Enabling Efficient Conflict Resolution in Multiple Spacecraft Missions via DCSP

Hyuckchul Jung[1], Milind Tambe[1], Anthony Barrett[2], and Bradley Clement[2]

[1]Dept. of Computer Science, University of Southern California
[2]Jet Propulsion Laboratory, California Institute of Technology
{jungh, tambe}@usc.edu, {barrett, bclement}@aig.jpl.nasa.gov

Abstract

While NASA is increasingly interested in multi-platform space missions, controlling such platforms via timed command sequences -- the current favored operations technique -- is unfortunately very difficult, and a key source of this complexity involves resolving conflicts to coordinate multiple spacecraft plans. We propose distributed constraint satisfaction (DCSP) techniques for automated coordination and conflict resolution of such multi-spacecraft plans. We introduce novel value ordering heuristics in DCSP to significantly improve the rate of conflict resolution convergence to meet the efficiency needs of multi-spacecraft missions. In addition, we introduce distributed POMDP (partially observable markov decision process) based techniques for DCSP convergence analysis, which facilitates automated selection of the most appropriate DCSP strategy for a given situation, and points the way to a new generation of analytical tools for analysis of DCSP and multi-agent systems in general.

Introduction

The past few years have seen missions with growing numbers of probes. Pathfinder has its rover (Sojourner), Cassini has its Huygens lander, and Cluster II has 4 spacecraft for multi-point magnetosphere plasma measurements. This trend is expected to continue to progressively larger fleets. For example, one proposed interferometer mission (Mettler and Milman 1996) would have 18 spacecraft flying in formation in order to detect earth-sized planets orbiting other stars. Another proposed mission involves 44 to 104 spacecraft to measure global phenomena within the Earth's magnetosphere. To date numerous multiple platform missions have been proposed, and they can be grouped into 3 families depending on why multiple platforms were proposed:

- Multi-point sensing for improved coverage when observing/exploring large areas (like the satellites with passive microwave radiometers for the Global Precipitation Mission and similar sensors on the Global Electrodynamics Mission, Leonardo-BRDF, and the Magnetospheric Constellation);
- Building large synthetic aperture sensors with many small spatially separated sensors for imaging very remote targets (Constellation-X, Terrestrial Planet Finder, and TechSat-21);
- Specialized probes with explicitly separate science objectives (like coincident Mars Program missions or the AM and PM trains within the Earth Observing System).

Using current operations techniques, controlling each will involve either sending a timed sequence to a master spacecraft with continuous communications links to its multiple slave spacecraft or sending timed sequences to each spacecraft. In both cases, the use of sequence limits a mission's ability to measure phenomena that are hard to predict. Such phenomena include violent weather within the atmosphere, rapid changes within the magnetosphere, and unexpected anomalies within the spacecraft themselves. Sequence based operations can measure these phenomena, but the probability of catching the desired measurements while performing a sequence rapidly drops as a phenomenon becomes less predictable. While combining onboard data analysis with onboard planning and scheduling enables measuring unpredictable phenomena by a single spacecraft, multiple spacecraft mission have an extra complexity that involves coordinating their evolving plans.

This paper focuses on DCSP (Distributed Constraint Satisfaction Problem) techniques for automated coordination and conflict resolution among spacecraft. While there are efficient DCSP algorithms available, real-time and dynamism in multi-spacecraft domains demands very fast conflict resolution convergence. We introduce several novel value ordering strategies that significantly improve the rate of DCSP convergence. However, empirical results show that selecting the right strategy is critical: very different strategies dominate in different domains.

