SATELLITE TELE-COMMUNICATIONS SCHEDULING AS DYNAMIC CONSTRAINT SATISFACTION

Christian Plaunt
Caelum Research Corporation
NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035, USA
phone: +1 650 604 2928
fax: +1 650 604 3594
plaunt@ptolemy.arc.nasa.gov

Ari K. Jónsson
RIACS
NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035, USA
phone: +1 650 604 2799
fax: +1 650 604 3594
jonsson@ptolemy.arc.nasa.gov

Jeremy Frank
Caelum Research Corporation
NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035, USA
phone: +1 650 604 2524
fax: +1 650 604 3594
frank@ptolemy.arc.nasa.gov

ABSTRACT

We consider a single satellite telecommunications problem in which a dynamic set of calls must be assigned to beams of the satellite. These assignments must satisfy beam-coverage constraints, capacity constraints and requirements based on the priorities of incoming calls; additionally, the satellite must respond quickly to the changing call load and environment. We show how this problem can be solved by using constraint satisfaction technology. We model the problem as a Dynamic Constrained Optimization Problem (DCOP) and present an algorithm inspired by hill-climbing search. We present empirical results from a simulation showing that the algorithm meets the requirements imposed by the problem domain, and finds solutions that are within 2% of the optimal.

1. INTRODUCTION

Advances in satellite communications technologies have given rise to new challenges for automated allocation techniques for dynamic resources such as satellite and bandwidth availability. In this paper, we look at a particular class of satellite tele-communications scheduling problems, and present an approach for solving those problems effectively within strict real-world time limits. The approach is based on viewing the satellite communications scheduling problem as a Dynamic Constraint Optimization Problem (DCOP). This well-known class of automated reasoning problems provides us with a well-defined framework and a number of possible approaches to solving such problems.

We present an automated dynamic scheduling method based on an encoding of this problem as a dynamic constrained optimization problem. The algorithm we selected to solve this problem was inspired by the local search paradigm, which provides good solutions to such problems in real-time. Experimental results show the scheduling method to be very effective at finding near-optimal solutions, even in the face of degraded communications capabilities.

Our problem domain consists of a single satellite with b communication links or beams, which cover a set of g ground stations. Each call is assigned a priority, and there are p priorities available. The satellite must support a dynamically changing set of call requests. Each call request consists of a source ground terminal, a destination ground terminal, a number of units of bandwidth, and a priority. A call must be assigned to both an uplink beam and a downlink beam or be rejected. The uplink beam must cover the source ground terminal and the downlink beam must cover the destination ground terminal. Each beam has a maximum capacity for uplinked and downlinked calls. That is, the total bandwidth of uplinked calls on a beam may not exceed a capacity c_u, and the total bandwidth of downlinked calls on the same beam may not exceed a capacity c_d. Furthermore, calls of priority p_i outweigh all calls of strictly lower priority; if there is not enough space capacity on a beam for an incoming call of high priority, the system is required to disconnect enough calls of lower priority to accommodate the new, higher priority call, if possible. Each call has a finite but unknown duration, so calls periodically are released, thereby freeing more capacity for new calls. Calls arrive at arbitrary intervals. A call does not change priority, source or destination station while it is in progress. However, the capacity of beams may increase or decrease, forcing some calls to other beams or requiring the premature termination of some calls.

The problem of managing calls on the satellite consists of the following:

1. When a new call arrives, the satellite must decide whether to accept the call or reject it; if the call is accepted, the satellite must decide which beams the call will utilize.

2. When the capacity of a beam changes, the satellite must decide whether to move or terminate any

Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space, 1-3 June 1999 (ESA SP-440)

calls, and if so, which ones.

