Approaches to Operative Decision Support in the Management of Aircraft Turnaround Disruptions

Jürgen Kuster and Dietmar Jannach

***Abstract*— With the availability of comprehensively integrated Airport Operational Databases, as motivated by the emergence of Collaborative Decision Making for example, new forms of automated information processing become available: Particularly, proactive Decision Support Systems can be developed based on the concepts of Artificial Intelligence. From various potential fields of application, this paper focuses on the problem of managing disruptions within the aircraft turnaround, the most typical airport ground process. With a view to supporting respective system development intentions, it describes the results of the evaluation of several approaches towards the operative provision of decision support: Particularly, strengths and limitations of simulation, planning and scheduling are discussed in the regarded context. Based on the respective findings, it is finally shown how the Resource-Constrained Project Scheduling Problem (RCPSP) – the most promising approach in our view – can be extended for its application to aircraft turnaround disruption management.**

***Index Terms*— Decision Support System, Aircraft Turnaround Process, Disruption Management, Collaborative Decision Making**

I. Introduction

Even though the process that aircrafts go through between arrival and departure is well-defined in the so-called *aircraft turnaround*, a high level of time-criticality as well as various forms of dependencies make its execution prone to disruptions, delays and inefficiencies. Since each of these irregularities causes costs by provoking deviations from a predetermined and optimized schedule, continuous intervention is necessary.

In the current situation, respective Disruption Management (DM) is typically the task of human operators: They evaluate alternatives with respect to changes of priorities and resource assignments in order to minimize delays and associated penalties, aiming at the maximal efficiency of aircraft turnaround execution. In the process of identifying optimal interventions and schedule modifications, they base their decisions on (1) the available information as well as (2) the qualitative assessment of the current situation (according to individual experience and domain knowledge): Both of these aspects have to be considered if performance improvements are intended.

As regards the former, enhancements in the quality and availability of relevant information are addressed by the concept of Collaborative Decision Making (CDM): Targeting at a higher level of decision quality and overall system performance in the domain of air traffic, operative collaboration between air traffic management, airlines and airports is improved through intensified information sharing and the development of a common situational awareness (see [1], [2] for example): Relevant data is made accessible for all involved stakeholders, at the time and in the quality required. These new forms of shared knowledge implicitly open new possibilities to coordinate decisions and dispositions early and easily.

As regards the latter aspect – the assessment of situations and respective interventions according to individual experience and domain knowledge – particularly the application of intelligent systems provides promising opportunities: Based on concepts of Artificial Intelligence (AI), the available information can be analyzed automatically and therefore form the basis for the proactive provision of decision support in the operative management of aircraft turnaround disruptions. However, only few such applications have yet been developed, even though the increasing application of CDM concepts has led to the emergence of comprehensively integrated databases, which are an important prerequisite of efficient decision support.

With a view to supporting the development of near real-time Decision Support Systems (DSS), we thoroughly analyze and evaluate potential techniques for solving respective problems in this paper: Their applicability, potentials and limitations are assessed before a novel conceptual framework for the realization of operative DSS is introduced. The work presented herein is motivated by the insights and findings of a study conducted in collaboration with Deutsche Lufthansa AG, aiming at the identification of elementary requirements and potentials of turnaround-related decision support.

This paper is organized as follows: Section II gives an overview on typical problems in the application domain and illustrates the concept of disruption management by means of a typical example. Section III discusses different candidate techniques for the basis of future decision support systems: In particular, simulation, planning and scheduling approaches to the development of DSS are presented, respective strengths and weaknesses are described. Based on the findings of this analysis, Section IV introduces a novel approach for using the Resource-Constrained Project Scheduling Problem (RCPSP) as the basis of turnaround-related DSS. Section V summarizes the gained results and gives an outlook on future work.

II. Problem Statement

The process considered as *aircraft turnaround* combines all activities carried out at the airport while an aircraft is on ground. For presentation purposes we will limit ourselves to the following version of the process, which basically corresponds to the combination of core processes as mentioned by Carr [3]: The turnaround starts when the plane reaches its

J. Kuster and D. Jannach are with the Department of Business Informatics and Application Systems of the University of Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria (e-mail: {jkuster, dietmar}@ifit.uni-klu.ac.at)

final gate or stand position. All incoming passengers leave the aircraft (*deboarding*) before it is *fueled*, *cleaned* and *catered* simultaneously. After the end of the last of these activities, the outgoing passengers enter the aircraft (*boarding*). The process finishes when the airplane finally leaves its position, heading for the runway.

Disruptions and unintended modifications of the predetermined schedule typically result from arrival delays, resource breakdowns or unavailabilities, flight cancelations, emergency landings, time slot modifications, bad weather conditions or strikes, for example. Thereby caused activity delays often ripple through the entire process to finally result in departure delays: They thus may even affect processes at remote airports. Since a link between processes conceptually corresponds to a link between process steps, we can consider a simple disruption of a single turnaround for illustrative purposes: We assume an instance of the process, in which a delay occurs already prior to deboarding. It propagates the entire turnaround due to the lack of slack times and finally results in a departure delay. Figure 1 illustrates this process: Grey bars indicate the execution times predicted *before*, white bars the ones predicted *after* the occurrence of the disruption.

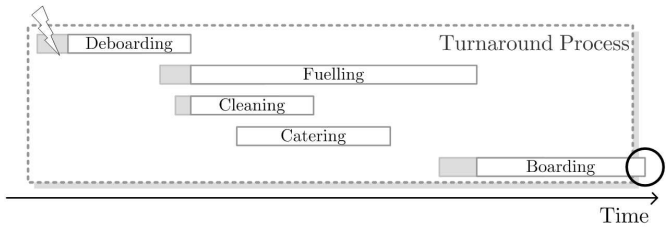

Fig. 1. Simplified Version of the Turnaround Process

In case of disruptions, deciding operators typically have several possibilities to intervene and to re-optimize the given schedule: They might accelerate processes through the assignment of additional resources, quality reductions or the parallelization of activities. They also might change priorities and reschedule affected operations, cancel flights or simply tell the apron workers to hurry up. All in all, their decisions aim at the minimization of delays and inefficiencies. For the regarded instance of a disrupted aircraft turnaround, let us assume to have a set of three potential options available: First, deboarding can be accelerated through the assignment of additional buses. Second, cleaning can be shortened if – in exchange – the cabin is additionally inspected by the cabin crew prior to boarding. And third, fueling and boarding can be parallelized if the fire brigade is available for supervision.

