Producing Satisficing Solutions to Scheduling Problems: An Iterative Constraint Relaxation Approach

Steve Chien
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099
chien@aig.jpl.nasa.gov

Jonathan Gratch
Beckman Institute
University of Illinois
405 N. Mathews Av., Urbana, 11.61801
gratch@cs.uiuc.edu

Abstract

One drawback to using constraint–propagation in planning and scheduling systems is that when a problem has an unsatisfiable set$ of constraints such algorithms typically only show that no solution exists. While, technically correct, in practical situations, it is desirable in these cases to produce a satisficing solution that satisfies the most important constraints (typically defined in terms of maximizing a utility function). This paper describes an iterative constraint relaxation approach in which the scheduler uses heuristics to progressively relax problem constraints until the problem becomes satisfiable. We present empirical results of applying these techniques to the problem of scheduling spacecraft communications for JPL/NASA antenna resources.

1 INTRODUCTION

Constraint Satisfaction Problems (CSP) area general formalism for representing a wide variety of problems ranging from scheduling, to planning, to design applications (Dechter 1992). Many approaches to solving CSP problems use a technique called constraint propagation in which assignments of values to some variables and constraints are used to infer necessary values of other variables in order to allow the constraint to be satisfied. Unfortunately, in many real–world CSPS, the entire set of constraints specified as the goal cannot be satisfied, In these cases, the goal is usually to satisfy as many constraints as possible, or to maximize some function of the satisfied constraints (e.g., representing the value of the satisfied constraints, etc.).

This paper describes an approach to constructing satisficing solutions to CSP problems that still relies heavily on constraint propagation techniques. In this approach, the constraint-propagation problem-solver (CPPS) serves as the basis for the larger iterative problem solver. In this approach, the basic problem-solver attempts to solve the CSP problem, and if failing, provides key information on sets of inconsistent constraints that represent difficult to satisfy sets of constraints. Using this information, the problem-solver then uses heuristics to(1) pick constraints to relax and (2) determine how to relax these constraints. The problem-solver then is called recursively on the reduced problem. This process continues until enough constraints have been relaxed to allow the problem-solver to solve the partial problem. This flow of control is shown in Figure 1 below.

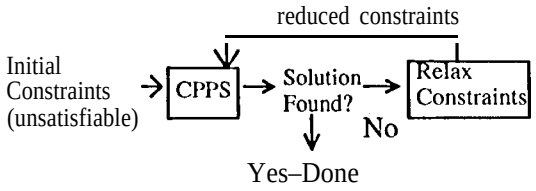

Figure 1: Iterative Constraint Relaxation

This iterative relaxation approach to solving CSP problems has been implemented and tested on a constraint–based scheduling system LR–26R that constructs schedules allocating NASA/JPL antennas for communication to earth-orbitting satellites, While the description of our approach in this paper refers to a specific problem formulation and scheduling application, our general approach applies to solving the general class of constraint satisfaction problems. While considerable effort has been devoted to CSP approaches to scheduling (Sadeh & Fox 1990, Minton et. al. 1992, Smith& Cheng, 1993), our approach differs in that it is a flexible approach designed to allow incorporation of a wide variety of heuristic strategies.

The remainder of this paper is organized as follows. Section 2 describes the specifics of the application to scheduling antenna resources and how these problems are formulated and solved as constraint satisfaction problems. Section 3 describes the preliminary results of testing the resulting scheduler and relaxation heuristics on DSN data. Section 4 discusses open research issues and future work and summarizes the main points of the paper.

