# Software Architecture for a System Combining Artificial Intelligence Approaches for Ground Station Scheduling

Michele M. Van Dyne, Costas Tsatsoulis

***Abstract*****—** Scheduling of contacts between space vehicles (SVs) and ground stations is of extreme significance since it is essential for data transmission to and from satellites, vehicle maintenance, and orbit tracking and maintenance. We looked at the problem of scheduling contacts between SVs and the U.S. Air Force's Satellite Control Network (SCN). To address the scheduling problem, our work combines case-based reasoning, rule based systems, and generate-and-test techniques, all adopted from artificial intelligence. Our system creates a preliminary, daily SCN schedule with between approximately 500 to 1500 contact requests. The goal is to create a schedule with as few conflicting contact requests as possible, which is then finalized by expert schedule planners. We evaluated our system looking at its performance using only one scheduling algorithm and also using a combination of the algorithms. The system was tested on real SCN schedules and it achieved an average of 75.3% conflict-free over all SCN schedules tested. We also tested the system on schedules created by experts and which contained scheduling conflicts that the experts could not resolve; in these tests our system managed to resolve on average 44.4% of these conflicts, showing performance better than human expert schedulers. This paper addresses the software architecture of our system.

***Keywords*****—**Artificial intelligence, case-based reasoning, generate-and-test, rule-based systems, scheduling.

## I. Introduction

OUR work looked at the problem of scheduling contacts between space vehicles (SVs) and the U.S. Air Force's Satellite Control Network (SCN). Task scheduling of the SCN is of extreme significance to the Air Force since it is essential for data transmission from and to satellites, vehicle maintenance, and orbit tracking and maintenance. Mission planners plan contacts between their SVs and SCN ground stations.

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. 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.

This work was supported in part by US Air Force Contract # F29601-98-C-0042.

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 and ad hoc language.

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. A good schedule must achieve as many of the following objectives as possible:

- Optimize network utilization;
- Maximize the number of satisfied requests;
- Satisfy all high-priority requests; and
- Ensure that no SV is denied too many consecutive requests, where "too many" is program dependent.

Human expert schedulers use a number of heuristics to produce good, flexible schedules. Schedules constructed using these principles tend to be easier to modify when real-time changes are required.

In addition to the heuristics, the schedule has to adhere to many constraints and priorities. Constraints may also be flexible and defeasible. For example, high-altitude satellites are more flexible in their scheduling requirements, and turnaround or maintenance down times are padded and can be negotiated down to achieve a workable schedule.

Scheduling is done for different time frames with the shortest one being the 24 hour schedule. This schedule must be conflict-free and is produced manually in a series of steps, described in detail in [1].

The real-time schedule revisions are driven by events that lead to changes in real-time: ground station outages and satellite emergencies. When the support schedule is changed, notification is transmitted primarily by phone calls or face-to-face communications.

SCN Scheduling needs increased automation to deal with the following problems:

- Scheduling SCN assets is a difficult and complex task.
- Priorities are not clearly stated and not uniform from station to station or satellite to satellite.
- The current scheduling process is manpower intensive.
- The input of scheduling data and the manipulation of the schedule is manual, and the process of schedule deconfliction requires significant amounts of effort.

To address the problem of deconfliction we developed the ICARUS system (Integration of CAse-based Reasoning and Utility theory for Satellite schedule resolution). The basic technology of ICARUS is case-based reasoning (CBR), i.e. acting intelligently (in this case, performing task deconfliction) using previously successful experiences. CBR is well suited to the deconfliction task since expert schedulers often use the same strategies used in previous deconfliction sessions.

At the same time our research established that there are two more ways by which expert planners deconflict satellite contact schedules: First, there are some well-defined rules that experts use to deconflict schedules. These rules are simple and address simple conflicts, but they are also powerful in that they can address a large number of conflicts.

Second, when experts do not know or cannot design a solution, they revert to "generate-and-test" problem solving. Basically, they try a number of deconfliction techniques hoping that one of them will work. Such problem solving may sound random, but in reality, the techniques used are few, focused, and are developed by decades of experience. These techniques offer an alternate way of solving difficult scheduling problems.

ICARUS does not rely only on case-based reasoning to deconflict schedules. Our experience with domain experts showed that they combine problem solving techniques, and so does ICARUS.