To analyze and predict strategy performance, a distributed POMDP (Partially Observable Markov Decision Process) based model is introduced. A DCSP strategy is mapped onto a policy in the model. Thus, we evaluate strategies by evaluating policies in this model. Initial result shows that this approach can enable agents to select the right strategy to apply in a given situation. More generally, this result indicates a promising direction for

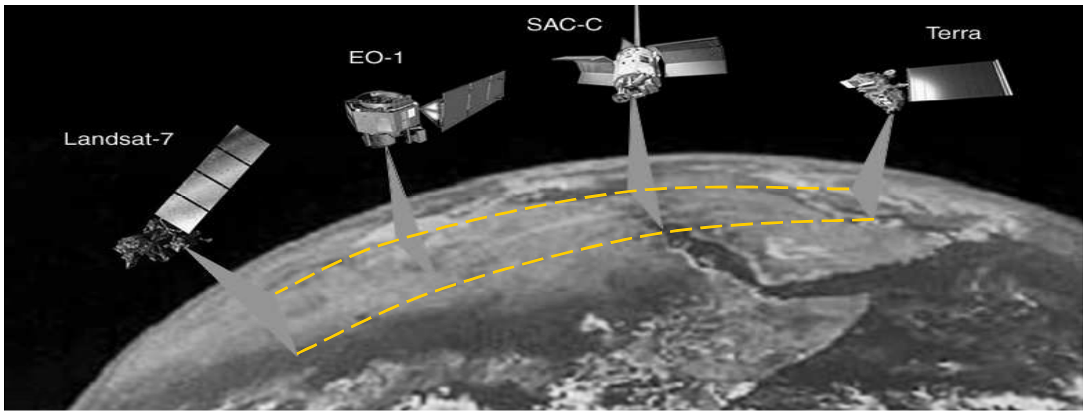


Figure 1: The four co-orbiting spacecraft of the "AM-train" of the Earth Observing System.

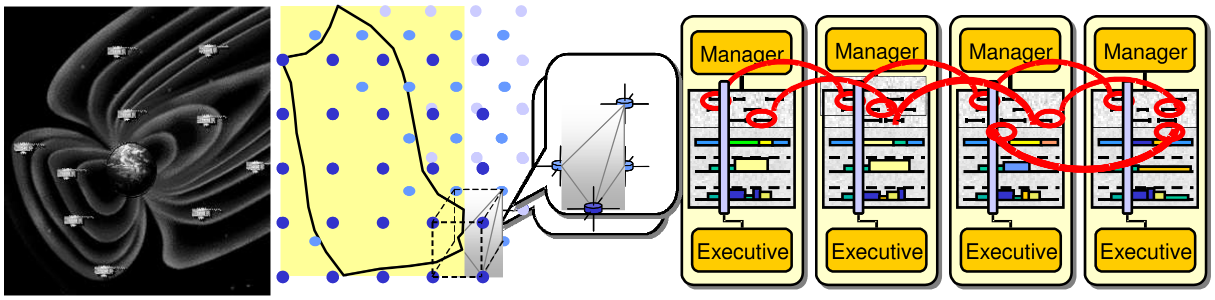


Figure 2: A multi-spacecraft mission as a sensor network with cooperating planners

performance analysis in DCSP, and potentially other multi-agent systems.

Next section describes the multi-spacecraft missions followed by a coordination and conflict resolution algorithm for such missions. After that, we formalize new value ordering strategies in DCSP, and introduce a formal model to analyze the performance of the DCSP strategies.

Managing Evolving Sensor Networks

The whole purpose of a typical NASA spacecraft is to measure phenomena for answering basic scientific questions. For instance, the AM train (Figure 1) within the earth observing system consists of four spacecraft in a related orbit to observe the earth with heterogeneous collections of sensors. Each of these satellites was launched for different reasons, but combining sensors with coincident observations makes the collection of spacecraft more capable than the sum of the parts. In this case the formation was not initially planned, but evolved as mission designers discovered ways to take advantage of existing satellites when developing new ones.

On the other hand, missions to study the magnetosphere fly a collection of spacecraft for identical reasons. These missions focus on making multi-point measurements in order to observe evolving three-dimensional structures within the Earth's magnetosphere. As an example of this class of mission, consider the proposed Magnetospheric Multi-Scale flight project. Here five identical spacecraft fly in prism formation around an evolving elliptical reference orbit. These spacecraft can make isolated observations, but by combining coincident observations, the satellites can measure properties of the enclosed space. For example, measurements of local magnetic field vectors at the four points defining a tetrahedron determine the average electric current through the tetrahedron.