2 DSN SCHEDULING AS CSP

The Deep Space Network (DSN) scheduling problem is a complicated real world task that has proved challenging to state-of-the-art scheduling techniques, The problem is to allocate communication requests bet ween earth-orbiting satellites and the three 26-meter antennas at Goldstone, Canberra, and Madrid. These antennas make up part of the DSN that is responsible for communication with earth-orbiting and interplanetary spacecraft. Each satellite has a set of constraints, called project requirements, that define its communication needs. For example, the Nimbus-7 satellite must have at least four 15-minute communication slots per day, and these slots

Portions of this work were performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration and portions at the Beckman Institute, University of Illinois, under National Science Foundation Grant NSF–IRI–92–09394.

cannot be greater than five hours apart. Two factors complicate the problem. First, antennas are a limited resource – two satellites cannot communicate with the same antenna at the same time. Second, satellites can only communicate with certain antennas at certain times, depending on their orbits.

Scheduling is done on a weekly basis. A weekly scheduling problem is defined by three elements:

(1) the set of satellites to be scheduled,

(2) the constraints associated with each satellite, and

(3) a set of time periods specifying all temporal intervals when a satellite can legally communicate with an antenna.

Two time periods conflict if they use the same antenna and overlap in temporal extent, A valid schedule specifies a non-conflicting subset of all possible time periods where each project's requirements are satisfied.

2.1 THE LR-26 SCHEDULER SYSTEM

1.1<-26 is a heuristic approach to the scheduling problem (Bell1992) developed at the Jet Propulsion Laboratory. LR–26 is implemented so that it can be run with a variety of heuristic control strategies because it is believed that different heuristics will perform better in different contexts and this flexibility will allow for better overall performance. Scheduling in LR–26 is formulated as a 0-1 integer programming problem (Taha 1982) – a methodology for finding an assignment to integer variables that maximizes the value of an *objective function*, subject to a set of linear constraints. The objective function characterizes the "value" of the solution. Many constraint satisfaction problems (CSP) are easily cast as integer programming problems (Mackworth 1992). In the DSN domain, time periods are treated as 0-1 integer variables (O if the time period is excluded from the schedule or 1 if it is included), the objective is to maximize the number of time periods in the schedule subject to the project requirements and temporal conflict constraints that are expressed as sets of linear inequalities.

Returning to the Nimbus-7 satellite example, Nimbus-7 must have at least four 15-minute communication slots per day, and these slots cannot be greater than five hours apart. Suppose there are 7 possible 15 minute time slots to schedule Nimbus-7 on a particular day, represented by the variables v_1–v_7. Furthermore, v_1 uses the Goldstone antenna, as dots the overlaps with a possible time slot v_8 that involves the Goldstone antenna communicating with another satellite. The first constraint, that there must be four time slots per day would be represented by the constraint that $v_1 + v_2 + v3 + v4 + v_5 + v_6$ +V72 4. The second constraint, that the antenna can only communicate with a single spacecraft at a time might be represented by the constraint $v_1 + v_8 \leq 1$.

Integer programming is NP-hard, and the size of our scheduling problems makes the conventional approach impractical: a typical problem has approximate] y 650 variables and 1300 constraints. LR-26 embodies a heuristic approach called *lagrangian relaxation* (Fisher 1981). Lagrangian relaxation requires identifying a set of constraints that, if removed, make the problem computationally easy. These constraints are "relaxed," meaning they no longer act as constraints but instead modify the objective function. A relaxed objective function is automatically generated such that satisfying relaxed constraints increases the value of the relaxed solution, Therelaxed problem is by definition easy to solve and often finding the highest value relaxed solution solves the original problem. Furthermore, each relaxed constraint has a weight associated with it when it is added to the objective function. By systematically adjusting these weights and re-solving the relaxed problem, a solution to the unrelaxed problem is often efficiently discovered. Even if the unrelaxed problem cannot be solved in this manner, this weight adjustment cycle can move the scheduler closer to a solution, allowing the unrelaxed solution to be discovered with less search. LR-26 relaxes inter-antenna constraints. This representation facilitates an efficient implicit representation of temporal conflict constraints, which makeup more than half of all constraints in a typical problem.