If we consider the satellite at any instant, there is a list of call requests in the system. The problem is to decide which calls to accept and which to reject, and then to assign each call uplink and downlink beams in such a way that each call's coverage requirements are satisfied and the capacity constraints on the satellite beams are met. A problem of this type can be encoded as a *Constraint Satisfaction Problem* or CSP. Informally, a CSP consists of a set of variables, a description of the possible values each variable can take on, and a list of constraints which define valid assignments to sets of variables. The problem we have described also includes a preference for assigning calls of high priority to the satellite. Adding such a preference order among solutions satisfying the constraints results in a *Constrained Optimization Problem* or COP. CSPs and COPs have been heavily studied, and many theoretical and practical results can be brought to bear to address such problems; for work on CSPs in general see Haralick & Elliot (1980), Nadel (1989), and for a specific application see Banerjee & Frank (1996). However, as calls arrive and depart, we have not just one but a *sequence* of such problems. These problems are closely related, as each problem in the sequence is derived from an earlier problem by the termination of an existing call, the addition of a new call, or the reduction in capacity on a beam. A modification of CSPs known as *Dynamic Constraint Satisfaction Problems* or DCSPs can be used to encode the sequence of problems which results from the arrival and departure of call requests.

There are a number of methods for solving CSPs and DCSPs. However, in this domain, the solver must meet performance requirements imposed by the telecommunications application, as the satellite must be able to respond rapidly to new call requests as well as to changes in the available bandwidth on the beams. The solver must nonetheless provide good solutions (i.e. allocate high priority calls) which are also valid (i.e. meet the coverage constraints and do not exceed beam capacity). In addition, the fact that we are presented with a sequence of closely related problems suggests that any algorithm to solve the sequence of problems *reuse* the solution to the previous problem to increase the speed of the solver. *Hill-climbing* algorithms for CSPs operate by perturbing solutions in order to find better solutions which are nearby. These methods have good problem solving performance in general Selman *et al.* (1992), Minton *et al.* (1990), and also promote the reuse of solutions between successive problems in the DCSP framework Freuder & Wallace (1998).

The rest of the paper is organized as follows. In Section 2 we formally define CSPs and DCSPs and discuss methods of solving these problems, including hill-climbing methods. In Section 3 we formally describe the satellite telecommunications problem as a Dynamic Constrained Optimization Problem (DCOP). In Section 4 we present a hill-climbing algorithm for responding to new connection requests and reductions in the capacity of beams. We also give a bound on the complexity of this procedure, and establish that it can respond to changes in real-time. In Section 5 we present the results of applying the hill-climbing algorithm to a set of telecommunications requests taken from a real satellite telecommunications problem. We show that hill-climbing can consistently find solutions within 2% of the best possible solution. In Section 6 we conclude and discuss some opportunities for future work.

2. DYNAMIC CONSTRAINT SATISFACTION AND OPTIMIZATION PROBLEMS

In this section we present the formal machinery we will use to solve the satellite telecommunications problem. We shall first formally define CSP, COP, DCSP and DCOP, then discuss hill-climbing algorithms to solve these problems.

2.1 CONSTRAINT SATISFACTION PROBLEMS

A *Constraint Satisfaction Problem* or CSP is a triple $P = (V, D, C)$, where:

1. $V = \{v_1, \ldots, v_n\}$ is a set of variables
2. $D = D_{v_i} \mid i \in \{1, \ldots, n\}$ are the domains of the variables, where each D_{v_i} is a finite set of possible values of v_i.
3. C is a set of constraints (Y_j, R_j), where each constraint consists of a scope $Y_j = \{v_{i_1}, \ldots, v_{i_k}\} \subseteq V$ and a relation $R_j \subseteq \prod_{\nu=1}^{k} D_{v_{i_\nu}}$.

It is worth mentioning that if the domains D_{v_i} are large and the scope contains many variables, then explicitly enumerating the relations of the constraints may be quite cumbersome. Consequently, relations are often written in a condensed form. For instance, if the variable domains are subsets of the integers, we can write relations as equations such as $C_1 = \{(x, y,), x + y < 5\}$ rather than enumerating all the legal pairs of values of x and y.

A *valid solution* to a constraint satisfaction problem $P = (V, D, C)$, where $V = \{x_1, \ldots, x_n\}$, is an n-tuple $(v_{x_1}, \ldots, v_{x_n})$, such that:

1. $v_{x_k} \in D_{x_k}$ for $k = 1, \ldots, n$, and
2. For any $(Y, R) \in C$ with $Y = \{x_{i_1}, \ldots, x_{i_k}\}$, we have $(v_{x_{i_1}}, \ldots, v_{x_{i_k}}) \in R$.

