Extending the Resource-Constrained Project Scheduling Problem for Disruption Management

Jürgen Kuster, Dietmar Jannach

Abstract— **This paper describes how the Resource-Constrained Project Scheduling Problem (RCPSP) can be used as a basis for comprehensive disruption management, concerned with both rescheduling as well as potential structural process modifications. It is illustrated, how the RCPSP can be extended by the possibility to represent alternative activities and how the respective constructs can be used to describe various forms of typical interventions. Moreover, an approach for schedule optimization and the resolution of the generalized problem is presented, based on the combination of well-established methodologies and specific evolutionary operators. In an illustrative example it is finally shown how the proposed framework can be applied for the development of real-time decision support systems in the domain of airport ground process management.**

Index Terms— **Real-Time Decision Support, Disruption Management, Resource-Constrained Project Scheduling Problem, Evolutionary Algorithm**

I. Introduction

Uncertainty is an intrinsic and pervasive aspect of the real-world [1]. Whenever it unfolds, deviations from a predetermined plan are likely: A so-called disruption occurs. Disruption management (DM, see [1], [2]) is concerned with the resolution of respective problems and the continuous optimization of the relationship between real and planned processes: Since predetermined schedules and plans are typically optimized according to some specific criterion, the main aim is to get back on track in case of process disturbances and to minimize associated costs. For this purpose, an optimal combination of applicable interventions has to be selected from a set of potential ones. Typically, both *rescheduling* as well as *structural process modifications* have to be considered in the resolution of real-world problems.

However, the currently existing applications of disruption management mainly focus on the rescheduling part of the problem: They are concerned with the mere temporal shift of activities within a schedule. Even though especially Zhu et al. [3] introduce basic aspects of structural flexibility when considering mode alternations (i.e. the change of the durations and the amount of resource usage of a specific activity) as a potential form of intervention, we claim that this is still not sufficient for the effective provision of decision support in realistic problems. Apart from the temporal shift and the parametric modification of activities, the responsible decision maker might want to insert or remove process steps, change their order or parallelize what has been planned for serial execution (or vice versa).

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

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

Research on disruption management is strongly driven by operations research and thus focusing on the application of mathematical programming. Although the use of respective methods makes it possible to identify exact optimal solutions, its main drawback compared to (only suboptimal) metaheuristic approaches is the significantly higher requirement of processing time: Therefore, the respective methods can only be applied to relatively small problems, if real-time results are required [4].

The work presented herein is motivated by the findings of a study conducted in collaboration with Deutsche Lufthansa AG, regarding the elementary requirements of DM-related decision support systems (DSS): We propose a novel approach to disruption management, considering both rescheduling and structural modifications as potential interventions in real-time DSS. For this purpose, the notion of alternative activities is introduced for the description of structural modifications and metaheuristic optimization is applied to an accordingly extended version of the Resource-Constrained Project Scheduling Problem (RCPSP). The remainder of this document is structured as follows: Section II introduces a framework for the formal description of potential process modifications. Section III shows how respective problems can be solved: Well-established methodologies are combined with specific methods based on the concepts of evolutionary algorithms. Section IV provides an illustrative example for the application of the introduced concepts from the domain of airport ground process management. Finally, Section V summarizes the contributions of this paper and gives an outlook on future work.

II. Modeling Process Interventions

This section describes how potential interventions can be modeled formally. With a particular focus on structural process modifications, the concept of alternative activities is introduced and used as the basis for an extension of the RCPSP. It is also illustrated how the associated constructs can be used for describing typical forms of intervention.

A. Overview

Comprehensive disruption management has to consider both rescheduling and structural modifications as potential repair activities. Possibilities of the former type are typically implicitly given by the definition of precedence relations and associated resource constraints. Any conceptual framework for schedule optimization can therefore form the basis for the identification of optimal rescheduling interventions. However, as far as structural modifications are concerned, additional modeling is usually required. Two different strategies can be distinguished:

• *External Description.* Potential structural modifications are detached from the process model. An intervention is described through sets of activities and constraints, which have to be added to or deleted from the originally planned process. Therefore, respective *add* and *delete lists* represent the elementary constructs for this form of representation. Consider for example a simple network of two activities a and b which are linked by a precedence constraint saying that a has to be finished at or before the start of b. If we assume that the parallelization of the activities represents a potential modification, this intervention corresponds to the removal of the single precedence relation. If alternatively it shall be possible to insert an activity c between a and b, the respective intervention corresponds to (1) the addition of c to the network and (2) the replacement of the existing precedence relation by two constraints defining that a needs to be executed before c which itself is executed before b.

• *Internal Description.* Potential structural modifications are contained within the process model. Given the existence of various available process variants, an intervention corresponds to the switch from a previously chosen path to a valid alternative. Therefore, *alternative activities*, which correspond to the starting points of variable process execution paths, represent the elementary constructs for this form of representation. Considering the example discussed above, the former option can be described by introducing a choice point into the reference (i.e. the default version of the) process, where it is possible to select a or an alternative a_1 for execution: a_1 differs from a in not being linked with b through a precedence relation. For the latter option, another alternative a_2 is inserted, which represents the origin for the sequence a_2 before c before b.