Given a requested schedule that may contain hundreds of conflicts, ICARUS will apply deconfliction rules acquired from experts, will try different changes to the schedule in a generate-and-test mode, and will also use case-based reasoning for deconfliction. ICARUS allows the user to turn on/off the three different deconfliction methodologies (CBR, rules, and generate-and-test), and to perform an analysis of the efficacy of each methodology.

ICARUS was applied to real SCN schedules that had been requested by personnel responsible for particular satellites, and which had hundreds of conflicts. We also applied ICARUS on schedules that had been deconflicted manually by human experts, to see whether an automated system would improve on the performance of experts.

## II. RELATED RESEARCH

The area of satellite scheduling can be broken into two major sub-areas. The area addressed in this research is that of space-ground communications, often called the satellite range scheduling problem. The other main area of research in satellite scheduling is that of scheduling tasks on the satellite itself, often called satellite mission planning.

Many different approaches have been investigated in the area of satellite range scheduling. A relaxed version of the satellite range scheduling problem occurs in the area of non-commercial, primarily academic, satellite projects. As described by Schmidt and Schilling [2], under these conditions, ground stations are generally more flexible, contact windows can be shifted to other participating ground stations, time limits are not as strict as those in paid contact situations, and communications are not restricted to a single time window. Schmidt and Schilling describe two approaches to optimize schedules under these conditions: branch and bound, and hill climbing. Their results show that in a test with stations located in four countries, all requests were satisfied after an initial set of requests showed 42 conflicts, and in general, requests were satisfied in an equitable manner. It is not clear, however, how the two approaches were combined, if at all, in producing these results.

Marinelli, et.al. [3] addressed the satellite range scheduling problem using a Lagrangian heuristic. They framed the problem as a multiprocessor task scheduling problem, an approach originating in the operating systems domain. Their approach allowed a relaxation of constraints, which resulted in near optimal performance on large scale test problems.

Yang and Xing [4] combine learnable ant colonies with a knowledge model to improve scheduling performance for the satellite range scheduling problem. The ant colony searches the feasible domain while the knowledge model looks at previous iterations and discovers information that can then be used by subsequent iterations of the ant colony. They tested their approach on 40 generated instances and found that tasks with high priority were consistently scheduled first, while lower priority tasks may have had limits placed on their time windows. Most tasks were scheduled, and schedules produced a high utilization rate and a balanced load.

Howe, et.al. [5] discuss the issues associated with satellite range scheduling and introduce an initial software framework as a basis for approaching the problem. Some of the issues they discuss are that automated scheduling will never be able to completely generate schedules, and that human intervention will always be required, because the problem is over-constrained, and information can become available to human experts that will not be available to an automated system. Furthermore, the nature of the problem is such that an objective optimization function may not be realizable. It is difficult to assign a meaningful weighting and not all the information needed to resolve a conflict is available to an algorithm. Their initial approach involves framing the problem as a job shop scheduling problem and using slack-based and texture-based heuristics coupled with different search algorithms. They use a problem generator to generate realistic, though not real, problems. They also use a web-based interface so that humans using the system at different locations have access to the same system.

Barbulescu, et.al. [6] build on previous work on the satellite range scheduling problem and prove several interesting results. First, they show that the problem is NP-complete. They also show that the results from a reduced problem, that of single resource scheduling, do not generalize into the multi-resource problem. Simple heuristic approaches perform well on "easier" problems, circa 1992, but as the number of requests has grown larger, these approaches do not scale up. Finally, they show that a genetic algorithm approach yields the best results on larger, more complex problems, which are more representative of present-day communications traffic.

As recently as the early 2000's, a system called ASTRO was in use as the satellite range scheduling system for SCN resources. ASTRO was a set of tools for compiling, storing, displaying, and manipulating SCN resource requests and the resulting schedules. ASTRO was a DOS-based system that allowed the human scheduler to enter schedule requests and manipulate this data to produce a network schedule, though it did not automate the decision process. ASTRO featured a large-screen monitor to display the schedule and a sonic pen used to manipulate the schedule [7].