LR-26 combines lagrangian relaxation with standard constraint satisfaction search techniques, The scheduler performs depth-first search through a space of partial schedules. A variable is assigned a value of *in* (or 1) if the associated time period is included in the partial schedule, *out* (or 0)if it is excluded from the partial schedule. The scheduler constructs a complete schedule by incrementally extending the partial schedule. First it attempts to completely extend the schedule using the lagrangian relaxation method. If the relaxed solution satisfies all constraints it is returned. Otherwise, a set of possible extensions to the partial schedule is created and these are recursive explored. Extensions are created by choosing an unsatisfied constraint, identifying a set of uncommitted variables in the constraint, and assigning possible values to these variables. The set of extensions are placed on a stack to implement the depth-first search. The search continues until a solution is uncovered or a time-bound is reached. Currently the scheduler implements a time-bound oft wo CPU minutes. Any problem not solved within this bound is deemed unsolvable.

Thus LR-26 can be viewed as a recursive application of four control decisions:

(1) decide on a lagrangian weight adjustment scheme, terminating if a viable solution is found; otherwise

(2) choose an unsatisfied constraint,

(3) determine a set of extensions to the partial schedule that satisfy the constraint, and

(4) determine an order to explore these extensions.

2.2 UNSATISFIABLE PROBLEMS

For many of the more congested DSN scheduling problems the set of project constraints may not be satisfiable (e.g., not all of the projects will get their requested antenna time). In these cases, the LR–26 scheduler will not be able to find a valid schedule because none exists. In some of these cases, LR–26 will be able to prove that no valid schedule exists and will provide this information. In other of these cases, LR-26 will simply exceed resource bounds for the scheduling problem without finding any solution. Unfortunately, from an operations standpoint, not finding a solution is not very useful. Ideally under these circumstances the scheduler would provide a schedule that satisfied as many of these constraints as possible. This goal can be made more concrete by assigning

a value to each constraint and the value of a schedule is determined by summing the value of every constraint it satisfies. While this simple model of schedule value or schedule quality is the one wc usc in the implementation and empirical tests described in this paper, we describe current work on more expressive models in the discussion section.

To ground the concept of unsatisfiable constraints, consider the following constraints:

$v_1 + v_2 + v_3 \geq 2$ (A1)	$v_7 + v_2 \leq 1$ (C1)
$v_2 + v_3 + v_4 \geq 2$ (A2)	$v_7 + v_3 \leq 1$ (C2)
$v_3 + V_4 + V_5 \geq 2$ (A3)	$v_8 + v_3 \leq 1$ (C3)
$v_6 + v_7 \geq 1$ (B1)	$v_8 + v_4 \leq 1$ (C4)
$v_7 + v_8 \geq 1$ (B2)	$v_9 + v_4 \leq 1$ (C5)
$v_8 + v_9 \geq 1$ (B3)	$v_9 + v_5 \leq 1$ (C6)

This example will bc used later to illustrate dead-end analysis and constraint relaxation in LR–26R.

In this case, constraints A 1–3 indicate that two out of every three consecutive views v_1–v_5 must be in the schedule, a case that frequently occurs when a satellite has a constraint that it must have N views in every time period of duration M. Constraints B 1 –3 indicate a similar constraint that for views v_6–v_8. Constraints Cl -6 indicate that V_7 conflicts with V_2 and v_3, v_8 conflicts with views v_3 and v_4, and that v9 conflicts with V_4 and V_5. This set of constraints is unsatisfiable because constraint B2 indicates that v_7 or v_8 must be in the schedule. If v_7 is in the schedule, Al is unsatisfiable, since both v_2 and v_3 conflict with V_7, and Al indicates that 2 out of VI, V_2, and v3 must be in the schedule. If v_8 is in the schedule, v3 and V_4 must not be in the schedule, and A2 requires that two of v_2, v_3, and v4 must be in the schedule.

2.3 CONSTRAINT RELAXATION

LR–26R extends LR-26 in that it can relax schedule constraints, LR–26R begins by attempting to find a solution to the complete version of a scheduling problem (e.g., with all constraints) by calling LR–26. If LR–26 cannot find a solution (e.g. either exceeds the resource bound or shows that the constraints are unsatisfiable), LR-26R selects a set of constraints to relax. This relaxed scheduling problem is then given to LR-26 for attempted solution. This process proceeds until either LR–26 is able to solve a relaxed problem or a bound on the number of relaxable constraints is reached.

