Generating Satellite Control Schedules Using Case-Based Scheduling

Costas Tsatsoulis and Julian Holtzman

Lawrence Applied Research Corporation (LARC)
Lawrence, KS 66047
{tsatsoul,holtzman}@larc.com

Abstract

ICARUS is an intelligent system that integrates Case-Based Reasoning and utility theory to remove conflicts from control and task schedules of the Air Force's Satellite Control Network. We describe the methodology that integrates Case-Based Reasoning with utility theory, the problem domain of satellite task scheduling, and how ICARUS is applied to the problem.

Introduction

In this paper we describe on-going work that applies techniques from Case-Based Reasoning (CBR) and decision theory to remove conflicts from control and task schedules of the Air Force's Satellite Control Network (SCN). Scheduling of the SCN is a task of extreme significance to the Air Force since it is absolutely essential for data receipt from and transmission to satellites, vehicle maintenance, and orbit tracking and maintenance. Mission planners request contacts between their space vehicles (SVs) and SCN ground stations. These limited duration contacts serve three primary purposes:

- All SVs require orbit maintenance and state-of-health processing. A regular schedule of contacts is put in pace to facilitate the needed commanding and telemetry monitoring.
- Data collected from SVs must be transmitted to or from the satellite using the SCN. Since the amount of data can be significant, frequent and, possibly, lengthy, contacts may be required.
- Contacts for tracking and orbit determination are also scheduled. During these contacts no commands are transmitted, and only the accuracy of the orbit is determined.

Contacts for tracking and orbit determination are also scheduled. During these contacts no commands are transmitted, and only the accuracy of the orbit is determined.

An additional complexity arises from the fact that some satellites require equipment or capabilities that are not available at all ground stations. So, when scheduling, one must keep track of the availability of the required support equipment in addition to conflicts arising between duration of visibility windows among a number of SV's. Additionally, set-up times to configure the equipment must be considered as part of the time required to provide the support. Finally, ground stations themselves require periodic maintenance or emergency repairs. These activities preclude the use of the ground station or a portion of the stations's equipment for SV support.

Currently support requirements are expressed and submitted to the scheduling system as Program Action Plans (PAPs). PAPs may be used to specify time windows, support criteria, late starts or early stops, or support preferences such as a required antenna side or unacceptable equipment. PAPs are written in a simplified *ad hoc* language. Information supplied in a PAP may include service start time, service duration, setup time, start time lead and lag, equipment configuration, and station or station side. The totality of PAPs is essentially

the preliminary schedule for the stated time period.

The challenge of scheduling is to create a schedule that satisfies the needs of the users while not violating any of the constraints inherent in the SCN. The goals of a good schedule are to:

- optimize network utilization;
- maximize the number of satisfied requests;
- satisfy all high-priority requests; and
- ensure that no satellite is denied too many consecutive requests

Currently the Air Force uses human experts to schedule contacts between SVs and the SCN. Control of the scheduling process lies with the 22SOPS at Schriever AFB. The human schedulers use ASTRO, a set of tools for compiling, storing, displaying, and manipulating SCN resource requests and the resulting schedules (Loral 1995a). ASTRO, a DOS-based system, allows the human scheduler to enter schedule requests and manipulate this data to produce a network schedule. ASTRO features a large-screen monitor to display the schedule and a sonic pen used to manipulate the schedule. Although ASTRO provides a number of useful tools, the scheduling process, essentially a deconfliction and rescheduling problem, requires a significant amount of manual effort on the part of highly trained schedulers.

We are developing a methodology and a system which, when given a preliminary, conflicting, incomplete set of PAPs, performs deconfliction and generates an executable schedule of SV contacts with the SCN. The system, named ICARUS, is based on Case-Based Reasoning (CBR) and decision theory, and an innovative integration of case-based scheduling, skeletal scheduling, heuristics, and utility measurements.

Functionality of Completed System

Before proceeding with the description of the theory behind our work, we present how the completed system will operate so as to show the expected functionality of ICARUS. ICARUS operates in three different modes: normal scheduling mode; rescheduling mode; and user-driven mode.