Case based reasoning has not been investigated much in the area of satellite range scheduling, but it has been used in other planning and scheduling areas. Related to ICARUS, described in this research, are case based planning systems, such as CaPER [8], a CBR system that uses high performance computing techniques for fast retrieval. CaPER also attempts to merge plans and to resolve harmful interactions between them. ForMAT uses CBR to retrieve old plans, represents temporal relationships, and assists the user for revisions and re-planning [9]. CABINS used CBR to schedule job shop activities. CABINS represented cases based on the temporal constraints they satisfied and the goals they achieved, and had a constraint-based scheduling component to iteratively repair schedules retrieved by CBR [10].

This paper focuses on the software architecture of ICARUS. A more complete description of the decision methodology contained in the software can be found in [11].

## III. ICARUS OPERATION AND ARCHITECTURE

### A. *Overview*

ICARUS takes as input a conflicted schedule and produces a schedule that has been deconflicted as much as possible. One constraint on the schedule input was that our system had to read and parse the schedule format produced by the tool being used, ASTRO. ICARUS was then required to output the deconflicted schedule in the same format. The input/output language is not effective for making scheduling decisions, however, so part of the process of parsing the input files was to generate data in a normalized database format.

Once a conflicted schedule and any necessary additional information was loaded into ICARUS, it iterates over its three deconfliction engines: rule-based, case-based, and generate-and-test. The user can control the number of iterations and which deconfliction engines can be used. The only constraint is that the user cannot change the sequence in which the engines are applied to the schedule.

ICARUS allows the user to view the original and the deconflicted schedules. The system also generates appropriate schedule statistics, such as total number of tasks, conflicts, and visibilities, and complexity of conflicts (i.e. the number of tasks conflicting with another task.)

Each of these processes is discussed below, and the overall architecture is shown in Figure 1.

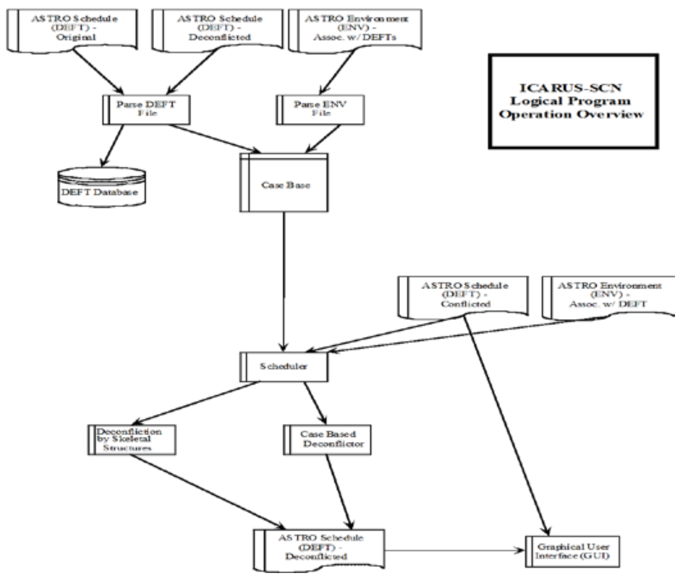
ASTRO Schedule (DEFT) - Original
ASTRO Schedule (DEFT) - Deconflicted
ASTRO Environment (ENV) - Assoc. w/ DEFTs
ICARUS-SCN Logical Program Operation Overview
Parse DEFT File
Parse ENV File
DEFT Database
Case Base
ASTRO Schedule (DEFT) - Conflicted
ASTRO Environment (ENV) - Assoc. w/ DEFT
Scheduler
Deconfliction by Skeletal Structures
Case Based Deconflictor
ASTRO Schedule (DEFT) - Deconflicted
Graphical User Interface (GUI)

Figure 1: ICARUS Architecture Overview

### B. *Input/Output – Parser and Inverse Parser*

The scheduling assistant tool used by the Air Force SCN was called ASTRO. The underlying language associated with ASTRO was ad hoc, and not structured in a way that it could be operated on programmatically. The first step, then, in building the system was to parse this language into meaningful and structured data. An example of two entries in the DEFT format used by ASTRO are:

```
QK350LION-BSTRNGTCS 0312201500003003080000060000 N N
N         008008        12202C008 STA TRNG, 30 MIN
BLK,    W=1800-2100Z,    NOT    W/OTHER    ANT    D/T..
S008                 ARTS                        AR33
L              1800-2100                 :30/:30/:30
QH905HULA-BSPMI HTS 0312163000040003081900060000 N N
N         013013 8      12166C013 PROTECTED PMI, -
00/+96  HRS  OF  MON/1700Z,  PREF  NO  SHIFT  CROSS  OF
17,01,09Z.         S013          ARTS            AR46
L                                            4/4/:30
N NT     CROSS OVER OK..KR
```