Regardless of whether the multi-platform mission evolved or was explicitly designed, it can be modeled as a sensor network where the communications channels and other local properties change due to orbital motion. In the case of the Magnetospheric Multi-Scale mission, the spacecraft can communicate with Earth around perigee by virtue of being closer to a ground station. Unfortunately orbital effects spread the spacecraft as they get closer to the Earth, which implies that communication between spacecraft is only available around apogee.

To develop an abstract problem that exhibits these issues, consider a three dimensional grid of nodes flying through the Magnetosphere (Figure 2). In this problem spacecraft can use their sensors for isolated measurements or they can take a coordinated measurement to observe a property of space between the spacecraft. In this case, the boundary between two space plasma regions passes through the grid, and a cluster of four spacecraft jointly measures the average current along a patch of the boundary as the boundary moves through a tetrahedron. As

```
Given:  a PLAN with multiple local and shared activities
        a PROJECTION of PLAN into the future
        a set of CONSTRAINTS on shared activities

    1. Revise PROJECTION using the currently perceived state, new goal activities from the mission
       manager, and received changes to shared activities.
    2. Heuristically choose a plan flaw found in PROJECTION.
    3. Heuristically choose a flaw repair method that honors CONSTRAINTS.
    4. Use method to alter CONSTRAINTS, PLAN, and PROJECTION.
    5. Communicate changes to shared activities in PLAN and CONSTRAINTS.
    6. Release relevant near-term activities in PLAN to the executive.
    7. Go to 1.
```

Figure 3: Distributed continual iterative repair algorithm

illustrated on the right, we characterize the autonomy software in each spacecraft in terms of three modules: a manager to oversee the spacecraft by inserting new science and maintenance goals; a planner/scheduler to turn goals into time-tagged activities; and a robust executive to perform these activities.

Distributed Continual Iterative Repair

Extending the CASPER continual planning framework (Chien et al. 2000, Barrett 2000) to perform continual iterative repair within Figure 2's the planner/scheduler modules results in Figure 3's algorithm where the mission manager can insert new goal activities into PLAN. Line 1 makes the PROJECTION variable (containing state variable profiles) always reflect how the spacecraft's state should evolve as its plan executes, and the sixth line causes this execution by passing near-term activities to the executive/diagnostician.

The expected state evolution changes as a plan gets new goal activities and the perceived state diverges from expectations. This divergence is caused by unexpected exogenous events and activities having unexpected outcomes. Since a planning model can only approximate the reality experienced during execution, these unexpected state changes can always happen. Thus revising PROJECTION can result in detecting flaws in a local plan at any moment, and lines 2 through 4 select and apply repair methods to fix these flaws. For instance, a satellite observation can take an unexpectedly long time to complete. Depending on the delay, a later observation may be impossible due to the target being too far behind the satellite when the observation starts. A repair method might fix the flaw by rescheduling the observation at a later time.

While changes to local activities can remain private to a single spacecraft, changes to shared activities must be made public, and line 5 communicates these changes. In general, the use of CONSTRAINTS and flaw repair methods can implement a number of coordination strategies. To be more concrete, suppose that a repair method suggests changing the start time of a shared activity. After changing the activity's *startTime* variable, the spacecraft has to inform other participating spacecraft of the change. In this way the spacecraft can follow an asynchronous weak commitment (AWC) (Yokoo and Hirayama 1998) search approach to maintaining plan coordination.

More precisely, the AWC approach, which is central to our architecture for managing shared activities, involves agents asynchronously assigning values to their variables from domains of possible values, and communicating the values to neighboring agents with shared constraints. Each variable has a non-negative integer priority that changes dynamically during search. A variable is consistent if its value does not violate any constraints with higher priority variables. A solution is a value assignment in which every variable is consistent.