Figure 1 demonstrates how the fully functioning ICARUS system behaves in normal scheduling mode. ICARUS receives the support requirements together with any station constraints, such as planned down times, and retrieves from its case base of successful plans fragments of schedules that satisfy as many of the requests as possible. The goal is to satisfy most scheduling requests using known, successful schedule fragments. This use of CBR in scheduling speeds up the process, generates a draft schedule quickly, and avoids the computational complexity of generative scheduling. Any requests not satisfied are scheduled by a generative scheduler which works from scratch. Since the old schedule fragments do not correspond exactly to the current requests, ICARUS adapts these fragments using the transmitted requests, and domain constraints and heuristics, using standard CBR adaptation techniques. The result is a set of schedule fragments that satisfy all requests, but are not integrated and deconflicted.

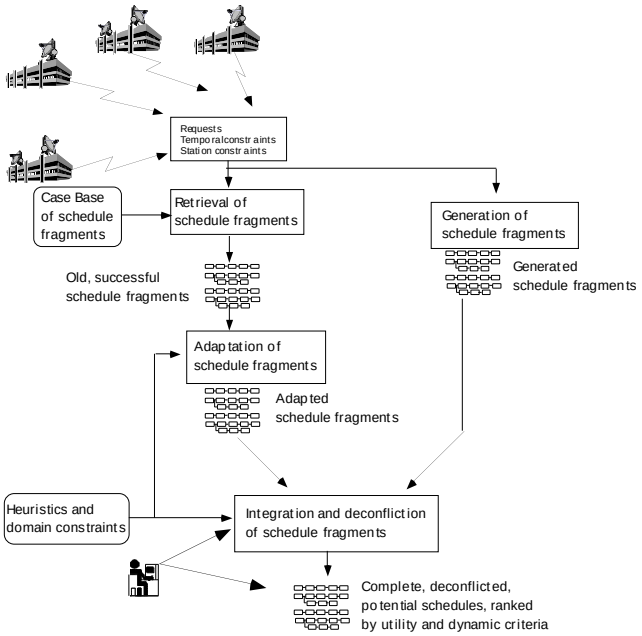

Figure 1: Operational diagram of ICARUS for normal scheduling of the SCN

Because we are using CBR, ICARUS has available to it at any time a set of draft schedules. While not integrated, complete, or deconflicted, these draft schedules provide early warnings of conflicts thereby allowing more time for their resolution. Additionally, adaptation and conflict resolution are step-wise processes, also called *anytime algorithms*; the more computational time is allotted to them the more correct the final schedule. If, though, the processes are interrupted before completion, ICARUS has always available a draft schedule.

Next, using the domain heuristics and constraints, ICARUS integrates all schedules and resolves as many conflicts as possible. Since multiple schedules are possible, some better than others, ICARUS generates multiple integrated schedules. Each schedule is evaluated based on multiple criteria, including the conflicts and hard constraints that could not be resolved, the heuristics that the schedule satisfies (e.g., are schedule event times easily divisible by five? is the schedule preserving large blocks of time for real-time adjustments? etc.), and any user-supplied optimization parameters. These criteria are weighted and the weight is automatically adjusted to conform to the current circumstances (e.g., a satellite that has been denied too many request has increased priority) and are also manually adjusted by the operator.

Figure 2 shows the operation of ICARUS in rescheduling mode. ICARUS receives the changed requirements and/or station and satellite constraints and uses its case base to change the current schedule and create a new one. Rescheduling is rapid, since it usually involves the retrieval of a new schedule fragment and the re-calculation of the utilities of the schedule.

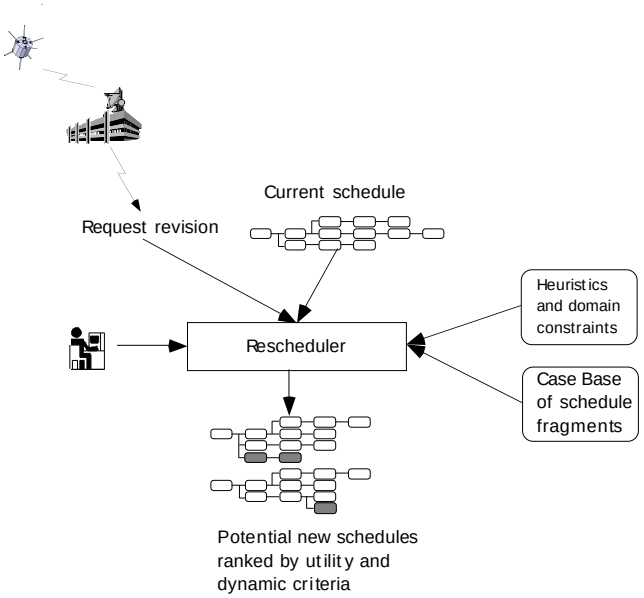

Figure 2: Operational diagram of ICARUS for rescheduling of the SCN

It is imperative that the operator can interface with ICARUS and can have direct input to the scheduling process. ICARUS employs a graphical user interface whose initial goal is to display graphically the whole SCN, the site configurations and constraints (such as, for example, scheduled down times), the resource scheduling requests, and the complete schedule,