The DEFT files describe scheduling requests, while ENV files describe ground stations and available equipment and resources on those stations, covering the time period for the requests in the associated DEFT files. Both of these files are parsed into an object format which represents a normalized structure for SQL database storage. As an example, task requests have the structure shown in Figure 2.

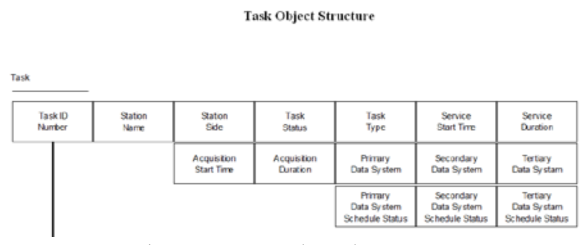

Figure 2: Parsed Task Structure

After completion of its deconfliction process, ICARUS performs the inverse parsing of its internal data representation structures to produce a file readable by the ASTRO program.

In addition to file I/O, ICARUS allows user control of certain operational parameters. A user can specify which of the three deconfliction engines, in combination or alone, can be used during the session and how many iterations the deconfliction process is allowed to make before stopping.

### *C. Database*

ICARUS uses the parsed tasks and environments to create a relational database. The database management system used is MySQL. ICARUS can create the database structure if needed, and populates the database with the task requests and environment data of the parsed DEFT and ENV files.

Figure 3 shows the overall layout of the database structure used by ICARUS in performing its deconfliction. While individual relation and field names in the image are not readable, the overall structure of the relations is evident in the diagram.

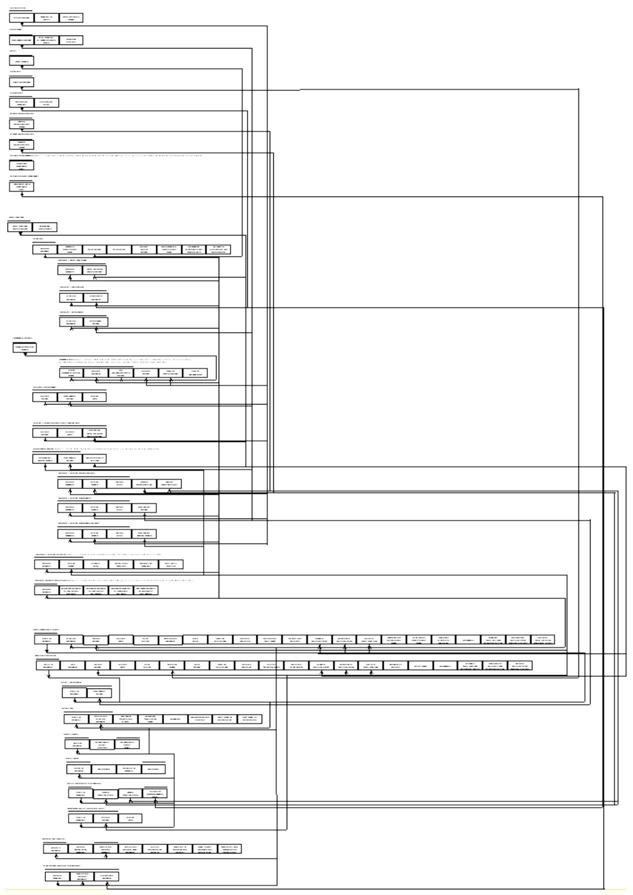
Figure 3: ICARUS Database Structure

### *D. Deconfliction Engines*

#### *1) Case-Based Reasoner*

In ICARUS a case contains a description of a conflicted task and the knowledge inside the case stores specific information about how the conflict was resolved. A case base is created by using two schedules for the same set of tasks. The first schedule (the "before" schedule) is the not yet deconflicted set of requests by the Satellite Operations Centers. The second schedule (the "after" schedule) is the deconflicted set of the same set of requests. Cases used in the ICARUS case base are those where deconfliction was performed by expert schedulers. ICARUS identifies the same task in the before and after schedules, making sure that the task had conflicts in the before schedule and has no remaining conflicts in the after schedule.

