeasible, so it must be avoided with sequencing decisions. As my S-graph approach is based on the same idea, it will be explained in more detail in Section 4.3.

Kyriakidis et al. [57] proposed a multi-mode model based on variable-length time slots and the Resource-Task Network (RTN) [83] resource representation method. The timing of the time slots are set by continuous duration variables, so the start time of a slot can be calculated by summing the durations of previous time slots. However, they do not need to be calculated, as precedences are modeled with RTN as production and consumption of virtual resources. 2 sets of binary variables represent whether an activity starts at the start of a time slot, or is in progress during a later time slot. These are used to constrain the lengths of the time slots to be long enough for executing their assigned tasks. Another set of binary variables handles operation mode selection. With the introduction of virtual resources for each precedence relation, the number of variables increase steeply with the number of tasks, so this model proved to be inefficient.

The event-based formulations by Koné et al. [54] for RCPSP were extended to the multi-mode variant by Chakrabortty et al. [17]. The events used in these models are similar to the starting points of the time slots mentioned above, but the continuous variables set the times of the events, not the lengths of the time slots. The binary assignment variables have 3 indices: task, operation mode, and event. Based on the types of events considered, two different models are defined. In the Start-Stop Event (SEE) model, 2 types of variables are used to assign starts and ends of tasks to events. In the On/Off Event (OOE) model, there is only one type of binary variable, which denotes whether a task is active at a certain event. Precedence relations are modeled without additional variables by taking advantage of the ordered events: a task can only start or be active at an event, if all of its predecessors have been finished at earlier events. This results in an efficient formulation, especially with the OOE model.

The problem variant with time-varying resource capacities attracted less attention in the literature, and most approaches only consider the single-mode case [9, 23, 50, 37]. Cheng et al. [19] developed a precedence tree-based branch-and-bound algorithm to solve the multi-mode case with also allowing non-preemptive activity splitting (tasks can only be paused due to lack of resources). Kreter et al. [56] further generalized the constraints for activity splitting with time-bounded breaks and resources remaining engaged dur-

ing breaks. The authors proposed 3 types of binary linear models, and heuristic search procedures for optimization.

4.3 Proposed S-graph solution method

One way to ensure that resource constraints are satisfied is to identify the groups of tasks whose parallel execution would violate them, and introduce precedence relations among them. Such task groups are called incompatible sets [73], and it is enough to find the minimal sets with that property, so they will be referred to as MRISs (Minimal Resource Incompatible Sets).

I developed a new branching procedure for the S-graph framework, which uses MRISs to solve the RCPSP. This is presented in Section 4.3.1, its extension to the multi-mode variant in Section 4.3.2, and to time-varying resource capacities in Section 4.3.3.

4.3.1 Solution for the single-mode problem

The proposed solution procedure starts with finding all MRIS, then resolves incompatibilities in the branching step of the B&B algorithm by making sequencing decisions.

To find the MRISs, I created a Constraint Programming (CP) model of a satisfaction problem, whose solutions are the MRISs. A CP solver can be used to find all feasible solutions, thereby providing the set of all MRISs. $x_i \in \{0,1\}\ \forall i \in T$ denotes whether a task is a member of the set. $u_k \geq 0\ \forall k \in R$ is the total resource usage of the set, set by Equation (4.1). $v_k \in \{0,1\}\ \forall k \in R$ denotes whether the capacity constraint of a resource is violated, set by Constraint (4.2). Constraint (4.3) requires that at least one resource capacity is violated, so the set is incompatible. Constraint (4.4) ensures that the set is minimal, so removing any member would make the set satisfy every resource constraint.

$$\sum_{i \in T} x_i \cdot r_{i,k} = u_k \quad \forall k \in R \tag{4.1}$$

$$u_k > b_k \iff v_k = 1 \quad \forall k \in R \tag{4.2}$$

$$\sum_{k \in R} v_k \geq 1 \tag{4.3}$$

$$x_i \cdot (u_k - r_{i,k}) \leq b_k \quad \forall i \in T, k \in R \tag{4.4}$$

An MRIS only violates resource constraints if all of its tasks are executed in parallel.

If a precedence relation is set between any two of its tasks, the set is resolved. When all MRISs are resolved, the schedule is resource-feasible, and mandatory precedence constraints are satisfied by recipe-arcs in S-graphs ($c(i,i') = d_i \ \forall (i,i') \in A_1$). Products are not considered in project scheduling, so a single product node can be used to represent the completion of the project. An illustrative example is taken from Mingozzi et al. [69], whose recipe graph is shown in Figure 4.1. All 3 resources have 4 unit capacity, and the resource usages of tasks are shown below the task nodes.

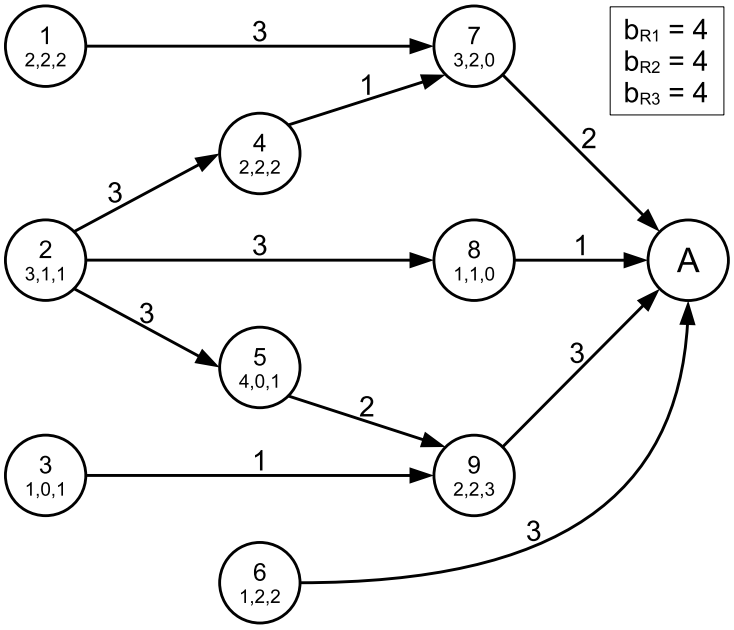


Figure 4.1: Example RCPSP recipe graph

The new branching method requires storing the set of unresolved MRISs ($\mathcal{I}$) at each B&B node. So a B&B node is represented by $(G(N, A_1 \cup A_2), c, \mathcal{I})$. The branching step creates child nodes from it in the following way:

1. If all MRISs have been resolved ($\mathcal{I} = \emptyset$), the node is a solution, so return an empty set of children.
2. Select an MRIS, $I \in \mathcal{I}$, which has not yet been resolved.
3. For each task pair in the MRIS ($\forall (i,i') \in I \times I : i \neq i'$), create a child node $(G(N, A_1 \cup A_2'), c', \mathcal{I}')$, where i must be finished before i' can be started:
   - Insert (i,i') schedule-arc: $A_2' = A_2 \cup \{(i,i')\}, c'(i,i') = d_i$
   - Other arc weights remain unchanged: $c'(j,k) = c(j,k) \ \forall (j,k) \in A_1 \cup A_2$
   - Remove resolved MRISs: $\mathcal{I}' = \{I' \in \mathcal{I} \mid \{i,i'\} \nsubseteq I'\}$
4. Return the set of children created in the previous step.

Schedule-arcs are inserted with the UIS method, as ample storage is assumed in the

RCPSP literature. However, if this approach is used for scheduling a production system where some of the resources are machines that act as intermediate storage between stages, the NIS schedule-arcs of the S-graph method can be used to correctly model the storage constraints.

The effect of the arc insertion is demonstrated in Figure 4.2, where the MRIS of $\{7, 9\}$ is selected and resolved by fixing their order. Here, the heights of the rectangles correspond to the resource usages of R1. Other tasks are only ordered by mandatory precedences. Continuing the resolution of MRISs leads to resource feasible schedules, from which an optimal solution is shown in Figure 4.3.

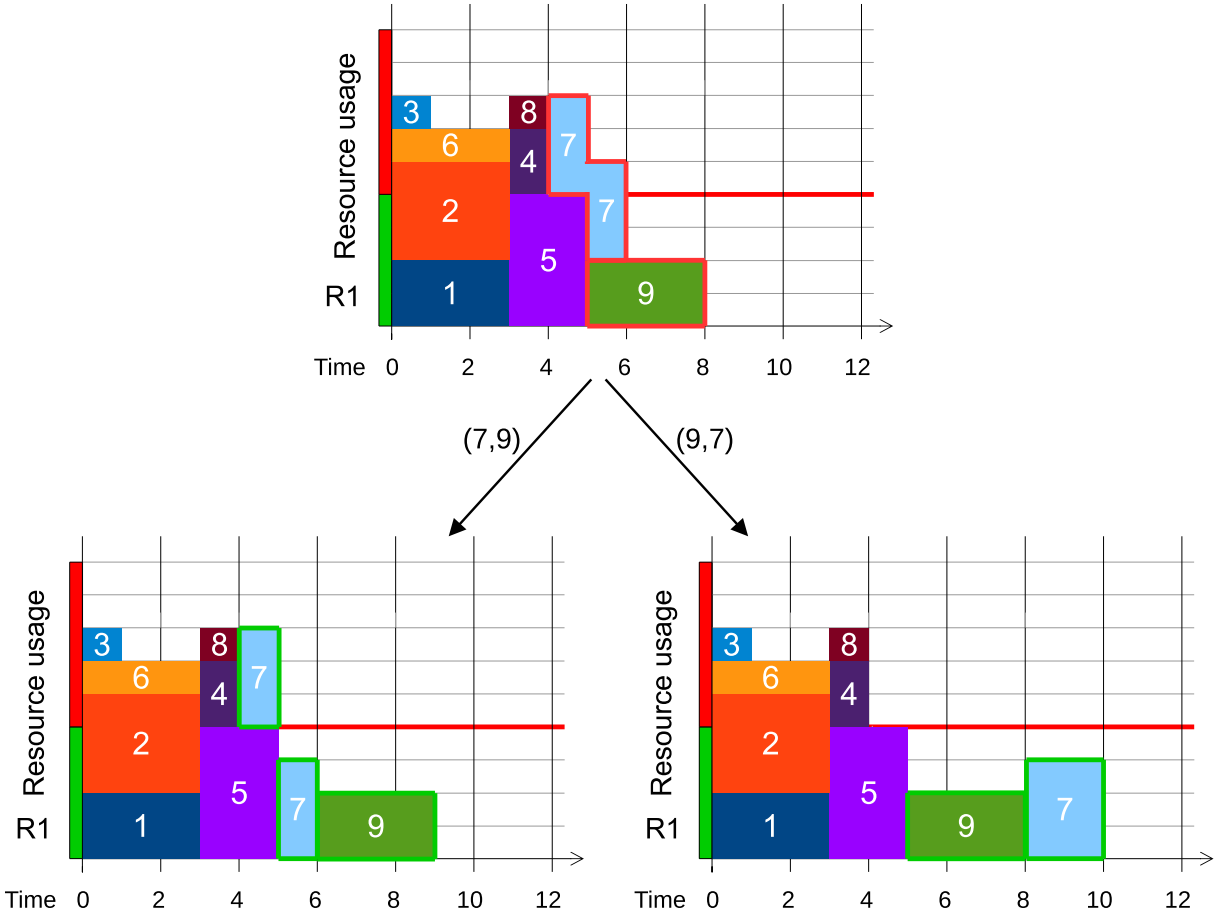


Figure 4.2: Resolving the incompatibility between tasks 7 and 9

4.3.2 Solution for the multi-mode variant

I extended the idea of incompatible sets for the multi-mode variant. As operation modes can differ in resource usages, task-mode pairs need to be the members of MRISs, not tasks.

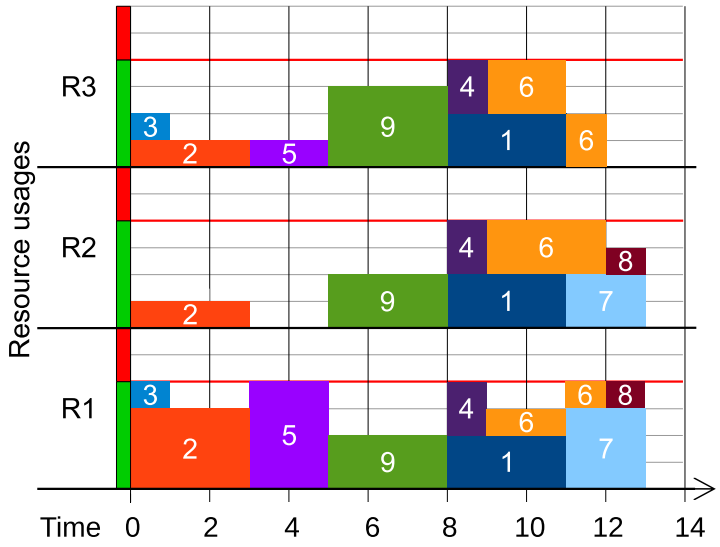


Figure 4.3: Optimal schedule with all 3 resource usages presented

So variable $x_{i,m}$ is now indexed by both task and mode but Constraint (4.1) ensures that multiple modes of the same task will not be members of an MRIS, as this incompatibility is resolved by any mode selection. The rest of the constraints are almost the same as in the single-mode model, just extended with mode indices and non-renewable resources.

$$\sum_{m \in M_i} x_{i,m} \leq 1 \quad \forall i \in T \tag{4.1}$$

$$\sum_{i \in T} \sum_{m \in M_i} x_{i,m} \cdot r_{i,m,k} = u_k \quad \forall k \in R_R \cup R_N \tag{4.2}$$

$$u_k > b_k \iff v_k = 1 \quad \forall k \in R_R \cup R_N \tag{4.3}$$

$$\sum_{k \in R_R \cup R_N} v_k \geq 1 \tag{4.4}$$

$$x_i \cdot (u_k - r_{i,m,k}) \leq b_k \quad \forall i \in T, m \in M_i, k \in R_R \cup R_N \tag{4.5}$$

In the multi-mode variant, there are two types of incompatibility, depending on whether only renewable resource constraints are violated by the set ($v_k = 0 \; \forall k \in R_N$), or non-renewable capacities as well. In the latter case, the set will be referred to as an infeasible set, which represents a forbidden mode assignment. The set of infeasible sets is denoted by $\mathcal{I}_N \subseteq \mathcal{I}$. These can only be resolved by different mode assignments, not by sequencing decisions, while regular incompatible sets can be resolved by both types of decisions.

Multi-mode branching also starts by selecting an incompatible set, but now there are

more ways to resolve it. Mode assignment decisions are also made, and stored in set $M \subseteq \{(i,m) \mid \forall i \in T, m \in M_i\}$ at each node.

Branching at node $(G(N, A_1 \cup A_2), c, M, \mathcal{I}, \mathcal{I}_N)$ is done in the following way:

1. If all MRISs have been resolved ($\mathcal{I} = \emptyset$), the schedule is a feasible solution[1], so return an empty set of children.
2. Select an MRIS, $I \in \mathcal{I}$, which has not yet been resolved.
3. If I is not an infeasible set ($I \notin \mathcal{I}_N$), create children $(G(N, A_1 \cup A_2'), c', M', \mathcal{I}', \mathcal{I}_N')$ for each pair of the set $(\forall((i,m),(i',m')) \in I \times I : i \neq i')$:
   - Make the mode assignments for the tasks inside I, and update the weights of their outgoing arcs:
     $M' = M \cup I,\ c'(j,j') = d_{j,n} \quad \forall (j,n) \in I, (j,j') \in A_1$
   - Insert (i,i') schedule-arc: $A_2' = A_2 \cup \{(i,i')\}, c'(i,i') = d_{i,m}$
   - Other arc weights remain unchanged.
4. Also create child nodes $(G(N, A_1 \cup A_2'), c', M', \mathcal{I}', \mathcal{I}_N')$ where I is resolved by different mode assignments of its unscheduled tasks ($I \setminus M$). The set of different mode assignments is $\mathcal{M} = \{M^* \mid \forall (i,m) \in I \setminus M, \exists m' \in M_i : (i,m') \in M^*, |M^*| = |I \setminus M|\} \setminus \{I \setminus M\}$. Create a child node with each such assignment ($\forall M^* \in \mathcal{M}$):
   - Make the mode assignments for the tasks inside M^*, and update the weights of their outgoing arcs:
     $M' = M \cup M^*,\ c'(j,j') = d_{j,n} \quad \forall (j,n) \in M^*, (j,j') \in A_1$
   - No schedule-arcs are added ($A_2' = A_2$) and other arc weights remain unchanged.
5. In the child nodes generated above, remove the infeasible and incompatible sets that have been resolved:
   $\mathcal{I}_N' = \mathcal{I}_N \setminus \{I' \in \mathcal{I}_N \mid \exists (i,m) \in I, (i,m') \in I' : m \neq m'\}$
   $\mathcal{I}' = \mathcal{I} \setminus \{I' \in \mathcal{I} \mid (\exists (i,m) \in I, (i,m') \in I' : m \neq m') \vee (\exists (i,m),(i',m') \in I',$
   $\{(i_0 := i, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k := i')\} \subseteq A_1 \cup A_2' : \sum_{v=0}^{k-1} c'(i_v, i_{v+1}) \geq d_{i,m})\}$
6. Return the set of child nodes.

This branching method can result in lots of child nodes, especially when tasks have many operation modes. However, the mode assignments resolve a lot of MRISs in the beginning, resulting in a wide but not too deep search tree.

[1] It is possible that some tasks have no selected modes, if that does not impact the feasibility. In this case, the fastest mode can be selected for each such task.

4.3.3 Solution for time-varying resource capacities

To model the time-varying resource availabilities, virtual tasks are used, as proposed by Bartusch et al. [9]. Resource capacities are set to their maximum availability, then virtual tasks with fixed timing decrease their availability for actual tasks, during time periods where the capacity is lower. To fix the timing of the virtual tasks, a reference node and ZW-arcs are introduced. The additional 0 node represents the start of the time horizon, it is a task with 0 duration and no resource usage. To fix the start time of a virtual task to s, a ZW-arc with weight s is added starting at 0 and going to the node of the virtual task. The length of the virtual task (the resource disruption) is modeled by the recipe-arc from the virtual task node to the product node.