Two types of constraints in LR–26 are considered for relaxation: set constraints and general constraints. A set constraint specifics that at least one of a set of variables should have the value IN. Set constraints are relaxed by elimination from the schedule. B 1–3 from the example above are set constraints, General constraints specify that a weighted sum of a set of variables must be greater than a fixed threshold (e.g., $10v_1 + 8v_{12} + 2v_{13}\, 5v_{32} + 12v_{64} > 11$). General constraints are relaxed by decrementing the right side constant. Relaxing a general constraint multiple times is valid until the right hand side becomes zero or negative and the constraint is eliminated. The constraints A 1–3 above are general constraints.

Because the project constraints represent important operational goals for the projects, finding schedules that satisfy as many of the constraints as possible is an important problem. The relaxation module implements a number of heuristics for selecting heuristics to relax. Determining effective heuristic strategies for specific domains and problem distributions is therefore of great importance.

1,1<–26 reaches a dead.+md in the scheduling process when it is able to show a set of constraints cannot be satisfied. This occurs when a number of search decisions (e.g. forcing variables IN or OUT) and constraint propagation detects an unsatisfiable set of constraints. If a set of dead-ends is spanning (e.g., every variable assignment is an extension of a dead-end) then LR–26 has proven that no solution exists for the set of constraints. In other cases LR-26 will find a number of dead–ends and exceed resource bounds. In these cases, there maybe a solution to the current set of constraints but it may require an unacceptable amount of time to find it.

Analysis of the dead–ends found by LR–26 can be helpful in determining which constraints are good candidates for relaxation to allow for (partial) solution of a scheduling problem. Thus, when solving a problem, LR–26 records information on dead-ends found in terms of the participating contexts (variables forced IN or OUT) and constraints (those constraints that, in combination, are unsatisfiable). This information can then be used to direct the constraint relaxation process (this information can be tracked using Reason Maintenance Systems (McDermott 1991)).

Returning to the example constraints described above, the scheduler might first search by attacking constraint 132. This would lead to determining two extensions to the schedule, one committing to having V_7 IN in the schedule, and then commit to having V_7 OUT of the schedule, leading to the following lines of constraint propagation.

V_7 IN the schedule
V_7 IN implies that V_2 and v3 are OUT by constraints Cl and C2
v_2 and v3 both OUT implies that constraint Al is unsatisfiable
V_7 OUT of the schedule
v_7 OUT implies that v_8 must be IN by constraint B2
v_8 IN implies that v_3 and v_4 both OUT by C3 and C4
v_3 and v_4 both OUT implies that constraint A2 is unsatisfiable

Leading to the following dead-end contexts:

1) the context {(v_7 IN) } implies that the set of constraints {Cl, C2, Al } is inconsistent, and

2) the context { (V_7 OUT) } implies that the set of constraints {B2, C3, C4, A2) is inconsistent.

These contexts have the general form that in context of variable assignments V (a conjunction of individual variable assignments), the conjunction of a set of constraints C is shown to be unsatisfiable. In the case(1) above, the context of V_7 being assigned the value IN, means that the conjunction of the constraints {Cl, C2, Al } cannot be satisfied. In this example, because the contexts (1) and (2) above cover all possible variable assignments (e.g., in any complete schedule, V_7 must be either IN or OUT), the constraint propagation algorithm has shown that the set of constraints {Al, A2, B2, Cl, C2, C3) is unsatisfiable without any assumptions on variable assignments. This dead-end information can be extremely useful in guiding heuristics for determining which constraints to relax.

For example, onc strategy might be to pick a constraint from the set {Cl, C2, Al } to relax and attempt to find a solution where V_7 is IN the schedule.

