Synthesis of Schedulers for Planned Shutdowns of Power Plants

Carla P. Gomes
Rome Laboratory*
525 Brooks Rd.
Rome Lab, NY 13441-4505
gomes@ai.rl.af.mil

Douglas Smith and Stephen Westfold
Kestrel Institute
3260 Hillview Avenue
Palo Alto, CA 94304
{smith,westfold}@kestrel.edu

Abstract

We describe the synthesis of efficient schedulers for planned shutdowns of power plants for refueling and maintenance (outages), using an automated programming tool, KIDS. Currently, the utility industry has no automated tools to generate schedules that are both safe and resource-efficient. We focused on safety constraints since they are critical in this application. There are several aspects of this project that go beyond previous applications of KIDS to scheduling problems. First, scheduling of outages of power plants has a planning-like character since the scheduler needs to represent and maintain the complex state of the plant at all times considered during the scheduling process. Second, the particular safety constraints we considered required scheduling a pool of resources in the presence of time windows on each activity. To our knowledge the control and data structures that we developed for handling such a pool are novel. In terms of design knowledge, the outage scheduling problem is modeled as a constraint satisfaction problem and the synthesized algorithm is an instance of global search with constraint propagation. The derivation of specialized representations for the constraints to perform efficient propagation is a key aspect of our approach. In addition, finite differencing complements constraint propagation by efficiently maintaining the state of the world.

1. Introduction

Planning and scheduling tasks are inherently complex. In computational terms, they are *intractable*, *i.e.*, NP-hard or worse. As a practical consequence, realistic size planning and scheduling problems cannot be solved optimally in a "reasonable" amount of time. Nonetheless, solutions have to be found for real-world problems, and therefore heuristic approaches have to be adopted, ideally with some guarantee on the quality of the solution.

Our approach emphasizes the fast generation of schedules to cope with large domains. We use a rich representation for the state of the world at any time (as in planning approaches) which allows efficient constraint reasoning, temporal reasoning in particular (as in scheduling). The problem is modeled as a *constraint satisfaction problem* combining a *global search tactic* with *constraint propagation*. The derivation of very specialized representations for the constraints to perform efficient propagation is a key aspect of our approach. Furthermore, *finite differencing* complements constraint propagation by efficiently maintaining the state of the world, allowing incremental computations. In our approach, constraints are *compiled* into the code - this is a novel aspect of our work using an automatic programming system, KIDS (Kestrel Interactive Development System)[13]. Another novel aspect of our approach is the generation of schedules that are feasible over time windows rather than having single time points as start times, which increases schedule robustness.

We describe the application of our approach to the real-world problem of multiple resource-constrained project management. This problem is very common in manufacturing and it is a generalization of the well-known job-shop scheduling problem [3, 17]. As a particular instance of this problem, we consider the management of outages of power plants.[1] An outage is a planned shutdown for refueling, repair, and maintenance. It is a rather daunting real-world task that may

*Carla P. Gomes works for Rome Laboratory as a Research Associate.

[1] Rome Laboratory has a project in collaboration with Electric Power Research Institute (EPRI), Kaman Science, and Kestrel Institute to evaluate the use of advanced AI/OR planning and scheduling technology for outage management.

involve from 5,000 up to 45,000 activities. Furthermore, in this domain, the existence of good automatic solutions is not only crucial for safety reasons but also for economic reasons — the cost per day of shutdown is in the order of $1,000,000.

There are several aspects of this project that go beyond previous applications of KIDS to scheduling problems. First, scheduling of outages of power plants has a planning-like character since the scheduler needs to represent and maintain the complex state of the plant at all times considered during the scheduling process. Second, the particular safety constraints we considered required scheduling a pool of resources in the presence of time windows on each activity. To our knowledge the control and data structures that we developed for handling such a pool are novel.

In the next section we discuss related work. In section 3 we describe the outage problem. Section 4 describes our approach to schedule synthesis. In section 5 we present performance results. We draw conclusions and discuss future work in section 6.

2. Related Work

Traditionally in AI, planning and scheduling are considered as two distinct phases of project management. Planning determines how to achieve a set of goals by performing a set of actions in a given (partial) order, and given an initial state of the world, without taking into consideration resource constraints. Scheduling, on the other hand, considers a more operational perspective: actions have times and resources assigned to them.