Upon changes in the process model, respective interventions have to be updated accordingly if potential modifications are regarded as separated from the processes (as with an external form of description): This synchronization represents a highly sensitive task since any mistake may cause inconsistencies in the disruption management problem. If alternatively only one model is used for the description of reference process and valid modification possibilities (as with an internal form of description), a potential drawback consists in the higher level of complexity associated with the description and maintenance of the process structure. In exchange, however, the difficulty of model synchronization can be avoided as consistency is guaranteed implicitly. Since we assume that this represents a major advantage for realistic applications with flexibly and dynamically changing processes, and that the increased model complexity can be efficiently handled through the provision of appropriate (abstract) modeling constructs, we will focus on the latter form of representation in the following.

B. Extending the RCPSP

The Resource-Constrained Project Scheduling Problem provides a well-established framework for the resolution of scheduling problems. In the regarded context, we claim that it is perfectly suited for the resolution of the *rescheduling* part of the problem, due to the following reasons:

• Defining an RCPSP is easy and intuitive. Based on abstract constructs such as activities, resources, precedence constraints, etc. it is possible to define entities and relationships on a conceptual level. Therefore, especially maintainability is significantly better than for comprehensive mathematical models.

• Metaheuristic approaches can be used for optimization. Local search, tabu search, genetic algorithms, ant colonies, etc. can be used to search for solutions. By the use of such incremental search procedures the provision of good results is even possible in real-time: This corresponds to the realistic requirements of disruption management, where decisions must rather be made in short time than in a (globally) optimal manner.

• The RCPSP has been and still is studied extensively. Research particularly focuses on the (further) improvement of optimization algorithms and the extension and generalization of the respective modeling concepts.

However, as far as structural flexibility is concerned, the RCPSP provides only little support. The only form in which respective modifications are possible, is the alternation of modes in the Multi-mode RCPSP (MRCPSP, see [5]): This generalization of the classical scheduling problem makes it possible to dynamically consider changes in durations and in the amounts of required resources. Moreover, Artigues et al. [6] and Elkhyari et al. [7] recently presented their ideas of providing the RCPSP with additional flexibility: The former focus on the dynamic insertion of activities, considering each arrival of an unexpected activity a disruption. The latter use explanations to handle over-constrained networks in dynamic scheduling problems. As regards the concept of alternative activities, only Beck et al. [8] have considered options of activity replacement in scheduling problems: Their approach is based on the association of a Probability of Existence (PEX) with any activity.

In a more general approach, we introduce a method for describing alternative activities (as discussed before) within the conceptual framework of the RCPSP. For this purpose, we define the Extended Resource-Constrained Project Scheduling Problem *x-RCPSP* as a generalization of the classical problem. The basic idea of the extension is a distinction between *active* and *inactive* elements, all grouped and described in a comprehensive model, where only the former group of elements is actually considered in the scheduling process. Thus, the *x-RCPSP* is based on the introduction of an additional layer on top of the original RCPSP: Depending on the current state of element *activation*, different instances of the classical problem can be generated from the respective supersets. This way, the well-established methods which have been defined for the resolution of the RCPSP [9] can be applied for the generation of valid schedules. For the *x-RCPSP*, the aim of optimization is the identification of an optimal *activation* state as well as the identification of an optimal sequence for all *active* activities. Note that each change of the *activation* state corresponds to the selection of an alternative process execution path.

The structure of the *x-RCPSP* can be described as follows. A project (or process respectively) is defined by a set of potential activities $\mathcal{A}^+ = \{\mathit{0}, \mathit{1}, ..., a, a+\mathit{1}\}$. The first and the last element correspond to abstract start and end activities having a duration of 0 and no resource requirements associated. All *active* activities form a subset $\mathcal{A} \subseteq \mathcal{A}^+$ which implies that all *inactive* activities are contained in $\mathcal{A}^+ \setminus \mathcal{A}$. The activities grouped in $\mathcal{A}^{\mathit{0}} \subseteq \mathcal{A}^+$ form the so-called reference process: This subset defines the default *activation* state and the preferred version of the process which is considered before a disruption occurs. The execution of the respective activities is based on a set of non-renewable resource types $\mathcal{R} = \{\mathit{1}, ..., r\}$. For each type k, a constant amount of u_k units is available. As regards the description of activity dependencies, the following constructs can be used:

- *Duration Value.* Each activity i has an duration d_i associated, describing how long its execution lasts.
- *Precedence Constraints.* Activities can be ordered by the use of precedence constraints: The existence of $p_{i,j}$ states that activity i has to be finished at or before the start of activity j. According to the distinction between *active* and *inactive* activities, two different sets are used for grouping precedence relations: $\mathcal{P}^+$ contains all potentially relevant constraints, whereas the subset $\mathcal{P} \subseteq \mathcal{P}^+$ groups only those $p_{i,j}$ for which both i and j are contained in the set of *active* activities $\mathcal{A}$.
- *Resource Requirements.* The relationship between activities and resource types is defined through resource requirements: An activity i requires $q_{i,k}$ units of type k throughout its execution. $\mathcal{Q}^+$ combines all potential dependencies of elements in $\mathcal{A}^+$ on elements in $\mathcal{R}$ whereas $\mathcal{Q}$ only comprises those requirements $q_{i,k}$ for which the associated activity i is currently *active*.