1.1<–261< can relax constraints by either focussing on dead-ends or individual constraints. In the case of focussing on dead-ends, LR–26R relaxes constraints as follows:

(1) select a set of dead-ends to focus upon using measures evaluating the constraints and contexts involved in the dead-ends; next

(2) determine how many constraints to relax in the dead-end(s) selected; then

(3) for each constraint to be relaxed, determine how it should be relaxed.

In the case of focussing upon individual constraints, LR–26R relaxes constraints as follows:

(1) order the constraints appearing in dead-ends based upon the number of dead-rids in which they appear, the contexts in which they appear, the inconsistent sets in which they appear, and other properties of the constraint graph and contexts involved in the dead-ends; next

(2) determine how many constraints to relax; then

(3) for each constraint to be relaxed, determine how it should be relaxed.

At each of these decision points, LR–26R uses a heuristic method to reach a decision. Once the appropriate constraints have been relaxed, LR-26R calls LR–26 with the relaxed problem continuing to relax additional constraints until either LR-26 can solve one of the relaxed problems or a bound on the number of constraint relaxations is exceeded.

2.4 FLEXIBLE PROBLEM-SOLVING AND EXPECTED UTILITY

General problem solving tasks such as CSP problem-solving are inherentl y complex. Nevertheless, in many practical situations these complex problems have reasonable solutions (e.g. traveling salesman problem (Held 1970)). Often heuristic approaches can take advantage of the structure of a domain or the distribution of problems to formulate effective solutions to complex problems. A flexible problem-solving architecture such as LR–26R can allow for a wide variety of heuristic approaches to be applied, resulting in a better solution to the practical problem at hand,

If one defines a utility function U which takes a control strategy and a problem and returns a real–valued number, one can now compare alternative problem-solving strategies to determine the best for a given distribution of problems. The goal of adaptive problem-solving can be expressed as: given a problem distribution *D,* find ***STRATSO*** as to maximize the ***expected utility*** of *PE* where expected utility is defined as:

$$\sum_{d \in D} U(PE(STRAT), d) \times \textbf{probability(d).}$$

1.R-26 and LR-26R have several control points relevant to searching for a viable schedule and relaxing constraints. Each of these control points is a point where a heuristic strategy (or composition of heuristic strategies) can be inserted to alter (and hopefully improve) the performance of the scheduler. These control points correspond to the choice points described in Section 2.1 control decisions (1)–(4) and Section 2.3 control decisions (1) – (3) for dead–end focussing and (1) – (3) for individual constraint focussing.

Measures of utility for solutions produced by LR–26 might be influenced by time to produce a solution, or measures of the quality of the plan, such as number of constraints satisfied in a certain way, corresponding to the cost of implementing the solution, robustness, simplicity, etc. Measures of utility for solutions produced by LR–26R might be functions of the constraints that were relaxed in finding a solution as well as the amount of time required to find a solution.

3 EXPERIMENT ANI) RESULTS

This section describes the results of applying our general approach to CSP problem--solving and scheduling to the specific problem of scheduling DSN antenna resources. Wc first describe the problem distribution used to perform the evaluation, and the several versions of the problem involved. Next, the search heuristics for LR–26 and relaxation heuristics for LR-26R are described. Then we describe the results of applying LR–26 and LR-26R to the different versions of the DSN scheduling problem.

3.1 PROBLEM I)ISTRIBUTION

Ideally, we would use the identical problem distribution faced by the human experts in this domain. Unfortunately, not all of this information is in electronic form and thus is difficult to present to the LR-26 scheduler. There does, however, exist a large electronic database of information for many of the projects in the deep space network, We used this database to construct a large body of scheduling problems that are representative of, if not identical to, the type of problems faced by the human schedulers. Problems are generated by randomly choosing combinations of projects from the available data with requirements and time periods representing current and future projected requirements and time periods of actual projects. Eight projects available in this format means a large number of combinations of project data, and thus no shortage of test problems. The primary difference between these and actual problems lies in the particular combinations of projects that appear in the schedule.