The case description is a description of the conflicted task. It is a descriptor of the context and specific details of the task, such as equipment it requires, its duration, its time constraints, and so on. It is meant to help the case-based deconflictor identify similar, conflicted tasks. The case description contains the information used in matching tasks, which are actual contacts between a satellite and a ground station.

The solution part of the case consists of the ways in which the conflict was resolved. This information is extracted by studying the task in the before and after schedules and identifying how the conflicted task was changed in the after schedule. The ICARUS case-based reasoner identifies the following changes to a task: 1.) change station; 2.) change station side; 3.) change start time; 4.) change turn-around-time; 5.) change duration; and 6.) change data system.

Given a case base, ICARUS uses it to resolve conflicts. To do so it selects the best case from memory by a weighted matching of all features of a conflicted task against all cases, followed by ranking of the cases by the matching weights. Each feature in ICARUS is given a weight between 0 (not used in matching) and 1. Features that are symbolic and single valued are matched one-to-one (binary match). Features that are numeric and single valued are matched by the value weighted by the inverse of the difference between the two values. Multivalued features are matched as the intersection of matching values (such multivalued attributes are, for example, the equipment list or the list of preferred stations). The matching value is weighted by the feature weight and all weighted values are summed to generate the final matching value for a case.

After the best case is selected, ICARUS attempts to apply the deconfliction solutions found in the case subject to the constraints defined in the task. If a case has more than one potential deconfliction action, ICARUS attempts to perform each one of the actions until it either deconflicts the task, or all steps fail.

Figure 4 shows the overall architecture of the case-based reasoning portion of the ICARUS system.

#### *2) Rule-Based Deconfliction*

Rule-based deconfliction is based on the way that expert schedulers initiate deconfliction of schedules. Schedulers attempt to first "slide" a task on the same station side, trying to find an opening wide enough to accommodate the task, and where equipment is available and the task is within its visibility window.

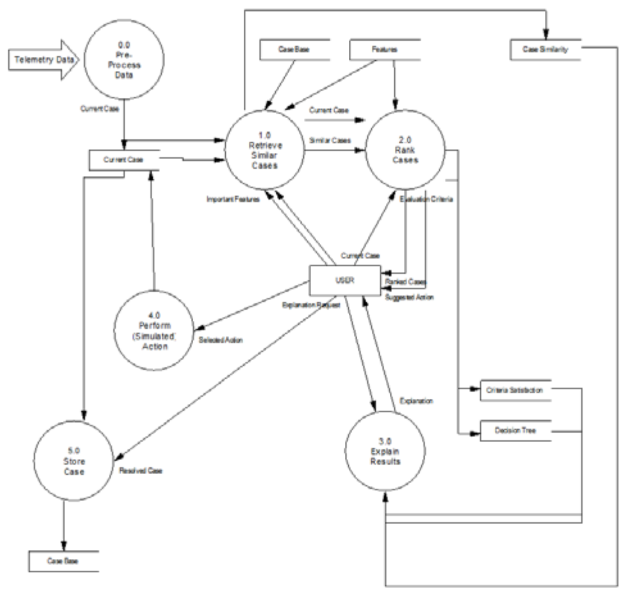

Figure 4: Case Based Reasoner Architecture Overview

ICARUS does the same thing. Without changing anything other than the start time of a task, it slides a task obeying the time constraints and making sure the task is visible and the necessary equipment is available. If the start time is fixed and the task start time has been defined incorrectly, the rule-based deconflictor will move the task to the correct time. The rule-based deconflictor also makes sure the open time it finds will accommodate the defined turn-around-time.