LW and ZW-arcs in S-graphs were first introduced by Hegyháti et al. [43], where a combinatorial algorithms was proposed for checking the waiting time constraints. Later, a different approach was proposed for LW/ZW tasks in wet-etch scheduling based on reversed arcs with negative weights [44]. A more recent study compared different LW/ZW techniques [42], and the negative arc method proved to be the most efficient. This technique was used here as well.

An LW task is modeled by an extra LW-arc for each recipe-arc but going in the reverse direction, and instead of having a weight of d_i, its weight is $-(d_i + max\_wait)$. In LW case, max_wait is the maximal allowed waiting time. ZW is the special case where $max\_wait = 0$.

The reverse arcs create directed cycles in the graph, which used to indicate infeasibility in the original S-graphs. With LW-arcs however, cycles with negative total weight are always feasible. If the weight of a cycle is 0, it is only infeasible if no LW-arcs are part of it. If a cycle with weight 0 contains LW-arcs, that means that the waiting time is at the maximal allowed amount. If a cycle has positive weight, it is always infeasible.

Virtual tasks have durations equal to the length of the resource disruptions. However, unlike regular tasks and the 0 node, nodes of virtual tasks do not have recipe-arcs. They are removed to prevent virtual tasks from unnecessarily increasing the length of the longest path, and hence the completion time. If the $V \rightarrow A$ recipe-arc was present between a virtual task V and the product node A, and V ends later than the regular tasks, the longest path would be $0 \rightarrow V \rightarrow A$ with a length equal to the finish time of the resource disruption. Durations of virtual tasks only appear as weights of schedule-arcs starting from their nodes. Virtual tasks are included in MRISs too, so schedule-arcs may be

introduced between them and regular tasks, to decide if a task requiring a resource whose capacity is decreased, is executed before or after the disruption.

Figure 4.4 shows the recipe graph of an example where no resources are available at the 4-5 and 7-9 time periods. Additional nodes are highlighted in blue, including the 2 virtual tasks, starting at 4 and 7 time units. They have durations of 1 and 2 time units, although they cannot be seen on the recipe graph, as the corresponding recipe-arcs are not present, as explained above. If the decreased capacity is not enough to perform a task, that task and the virtual task will be in an MRIS. Through sequencing decisions, the actual task will be scheduled before or after the virtual task. Figure 4.5 shows a solution of the example.

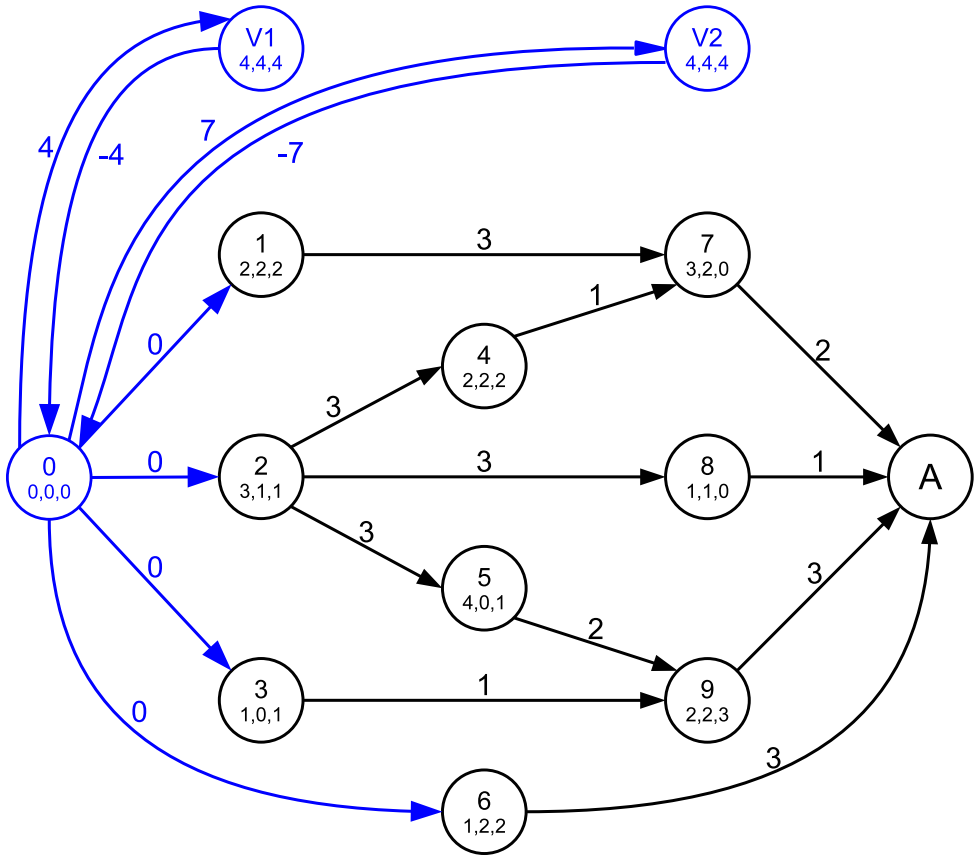


Figure 4.4: Example RCPSP recipe graph with virtual tasks

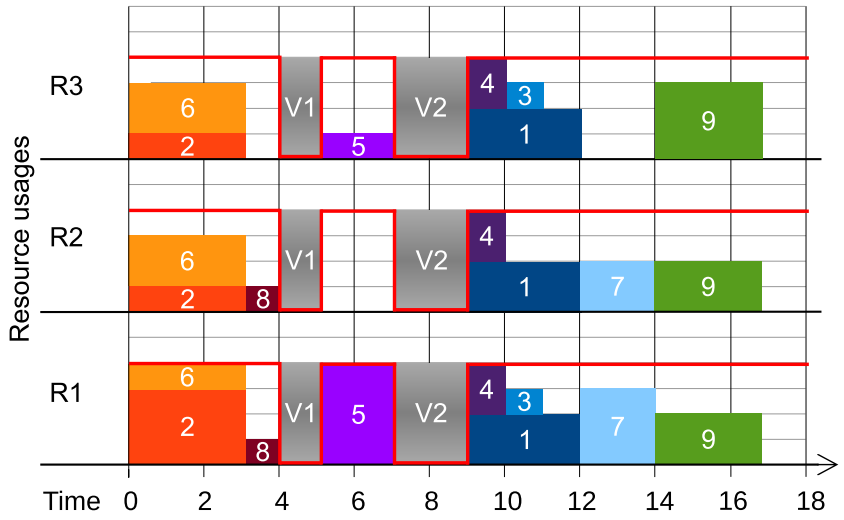


Figure 4.5: Schedule of the example with time-varying resource capacities

4.4 Computational tests

I validated the correctness of the proposed approach on benchmark problems of the PSPLIB [51] collection. I also compared its performance to MILP models from the literature. A 1000 s solution time limit was set for each instance. MILP models were solved with Gurobi and Coin-OR (CBC) solvers, and the S-graph algorithms were implemented into the preexisting solver written in C++. The CP models used by the S-graph preprocessor were solved with Google OR-tools solver, which took less than 1 second in most cases (this time is included in the reported solution times).

4.4.1 Single-mode results

For the single-mode RCPSP, the j30 dataset of the PSPLIB was used, which contains 480 instances that were randomly generated with 48 different parameter settings. Each instance contains 30 tasks and 4 resources.

I implemented 2 MILP models by Kopanos et al. [55] for the comparison, a discrete-time (Kop-DT1) and a continuous-time (Kop-CT1) formulation. The single-mode tests ran on a computer with an Intel Core i5-660 3.33 GHz CPU and 4 GB RAM. The MILP models were solved with both CBC 2.9.7 and Gurobi 7.0.

From the 480 instances the following numbers were solved to optimality within the 1000 s time limit:

Kop-DT1 (Gurobi)	423
Kop-CT1 (Gurobi)	462
Kop-DT1 (CBC)	321
Kop-CT1 (CBC)	335
S-graph	360

Average solution times of solved cases showed a similar ranking, with Gurobi in the lead, followed by the S-graph, and CBC at the end. From the 2 MILP models, Kop-CT1 performed better. Instead of showing the results for all 480 instances, I selected 2 parameter sets to show the performance differences. The set j30_1 proved to be one of the easiest, while set j30_5 contains much more difficult instances.

Table 4.1 shows the solution times for j30_1. The best-performing methods, Kop-CT1 (with Gurobi) and S-graph, managed to solve these instances under a few milliseconds. However, CBC could not solve the Kop-DT1 model in instance 4 under the time limit.

Table 4.1: Solution times (s) for dataset j30_1

Instance	Gurobi		CBC		S-graph
	Kop-DT1	Kop-CT1	Kop-DT1	Kop-CT1	
1	3.26	0.04	366.0	1.8	0.04
2	3.77	0.05	114.9	0.9	0.05
3	1.72	0.05	56.2	0.3	0.04
4	8.29	0.14	–	5.7	0.19
5	7.42	0.12	309.1	2.7	0.05
6	2.38	0.10	112.5	1.1	0.06
7	1.03	0.02	40.2	0.2	0.04
8	3.55	0.11	75.1	0.3	0.04
9	7.65	0.05	77.6	3.5	0.07
10	3.66	0.10	53.6	1.2	0.04

– : Terminated after 1000 s.

The solution times for j30_5 are shown in Table 4.2. Here, the larger differences show the performance ranking more clearly. Kop-CT1 is the best approach but only if using Gurobi. Compared to the open-source CBC solver, the proposed S-graph approach performed better in most cases.

4.4.2 Multi-mode results

For testing the multi-mode variant, I used the j10 and j12 datasets of PSPLIB. These instances were generated with 64 different parameter settings in each dataset, creating 10

Table 4.2: Solution times for dataset 5 of j30

Instance	Gurobi		CBC		S-graph
	Kop-DT1	Kop-CT1	Kop-DT1	Kop-CT1	
1	31.68	1.54	–	277.0	423.10
2	659.77	1.77	–	157.9	13.89
3	676.62	5.71	–	355.9	96.03
4	854.70	8.59	–	–	270.55
5	220.26	2.69	–	139.3	1.68
6	89.52	2.17	163.9	522.3	6.25
7	–	31.72	–	–	–
8	234.84	4.72	–	–	9.88
9	31.24	3.78	–	384.4	116.88
10	143.72	1.45	–	187.2	3.61

– : Terminated after 1000 s.

instances with each. However, some instances are infeasible due to lack of non-renewable resource capacities, so there are only 536 and 547 feasible instances, respectively. There are 10 (j10) or 12 (j12) tasks in each instance, with 3 operation modes for every task, and 2 renewable, 2 non-renewable resources.

I implemented literature methods for comparison and validation of the proposed method. The models by Kyriakidis et al. [57] could barely solve the most simple instances under the 1000 s time limit, and even those took tens or hundreds of seconds compared to under 1 s solution times of other methods. The SEE and OOE models by Chakrabortty et al. [17] performed much better, so I omit the test results of the Kyriakidis models.

The multi-mode tests ran on a laptop with an Intel i7-8750H 6-core 2.20 GHz CPU, 16 GB RAM, using Gurobi 9.1 as the MILP solver. Solution times on the j10 dataset are shown in Figure 4.6.

From the 2 MILP models, OOE performed better, with an average solution time of 28.88 s, compared to the 49.20 s average time of SEE, which even hit the time limit for 2 instances. The S-graph solver had an average solution time of 7.73 s, with solving 522 instances in under 20 s. However, the S-graph solver was terminated due to lack of memory in 3 instances and failed to provide a solution. These are highlighted with solid markers in the chart. The other few cases of high solution times were also caused by the lack of physical memory, and resorting to the slower swap memory.

On the j12 dataset, the results are similar, only the MILP models hit the time limit in more cases, and the S-graph ran out of memory more times too. Table 4.3 presents the

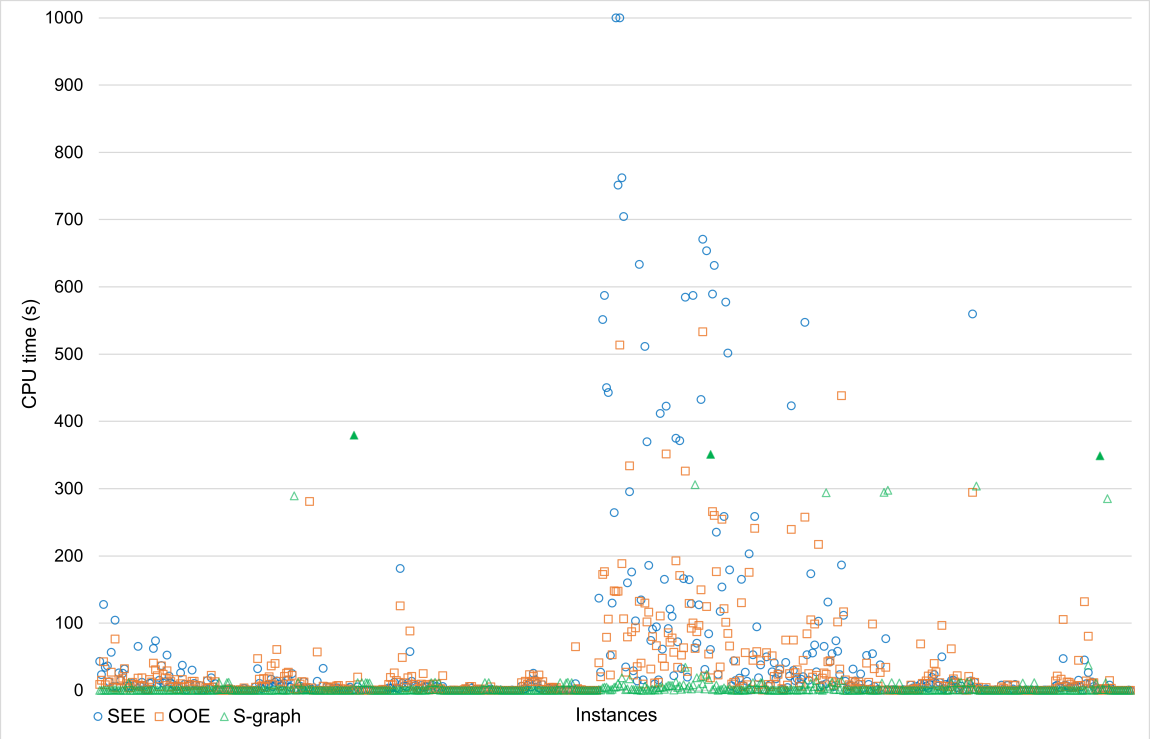


Figure 4.6: Solution times for the j10 dataset

results in more detail. The tests showed that the S-graph is faster on average but needs a lot of memory space for some instances. The MILP models take a long time to prove the optimality of a solution but they can still provide optimal solutions most of the time, thanks to the heuristics in Gurobi.

Table 4.3: Solution statistics for the j12 dataset

	SEE	OOE	S-graph
Total number of instances	547	547	547
Solved to proven optimality in 1000 s	443	480	504
Found optimal solution even if optimality was not proven under 1000 s	536	547	509
Finished without running out of memory	547	547	509
Average running time (on all instances)	289.86 s	248.47 s	72.25 s
Average solution time (on proven optimal instances)	125.07 s	145.34 s	65.41 s
Average solution time (on instances solved to optimality by all methods)	124.80 s	107.43 s	35.88 s

4.5 Summarizing statements

Thesis statement 2 *I have extended the S-graph framework with new branching algorithms and modeling methods to solve the RCPSP with single or multiple modes per task, and with constant or piecewise linear time-varying resource capacities.*

Thesis statement 2/a *I have formulated Constraint Programming models to generate minimal resource incompatible sets of tasks or task-mode pairs for single- or multi-mode RCPSPs.*

Thesis statement 2/b *I have developed branching methods for the S-graph based solution of the single- and multi-mode variants of the RCPSP, which can be used to partition the search space by resolving incompatible and infeasible sets.*

Thesis statement 2/c *I have extended the S-graph model with virtual tasks to represent resource disruptions at fixed time intervals.*

My publications related to the statement: [80] and presentations [74, 75]

Chapter 5

Scheduling a forge with die deterioration

The research presented in this chapter was motivated by a case study conducted at an axle manufacturer. I thank Balázs Ferenczi for showing us the manufacturing processes and providing data.

This research is aimed at scheduling a steel forge with the goal of minimizing setup and storage costs under strict deadlines and special resource constraints. The main distinctive feature of the problem is the deterioration of some equipment, in this case, the so-called forging dies. While the aging effect – where production speed decreases through time – has been widely investigated in the scheduling literature, durability, or lifespan deterioration caused by equipment setup has not yet been addressed to my knowledge.

Equipment deterioration is an interesting scheduling challenge, which was discovered while studying the operations of the forging process. The dies – used as templates for forming the axles into the desired shapes – are deteriorating not only when they are being used, but also with each changeover. Each die has a durability, measured in the number of axles it can process before needing a restoration. This lifespan can only be fully utilized if the die is used continuously. If the die is switched for another one, the remaining number of uses will decrease. This is due to the calibration process required on each setup of the die, and the physical deterioration from heating and cooling of the die. Even though it seems wasteful to suspense the production of an axle type until its die is completely deteriorated, sometimes it is necessary in order to meet deadlines, or it may simply be advantageous in reducing storage costs. There is a trade-off between minimizing the number of changeovers to decrease restoration costs, and minimizing the

inventory levels, storage costs and production delays.

I defined the identified scheduling problem, and formulated a discrete-time MILP model with the RTN representation method to solve this problem. I validated the approach and tested its performance on 3-week long scenarios generated based on real life data.

The chapter is structured as follows. Literature approaches dealing with similar scheduling problems are reviewed in Section 5.1. Section 5.2 contains a detailed description of the investigated scheduling problem. My proposed MILP model to solve this scheduling problem is presented in Section 5.3. In Section 5.4, an illustrative example is presented, and the solution efficiency of the proposed approach is measured on randomly generated instances as well.

5.1 Related literature

Even though the issue of equipment durability deterioration is present in other heavy industry production systems, approaches in scheduling literature have not integrated it into their models. Gascon and Leachman [32] proposed a dynamic programming approach for minimizing changeover and storage costs under deterministic demands. Despite the similar trade-off in their objective, the method can only be used for unit-sized batches and equal processing times. These assumptions do not hold for the forge scheduling problem, and the approach provides no way to model equipment deterioration.