Current AI planners, e.g., SIPE-2 [19, 20] and O-PLAN [16], have very limited temporal and resource reasoning capabilities. In order to overcome this limitation and to improve the quality of the generated plans, for instance, SIPE-2 was integrated with TACHYON [1], a temporal reasoner, and with the capacity analysis module of OPIS (DITOPS), a scheduling system developed at CMU([14]). These experiments are reported in [2]. OPLAN's conceptual model includes *constraint managers* sharing a common constraint representation. For example, resource utilization is handled by one manager, temporal reasoning by another, and so on. Tate ([15]) suggests replacing O-PLAN's simple time constraint manager with a better temporal system — an approach similar to the integration of SIPE-2 and TACHYON. Tate also suggests adding other types of constraint managers, such as spatial reasoning, through a modular interface with the planner. Although this type of integration adds capabilities to the planners (e.g., temporal, resource or spatial reasoning), its modular nature (*plan critics* in SIPE-2 or *constraint managers* in OPLAN) is not computationally efficient.

The separation of planning and scheduling does not reflect operational reality — it is an artifact of current approaches of automated planners and schedulers. A plan should not be feasible without having a feasible schedule instantiation because of resource constraints. Therefore, the integration of planning and scheduling seems natural. Moreover, temporal and resource constraints from the scheduling problem can be used in the planning phase to prune large parts of the search space.

AI schedulers, on the other hand, incorporate temporal and resource reasoning techniques (e.g., [14], [10] [7]). However, scheduling approaches typically do not model the state of the world (as in planning).

Some researchers have addressed the integration of planning and scheduling. For instance, HSTS ([8]) is a framework unifying planning and scheduling with a rich domain representation allowing constraint propagation. The main limitation of such approaches is the lack of guarantee that fast good quality solutions are produced. The collection *Intelligent Scheduling*, includes several AI planners and schedulers as well as systems that integrate planning and scheduling [21].

The main innovation of our approach compared to other planning and scheduling approaches is the derivation of very specialized constraints that are compiled into the search and control mechanisms [5, 6]. Existing planning and scheduling approaches use constraint representations and operations that are geared for a broad class of problems. Our approach, derives specialized representations for constraints allowing fast constraint checking and constraint propagation.

Another novel aspect of our approach is the generation of schedules that have time windows as start times, which adds robustness to the solution. Existing AI approaches to scheduling with resource constraints only generate a single solution, feasible for single start times, without any guarantees of feasibility over time windows. With our approach, we generate an infinite family of feasible schedules.

The framework selected for this project was KIDS (Kestrel Interactive Development System)[13], which supports users in transforming declarative problem specifications into correct and efficient programs. The transformations provided in KIDS are designed to perform significant and meaningful actions in terms of search efficiency. The various transformations in KIDS include: algorithmic transformations, program optimization techniques and data structures refinement. The algorithmic transformations allow the user to add search and control mechanisms to a given problem specification. Finite differencing is another important transformation provided by KIDS. KIDS uses a form of deductive inference called *directed inference* to reason about the problem specification in order to automatically apply tactics, derive filters and perform constraint propagation. KIDS has been used to derive fast and accurate transportation schedulers from formal specifications on large-scale transportation planning problems. A typical transportation problem with $10,000$ movement requirements takes the derived scheduler 1 to 3 minutes to solve, compared with 2.5 hours for a deployed feasibility estimator (JFAST) and 36 hours for deployed schedulers (FLOGEN, ADANS). The computed schedules use relatively few resources and satisfy all specified constraints [11].

In this paper we show how previous applications of KIDS to scheduling problems can be extended to tackle a much richer real-world scheduling task with planning-like features involving complex safety constraints and resource constraints in the presence of time windows on each activity, a novel aspect of the approach reported here.

3. Management of Outages of Power Plants

The planning and scheduling of outages of power plants have a great impact in terms of the outage costs (replacement power, labor cost, etc.), use of scarce resources and implementation of safety procedures. During an outage several activities are performed, such as refueling operations, plant betterment, preventive maintenance, corrective maintenance, and technical specification requirements for inspections or surveillance [9, 18]. Depending on the particular plant, as well as the scope of the activities performed during the outage, the planning and scheduling of outages for power plants might involve from $5,000$ up to $45,000$ activities. Explicit precedence constraints between activities are defined in *work order activities*. Other constraints between activities arise as a result of different technological constraints. The general principle underlying the outage procedures is that outages should be as short as possible, maintaining the appropriate level of safety. The main safety functions that are monitored during an outage are: AC power control system, primary and secondary containment, fuel pool cooling system, inventory control, reactivity control, shutdown cooling, and vital support systems. In this paper, we describe the generation of schedules for outages of power plants enforcing safety conditions regarding AC power control.