To simplify describing the algorithm, suppose that each agent has exactly one variable and constraints between variables are binary. When an agent's variable value is not consistent with neighboring variable values, there can be two cases: (i) *good* case where there exists a consistent value in the variable's domain; (ii) *nogood* case that lacks a consistent value. In the *good* case with one or more value choices available, an agent selects a value that minimizes conflicts with lower priority agents – the *min-conflict* heuristic. On the other hand, in the *nogood* case, the priority of the agent is increased to *max+1,* where *max* is the highest priority of neighboring agents. In this approach, a bad value selection by a higher agent make does not force lower agents to exhaustively search for local solutions; nogood situations locally increase a priority to make previously higher agents choose new values. Furthermore, an increasing agent sends a *nogood message* with its *agentView* (the values and priorities of neighboring agents). These messages are used to avoid repeating past situations where an agent has no consistent values in its domain. Extension of AWC to multiple variables per agent has been investigated in (Yokoo and Hirayama 1998).

```
Given:  local variable $X_i$ (with domain $D_i$), whose current value violates either local constraints ($LC_i$) or
        external constraints ($C_{ij}$) with higher priority neighbors $(N_i^{high})$.

Repair method **Alter_variable_in_shared_activity**  // for a strategy $S_\alpha$-$S_\beta$
1. Find a value set $D_{co} \subseteq D_i$ whose values remove detected violations with $X_i$;
2. If $D_{co} \neq \varnothing$, signifying that the violations can be resolved locally (good case),
    a. Let $X_i$ = new_cooperative_value($\alpha$, $D_{co}$);
3. Else the violations cannot be resolved locally (nogood case)
    a. Record and communicate the conditions for the impasse in CONSTRAINTS;
    b. Let $X_i$'s priority = max of neighbors' priorities + 1 in CONSTRAINTS;
    c. Let $X_i$ = new_cooperative_value($\beta$, $D_i$)

Procedure new_cooperative_value (strategy $\sigma$, domain $\Delta \subseteq D_i$): returns $X_i$'s new value $v_{new}$
1. If $\sigma \equiv$ basic,
    a. Select $v_{new} \in \Delta$, minimizing number of violated constraints with lower priority agents;
2. Else ($\sigma \in$ {high, low, all})
    a. For each value $v \in \Delta$, $v$'s flexibility = $f_{co}^{\oplus}(v, N_i^{\sigma})$, where $N_i^{all}$ denotes all neighbors $N_i$;
    b. Find $v_{new} \in \Delta$ with max flexibility – breaking ties with min-conflict heuristic;
3. Return $v_{new}$;
```

Figure 4: Flaw repair using AWC framework extended by cooperative strategy and legal value communication

Conflict Resolution Strategies Based on Value Selection Heuristics

While AWC is one of the most efficient conflict resolution protocols, real-time and dynamism in multi-spacecraft domains motivate a need for faster conflict resolution convergence. This section introduces novel value ordering heuristics for faster convergence. In particular, AWC relies on the min-conflict value ordering heuristic for value ordering: given a variable assignment with conflicts, assign a value that minimizes the number of conflicts with other variable assignments (Minton et al. 1990). This heuristic is used as a baseline conflict resolution strategy that we refer to as S_{basic}. Here, S_{basic} strategy selects values only based on the values of the other agents' external variables.

Legal value communication lets agents consider the restrictions that neighboring agents have on their variables' domains. By considering neighboring agents' local constraint induced legal values, an agent can generate a more locally cooperative response, i.e., select a value that gives more choices to neighbors, potentially leading to faster conflict resolution convergence. For instance, agent A_i might have a variable X_i with domain D_i. After applying local constraints, A_i discovers that X_i's revised domain is $D_i' \subseteq D_i$. After communicating D_i', neighboring agents can improve their cooperation with A_i. To elaborate on this, we first define our notion of local cooperativeness.

Definition 1: For agent A_i with value v from domain D_i assigned to variable X_i and a subset of neighboring agents $N_i^{sub} \subseteq N_i$, the *flexibility function* $f_{co}^{\oplus}(v, N_i^{sub})$ is $\bigoplus_{A_j \in N_i^{sub}} [c(v, A_j)]$, where (i) $c(v,A_j)$ is the number of values of X_j that are consistent with v; and (ii) $\oplus$ is the flexibility base, which can be *sum, min, max, product*, or *weighted sum*.

Definition 2: For a value v of X_i, *local cooperativeness* of v is defined as $f_{co}^{\oplus}(v, N_i)$: the *local cooperativeness* of v measures how much flexibility is given to all A_i's neighbors by v.