The operator decides on an intervention by selecting either one or a combination of several of these $n = 3$ options. If the schedule contains m processes, which are linked through resource requirements and order constraints, the number of interventions that have to be considered theoretically goes up to 2^{n*m} (including the option of not intervening at all). In realistic scenarios where the number of both possibilities and processes is high, it is therefore impossible for the human operator to analyze all of the given alternatives: Instead, the number of actually considered options is reduced to a reasonable and manageable amount based on intuition and individual experience.

Since premature restrictions of the search space may result in suboptimal decisions, the kind of DSS considered herein is intended to support the operator through the automated filtering and the proposal of relevant forms of interventions: The intelligent system shall therefore assess problematic situations and available options. This represents a complex task due to the following reasons:

- *Process Complexity.* Scheduling the aircraft turnaround can be considered complex due to a high number of dependencies: Processes are interrelated through shared resource requirements and resource flows (consider transfer passengers and cargo). Moreover, their execution depends on the current state of the environment, consisting of the given weather situation, air space restrictions as well as relevant guidelines, business rules and laws, for example.
- *Data Complexity.* Various forms of deficiencies can be distinguished for the information, which is typically available at an airport: *Inaccurateness* means that the available data does not describe reality correctly. *Inconsistency* means that various sources of information describe reality differently. *Incompleteness* means that reality is only partly described by the available data. *Instability* means that changes in the real world can implicate changes of data contents at any time. The minimization of inaccurateness, inconsistency and incompleteness is addressed by the concept of CDM and thereby motivated applications such as Lufthansa's ALLEGRO, which is concerned with the automated collection and aggregation of turnaround-related time stamp information [3]. Although these efforts are intended to finally make more precise and stable predictions possible, handling the inevitable instability remains a principle challenge in the domain of air traffic.
- *Operations Complexity.* The process of decision making is complex by itself. On the one hand, there is a high number of aircraft turnarounds that have to be managed: About 800 aircrafts per day are handled by the Hub Control Center in Frankfurt, Germany, for example. An intelligent system has to regard all of them in context: And ideally, even temporally and locally distant processes are considered if they are potentially affected. On the other hand, only little time is available for the identification of interventions: Minimum ground times are usually only about 35 to 45 minutes in Europe and even lower at US hubs [4]. It is obvious that almost no time remains for the management of disruptions and the disposition of interventions within this short period.

As regards the development of decision support systems for the operative management of turnaround disruptions, the following requirements can therefore be summarized:

- *The underlying modeling concept has to be easy and intuitive.* Since the domain of air traffic is characterized by frequent changes of operative conditions, it is of particular interest to use a model which is intelligible to and therefore maintainable by the final user. It has to support the flexible formulation and modification of process structures, time and resource constraints as well as potential interventions, costs and associated goals.

- *Applied algorithms have to provide results within reasonable response times.* The operative use of a DSS in turnaround disruption management is only possible if interventions are assessed and proposed in near real-time. The system has to find a tradeoff between optimality and computational effort: Identifying the optimal solution is desirable but not imperative.
- *Decision support has to be provided in a proactive way.* The given situation has to be analyzed and evaluated continuously, existing and pending disruptions have to be detected, a proposal on appropriate forms of interventions has to be generated and presented to the user, finally.

It is particularly the former two aspects that have to be considered in the following evaluation of fundamental approaches for the development of automated DSS.

III. Potential Approaches

In this section, simulation, planning and scheduling are presented and evaluated as candidate techniques for the provision of decision support in the operative management of turnaround process disruptions.

A. Simulation

Simulation is the process of designing a simplified model of the real world and conducting experiments on this model, aiming at an improved understanding of the regarded system's behavior or the optimization of its operation. It is typically applied whenever considered systems are too complex to be efficiently modeled through analytical methods: Particularly stochastic and dynamic real-world systems fall into this category. Depending on the points in time at which system state changes may occur, computer-based simulation can be classified into discrete-event, continuous or hybrid forms.

The power of simulation lies in the provided possibility to analyze even highly complex real-world systems in experimental execution: Performance numbers can be estimated for new policies, parameters or operating conditions; alternative designs or scenarios can be compared and evaluated; answers to „what-if" questions can be found. Computer-based simulation makes it possible to predict the behavior of the real-world system under various settings without disturbing actual operations or causing additional costs [5].

However, simulation itself is not a design tool, but is intended to provide the evaluation of a design which is input to the system [6]. For this reason, *simulation optimization* combines the methodologies of simulation and optimization in the search for an optimal system configuration: Given a relationship between input and output parameters which is so complex that simulation is used to estimate the output, it is concerned with the identification of a feasible setting of input variables, which minimizes or maximizes the expected value. The optimization procedure can be based on random search algorithms, sample path optimization or metaheuristics, for example (see [7] for an introductory overview). For a decision support system, which is typically concerned with the identification of an optimum, simulation therefore always has to be combined with some kind of search method.

Computer-based simulation has been applied in a variety of applications which are intended to support decision processes in the domain of airport operations: Gatersleben et al. [8] investigated passenger flows and analyzed the causes of existing bottlenecks at Schiphol Airport in the Netherlands. Similarly, Kiran et al. [9] regarded passenger and aircraft flow for the international terminal at Istanbul Ataturk Airport. Cao et al. [10] evaluated queue structures and working schedules for the check-in system at Ottawa International Airport. Lee et al. [11] analyzed the time requirements for the deicing procedure at Detroit Metro Airport and Schumacher [12] evaluated the effects of runway capacity changes at Atlanta Hartsfield International Airport by means of simulation.

Since one of the main purposes of computer-based simulation is the evaluation of various input parameters and operating policies, it has traditionally primarily been applied during system design and configuration, where typical goals are the identification of bottlenecks, requirements and optimal distributions of resources. The time required for a simulation run is not actually crucial during these early phases of system analysis. As regards the application of respective techniques in operative and near real-time decision support systems, however, the computational overhead which is typically associated with simulation [13] becomes a problem.

Even though the underlying idea of evaluating and comparing different scenarios fits pretty well with the regarded problem of turnaround process disruption management, basing a respective decision support system on simulation optimization may have its limitations:

- What is considered simulation is typically rather concerned with the analysis of artificial cases, where events and system state changes are generated randomly according to some predetermined probability distribution: It rather aims at the analysis of potential future situations than at the evaluation of an actual present situation.
- Simulation is all about abstracting the relationship between input and output variables: Each output value is considered the result of a specific combination of input values, no system-internal dependencies are analyzed. It is rather the experimental execution of a scenario based on a potential setup that is of interest.
- Due to the fact that the system transforming input into output is considered a black box in simulation, optimization can be characterized as trial-and-error procedure [14]: Such approaches are inefficient whenever the internal structure of the problem can reveal information on the reasons of disruptions and appropriate modification options: This is the case for the regarded domain.
- The inefficiency of optimization causes high levels of computational overhead: More combinations than actually required have to be evaluated in separate simulation runs. This clearly contradicts with the posted requirements of maximum efficiency and near real-time performance.