The current scheduling software tools used by the utilities are very simple - planning and scheduling still heavily rely on the experience of the manual schedulers, rather than on automatic procedures. Current automatic approaches to outage scheduling mainly consist of the application of automatic project management techniques, such as PERT and CPM techniques, which only handle simple before/after precedence relationships between activities without taking into consideration resource constraints and safety constraints. Safety and risk assessment is usually a manual process which calls on the expertise of the personnel involved to make decisions based on published policies and procedures. In order to ensure that the sequence of activities performed during an outage follows the safety requirements, the utilities perform the risk assessment of the schedules using the software ORAM (Outage Risk Assessment Methodology). ORAM, an EPRI tool for risk assessment, simulates the execution of the schedule keeping track of the configuration of the plant at any time. If the schedules do not satisfy the safety constraints, manual adjustments have to be performed in order to meet the safety requirements.

4. Schedule Synthesis

Our approach to planning and scheduling combines a constraint satisfaction paradigm with a global search tactic with constraint propagation. We developed a prototype for the domain of outages of power plants, ROMAN (Rome Lab Outage MANager). ROMAN includes all the technological constraints currently incorporated in the automatic tools used by the utilities for schedule generation. In addition, it includes all the constraints regarding the safety function AC power. Other safety functions could be modeled in a similar way. A top level formal specification of the outage problem including the safety function AC power follows:[2]

$$
\begin{array}{l}
\mathbf{function} : \textit{safe-outage-windows}(\textit{activities}) \\
\textit{returns}(\textit{schedule} \mid \\
\quad \textit{Consistent-Activity-Separation}(\textit{schedule}) \wedge \\
\quad \textit{Consistent-AC-power}(\textit{schedule}) \wedge \\
\quad \textit{All-activities-scheduled}(\textit{activities}, \textit{schedule}))
\end{array}
$$

In this formulation *activities* correspond to the set of activities to be performed. Each activity has a given duration, a set of predecessors, and a set of effects on resources. The *schedule* is a partial order of activities. Activities in the schedule have

[2] We modeled the AC power safety function as a proof of concept. Other safety functions could be modeled in a similar way.

time windows assigned to it. A time window defines the earliest start time (*est*) and latest start time (*lst*) of an activity, such that the activity can start at any time during the window without increasing the overall duration of the project. Given the duration of the activity, the earliest finish time (*eft*) and latest finish time (*lft*) can be calculated. The predicate *Consistent-Activity-Separation*(*schedule*) states that all the activities in the schedule satisfy the precedence constraints. The predicate *Consistent-ac-power*(*schedule*) states that the schedule verifies the safety constraints, from an AC power point of view. As a completeness condition, the predicate *All-activities-scheduled*(*activities*, *schedule*) states that all the activities have to be scheduled.

The notion of *state of the plant* is a key concept in enforcing safety constraints. In outage management, the state of the plant is measured in colors — green, yellow, orange or red, in this order of increasing risk. Figure 1 illustrates the types of decision trees regarding safety levels for the case of AC power. Basically, the AC power safety constraint states the conditions in terms of availability of AC power resources and types of activities being executed, for which the state of the plant is safe (green or yellow). For instance, if there is an activity being executed that has the potential to cause AC power loss, then in order for the plant to be in a yellow state it is required to have two off-site AC power sources available and three operable emergency safeguard buses.

Since the start times of activities are defined over time windows, we introduce two concepts regarding the execution of an activity: the *definite period* and the *potential period* of an activity. The definite period of an activity corresponds to the period of time during which the activity is definitely being execute — it is the interval of time between the latest start time of the activity (*lst*) and its earliest finish time (*eft*). The potential period of an activity corresponds to the period of time during which the activity may be executed — it is the time period between the earliest start time of the activity (*est*) and its latest finish time (*lft*). Figure 2 illustrates the notion of *definite* period of an activity. Notice that activity A does not have a definite period, since its earliest finish time is before its latest start time.

In addition, we define two other concepts: *definite state of the plant* and *potential state of the plant*. The definite state of the plant is associated with the concept of definite period: it represents the state of the plant for a given safety function (*e.g.*, AC power) assuming that activities are only executed during their definite period. The concept of potential state of the plant is associated with the concept of potential period of an activity: it represents the state of the plant for a given safety function assuming that activities are executed during the whole extension of their potential periods. The potential state of the plant is always "equal" or "greater" than the state of the plant since the definite period of an activity tends to underestimate the duration of activities while the potential period of an activity tends to overestimate the duration of activities. Figure 3 gives an example. Note that during certain time intervals, the definite and potential states of the plant coincide.