Definition 3: The *most locally cooperative value* of X_i is defined as v_{max} such that, for any other value $v_{other} \in D_i$, $f_{co}^{\oplus}(v_{\max}, N_i) \geq f_{co}^{\oplus}(v_{other}, N_i)$.

For example, consider a spacecraft data downlink that is constrained in relation to downlinks for two neighboring spacecraft A_1 and A_2, where a start time v leaves 7 consistent start times to A_1 and 4 to A_2 while another start time v' leaves 5 consistent times for A_1 and 5 to A_2. Now, assuming that values are ranked based on flexibility, a cooperative agent that uses the flexibility base *sum* will prefer v to v': $f_{co}^{\oplus}(v, \{A_1, A_2\})$=11 and $f_{co}^{\oplus}(v', \{A_1, A_2\})$=10. If $\oplus$ is *min* however, a cooperative

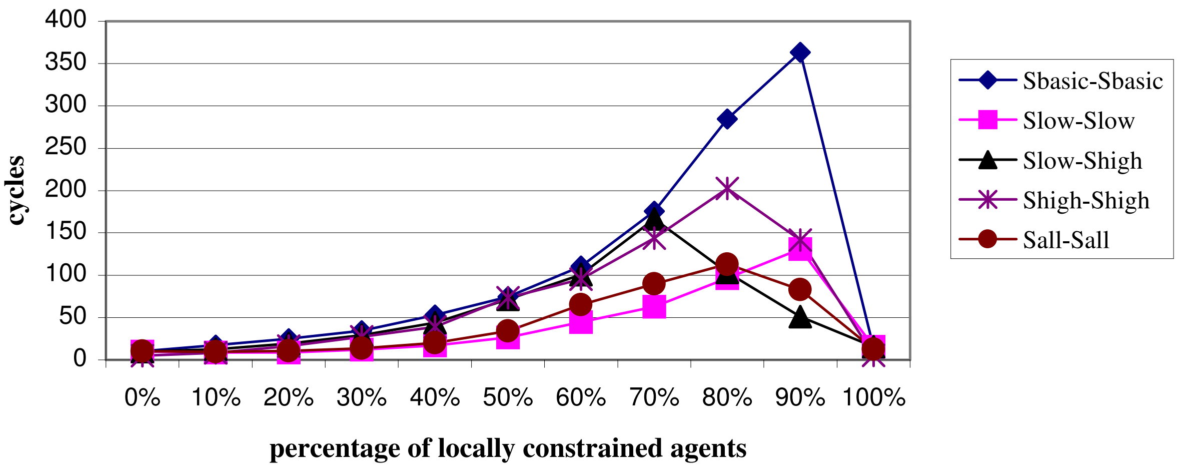


Figure 5: Empirical comparison of strategy performance

agent will rank v' higher than v: $f_{co}^{\oplus}(v,\{A_1,A_2\}) = 4$ and $f_{co}^{\oplus}(v',\{A_1,A_2\}) = 5$.

The concept of local cooperativeness goes beyond merely satisfying constraints of neighboring agents to accelerate convergence. That is, an agent A_i cooperates with a neighbor agent A_j by selecting a value for its variable that not only satisfies the constraint with A_j, but also maximizes A_j's flexibility (choice of values). If A_i selects v_{max}, then A_j has more choices for a value that satisfies A_j's local constraints and other external constraints with its neighboring agents, which can lead to faster convergence.

Given these definitions of local cooperativeness and flexibility, we can generalize the AWC algorithm by defining three extra cooperation strategies with respect to a flexibility base ⊕:

- ***S**$_{high}$*: Each agent A_i selects value v from V_i which maximizes $f_{co}^{\oplus}(v, N_i^{high})$, i.e., A_i attempts to give maximum flexibility towards its higher priority neighbors;
- ***S**$_{low}$*: Each agent A_i selects value v from V_i which maximizes $f_{co}^{\oplus}(v, N_i^{low})$, i.e., A_i attempts to give maximum flexibility towards its lower priority neighbors; and
- ***S**$_{all}$*: Each agent A_i selects value v from V_i, which maximizes $f_{co}^{\oplus}(v, N_i)$, i.e. max flexibility to all neighbors.