For these reasons, we propose to focus on more analytical approaches in the identification of an optimal form of intervention: Respective methods will be discussed in the following.

B. Planning

Planning is the process of selecting a sequence of actions suitable for the achievement of a previously defined goal: AI Planning is concerned with the development of automated methods for the identification of such sequences (see [15] for an overview). In this context, a planning problem is composed of the descriptions of an initial state, a goal and various actions, used to transform a given system state into another.

The aircraft turnaround is closely related to planning by nature, since already the process of originally assigning starting times to the set of relevant activities falls into the category of (temporal) planning. And also the regarded problem of managing disruptions is exactly what can be considered a planning problem: It is about *planning* interventions to return to an original schedule or about *replanning* all future activities with respect to some predefined optimization criterion. Accordingly, the exemplary case described in Section II can be regarded as the following replanning problem:

- *Relevant Actions.* The set of plannable actions consists of all remaining (i.e. future) process steps as well as all predetermined possibilities of intervention. Therefore, `deboarding`, `fueling`, `cleaning`, `catering` and `boarding` have to be considered as well as the options of assigning additional buses (`additional-buses`), reducing the scale of cleaning (`shortened-cleaning`) or parallelizing fueling and boarding (`parallelize`).
- *Initial State.* The initial state describes the current real-world situation. Since in the regarded example the disruption occurs prior to the start of the first process step, the respective initial state description merely defines that nothing has started yet.
- *Goal.* The goal of the planning process defines that all remaining activities have to be considered in the plan and that some additional optimization criterion has to be met: This auxiliary condition makes sure that optional process modifications are also taken into account. In our example, the goal consists in the identification of a plan, in which the aircraft turnaround ends on time (or at least with a minimum of delay). Moreover, it is of utmost importance in this kind of replanning problem that deviations from the original schedule are avoided as far as possible.

Note that the necessity of considering all future activities for plan generation is due to the fact that the consequences associated with the application of interventions can only be evaluated efficiently if regarded within the context of all actually and potentially affected processes.

Another reason for the consideration of planning as a basic technique is the fact, that during recent years remarkable algorithmic advances have been made in the development of automated planning systems, which can be used to solve the formulated problem: These make especially domain-independent planning applicable to realistic problems in realistic sizes (see [16] for example). Automated planning and respective systems are usually classified into the following categories [17]:

- *Domain-Independent Planning.* Concepts and systems of the most abstract class can be applied to any problem in any domain.
- *Domain-Configurable Planning.* Domain-specific knowledge is encoded directly in the provided model. This approach to planning therefore considers the fact that often a certain idea of how problems can be solved is given. Typically, knowledge concerning the order of activities falls into this category: Regarding the aircraft turnaround domain, it can – for example – be defined that deboarding always has to be executed before catering, which itself has to be finished before boarding starts.
- *Domain-Specific Planning.* Representations and algorithms are optimized for and usually limited to a certain application within a predefined domain.

Since we are particularly interested in more general and reusable approaches, we will focus on the evaluation of the two former categories in the following.

1) Domain-Independent Planning: In this section, potentials and limitations of domain-independent planning will be analyzed. After an introductory overview on formal modeling concepts, it will be discussed if and how certain parts of the regarded problem can be encoded. Respective solvers will be presented briefly before the adequacy of this form of planning for aircraft turnaround DM is summarized.

a) Formalisms and Languages: The Stanford Research Institute Problem Solver (STRIPS) was developed by Fikes and Nilsson [18]: It provides a simple and compact way of expressing planning problems, based on objects (i.e. variables or constants) and simple positive facts (i.e. ground literals without variable symbols), which in conjunction describe world states and goals. STRIPS is particularly famous for its way of action representation: A list of propositions which have to be true forms the set of preconditions, a delete-list describes the propositions which will become false, an add-list the ones which will become true upon action execution.

The Action Description Language (ADL) as proposed by Pednault [19] intends to combine the advantages of the semantically powerful Situation Calculus [20] and the rather syntactically oriented STRIPS notation. For this purpose, STRIPS is extended by negations, quantifiers, disjunctions as well as quantified and conditional effects.

During the last few years, the Planning Domain Definition Language (PDDL), which derived from ADL and several other formalisms, became the de-facto standard language for the formal description of planning problems. In domain definitions, types, constants, axioms, predicates and actions are described whereas problem definitions are used to formulate the initial state and the goal of the planning process. PDDL was introduced in 1998 in its initial version [21] and has been discussed, modified and extended ever since. With version 2.1 [22], numeric fluents, respective plan metrics for quality evaluation and durative actions have been introduced. PDDL 2.2 [23] brought derived predicates as well as timed initial literals, which can be used to express time-triggered state modifications. Finally, version 3.0 of the Planning Domain Definition Language [24] introduced state trajectory constraints as well as preferences (i.e. soft constraints). We will focus on this last and current version, which has been developed for the International Planning Competition (IPC) held in 2006, in the following.

b) Domain and Problem Encoding: The Planning Domain Definition Languages provides many of the required modeling concepts for the description of the aircraft turnaround. Numeric values are available for the dynamic modification of resource availabilities and requirements. Concepts of modeling time make it possible to schedule processes and to regard them within their temporal context: Even conditions and effects can be timed in an elementary way. However, when encoding the turnaround DM problem for the subsequent resolution within an existing domain-independent planning system, several particularities have to be considered. These are discussed in more detail in the following.

The problem that *hierarchical actions* are not supported by PDDL and that it therefore is not possible to explicitly group individual activities in one process `turnaround` can be resolved by the use of auxiliary predicates in the following structure: A process is modeled as a separate activity, which triggers the execution of its contained elements in a start effect and has the (post)condition assigned that the last of its subactivities has to be finished at its end. The resulting abstract action is described in PDDL notation in the following listing: `?p` corresponds to an instance of `process`, for which the `time` function describes the remaining duration. The `ready` and `stepready` literals are true whenever the associated process or process step is ready for execution. Similarly, `finished` and `stepfinished` are true as soon as execution has finished. In the `deboarding` activity, the preconditions must contain a reference to the associated `stepready` literal, and one end effect of the `boarding` activity has to be the activation of the respective `stepfinished` literal.

```
(:durative-action turnaround
 :parameters (?p process)
 :duration (= ?duration (time ?p))
 :condition (and (at start (ready ?p))
                 (at end (stepfinished boarding ?p)))
 :effect    (and (at start (stepready deboarding ?p))
                 (at end (finished ?p))))
```