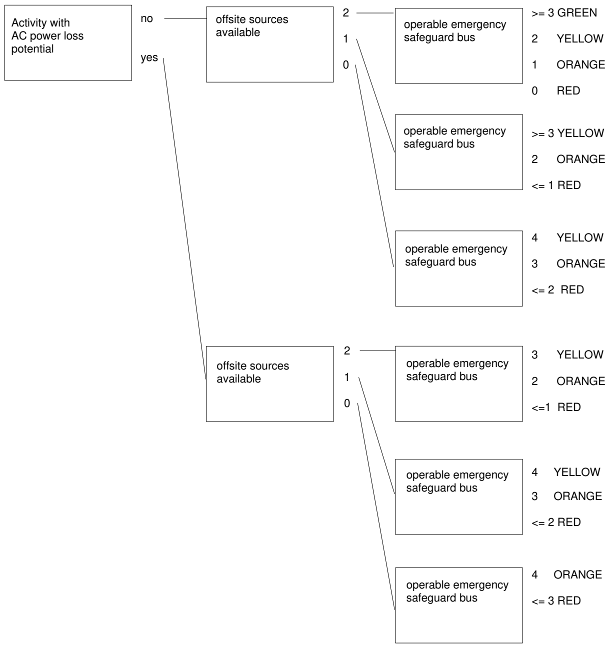

Figure 1. Example of a decision tree for the safety function AC Power

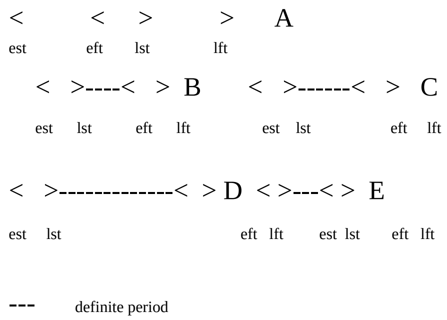

Figure 2. Notion of a definite period.

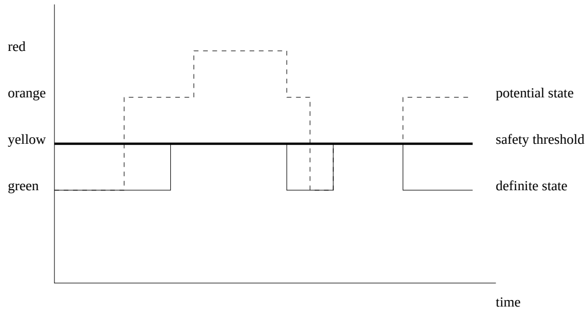

Figure 3. Definite and potential states of the plant.

4.1. Search and Control Mechanisms

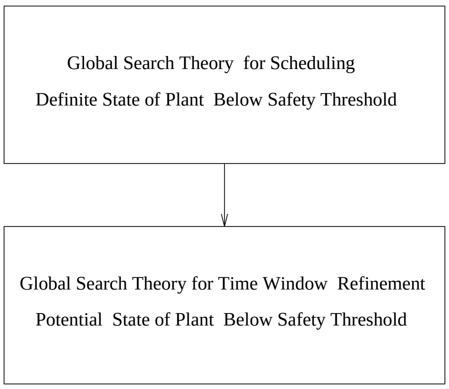

Figure 4. ROMAN's approach

KIDS provides algorithmic transformations that add control and search mechanisms to a given specification. The search tactic selected for the outage problem was *global search* (see next section). Figure 4 summarizes the approach adopted in ROMAN.

Initially global search is applied to the formal specification of the outage problem in order to generate a schedule, assuming the definite period of activities. Since the notion of definite period tends to underestimate the duration of the activities, it is very likely for the schedule produced in this initial phase not to be feasible from the point of view of the potential state of the plant. In order to enforce the safety threshold for the potential state of the plant at any time during the outage, "refinement" of the time windows of the initial schedule takes place. In the next section, we describe global search theory.