This process is described more formally using set theory constructs. Given are a set of antenna sites $\{S_i\}$, and a set of space vehicles $\{V_j\}$. In general, a vehicle $V_j$ is visible to a site $S_i$ for various time intervals during the day. There are a set of assigned tasks for the sites. This implementation of the rule-based portion of ICARUS attempts to move the time intervals to reduce the number of conflicts. First it reads the tasks for a given site, and for each task, stores the time interval, identification number, and task type. Let this set of time intervals be

$$T = \{t_1, t_2, \dots, t_n\} \tag{1}$$

These intervals are sorted according to starting time, then any set of sequential overlapping intervals are combined into a single interval. Then the resulting set

$$U = \{s_1, s_2, \dots, s_n\} \tag{2}$$

is unavailable time space. Initially no task could be moved to this space to eliminate a conflict. The available space is the complement

$$A = U^C. \tag{3}$$

For each task P to be shifted to the available space A, there is an associated visibility set, Vp, which is a set of time intervals, so the available space is refined as

$$A_p = A \cap V_p \tag{4}$$

Calling the ith overlapping set $O_i$, for each interval i in $O_i$, we examine the available space $A_p$, and find the translation seconds to shift I to the closest available space. If this is successful, we subtract the translated interval $I_T$ from A to get a new available space. We continue this until we reach the last element of the overlapping set $O_i$ and can either report success or failure.

### *3) Generate-And-Test Deconfliction*

We noticed that there was a finite set of actions schedulers (and ICARUS) can take to deconflict a task: change station, change side, change service start time, change turn-around-time, and change task duration. Consequently, we added one more deconfliction engine to ICARUS, one that cycles through all possible deconfliction actions until it finds one that resolves the conflict. In other words, this engine generates a possible deconfliction action and tests it to see if it will work. This is the "generate-and-test" deconfliction engine.

The major difference between case-based and generate-and-test deconfliction is that the former elects to perform only the best deconfliction actions based on its experience, while the latter will try all steps.

Each change is tested against the same constraints as the changes performed by the case-based deconfliction engine. The sequence of changes attempted in this approach was established in collaboration with experts, and represents an increasing disturbance of the task. So, ICARUS will first attempt the least intrusive changes, the ones that leave the scheduling requests as unchanged as possible, and will increasingly disturb these requests (within constraints) until a solution is found.

## *E. Viewer*

As deconfliction progresses, ICARUS displays messages and progress bars on the screen to keep the user informed. The user may also view the schedule before and after deconfliction using our schedule viewer. An example of this is shown in Figure 5.

The x-axis is time, here spanning four and a half days. The span depends on the input schedule. The y-axis consists of the satellite stations, read in from the ENV file. The color bars represent the satellite contact tasks (a different color for a different contact), red bars represent conflicts in the schedule, and green bars represent station maintenance tasks.

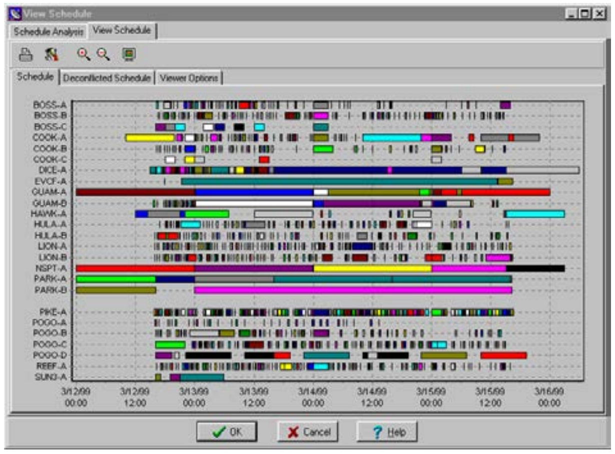

Figure 5: Graphical View of Conflicted Schedule

## IV. RESULTS

We evaluated our system on actual SCN schedules and some of the results are shown on Table 1. The schedules started with a number of tasks and conflicts due to conflicting contact requests. ICARUS used all its deconfliction methods sequentially (rule based, CBR, and generate-and-test), and the results of each deconfliction step are listed in the table. For example, the first schedule shown in Table 1 started with 379 conflicts and after applying the rule based deconflictor there were 301 remaining conflicts, after applying CBR there were 300 conflicts, which were then lowered to 244 by generate-and-test. The system iterated three times, and stopped after the conflicts did not change after an iteration cycle.

