A Transformational Approach Applied to Outage Management of Nuclear 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 report on a successful project for transference of advanced planning and scheduling technology for outage management, a collaboration between Rome Laboratory, the Electrical Power Research Institute, Kaman Science, and Kestrel Institute, as part of DOD's dual-use program. The main goal of the project was to evaluate the use of transformational approaches and AI technology to solve real-world planning and scheduling problems involving complex constraints. ROMAN (Rome Lab Outage Manager) is the prototype system that was developed as a result of this project. ROMAN's main innovation compared to the current state of the art of outage management tools is its integrated approach to outage management automatically enforcing safety constraints during the planning and scheduling phase. Another innovative aspect of ROMAN is its generation of more robust schedules that are feasible over time windows. In other words, ROMAN generates a family of schedules by assigning time intervals as start times to activities rather than single point start times, without affecting the overall duration of the project.

1 Introduction

This paper focus on the real-world problem of outage management. An outage is a planned shutdown for refueling, repair, and maintenance. It is a rather daunting real-world task that may involve up to 45,000 activities. In the domain of nuclear power plants, risk and safety management are sine qua non conditions and therefore a planning and scheduling system (automatic or manual) has to enforce safety constraints guaranteeing that the state of the plant is safe at any time during an outage. The current automatic technology for outage scheduling used by the utilities does not take into consideration safety requirements — currently, safety and risk management still heavily rely on the experience of the manual schedulers, rather than on automatic procedures. Furthermore, in this domain, the existence of good automatic solutions is not only crucial for nuclear safety reasons but also for economic reasons — the cost per day of shutdown is in the order of $1,000,000.

We report on a successful project for transference of advanced planning and scheduling technology for outage management, a collaboration between Rome Laboratory, the Electrical Power Research Institute, Kaman Science, and Kestrel Institute, as part of DOD's dual-use program. The software environment selected for this project was KIDS (Kestrel Interactive Development System)[11], which is a set of semiautomatic tools to transform declarative problem specifications into correct and efficient programs.

ROMAN (Rome Lab Outage Manager) is the prototype system that was developed as a result of this project. ROMAN's main innovation compared to the current state of the art of outage management tools is its integrated approach to outage management automatically enforcing safety constraints during the planning and scheduling phase. Another innovative aspect of ROMAN is its generation of more robust schedules that are feasible over time windows. In other words, ROMAN generates a family of schedules by assigning time intervals as start times to activities rather than single point start times, without affecting the overall duration of the project.

Roman uses a rich representation for the state of the plant at any time (as in planning approaches) which

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

1060-3425/97 $10.00 © 1997 IEEE

allows for efficient constraint-based reasoning, in particular, temporal reasoning (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 for the generation of very fast schedules — constraints are *compiled* into the code, which is a novel aspect of our work using an automatic programming system, KIDS. In order to increase schedule robustness our approach entails the generation of families of schedules with the same completion time and that are feasible over time intervals.

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 give a brief overview of KIDS. In section 3 we describe the problem of planning and scheduling outages of nuclear power plants. Section 4 describes our approach to schedule synthesis. Section 5 describes ROMAN's interface and in section 6 we present performance results of the system, draw conclusions, and discuss future work.

2 About KIDS

KIDS is a program-transformation system. A user of KIDS develops a formal specification into a program by interactively applying a sequence of high-level transformations. KIDS' unique features are the algorithm design tactics (e.g., global search or divide-and-conquer) and its use of a deductive inference component. All the KIDS transformations are correctness preserving. KIDS runs on SUN-4 workstations and it is built on top of REFINE, a commercial knowledge-based programming environment and a high-level language. REFINE supports (a subset of) first-order logic, set theory, pattern matching, and transformation rules. Refine has its own compiler that generates CommonLisp code.