Formalizing our definition of new value ordering strategies and couching it in terms of our distributed continual planning algorithm results in Figure 4's plan repair method for altering a variable that participates in a constraint with another spacecraft. This repair method's behavior depends on the selected cooperation strategy S_α-S_β, where the subscripts are "basic", "low", "high", or "all". As lines 2 and 3 within the repair method imply, the S_α cooperation strategy applies in cases in the previously mentioned *good* case, and the S_β strategy applies in *nogood* cases. Since each subscript can vary over 4 different choices, there are 16 possible strategies for each flexibility base. Since, henceforth, we will only consider strategy combinations, we will refer to them as strategies for short. Note that all the strategies are enhanced with legal value communication (constraint propagation): *indeed, except for S_{basic}, these strategies cannot be applied without legal value communication*. Here, two exemplar strategies are listed:

- ***S**$_{basic}$* **-** ***S**$_{basic}$*: This is the original AWC. Min-conflict heuristic is used for the *good* and *nogood* case.
- ***S**$_{low}$* **–** ***S**$_{high}$*: For the *good* case, an agent is most locally cooperative towards its lower priority neighbor agents by using S_{low} (the selected value doesn't violate the constraints with higher neighbors). On the contrary, for the *nogood* situations, an agent attempts to be most locally cooperative towards its higher priority neighbors by using S_{high}.

Empirical Strategy Comparisons

To provide an initial principled evaluation of these strategies, a number of DCSP experiments were done with an abstract problem setting. Here, agents (variables) are in a 2D grid configuration (each agent is externally constrained with four neighbors except for the ones on the grid boundary). All agents share an identical binary constraint by which a value in an agent is not compatible with a set of values in its neighboring agent: a value is represented as a coordinate in a 2D grid, and the constraint between agents is that the Euclidian distance between two values (coordinates) must be greater than a threshold. In the experiments, the total number of agents was 512 and each agent has 36(=6×6) values in its domain. In addition to the external binary constraint, agents can have a unary local constraint that restricts legal values into a set of randomly selected values among its original 36 values.

Our experiments followed the method used in (Yokoo and Hirayama 1998). In particular, evaluations were performed by measuring Performance evaluation is measured in terms of *cycles* consumed until a solution is found. *Sum* was used for flexibility base (⊕), and the experiments ranged over all possible strategies with this flexibility base. In Figure 5, for expository purpose, only five strategies are presented, which does not change the conclusions in our work. Each data point in the Figure was averaged over 500 test runs. The vertical axis plots the number of cycles and the horizontal axis plots the percentage of locally constrained agents. Each locally constrained agent has a local constraint that restricts available values into randomly selected values. Thus, for example, local constraint ratio 0.1 means that 10 percent of the agents have a local constraint described above.

The results above show that our new value ordering strategies improved conflict resolution performance. Yet, surprisingly, the *most locally cooperative* strategy S_{all}-S_{all} was not the best one. Another key point to note is that choosing the right strategy has significant impact on performance. Certainly, choosing S_{low}-S_{low} or S_{low}-S_{high} instead of S_{basic}-S_{basic} can lead to significant performance improvement, in particular when a large portion of agents are locally constrained. Furthermore, there exists no single dominant strategy for different problem settings.

Predicting DCSP Resolution Strategy Performance

While the previous section shows that different strategies can lead to significant speedups over the original "basic-basic" strategy in AWC, we observe that different strategies dominate in different domains and at different local constraint ratios. Indeed, while "S_{low}-S_{low}" is the best in one domain, "S_{all}-S_{high}" is a more preferable strategy in another domain. Given the variable nature of multi-spacecraft domains, predicting the right strategy to use in a given domain is essential to gain maximum efficiency.

As a formal framework for strategy performance analysis, we have used a MTDP, Multiagent Team Decision Process, model (Pynadath and Tambe 2002). MTDP is based on distributed POMDPs and has been proposed as a framework for analysis. Here we illustrate its actual use in analyzing DCSP performance. MTDP provides a tool for varying key domain parameters to compare the performance of different DCSP value ordering strategies, and thus select the most appropriate strategy in a given situation. We first briefly introduce the MTDP model. Refer to (Pynadath and Tambe 2002) for more details.