Deterioration of production equipment was addressed in various papers in the literature [97, 109] but this deterioration, also called aging, has different effects. In their investigated problems, the production speed of the equipment was decreasing with usage, not their remaining number of uses, which is the case with forging dies. Zhao and Tang [109] presented an approach that also addressed the scheduling of restoration jobs reverting the aging of equipment. Their method only considers a single machine, while in the present problem, multiple dies must be scheduled.

The lack of research dealing with scheduling the operation and restoration of equipment whose durability deteriorates with usage and setup motivated this work. I proposed a general MILP model for optimizing the production schedules of systems with these characteristics. While the investigated problem deals with a specific forge used for axle production, the model is general enough to provide means for modeling other systems

with similar issues.

5.2 Problem definition

The investigated process transforms bare steel rods into axles of many different type, whose set is denoted by **A** in the formal definition. Each axle has to be manufactured from a single steel rod from a specific type (size). Different axle types, however, can share the same rod type. **R** is the set of all different rod types, and for each axle type $a \in \mathbf{A}$, its required rod type is denoted by r_a.

The steel rods are supplied by other companies, and the shipments are ordered based on a long-term stock management plan, which is outside of the scope of this research. Thus, the planned shipments are known and their set is denoted by **S**. Each shipment $s \in \mathbf{S}$ arriving at T_s^S may contain different rod types. The number of rods of type $r \in \mathbf{R}$ delivered by shipment $s \in \mathbf{S}$ is denoted by $Q_{s,r}^S$.

On the other end of the process, all orders $o \in \mathbf{O}$ must be satisfied by their delivery date T_o^O. Early shipment is not possible, as transportation is managed by the clients, and any delay would impose a serious risk of losing future contracts. An order may contain different axle types, and the required quantity is denoted by $Q_{o,a}^O$ for each axle type $a \in \mathbf{A}$ in order $o \in \mathbf{O}$.

The manufacturing process consists of 4 main consecutive stages with different characteristics and challenges:

1. Die forging (df)
2. Heat treatment (ht)
3. Preparation (p)
4. Machining (m)

5.2.1 Forging

The steel rods first go through the forging stage one by one: they are heated up, placed into a forging die and a hammering machine is forming them into the desired shape. This is a very expensive machine, so there is only one available, which makes it the bottleneck of the whole process. The scheduling planner has to decide when to switch from the forging of one axle type to another.

Each axle type has its dedicated die that has to be used on the appropriate rods. The

dies deteriorate only while being used. After a certain level of distortion, they require restoration, which takes t_a^{re} hours and costs C^{re} units.

The durability of a recently restored die can be measured by the number of steel rods that can be shaped with it. This number, $Q_a^{D,max}$ can vary between different axle types $a \in \mathbf{A}$, and is well known from historical data. The number of die-forged axles produced with a die between two restorations is, however, always lower, as each setup of a die requires heating and producing some trial-axles. These operations decrease the durability of the die by $Q_a^{D,su}$ every time it is installed into the forge. The setup of a die also takes t^{su} hours, and costs C^{su} cost units regardless of the selected die. In the beginning of the time horizon, it is assumed that each die for axle type $a \in \mathbf{A}$ has a deteriorated durability of $Q_a^{D,0}$, and none of them is currently installed in the forge.

After the setup, the time it takes to process each single steel rod is t_a^{df}, depending on the axle type $a \in \mathbf{A}$.

5.2.2 Heat treatment

A die-forged axle needs to cool down and get some grinding before it can undergo the heat treatment process. Since grinding does not need scarce resources, and can be done anytime after the cooldown and before the next stage, the time required for these two steps are merged into a single parameter, t_a^{cd} that depends on the axle type $a \in \mathbf{A}$.

The heat treatment furnace is regularly shut down for two reasons:

- Regardless of their type, the number of axles that can be treated in an hour, q^{ht}, is significantly higher than the best hourly throughput of the die forge, and keeping the furnace at high temperature requires a lot of energy, thus it is an unnecessary cost.
- Regular maintenance is mandatory for the furnace.

The time intervals when the furnace is heated up and is operational are provided by the long-term plan of the factory, which is based on historical data and industrial best practices. These intervals are given by $\mathbf{T}^F \subseteq [0, \infty[$.

Axles enter the furnace in batches, go through a corridor, then leave after a given time. The temperature and duration parameters of heat treatment in the furnace can be considered homogeneous over all axle types. There can be multiple batches inside the furnace up to its maximum capacity, therefore, heat treatment is considered as a continuous operation with a maximum flow-rate of q^{ht} pieces per hour.

5.2.3 Preparation and machining

During the preparation stage, a hydraulic press forms the product into its final geometry. Both this and the next, machining stage have high capacity and flexible flow-rate, thus, their resource needs are discarded, and only the time needed for the two steps for a single heat-treated axle is given by t_a^{pm} for each axle type $a \in \mathbf{A}$.

5.2.4 Flexibility, objective and cost evaluation

The mid-range planner can make two types of decisions:

- when and how long will a die be used in the forge
- when should the die-forged axles go to heat treatment

The planned schedule must meet the following set of constraints:

- a step can only be carried out if the inputs (rods for forging, forged axles for heat treatment) are available in the necessary quantity
- a die can not be used without restoration after its durability is depleted
- the orders have to be fulfilled without error

The goal is to minimize the cost over the planning period, that constitutes of:

- Production costs
- Die setup costs
- Die restoration costs
- Storage costs

Production costs are relative to the quantity, that is fixed by the orders, thus these costs are independent of the schedule. The hourly cost of forging is denoted by c^{df}. Setup and restoration costs are already discussed, these are axle-independent fixed costs that occur upon each setup or restoration of a die, and has the magnitudes of C^{su} and C^{re}, respectively.

There is a trade-off between storage and restoration costs. While actual storage costs are negligible, the loss in capital causes a hidden cost of unused, unnecessarily stored assets. So, a time- and piece-proportional storage cost is provided by the company to account for this. The parameters $c_r^{st,r}$, $c_a^{st,df}$, $c_a^{st,ht}$, and $c_a^{st,pm}$ denote these costs for the rods, die forged axles, heat treated axles, and delivery-ready axles, respectively. The storage capacity can be considered as unlimited, and the intermediates, products do not deteriorate over time.

The parameters are listed in the Nomenclature (5.6), along with the model variables and short descriptions.

5.3 Proposed MILP model

I developed a MILP model with a global discrete uniform time grid to solve the scheduling problem described above. This time representation has some advantages compared to continuous time models in handling the types of constraints present in this problem [25, 36]. One challenge is that it is not known in advance how many times a die will be changed before its restoration, and in what ratios will its lifespan be subdivided. This would make it very difficult to determine the number of time points in a continuous time formulation. Inventory costs, resource demands and arrivals are also easier to model in discrete time models.

Resources are modeled according to the RTN (Resource Task Network) formulation [83]. It provides a general representation of materials and production equipment, and it is suitable for modeling the durability of the dies and its associated constraints. This modeling method will be discussed in Section 5.3.3 in more detail.

5.3.1 Defining the discrete uniform time grid

To use a discrete uniform time grid, its resolution must be determined. The planning horizon is partitioned into equal-length time slots, indexed by τ. The length of each time slot, t^τ should be chosen based on the values of other timing parameters. In an ideal case, each parameter associated with a time is a multiple of t^τ. Otherwise, the solution space of the model may not include the optimal schedule. In the following, $\tilde{t}$ denotes the number of time slots a time parameter t is stretched over: $\tilde{t} = \lceil \frac{t}{t^\tau} \rceil$.

While shorter time slots can guarantee more precise solutions, having more time slots leads to more variables, and hence, more complex models. The end of the planning horizon is the latest delivery date among the orders, therefore, the number of required time slots is $n = \max_{o \in \mathbf{O}} \left\{ \lceil \frac{T_o^O}{t^\tau} \rceil \right\}$.

The steel rods arriving in shipments are available at the end of the respective time slot. To simplify the formulation, a computed parameter $Q_{\tau,r}^S$ is defined for the quantity of steel rod r available from the end of time slot τ, and at the beginning in the case $\tau = 0$.

$$Q^S_{\tau,r} = \sum_{\substack{s\in\mathbf{S}:\\ (\tau-1)t^\tau < T^S_s \le \tau\cdot t^\tau}} Q^S_{s,r} \quad \forall r \in \mathbf{R}, \tau = 0,\dots,n \tag{5.1}$$

Because of the flexibility of the preparation and machining steps, these operations do not need to be scheduled. Instead, the heat treated axles must be delivered before the delivery dates with leaving enough time for the last stages. A computed parameter $Q^O_{\tau,a}$ is introduced to represent the quantity of heat treated axles required by the end of time slot τ from axle type a.

$$Q^O_{\tau,a} = \sum_{\substack{o\in\mathbf{O}:(\tau-1)t^\tau < \\ T^O_o - t^{pm}_a\cdot Q^O_{o,a} \le \tau\cdot t^\tau}} Q^O_{o,a} \quad \forall a \in \mathbf{A}, \tau = 0,\dots,n \tag{5.2}$$

For the same reasons, storage costs of prepared and machined axles are omitted from the cost function, as the last stages should be executed as late as possible, assuming the storage costs of finished products are higher (otherwise, as early as possible).

5.3.2 Forging and heat treatment processes

The operations of the forge are modeled with 2 binary variables per pairs of time slot and die type. $x_{\tau,a}$ denotes whether die for axle type a is being used in time slot τ, and $x^{su}_{\tau,a}$ whether it is being setup. Constraint (5.1) ensures that the forge cannot work on more than one die at a time.

$$\sum_{a\in\mathbf{A}} \left(x_{\tau,a} + x^{su}_{\tau,a}\right) \le 1 \quad \forall \tau = 1,\dots,n \tag{5.1}$$

The necessary setup time is enforced by (5.2). A die can only be used in time slot τ if it was being setup in the previous required number of time slots, or if it was already in use. Consequently, the die cannot be in use in the first time slot (5.3).

$$\tilde{t}^{su}\cdot x_{\tau,a} \le \tilde{t}^{su}\cdot x_{\tau-1,a} + \sum_{\tau'=\max\{1,\tau-\tilde{t}^{su}\}}^{\tau-1} x^{su}_{\tau',a} \quad \forall a \in \mathbf{A}, \tau = 2,\dots,n \tag{5.2}$$

$$x_{1,a} = 0 \quad \forall a \in \mathbf{A} \tag{5.3}$$

Heat treatment is modeled with a continuous variable $y_{\tau,a}$, which denotes the number

of die-forged axles of type a heat treated during time slot τ.

$$y_{\tau,a} \geq 0 \quad \forall a \in \mathbf{A}, \tau = 1, \ldots, n \tag{5.4}$$

Multiple types of axles can be processed in the same time slot, only their total quantity is constrained by q^{ht}, the throughput ratio of the furnace. Constraint (5.5) shows how the furnace capacity can be calculated from its set of active time periods, $\mathbf{T}^F$. It is assumed that $\mathbf{T}^F$ is a union of continuous time intervals.

$$\sum_{a \in \mathbf{A}} y_{\tau,a} \leq \int_{[(\tau-1)t^\tau, \tau \cdot t^\tau] \cap \mathbf{T}^F} q^{ht} \quad \forall \tau = 1, \ldots, n \tag{5.5}$$

Die-forged intermediates require t_a^{cd} hours for cooling and grinding before their heat treatment. Constraints (5.6)-(5.7) ensure that the axles going into the furnace in time slot τ were forged long enough ago. Variable $r_{\tau,a}$ will be described in Subsection 5.3.3.

$$y_{\tau,a} \leq r_{\tau - \tilde{t}_a^{cd}, a} \quad \forall a \in \mathbf{A}, \tau = \tilde{t}_a^{cd} + 1, \ldots, n \tag{5.6}$$

$$y_{\tau,a} = 0 \quad \forall a \in \mathbf{A}, \tau = 1, \ldots, \tilde{t}_a^{cd} \tag{5.7}$$

5.3.3 Material balance

In an RTN representation, both materials and available equipment are treated equally as resources that can be consumed and produced by the processes. The resource levels at the end of each time slot are represented by nonnegative continuous variables. $r_{\tau,r}^R, r_{\tau,a}^{df}$, and $r_{\tau,a}^{ht}$ denote the resource levels of steel rods, die-forged, and heat treated axles, respectively. The durabilities of dies are represented similarly by variables $r_{\tau,a}^D$.

Starting resource levels are set by the shipments arriving at 0 time unit, and by $Q_a^{D,0}$ for die durabilities.

$$r_{0,r}^R = Q_{0,r}^S \qquad \forall r \in \mathbf{R} \tag{5.1}$$

$$r_{0,a}^{df} = 0 \qquad \forall a \in \mathbf{A} \tag{5.2}$$

$$r_{0,a}^D = Q_a^{D,0} \qquad \forall a \in \mathbf{A} \tag{5.3}$$

Material levels at the end of time slots are calculated from the previous levels and the

changes caused by production processes (5.4)-(5.5).

$$r^R_{\tau,r} = r^R_{\tau-1,r} + Q^S_{\tau,r} - \sum_{a\in\mathbf{A}:r_a=r} \left(\frac{Q^{D,su}_a}{\tilde{t}^{su}} x^{su}_{\tau,a} + \frac{t^\tau}{t^{df}_a} x_{\tau,a} \right) \quad \forall r \in \mathbf{R}, \tau = 1,\dots,n \tag{5.4}$$

$$r^{df}_{\tau,a} = r^{df}_{\tau-1,a} - Q^O_{\tau,a} - \frac{Q^{D,su}_a}{\tilde{t}^{su}} x^{su}_{\tau,a} - \frac{t^\tau}{t^{df}_a} x_{\tau,a} \quad \forall a \in \mathbf{A}, \tau = 1,\dots,n \tag{5.5}$$

Die durabilities are calculated similarly but restorations make it more complex, as the regained amount of durability depends on the current durability, which would introduce a non-linear term into the equation, as shown in (5.6). Restoration of die for axle a is represented by the binary variable $w_{\tau,a}$, which equals to 1 if the restoration is finished upon the end of time slot τ.

$$r^D_{\tau,a} = r^D_{\tau-1,a} - (1 - w_{\tau,a}) \left(\frac{Q^{D,su}_a}{\tilde{t}^{su}} x^{su}_{\tau,a} + \frac{t^\tau}{t^{df}_a} x_{\tau,a} \right) + w_{\tau,a} \cdot Q^{D,max}_a \quad \forall a \in \mathbf{A}, \tau = 1,\dots,n \tag{5.6}$$

This constraint is linearized with the following inequalities (5.7)-(5.9), replacing (5.6) in the model. The lower bound (5.7) is similar to the material balance equations. The upper bound (5.8) includes the restored durability as a big-M term, and a constant upper bound (5.9) limits the durability to its maximum to prevent exceeding it with an early restoration.

$$r^D_{\tau,a} \geq r^D_{\tau-1,a} - \frac{Q^{D,su}_a}{\tilde{t}^{su}} x^{su}_{\tau,a} - \frac{t^\tau}{t^{df}_a} x_{\tau,a} \quad \forall a \in \mathbf{A}, \tau = 1,\dots,n \tag{5.7}$$

$$r^D_{\tau,a} \leq r^D_{\tau-1,a} - \frac{Q^{D,su}_a}{\tilde{t}^{su}} x^{su}_{\tau,a} - \frac{t^\tau}{t^{df}_a} x_{\tau,a} + w_{\tau,a} \cdot Q^{D,max}_a \quad \forall a \in \mathbf{A}, \tau = 1,\dots,n \tag{5.8}$$

$$r^D_{\tau,a} \leq Q^{D,max}_a \quad \forall a \in \mathbf{A}, \tau = 1,\dots,n \tag{5.9}$$

The restoration time is enforced in Constraint (5.10) similarly as the setup time in (5.2). While a die is being restored, it cannot be in use, therefore, a restoration can finish only after enough idle time of the die.

$$w_{\tau,a} \cdot \tilde{t}^{re}_a \leq \tilde{t}^{re}_a - \sum_{\tau'=\tau-\tilde{t}^{re}_a+1}^{\tau} \left(x_{\tau',a} + x^{su}_{\tau',a} \right) \quad \forall a \in \mathbf{A}, \tau = \tilde{t}^{re}_a + 2,\dots,n \tag{5.10}$$

$$w_{\tau,a} = 0 \quad \forall a \in \mathbf{A}, \tau = 1,\dots,\tilde{t}^{re}_a \tag{5.11}$$

5.3.4 Objective

The goal is to minimize the total cost of the production. As (5.1) shows, with the discrete time grid, costs can be calculated easily by multiplying the cost factors with their associated variables, and summing the costs imposed in each time slot.

$$\begin{aligned}\min z = &\sum_{\tau=1}^{n}\sum_{r\in\mathbf{R}} c_r^{st,r}\cdot r_{\tau,r}^{R} + \\ &\sum_{\tau=1}^{n}\sum_{a\in\mathbf{A}}\left(c_a^{st,df}\cdot r_{\tau,a}^{df} + c_a^{st,ht}\cdot r_{\tau,a}^{ht} + c^{df}\cdot t^{\tau}\cdot x_{\tau,a} + \frac{C^{su}}{\tilde{t}^{su}}\cdot x_{\tau,a}^{su} + C^{re}\cdot w_{\tau,a}\right)\end{aligned} \tag{5.1}$$

5.3.5 Model improvements

The previous constraints are enough to model the scheduling problem and obtain the optimal schedule. However, there is room to improve solution quality and performance.

[26] summarized the possible techniques for accelerating the solution process of MILP models. Here, improvement of the model is attempted by adding tightening constraints to decrease the search space.