We tested our system on SCN provided schedules, and after ICARUS the average schedule was 75.3% clear of conflicts. We also tested ICARUS on schedules created by experts and which contained scheduling conflicts that the experts could not resolve; in these tests our system managed to resolve on average 44.4% of these conflicts, showing performance better than human expert schedulers.

Table 1: Sample Deconfliction Results

| Tasks | Initial Conflicts | Iterations (Orig→RuleBased→CBR→G&T) | Final Conflicts |
|---|---|---|---|
| 562 | 379 | 379→301→300→244<br>244→241→241→239<br>239→239→239→239 | 239 |
| 717 | 69 | 69→63→63→23<br>23→22→22→22<br>22→22→22→22 | 22 |
| 535 | 241 | 241→138→135→116<br>116→116→116→116 | 116 |
| 587 | 300 | 300→202→199→163<br>163→161→161→161<br>161→161→161→161 | 161 |
| 1505 | 617 | 617→458→447→352<br>352→350→350→348<br>348→348→348→348 | 348 |

## V. CONCLUSIONS AND FUTURE WORK

Our work addressed an important operational need of satellite control networks: how to resolve conflicting requests for access to the ground station by space vehicles. This problem is different from traditional scheduling or planning ones, since it starts with an existing schedule which corresponds to scheduling requests. These requests are often conflicting, and require correction.

There are two potential extensions to ICARUS, both improving its deconfliction performance. ICARUS could implement hand-offs between ground stations. In other words, a contact task could be shared between two stations, if it could not be fully satisfied at one station. A large number of conflicts can be resolved if a long contact request can be broken into smaller contacts that are distributed over a set of stations. Also, certain resources can be shared by multiple tasks. Allowing sharing of resources and equipment will improve deconfliction performance.

There will almost always be conflicts in every schedule that cannot be resolved because of hard constraints. These scheduling requests are denied by human users, something our system is not allowed to do. Consequently, regardless of improvements to our system, it will never generate a 100% conflict-free schedule.

## REFERENCES

[1] Loral Federal Services Corp. 1995. CCSU Resources Scheduling Study Report Contract F04701-91-C-108, CDRL A115.
[2] M. Schmidt, and K. Schilling. 2009."A Scheduling System with redundant scheduling capabilities." International Workshop for Planning and Scheduling in Space, Pasadena, USA. 2009
[3] F. Marinelli, S. Nocella, F. Rossi, & S. Smriglio. 2011. "A Lagrangian heuristic for satellite range scheduling with resource constraints", Computers and Operations Research, 38 (2011), 1572-1583.
[4] K. Yang and L. Xing. 2012. "The Learnable Ant Colony Optimization to Satellite Ground Station System Scheduling Problems", Electrical Review, R. 88, NR 9b/2012, 62-65.
[5] A.E. Howe, L.D. Whitley, L. Barbulescu, J.P. Watson. 2000. "Mixed Initiative Scheduling for the Air Force Satellite Control Network", Second International NASA Workshop on Planning and Scheduling for Space, March 2000.
[6] L. Barbulescu, J.P. Watson, L.D. Whitley, A.E. Howe. 2004. "Scheduling Space-Ground Communications for the Air Force Satellite Control Network", Journal of Scheduling, Vol. 7, Issue 1, pp. 7-34, January.
[7] Loral Federal Services Corp. 1995. Automated Scheduling Tools for Range Operations (ASTRO), Contract F04701-91-C-108, CDRL A058.
[8] Hendler et al. 1994. "Massively Parallel Support for Case-Based Planning," Proc. of APA/Rome Lab Planning Initiative Workshop, Morgan Kaufmann.
[9] Mulvehill, A. 1995. "Reusing Force Deployment Plans," AAAI Fall Symposium on Adaptation of Knowledge for Reuse.
[10] Miyashita, K. and K. Sycara. 1994. "Adaptive Case-Based Control of Schedule Revision," in: Intelligent Scheduling, M. Zweben and M.S. Fox (Eds.), San Francisco: Morgan Kaufmann, 291-308.
[11] Tsatsoulis, C., and Van Dyne, M. " Integrating artificial intelligence techniques to generate ground station schedules", Proceedings of the 2014 IEEE Aerospace Conference, March 1-8, 2014, Big Sky, MT.