The steps to follow in order to build a program in KIDS are:

- Develop a *domain theory* to state and reason about the problem - the user defines appropriate types and functions that describe the problem. The user also provides laws that allow high-level reasoning about the defined functions - in particular distributive and monotonicity laws provide most of the laws that are needed to support design and optimization
- Apply a *tactic* - a well-structured but inefficient algorithm is created - the user selects an algorithm design tactic (method) from a menu and applies it to the specification. Existing tactics are: divide-and-conquer and global search (binary search, backtrack, branch-and-bound).
- *Simplifications, partial evaluation and finite differencing*, and other transformations are applied in order to improve the efficiency of the code
- *Compilation* of the code

KIDS uses a form of deductive inference called *directed inference* to reason about the problem specification in order to optimize the code, apply tactics, and derive filters [8, 9]. Directed inference allows the derivation of sufficient conditions (antecedents) as well as the derivation of necessary conditions (consequents). In directed inference a *source formula (S)* is transformed into a *target formula (T)* bearing a specified relationship (inference direction (*rightarrow*)) to the first and given certain *assumptions (A).* The *rightarrow* is a reflexive and transitive ordering relation between terms.

$$\textit{Find Some } (Target)(A \Rightarrow (Source(x_1..,x_m) \rightarrow Target(Y_1,..,Y_M)))$$

Directed inference performs first-order theorem-proving and formula simplification as special cases. The inference direction is one of the following: forward inference ($\Rightarrow$), backward inference ($\Leftarrow$), simplification ($=$), derivation of a lower bound ($>=$), and derivation of an upper bound ($<=$).

The inference process involves applying a sequence of transformations to the original form. The transformations are restricted to those that preserve the specified inference direction.

3 Planning and Scheduling of Nuclear Power Plant Outages

The planning and scheduling of the operations involved in the outages of nuclear power plants has a great impact in terms of the outage costs (replacement power, labor cost, etc.), use of scarce resources and implementation of safety procedures. Prior to 1979,

before the accident at Three Mile Island (TMI), refueling was the driving factor of outages of nuclear power plants: maintenance plans were governed by the projected duration of refueling activities. After the TMI accident, the focus turned to improving nuclear power plant effectiveness. The duration of an outage was determined not only by refueling activities, but by the work and plant modifications required to make the plant safer and more effective [6, 13]. Throughout the 1980s, backfits and the aging of nuclear power plants has reversed outage scope priorities and methodologies. Often the refueling activities no longer dictate the critical path in an outage.

3.1 Definition of Problem

The problem of planning and scheduling nuclear power plant outages can be stated as follows:

> Given a set of outage activities (refueling operations, repairs, modifications, and maintenance activities), a set of resources, and a set of technological constraints[1] assign times and resources to the activities in such way that the completion of the outage is minimized while safely performing all the activities required by the outage.

3.1.1 Activities

Depending on the planning and scheduling procedures of each particular plant, as well as the scope of the activities performed during the outage, the planning and scheduling of outages for nuclear power plants might involve up to 45,000 activities. During an outage several activities are performed, such as:

- Refueling operations
- Plant betterment
- Preventive maintenance
- Corrective maintenance
- Technical specification requirements for inspections or surveillance.

Relationships between activities are explicitly defined in *work order activities.*

Constraints between activities also arise as a result of different requirements in terms of feasible plans and schedules. Requirements regarding feasible plans and schedules are outlined in the next paragraphs.

[1] Precedence and temporal relationships between activities as well as resource constraints.

3.1.2 Plant Configuration and Risk Management

The general principle underlying the outage procedures is that outages should be as short as possible, maintaining the appropriate level of nuclear safety. In other words, the outage should be planned and managed to reduce shutdown risks through the appropriate consideration of *defense in depth* and preventive measures. The concept of *defense in depth*, used for the purpose of managing risk during shutdown consists of:

- providing systems, structures and components to ensure backup of key safety functions using redundant, alternate or diverse methods;
- planning and scheduling outage activities in a manner that optimizes safety system availability

Main safety functions and system components that are monitored to implement the concept of *defense in depth* are: electricity power control system, primary and secondary containment, fuel pool cooling system, inventory control, reactivity control, shutdown cooling, and vital support systems.

Figure 1 depicts an example of a decision tree regarding safety levels for a simple safety function, electricity power control.