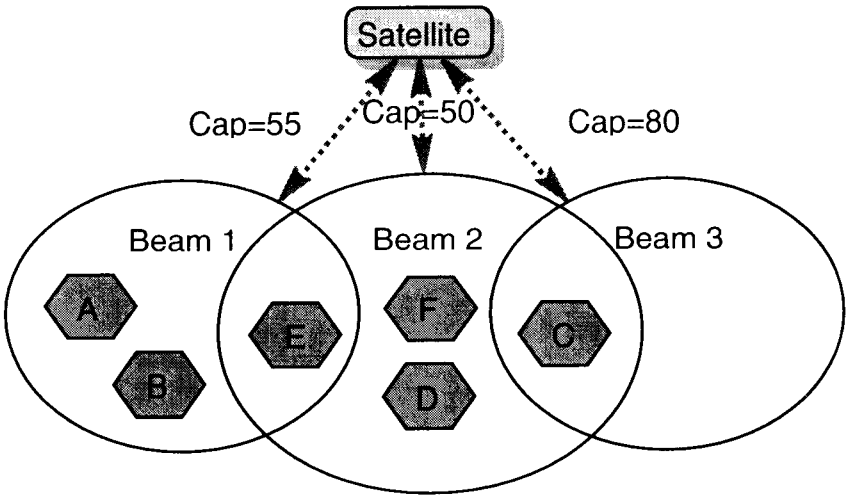

Figure 1: A network consisting of one satellite with three uplink/downlink beams. Beam 1 of capacity 55 covers stations A, B and E, beam 2 of capacity 50 covers stations D,E,C and F, and beam 3 of capacity 80 covers station C.

2.2 Constraint Optimization Problems

Many problems consist of both constraints and an *optimization criteria* which differentiates between valid solutions. For example, in our telecommunications domain, an allocation of calls to the beams of the satellite may satisfy the constraints, but we prefer assignments which assign more high priority calls to the satellite. To formalize this concept, we define a *Constrained Optimization Problem* or COP as a pair (P, g) where P is a constraint satisfaction problem and g is a function that maps every valid solution of P into $\mathbb{R}$. The goal of constraint optimization is to find a valid solution that maximizes g.

As an example of a COP, let us consider a simple scheduling problem consisting of two low-capacity beams and a few call requests (disregarding geographic constraints for simplicity). The calls must be assigned to beams or rejected, and the constraints limit the overall bandwidth requirements for all calls assigned to a beam. Finding a satisfactory assignment of calls to beam 1, beam 2 or rejection, is a CSP. If we now specify a preference among solutions, or equivalently, specify an optimization function on the set of valid assignments, we have a COP. In other words, the CSP defines a set of valid call assignments, and the preference functions defines an ordering among solutions.

2.3 Dynamic Constraint Satisfaction Problems

As mentioned earlier, the satellite telecommunications problem is not a CSP, since calls are constantly being added and deleted. However, the changing set of calls can be represented as a sequence of closely related CSPs. To formalize this notion, let $P = (V, D, C)$ be a constraint satisfaction problem. Any problem of the form $Q = (V', D', C')$ such that $V' \supseteq V$ (i.e. there are more variables), $D'_v \subseteq D_v$ for each $v \in V$ (i.e. there are fewer legal values for variables) and $C' \subseteq C$, (i.e. there are fewer legal combinations for variables in a constraint) is a *restriction* of P. Any problem of the form $Q = (V', D', C')$ such that $V' \subseteq V$ (i.e. there are fewer variables), $D'_v \supseteq D_v$ for each $v \in V$ (i.e.there are more values for variables) and $C' \supseteq C$ (i.e. there are more legal combinations for variables in a constraint), is a *relaxation* of P. A *Dynamic Constraint Satisfaction Problem* or DCSP is a sequence of constraint satisfaction problems $P_0, P_1, \ldots$, such that each problem P_i is either a *restriction* or a *relaxation* of P_{i-1}. This definition is consistent with similar definitions given in Dechter & Dechter (1988) and Verfaillie & Schiex (1994).