The set of *active* activities represents the most important of all subsets: Whenever a change occurs therein, the set of *active* precedence constraints and resource requirements have to be synchronized accordingly. Li et al. [10] have argued the necessity to consider mutual dependencies when regarding alternative *resources*: We therefore use the following relationships to define modification possibilities:

- *Activity Alternatives $\mathcal{X}^+$.* An activity i can be *deactivated* upon the *activation* of activity j if $x_{i,j}$ is contained within this set. It therefore describes the elementary possibilities of activity substitution. Note that the respective relationship is not necessarily commutative since the option of replacing activity i with activity j does not automatically imply the possibility to substitute j with i.
- *Mutual Dependencies $\mathcal{M}^+$.* Changing the state of an activity might have an impact on other activities. For this purpose, this set describes two types of binary and non-commutative relationships between the elements of $\mathcal{A}^+$: Mutual *exclusion* is described through the addition of an element $m^{\ominus}_{i,j}$, which defines that activity j shall be removed from the schedule upon *activation* of activity i. Correspondingly, mutual *inclusion* is described through the addition of an element $m^{\oplus}_{i,j}$, which defines that j is always added to $\mathcal{A}^+$ along with i.

The *x-RCPSP* represents a generalization of the classical RCPSP: Any instance of an *x-RCPSP* with $\mathcal{X}^+ = \mathcal{M}^+ = \emptyset$ can be converted into an equivalent RCPSP. Correspondingly, the methodologies for the resolution of the classical problem can be applied to our generalization, as soon as $\mathcal{A}$ is stable. Furthermore, the Extended Resource-Constrained Project Scheduling Problem generalizes the Multi-Mode RCPSP: An *x-RCPSP* can be converted into an MRCPSP if all of the following statements are true:

$$\exists x_{j,i} : x_{j,i} \in \mathcal{X}^+ \qquad \forall x_{i,j} \in \mathcal{X}^+ \tag{1}$$

$$\exists x_{i,k} : x_{i,k} \in \mathcal{X}^+ \qquad \forall x_{i,j}, x_{j,k} \in \mathcal{X}^+ \tag{2}$$

$$\begin{array}{ll} \exists p_{j,k} : p_{j,k} \in \mathcal{P}^+ & \forall x_{i,j} \in \mathcal{X}^+, p_{i,k} \in \mathcal{P}^+ \\ \exists p_{k,j} : p_{k,j} \in \mathcal{P}^+ & \forall x_{i,j} \in \mathcal{X}^+, p_{k,i} \in \mathcal{P}^+ \end{array} \tag{3}$$

$$\mathcal{M}^+ = \emptyset \tag{4}$$

Statement (1) means that any potential exchange of activities can be inverted directly. The possibility of returning to an original version is never restricted to cyclic paths only. Statement (2) defines the requirement that all indirectly reachable alternatives can also be reached directly. Switching from one alternative to another one must never require a detour. Statement (3) says that all exchangeable activities have the same sets of preceding and succeeding activities assigned. They are located at exactly the same position in the process. Statement (4), finally, defines that the set of mutual dependencies must be empty. The number of elements in $\mathcal{A}$ is therefore constant.

C. Modification Patterns

This section illustrates how the concept of alternative activities can be used for the description of typical forms of process modification. In the following, mode alternation, resource alternation/capacity change, activity insertion/removal, order switch and serialization/parallelization are discussed. For improved readability a simplified form of notation is used in the following: $i \rightarrow j$ defines that $p_{i,j} \in \mathcal{P}^+$, $i \rhd n \times k$ 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 $m^{\ominus}_{i,j} \in \mathcal{M}^+$.

C.1 Mode Alternation

An activity *mode* can be defined as a fixed combination of duration and resource requirements [5]. Activities, for which different execution modes are available, are considered *multi-mode*. A mode *alternation* corresponds to the switch from a previously chosen mode to another one. In the context of the RCPSP, it is a particular characteristic of the MRCPSP to be able to optimize the current mode selection along with the activity sequence. It has already been discussed under which circumstances the *x-RCPSP* can be converted into the more specific Multi-mode RCPSP (see Section II-B). In this section, it is shown how potential mode alternations can be described within the *x-RCPSP*.

For this purpose, we consider an original network of three activities a, b and c, forming a sequence of alphabetical order. The mode of activity b shall be variable: Mode α

corresponds to the original version, mode β to a slower but less resource-intense version and mode γ, finally, to a faster version requiring more resources. To describe the possibility of changing the execution mode of an activity, we insert one alternative activity per option into $\mathcal{A}^+$: Instead of having b in the network we thus distinguish b_α, b_β and b_γ. All alternatives are based on the original version of the activity: The associated precedence relations are identical and differences merely concern durations and resource requirements. The execution of a mode alternation corresponds to the exchange of an activity with another one. It is necessary to describe the respective possibilities in $\mathcal{X}^+$, where the option of switching any pair of alternatives has to be defined. For the considered example, Table I compares the original with the accordingly modified network.

TABLE I
MODE ALTERNATION IN THE *x-RCPSP*.