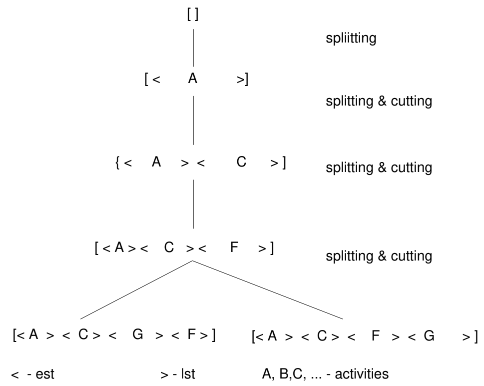

Figure 5. Global search theory for the Outage Problem

4.2. Global Search Theory

Global search [12, 11] is a backtrack algorithm, a refinement of generate-and-test. The tactic is implemented by finding a space containing all the solutions to the problem that can be divided into nested subspaces. The global search algorithm starts with an initial set that contains all the solutions to the given problem instance, repeatedly extracts solutions, splits sets, and eliminates subsets using propagation, until no sets remain to be split. The process can be described as a tree search in which a node represents a set of candidates, and an arc represents the split relationship between a set and a subset. The principal operations are to extract candidate solutions from a set and to split a set into subsets. The derivation of efficient cutting constraints that eliminate subspaces that do not contain any feasible solution is an important complementary operation in the derivation of the global search tactic.

Figure 5 illustrates the global search theory for the initial scheduling of the activities considering their definite periods. In this global search theory the initial subspace descriptor (partial schedule) is the empty sequence (empty schedule). *Splitting* corresponds to appending an unscheduled activity, with a given time window, to the partial schedule. *Cutting* corresponds to propagating the constraints over the time windows of the activities in the partial schedule. Notice that *cutting* makes the time windows shrink. It can also split a time window as in the case of activity G - due to propagation, activity G's window was split into two. As we can see from figure 5 most of the work in this global search theory is performed by constraint propagation. *Splitting* corresponds to just selecting the next activity to schedule, using a heuristic that favors shorter schedules.[3] *Extraction* takes place when all the activities have been scheduled.

The operator *extract* corresponds to the second global search algorithm. Refinement of time windows takes place if after applying the initial global search to the outage problem the potential state of plant does not satisfy the safety requirements. In other words, refinement of time windows is required to enforce the safety constraints over the potential period of all the activities in the initial schedule. This is achieved by applying a new global search to the formal specification of the outage but using as input the schedule generated in the initial phase. In this second phase the windows of the activities that contribute to the contention periods, *i.e.*, the periods in which the potential state of the plant is above the safety threshold, are systematically reduced until the potential state of the plant becomes consistent from the safety point of view for all the times during the outage. In this global search theory for the refinement phase splitting corresponds to reducing the size of the windows of the activities involved in the contention periods.

4.3. Constraint Propagation

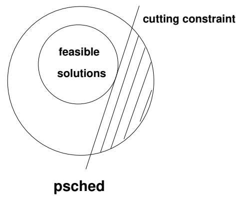

Figure 6. Cutting Constraints

As pointed out in the previous section, one of the important features of our approach is the propagation of constraints. Figure 6 illustrates the concept of constraint propagation. *Psched* represents a partial schedule, a set of candidate solutions, a node of the global search tree. The following test states that a partial schedule can be extended to a complete feasible schedule:[4]

[3] We also define a topological sort of the unscheduled activities according to their levels. An activity has level 0 if it has no predecessors. Activities that only have as predecessors activities of level 0 have level 1. Activities of level 2 only have as predecessors activities that have level 0 or 1, etc.

[4] In the particular case of the outage problem, $(sched \in psched) \iff (domain(psched) \subseteq domain(sched) \wedge \forall(i) i \in domain(psched)) \implies psched(i).est \leq sched(i).st \leq psched(i).lst)$ and $feasible(sched, activities) \iff (consistent\text{-}separation(sched) \wedge consistent\text{-}acp(sched)) \wedge all\text{-}activities\text{-}scheduled(activities, sched))$

$$\exists(sched)\ (sched \in psched \wedge feasible(sched, activities))$$

However, this test is in general too expensive, computationally. Instead, we derive necessary conditions for it, *filters*, i.e.:

$$\exists(sched)\ (sched \in psched \wedge feasible(sched, activities)) \implies \Psi(sched, psched))$$

The next step consists in incorporating $\Psi(sched, psched))$ into *psched*, i.e.:

$$\xi(psched) \iff \forall(sched)\ (sched \in psched \implies \Psi(sched, psched))$$