Not surprisingly, it is relatively straightforward to generalize the idea of dynamic constraint satisfaction to dynamic optimization problems. Formally, a *Dynamic Constrained Optimization Problem* or DCOP is a sequence of optimization problems, such that each entry is a relaxation or a restriction of the previous problem. This means that the optimization function remains unchanged throughout, but the set of variables, domains and constraints may change.

2.4 Search Methods and Hill-Climbing

There are two main families of procedures for solving CSPs and COPs. *Complete* methods are guaranteed either to find a valid assignment of values to variables or prove that no such assignment exists. Complete methods frequently exhibit good performance, and guarantee

a correct and optimal answer for all inputs. Unfortunately, they require exponential time in the worst case, which is not acceptable for the satellite telecommunications domain.

Recently, researchers have become interested in *incomplete* search methods which do not guarantee correct answers for all inputs. These methods can find satisfying assignments for solvable problems with high probability. These incomplete algorithms have gained popularity in recent years, due to their simplicity, speed and observed effectiveness at solving certain types of problems.

Hill-climbing is one of the most popular incomplete approaches to solving constraint satisfaction problems. These algorithms map assignments to a set of assignments by making minor changes to the original assignment. Each element of the set is evaluated according to some criteria designed to move closer to a valid assignment and/or improve the evaluation score of the state. The best element of the set is made the next assignment. This basic operation is repeated until either a solution is found or a stopping criteria is reached. A hill-climbing algorithm requires two components: a *candidate generator* which maps one solution candidate to a set of possible successors, and a *evaluation criteria* which ranks each valid solution (or invalid full assignments), such that improving the evaluation leads to better (or closer to valid) solutions.

To take a concrete example of hill-climbing, consider the following scenario for a slightly unrealistic satellite telecommunications problem. We have a satellite with only one beam and one station. Assume we have assigned a call of priority 5 requiring a bandwidth of 2, to a single given beam with capacity of 4 units. Two calls are currently rejected, one with priority 3 and bandwidth requirement of 3, and the other with priority 7 and bandwidth requirement of 1. This current solution could be described as $(\{C_{5,2}\}, \{C_{3,3}, C_{7,1}\})$, with the first set being calls assigned to the beam and the second set consisting of rejected calls.

Let us then choose a simple optimization function, which sums up $(10 - p) \cdot b$, where p is the priority (1 highest, 8 lowest) and b is the bandwidth used. Our current solution then evaluates to $(10 - 5) \cdot 2 = 10$. Our successor function might then give the following options:

$$(\{C_{3,3}\}, \{C_{5,2}, C_{7,1}\})$$

which evaluates to 21, and

$$(\{C_{5,2}, C_{7,1}\}, \{C_{3,3}\})$$

which evaluates to 13. We therefore pick the first candidate as the new current solution.

A second hill-climbing iteration would then result in

$$(\{C_{3,3}, C_{7,1}\}, \{C_{5,2}\})$$

which is indeed an optimal solution at 24.

Hill-climbing algorithms do not always find optimal solutions for real problems. However, hill-climbing methods have the distinct advantage that they can often provide a valid solution at any time-point. This makes the technique very suitable for systems that must perform with real-time guarantees. An added bonus is that the more time the hill-climbing process is given, the better the solution will typically be. Gent & Walsh (1993) Finally, hill-climbing is especially attractive for DCSPs, because it is likely that the solution to problem C_i is a good starting assignment for problem C_{i+1}. Freuder & Wallace (1998) For these reasons, we chose to base our solution to the satellite telecommunications problem on hill-climbing.

3. THE PROBLEM AS A DCOP

As mentioned earlier, our problem domain consists of a single-satellite, multiple ground-station communications network with variable connection coverage and varying bandwidth due to technical glitches, maintenance and other factors. In order to handle a communications request, we need to allocate sufficient bandwidth from the source terminal node to the central node, along a link that covers that node, and from the central node to the destination node, along a link covering the destination.