Disjunctive conditions are required for the support of different activity execution states and varying process structures. As regards the former, consider the distinction between pending and running operations: The action `catering` has to be scheduled immediately if all requirements concerning predecessors and temporal constraints are fulfilled *or* if activity execution has already started. As regards the latter, consider the simple case of process step parallelization: The action `boarding` shall start if `fueling` has been finished *or* if the specific requirements for parallel execution are fulfilled. Although PDDL does not support disjunctions of temporally annotated expressions, disjunctive conditions can be used as long as only one temporal reference is regarded. The following listing illustrates a potential definition of the `catering` action.

```
(:durative-action catering
 :parameters (?p - process)
 :duration (= ?duration (steptime catering ?p))
 :condition (and
    (at start (not(stepfinished catering ?p))
    (at start (or (stepstarted catering ?p)
                  (stepfinished deboarding ?p))))
    (at start (stepready catering ?p)))
 :effect (and (at start(stepstarted catering ?p))
              (at end(stepfinished catering ?p))))
```

Again, the literals `stepready`, `stepstarted` and `stepfinished` are used to describe the current state of an activity. `steptime` quantifies the time required by a certain process step. The first condition in the above example defines that the action `catering` can only be started if it has not already been executed. The second condition states that the activity must either be running or that its predecessor `deboarding` has to be finished. And finally, the last condition defines that the `stepready` literal has to be true for the regarded instance.

Another problem is associated with the question of how the *effects* of an intervention can be handled: If we assume *discrete* costs (which means that all costs are charged for the first step of intervention realization), expenses can simply be associated with the execution of the initial activity of the reparation process. Whereas the assumption of discreteness is mostly acceptable for the costs of interventions, it is typically not for positive effects. The mere assignment of additional buses, for example, is not sufficient for the acceleration of `boarding` and `deboarding` by a predetermined amount of time: Instead, the buses help to accelerate the process *as long as* they are available for the turnaround. The problem of such *continuous* effects can be addressed in various ways:

- *Internal clock.* PDDL itself provides possibilities to describe continuous effects: An internal clock value `#t` can be used within the effect definition of a durative action to dynamically modify fluent values. However, since it is not possible to change an activity's duration after its initialization, this approach is inappropriate for the implementation of process acceleration.
- *Unit-wise Planning.* The logical consequence of this restriction is a reduction in the level of abstraction through the consideration of singular units: Instead of regarding `deboarding` as a rather abstract activity, the exact way of each passenger is therefore planned. However, the high number of relevant entities makes plan generation complex and the difficulty of estimating the duration of individual movements makes plans sensitive and fragile.
- *Reservation.* Another possibility to handle continuous effects is to make them discrete: This can be done by grouping all intervention-related activities into one abstract (and non-preemptive) process, which lasts as long as respective resources are required. Basically, this approach corresponds to the disposition of durative resource reservations. Since, however, it not possible to access start and end times of related (affected) activities, the required duration can not be determined exactly and has thus to be estimated in a generous way, which is (of course) highly inefficient.

The problem of providing decision support in the operative management of turnaround disruptions is a *replanning* problem: It requires the consideration of an existing schedule. Unfortunately, however, PDDL does not support respective plan modifications explicitly. The only thing possible is the definition of temporal constraints by the use of timed initial literals: Release times can be considered if the respective activity depends on the truth value of a specific literal, which

is only activated at a certain point in time. To make sure, for example, that `catering` for turnaround `TA1` does not start before a given time *t*, a timed initial literal `(at` *t* `(stepready catering TA1))` shall be added to the problem description. Note, however, that this is not sufficient for comprehensive replanning, since only minimum or maximum times can be considered this way. A more flexible solution would be the use of `preference` descriptions in the goal definition: But unfortunately, even these PDDL 3.0 specific constructs do not allow full rescheduling, since it is not possible to access activity-related time points for the assessment of plans.

Traditionally, the only criterion of *plan evaluation* was goal compliance. It was only with PDDL 2.1 and the introduction of plan metrics that more flexible quality assessments became possible. In PDDL 3.0 one can finally also define *weak* goals, which shall but do not need to be fulfilled for a plan to be accepted: The `preference` element is used for the description of respective constraints, their violation can be considered by using associated `is-violated` expressions within the `:metric` section. Accordingly, the goal section in a problem definition for aircraft turnaround disruption management should look like the PDDL segment described in the following listing:

```
(:goal (and
   (all-turnarounds-finished)
   (forall (?p - process)
           (preference in-time (punctual ?p)))))
(:metric minimize (+
   (* (is-violated in-time) (cost-per-delay)))
   (* (number-delayminutes) (cost-per-delayminute)))
```

The goal definition consists of two conjunctive parts: Based on an `all-turnarounds-finished` literal, the first one defines the (hard) goal that all processes must be finished within the generated plan. The second one describes the preference `in-time`: For all instances of `process` the literal `punctual` shall be true. And also the plan metric is composed of two parts: The former considers the number of delayed turnarounds (the number of processes for which `punctual` is not true), the latter considers overall delay minutes. For both positions, associated cost values are taken from the `cost-per-delay` or `cost-per-delayminute` literal, respectively.

Whereas timed initial literals can be used as discussed above to identify if a turnaround finishes with delay and to update the truth value of `punctual` accordingly, the number of delay minutes – represented by `number-delayminutes` – can not be determined at all: It would be necessary to have access either to the current time within the action's effect declaration (in order to update the respective fluent) or to the ending time of an activity within the goal and metrics definition. However, neither is supported by PDDL, unfortunately.

c) Problem Solution: Many domain-independent planning systems are available: All of them differ in available search procedures (optimal or suboptimal planning), supported version of PDDL, the extent of implementation of respective concepts or applied algorithms. Since PDDL 3.0 has been released for the upcoming IPC only, no respective planners are yet available. Instead, some of the candidates of the last competition shall be introduced briefly.

- *CPT*[1]. The Constraint Programming Temporal (CPT) planner is an optimal planner for temporal STRIPS: However, neither durative actions with pre-, invariant and post-conditions nor numerics or metrics are supported.
- *HSP*[2]. The planning systems grouped under the name Heuristic Search Planners (HSP*) are optimal planners based on heuristic search methods. They process a PDDL2.1-like input language and can cope with durative actions, numerics and metrics: Only the fact that activity durations must not be defined through dynamically changeable values can make modeling difficult.
- *LPG-td*[3]. The Local Search for Planning Graphs planner with timed initial literals and derived predicates (LPG-td) is one of the few planning systems supporting PDDL 2.2 in its full extent. LPG-td is a suboptimal planner based on stochastic algorithms: Generated plans are therefore not necessarily identical.

d) Summary: PDDL and associated planning systems have been considered particularly due to the fact, that aircraft turnaround disruption management represents a replanning problem by nature: An existing schedule has to be updated and modified according to some real-world disruption. The algorithmic advances that have been made recently represent another argument for the consideration of domain-independent planning. Although the conceptual framework provided by PDDL allows the description of even complex structures and dependencies, there are various peculiarities of the regarded domain which are not currently supported: A major problem is the lack of possibilities to access the current time or start and end time values of contained activities: Especially the planning goal can not be described correctly within these boundaries. Another fundamental problem is the fact that no explicit support for replanning is provided. As regards associated planners, most of the potentially relevant systems are not yet able to cope with PDDL 2.2 in its full complexity. The performance of some of them has been evaluated for the airport ground-traffic control problem in a feasibility study by Trüg et al. [25], who – despite all recently made advances – declared generic planning systems as still too weak for their application to realistic problems in realistic sizes.