The inequalities introduced in (5.7)-(5.9) to model the durability changes of dies allow solutions where a die is not fully restored to $Q_a^{D,max}$. There is no benefit in partially restoring a die, unless it has less cost or time requirement, which would require a different model. Apart from the practical reasons, this may also make it more difficult to solve the model. To enforce that dies are always fully restored, Constraint (5.1) is added to the model.

$$r_{\tau,a}^{D} \geq w_{\tau,a}\cdot Q_a^{D,max} \quad \forall a\in\mathbf{A}, \tau = 1,\ldots,n \tag{5.1}$$

Another simplification can be made for scheduling die restoration. It can be observed that the restoration can be carried out anytime between two uses of a die with the timing having no effect on the objective value. Therefore, forcing the restoration to end directly before it is used again with Constraint (5.2) will tighten the solution space without loss of optimality.

$$w_{\tau-1,a} \leq x_{\tau,a}^{su} \quad \forall a\in\mathbf{A}, \tau = 2,\ldots,n \tag{5.2}$$

Further practical considerations can be made about allowing a die to be restored even if it still has enough durability to be used further. The unused durabilities are wasted, which could be handled by introducing a cost parameter for it, or a constraint like (5.3) can be added to the model that prevents restoration if the remaining durability is over a

given minimum. In (5.3), the minimum is computed to be the durability used up during a whole time slot.

$$w_{\tau,a} \leq 1 + \frac{\frac{t^{\tau}}{t_a^{df}} - r^{D}_{\tau-1,a}}{Q_a^{D,max} - \frac{t^{\tau}}{t_a^{df}}} \quad \forall a \in \mathbf{A}, \tau = 2, \ldots, n \tag{5.3}$$

Adding this constraint may remove the optimal solution of the original model from the solution space. However, it eliminates the waste caused by not utilizing the full durabilities of dies. A better solution would be to add the associated costs of this wasted capacity to the objective function, but determining the cost factor is difficult, and it would make the model more complex.

The original model also allows to start setting up a die but not using it after setup. Solutions like this would be suboptimal because of the added costs of the unnecessary setup, so they can be eliminated from the search space without loss of optimality. Constraint (5.4) solves this by ensuring that if a setup is started in time slot τ, it is continued until setup is finished, then forging must be started.

$$(x^{su}_{\tau,a} - x^{su}_{\tau-1,a})(\tilde{t}^{su} + 1) \leq x_{\tau+\tilde{t}^{su},a} + \sum_{\tau'=\tau}^{\tau+\tilde{t}^{su}-1} x^{su}_{\tau',a} \quad \forall a \in \mathbf{A}, \tau = 2, \ldots, n - \tilde{t}^{su} \tag{5.4}$$

5.4 Computational results

The proposed approach was tested on several test cases. The parameters were chosen based on real data provided by the industrial partner.

First, a detailed example is shown with all input parameters and the obtained optimal solution. Then, the results of further empirical analysis are presented based on randomly generated instances.

For MILP optimization, Gurobi 8.0 was used on a computer with a dual-core Intel i5-660 (3.33 GHz) CPU and 8 GB RAM. A time limit of 1000 s was set for the solver.

The basic model consists of the Constraints (5.1)-(5.5), and (5.7)-(5.1). This can be extended with any combination of the Constraints (5.1)-(5.4).

5.4.1 Illustrative example

In this example problem, 3 weeks need to be scheduled. The length of the time slots is chosen to be 8 hours. There are 4 axle types to be produced ($\mathbf{A} = \{a1, a2, a3, a4\}$),

and they are forged from 3 rod types ($\mathbf{R} = \{r1 = r_{a1}, r2 = r_{a2}, r3 = r_{a3} = r_{a4}\}$). Most supplies are available from the start, and 1 shipment is arriving later (Table 5.1).

There are product orders to be satisfied at the end of each week (Table 5.2). Other input parameters are shown in Tables 5.3-5.5.

Table 5.1: Supply shipments

s	T_s^S	$Q_{s,r1}^S$	$Q_{s,r2}^S$	$Q_{s,r3}^S$
1	0	2500	2000	3000
2	168	0	0	4000

Table 5.2: Product orders

o	T_o^O	$Q_{o,a1}^O$	$Q_{o,a2}^O$	$Q_{o,a3}^O$	$Q_{o,a4}^O$
1	168	410	320	460	0
2	336	730	350	1440	840
3	504	600	480	1650	1120

Table 5.3: Cost parameters

Parameter	Value
$c_r^{st,r}$ [cu/h]	r1: 0.000133, r2: 0.000104, r3: 0.000085
$c_a^{st,df}$ [cu/h]	a1: 0.000213, a2: 0.000201, a3: 0.000190, a4: 0.000196
$c_a^{st,ht}$ [cu/h]	a1: 0.000321, a2: 0.000307, a3: 0.000279, a4: 0.000292
C^{su} [cu]	800
C^{re} [cu]	2000

Table 5.4: Axle-dependent parameters

Parameter	a1	a2	a3	a4
t_a^{df} [h]	1/20	1/15	1/25	1/22
t_a^{cd} [h]	4	4	4	4
t_a^{pm} [h]	1	1	1	1
t_a^{re} [h]	48	48	48	48
$Q_a^{D,max}$ [pcs]	2000	1800	2500	2300
$Q_a^{D,0}$ [pcs]	2000	1800	2500	2300
$Q_a^{D,su}$ [pcs]	160	120	200	176

Table 5.5: Other parameters

Parameter	Value
τ [h]	8
t^{su} [h]	8
q^{ht} [pcs/h]	72
$\mathbf{T}^F$	$\bigcup_{k=0}^{\infty} [0 + 0k, 54 + 70k]$

The example was solved with the basic model and with each possible combination of the last 4 constraints. Each variant obtained the same optimal solution with an objective value of 8080.432048 cost units, in less than 5 minutes.

Figure 5.1 shows the Gantt chart of the forge, displaying when the dies are being used, setup, or restored. It can be seen that dies are switched out several times in order to meet the delivery dates. The resource levels are displayed in Figure 5.2, where order deliveries (of heat treated intermediates), the supply shipment, and die durabilities can be seen.

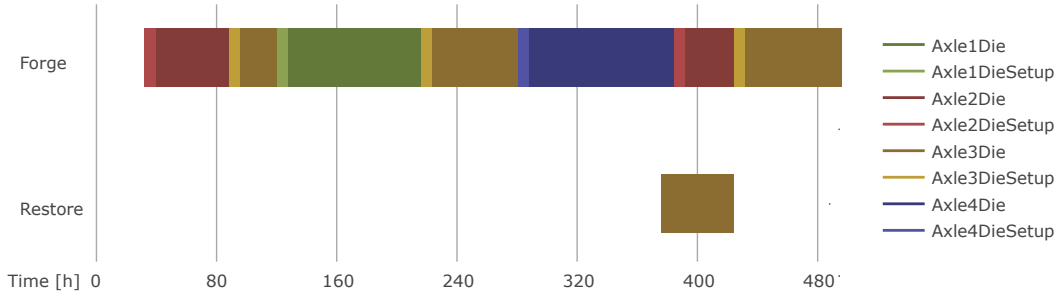


Figure 5.1: Optimal schedule of the forging dies

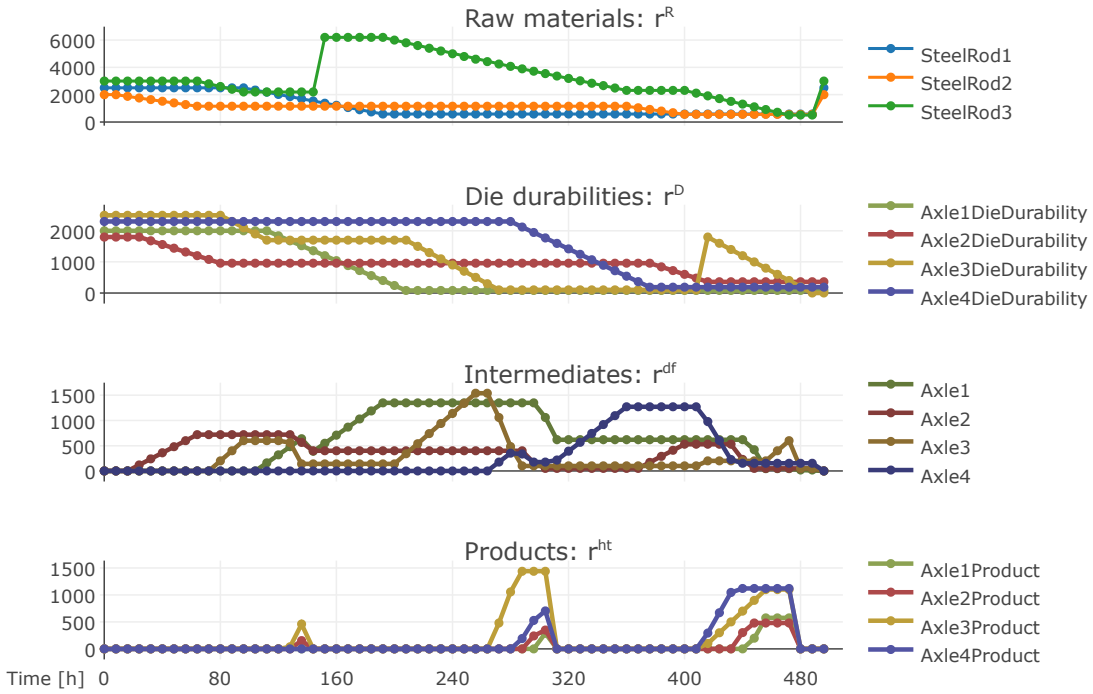


Figure 5.2: Resource levels of the optimal schedule

5.4.2 Performance analysis

To gain more insight about the effects of the 4 optional constraints on the performance, they were tested on randomly generated instances.

20 instances were generated, each contains 4 products with a different rod used for each. Raw materials are all present at the start. The planning horizon is 3 weeks, with product orders due by the end of each week. Tables 5.6 and 5.7 show the intervals, from which the parameter values were chosen randomly with uniform distribution. Values associated with timing or quantity were chosen from integer intervals.

From the 20 instances 11 were infeasible, which was reported by the solver in under 15 s on each such instance. 1 instance was significantly more difficult to solve than others, where none of the model variants could prove optimality in the 1000 s time limit, with

Table 5.6: Parameter intervals for orders and starting supplies

o	T^O_o	$Q^O_{o,a1}$	$Q^O_{o,a2}$	$Q^O_{o,a3}$	$Q^O_{o,a4}$
1	168	0	[500 .. 1500]	[500 .. 1500]	[500 .. 1500]
2	336	[500 .. 1500]	0	[500 .. 1500]	[500 .. 1500]
3	504	[500 .. 1500]	[500 .. 1500]	0	[500 .. 1500]
s	T^S_s	$Q^S_{s,r1}$	$Q^S_{s,r2}$	$Q^S_{s,r3}$	$Q^S_{s,r4}$
1	0	[4000 .. 6000]	[4000 .. 6000]	[4000 .. 6000]	[4000 .. 6000]

Table 5.7: Intervals of other parameters

Parameter	Interval
$c^{st,r}_r$ [cu/h]	[0.00008, 0.00015]
$c^{st,df}_a$ [cu/h]	[0.00020, 0.00030]
$c^{st,ht}_a$ [cu/h]	[0.00030, 0.00040]
C^{su} [cu]	[600, 1000]
C^{re} [cu]	[1500, 2000]
$1/t^{df}_a$ [pcs/h]	[20 .. 30]
t^{su} [h]	8
t^{cd}_a [h]	4
t^{pm}_a [h]	1
t^{re}_a [h]	48
$Q^{D,max}_a$ [pcs]	[1800 .. 2500]
$Q^{D,0}_a$ [pcs]	$Q^{D,max}_a$
$Q^{D,su}_a$ [pcs]	[100 .. 200]
τ [h]	8
q^{ht} [pcs/h]	72
$\mathbf{T}^F$	$\bigcup_{k=0}^{\infty} [0+0k, 54+70k]$

the best gap among them being 20%.

The solution times of the remaining 8 instances are shown in Table 5.8. The 4 bits in the first column represent whether each of the last 4 constraints are included in that model variant. The rows are ordered first by the number of instances solved to optimality (Opt), then by their average solution times (Avg). The best solution time for each instance is highlighted in bold.

From the 4 constraints, (5.3) had the largest impact on solution time. Adding this constraint to the model resulted in higher solution times. It also increased the objective values by 10-15%, as this constraint does not allow restoration of a die that still has durability for more forging.

The other 3 constraints do not affect the optimal objective value, their purpose is to

Table 5.8: Solution times (CPU s) of different model variants

Variant (20-23)	Instance 3	4	7	8	9	11	13	18	Opt	Avg
1001	626.84	**103.91**	469.95	18.91	506.02	190.36	30.89	**145.49**	8	343.60
0100	837.79	148.71	246.36	26.55	426.94	267.40	27.89	278.86	8	362.28
1101	866.91	317.47	**169.28**	12.13	412.26	241.16	27.86	333.82	8	375.65
1100	990.01	292.54	520.21	**7.49**	**175.92**	**91.90**	69.68	481.47	8	403.25
1000	**569.78**	133.22	209.09	24.08	–	138.55	59.57	161.47	7	366.20
0101	569.96	425.12	567.54	23.19	309.59	333.47	43.29	–	7	474.68
1110	–	711.86	294.65	52.35	693.62	515.32	**11.06**	233.57	7	501.38
0001	725.71	300.85	513.43	18.98	–	197.56	60.04	–	6	535.17
1111	–	675.92	286.33	49.54	–	472.78	37.42	677.56	6	577.73
0000	–	154.79	737.37	100.67	–	233.75	48.29	–	5	586.10
0110	–	–	213.17	134.01	–	593.66	39.33	615.16	5	621.70
0111	986.67	–	707.98	50.73	–	–	17.36	528.99	5	699.08
1010	–	762.81	–	94.62	–	–	28.54	821.13	4	745.23
1011	–	835.25	982.05	95.70	–	–	22.70	–	4	770.63
0011	–	802.83	–	269.15	–	–	26.00	–	3	788.66
0010	–	–	–	140.99	–	–	31.94	–	2	796.99
Avg	885.85	541.58	557.34	69.94	782.77	517.24	36.37	642.35		

– : Terminated after 1000 s.

reduce the solution space by eliminating suboptimal and redundant solutions. There is a high variance in the effects of using them but adding any of them to the model was better than using none of them. The impacts of individual constraints also varies among the instances, neither constraint dominates the others in performance.

Based on the test results, the developed model is a successful first attempt in solving the newly described problem. To solve larger, more complex examples efficiently, new improvement methods or heuristic approaches need to be developed, which opens potential for future research on the topic.

5.5 Summarizing statements

Thesis statement 3 *I have identified and defined a scheduling problem with novel features at an axle-manufacturing system, and developed a discrete-time MILP model that can solve it.*

Thesis statement 3/a *I proposed 4 additional constraints to tighten the formulation, and compared their impacts on solution performance with randomly generated test cases.*

My publications related to the statement: [78] and presentation [76]

5.6 Nomenclature

Sets

$\mathbf{R}$	Set of all raw materials, i.e., rod types
$\mathbf{A}$	Set of axle types
$\mathbf{O}$	Set of orders
$\mathbf{S}$	Set of supply shipments

Symbolic parameters

$r_a \in \mathbf{R}$ the required rod type for axle type $a \in \mathbf{A}$

Timing related parameters and intervals

T_s^S **[h]** the expected arrival time of shipment $s \in \mathbf{S}$

T_o^O **[h]** the delivery date of order $o \in \mathbf{O}$

t^{su} **[h]** the setup time needed for the forge before starting to work on a new series of axles

$\tilde{t}^{su} = \lceil \frac{t^{su}}{t^\tau} \rceil$ the setup time converted to number of time slots

t_a^{df} **[h]** the time needed to die-forge an axle from type $a \in \mathbf{A}$

t_a^{cd} **[h]** the time needed for a forged axle $a \in \mathbf{A}$ to cool down before the heat treatment

$\tilde{t}_a^{cd}$ **[h]** the cooldown time converted to number of time slots

t_a^{pm} **[h]** the time needed to prepare and machine a heat-treated axle $a \in \mathbf{A}$

t_a^{re} **[h]** the time required for the restoration of the deteriorated die for an axle $a \in Axles$

$\tilde{t}_a^{re} = \lceil \frac{t_a^{re}}{t^\tau} \rceil$ the restoration time converted to number of time slots

$\mathbf{T}^F \subseteq [0, \infty[$ the time intervals when the furnace is working

t^τ **[h]** the length of each time slot

n the number of time slots

Material quantity related parameters

$Q_{s,r}^S$ **[pcs]** the quantity of rod $r \in \mathbf{R}$ supplied by shipment $s \in \mathbf{S}$

$Q_{\tau,r}^S$ **[pcs]** the quantity of rod $r \in \mathbf{R}$ arriving during time slot τ

$Q_{o,a}^O$ **[pcs]** the quantity of finished axle $a \in \mathbf{A}$ to be shipped for order $o \in \mathbf{O}$

$Q_{\tau,r}^O$ **[pcs]** the quantity of heat treated axle $a \in \mathbf{A}$ required to be completed at the end of time slot τ

$Q_a^{D,max}$ **[pcs]** the maximal durability of the die for axle type $a \in \mathbf{A}$ measured in number of rods processed

$Q_a^{D,0}$ **[pcs]** the starting durability of the die for axle $a \in \mathbf{A}$ in the beginning of the time horizon

$Q_a^{D,su}$ **[pcs]** the number of rods wasted during the setup phase for the axle type $a \in \mathbf{A}$

q^{ht} **[pcs/h]** the number of axles (of any kind) that can go under heat treatment hourly

Cost related parameters

C^{su} **[cu]** the cost of a complete forge setup

$c_r^{st,r}$ **[cu/h/pcs]** the hourly storage cost for rods with type $r \in \mathbf{R}$

$c_a^{st,df}$ **[cu/h/pcs]** the hourly storage cost for die forged axles of type $a \in \mathbf{A}$

$c_a^{st,ht}$ **[cu/h/pcs]** the hourly storage cost for heat treated axles of type $a \in \mathbf{A}$

$c_a^{st,pm}$ **[cu/h/pcs]** the hourly storage cost for prepared and machined axles of type $a \in \mathbf{A}$

c^{df} **[cu/h]** the hourly cost of forging

C^{re} **[cu]** the cost of restoring any type of die to its maximum durability

Variables

$x_{\tau,a}$ binary variable, denotes whether the die for axle type a is being used in time slot τ

$x_{\tau,a}^{su}$ binary variable, denotes whether the die for axle type a is being setup in time slot τ

$y_{\tau,a} \geq 0$ denotes the number of axles from type a going through heat treatment in time slot τ

$w_{\tau,a}$ binary variable, denotes whether the restoration of the die for axle type a is finished at the end of time slot τ

$r_{\tau,r}^{R}$ the number of steel rods on hold at the end of time slot τ

$r_{\tau,a}^{df}$ the number of die-forged axles on hold at the end of time slot τ

$r_{\tau,a}^{ht}$ the number of heat treated axles on hold at the end of time slot τ

$r_{\tau,a}^{D}$ the remaining durability of the die for axle a at the end of time slot τ

Chapter 6

S-graph approach for minimizing freshwater usage

In this chapter, I present my extension of the S-graph framework which solves the simultaneous scheduling and water minimization of batch processing systems. The proposed approach tackles truly batch processes with a single contaminant, and allows only a single water source to be reused for each sink. The presented algorithm and model extension has been implemented and tested on various case-studies from the literature. This development provides an opportunity for further extensions to address wider range of problems, such as having multiple contaminants, semi-continuous behavior, or cyclic operations.