Let us assume that we are given a single satellite communication assignment problem with n calls, b beams, g ground stations, p priorities and s ground stations. Each beam has capacity of d_i slots for downlink, u_i slots for uplink.

A satellite communication assignment problem is defined by:

- $\{t_1, \ldots, t_g\}$, a set of ground stations
- $\{l_1, \ldots, l_b\}$, a set of links between a set of ground stations and the satellite
- $\mathtt{cap}_u(l_i)$, function identifying the uplink capacity of each link
- $\mathtt{cap}_d(l_i)$, function identifying the downlink capacity of each link
- $\mathtt{cover}(l_i, t_j)$ a predicate indicating whether beam i covers the location of ground station j
- $\{c_1, \ldots, c_n\}$, set of calls, including start times and durations.
- $p(c_i)$, function giving priority of call
- $\mathtt{use}(c_i)$, function indicating bandwidth required for call

- $\texttt{source}(c_i)$, indicates the terminal source of call
- $\texttt{dest}(c_i)$, indicates the terminal destination of the call

To describe our problem as a DCOP, we first determine what our variables and values are. The key decisions are how each call is routed, i.e, which beams the uplink and downlink are assigned to. We define two variables u_i and d_i for each call that is in the system, the uplink-beam and the downlink-beam. The values for these variables include all possible beams, but we also need a value to represent that a call is rejected. So, for each call variable u_i or d_i, we have a set of links that we can assign to it, $\{l_1, \ldots, l_b\}$, and a flag indicating that the call is rejected. To facilitate the specification of this problem, let us represent this domain as the numbers from 0 to b, with 0 standing for the rejection flag and $i \in (1, \ldots, b)$ standing for $(l_1, \ldots, l_b)$ respectively.

The constraints that must be satisfied are the following:

- For each link l_i,

$$\sum_{j=1}^{n} I(u_j = i)(\texttt{use}(c_j)) \leq \texttt{cap}_u(b_i)$$

- For each link l_i,

$$\sum_{j=1}^{n} I(d_j = i)(\texttt{use}(c_j)) \leq \texttt{cap}_d(b_i)$$

- For each call c_i, if $u_i = j$ then either $j = 0$ or $\texttt{cover}(b_j, \texttt{source}(c_i))$
- For each call c_i, if $d_i = j$ then either $j = 0$ or $\texttt{cover}(b_j, \texttt{dest}(c_i))$
- For each call c_i, $u_i = 0$ if and only if $d_i = 0$

where I is the indicator function. The optimization function is defined as follows:

$$g(\langle (c_1, b_{j_1}), \ldots, (c_n, b_{j_n}) \rangle) = \sum_{i=0}^{n} p(c_i) I(u_i = 1)$$

The problem is dynamic in that calls arrive and are accepted or rejected; calls are terminated or completed; and the beam capacity changes. In terms of DCOPs, the relaxations that can occur are:

- The two variables corresponding to an existing call are deleted from the problem along with all associated constraints. This occurs if a call is either rejected, terminated or completed.
- The capacity of a link increases.

The restrictions that can occur are:

- Two new variables corresponding to a new call are added to the problem along with all associated constraints.
- The capacity of a link decreases.

4. SOLVING THE DCOP USING LOCAL SEARCH

We are now ready to describe our solution to the satellite telecommunications DCOP. When a relaxation occurs, we do nothing; the solution to the previous problem is always adequate when the problem is relaxed. There are two categories of restrictions in this problem: call arrival and capacity reduction. When a new call arrives, it is assigned to a pair of beams which have the appropriate coverage and have the most remaining capacity. The resulting assignment may overload one or both beams, which also happens when the capacity of a beam is reduced. Consequently, the main issue is moving or terminating calls in such a way that we preserve the high priority calls and no beams are overloaded.

We solve the problem of reassigning calls on overloaded beams using hill-climbing. Recall that a hill-climbing algorithm requires two components: a candidate-generation component to take an initial assignment and generate new assignments, and an evaluation function which ranks the new assignments. The best of the candidate assignments according to the objective function is then selected as the new assignment. We first discuss these two components then show how the algorithm works as a whole.