2) Domain-Configurable Planning: After the introduction of the concept of Hierarchical Task Networks, several planning systems are presented: The analysis of their strengths and weaknesses forms the basis for the finally provided assessment of domain-configurable planning in the regarded context.

a) Hierarchical Task Networks: In domain-configurable planning the representation of domain-specific knowledge is the core task. The most common approach is to use Hierarchical Task Networks (HTN, see [26], [27] for example), which provide possibilities to describe actions in groups and ordered structures. The following elements shall be introduced:

- A *task network* corresponds to the problem description, defining the set of tasks that have to be executed, associated ordering constraints as well as variable instantiations.

[1]Available at http://www.cril.univ-artois.fr/ vidal/cpt.en.html
[2]Available at http://www.ida.liu.se/∼pahas/hsps
[3]Available at http://zeus.ing.unibs.it/lpg

- Two categories of *tasks* can be distinguished: A *primitive* task corresponds to an action in STRIPS-style planning: It describes a state transition and can be solved by an *operator*. A *non-primitive* task, however, can not be executed directly: Since it may contain several other tasks, the planner has to figure out how to accomplish it best. A *method* is used to describe respective ways of realization.
- A *plan* corresponds to a sequence of ground primitive tasks: It describes how a given task network can be accomplished.

In contrast to STRIPS-style planning, HTN are rather task- than goal oriented: A task network is not restricted to attainment goals. It describes the tasks to execute rather than the states to achieve. HTN planning is therefore based on task decomposition and conflict resolution: Non-primitive tasks are expanded and reduced until a feasible sequence of executable primitive tasks is identified. The fact, that there exists no common language for the definition of HTN represents another difference between domain-independent and domain-configurable planning.

As far as the regarded problem of aircraft turnaround management is concerned, we assume that the finalization of the process represents a non-primitive task (`turnaround`). As such, it can be accomplished in various ways: All contained activities correspond to primitive tasks grouped in the respective method, where interventions may or may not be applied. In SHOP2's (see below) LISP-like notation, the default version of the `turnaround` task might look like described by the following listing, where `ready ?p` represents a precondition:

```
(:method (turnaround ?p)
         (ready ?p)
         ((!deboarding ?p)
          (!fueling ?p)
          (!cleaning ?p)
          (!catering ?p)
          (!boarding ?p)))
```

b) HTN Planning Systems: Domain-configurable planning can be positioned between domain-independent and domain-specific approaches: Domain-independent planning techiques are used to compile a domain-specific planner based on a certain domain description. The following systems are potential candidates for the resolution of HTNs:

- *SHOP*[4]. Due to its notable performance at the IPC 2002 and the continuous emergence of new derivatives (SHOP2 as well as the Java-based JSHOP and JSHOP2), the Simple Hierarchical Order Planner (SHOP) can be considered as one of the most vivid HTN-based planning systems right now. The current version 1.2 of SHOP2 supports many of the features provided by PDDL: Derived predicates, quantifiers, conditional paths and numerics are available. Even late-bound values, external function calls and debugging are possible.
  However, SHOP2 provides no explicit support for time and durations. The Multi-Timeline Preprocessing (MTP) approach [28], which proposes a way of describing durative activities arithmetically, is not sufficient for the regarded context since especially the modeling of resource requirements reveals problems: Whereas it is possible to consider periods of unavailability in the planning procedure, temporal shifts of requiring processes to the earliest period of availability are not.
  While the lack of support for disjunctive preconditions can be resolved by the use of separate `:method` definitions, another elementary drawback is the fact that the Simple Hierarchical Order Planner provides no explicit possibilities of considering existing plans. Moreover, the fact that SHOP2 represents a fully task-oriented planning system has to be kept in mind: It is not possible to resolve operator preconditions automatically.
  As regards its performance, an instance of the problem presented in Section II has been implemented as far as possible: By assigning durations and cost values to all activities and delays, an example in which the `parallelize` option represents the optimal form of intervention has been constructed. On a standard desktop PC, the identification of this optimum took SHOP2 about 2 seconds, which is obviously insufficient for the application of the HTN planner to realistic problems in realistic sizes.
- *HyHTN*[5]. The Hybrid HTN Planner (HTN) combines HTN resolution with fast-forward search: It therefore supports task decomposition as well as precondition achievement. HyHTN processes Object Centered Language (OCLh) descriptions, which provide no support for time and numerics.
- *SIPE-2*[6]. The System for Interactive Planning and Execution (SIPE-2) was developed by the AI center at SRI International. It supports resources and temporal constraints and can be used for execution monitoring, user interventions and replanning. However, the system is a commercial product with no evaluation version available. Moreover, it seems orphaned when the copyright is declared only from 1989 to 1995.
- *O-Plan*[7]. The Open Planning Architecture (O-Plan) provides explicit support for resources and temporal constraints: Precedence relationships, durations and allowed time windows can be assigned to activities. However, also this planning system seems to lie dormant right now.

c) Summary: All in all, the field of domain-configurable planning seems a lot less active than the one of domain-independent planning: Hardly any HTN planning systems have been developed during recent years, the number of respective IPC participants is small. Recent developments focus on planners which basically suffer from the same problems as described for PDDL-based systems: It is mainly a lack of comprehensive support for durative elements, temporal requirements and replanning, which therefore makes the use of domain-configurable planning systems impossible in the regarded context. Moreover, the observed performance indicates that respective planners are currently too weak for the application to realistic problem sizes.

[4]Available at http://www.cs.umd.edu/projects/shop

[5]Available at http://scom.hud.ac.uk/scomdl2/hyhtn/index.html

[6]Available at http://www.ai.sri.com/∼sipe

[7]Available at http://www.aiai.ed.ac.uk/∼oplan

C. Scheduling

Scheduling is the process of assigning times and resources to activities, respecting precedence and availability constraints. Compared to planning, which rather focuses on *what* to do, scheduling is therefore mainly concerned with the question of *when* to take action. Respective concepts have been considered as a basic technique for the DSS of interest since turnaround DM can be regarded as a rescheduling problem: The starting times of future activities have to be updated according to a given real-world disruption. Moreover, the limitations that have been identified for automated planning systems implicitly suggest to focus on rather time-oriented approaches.