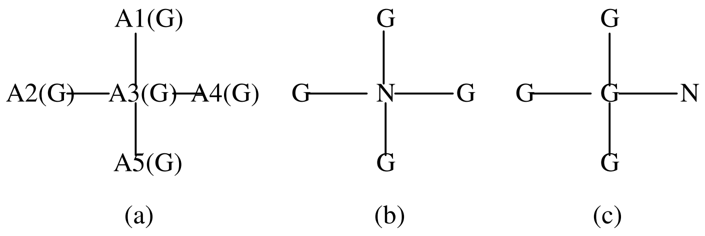


Figure 6: Agent state representation

The MTDP model involves a team of agents operating over a set of world states during a sequence of discrete instances. At each instant each agent chooses an action to perform and the actions are combined to affect a transition to the next instance's world state. Borrowing from distributed POMDPs, the current state is not fully observed/known and transitions to new world states are probabilistic. Each agent makes its own observations to compute its own beliefs, and the performance of the team is evaluated based on a joint reward function over world states and combined actions.

More formally, a MTDP model for a team of agents, α, is a tuple, $\langle S, A_\alpha, P, \Omega_\alpha, O_\alpha, B_\alpha, R\rangle$. S is a set of world states. $A_\alpha = \prod_{i\in\alpha} A_i$ is a set of combined actions where A_i is the set of agent i's actions. P controls the effect of agents' actions in a dynamic environment:

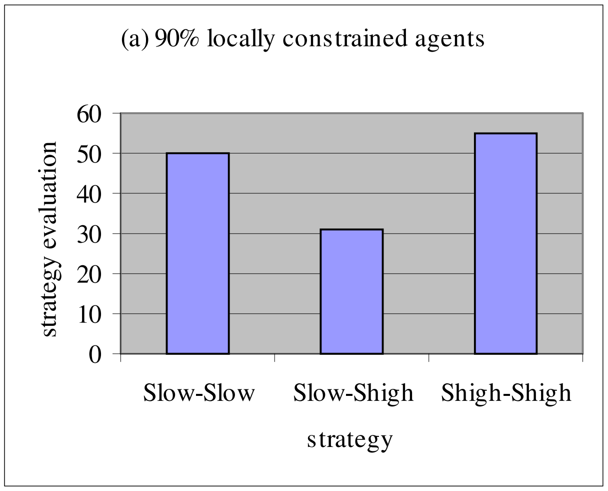
$$P(s,a,s') = Pr(S^{t+1} = s' \mid S^t = s, A^t_\alpha = a)$$

R: $S \times A_\alpha \rightarrow \Re$ is a reward function over states and joint actions. Here, S, A_α, P, and R are the most relevant aspects of the model for this paper: while belief states B_α, Observations O_α, and observation function (which defines the probability distribution of possible observations for an agent i) are key parts of the model, they are not as relevant here and will not be discussed.

A policy in MTDP maps individual agents' belief states to actions; the combination of individual policies thus forms a joint policy for the MTDP. A DCSP strategy is mapped onto a policy in the model. Thus, we compare strategies by evaluating policies in this model. Our initial results from policy evaluation in this model match the actual experimental strategy comparisons shown in Figure 5. Thus, the model could potentially form a basis for predicting strategy performance in novel domains.

Mapping DCSP onto MTDP

In a general mapping, the first question is selecting the right state representation for the MTDP. One typical state representation could be a vector whose elements are the values of all the variables in a DCSP. However, this representation leads to a huge state space. For instance, if there are 10 variables (agents) and 10 possible values per variable, the number of states is 10^{10}. To avoid this combinatorial explosion in state space, we use an abstract state representation in the MTDP. In particular, as described in the previous section, each agent can be