We generate new candidates by trying to move the lowest priority calls on overloaded beams. Let L be the set of all of the lowest priority calls which could be moved to relieve the capacity of any overloaded beam, and let m be the highest priority call in L. (Should there be several calls of the same priority, assume that each call has a unique identifier and pick the one with the smallest id.) Now let B be the set of beams satisfying the coverage requirements for this call. The candidates that are generated consist of moving the call to each of these beams in turn or rejecting the call. Notice that there is always at least one option because we can always reject a call.

Now let us see how we compare the candidates. Our preference is to keep calls of high priority; for the objective function we interpret each priority as an integer and sum the priorities of calls which are assigned to beams. We do this by counting the highest priority calls on each beam until the capacity is reached. Under this scheme, the priorities of calls must be chosen so that calls of lower priority are appropriately comparable to calls of higher priority; for instance, if 2 calls of priority p are

worth more than 1 call of priority p+1 then $p + 1 < 2p$. Additionally, we do not count a call if either its uplink or downlink is on an overloaded beam and is of low enough priority that it might be terminated.

In some cases the candidates can all have the same rank. If the call being moved has a priority low enough that it would not stay on any beam, we reject the call to remove it from the system. In any other situation, we randomly select one of the best candidates as the new configuration.

We now have an operation which determines what to do with a single call on an overloaded beam. The call is moved to another beam if it is of high enough priority to displace calls of lower priority or if there is excess capacity. Otherwise, the call is terminated. We execute this operation repeatedly until no beams are over capacity any longer. Figure 2 shows the sketch of the full algorithm, which we call *load-balance*.

```
procedure load-balance()
  O = set of overloaded beams
  while O is not empty
    L = set of calls to be bumped from O
    m = highest priority call from M
    B = set of beams m can move to
    for b ∈ B
      if b satisfies coverage requirements of m
        rank moving m to b
    if rank of terminating m == rank of best move
      terminate m
    else make best move
    update O
  end # while
end
```

Figure 2: The load-balance hill-climbing algorithm.

The *load-balance* procedure may be called many times, since moving a call may exceed the capacity of some other beam, and several calls may be required to reduce the capacity on a single, heavily overloaded beam. We now provide a worst-case complexity of the number of times *load-balance* will be called in order to satisfy the capacity constraints on all the beams. Let C be the total number of calls in the system at the time that load balancing occurs. We shall show that no call is ever handled by the procedure more than once. If *load-balance* rejects a call it is never manipulated again, so let us consider what happens if the procedure moves part of a call c_i. Recall that c_i is the highest priority call of all the calls which must be moved from any overloaded beam. Because c_i is moving and not being terminated, we know that either there is space on the destination beam, or some other calls can be moved from the destination beam. But all of these calls are of strictly lower priority than c_i. Therefore, no call moved after c_i can displace c_i from it's new home. Consequently, in the worst case, each component (uplink and downlink) of a call would have to move once. Since each call is either terminated or each component of a call is moved only once, *load-balance* is called fewer than $2C$ times.[1] Also notice that the complexity of any single call manipulation is $O(b)$ since there are b beams and each call manipulation must consider all beams in the worst case. So overall the algorithm requires $O(Cb)$ elementary operations.

We now return to the issue of call acceptance. When a new call arrives, the satellite must decide whether to accept or reject the call. The procedure to do this requires first finding a pair of uplink-downlink beams which satisfy the coverage requirements. We tentatively assign the call to those beams with the highest capacity path, and then call *load-balancing*. Once load balancing is done, if the new call request is part of the assignment, then the call is accepted and all changes required to realize the new solution are taken. If the new request is *not* part of the new solution, which can happen when one of the beams in at or near capacity and the new request is of low priority, the call is denied.