Water is one of the most important natural resources of life. While it is considerably cheap and vastly available currently, except for extreme locations, this is not guaranteed in the future. Being provident with water has multitude of advantages in both short- and long term. Using less clean water not only brings immediate financial benefits, it simultaneously reduces wastewater production, related treatment costs, and the impact on the environment. Reducing the water footprint of a batch system is not a trivial task, as water sources and sinks need to be matched not only in amount and quality, but in time as well.

The chapter is structured as follows. In Section 6.1, the investigated problem and its features are defined. Existing literature methods for water minimization and integrating it with scheduling are summarized in Section 6.2. My extension to the S-graph framework for the investigated problem is presented in Section 6.3. Then Section 6.4 presents the validation tests of the proposed method on literature examples.

6.1 Problem definition

The objective of the investigated problem class is to simultaneously minimize fresh water usage and wastewater production by maximizing the reuse of process water. The non-water related features (task dependencies, heterogeneous machines, etc.) of the scheduling problem can be handled by the equipment-based branching (EQB) algorithm of the S-graph framework, which was presented in Chapter 2.

Apart from raw materials and intermediates, each task requires a certain amount of water (m_i^{in}), that has to meet a predefined quality criteria, i.e., the level of a single addressed contaminant has to stay below the maximal allowed level (c_i^{max}) in the provided water. After the task is executed, it provides a given amount of water (m_i^{out}), whose quality is assumed to be the fixed value (c_i) belonging to the worst case scenario, regardless of the input contamination level. It is assumed that each task is executed in a fully batch fashion, i.e., all of its raw materials, intermediates and input water are consumed at once at the beginning of its execution, and all of the intermediates, products, output water are produced at the end of its execution.

While clean water is assumed to be an unlimited resource, tasks may reuse the output water of other tasks partially or completely. Splitting of output water streams is allowed, i.e., the output of a task may be used as the input for several different tasks. However, a task is only allowed to receive water from at most one other task, whose output may be mixed with clean water, if the volume or quality dictates so.

Transfers of intermediates and water are assumed to take a negligible amount of time. Dedicated storage may or may not be available for intermediates, however, it is assumed, that water storages are readily available for as many different contamination levels as required, with sufficient capacity. Mixing of output water flows is not allowed with each other, only with clean water at the input of another task.

All tasks have to be carried out by one of the suitable units, and the products must be produced before the predefined global deadline. Within this time horizon, tasks may be scheduled freely, as long as production dependencies are met, and the products are finished before the deadline. Naturally, units can process only one task at a time, and water can only be reused from a task that finishes the latest at the starting time of its destination task.

6.2 Literature summary

Environment-friendly operation of the production industry has gained major focus over the last few decades in the literature. Sustainable production has become more and more prevalent, and a lot of research effort were made to reduce environmental impact by developing new technologies, integrating different planning phases, exploring a wider operational spectrum, etc. Apart from other indicators, freshwater consumption has gained decent interest in the literature, and dedicated sessions and even conferences to tackle this issue.

While in most developed countries, clean water is currently a relatively cheap and mostly available resource, 33 percent of the population do not have any access to safe drinking water [103]. Only 0.5 percent of world's water is both drinkable and accessible, and has to satisfy the water needs of the worlds population and its ecosystem [100, SDG 12]. Moreover, global water demand has been steadily increasing about 1 percent per year since 1980 and is projected to increase by 55 percent by 2050 [101]. Even some developed countries have water shortage problems, see the increasing severity of droughts and unsustainable irrigation usage of the Colorado River [21], or the extreme shrinking of the Aral Sea [33].

A prognosed 40 percent shortfall in freshwater resources by 2030 drives the world towards a global water crisis [100, SDG 6]. This shortage is expected to have a significant effect on the industry as well. If unsustainable pressures are put on global water resources and the natural environment continues to be degraded, an estimated 45 percent of the global gross domestic product will be at risk by 2050 [100, SDG 6].

Thus, preparing production processes to operate in a water preserving manner is of utmost importance in both delaying or avoiding a water shortage crisis, and to accommodate for one if it is unavoidable. This battle can be fought on many fronts with developing less water intensive technologies, improving water treatment and conservation options, etc. [100, SDG 6, 12]. A promising direction is the development of improved management for wastewater resources. Today, 80 percent of all wastewater is released to the environment without any treatment [102]. Improved treatment technologies, and reuse can significantly ease the pressure on the environment.

Early approaches in water minimization focused on continuous processes. These techniques have been well established and may be found in reviews [8, 28] and textbooks [30, 95]. On the other hand, water minimization for batch processes has gained relatively less

attention, partly due to the discrete nature of time dimension which makes the modeling of batch processes more complex.

Early approaches on water minimization are mostly based on predetermined fixed schedule, water usage is optimized in a subsequent step. The developed techniques based on this fixed schedule-type approach can be broadly categorized as pinch analysis-based and mathematical programming. In the former, graphical [105, 65] or algebraic targeting tools [29] have been developed to identify water flow targets ahead of detailed design. The main limitation of the pinch analysis-based techniques is that they are bound to handle single contaminant cases. This limitation can be handled conveniently using mathematical programming techniques [7, 6, 49, 60]. Note however that these earlier works mainly developed based on pre-determined fixed schedule. A review that summarized the state-of-the-art water minimization problem for batch processes up to last decade may be found in [35]. The latter also outlined that simultaneous water minimization and scheduling will be the next trend of development for research in this area, as they can achieve more efficient water integration than the fixed schedule techniques.

The method proposed by [20] is a non-linear (MINLP) model with equidistant, discrete time points. The model covers a wide range of constraints, including water treatment operations, and the objective function combines the minimization of both wastewater generation and annual operating costs. However, the non-linear formulation and the large number of integer variables that can result from the discrete time model makes its application limited to relatively small instances.

The MILP model proposed by [64] uses a continuous time representation and the state sequence network (SSN) methodology to represent tasks as state transitions. Additionally to the problem class defined in Section 6.1, the model by [64] can handle variable batch sizes with equipment capacities, splitting tasks among multiple units, storage vessel for water, and reusing water from multiple sources in one task. The methodology has been further improved in the following years, for example, [71] extended the approach for long-term scheduling scenarios with cyclic operations. [18] formulated a similar model, and included multiple freshwater streams with varying contamination and cost. [91] extended the formulation to simultaneously optimize energy usage and it also handles multiple contaminants.

The approach proposed by [59] aims to combine the advantages of the insight-based water minimization methods and the MILP-based scheduling models. The model consists

of two parts: water contamination levels are modeled based on the automated targeting model (ATM) developed by [27], and a discrete time scheduling model. The approach can model systems with separate, limited storage vessels for each contamination level, but the task-unit assignment must be fixed in the problem input.

In recent papers, Li and Majozi proposed a dynamic programming based approach [62, 61] for simultaneous scheduling and water minimization, with wastewater regeneration.

6.3 Proposed approach

The aim of this work is to extend the S-graph framework for minimizing wastewater production by water reuse. While the investigated problem class considers a simple form of water treatment model that has been tackled by other approaches from the literature, it also comprises a general and detailed scheduling side. The S-graph framework can effortlessly address scheduling-related parameters and restrictions such as general precedential recipes, unit-specific processing times, changeover times and costs, or limitations on storage time for intermediate products. An additional advantage of any S-graph based approach is the ability to easily report several solutions. These can be, for example, the n best solutions, the pareto-optimal ones if two objectives are given, the solutions with objective values within a certain range of the optimal one, or even all feasible solutions.

6.3.1 New branching method

My proposed approach for solving the problem defined in Section 6.1 is a modification of the EQB algorithm detailed in Chapter 2.

As a reminder, these are the new input parameters of the problem, aside from the ones already covered in the S-graph framework:

m_i^{in} Required input water mass of task i

m_i^{out} Output water mass of task i

c_i^{max} Maximum allowed input contamination of task i

c_i Output contamination of task i

To model water flow decisions, a set of triplets is stored at each search node: $F \subset N_t \times N_t \times \mathbb{R}$. A triplet (s, d, f) means that f units of water are transferred from the output of task s to the input of task d. The remaining water output of task s is calculated as $w_s = m_s^{out} - \sum_{(s,d,f)\in F} f$.

The original bound function calculates the lower bound of the makespan. Here, this is not the objective but used as a feasibility test to check against the allowed time horizon. The new bound function returns the total unused water output ($\sum_{i \in N_t} w_i$) if the makespan is feasible, or ∞, if infeasible.

The new branching method works in the following way:

1. Generate child nodes with the EQB method. These nodes represent the cases when the newly scheduled task only uses clean water.
2. For each of these nodes ($G(N, A_1 \cup A_2)$, c, $\{S_i \ \forall i \in N_t\}$, SOUN, last_node, F), generate sibling nodes with alternative water transfers into the scheduled task d:
   - For all possible water sources ($\forall s \in N_t \setminus \{d\} : w_s > 0$) that are not scheduled after d (there is no directed path in A from d to s), generate alternative nodes with water reuse:
     - Calculate how much of the source water can be reused:

$$f = \min \left\{ w_s, m_d^{in}, \frac{c_d^{max}}{c_s} m_d^{in} \right\} \tag{6.1}$$

     - If $f \neq 0$, create an alternative node ($G'(N, A_1 \cup A_2')$, c', $\{S_i \ \forall i \in N_t\}$, SOUN, last_node, $F \cup \{(s, d, f)\}$) where a schedule-arc is inserted from s to d: $A_2' = A_2 \cup \{(s, d)\}, c'(s, d) = \min_{j \in S_s} \{pt_{s,j}\}$
3. Return the set of child nodes generated above.

The proposed approach serves as a basis for several future extensions. Addressing multiple contaminants can easily be achieved by extending Equation (6.1) with a concentration limit for each contaminant. If the set of contaminants is denoted by K, the equation becomes:

$$f = \min \left\{ w_s, m_d^{in}, \min_{k \in K} \left\{ \frac{c_{d,k}^{max}}{c_{s,k}} m_d^{in} \right\} \right\} \tag{6.2}$$

Cyclic scheduling can also be addressed with some modifications to the S-graph model and the bounding function. A simple cyclic solution is presented in Example 1 of Section 6.4, however a general cyclic solution with S-graphs has not been developed yet.

The solution method can be extended to allow mixing of different water sources, by replacing the bounding function with a linear programming model. Modeling semicontinuous operations is also a potential topic for future research. While these changes require additional research, they are suitable extensions for the S-graph. Introducing a shared water storage, however, is something that seems rather challenging to do in a natural way, and poses a difficult task to overcome in future research.

6.3.2 Demonstrative example

The solution approach is demonstrated on the case study by [59], with the tasks modified to be truly batch. This example entails two products, A and B, that has to be produced through 2 consecutive steps, each on 4 dedicated units within a 5 hour time horizon. Because of the dedicated units, no assignment or sequencing decisions need to be made, which renders this example adequate to better highlight the novel extensions of the algorithm. The recipe-graph for the example is shown in Figure 6.1.

For each task, the water requirement and maximal contamination level is indicated in the top pocket, e.g., A1 requires 20 tons of clean water, while B2 needs 80 tons of water whose contamination level is below 50 ppm. At the bottom, the amount and quality of the output water is indicated. For example B1 produces 20 tons of water with 800 ppm contaminant.

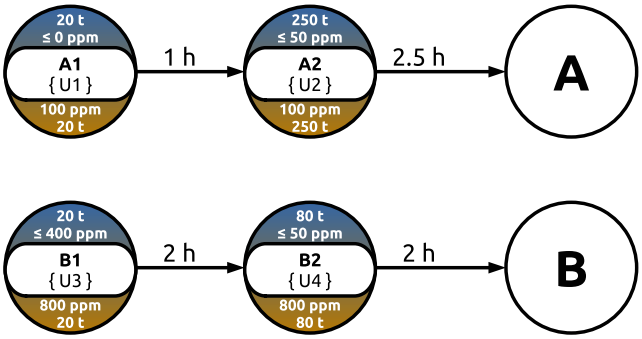


Figure 6.1: Recipe graph of case study by [59] (node 0)

In the trivial solution, where all of the water requirements are satisfied with clean water, the freshwater usage (and hence the wastewater production) is 370 tons (20 + 250 + 20 + 80).

The proposed S-graph algorithm starts from this recipe graph (G0), and explores all possible schedules considering water reuses as well. The original EQB method selects an equipment unit, let this be U1 now. The only task that can be assigned to U1 is A1, so a single child node is generated by the EQB. Then, the alternative water assignments would be generated. However, A1 requires clean water, so there is no option to reuse water here, thus, only one child node is returned, where all of its water requirement is supplied with freshwater. The S-graph of this node is shown in Figure 6.2, which is very similar to the recipe graph. However, here A1 is already assigned to U1, and the 20 tons in the top pocket indicates that this much freshwater was required.

Now this node gets branched by the algorithm and for example, U2 is selected. The

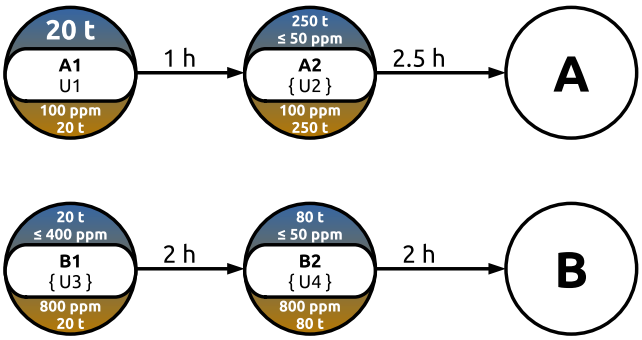


Figure 6.2: S-graph after the first assignment (node 1)

only task for U2 is A2, so a single child node is created by the EQB. However, A2 does not require completely clean water, so there are 4 alternatives for its water supply:

1. Use clean water only.
2. Use water output of A1.
3. Use water output of B1.
4. Use water output of B2.

Note that reusing water from multiple sources is not considered, as stated in Section 6.1.

For the first case, the procedure is similar as it was for U1 and A1 in the previous branching step. The resulting node with 270 tons of freshwater usage is shown in Figure 6.3.

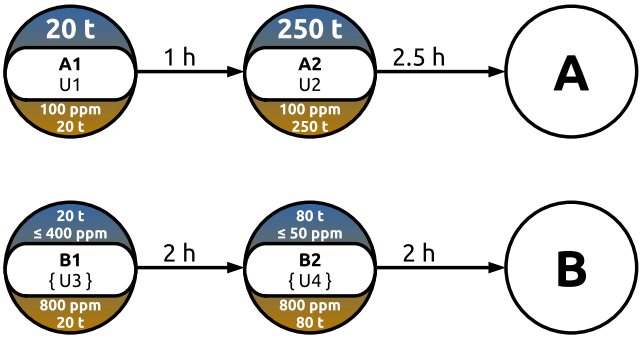


Figure 6.3: S-graph of node 2

For the second alternative, A1 can provide 20 tons of water with 100 ppm contamination. 100 ppm is twice the limit for A2, however, 230 tons of freshwater has to be mixed for the required amount, thus the contamination will drop to 8 ppm, which is suitable for A2, so all 20 tons can be reused. This is calculated using the formula shown in Equation (6.1):

$$f = \min\left\{w_{A1}, m^{in}_{A2}, \frac{c^{max}_{A2}}{c_{A1}} m_{A2}\right\} = \min\{20t, 250t, \frac{50ppm}{100ppm}250t\} = 20t \qquad (6.3)$$

Figure 6.4 shows the resulting S-graph, where the schedule-arc associated with this water transfer is highlighted with a green dashed line and annotated with the water amount. Note that the top pocket of A2 now only indicates 230 tons of freshwater usage, and the remaining water output of A1 (w_{A1}) in the bottom pocket is reduced to 0 tons.

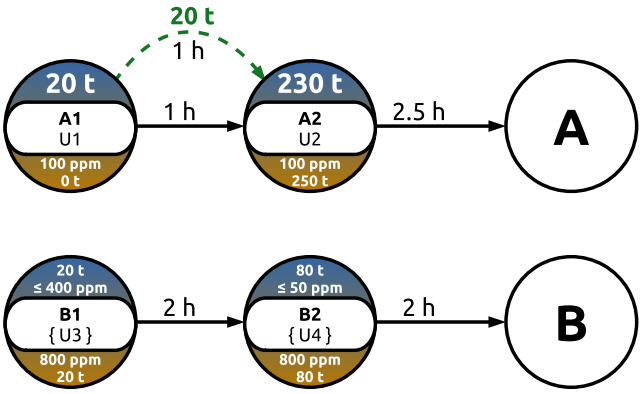


Figure 6.4: S-graph of node 3

In the third case, B1 can provide 20 tons of water as well. However, it is much more contaminated, having 800 ppm. Even with the added 230 tons of clean water, the contamination level would be 64 ppm, which is above the limit. Thus, only part of that 20 tons can be reused. To determine the amount, Equation (6.1) is used again:

$$f = \min\{20t, 250t, \frac{50ppm}{800ppm}250t\} = 15.625t \tag{6.4}$$

With reusing 15.625 tons, 234.375 tons of freshwater is needed, as indicated in the top pocket of A2 in Figure 6.5.