Given this distribution of scheduling problems, we defined three versions of the scheduling problem, DSN–S, DSN–US, DSN–R. The DSN–S problem set consists of all of the problems that LR–26 can solve with some heuristic strategy (solving a problem indicates either finding a solution orproving that no solution exists). The DSN–US is a superset of DSN–S and is all the problems in the DSN distribution without allowing constraint relaxation (e.g., DSN–US distribution includes problems for which all implemented heuristic strategies exceeded resource limits without finding a solution or showing that no solution existed). DSN–R consists of the same problems as DSN–US, but the scheduler is allowed to relax constraints in order to solve the problems and the goal is to solve the problems relaxing the fewest constraints (as determined by some metric as described below),

3.2 EXPECTED UTILITY

Depending upon the problem distribution being evaluated (e.g., DSN–S, DSN–US, or DSN–R), differing utility functions are possible. For the DSN–S distribution, possible utility functions arc the amount of time required to generate a schedule (or to prove that no schedule exists), and possible measures of schedule quality (such as number of views included in the schedule, robustness of the schedule, cost to implement the schedule, etc.), Additionally, utility metrics that account for the proportion of problems solvable by a strategy are very useful. For the DSN–US distribution, DSN–S measures are valid – those which account for the proportion of problems solvable become even more important.

In the DSN application satisfying as many project constraints as possible is key. This behavioral preference can be expressed by a utility function negatively related to the number of constraints relaxed before solving the problem. As the scheduler produces schedules that satisfy more constraints, the utility of the problem solver on that problem increases. We characterize this preference by making utility the negative of the number of constraints relaxed by the scheduler before solving a problem. If a general constraint was relaxed several times before LR-26 was able to solve the problem, the value of the schedule reflects each of these separate relaxations.

3.3 HEURISTICS FOR LR-26 ANI) LR-26R

We have implemented a number of search heuristics for LR–26. These search heuristics provide possible strategies to insert into the schedule construction control points described in Section 2.4: lagrangian weight search, constraint selection, extension generation, and child ordering. Heuristic strategies for the Lagrangian Weight Settings include: full weight search, search only at the top–level search node, no weight search, or subgradient optimization. Heuristic strategies for constraint selection involve analyzing the topology of the constraint graph to capture strategies such as: prefer constraints with views that satisfy many constraints, penalize constraints with views that conflict with many other views, prefer constraints with few views, prefer constraints that are almost satisfied. There are currently two heuristic strategies for extending schedules. The original method developed by Colin Bell involves creating a child for each view in the constraint, with that view forced IN. The systematic method selects a single view in the constraint and creates two children, one with the view forced IN, one with the view forced OUT. The search child ordering methods analyze the topology of the constraint graph in a manner similar to the constraint sort methods to implement such strategies as: prefer those children that force IN views which assist in satisfying many constraints, penalize those children that force IN views which conflict with many other views, prefer those children that force IN views which conflict with many other views.

We have implemented a number of heuristic strategies for selecting constraints to relax in LR–26R. These heuristics can bc viewed as applying to the three control points described in Section 2.4: 1) focus type – focussing on dead-ends or constraints, 2) focus–method – determining which constraints to prefer relaxing, 3) stcI>–method – determining how many constraint to relax, and 4) relaxation–method – how to relax each constraint selected.

The focus type determines whether the system focusses on dead-end records or individual constraints. The focussing method determines which constraints or dead-ends are best suited for relaxation and produces an ordering on constraints or dead–ends in the scheduling problem. Examples of heuristic methods for this control point are depth first (focus on those constraints participating in the most recent dead-end found), smallest clique (focus on those constraints that participate in dead-ends involving the fewest constraints), and most–commonly–occurring (focus on those constraints that appear in the most dead-ends). We are currently experimenting with a number of other heuristics that involve factors such as project priorities and congestion of the constraint graph.