5. EMPIRICAL RESULTS

In this section we present the results of a simulation using call requests from a real telecommunications application. The scenario we experimented with uses a satellite with 2 beams covering 5 ground stations. Three of the stations are covered by both satellites, and the remaining two stations are covered by only 1 satellite each. Calls have 8 possible priorities, with priority 1 calls the highest. As the simulation proceeded, we ratcheted down the capacity of the uplink and downlink capcity of both the beams from 1000 to 400 units of bandwidth over the course of 1000 seconds. This scenario was designed to show that, as the capacity of this beam changes, our hill-climbing algorithm terminates calls in lowest-first priority order.

We encoded the problem as a DCOP as we have indicated in the previous sections of the paper. In addition, we carefully crafted an objective function in order to ensure that calls of the highest priority stayed on the beams as capacity changed.

Figure 3 shows a graph of bandwidth usage by relative priority [2] over time on one of the downlink beams during a test run of 7769 dynamic call requests over a period of 33 minutes (an average of about 3.9 calls per

[1] We can put a tighter bound on the number of manipulations at runtime, but space prohibits us from including these results.

[2] These are not the priority values used to compute the objective function during search.

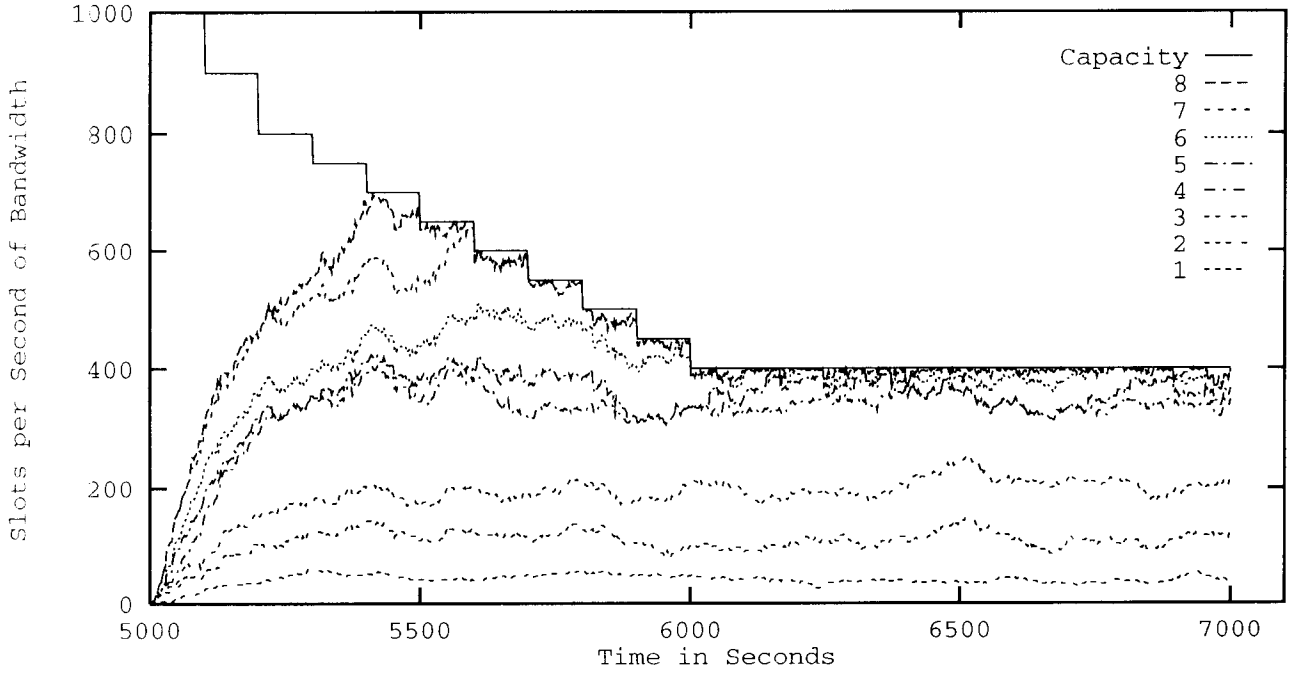

Figure 3: A graph of bandwidth capacity and usage by priority over time on a downlink beam. The highest priority calls (1) are at the bottom, with successive lower priority call in increasing order above (2-8), total available bandwidth capacity at the top.