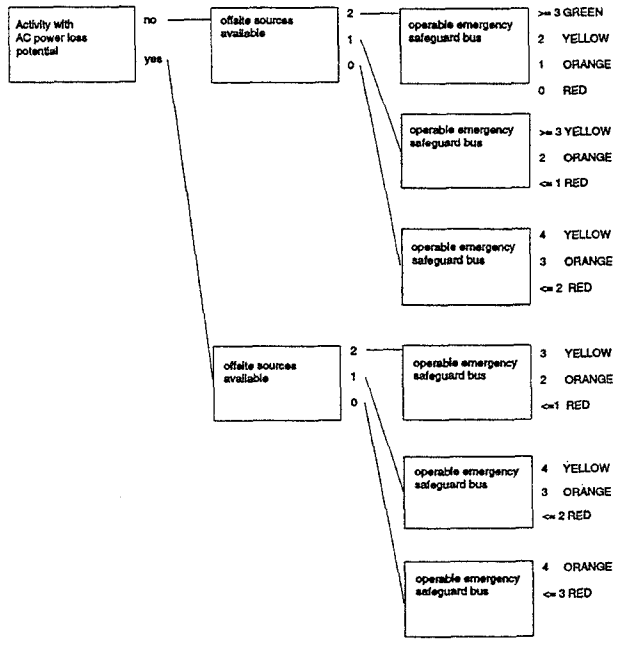

Figure 1. Example of a decision tree for the safety function AC Power

3.2 Why is the Outage Problem Hard?

Planning and scheduling problems are very complex. Their complexity can be categorized into two types [4]:

- *Intrinsic* complexity
  - NP-Hard problem
- *Extrinsic* complexity
  - The vast number of constraints (hard and preferential)
  - Several goals usually interacting in conflicting and unforeseen ways (difficulty in evaluating the solutions)
  - Dynamic and stochastic environment

The *extrinsic* complexity of the outage problem refers to the formulation of the problem in a real world context. It relates to identifying the constraints and goals that affect the validity and quality of schedules. Hard constraints, if violated, invalidate a schedule. While most hard constraints are obvious, they are also numerous and easy to overlook when manually planning and scheduling vast amounts of data. Preferential constraints are much more difficult to detect and incorporate in automatic systems. What appears to be a minor detail can have significant impact on the quality of the schedule. An error in the judgment can be much less obvious than a hard constraint violation and, while not invalidating the schedule, its quality might result very poor. The *extrinsic* complexity also refers to dealing with the dynamic and stochastic nature of the environment, which means that quite often plans and schedules have to be changed due to unforeseen events. Dealing with an uncertain environment requires a compromise between *predictiveness* and *reactiveness*. *Predictiveness* is the capability of making the best scheduling decisions assuming that unexpected events will not occur, while *reactiveness* is the capability of adapting an existing schedule to the constant changes that happen in the environment.

The *intrinsic* complexity refers to the complexity of the problem itself once formulated, which falls into the category of NP-hard problems [3]. Outage management is a particular instance of 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 [1, 12]. The practical consequence of dealing with an NP-hard problem is that the planning and scheduling of outages of nuclear problems, with realistic dimensions, cannot be optimally solved in a "reasonable" amount of time, even assuming that a rigorous mathematical formulation of the problems can be found, considering the current state of the art. Nonetheless, despite the difficulties, solutions have to be found for real world problems and therefore heuristic approaches have to be adopted, with some sort of guarantees on the quality of the solution.

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]

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

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 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.

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

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.

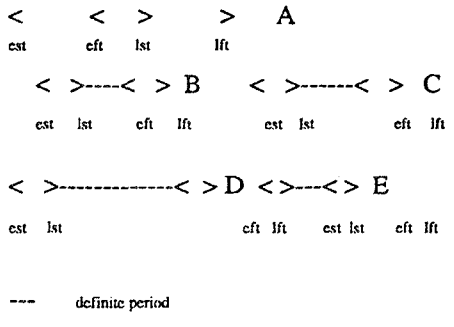

Figure 2. Notion of a definite period.

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.

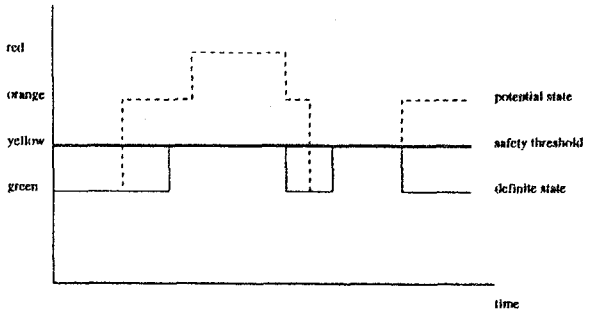

Figure 3. Definite and potential states of the plant.