The test $\xi(psched)$ holds when all the candidate solutions in *psched* satisfy Ψ. The main issue is, when a given *psched* does not satisfy ξ, how can we incorporate ξ into *psched*? The answer is to find the greatest refinement of *psched*, $\widehat{psched}$, that satisfies ξ.

$$\widehat{psched} = max_{\sqsupseteq}\{qsched \mid psched \sqsupseteq qsched \wedge \xi(x, qsched)\}$$

which asserts that $\widehat{psched}$ is maximal over the set of descriptors that refine *psched* and satisfy ξ, with respect to ordering $\sqsupseteq$. We want $\widehat{psched}$ to be a refinement of *psched* so that all of the information in $\widehat{psched}$ is preserved and we want $\widehat{psched}$ to be maximal so that no other information than *psched* and ξ is incorporated into $\widehat{psched}$. The refinement relation $psched_j \sqsupseteq psched_i$ holds when the completions of $psched_i$ are a subset of the completions of $psched_j$.

KIDS instantiates a program scheme for global search with constraint propagation, incorporating ξ. For more detail on propagation in KIDS see [11].

The challenge in order to take advantage of the propagation mechanisms provided in KIDS lies in finding ξ — even though KIDS provides a tactic to synthesize propagation code incorporating ξ, the derivation of ξ is not a straightforward task and has to be done manually.

In the case of the outage problem, the predicate *Consistent-Activity-Separation*(*schedule*) states that all the activities in the schedule satisfy the precedence constraints. The derivation of cutting constraints from the constraint *Consistent-Activity-Separation*, using the formulas for calculating $\Psi(sched, psched)$ and the test $\xi(psched)$ presented above, leads to the well known constraints on *est* and *lst*, as used in PERT. The derivation of cutting constraints for *Consistent-ACP* is less straightforward. An example of an inferred constraint from *Consistent-ACP* follows:

$$
\begin{array}{l}
\forall(i, t1, t2, act) \\
i \in domain(se(psched)) \\
\;\wedge\, t1 = se(psched)(i).time \\
\;\wedge\, t2 = se(psched)(i+1).time \\
act \in domain(psched) \\
\;\wedge\, sacpl?(t1, psched) \\
\;\wedge\, unav\text{-}sources(t1, psched) = TSACPL \\
\;\wedge\, affects\text{-}avail\text{-}acps?(act, psched) \\
\Longrightarrow \\
\;psched(act).lft < t1 \\
\;\vee\, psched(act).est \geq t2
\end{array}
$$

The state events of the partial schedule considering the definite periods of activities are computed by $se(psched)$. A state event corresponds to any event that affects the state of the plant. The time of the ith state event of the partial schedule is represented by se(psched)(i).time, the predicate $sacpl?(t, sched)$ tests if at time t the plant is in a state of AC power loss, $unav\text{-}sources(t, psched) = TSACPL$ tests if at time t the number of unavailable AC power resources equals the threshold for AC power resource unavailability for a state of AC power loss, $affects\text{-}avail\text{-}acpsi(act, psched)$ tests if the activity *act* affects an available AC power resource, and $psched(act).lft$ and $psched(act).est$ correspond respectively to the latest finish time and earliest start time of the activity *act* of the partial schedule *psched*. This constraint triggers propagation for the activities that affect available AC power resources — propagation eliminates from the activities' time windows the periods that overlap the intervals that correspond to a state of AC power loss with number of unavailable AC power resources equal to $TSACPL$ (the threshold). In other words, a new activity that affects available AC power resources cannot occur during a period for which the plant is operating at the threshold regarding the AC power safety function.

In this paper there is no room for elaborating on the termination conditions for propagation. Nevertheless, briefly we point out that such guarantee follows from the application of Tarki's fixpoint theorem. The assumptions required for the theorem are verified in the global search theory for the outage problem. For more detailed information on this issue see e.g., [4, 11]. For a formal description of the (manual) derivation of the constraints incorporating the test ξ for *Consistent-Activity-Separation* and *Consistent-ACP* see [5]. The manual derivation of these constraints represented more than 60% of time of the design of the outage domain theory.