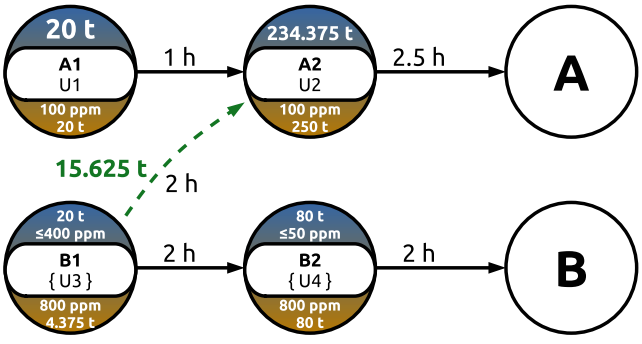


Figure 6.5: S-graph of node 4

Note that the introduced schedule-arc has the timing weight of 2 hours, thus the earliest starting of A2 is moved from 1 hour to 2 hours, and it will finish at 4.5 hours, half hour before the end of the time horizon. This makespan is indicated by the longest path B1 → A2 → A in the S-graph. The lower bound on the freshwater requirement at

this node is 254.375 tons, based on the tasks whose inputs have been determined (A1 and A2). Note that 4.375 tons of water still remains at B1, which can be used for other tasks as an input.

The last option is to provide water from B2. Using the same formula, at most 15.625 tons of water can be used from that source as well, as its contamination level is the same. The generated node is also similar, it is shown in Figure 6.6. However, the longest path in this graph is B1 → B2 → A2 → A, with a makespan of 6.5 hours, which is above the time horizon, so this node is pruned from the search.

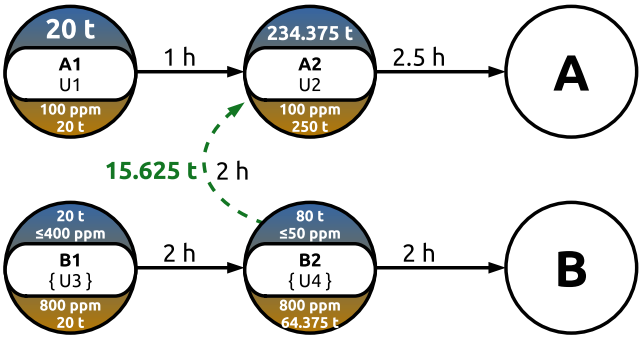


Figure 6.6: S-graph of node 5

The progress of the B&B algorithm so far is shown in Figure 6.7. At this stage, node 2, 3, and 4 are still unexplored. For each node, the lower bound on the makespan and on the freshwater usage is indicated. If the former ever exceeds the time horizon (as it was the case for node 5), the partial schedule is pruned because of infeasibility.

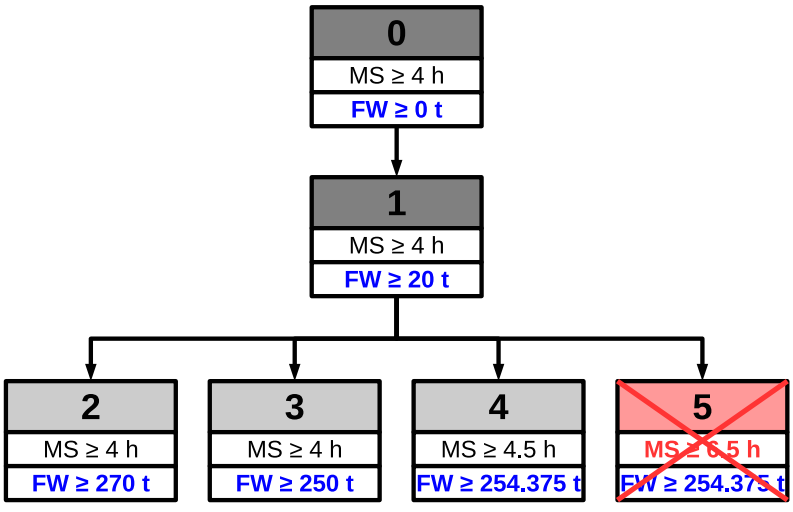


Figure 6.7: Top of the B&B tree of the algorithm

A possible way to continue the search is to branch at node 4, select unit U3, and generate child nodes for B1 with the alternatives: i) using only clean water, ii) reuse

water from A1, iii) reuse water from A2, iv) reuse water from B2. The latter two would create a cycle in the S-graph, indicating an infeasibility, and even the second option would violate the deadline, leaving only clean water usage as an option after reusing water from B1 for A2. Further exploring this branch, the last unit to schedule is U4 for B2, which can either use clean water, or water from A1, A2, B1. Except for using water from A2, all the options are feasible and provide complete schedules. The state of the search tree after these branching steps would look like as indicated in Figure 6.8.

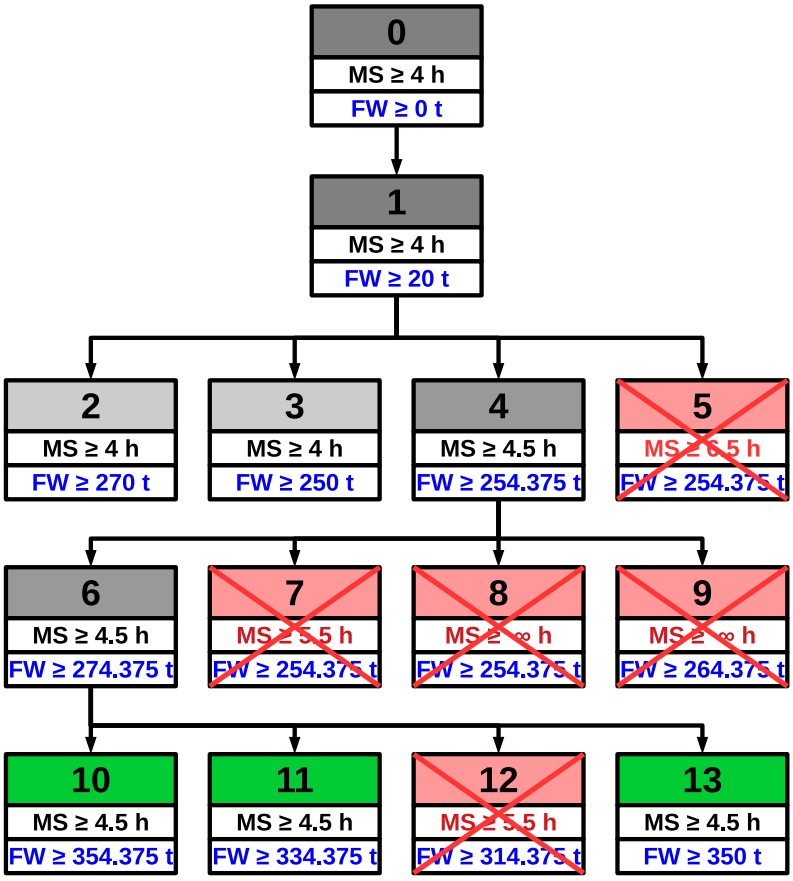


Figure 6.8: B&B tree of the algorithm - extended

At this stage, node 11 provides the best schedule with 334.375 tons of freshwater usage. This schedule corresponds to the option of using 15.625 tons of the output of B1 for A2 and 20 tons of the output of A1 for B2. The tree still has two unexplored nodes (2 and 3) where a better solution may be found. After enumerating the rest of the tree, it can be found that node 11 is actually the optimal solution.

6.4 Empirical validation

The proposed approach was tested on literature examples for validation. Solution times were less than 10 ms for each example, so the computational need was negligible. Note that in these examples, there is exactly 1 dedicated unit for each task but the proposed approach can also be used for problems where multiple units may be suitable for processing a task.

6.4.1 Example 1

Example 1 is adapted from [65], with its limiting water data given in Table 6.1. Two sub-scenarios are analyzed here, i.e. single batch and cyclical operations. To test the proposed scheduling approach, the fixed start and end times of the original example are ignored, and a flexible schedule is assumed, within the time horizon of 8 h.

For single batch operation, the optimal solution from the S-graph approach shows (Figure 6.9) that 1560 kg freshwater is needed, same as that reported by [65]. However, the S-graph solver reported a slightly different schedule (Figure 6.10), where instead of splitting the output water from Wash A between Wash B and Wash C, its output is reused entirely by Wash B, while the output of the latter gets reused again for Wash C. This solution has the same objective value as the one given by [65]. The resulting schedule does not require water storage, while the original, fixed schedule required the output of Wash A to be stored either in a dedicated tank, or in an idle unit [65].

Table 6.1: Limiting water data for Example 1

Task	Predecessors	Unit	Water input and output (kg)	Max. inlet contamination (kg salt / kg water)	Outlet contamination (kg salt / kg water)	Duration (h)
Wash A	-	Reactor A	1000	0	0.1	3
Reaction B	-	Reactor B	280	0.25	0.51	4
Wash B	B reaction	Reactor B	400	0.1	0.1	1.5
Reaction C	-	Reactor C	280	0.25	0.51	4
Wash C	C reaction	Reactor C	400	0.1	0.1	1.5

[65] showed that more water can be reused in the cyclic state of the network with storage tanks, reducing freshwater usage to 1000 kg. When cyclic scheduling is considered, there are several methods to model the relationship between consecutive cycles [13]. Since in this example there are no precedence relations between tasks in different units, the job repetition and machine chain repetition models are equivalent, and the resulting cycle time is the longest path in the recipe graph, which is 5.5 h. In cyclic scheduling, the cycle time

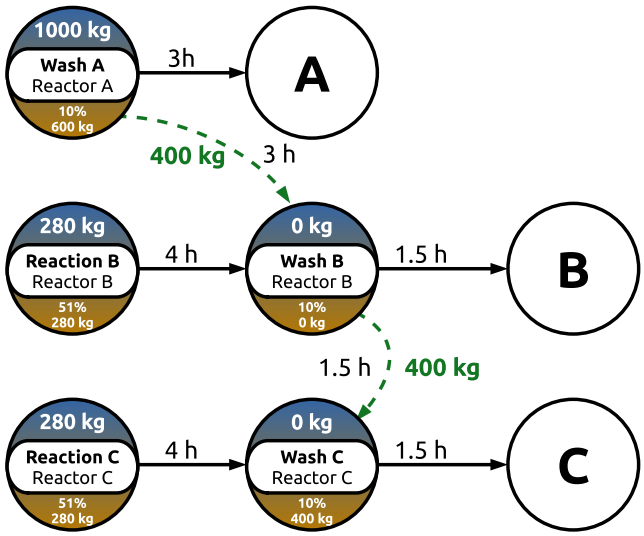


Figure 6.9: S-graph solution for Example 1

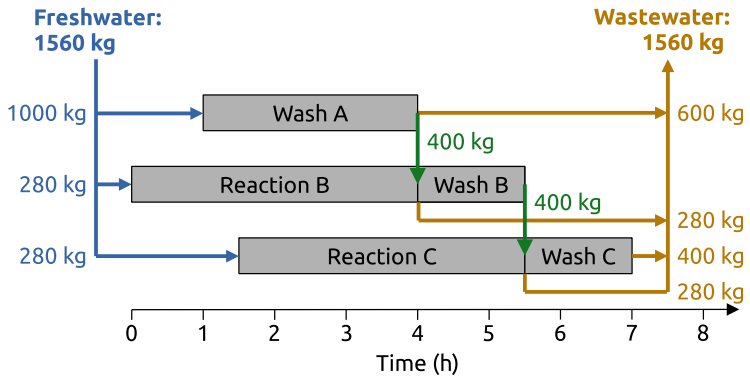


Figure 6.10: Schedule obtained for Example 1

is used instead of the makespan to determine if the schedule fits into the time horizon. In consequence, the S-graph shown in Figure 6.11 is allowed despite the makespan of 10 h, and the time horizon being 8 h. Figure 6.12 shows the cyclic schedule. This solution achieves the same 1000 kg freshwater usage that was proposed by [65]. Assuming that the reactors can act as storage tanks while they are inactive, only 400 kg of storage is needed for storing the part of the output water of Wash A that is reused by Wash B. If a storage tank is not available, the cycle time increases to 7 h.

The previously described cyclic scheduling method does not allow water to be transferred between consecutive batches of the recipe (e.g. from the dark gray tasks to the light gray ones). Also, the machine chain repetition model would require a more complex cycle time computation method if there are tasks with multiple suitable units. A more general approach for cyclic scheduling with S-graphs is a topic of future research [81].

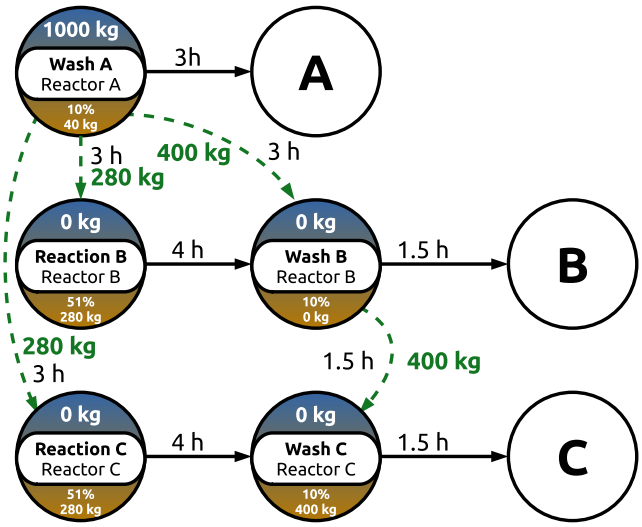


Figure 6.11: S-graph solution for the cyclic variant of Example 1

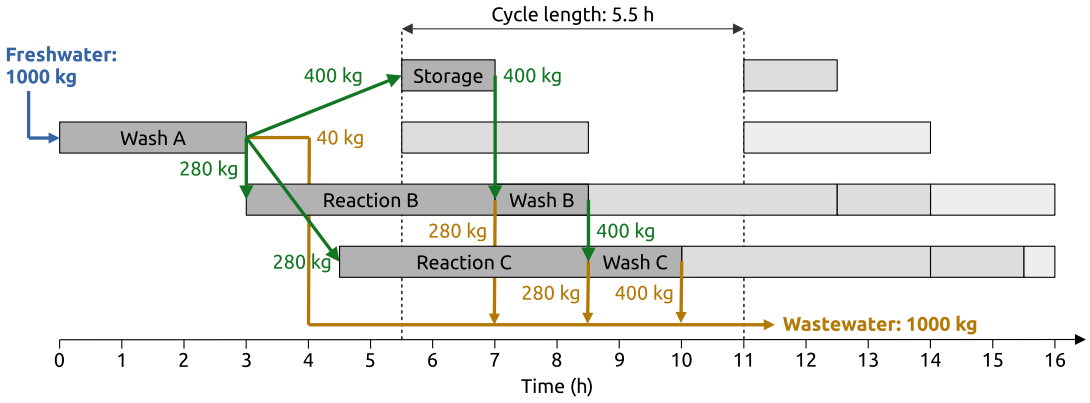


Figure 6.12: Cyclic schedule for Example 1

6.4.2 Example 2

Example 2 was originally reported by [63]. The example was later reinvestigated by [61], where solution without the use of water storage was proposed. While the original case study was a fixed load problem, here it is converted to a fixed flowrate problem, where the amount of water needed by a task is independent of its contamination level, as described in Section 6.1. The required water amounts and concentrations have been adjusted according to the fixed load solutions reported by [61] to get comparable solutions for validation. The limiting water data after the conversion are shown in Table 6.2.

[61] reported a solution with 80.5 t freshwater usage, with a water storage tank of 10.83 t, and its corresponding makespan is 8.5 h. Two improved solutions were also proposed, where no water storage is required, with the makespan reduced to 6 and 5 h [61].

With the time horizon set to 5 h, the S-graph solver found the same schedule as [61]

Table 6.2: Limiting water data for Example 2

Task	Unit	Water input and output (t)	Max. inlet contamination (ppm)	Outlet cont. (ppm)	Duration (h)
A	U1	50	0	400	2
B	U2	22.5	0	400	1
C	U3	10	200	500	3
D	U4	24	350	600	4
E	U5	33.33	400	700	2.5

with 80.5 t freshwater usage, it is shown in Figure 6.13. This schedule is also the optimal solution for the cyclic variant of this scheduling problem with a cycle time of 4 h. The schedule does not require a water storage tank.

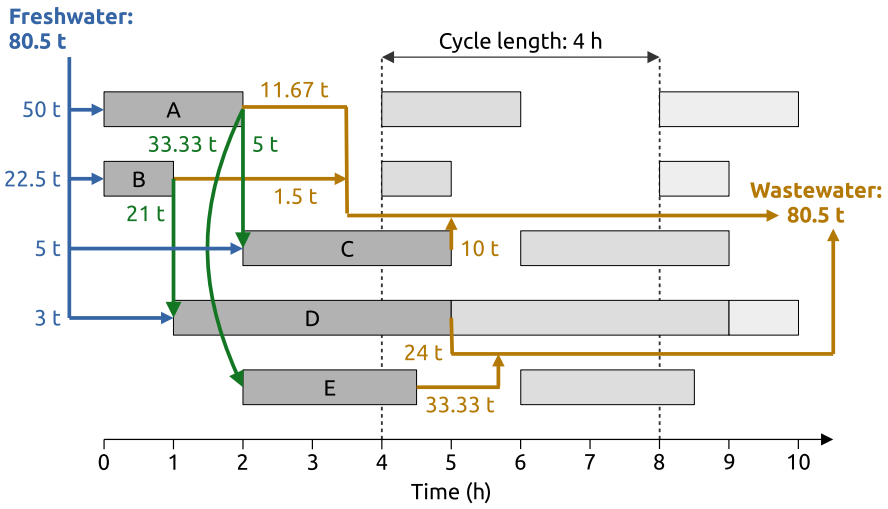


Figure 6.13: Schedule for Example 2 with makespan = 5 h, cycle time = 4 h

Increasing the time horizon does not result in any better solutions in terms of water reuse. On the other way, smaller time horizons were investigated, and resulted with two other Pareto-optimal solutions for the non-cyclic variant. The objective values of these solutions are shown in Figure 6.14, while their schedules are presented in Figure 6.15-6.16. These alternative solutions have higher water consumption (+26% and +46%), but the reduced makespan (−10% and −20%) can be beneficial for production. This demonstrates how the algorithm can be applied iteratively to obtain the optimal water reuse solutions for different makespan values.