The step method determines how the ordering generated by the focussing method is used to relax constraints. Currently we are experimenting with two general step methods: breakthrough and k–beam. Breakthrough uses the focussing method to order dead-ends and then relaxes every constraint in the dead-end, K–beam relaxes the k most preferred constraints.

The relaxation method determines how each constraint selected for relaxation in the previous phases will actually be relaxed. Methods for determining how to relax these constraints include averaging the coefficients on the left hand side of the constraint and decrementing the right hand side constant by this average, decrementing by the minimum or maximum value of the left hand side, or by looking at a measure of how many views must be forced in to satisfy the constraint.

3.4 EMPIRICAL RESULTS

In order to assess the effectiveness of our general approach to solving CSP problems as applied to scheduling we have evaluated LR-26 and LR-26R extensively on all three of theDSN problem distributions described in Section 3.1. The remainder of this section describes the results of this evaluation and draws some conclusions on the promise of adaptive problem-solving for this problem domain.

In order to test if there is a significant variation in expected utility for different heuristic strategies for LR–26 we developed 52 additional heuristic search strategies to augment the single strategy derived by the human expert (Bell 1993). Figure 2 shows in histogram format the number of strategies with each average CPU problem-solving time on each of the problem distributions DSN–S and DSN–US (recall that for this problem set the utility = – CPU seconds) as estimated over 1000 problems. The single human expert derived strategy had an average performance of 165 CPU seconds on DSN–US and 57 CPU seconds on DSN–S. Other related work (Gratch et al. 93a) included using machine learning techniques to automatically search this space of strategies to find good heuristic strategies, the average utility of machine derived strategies was 147 CPU seconds for DSN–US and 27 CPU seconds for DSN–S.

The second set of tests is to verify that there is a significant variation in the expected utility using the different relaxation heuristics with LR–26R. Figure 2 shows the number of relax-

ation strategies with their average number of constraints relaxed (recall that the utility for this problem distribution is – the number of relaxations required) as estimated over 100 problems. '1'here is no human expert derived strategy for this problem. Again, in related work (Gratch et al. 93b) machine learning techniques were used to find good strategies for relaxing constraints – the average number of constraints relaxed for the average machine derived strategies for this distribution is 44, whereas the average for all defined strategies is 137.

4 DISCUSSION AND CONCLUSIONS

Wc now briefly describe a number of areas of current work. The current measure for schedule quality (negative the number of constraints relaxed), is syntactic at the constraint level. Unfortunately, not satisfying different constraints that appear identical at the constraint level can have widely differing effects in the real world. For example, the constraint that the maximum time bet ween communications with a particular satellite should be no greater than a time amount T (such as every possible 24 hour period) is represented constraint–wise by taking all 24 hour periods, for each such period, constructing a set constraint stating that at least one view representing an allocation of an antenna to that spacecraft must be in the schedule. Thus, if views v_3 v_4 v_5 and v_6 were the only views in a 24–hour period communicating with the satellite, there would be the set constraint Cl $v_3 + v_4 + v_5 + v_6 \geq 1$. Another set constraint C2 for another 24–hour period might look like $v9 + v_{10} + v_{11} + v_{12} \geq 1$. However, depending upon the other views and constraints in the problem, relaxing constraint Cl might allow a schedule with a gap of 28 hours between communications with the satellite, while relaxing constraint C2 might allow agapof 38 hours between communications with the satellite. This distinction can only be captured by mapping the schedule back into the real–time space. Another important area for work is enabling LR-26R to use more sophisticated analysis techniques in selecting constraints for relaxation such as bottleneck analysis.

Wc have described an iterative constraint–relaxation approach to finding satisficing solutions to problems with unsatisfiable sets of constraints. In our approach the scheduler uses heuristics to progressively relax problem constraints until the problem becomes satisfiable in an effort to find a solution that satisfies the most important constraints (in terms of maximizing a utility function). We presented empirical results of applying these techniques to the problem of scheduling spacecraft communications for JPL/NASA antenna resources.