with any unresolved conflicts clearly marked. The user is also able to view the criteria, constraints and heuristics used in scheduling, how they affected the final schedule, the other possible schedules or schedule fragments with an evaluation based on these constraints, and any *what if* scenaria the users decide to implement. The operator can also edit the schedule, and deconflict any remaining scheduling conflicts.

The ICARUS Methodology

Our methodology to case-based scheduling (CBS) integrates whole schedule cases, fragments of schedules (sub-schedules), and skeletal schedule prototypes into a unified framework. Case-based scheduling is similar to case-based planning, in that events and resources need to be organized in a temporal fashion to satisfy constraints, goals, and requirements. In the following we will use “schedule” and “plan” interchangeably.

A case is viewed as a collection of components - including schedules, components of physical systems, or specification elements - the ICARUS system takes advantage of this structure when adapting and re-using cases. Viewing a case as a collection of components allows adaptation through replacement of components by defining new sub-problems based on the overall problem goal and the internal problem solving state. Interconnections between components allow the re-use of schedule sub-structures rather than entire schedule.

Case Representation

Cases in a traditional case-base planner contain data of fine graininess, that is, low-level attributes and knowledge. However, in realistic domains a schedule can be extremely complex, and the ability to represent schedule case information on different levels provides a wealth of information including problem modularization and hierarchical organization. Our methodology generalizes the idea of case representation by: (1) including other types of memories in addition to completely instantiated schedule cases; and (2) taking advantage of the internal structure of schedule cases. In our work a complex schedule is represented by cases, sub-cases, and skeletal plan prototypes.

Each case is represented as a collection of schedule sub-components, organized in a specific sequence and governed by constraints. Knowing the structure of a schedule and the constraints organizing sub-schedules allows us to replace and re-use specific schedule actions or action sequences.

Human planners often use prescriptive information to plan and schedule actions: they know the structure of the finished plan without knowing the details of the individual actions that constitute it. This prescriptive information is similar to design prototypes (Gero 1990) and to skeletal plans (Iwasaki 1982). With the schedule's structure known, the missing actions can be determined and retrieved from the case base. ICARUS uses such structures as generic, high-level skeletal schedule prototypes. Each prototype instantiation is treated as a case-based reasoning problem, and the case-based reasoner retrieves and adapts sub-schedules (or collections of sub-schedules) for each prototype.

Cases and skeletal schedules in ICARUS consist of a task environment associated with the schedule memory, a set of features that describe the schedule memory, and a case or skeletal schedule. The task environment represents the global context of the schedule memory. This structure situates the memory in an execution environment, constraining possible solutions. Features describe various aspects of the schedule memory that may be useful in determining when the schedule structure might be re-used. The case or skeletal schedule is either a single action, or a partially ordered list of other memories.

One key element in ICARUS is the unified representation of skeletal schedules and cases. The concepts of features and task environment are universal for skeletal schedule prototypes and cases; the details of the schedule are simply missing in a skeletal schedule structure. Features still describe the skeletal schedule prototype in the same way that cases are described, allowing the skeletal schedule to be retrieved when appropriate. In addition, features support the proper instantiation of a skeletal schedule in a similar manner to the way that features can support the adaptation of a case. Thus, one memory structure is used to represent both an abstract skeletal schedule and a specific, episodic memory.

The environment structure associated with each schedule memory structure situates the schedule and provides a scheduling environment, giving the memory a global context. Environment includes such information as the physical environment associated with the system, objects and resources available for manipulation, temporal constraints, and additional necessary information not a part of the schedule itself. Generally, the goal of the scheduling process is expressed with respect to the scheduling environment.