	Original	Modified Network
$\mathcal{A}^+$	a, b, c	$a, b_\alpha, b_\beta, b_\gamma, c$
$\mathcal{P}^+$	$a \to b,$	$a \to b_\alpha, a \to b_\beta, a \to b_\gamma,$
	$b \to c$	$b_\alpha \to c, b_\beta \to c, b_\gamma \to c,$
$\mathcal{X}^+$	$\emptyset$	$b_\alpha \Leftrightarrow b_\beta, b_\alpha \Leftrightarrow b_\gamma, b_\beta \Leftrightarrow b_\gamma$
$\mathcal{M}^+$	$\emptyset$	$\emptyset$

C.2 Resource Alternation and Capacity Change

Interventions regarding the dynamic modification of resource requirements and availabilities can also be modeled within the framework of the *x-RCPSP*. Both the option of switching between alternative resources and the option of modifying resource capacities can be defined based on the previously described concepts of activity mode alternation.

The definition of alternative resources is first of all based on the introduction of an additional resource type. The possibility to execute an activity on this *or* the originally intended resource is described through the introduction of an additional activity mode. This way, full flexibility is provided in the definition of alternatives: Effects on costs, durations and resource requirements can be described per activity, for example. The drawback of the additional workload, which is required for modeling, can be alleviated through the provision of abstract modeling constructs.

The possibility of capacity change is described in a similar way: Instead of changing u_k directly, an additional (alternative) resource type is introduced, representing the available standby units. For each activity, which might trigger a temporary extension of resource capacities, an alternative activity is introduced, defining the relationship between the process step and the reserves. We claim that this approach represents an appropriate method for the description of respective interventions: In realistic scenarios capacity changes are typically motivated by and executed for *specific* activities. Moreover, a high level of flexibility is provided through the possibility to define dependencies per activity. Again, additional modeling workload can be eliminated through the provision of appropriate modeling constructs.

C.3 Activity Insertion/Removal

The dynamic insertion or removal of an activity represents an elementary form of potential structural process modification. This section illustrates how this option can be described by the use of the proposed constructs.

Again, we consider a simple sequence of three activities a, b and c as the original network. The possibility of inserting an additional process step e between b and c (or remove it from there, respectively) shall be described. For this purpose we distinguish two alternative versions of the optional activity's predecessor b: b_+ is bound with the execution (i.e. *activation*) of e whereas $b_\neg$ is bound with the omission (i.e. *deactivation*) of e. As regards precedence constraints, e is executed after its predecessor b_+ and before the start of all successors of the original b. The insertion or removal of the optional activity corresponds to the switch from one alternative predecessor to the other one. The respective possibility is described through the insertion of appropriate elements into $\mathcal{X}^+$. As regards the associated modifications in the set of *active* activities, $\mathcal{M}^+$ is used to define internal dependencies: Activity e is linked to b_+ in mutual inclusion and to $b_\neg$ in mutual exclusion. Correspondingly, Table II compares the original network to a modified version, in which it is possible to insert and remove activity e.

TABLE II
ACTIVITY INSERTION/REMOVAL IN THE *x-RCPSP*.

	Original	Modified Network
$\mathcal{A}^+$	a, b, c	$a, b_+, b_\neg, c, e$
$\mathcal{P}^+$	$a \to b,$	$a \to b_+, a \to b_\neg,$
	$b \to c$	$b_+ \to e, e \to c, b_\neg \to c$
$\mathcal{X}^+$	$\emptyset$	$b_+ \Leftrightarrow b_\neg$
$\mathcal{M}^+$	$\emptyset$	$b_+ \oplus e, b_\neg \ominus e$

C.4 Order Switch

Alternative activities can also form the basis for the description of the possibility to switch the execution order of two arbitrary process steps.

If we consider a sequence of four activities a, b, c and d and if the possibility to exchange b and d shall be described, two versions of process execution can be distinguished: In one version b is considered before, in the other version b is considered after d. Correspondingly, each of the movable process steps is replaced by two alternative activities: One of these alternatives is positioned according to the original reference process and the other one is executed at the position of the respective counterpart. In the considered example, $b_1(d_1)$ inherits all related precedence constraints from $b(d)$ whereas $b_2(d_2)$ is attached to the predecessors and successors of $d(b)$. The possibility to execute the switch operation is defined in $\mathcal{X}^+$ and the dependencies between b and d are described in $\mathcal{M}^+$: It has to be guaranteed that always the same alternatives are *active* for both activities. Table III summarizes the differences between the original and the modified network, which provides the possibility to switch b and d.

TABLE III

Order Switch in the *x-RCPSP*.

	Original	Modified Network
$\mathcal{A}^+$	a, b, c, d	a, b_1, b_2, c, d_1, d_2
$\mathcal{P}^+$	$a \rightarrow b,$	$a \rightarrow b_1, a \rightarrow d_2,$
	$b \rightarrow c,$	$b_1 \rightarrow c, d_2 \rightarrow c,$
	$c \rightarrow d$	$c \rightarrow d_1, c \rightarrow b_2$
$\mathcal{X}^+$	$\emptyset$	$b_1 \Leftrightarrow b_2, d_1 \Leftrightarrow d_2$
$\mathcal{M}^+$	$\emptyset$	$b_1 \oplus d_1, b_1 \ominus d_2,$
		$d_1 \oplus b_1, d_1 \ominus b_2$
		$b_2 \oplus d_2, b_2 \ominus d_1$
		$d_2 \oplus b_2, d_2 \ominus b_1$

C.5 Serialization/Parallelization

Another potential form of structural modification is the serialization of what has been planned for parallel or the parallelization of what has been planned for serial execution. In the following it is described how this can be formulated within the *x-RCPSP*.