1) Conceptual Framework: From the set of available classes of scheduling problems (see [29] for an overview), it is particularly the Resource-Constrained Project Scheduling Problem (RCPSP) which shall be considered for modeling the aircraft turnaround domain: As a generalization of the job-shop scheduling problem it provides possibilities to define arbitrarily linked activities which are processed on arbitrary resource types. It can be formalized as follows: A project consists of a set of activities $\mathcal{A} = \{0, 1, ..., a, a+1\}$: The first and the last element correspond to fictitious start and end activities, which require no resources and have a duration of 0. Each remaining $i \in \mathcal{A}$ has a non-negative duration d_i assigned. For activity execution, a set of renewable resource types $\mathcal{R} = \{1, ..., r\}$ is available, with a constant amount of u_k units disposable for a type k. Activities are ordered by a set of precedence constraints $\mathcal{P}$: The existence of $p_{i,j} \in \mathcal{P}$ states that activity j must not start before the end of activity i. The relationship between activities and resource types is defined by a set of resource requirements $\mathcal{Q}$: Activity i requires $q_{i,k} \in \mathcal{Q}$ units of resource type k throughout its execution.

Describing an RCPSP is easy and intuitive: Activities, resources and constraints represent abstract constructs which allow the definition of problems on a conceptual level. In this context, the turnaround process can be described as a resource-constrained project, with process steps modeled as ordered activities. Unfortunately, however, the description of available interventions is not possible within the Resource-Constrained Project Scheduling Problem: Neither optional activities nor dynamic structural modifications are supported.

2) Solution Methodologies: For the resolution of scheduling problems, basically the following methodologies can be distinguished (for comprehensive overviews see [30], [31]):

- *Mathematical Programming (MP).* Particularly early research focused on the identification of optimal solutions based on MP: Integer Programming [32], Dynamic Programming [33] and Branch-and-Bound procedures [34], [35] have been applied, for example. Integer linear formulations of the RCPSP are mostly based on time-indexed variables, where one decision variable is necessary for any potential combination of activity and starting time [36]: This makes problems large and implies high computational requirements. Another difficulty results from the underlying low-level formalism: Modeling usually requires deep mathematical knowledge and resulting models are quite hard to understand.
- *Constraint Programming (CP).* In Constraint Programming, problems are modeled as Constraint Satisfaction Problems (CSP): Respective instances contain a set of variables, associated domains, and a set of constraints describing the relationships between the variables. In the process of searching an optimal solution, constraint propagation and consistency enforcing techniques can be applied: The domains of the contained variables are reduced as far as possible in a preprocessing step. Afterwards, particularly techniques of Artificial Intelligence and (increasingly) of Operations Research are used for the identification of an optimum (see [37], for example). Scheduling problems can be expressed as CSP based on precedence and resource constraints: Compared to mathematic formulations, Constraint-Based Scheduling (CBS, see [38] for example) thus provides possibilities of higher level modeling, which means that the complexity of defining and maintaining domain models is significantly reduced. However, the problem of rapidly increasing problem sizes remains, due to the common approach of applying time-indexed variables: Computational efforts remain high for the identification of optimal solutions.
- *Heuristic Procedures.* The combinatorial nature of the RCPSP makes the determination of exact optimal solutions difficult: The successful application of such optimization approaches and the resolution of problems within reasonable time have only been reported for relatively small problem sizes. By contrast, Heuristic Procedures rather focus on the generation of *good* solutions in shorter time. They make it possible to dynamically trade off between solution quality and the time spent searching. In the context of the RCPSP, it is particularly an abstraction of temporal concepts, which makes remarkable reductions of the search space possible: Instead of working directly with time values, a common approach is to use abstract representations of the schedule (activity lists, random keys, etc.), which can be optimized more easily [39]. As regards the optimization procedure itself, various forms of metaheuristics have been implemented: Tabu search, simulated annealing and genetic algorithms represent only examples, comprehensive overviews and evaluations are available (as in [40] for example).

For the problem of providing decision support in turnaround DM, the requirement of near real-time performance clearly suggests the use of heuristic procedures.

IV. A Framework for Near Real-Time Decision Support Systems

According to the evaluation results, we propose to base near real-time decision support systems in the operative management of turnaround disruptions on the concept of the Resource-Constrained Project Scheduling Problem. It is considered the most promising approach due to the following reasons:

- The RCPSP provides explicit support for many of the required concepts: It is possible to model time, resources and respective constraints in an easy and intuitive way.

- Heuristic and metaheuristic procedures can be used for the identification and optimization of solutions. Incremental search makes it possible to provide good results even in minimal time. This corresponds to the requirements of operative disruption management, where short performance times are rather of interest than global optima.
- The RCPSP has been and still is subject to extensive research: Respective work focuses on conceptual extensions as well as the further improvement of algorithms.

In this section, we propose an approach to overcome the previously mentioned limitations of the RCPSP. It is shown, how the existing modeling concept can be extended to support optional activities and structural process modifications before some remarks on resolution methodologies are made.

A. Extending the RCPSP

It has already been mentioned that the RCPSP is not sufficient for the description of optional process modifications. Therefore, we propose its generalization in the Extended Resource-Constrained Project Scheduling Problem (*x-RCPSP*), where available interventions are described by means of alternative activities: The application of an intervention corresponds to the switch from a default version of an activity to one of its alternatives. Accordingly, only the subset of currently chosen (*active*) activities is considered and contained in the final schedule.

The *x-RCPSP* is based on the idea of placing an abstract layer on top of the elementary constructs of the classical RCPSP: $\mathcal{A}^+$, $\mathcal{P}^+$ and $\mathcal{Q}^+$ contain all potentially relevant activities, precedence constraints and resource requirements. From these supersets, concrete instances of the RCPSP can be generated: $\mathcal{A} \subseteq \mathcal{A}^+$, $\mathcal{P} \subseteq \mathcal{P}^+$ and $\mathcal{Q} \subseteq \mathcal{Q}^+$ group actually relevant elements. In their instantiation process, a choice on a set of active activities is made before all associated precedence constraints and resource requirements are activated. The modification of the activation state is driven by three sets:

- $\mathcal{A}^0$ describes the reference process: Contained activities are activated by default and are preferred over others.
- $\mathcal{X}^+$ describes valid activity substitutions: The existence of $x_{i,j} \in \mathcal{X}^+$ states that activity i can be replaced by activity j. Note, that this relationship is not commutative.
- $\mathcal{M}^+$ describes mutual dependencies: The existence of an element $m^{\ominus}_{i,j} \in \mathcal{M}^+$ states that activity j has to be deactivated upon the activation of i whereas the existence of $m^{\oplus}_{i,j} \in \mathcal{M}^+$ states that j has always to be activated along with activity i.