Time	% Opt. Score	% Opt. Bandwidth
5450	99.98	97.37
5550	99.96	98.29
5650	99.76	95.69
5750	99.88	97.95
5850	99.67	95.99
5950	99.69	96.89
6050	99.27	93.50
6200	99.80	99.50
6500	99.33	99.00
6700	98.14	100.00

Figure 4: Closeness of the solution found by load-balance to the optimal solution as simulation time increases.

second). Note that the highest priority calls are graphed at the bottom, with successive ranks of lower priority calls in increasing order above. The top-most (solid) line shows the total available bandwidth capacity of the beam during the run. As we see from the figure, the calls of low priority are terminated to keep the high priority calls on the system, as desired.

As we have said before, hill-climbing algorithms may not always find the optimal solution to a COP. To test how close we came to the optimal solution, we created the COP induced at a particular time instant of the simulation and found the optimal solution. Figure 4 shows the results. The percentage is the value of the solution found by the load balancer over the optimal value. We see that throughout the simulation the hill-climbing algorithm was able to consistently find solutions within 2% of optimal.

We also analyzed the bandwidth use achieved by the hill-climbing load balancer. Figure 4 also shows the percentage of optimal bandwidth achieved by the load balancer. We see that the load balancing algorithm consistently uses more than 94% of the bandwidth used by the optimal solution.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced the satellite telecommunications, and showed how it can be modeled as a Dynamic Constrained optimization Problem. We then designed and implemented a hill-climbing algorithm to solve the problem. Our empirical results show that hill-climbing is capable of solving this problem very well, as it is consistently able to achieve within 2% of the best possible solution.

There are several future directions for this work. One direction is to consider the impact of moving beams. As beams change position, ground station coverage patterns will change, introducing a new set of relaxations

and restrictions. The framework we have described is adequate to address this type of dynamism, and we believe that the results will be as good as those presented here. Another research question involves increasing the number of beams in the simulated satellites. More beams and more ground stations may result in larger search problems, thereby causing more difficulties as the hill-climbing search must work harder to solve the problem instances that arise over time. A third option is to consider situations with multiple satellites. This problem presents a challenge to modeling via DCOPs, because now a call must take several "hops" to get from it's source ground station to it's destination ground station. In theory, this should be addressable simply by adding some more variables to the problem, but in practice it may prove more difficult.

Another possible research direction is to consider using other algorithms to solve this problem. In the previous section we used a complete search algorithm with hand-crafted heuristics to generate the optimal solutions to the optimization problem induced at fixed timepoints. We found that while it would often take this program a long time to find the optimal solution, it frequently found good solutions early. This raises the prospect of using other algorithmic techniques for constraint satisfaction to address this problem.

REFERENCES

Banerjee, D. & J. Frank (1996). Constraint satisfaction in optical routing for passive wavelength route networks. In *Proceedings of the 2nd International Conference on the Principles and Practices of Constraint Programming*, pages 31–45. Springer Verlag. Lecture Notes in Computer Science.

Dechter, R. & A. Dechter (1988). Belief maintenance in dynamic constraint networks. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 37–42, Palo Alto, CA. Morgan Kaufmann.

Freuder, E. & R. Wallace (1998). Stable solutions for dynamic constraint satisfaction problems. In *Proceedings of the 4th International Conference on the Principles and Practices of Constraint Programming*, pages 447–461. Springer Verlag. Lecture Notes in Computer Science.

Gent, I. P. & T. Walsh (1993). Empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, 1:47–59.

Haralick, R. & G. Elliot (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313.

Minton, S., M. D. Johnston, A. Phillips, & P. Laird (1990). Solving large scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 17–24. AAAI Press.

Nadel, B. (1989). Constraint satisfaction algorithms. *Computational Intelligence*, 5:188–224.

Selman, B., H. Levesque, & D. Mitchell (1992). New method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 440–446. AAAI Press.

Verfaillie, G. & T. Schiex (1994). Solution reuse in dynamic constraint satisfaction problems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 307–312, Cambridge, MA. MIT Press.