We consider a sequence of six activities a to f. The possibility of parallelizing the subsequence b to d with the execution of e shall be described. For this purpose, it is sufficient to introduce an alternative for the first activity of the latter sequence: $e_|$ represents the option of serial execution whereas $e_{\|}$ represents the option of parallel execution. As regards precedence relations, $e_|$ merely replaces the original activity e, whereas the parallelized version is a successor of all predecessors (a) of the first activity of the former sequence (b) and a predecessor of all successors (f) of the original activity (e). Moreover, the last element of the originally preceding subsequence (d) has to be linked directly with the successor/s of the originally succeeding subsequence (f). The possibility to switch between serial and parallel execution is expressed through the insertion of a bidirectional exchange relationship into $\mathcal{X}^+$. No mutual dependencies have to be defined. Table IV compares the original with the accordingly modified network.

TABLE IV

Parallelization/Serialization in the *x-RCPSP*.

	Original	Modified Network
$\mathcal{A}^+$	a, b, c, d, e, f	$a, b, c, d, e_|, e_{\|}, f$
$\mathcal{P}^+$	$a \rightarrow b,$	$a \rightarrow b, a \rightarrow e_{\|},$
	$b \rightarrow c,$	$b \rightarrow c,$
	$c \rightarrow d,$	$c \rightarrow d,$
	$d \rightarrow e,$	$d \rightarrow e_|, d \rightarrow f$
	$e \rightarrow f$	$e_| \rightarrow f, e_{\|} \rightarrow f$
$\mathcal{X}^+$	$\emptyset$	$e_| \Leftrightarrow e_{\|}$
$\mathcal{M}^+$	$\emptyset$	$\emptyset$

III. Solving the Extended Model

This section describes how the extended version of the RCPSP can be solved based on an evolutionary approach. The choice of a metaheuristic optimization procedure has mainly been made for reasons of performance: Respective search methods typically can provide good results for larger problems in shorter time than exact optimization approaches of mathematical programming [11]. This feature corresponds to the realistic requirements of real-time disruption management.

Even though the main focus of this section is on the provision of decision support in the area of process disruption management in appropriate time, the possibilities of using the *x-RCPSP* for classical scheduling tasks such as the generation and optimization of an initial schedule are also described. In the following, first the notion of schedule and activity list is introduced. Then, the generation of an initial solution is discussed before finally the evolutionary approach for its incremental optimization is presented.

A. The Object of Optimization

The aim of disruption management is the identification of a set of interventions, which can be applied to the currently existing schedule in response to a disruption. In the context of the *x-RCPSP* it is thus necessary to optimize the *activation* state along with the activity sequence.

Schedules represent the start and end point of optimization. Basically, a *schedule* corresponds to a vector $(\beta_1, \beta_2, ..., \beta_n)$, grouping the starting times β_i of all *active* activities. If we assume $\mathcal{A}_t$ to be the set of activities carried out at time t, a schedule is considered valid, if all of the following statements are true (cf. [9]):

$$\beta_i \geq 0 \qquad \forall i \in \mathcal{A} \tag{5}$$

$$\beta_i + d_i \leq \beta_j \qquad \forall p_{i,j} \in \mathcal{P} \tag{6}$$

$$\textstyle\sum_{i \in \mathcal{A}_t} q_{i,k} \leq u_k \qquad \forall k \in \mathcal{R},\ \forall t \tag{7}$$

Constraint (5) defines the domain for all starting times: No negative values are allowed. Constraint (6) checks if all precedence constraints are respected: The difference between the starting times of two linked activities must be equal to or greater than the duration of the preceding one. Constraint (7) finally defines that resource requirements must never exceed the available capacities.

Due to the difficulty of operating directly on time values when doing optimization [4], it is a common approach to introduce an intermediary layer of solution representation [9]: Respective forms of representation are composed of easily describable and modifiable elements and can be transformed into a schedule unambiguously. From the various potential candidates (see [9] for an overview) we decided on the use of *activity lists*: λ corresponds to a precedence feasible list sorting all *active* activities in the order they shall be considered in the schedule generation. Respective schemes for the conversion of λ to a final set of time values have been described by Kolisch et al. [9] and Hindi et al. [4], for example: Their approaches are based on the sequential insertion of the list's activities at the earliest possible starting time. Note that such a Schedule Generation Scheme (SGS) always generates the same set of starting times for an activity list whereas one schedule might be associated with various different lists.

B. The Initial Solution

Optimization in the context of disruption management targets at the identification of the activity list which provides the best combination of final schedule and associated interventions. Since the proposed approach is based on incremental improvement, the starting point of optimization is an initial activity list corresponding to a disrupted or unoptimized schedule. As regards the generation of the respective λ_0, basically two scenarios can be distinguished: Either a currently existing schedule has to be considered and optimized (as in disruption management) or no schedule is given and the reference process represents the only starting point for optimization (as in classical scheduling).