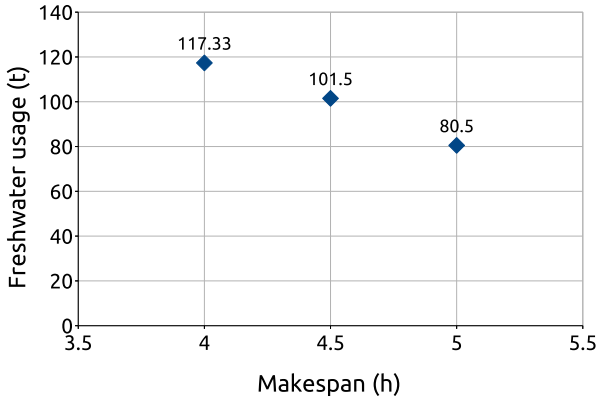


Figure 6.14: Pareto-optimal solutions for Example 2

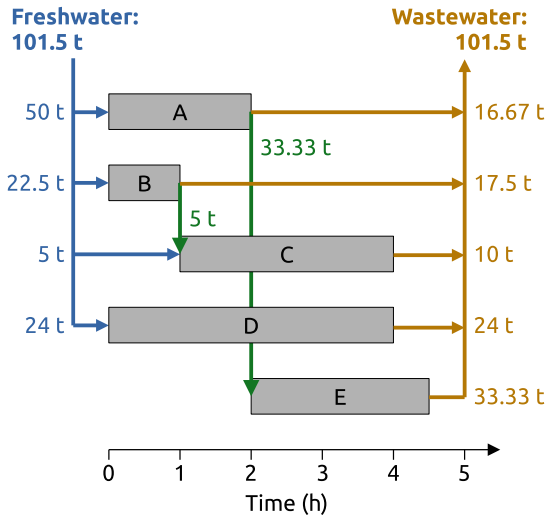


Figure 6.15: Schedule for Example 2 with makespan = 4.5 h

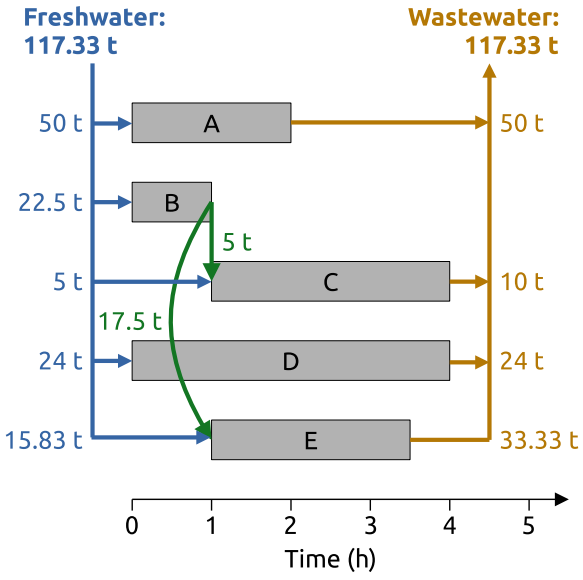


Figure 6.16: Schedule for Example 2 with makespan = 4 h

6.5 Summarizing statements

Thesis statement 4 *I have developed a solution approach based on the S-graph framework for scheduling batch processes with the objective of freshwater usage minimization through wastewater recycling, and validated it on literature examples.*

Thesis statement 4/a *I have extended the S-graph model to store decisions about reusing the output water of a task as input water for other tasks.*

Thesis statement 4/b *I have modified the equipment-based branching method of the S-graph framework to generate partial schedules with alternative water usage decisions subject to contamination constraints, and to introduce the necessary precedence relations between the affected tasks.*

My publication related to the statement: [79]

Chapter 7

Conclusions and future prospects

I studied several very different scheduling problems and solution approaches. Most practical applications require specific modeling considerations, leading to even more types of problems and needs for research efforts in finding adequate solution methods for them. It is clear that scheduling is a hard problem, so the process of improving existing approaches will continue in the foreseeable future.

In Chapter 3, I presented improved methods for scheduling automated manufacturing systems. This research topic is getting more and more attention, as automation levels rise throughout many industrial sectors. Future research can further improve the efficiency of these systems through cyclic scheduling, and online, reactive scheduling based on real-time data received from IoT sensors.

The RCPSP presented in Chapter 4 is a very general problem class that can serve as the base of a wide range of scheduling problems where tasks require multiple scarce resources at the same time. Just as I proposed an S-graph-based solution method for dealing with these resource constraints, integrating them into other existing machine scheduling, batch process scheduling, or other solution approaches is a promising topic for future research.

In Chapter 5, I presented an example from the steel-processing industry for problem-specific constraints that have not yet been addressed by scheduling methods. Despite the specific nature of this problem, the deterioration of production tools is apparent in many industries, so my presented approach may be used in other fields as well.

Integrating water minimization into scheduling, as presented in Chapter 6, and considering other environmental impacts of the production systems is an important aspect that needs more attention in scheduling and production planning. I am happy to see that this is already happening, and that I could contribute to this trend.

Apart from the possible research continuation opportunities discussed above, I have several plans to continue my academic research within the field of scheduling. As automation and IoT are becoming a dominant factor in the industry, scheduling is getting more and more important and also more complex. More available information can help to obtain more fine-grain and thus, more efficient schedules, which can be more precisely executed with automated robots. However, it also increases the complexity of optimization models.

From the various problem classes, I am most interested in cyclic scheduling. I plan to study the techniques for modeling periodic operations and possibly come up with new methods, for example, extending the S-graph framework for cyclic scheduling.

I am also interested in studying other solution methods. Until now, I concentrated on exact approaches but I would like to get more experience with heuristic methods as well. These days, the field of metaheuristics is a very popular research area with practical applications in scheduling and many other problems. Another area I plan to study is artificial intelligence and its integration with operations research. It can support the decision-making when the amount of information becomes too big to handle with conventional optimization methods.

It is clear that scheduling will only gain more research interest in the future. I am confident that many bachelor, master, and doctoral students will choose a similar research topic in the next few years, and I hope they will find this work useful. Maybe I will even have the chance to supervise some of them.

Summary

There are many different problem classes in the research area of scheduling, and also a lot of solution techniques one can use for them. I presented solution approaches which I developed based on MILP modeling techniques and the S-graph framework. They were made for 4 distinct scheduling problems and their variants.

For scheduling automated wet-etch stations and other automated manufacturing systems, I proposed 2 improved formulations of literature MILP models. I improved the performance of a model that can solve the most general variant of this problem class. For the less general model, I proposed extensions to fix a potential infeasibility in the reported solutions, and to handle more problem features.

I developed new branching algorithms for the S-graph framework to solve the RCPSP, its multi-mode variant, and its variant with time-varying resource capacities. The proposed methods have comparable performances with literature MILP models.

I identified a novel scheduling problem in a steel-processing forge, where production equipment deteriorates through setups and changeovers, and developed an MILP model to solve it. I also proposed several optional constraints to improve the performance of the model.

I proposed a modified version of an S-graph branching algorithm, which can be used for the minimization of freshwater usage and wastewater generation of a process during its scheduling by allowing the reuse of the output water of a process as the input water of processes.

My proposed approaches have been tested on problem instances either taken from the literature or generated systematically. The tests confirmed the correctness of the proposed methods, and the solution performance was compared to literature approaches, where it was applicable.

Összefoglaló

Az ütemezés tudományterületén számos különféle feladatosztályt tartanak számon, melyekre sokféle megoldási technika ismert. Az értekezésemben bemutatott megoldó módszerek MILP modellezési technikákra és az S-gráf módszertanra épülnek. Ezen módszerek 4 különböző ütemezési feladatot, és azok további változatait képesek megoldani.

A nedves-marási rendszerek és más automatizált gyártórendszerek ütemezésére a szakirodalomban megjelent MILP modellek közül 2-t továbbfejlesztettem. Az általánosabb feladatosztályú modellnek javítottam a megoldási hatékonyságát. A speciálisabb feladatosztályú modellt pedig kiterjesztettem további korlátozások kezelésére, köztük egy gyakorlatban megvalósíthatatlan megoldást okozó hiányosságot is javítottam.

Új elágazási algoritmusokat fejlesztettem ki az S-gráf keretrendszerhez, hogy képes legyen megoldani az RCPSP-t és annak a többmódú, valamint az időben változó erőforráskapacitásokat tartalmazó változatát. Ezen módszerek a szakirodalmi MILP modellekhez hasonló megoldási hatékonyságúak.

Egy újfajta ütemezési feladatot azonosítottam egy acélfeldolgozó kovácsüzemben, amiben a gyártóeszközök élettartama a termékváltások során szükséges konfigurálási folyamatok hatására csökken. Egy MILP modellt és hozzá különböző javító korlátozásokat készítettem a feladat megoldására.

Kifejlesztettem egy módosított elágazási eljárást az S-gráf keretrendszerhez, amely képes minimalizálni a víz-felhasználást és szennyvíz-termelést az ütemezendő gyártási folyamatban azáltal, hogy lehetővé teszi, hogy egy folyamatból kijövő szennyezett víz felhasználásra kerüljön más folyamatok bemenetén.

Az általam kifejlesztett módszereket szakirodalmi vagy véletlenszerűen generált feladatokon teszteltem, hogy ellenőrizzem a helyességüket. Valamint a megoldási hatékonyságukat is összehasonlítottam szakirodalmi módszerekével, ahol erre lehetőség volt.

Bibliography

[1] R. Adonyi, E. Shopova, and N. Vaklieva-Bancheva. "Optimal schedule of a dairy manufactory". In: *Chemical and Biochemical Engineering Quarterly* 23.2 (2009), pp. 231–237. URL: http://silverstripe.fkit.hr/cabeq/past-issues/article/316.

[2] R. Adonyi, G. Biros, T. Holczinger, and F. Friedler. "Effective scheduling of a large-scale paint production system". In: *Journal of Cleaner Production* 16.2 (2008), pp. 225–232. DOI: 10.1016/j.jclepro.2006.08.021.

[3] R. Adonyi, I. Heckl, A. Szalamin, and F. Olti. "Routing of railway systems with the S-graph framework for effective scheduling". In: *Chemical Engineering Transactions* 21 (2010), pp. 913–918. DOI: 10.3303/CET1021153.

[4] A. M. Aguirre, C. A. Méndez, and P. M. Castro. "A novel optimization method to automated wet-etch station scheduling in semiconductor manufacturing systems". In: *Computers & Chemical Engineering* 35.12 (2011), pp. 2960–2972. DOI: 10.1016/j.compchemeng.2011.02.014.

[5] A. M. Aguirre, C. A. Méndez, Á. García-Sánchez, M. Ortega-Mier, and P. M. Castro. "General framework for automated manufacturing systems: Multiple hoists scheduling solution". In: *Chemical Engineering Transactions* 32 (2013), pp. 1381–1386. DOI: 10.3303/CET1332231.

[6] M. Almató, A. Espuña, and L. Puigjaner. "Optimisation of water use in batch process industries". In: *Computers & Chemical Engineering* 23.10 (1999), pp. 1427–1437.

[7] M. Almató, E. Sanmartí, A. Espuña, and L. Puigjaner. "Rationalizing the water use in the batch process industry". In: *Computers & chemical engineering* 21 (1997), S971–S976.

[8] M. Bagajewicz. "A review of recent design procedures for water networks in refineries and process plants". In: *Computers & chemical engineering* 24.9-10 (2000), pp. 2093–2113.

[9] M. Bartusch, R. H. Möhring, and F. J. Radermacher. "Scheduling project networks with resource constraints and time windows". In: *Annals of Operations Research* 16.1 (1988), pp. 199–240. DOI: 10.1007/BF02283745.

[10] C. Bessiere. "Constraint Propagation". In: *Handbook of Constraint Programming*. Ed. by F. Rossi, P. van Beek, and T. Walsh. Amsterdam: Elsevier, 2006. Chap. 3, pp. 29–83.

[11] S. Bhushan and I. A. Karimi. "An MILP approach to automated wet-etch station scheduling". In: *Industrial & Engineering Chemistry Research* 42.7 (2003), pp. 1391–1399. DOI: 10.1021/ie020296c.

[12] S. Bhushan and I. A. Karimi. "Heuristic algorithms for scheduling an automated wet-etch station". In: *Computers and Chemical Engineering* 28.3 (2004), pp. 363–379. DOI: 10.1016/S0098-1354(03)00192-3.

[13] P. Brucker and T. Kampmeyer. "A general model for cyclic machine scheduling problems". In: *Discrete Applied Mathematics* 156.13 (2008), pp. 2561–2572. DOI: 10.1016/j.dam.2008.03.029.

[14] P. M. Castro, I. E. Grossmann, and Q. Zhang. "Expanding scope and computational challenges in process scheduling". In: *Computers & Chemical Engineering* 114 (2018), pp. 14–42. DOI: https://doi.org/10.1016/j.compchemeng.2018.01.020.

[15] P. M. Castro, L. J. Zeballos, and C. A. Méndez. "Hybrid time slots sequencing model for a class of scheduling problems". In: *AIChE Journal* 58.3 (2012). DOI: 10.1002/aic.

[16] J. Cerdá, G. P. Henning, and I. E. Grossmann. "A Mixed-Integer Linear Programming Model for Short-Term Scheduling of Single-Stage Multiproduct Batch Plants with Parallel Lines". In: *Industrial & Engineering Chemistry Research* 36.5 (1997), pp. 1695–1707. DOI: 10.1021/ie9605490.

[17] R. K. Chakrabortty, R. A. Sarker, and D. L. Essam. “Event Based Approaches for Solving Multi-mode Resource Constraints Project Scheduling Problem”. In: *Lecture Notes in Computer Science* 8838 (2014), pp. 375–386. DOI: 10.1007/978-3-662-45237-0_35.

[18] N. D. Chaturvedi and S. Bandyopadhyay. “Optimization of multiple freshwater resources in a flexible-schedule batch water network”. In: *Industrial and Engineering Chemistry Research* 53.14 (2014), pp. 5996–6005. DOI: 10.1021/ie403638v.

[19] J. Cheng, J. Fowler, K. Kempf, and S. Mason. “Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting”. In: *Computers and Operations Research* 53 (2015), pp. 275–287. DOI: 10.1016/j.cor.2014.04.018.

[20] K.-F. Cheng and C.-T. Chang. “Integrated Water Network Designs for Batch Processes”. In: *Industrial & Engineering Chemistry Research* 46.4 (2007), pp. 1241–1253. DOI: 10.1021/ie0611150.

[21] N. Christensen and D. Lettenmaier. “A multimodel ensemble approach to assessment of climate change impacts on the hydrology and water resources of the Colorado River Basin”. In: *Hydrology and Earth System Sciences* 11.4 (2007), pp. 1417–1434. DOI: 10.5194/hess-11-1417-2007.

[22] B. Dávid, **O. Ősz**, and M. Hegyháti. “Robust Scheduling of Waste Wood Processing Plants with Uncertain Delivery Sources and Quality”. In: *Sustainability* 13.9 (2021). DOI: 10.3390/su13095007.

[23] E. L. Demeulemeester and W. S. Herroelen. “Modelling setup times, process batches and transfer batches using activity network logic”. In: *European Journal of Operational Research* 89.2 (1996), pp. 355–365. DOI: 10.1016/0377-2217(94)00249-5.

[24] S. Ferrer-Nadal, E. Capón-García, C. A. Méndez, and L. Puigjaner. “Material transfer operations in batch scheduling. A critical modeling issue”. In: *Industrial and Engineering Chemistry Research* 47.20 (2008), pp. 7721–7732. DOI: 10.1021/ie800075u.

[25] C. A. Floudas and X. Lin. "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review". In: *Computers & Chemical Engineering* 28.11 (2004), pp. 2109–2129.

[26] C. A. Floudas and X. Lin. "Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications". In: *Annals of Operations Research* 139.1 (2005), pp. 131–162.

[27] D. C. Y. Foo. "Automated targeting technique for batch process integration". In: *Industrial & engineering chemistry research* 49.20 (2010), pp. 9899–9916.

[28] D. C. Y. Foo. "State-of-the-Art Review of Pinch Analysis Techniques for Water Network Synthesis". In: *Industrial & Engineering Chemistry Research* 48.11 (2009), pp. 5125–5159. DOI: 10.1021/ie801264c.

[29] D. C. Y. Foo, Z. A. Manan, and Y. L. Tan. "Synthesis of maximum water recovery network for batch process systems". In: *Journal of Cleaner Production* 13.15 (2005), pp. 1381–1394.

[30] D. C. Foo. *Process integration for resource conservation*. CRC press, 2012.

[31] Y. Fu, G. Tian, A. Fathollahi-Fard, A. Ahmadi, and C. Zhang. "Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint". In: *Journal of Cleaner Production* 226 (2019), pp. 515–525. DOI: 10.1016/j.jclepro.2019.04.046.

[32] A. Gascon and R. C. Leachman. "A Dynamic Programming Solution to the Dynamic, Multi-Item, Single-Machine Scheduling Problem". In: *Operations Research* 36.1 (1988), pp. 50–56.

[33] B. Gaybullaev, S.-C. Chen, and Y.-M. Kuo. "Large-scale desiccation of the Aral Sea due to over-exploitation after 1960". In: *Journal of Mountain Science* 9.4 (2012), pp. 538–546. DOI: 10.1007/s11629-012-2273-1.

[34] C. D. Geiger, K. G. Kempf, and R. Uzsoy. "A Tabu search approach to scheduling an automated wet etch station". In: *Journal of Manufacturing Systems* 16.2 (1997), pp. 102–116.

[35] J. F. Gouws, T. Majozi, D. C. Y. Foo, C. L. Chen, and J. Y. Lee. "Water minimization techniques for batch processes". In: *Industrial and Engineering Chemistry Research* 49.19 (2010), pp. 8877–8893. DOI: 10.1021/ie100130a.