Based on these constructs, the aircraft turnaround can be formalized as is summarized in Table I: $Start$ and End correspond to fictious start and end activities, all other process steps are represented by the first three letters of their name. Deb^{Bus} is the accelerated alternative to deboarding, as is Cle^{Red} to cleaning. Ins is the activity of cabin inspection by the crew and Fue^{Par} is the parallelized version of fueling. As regards the notation of relations, $i \rightarrow j$ defines that $p_{i,j} \in \mathcal{P}^+$, $i \triangleright n \times k$ defines that $q_{i,k} = n \in \mathcal{Q}^+$, $i \Rightarrow j$ defines that $x_{i,j} \in \mathcal{X}^+$, $i \Leftrightarrow j$ defines that $x_{i,j}, x_{j,i} \in \mathcal{X}^+$, $i \oplus j$ defines that $m^{\oplus}_{i,j} \in \mathcal{M}^+$ and $i \ominus j$ defines that $m^{\ominus}_{i,j} \in \mathcal{M}^+$.

TABLE I
FORMAL DESCRIPTION OF THE EXEMPLARY TURNAROUND PROCESS

Set	Content
$\mathcal{R}$	$Bus, Firebrigade$
$\mathcal{A}^0$	$Start, Deb, Fue, Cat, Cle, Boa, End$
$\mathcal{A}^+$	$Start, Deb, Deb^{Bus}, Fue, Fue^{Par}, Cat, Cle, Cle^{Red}, Ins, Boa, End$
$\mathcal{P}^+$	$Start \rightarrow Deb$, $Start \rightarrow Deb^{Bus}$, $Deb \rightarrow Fue$, $Deb \rightarrow Fue^{Par}$, $Deb \rightarrow Cat$, $Deb \rightarrow Cle$, $Deb \rightarrow Cle^{Red}$, $Deb^{Bus} \rightarrow Fue$, $Deb^{Bus} \rightarrow Fue^{Par}$, $Deb^{Bus} \rightarrow Cat$, $Deb^{Bus} \rightarrow Cle$, $Deb^{Bus} \rightarrow Cle^{Red}$, $Fue \rightarrow Boa$, $Fue^{Par} \rightarrow End$, $Cat \rightarrow Boa$, $Cle \rightarrow Boa$, $Cle^{Red} \rightarrow Ins$, $Ins \rightarrow Boa$, $Boa \rightarrow End$
$\mathcal{Q}^+$	$Deb \triangleright 1 \times Bus$, $Deb^{Bus} \triangleright 2 \times Bus$, $Fue^{Par} \triangleright 1 \times Firebrigade$
$\mathcal{X}^+$	$Deb \Leftrightarrow Deb^{Bus}$, $Fue \Leftrightarrow Fue^{Par}$, $Cle \Leftrightarrow Cle^{Red}$
$\mathcal{M}^+$	$Cle^{Red} \oplus Ins$, $Cle \ominus Ins$

The knowledge on available interventions is therefore encoded directly within the domain model. The option of assigning additional buses can be applied by switching from Deb to Deb^{Bus}, as made possible by the existence of $Deb \Leftrightarrow Deb^{Bus}$ in $\mathcal{X}^+$. For the switch from Cle to Cle^{Red}, the set of mutual dependencies $\mathcal{M}^+$ defines that the *optional activity* of cabin inspection has to be activated. And the *structural modification* of parallelizing fueling and boarding corresponds to a simple switch from a default version Fue to an alternative Fue^{Par}, that is not linked to boarding through a precedence relation.

B. Problem Resolution

The aim of DM is the identification of an optimal set of interventions: Potential options include both mere temporal shifts of activities as well as structural modifications. While the former kind is covered by the existing RCPSP-specific optimization procedures, it is particularly its combination with the latter type of intervention that has to be addressed in the *x-RCPSP*. Apart from responding to the question of *when* to start activities, it has also to be considered *what* steps to execute at all: Planning has to be combined with scheduling.

For this purpose and a first evaluation of the proposed extensions, we have adopted an evolutionary algorithm for the heuristic resolution of the RCPSP [41] to our specific requirements: As a metaheuristic approach, the respective procedure makes it possible to identify good solutions even in short time. Optimization is accomplished through the continuous evolution of a population (consisting of different solutions): Based on the idea that combinations of good solutions might result in even better ones, the best solutions are selected, recombined and slightly modified in each iteration. Recombination (so-called *crossover*) is based on two parents, with one of them prescribing the activation state of the child and the other one defining the order of contained activities. Modification (so-called *mutation*) is based on the random shift or exchange of the child's active process steps.

On a first prototype implementation, 20 instances of the exemplary process have been regarded in context: The best solution found within 4 seconds on a standard PC was only about 25% below the theoretical optimum. These results are promising, particularly since the current implementation is still unoptimized and does not consider domain-specific knowledge. For further details see [42].

V. Conclusion

In this paper, potential approaches to the problem of providing decision support in the operative management of aircraft turnaround disruptions have been evaluated: Potentials and limitations of simulation, planning and scheduling approaches have been discussed and assessed in this context. Finally, it has been shown how the RCPSP – as the most promising approach in our view – can be extended by support for optional activities and alternative process execution paths for its application to the regarded problem: The respective approach to modeling interventions can easily be adopted to other problem classes.

Future work will be directed at the conduction of additional measurements, the further enhancement of the discussed optimization procedures and the development of a first prototype version of a respective decision support system: Results from practical testing and a final evaluation based on real-world test cases from industrial partners will be provided.

Acknowledgment

This work is part of the *cdm@airports* project, which is carried out in cooperation with FREQUENTIS GmbH (Austria) and is partly funded by grants from FFF (Austria).

References

[1] R. Hoffman, M. Ball, A. Odoni, W. Hall, and M. Wambsganss, "Collaborative decision making in air traffic flow management," UC Berkeley, Tech. Rep., 1999.

[2] M. Wambsganss, "Collaborative decision making through dynamic information transfer," *Air Traffic Control Quarterly*, vol. 4, pp. 107–123, 1997.

[3] F. R. Carr, "Robust decision support tools for airport surface traffic," Ph.D. dissertation, 2004.

[4] C.-L. Wu and R. E. Caves, "Modelling and optimization of aircraft turnaround time at an airport," *Transportation Planning & Technology*, vol. 27, no. 1, pp. 47–66, 2004.