4.1. Search and Control Mechanisms

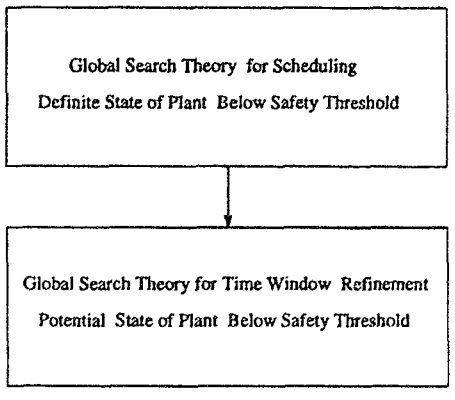

Figure 4. ROMAN's approach

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.

4.2. Global Search Theory

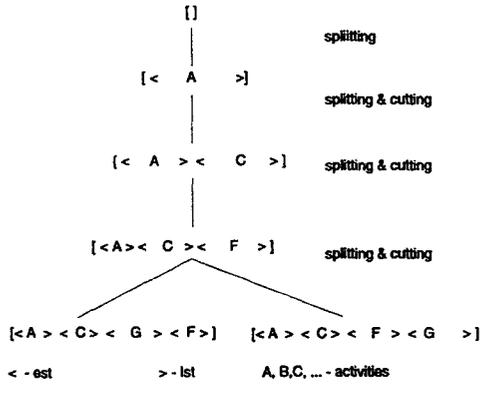

Figure 5. Global search theory for the Outage Problem

Global search [9, 7] 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

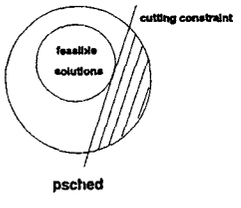

Figure 6. Cutting Constraints

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

[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.

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]

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

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

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

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

$$\xi(psched) \Longleftrightarrow \forall(sched)\ (sched \in psched \Longrightarrow \Psi(sched, psched))$$

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 *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 $psched_j$.

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

KIDS instantiates a program scheme for global search with constraint propagation, incorporating ξ. For more detail on propagation in KIDS see [7].

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:

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

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

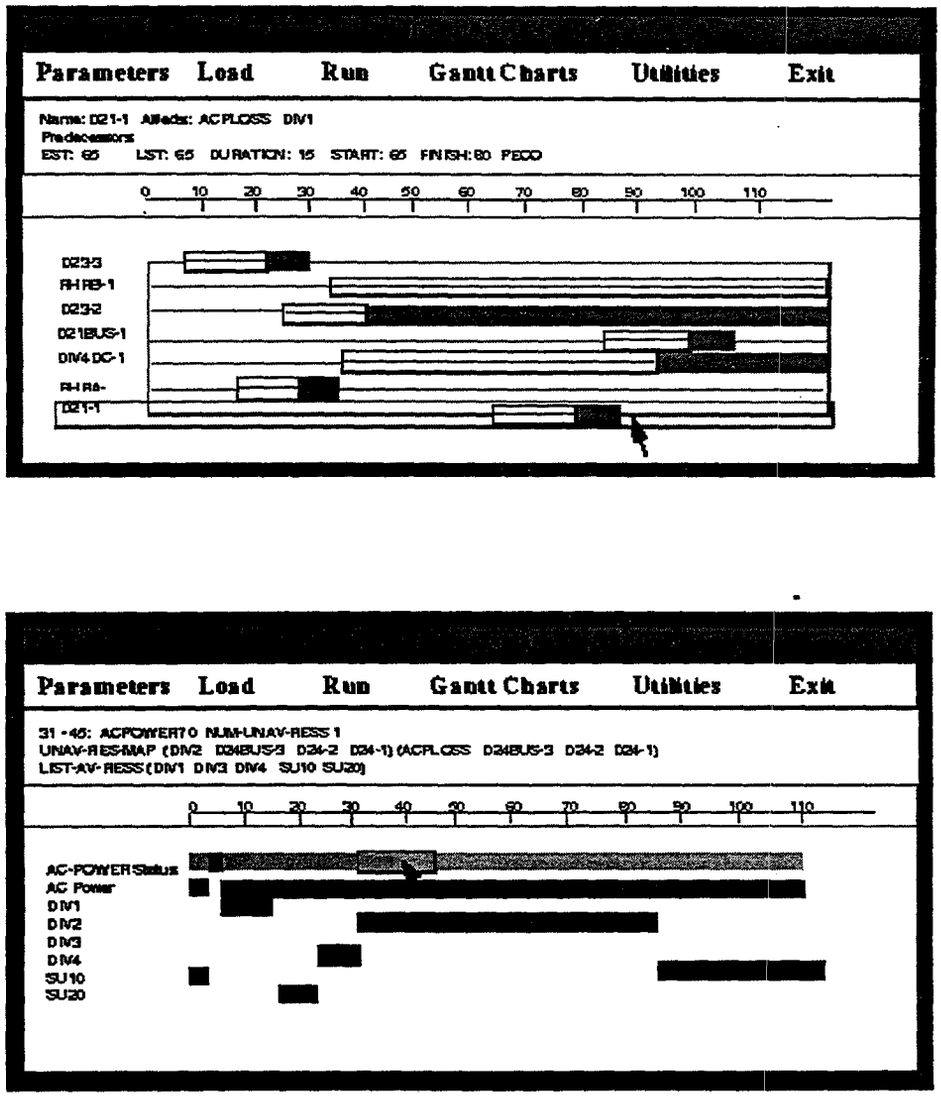

Figure 7. Screen Shots from ROMAN

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., [2, 7]. 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.

5. ROMAN's Interface

ROMAN currently comes configured with a GUI that displays an interactive Gantt chart for tasks. This Gantt chart displays the raw data, showing the relevant information for each task: task description, duration, predecessors, and effects of the task on the resources. ROMAN also allows the visualization of the final schedule in the activities Gantt chart, displaying the start time windows associated with each task.

Another Gantt chart shows the history of the state of the plant with respect to AC power. This Gantt chart displays the overall state of the plant in different colors (green, yellow, orange, and red), execution times of high risk evolution activities, i.e., activities that are considered of high risk regarding AC Power, as well as the pattern of resource utilization during the execution of the schedule. Figure 7 shows two screen shots from ROMAN.

6. Conclusions: Performance and Expected PayOff

ROMAN has proven successful since it clearly extends the current functionality offered by existing software tools for outage management:

- it incorporates all the technological constraints currently used for automatic generation of schedules,
- it offers the additional capability of generating schedules enforcing safety constraints,
- it generates more robust schedules guaranteeing feasibility over time windows,
- it generates schedules very fast — the current version of ROMAN handles up to 2,000 activities and it takes approximately 1 minute on a Sparc 2,
- the schedules generated are potentially better than the current solutions since new possibilities are explored.

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

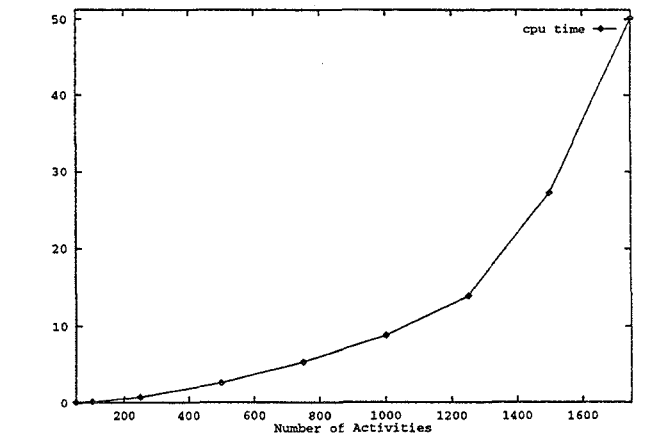

Figure 8. Time performance

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 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 and by AFOSR under the laboratory LRIR 92RL013. 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] J. Blazewicz, J. Lenstra, and A. R. Kan. Scheduling Projects to Resource Constraints: Classification and Complexity. *Discrete Appl. Math.*, 5:11–24, 1983.

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

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., San Francisco, 1979.

[4] C. O. P. Gomes and H. Beck. Synchronous and Asynchronous Factory Scheduling. *Journal of the Singapore Computer Society*, 5(2), 1992.

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

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

[8] D. R. Smith. Derived Preconditions and Their Use in Program Synthesis. In D. W. Loveland, editor, *Sixth Conference on Automated Deduction. Lecture Notes in Computer Science*, volume 138. Berlin: Spring-Verlag, 1982.

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

[10] D. R. Smith. KIDS: A Semi-automated Program Development System. *IEEE Transactions on Software Engineering, Special Issue on Formal Methods*, 16(9):1024–1043, September 1990.

[11] 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.

[12] 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.

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