[36] I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand, and J. Wassick. "Scope for industrial applications of production scheduling models and solution methods". In: *Computers and Chemical Engineering* 62 (2014), pp. 161–193. DOI: 10.1016/j.compchemeng.2013.12.001.

[37] S. Hartmann. "Project scheduling with resource capacities and requests varying with time: a case study". In: *Flexible Services and Manufacturing Journal* 25.1 (2013), pp. 74–93. DOI: 10.1007/s10696-012-9141-8.

[38] S. Hartmann and D. Briskorn. "A survey of variants and extensions of the resource-constrained project scheduling problem". In: *European Journal of Operational Research* 207.1 (2010), pp. 1–14. DOI: 10.1016/j.ejor.2009.11.005.

[39] S. Hartmann and D. Briskorn. "An updated survey of variants and extensions of the resource-constrained project scheduling problem". In: *European Journal of Operational Research* 297.1 (2022), pp. 1–14. DOI: https://doi.org/10.1016/j.ejor.2021.05.004.

[40] M. Hegyháti, T. Majozi, T. Holczinger, and F. Friedler. "Practical infeasibility of cross-transfer in batch plants with complex recipes: S-graph vs MILP methods". In: *Chemical Engineering Science* 64.3 (2009), pp. 605–610. DOI: 10.1016/j.ces.2008.10.018.

[41] M. Hegyháti, **O. ŐSz**, B. Kovács, and F. Friedler. "Scheduling of automated wet-etch stations". In: *21st International Congress of Chemical and Process Engineering and 17th Conference on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction*. CHISA/PRES. Prague, Czech Republic, Aug. 2014.

[42] M. Hegyháti, T. Holczinger, and **O. ŐSz**. "Addressing storage time restrictions in the S-graph scheduling framework". In: *Optimization and Engineering* 0123456789 (2020). DOI: 10.1007/s11081-020-09548-1.

[43] M. Hegyháti, T. Holczinger, A. Szoldatics, and F. Friedler. “Combinatorial approach to address batch scheduling problems with limited storage time”. In: *Chemical Engineering Transactions* 25 (2011), pp. 495–500. DOI: 10.3303/CET1125083.

[44] M. Hegyháti, **O. Ősz**, B. Kovács, and F. Friedler. “Scheduling of Automated Wet-Etch Stations”. In: *Chemical Engineering Transactions* 39 (2014), pp. 433–438. DOI: 10.3303/CET1439073.

[45] T. Holczinger. “Módszer köztes tárolókat nem tartalmazó szakaszos működésű rendszerek ütemezésére”. PhD thesis. Veszprémi Egyetem, 2004.

[46] T. Holczinger, T. Majozi, M. Hegyhati, and F. Friedler. “An automated algorithm for throughput maximization under fixed time horizon in multipurpose batch plants: S-Graph approach”. In: *Computer Aided Chemical Engineering*. Vol. 24. 2007, pp. 649–654. DOI: 10.1016/S1570-7946(07)80131-3.

[47] T. Holczinger and Á. Orosz. “Throughput Maximization with S-graph Framework using Global Branching Tree”. In: *MACRo 2015* 1.1 (2015), pp. 201–210. DOI: 10.1515/macro-2015-0020.

[48] C. Jiang, F. Wen, Y. Xue, F. Chen, Y. Sun, and L. Zhang. “Optimal power management strategy for industrial users based on the state task network considering user preferences”. In: *2021 IEEE Kansas Power and Energy Conference, KPEC 2021*. 2021. DOI: 10.1109/KPEC51835.2021.9446210.

[49] J.-K. Kim and R. Smith. “Automated design of discontinuous water systems”. In: *Process Safety and Environmental Protection* 82.3 (2004), pp. 238–248.

[50] R. Klein. “Project scheduling with time-varying resource constraints”. In: *International Journal of Production Research* 38.16 (2000), pp. 3937–3952. DOI: 10.1080/00207540050176094.

[51] R. Kolisch and A. Sprecher. “PSPLIB - A project scheduling problem library”. In: *European Journal of Operational Research* 96.1 (1997), pp. 205–216. DOI: 10.1016/S0377-2217(96)00170-1.

[52] R. Koller, L. Ricardez-Sandoval, and L. Biegler. "Stochastic back-off algorithm for simultaneous design, control, and scheduling of multiproduct systems under uncertainty". In: *AIChE Journal* 64.7 (2018), pp. 2379–2389. DOI: 10.1002/aic.16092.

[53] E. Kondili, C. Pantelides, and R. Sargent. "A general algorithm for short-term scheduling of batch operations—I. MILP formulation". In: *Computers & Chemical Engineering* 17.2 (1993), pp. 211–227. DOI: 10.1016/0098-1354(93)80015-F.

[54] O. Koné, C. Artigues, P. Lopez, and M. Mongeau. "Event-based MILP models for resource-constrained project scheduling problems". In: *Computers & Operations Research* 38.1 (2011), pp. 3–13. DOI: 10.1016/j.cor.2009.12.011.

[55] G. M. Kopanos, T. S. Kyriakidis, and M. C. Georgiadis. "New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems". In: *Computers & Chemical Engineering* 68 (2014), pp. 96–106. DOI: 10.1016/j.compchemeng.2014.05.009.

[56] S. Kreter, J. Rieck, and J. Zimmermann. "Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars". In: *European Journal of Operational Research* 251.2 (2016), pp. 387–403. DOI: 10.1016/j.ejor.2015.11.021.

[57] T. S. Kyriakidis, G. M. Kopanos, and M. C. Georgiadis. "MILP formulations for single- and multi-mode resource-constrained project scheduling problems". In: *Computers & Chemical Engineering* 36.1 (2012), pp. 369–385. DOI: 10.1016/j.compchemeng.2011.06.007.

[58] J. M. Laínez, M. Hegyháti, F. Friedler, and L. Puigjaner. "Using S-graph to address uncertainty in batch plants". In: *Clean Technologies and Environmental Policy* 12.2 (2010), pp. 105–115. DOI: 10.1007/s10098-009-0240-5.

[59] J.-Y. Lee and D. C. Y. Foo. "Simultaneous Targeting and Scheduling for Batch Water Networks". In: *Industrial & Engineering Chemistry Research* 56.6 (2017), pp. 1559–1569. DOI: 10.1021/acs.iecr.6b03714.

[60] B.-H. Li and C.-T. Chang. "A mathematical programming model for discontinuous water-reuse system design". In: *Industrial & Engineering Chemistry Research* 45.14 (2006), pp. 5027–5036.

[61] Z. Li and T. Majozi. "Optimal Design of Batch Water Network with a Flexible Scheduling Framework". In: *Industrial & Engineering Chemistry Research* 58.22 (2019), pp. 9500–9511. DOI: 10.1021/acs.iecr.9b00399.

[62] Z. Li and T. Majozi. "Optimal Synthesis of Batch Water Networks Using Dynamic Programming". In: *Process Integration and Optimization for Sustainability* 2.4 (2018), pp. 391–412. DOI: 10.1007/s41660-018-0061-2.

[63] Y. Liu, G. Li, L. Wang, J. Zhang, and K. Shams. "Optimal design of an integrated discontinuous water-using network coordinating with a central continuous regeneration unit". In: *Industrial and Engineering Chemistry Research* 48.24 (2009), pp. 10924–10940. DOI: 10.1021/ie9000053.

[64] T. Majozi. "Wastewater minimisation using central reusable water storage in batch plants". In: *Computers & Chemical Engineering* 29.7 (2005), pp. 1631–1646. DOI: 10.1016/j.compchemeng.2005.01.003.

[65] T. Majozi, C. J. Brouckaert, and C. A. Buckley. "A graphical technique for wastewater minimisation in batch processes". In: *Journal of Environmental Management* 78.4 (2006), pp. 317–329. DOI: 10.1016/j.jenvman.2005.04.026.

[66] T. Majozi and F. Friedler. "Maximization of throughput in a multipurpose batch plant under a fixed time horizon: S-graph approach". In: *Industrial and Engineering Chemistry Research* 45.20 (2006), pp. 6713–6720. DOI: 10.1021/ie0604472.

[67] C. Méndez and J. Cerdá. "State-of-the-art review of optimization methods for short-term scheduling of batch processes". In: *Computers & Chemical Engineering* 30.6-7 (2006), pp. 913–946. DOI: 10.1016/j.compchemeng.2006.02.008.

[68] C. Méndez, G. Henning, and J. Cerdá. "An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities". In: *Computers & Chemical Engineering* 25.4 (2001), pp. 701–711. DOI: 10.1016/S0098-1354(01)00671-8.

[69] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. "An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation". In: *Management Science* 44.5 (1998), pp. 714–729. DOI: 10.1287/mnsc.44.5.714.

[70] C. Ning and F. You. "Data-driven decision making under uncertainty integrating robust optimization with principal component analysis and kernel smoothing methods". In: *Computers and Chemical Engineering* 112 (2018), pp. 190–210. DOI: 10.1016/j.compchemeng.2018.02.007.

[71] D. R. Nonyane and T. Majozi. "Long term scheduling technique for wastewater minimisation in multipurpose batch processes". In: *Applied Mathematical Modelling* 36.5 (2012), pp. 2142–2168. DOI: 10.1016/J.APM.2011.08.007.

[72] J. M. Novas and G. P. Henning. "A comprehensive constraint programming approach for the rolling horizon-based scheduling of automated wet-etch stations". In: *Computers and Chemical Engineering* 42 (2012), pp. 189–205. DOI: 10.1016/j.compchemeng.2012.01.005.

[73] R. A.-V. Olaguíbel and J. T. Goerlich. "The project scheduling polyhedron: Dimension, facets and lifting theorems". In: *European Journal of Operational Research* 67.2 (1993), pp. 204–220. DOI: 10.1016/0377-2217(93)90062-R.

[74] **O. Ősz** and M. Hegyháti. "A novel combinatorial approach for the resource-constrained scheduling problem". In: *VOCAL Optimization Conference: Advanced Algorithms*. Esztergom, Hungary, Dec. 2016.

[75] **O. Ősz** and M. Hegyháti. "Combinatorial approach for the multi-mode resource-constrained project scheduling problem". In: *Joint EURO / ORSC / ECCO Conference 2017 on Combinatorial Optimization*. Koper, Slovenia, Mar. 2017.

[76] **O. Ősz** and M. Hegyháti. "Scheduling a forge with due dates and die deterioration". In: *16th Int'l Conference on Project Management and Scheduling*. PMS. Rome, Italy, Apr. 2018.

[77] **O. Ősz**, B. Kovács, and M. Hegyháti. "Combinatorial approach for the scheduling of Automated Wet-etch Stations". In: *Veszprém Optimization Conference: Advanced Algorithms and Annual Scientific Conference of the Hungarian National Coordinating Center for Infocommunications (NIKK)*. VOCAL/ASCONIKK. Veszprém, Hungary, Dec. 2014.

[78] **O. Ősz**, B. Ferenczi, and M. Hegyháti. "Scheduling a forge with due dates and die deterioration". In: *Annals of Operations Research* (2019). DOI: 10.1007/s10479-019-03336-6.

[79] **O. Ősz**, D. C. Y. Foo, and M. Hegyháti. "Minimizing Freshwater Usage in Batch Process Scheduling: S-Graph Approach". In: *Process Integration and Optimization for Sustainability* (2020). DOI: 10.1007/s41660-020-00142-7. Number of citations: 1.

[80] **O. Ősz** and M. Hegyháti. "An S-graph based approach for multi-mode resource-constrained project scheduling with time-varying resource capacities". In: *Chemical Engineering Transactions* 70 (2018), pp. 1165–1170. DOI: 10.3303/CET1870195. Number of citations: 2.

[81] **O. Ősz** and M. Hegyháti. "Interlacing in cyclic scheduling". In: *8th VOCAL Optimization Conference: Advanced Algorithms Esztergom, Hungary, December 10-12, 2018 Short Papers*. 2018, pp. 50–55.

[82] E. Oztemel and A. A. Selam. "Bees Algorithm for multi-mode, resource-constrained project scheduling in molding industry". In: *Computers & Industrial Engineering* 112 (2017), pp. 187–196. DOI: 10.1016/j.cie.2017.08.012.

[83] C. C. Pantelides. "Unified frameworks for optimal process planning and scheduling". In: *Proceedings of the second international conference on foundations of computer-aided process operations*. Ed. by D. Rippin, J. C. Hale, and J. F. Davis. 1994, pp. 253–274.

[84] M. E. Pfund, S. J. Mason, and J. W. Fowler. "Semiconductor Manufacturing Scheduling and Dispatching". In: *Handbook of Production Scheduling*. Ed. by J. W. Herrmann. Boston, MA: Springer US, 2006, pp. 213–241. DOI: 10.1007/0-387-33117-4_9.

[85] J. M. Pinto and I. E. Grossmann. "Assignment and sequencing models for the scheduling of process systems". In: *Annals of Operations Research* 81 (1998), pp. 433–466.

[86] A. A. B. Pritsker, L. J. Waiters, and P. M. Wolfe. "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach". In: *Management Science* 16.1 (1969), pp. 93–108. DOI: 10.1287/mnsc.16.1.93.

[87] T. Samukawa and H. Suwa. "An Optimization of Energy-Efficiency in Machining Manufacturing Systems Based on a Framework of Multi-Mode RCPSP". In: *International Journal of Automation Technology* 10.6 (2016), pp. 985–992. DOI: 10.20965/ijat.2016.p0985.

[88] E. Sanmartí, F. Friedler, and L. Puigjaner. "Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation". In: *Computers & Chemical Engineering* 22 (1998), S847–S850. DOI: 10.1016/S0098-1354(98)00163-X.

[89] E. Sanmartí, L. Puigjaner, T. Holczinger, and F. Friedler. "Combinatorial framework for effective scheduling of multipurpose batch plants". In: *AIChE Journal* 48.11 (2002), pp. 2557–2570. DOI: 10.1002/aic.690481115.

[90] G. Schilling and C. Pantelides. "A simple continuous-time process scheduling formulation and a novel solution algorithm". In: *Computers & Chemical Engineering* 20 (1996), S1221–S1226. DOI: 10.1016/0098-1354(96)00211-6.

[91] E. Seid and T. Majozi. "Optimization of energy and water use in multipurpose batch plants using an improved mathematical formulation". In: *Chemical Engineering Science* 111 (2014), pp. 335–349. DOI: 10.1016/j.ces.2014.02.036.

[92] M. Shaik and P. Mathur. "Generalization of Scheduling Models for Batch Plants and Pipeless Plants". In: *Industrial and Engineering Chemistry Research* 58.19 (2019), pp. 8195–8205. DOI: 10.1021/acs.iecr.9b00106.

[93] C. Shang and F. You. "Distributionally robust optimization for planning and scheduling under uncertainty". In: *Computers and Chemical Engineering* 110 (2018), pp. 53–68. DOI: 10.1016/j.compchemeng.2017.12.002.

[94] G. W. Shapiro and H. L. W. Nuttle. "Hoist Scheduling For A PCB Electroplating Facility". In: *IIE Transactions* 20.2 (1988), pp. 157–167. DOI: 10.1080/07408178808966165.

[95] R. Smith. *Chemical process design and integration*. Chichester, West Sussex, United Kingdom: John Wiley & Sons, Inc, 2016.

[96] F. B. Talbot. "Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case". In: *Management Science* 28.10 (1982), pp. 1197–1210. DOI: 10.1287/mnsc.28.10.1197.

[97] L. Tang and P. Liu. "Flowshop scheduling problems with transportation or deterioration between the batching and single machines". In: *Computers and Industrial Engineering* 56.4 (2009), pp. 1289–1295.

[98] R. Théry Hétreux, G. Hétreux, P. Floquet, and A. Leclercq. "The energy Extended Resource Task Network, a general formalism for the modeling of production systems: Application to waste heat valorization". In: *Energy* 214 (2021), p. 118970. DOI: https://doi.org/10.1016/j.energy.2020.118970.

[99] I. Uhlmann and E. Frazzon. "Production rescheduling review: Opportunities for industrial integration and practical applications". In: *Journal of Manufacturing Systems* 49 (2018), pp. 186–193. DOI: 10.1016/j.jmsy.2018.10.004.

[100] UN. *Sustainable Development Goals*. 2015. URL: https://un.org/sustainabledevelopment.

[101] UN General Assembly. *International Decade for Action: Water for Sustainable Development: 2018–2028*. Tech. rep. RES/71/222 (7 February 2017), 2017.

[102] UNESCO. *The United Nations World Water Development Report 2017: Wastewater, The Untapped Resource*. WWAP (United Nations World Water Assessment Programme). Paris, 2017.

[103] UNICEF and WHO. *Progress on household drinking water, sanitation and hygiene 2000-2017. Special focus on inequalities*. United Nations Children's Fund (UNICEF) and World Health Organization. New York, 2019.

[104] R. Uzsoy, C.-Y. Lee, and L. A. Martin-Vega. "A review of production planning and scheduling models in the semiconductor industry part I: system characteristics, performance evaluation and production planning". In: *IIE transactions* 24.4 (1992), pp. 47–60. DOI: 10.1080/07408179208964233.

[105] Y. Wang and R. Smith. "Time pinch analysis". In: *Chemical Engineering Research & Design* 73.8 (1995), pp. 905–914.

[106] L. J. Zeballos, P. M. Castro, C. A. Méndez, and C. A. Meendez. "Integrated Constraint Programming Scheduling Approach for Automated Wet-Etch Stations in Semiconductor Manufacturing". In: *Industrial & Engineering Chemistry Research* 50.3 (2011), pp. 1705–1715. DOI: 10.1021/ie1016199.

[107] T. Zhang, Y. Wang, X. Jin, and S. Lu. "Integration of production planning and scheduling based on RTN representation under uncertainties". In: *Algorithms* 12.6 (2019). DOI: 10.3390/a12060120.

[108] Z. Zhang and J. a. Xu. "A multi-mode resource-constrained project scheduling model with bi-random coefficients for drilling grouting construction project". In: *International Journal of Civil Engineering* 11.1 (2013). URL: http://ijce.iust.ac.ir/article-1-699-en.html.

[109] C.-l. Zhao and H.-y. Tang. "Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan". In: *Applied Mathematical Modelling* 34.3 (2010), pp. 837–841.