References

C. E. Bell, 1993 "Scheduling Deep Space Network Data Transmissions: A Lagrangian Relaxation Approach," **Proc.** of **the SPIE Conf.** on Applications of AI 1993: **Knowledge–based** Systems in **Aerospace** and Industry, Orlando, FL.

R. Dechter, 1992, "Constraint Networks," in ***Encyclopedia of Artificial Intelligence***, Stuart C Shapiro (cd.), Wiley.

M. Fisher, 1981, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," ***Management Science*** 27, 1.

J. Gratch and S. Chien, 1993, "Learning Search Control Knowledge for the Deep Space Network Scheduling Problem: Extended Report and Guide to Software," Tech. Rep. UIUCDCS–R–93–1789, Dept. of Comp. Sci., Univ. of Illinois, Urbana, 11,.

J. Gratch, S. Chien, and G. F. DcJong, 1993a "Learning Search Control Knowledge for Deep Space Network Scheduling," **Proc. Int. Conf. on** Machine Learning, Amherst, MA.

J. Gratch, S. Chien, and G. F. DcJong, 1993b "Learning Search Control Knowledge to improve Schedule Quality," **Proceedings** of the **IJCAI93** Workshop on Production Planning Scheduling and Control, Chamberry, France.

M. Held and R. M. Karp, 1970, " "The Traveling Salesman Problem and Minimum Spanning Trees"," ***Operations Research 18, 1138-1162.***

A . Mackworth, 1992, "Constraint Satisfaction," in ***Encyclopedia of Art. Int.***, Stuart C Shapiro (ed.), Wiley.

D. McDermott, 1991, "A general framework for reason maintenance," Artificial **Intelligence** SO, 289–329.

S. Minton, 1988, ***Learning Search Control Knowledge: An Explanation-Based Approach,*** Kluwer, Norwell, MA,

S. Minton, M. Johnston, A. Philips, & P. Laird, 1992, "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems," Artificial **Intelligence** 58, 161-205.

N. Sadeh & M. Fox, 1990, "Variable and Value Ordering Heuristics for Activity-based Job–shop Scheduling," **Proc.** 4th **Int. Conf. on Expert** Systems in Prod. and Operations Management, Hilton Head, SC,

S. Smith&C. Cheng, 1993, "Slack-based Heuristics for Constraint–satisfaction Scheduling," **Proc. AAAI93**, Wash., ***D C.***

H. A. Taha, 1982, ***Operations Research: An Introduc-don,*** Macmillan Publishing Co., Inc., 1982.

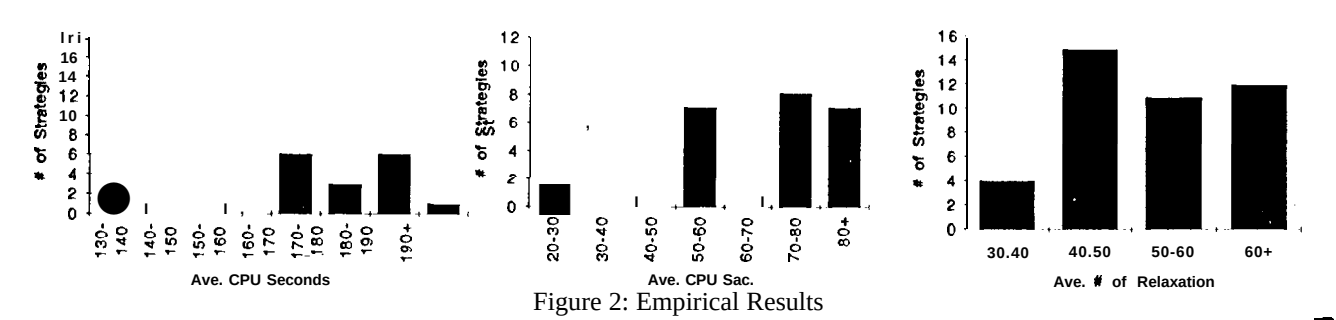

Figure 2: Empirical Results