[5] A. M. Law and W. D. Kelton, *Simulation Modelling and Analysis*. McGraw-Hill, 1982.

[6] M. Mollaghasemi and G. Evans, "Multicriteria design of manufacturing systems through simulation optimization," *IEEE Transactions on System, Man, and Cybernetics*, vol. 24, no. 9, 1994.

[7] M. C. Fu, F. W. Glover, and J. April, "Simulation optimization: A review, new developments, and applications," in *Proceedings of the 2005 Winter Simulation Conference*, 2005, pp. 83–95.

[8] M. R. Gatersleben and S. W. van der Weij, "Analysis and simulation of passenger flows in an airport terminal," in *Proceedings of the 1999 Winter Simulation Conference*, 1999, pp. 1226–1231.

[9] A. S. Kiran, T. Cetinkaya, and S. Og, "Simulation modeling and analysis of a new international terminal," in *Proceedings of the 2000 Winter Simulation Conference*, 2000, pp. 1168–1172.

[10] Y. Cao, A. L. Nsakanda, and I. Pressman, "A simulation study of the passenger check in system at the ottawa international airport," in *Proceedings of the 2003 Summer Computer Simulation Conference*, 2003, pp. 573–579.

[11] J. T. Lee, S. Chen, and A. Daskalakis, "Deicing decision support tool," in *Digital Avionics Systems Conference*, 2004, pp. 2.E.6.1–2.E.6.9.

[12] B. Schumacher, "Proactive flight schedule evaluation at delta air lines," in *Proceedings of the 1999 Winter Simulation Conference*, 1999, pp. 1232–1237.

[13] C. M. Harmonosky and S. F. Robohn, "Investigating the application potential of simulation for real time control decisions," *International Journal of Computer Integrated Manufacturing*, vol. 8, p. 126132, 1995.

[14] J. Vankateswaran and Y. J. Son, "Impact of modelling approximations in supply chain analysis - an experimental study," *International Journal of Production Research*, vol. 42, no. 15, pp. 2971–2992, 2004.

[15] J. Rintanen and J. Hoffmann, "An overview of recent algorithms for ai planning," *Künstliche Intelligenz*, vol. 2, pp. 5–11, 2001.

[16] D. S. Weld, "Recent advances in planning," *AI Magazine*, vol. 20, no. 2, pp. 93–123, 1999.

[17] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: theory and practice*. Morgan Kaufmann Publishers, 2004.

[18] R. E. Fikes and N. J. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

[19] E. P. D. Pednault, "ADL: exploring the middle ground between STRIPS and the situation calculus," in *Proceedings of the first international conference on Principles of knowledge representation and reasoning*. Morgan Kaufmann Publishers Inc., 1989, pp. 324–332.

[20] J. McCarthy, "Situations, actions and causal laws," Stanford University, Tech. Rep., 1963, reprinted in: *Semantic Information Processing*, M. Minsky, Ed. MIT Press, 1968, pp. 410-417.

[21] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL—the planning domain definition language," Yale University, Tech. Rep., 1998.

[22] M. Fox and D. Long, "PDDL 2.1: An extension to PDDL for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[23] S. Edelkamp and J. Hoffmann, "PDDL 2.2: The language for the classical part of the 4th international planning competition," Tech. Rep., 2004.

[24] A. Gerevini and D. Long, "Plan constraints and preferences in PDDL3 - the language of the fifth international planning competition," University of Brescia, Italy, Tech. Rep., 2005.

[25] S. Trüg, J. Hoffmann, and B. Nebel, "Applying automatic planning systems to airport ground-traffic control a feasibility study," in *KI 2004*, ser. Lecture Notes in Computer Science 3238, S. Biundo, T. Frühwirth, and G. Palm, Eds. Springer Verlag, 2004, pp. 183–197.

[26] E. D. Sacerdoti, *A Structure jor Plans and Behavior*. Elsevier Scientific Publishing, 1977.

[27] K. Erol, J. Hendler, and D. S. Nau, "Semantics for hierarchical task-network planning," Computer Science Department, University of Maryland, Tech. Rep., 1994.

[28] D. Nau, T.-C. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN planning system," *Journal on Artificial Intelligence Research*, vol. 20, 2003.

[29] P. Brucker, *Complex Scheduling Problems*, ser. Osnabrücker Schriften zur Mathematik, 1999, no. 214.

[30] O. Icmeli, S. S. Erenguc, and C. J. Zappe, "Project scheduling problems: A survey," *International Journal of Operations & Production Management*, no. 11, pp. 80–91, 1993.

[31] W. Herroelen, B. D. Reyck, and E. Demeulemeester, "Resource-constrained project scheduling: a survey of recent developments," *Computers and Operations Research*, vol. 25, no. 4, pp. 279–302, 1998.

[32] A. A. B. Pritsker, L. J. Watters, and P. M. Wolfe, "Multiproject scheduling with limited resources: A zero-one programming approach," *Management Science*, vol. 16, no. 1, pp. 93–108, September 1969.

[33] S. E. Elmaghraby, "Resource allocation via dynamic programming in activity networks," *European Journal of Operational Research*, vol. 64, no. 2, pp. 199–215, 1993.

[34] J. Stinson, E. Davis, and B. Khumawala, "Multiple resource-constrained scheduling using branch-and-bound," *AIIE Transactions*, vol. 10, no. 3, pp. 252–259, 1978.

[35] E. Demeulemeester and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem," *Management Science*, vol. 38, no. 12, pp. 1803–1818, December 1992.

[36] S. Demassey, C. Artigues, and P. Michelon, "Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem," *Journal on Computing*, vol. 17, pp. 52–65, 2005.

[37] P. Baptiste, C. L. Pape, and W. Nuijten, "Incorporating efficient operations research algorithms in constraint-based scheduling," in *1^{st} Joint Workshop on Artificial Intelligence and Operational Research*, 1995.

[38] P. Baptiste, C. LePape, and W. Nuijten, *Constraint-Based Scheduling*. Kluwer Academic Publishers, 2001.

[39] R. Kolisch and S. Hartmann, "Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis," in *Project scheduling: Recent models, algorithms, and applications*, J. Weglarz, Ed., 1999, pp. 147–178.

[40] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for resource constrained project scheduling," *European Journal for Operational Research*, vol. 127, no. 2, pp. 394–407, 2000.

[41] K. Hindi, H. Yang, and K. Fleszar, "An evolutionary algorithm for resource-constrained project scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 512–518, 2002.

[42] J. Kuster and D. Jannach, "Extending the resource-constrained project scheduling problem for disruption management," in *Proceedings of the 3rd IEEE Conference on Intelligent Systems*, 2006, to appear.