For disruption management, the given schedule can be converted into an original activity list easily: Since the existing timetable is assumed to respect all precedence requirements, it is sufficient to simply sort all elements $i \in \mathcal{A}^+$, which have actually been considered for execution and which have not been started yet, according to their starting times. If, alternatively, no existing schedule is given, a valid and feasible sequence has to be generated based on the reference process $\mathcal{A}^0$. Algorithm 1 summarizes the respective procedure (cf. [4]) which can be used for the transformation of any combination of activity set and associated precedence constraints into an activity list λ: For this purpose, all contained activities are added sequentially. In each step, first the set of currently schedulable process steps $\mathcal{A}^*$ is determined: It basically consists of all activities which have not been scheduled so far and which do not have any or only previously considered predecessors. Note that $\mathcal{P}_i$ is used to refer to the set of all preceding activities of process step i. If $\mathcal{A}^*$ is empty before all elements have been added to λ, the network described by the processed sets is over-constrained and no valid activity list can be identified: The method returns without any result in line 3. Otherwise, an arbitrary element of $\mathcal{A}^*$ is selected and added at the end of λ. After all elements of the considered set of activities have been added, the method returns the thereby created activity list.

Algorithm 1 Generate Activity List $(\mathcal{A}, \mathcal{P})$

```
1: repeat
2:     $\mathcal{A}^* \leftarrow \{i \in \mathcal{A} | i \notin \lambda \wedge (\mathcal{P}_i = \emptyset \vee \mathcal{P}_i \subseteq \lambda)\}$
3:     if $\mathcal{A}^* = \emptyset$ then return *false*
4:     else add an arbitrary element of $\mathcal{A}^*$ at the end of $\lambda$
5: until $|\lambda| = |\mathcal{A}|$
6: return $\lambda$
```

C. An Evolutionary Algorithm

In an evolutionary algorithm, optimization is accomplished through the continuous evolution of a population: Each generation comprises the fittest individuals of the previous one and their children, which are generated through recombination and mutation. The main idea behind this concept is that a combination of good solutions might result in or be at least close to even better ones.

This section introduces an evolutionary algorithm for the optimization of the *x-RCPSP*. First, some general remarks are made on initial population, fitness function and selection scheme before afterwards specific versions of the crossover and mutate operators are described, which take the existence of alternative activities into account.

C.1 Initial Population, Fitness and Selection

The initial population consists of the initial solution and a certain amount of fellows: All of them are deduced directly from λ_0 through the application of the mutation operator (as discussed below). This first generation therefore combines the option of not intervening at all with several possibilities to apply exactly one form of intervention.

The assessment of the quality of a population and the comparison of solutions is based on a so-called fitness function. In the context of the *x-RCPSP*, this function evaluates activity lists through their conversion into one single numeric value: Both the set of applied interventions as well as the implicitly defined schedule are considered in light of the predetermined goals of schedule optimization. Whereas most scheduling approaches for the resolution of the RCPSP focus on the minimization of total process execution time (the so-called makespan), disruption management is rather concerned with the implications of earliness and tardiness, costs for interventions as well as the difference from the original plan, for example.

As long as the current population does not fulfil the specified optimization criteria, a new generation is deduced from the existing one. It is composed of the fittest individuals and several children, the parents of which are selected with a probability proportional to their relative fitness.

C.2 A Crossover Operator for the *x-RCPSP*

Handling the potentially distinct sets of activities contained within the activity lists represents the main difficulty in the combination of two parent solutions. A respective procedure, which is based on the idea that one parent λ_a prescribes the interventions to consider (i.e. the *activation* state of the child) whereas the other one λ_b defines relative priorities of the contained process steps (i.e. the list order), is summarized in the following.

First it is checked whether the activity sets associated with both lists are identical: If so, an RCPSP-related crossover operator can be applied to the lists (see [4], for example). Otherwise, an *x-RCPSP*-specific procedure is executed. The problem that one activity list shall prescribe the order of a distinct set of activities is resolved by the use of a so called *transition set* $\mathcal{T} \subseteq \mathcal{X}^+$. This set basically describes how to convert the elements of λ_b into the elements of λ_a. If $\mathcal{X}_a$ is the set of structural modifications which led from λ_0 to λ_a, $\mathcal{T}$ combines all elements which either exist in $\mathcal{X}_a$ or $\mathcal{X}_b$: Note that for a successful conversion it has to be possible to invert all modifications which are exclusive to the latter set. Based on this transition set and the prescribed order of activities, a new activity list is generated in an iterative procedure: All elements of λ_b are either appended themselves, replaced by potential substitutes or ignored (e.g. if an activity is mutually excluded by a future replacement). Any activity exchange has of course to consider all associated dependencies as defined in $\mathcal{M}^+$.

C.3 A Mutation Operator for the *x-RCPSP*

As regards mutation, which is potentially applied to newly generated children in order to avoid early convergence to only local optima, again a specific version of the operator has to be introduced for the *x-RCPSP*. A procedure considering also the possibility to change the structure of the process (i.e. to exchange alternative activities) is briefly described in the following.