4.4. Interaction Between The Schedule and the State of Plant

A main principle embodied in our approach is incremental computation - propagation illustrates that concept - whenever a new activity is scheduled, all constraints are immediately propagated over the schedule. Finite differencing is another transformation that allows for incremental computation, by efficiently maintaining the state of plant. Roughly, the idea behind finite differencing is to incrementally evaluate an expensive expression in a loop, rather than recomputing it from scratch each time. As an example, let us assume that function $f(x)$ calls function $g(x)$ and that x changes in a regular way. In this case, it might be worthwhile to create a new variable, whose value is maintained and which allows for incremental computation. By abstracting function f with respect to expression $g(x)$ a new parameter c is added to f's parameter list (now $f(x, c)$) and $c = g(x)$ is added as a new input invariant to f. Any call to f, whether a recursive call within f or an external call, must now be changed to supply the appropriate new argument that satisfies the invariant — $f(x)$ is changed to $f(x, g(x))$. In this process all occurrences of $g(x)$ are replaced by c. Often, distributive laws[5] apply to $g(h(x))$ yielding an expression of the form $h'(g(x))$ and so $h'(c)$. The real benefit in the optimization comes from the last step, because this is where the new value of the expression $g(h(x))$ is computed in terms of the old value of $g(x)$.

In the outage problem there are several opportunities for finite differencing since the state of the plant is a function of the schedule represented by the constraint *consistent-acp*$(schedule)$. Figure 7 shows the interactions between the *state of plant* and the schedule - when a new activity is scheduled, it impacts the

[5] Laws are assertions that define axioms or theorems, i.e., statements that are always true. An assertion is simply a true statement - an example of a law is $(A+B)*C = (A*C)+(B*C)$, or (A and $B \rightarrow A$). The idea is to provide information on how to distribute predicates and functions over the main constructors of the variable that changes in a regular way, exactly in the same way one would write a law about how to distribute multiplication over addition. Additionally, laws also specify special cases, for instance when dealing with base cases (e.g., empty sequences).

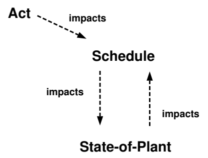

Figure 7. Interaction between the schedule and the state

schedule and propagation is triggered. Changes in the schedule impact the state of the plant, which is incrementally maintained by finite differencing. Changes in the state impact the schedule and propagation is triggered, which impacts the schedule and so on. The key issue to take advantage of finite differencing is to provide good laws on how to distribute the functions to be finite differenced over the main constructors of the partial schedule, e.g., over appending an activity to the schedule, increasing the *est* of an activity, etc.

5. Performance Results

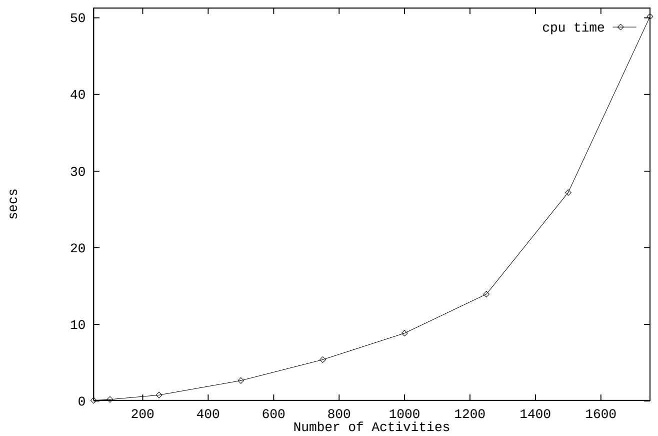

Figure 8. Time performance

The current version of ROMAN was completed in November 1995, and it has been demonstrated to several large power plants such as American Electric Power Service, Baltimore Gas & Electric, PECO Energy, *etc.* The demonstration was successful, and EPRI, a consortium of more than 90% of the utilities in the US, is looking into using the approach embodied in ROMAN to build the next generation of outage scheduling tools — referred to as Advanced Technology Outage Scheduler.

ROMAN has proven successful since it clearly extends the current functionality offered by existing software tools for outage management. All the technological constraints currently used for automatic schedule generation are incorporated into the system. In addition, ROMAN produces schedules enforcing safety constraints — AC power was used as a proof of concept.

The current version of ROMAN schedules up to 2,000 activities in approximately 1 minute on a Sparc 2 (see figure 8). The schedules produced by ROMAN are often better than the current solutions since many new possibilities are explored compared to manual solutions. Human schedulers tend to aggregate tasks and schedule them as blocks rather than exploring interesting possibilities that occur when the activities are scheduled separately.

A key feature of ROMAN that utility personnel find attractive is the robust schedules that are generated. The current scheduler generates a schedule that includes start time windows for each task. Choosing *any* start time within the window for a task still permits feasible execution of the schedule. The window provides information about how critical the start time for a task is – if a predecessor task is delayed, a user can decide whether there still enough freedom in the start time window to allow on-time completion, or whether it is time to reschedule parts of the overall operation.