Each schedule is a partially ordered set of independent memory structures, complete with features describing each. Both the partial ordering of the memory structures and the requirements features indicate how the memory structures together represent a more complex schedule. Each partial schedule is a free standing memory structure and can be accessed as such.
Skeletal schedule prototypes and cases differ in their contents: cases are completely instantiated, while skeletal schedules cannot be executed prior to filling in missing details.

Integrating Case-Based Scheduling with Utility Theory

In realistic scheduling activities the problem solver is faced with two major issues: how to deal with problem features that are unknown, and how to schedule in the presence of these unknowns. We have developed an innovative methodological approach to case-based reasoning that allows it to use utility theoretic approaches to deal with these two problems.

The retrieval of old cases in CBR is viewed as a decision problem, where each schedule from the case base provides an alternative solution and a prediction for the possible outcomes for the current problem. When uncertainty is encountered during case-based problem solving, decision theory is applied to evaluate each potential case in terms of the attributes that are significant for the current problem, so that the most desirable old case can be selected. Such integration provides a perfect complement between CBR and decision analysis.

Utility theory emphasizes making a choice among a set of alternatives. The criterion for optimal choice is the maximum *expected utility* of the projected outcomes, which would allow the decision maker to select among them. There are two fundamental ways to approach utility theory: a *normative model* or a *descriptive model*. The descriptive model of utility theory conjectures how things are or how they are behaving (French 1988). The normative model of utility theory describes how decisions should be made: given the probability of the events, that is the quantified probability distribution of uncertain states of nature, and the utility of the outcomes, that is the preference we have towards each alternative. Usually, the probability and utility values are treated as subjective judgments, expressing the knowledge, experience, and intuition of the expert (Raiffa 1968).

Although utility theory is an appealing problem solving framework, it often is not computationally manageable when the problem becomes complicated. In addition to its natural deficiency in alternative generation and outcome prediction, another difficulty results from its

inability to identify the decision variables when a large number of state variables are involved and to constrain the number of plans to evaluate. In our work we discovered, proved and demonstrated that CBR can solve many of the deficiencies of decision theory and many techniques in CBR can be used to enhance the performance of decision analysis. Both CBR and utility theory can benefit from each other by working together.

A critical phase during case-based scheduling is retrieval. If the case retrieved is very similar to the problem being solved, then adaptation and testing will be easy. If, on the other hand, the case retrieved is "far" from the current problem, it will require a lot of adaptation. To deal with uncertainties in CBR we treat the selection of the best case as a decision making problem. The task then became how to analyze and incorporate uncertainties, utilities, and preferences so that the factors influencing a decision be part of the case retrieval process. The issues resolved are: how to determine the decision variables, how to predict the possible outcomes, and how to establish subjective probabilities and utilities.

After the decision tree has been completed with all values, traditional utility theory (French 1988; Scholz 1983; Raiffa 1968) is used to rank the schedules based on the subjective probability and utility values. The methodology described offers a way to handle scheduling problems in domains of incomplete or continuous information. CBR can provide the most similar schedules, and utility theory can select the "best" schedule based on various criteria (such as, minimum scheduling conflicts, minimum resource utilization, optimality, and so on).

Conclusions

We are developing an intelligent system to support the schedule generation and deconfliction of satellite support schedules for the Air Force's SCN. The system, named ICARUS, is based on an innovative integration of CBR and decision theory, and a novel definition of Case-Based Scheduling, which integrates cases, sub-cases, and skeletal schedule prototypes. ICARUS is currently under development.

Acknowledgements

This work was supported by an SBIR Phase II Award number F29601-98-C-0042

References

French, S. 1988. *Decision Theory: an Introduction to the Mathematics of Rationality*. Ellis Horwood Limited.

Gero, J. S. 1990. Design Prototypes: A Knowledge Representation Schema for Design. *AI Magazine* 11(4): 27-36.

Iwasaki, Y.; and Friedland, P. 1982. SPEX: A Second Generation Experiment Design System. In *Proceedings of the National Conference on Artificial Intelligence*, 341-344. Cambridge, MA.: The MIT Press.

Loral Federal Services Corp. 1995a. Automated Scheduling Tools for Range Operations (ASTRO), Contract F04701-91-C-108, CDRL A058.

Loral Federal Services Corp. 1995b. CCSU Resources Scheduling Study Report Contract F04701-91-C-108, CDRL A115.

Raiffa, H. 1968. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*. Addison-Wesley.

Scholz, R. W. ed. 1983. *Decision Making Under Uncertainty*. North Holland.