First, a random value is generated which determines, if either rescheduling or a structural modification is applied: A fixed value θ defines respective probabilities. In the former case, again an RCPSP-related method can be applied for the mere rearrangement of the elements contained within the activity list. In the latter case, an activity is replaced by an alternative: For this purpose a valid modification $x_{i,j} \in \mathcal{X}^+$ is randomly selected for an arbitrary element of λ: Activity i and all elements excluded by j are removed from the activity list, whereas j and all mutually included activities are inserted at the former position of i. Note that this exchange operation has to be precedence feasible: An added activity can only be inserted after its last predecessor and all associated successors have to be shifted to its right-hand side.

IV. Exemplary Application

This section provides an illustrative example for the application of the previously introduced concepts to a realistic problem of real-time disruption management. After the introduction of the turnaround process – the most typical airport ground process – and three exemplary forms of potential interventions, it is shown how the *x-RCPSP* can be used for its description and how the evolutionary optimization approach can be applied. Finally, some remarks on the respective prototype implementation are made.

A. Overview

The presented approach of disruption management can be applied to various problems in various domains. Wherever it is necessary to provide comprehensive decision support in the operative management of disruptions of time- and resource-dependent processes, the respective concepts can be used as a basis for the proposal of interventions concerning rescheduling and structural process modifications. Project management (see [1], [3]), production planning (see [12], [13]), supply chain management (see [1], [14]), logistics management (see [1]) or traffic flow management represent typical examples of potential fields of application.

Another field, in which disruption management plays a particularly important role, is the domain of air traffic (see [2], [15], [16], [17]). Whereas existing applications mainly focus on aircraft and crew scheduling, we will alternatively illustrate how the proposed concepts can be applied for real-time disruption management in the turnaround process. This process basically combines all activities carried out at an airport while an aircraft is on ground. Instead of considering all actually relevant process steps, a simplified version will be regarded, basically corresponding to the combination of core processes as mentioned by Carr [18]: After the plane reaches its gate or stand position, first the incoming passengers leave the aircraft. It is then fueled, cleaned and catered simultaneously before the outgoing passengers enter the plane. Finally, it leaves its position heading for the runway.

In the following, we will assume an instance of this process, in which a disruption occurs during taxi-in, prior to deboarding. This way, a departure delay is caused by the delay of the first activity and the implied shift of all succeeding process steps. For the resolution of this problem we assume the existence of three basic forms of potential structural modifications: First, an acceleration of deboarding can be reached through the assignment of additional busses. Second, it is possible to shorten cleaning, if in exchange the cabin is additionally inspected by the cabin crew prior to boarding. Third, fueling and boarding can be parallelized if the fire brigade is present for supervision. As regards potential options of rescheduling, respective possibilities are defined by the process structure itself.

B. Modeling the Turnaround Process

Along with the reference process we define the possibilities of structural modifications based on the patterns introduced in Section II-C. A potential acceleration of boarding through the assignment of additional resources corresponds to the simple option of mode alternation. Shortening cleaning and inserting an additional step of inspection corresponds to a mixture of mode alternation and activity insertion. Finally, the possibility to execute boarding and fueling in parallel corresponds to a specific version of activity parallelization. The model resulting from the application of the respective patterns is summarized in Table V. Note that in realistic applications graphical interfaces can be used for the intuitive creation of such models.

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

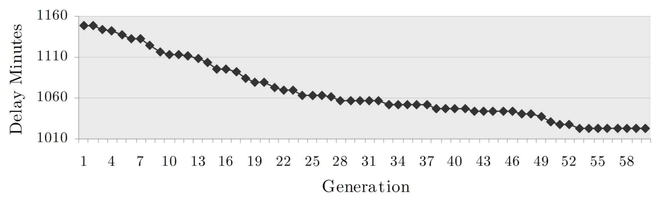

Fig. 1. Reduction of Delay Minutes in Turnaround Optimization

C. Airport Ground Process Disruption Management

Given the disruption of the regarded example, the aim of turnaround process disruption management is the minimization of the associated negative impact: An exemplary goal might be the elimination of all pending delays. As regards the *initial solution*, we assume that $\langle Deb, Fue, Cle, Cat, Boa\rangle$ can be extracted as λ_0. For this activity list, various fellows can be generated through *mutation*, forming the first generation: Examples of respective instances are $\langle Deb, Cle, Fue, Cat, Boa\rangle$ and $\langle Deb, Fue, Cle^{Red}, Ins, Cat, Boa\rangle$. If none of the contained solutions fulfils a predetermined stopping criterion, new generations are deduced from the fittest individuals: Applying the *crossover* operator on two parents $\lambda_a = \langle Deb^{Bus}, Fue, Cle, Cat, Boa\rangle$ and $\lambda_b = \langle Deb^{Bus}, Cat, Cle^{Red}, Ins, Boa, Fue^{Par}\rangle$ generates the child activity list $\langle Deb^{Bus}, Cat, Cle, Fue, Boa\rangle$ by the use of the transition set $\mathcal{T} = \{Cle^{Red} \Rightarrow Cle, Fue^{Par} \Rightarrow Fue\}$. As soon as the goal of optimization is reached or a certain amount of time has passed, the genetic algorithm stops and provides a set of the best solutions found so far. The respective schedules implicitly describe associated interventions.