ROMAN currently comes configured with a GUI that displays an interactive Gantt chart for tasks, showing their start time window, duration, task description, and predecessors. Another Gantt chart shows the history of the state of the plant with respect to AC power.

6. Conclusions and Future Work

ROMAN has successfully demonstrated that outage schedules that satisfy safety constraints can be quickly generated. To develop ROMAN into a practical tool requires (1) handling a richer model of the outage domain to incorporate other safety constraints, and (2) faster code. Future work includes developing better techniques for scaling up the scheduler through better data structures and search strategies that are better suited to the problem domain. To date we have focused on one particular safety function dealing with maintaining adequate sources of AC power. Future work for the domain of outage management is planned to deal with other critical safety constraints and scheduling scarce resources such as heavy lifts and skilled personnel.

Acknowledgments

This work has been supported by Rome Laboratory under contracts F30602-93-D-0075 and F30602-95-C-0063. The authors would like to thank several people who contributed to the success of ROMAN. Lou Hoebel for creating the conditions and putting together the resources for this project. Karen Alguire for her contribution to this project, in particular helping with the programming. Jeff Mitman from ERIN/EPRI was instrumental in providing information about the the domain of outage management. Dick Wood and Shyam Kamadolli provided data and information for our experiments.

References

[1] R. Arthur and J. Stillman. Tachyon: A model and environment for temporal reasoning. Technical report, GE Corporate Research and Development Center, 1992.

[2] M. A. Bienkowski and M. E. desJardins. Planning-Based Integrated Decision Support Systems. In *Proceedings of the Conference of AI Planning Systems*, Chicago, 1994.

[3] J. Blazewicz, J. Lenstra, and A. R. Kan. Scheduling Projects to Resource Constraints: Classification and Complexity. *Discrete Appl. Math.*, 5:11–24, 1983.

[4] J. Cai and R. Paige. Program Derivation by Fixed Point Computation. *Science of Computer Programming*, 11:197–261, 1989.

[5] C. P. Gomes. Automatic Scheduling of Outages of Nuclear Power Plants with Time Windows. Technical Report RL-TR-96-157, Rome Laboratory, 1996.

[6] C. P. Gomes and D. R. Smith. Synthesis of Power Plant Outage Schedulers. Tech. Rep., Kestrel Institute, 1996.

[7] C. Le Pape. Scheduling as Intelligent Control of Decision-Making and Constraint Propagation. In M. Fox and M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[8] N. Muscettola. HSTS: Integrating Planning and Scheduling. In M. Fox and M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[9] PSDI. Managing Outages on the Desktop. Technical Brochure, 1994.

[10] N. Sadeh. Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler. In M. Fox and M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[11] D. Smith, E. Parra, and S. Westfold. Synthesis of High Performance Transportation Schedulers. Technical Report Tech. Rep. KES.U.95.1, Kestrel Institute, 1995.

[12] D. R. Smith. Structure and Design of Global Search Algorithms. Technical Report KES.U.87.11, Kestrel Institute, 1987.

[13] D. R. Smith. KIDS: A Knowledge-based Software Development System . In M. Lowry and R. McCartney, editors, *Automating Software Design*, pages 483–514. MIT Press, 1991.

[14] S. F. Smith. OPIS: A Methodology and Architecture for Reactive Scheduling. In M. Fox and M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[15] A. Tate. Integrating Constraint Management into an AI Planner. *Journal of Artificial Intelligence in Engineering*, 1994.

[16] A. Tate, B. Drabble, and R. Kirby. O-Plan2: An Open Architecture for Command, Planning and Control. In M. Fox and M. Zweben, editors, *Knowledge-Based Scheduling*. Morgan Kaufmann, 1994.

[17] R. J. M. Vaessens, E. H. L. Aarts, and J. K. Lenstra. Job Shop Scheduling by Local Search. Memorandum COSR 94-05, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1994.

[18] R. C. Wallace. A History of the Project Management Applications in the Utility Industry. *Project Management Journal*, September 1990.

[19] D. E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann Publishers, INC., 1988.

[20] D. E. Wilkins. *Using the SIPE-2^{TM} Planning System - A Manual for SIPE-2 Version 4.3*. SRI International, 1993.

[21] M. Zweben and M. Fox. *Intelligent Scheduling*. Morgan Kaufmann, 1994.