D. Prototype Implementation

The presented approach has been implemented in a prototype based on Java. In an exemplary setting, we considered $p = 20$ instances of the discussed version of the turnaround process: We defined the durations of the elements in $\mathcal{A}^+$ to be $(0, 15, 7, 20, 20, 9, 14, 8, 1, 15, 0)$ and assumed the deadline of all processes to be 0. Given the availability of 10 busses and 3 fire brigades, the unmodified reference processes therefore cause an overall delay of 1150 minutes. Based on the $n = 3$ provided modification possibilities, this value can be reduced to 984 minutes at most: As regards structural interventions only, in the worst case $2^{n*p} = 2^{60}$ schedules have to be evaluated for the identification of this theoretical optimum. In our heuristic approach, we considered 60 generations with 10 members each: By evaluating only 600 solutions, about 75% of the full optimization potential could be tapped within 4 seconds on a standard PC with 1800 MHz and 512 MB RAM. Figure 1 illustrates an exemplary reduction of the delay minutes associated with the best known solution throughout the generations.

These first results are promising, especially since the current implementation is still unoptimized and no domain-specific knowledge is considered in the search procedure. Intended improvements concern further reductions of the search space and the optimization of the selection process.

V. Conclusions and Future Work

This paper described how both rescheduling and structural process modifications can be considered in a comprehensive approach to disruption management. In various modification patterns, the possibilities to describe potential forms of interventions based on the concept of alternative activities have been illustrated. An extended version of the well-known RCPSP has been introduced and a metaheuristic optimization approach based on an evolutionary algorithm has been presented. Finally, the application of the introduced concepts to a realistic problem has been discussed based on the airport turnaround process.

As mentioned before, future work will be directed on the development of techniques for reducing the search space (based on a matchup point strategy, for example), the provision of possibilities to introduce domain-specific knowledge and the respective optimization of the prototype. A final aim is the generation of comparable benchmark results for realistic problems in realistic sizes.

References

[1] G. Yu, X. Qi, *Disruption Management: Framework, Models and Applications*, World Scientific Publishing, Singapore, 2004.
[2] J. Clausen, J. Hansen, J. Larsen, A. Larsen, *Disruption Management*, ORMS Today, 28: 40-43, 2001.
[3] G. Zhu, J.F. Bard, G. Yu, *Disruption management for resource-constrained project scheduling*, Journal of the Operational Research Society, 56: 365-381, 2005.
[4] K.S. Hindi, H. Yang, K. Fleszar, *An Evolutionary Algorithm for Resource-Constrained Project Scheduling*, IEEE Transactions on Evolutionary Computation, 6: 512-518, 2002.
[5] S. Hartmann, *Project Scheduling with Multiple Modes: A Genetic Algorithm*, Annals of Operations Research, 102: 111-135, 2001.
[6] C. Artigues, P. Michelon, S. Reusser, *Insertion techniques for static and dynamic resource constrained project scheduling*, European Journal of Operational Research, 149: 249-267, 2003.
[7] A. Elkhyari, C. Guéret, N. Jussien, *Constraint Programming for Dynamic Scheduling Problems*, ISS'04, Japan, 84-89, 2004.
[8] J.C. Beck, M.S. Fox, *Constraint Directed Techniques for Scheduling with Alternative Activities*, Artificial Intelligence, 121: 211-250, 2000.
[9] R. Kolisch, S. Hartmann, *Heuristic Algorithms for solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis*, Project scheduling: Recent models, algorithms and applications, 147-178, Ed: J. Weglarz, Kluwer, Amsterdam, Netherlands, 1999.
[10] R.K-Y. Li, R.J. Willis, *Alternative resources in project scheduling*, Computers and Operations Research, 18: 663-669, 1991.
[11] J. Bautista, A. Lusa, R. Suárez, M. Mateo, R. Pastor, A. Corominas, *Application of Genetic Algorithms to Assembly Sequence Planning with Limited Resources*, IEEE Int. Symposium on Assembly and Task Planning, Porto, Portugal, 411-416, 1999.
[12] J. Yang, X. Qi, G. Yu, *Disruption management in production planning*, Naval Research Logistics, 52: 420-442, 2005.
[13] Y. Xia, M.H. Yang, B. Golany, S. Gilbert, G. Yu, *Real-time disruption management in a two-stage production and inventory system*, IIE Transactions, 36: 111-125, 2004.
[14] M. Xu, X. Qi, G. Yu, H. Zhang, C. Gao, *The demand disruption management problem for a supply chain system with nonlinear demand functions*, JSSSE, 12: 82-97, 2003.
[15] B. Thengvall, J.F. Bard, G. Yu, *Balancing user preferences for aircraft schedule recovery during irregular operations*, IIE Transactions on Operations Engineering, 32: 181-193, 2000.
[16] N. Kohl, A. Larsen, J. Larsen, A. Ross, S. Tiourine, *Airline Disruption Management - Perspectives, Experiences and Outlook*, Technical Report, IMM, Technical University of Denmark, 2004.
[17] J. Clausen, A. Larsen, J. Larsen, *Disruption Management in the Airline Industry - Concepts, Models and Methods*, Technical Report 2005-01, IMM, Technical University of Denmark, 2005.
[18] F. R. Carr, *Robust Decision Support Tools for Airport Surface Traffic*, PhD Thesis, Massachusetts Institute of Technology, 2004.