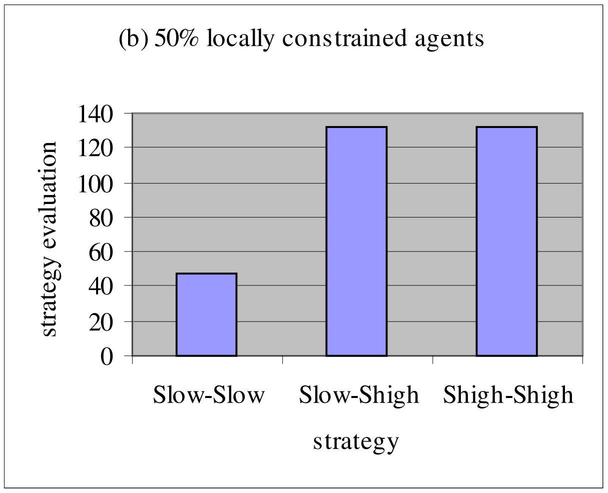


Figure 7: Strategy (policy) evaluation in two different problem settings: analytical results with the MTDP model match to the real experimental results (in Figure 5).

abstractly characterized as being in a good or a nogood state in the enhanced AWC. We use this abstract characterization in our MTDP model. To further reduce the combinatorial explosion, our MTDP state currently models a five-agent system (A1, A2, A3, A4, A5) which represent a local configuration in a 2D grid setting for the above experiments where a middle agent (A3) is surrounded by the other four neighboring agents. Each agent can be in either good (G) or nogood (N) state (Figure 6). For simplicity, the case where agents have no violation is not considered.

Thus, an individual MTDP state 's' is the tuple of local states <s_1, s_2, s_3, s_4, s_5> (e.g., <G, G, G, G, G> if all the five agents are in good case). Thus, there are totally 32 states in the MTDP model, e.g., <G, G, G, G, G>, <G, G, G, G, N>, <G, G, G, N, G>, etc). Here, <G, G, G, G, G> is an initial state since, in AWC, an agent finds no inconsistency for its initial values until it receives the values of its neighboring agents. Agents' value selection actions will cause a transition from one state to another. For instance, if agents are in a state <G, G, G, G, G> (Figure 6-a) and all the agents choose the action "S_{high}", there is a certain transition probability that the next state will be <G, G, N, G, G> (Figure 6-b) as only the third agent is forced into a "nogood" or "N" state. However, the agents may also transition to <G, G, G, N, G> (Figure 6-c) as only the fourth agent may enter the "N" state. While this simple model may appear limiting at first glance, it has already shown promising results --- and we do not expect to need significantly more complex MTDP models for DCSP performance analysis (at least for the types of domains under consideration here).

Agents' value selection actions taken in DCSP control the state transition in MTDP. A DCSP value selection strategy such as "S_{basic}-S_{basic}" provides a function to select a particular value in a given problem state, i.e., a value selection strategy is akin to a policy in the MTDP. In the enhanced AWC, agents' local information is communicated. This communicated information can be modeled as observations in MTDP. Here, note that we do not try to find an optimal policy. Instead, we try to evaluate the utility of a given policy (i.e., a strategy as "S_{low}-S_{low}").

More details on the Initial model and analytical results

Our initial model is based on the five agents state representation described above. Given a state and an action, state transition is derived by combining the local state changes of individual agents. The local state change for an agent is governed by the agent's local state and the states and actions of its neighboring agents. Now, the state transition probability is defined as a product of the probabilities of local state changes as follows.

- $P(s,a,s') = \prod_{i=1}^{5} Pr(s'_i \mid s,a)$ where $s' = \langle s_1', s_2', s_3', s_4', s_5' \rangle$ denotes the next state after state $s = \langle s_1, s_2, s_3, s_4, s_5 \rangle$ with an action 'a'.

Note that the probabilities for individual agents' local state changes are derived from the simulation with 512 agents, not from a simple experiment with only five agents. Rewards are specified for each state in MTDP: the state with a nogood (N) has a positive value (a cost) and the initial state ⟨G, G, G, G, G⟩'s value is zero.

Performance analysis is based on the fact that the best performing strategy has less chance of forcing agents into the nogood case than other strategies. Since the state with 'N' has a cost (which is proportional to the number of N's), as a DCSP strategy performs worse in a given problem setting, the value of an initial state for its corresponding policy will increase since its successive states have more N's. That is, in comparing two policies, the policy with a smaller initial state value is better than the other with a larger value. Here, the value of an initial state is computed with an iterative policy evaluation method.

Figure 7 shows the evaluation (the value of initial states in the MTDP-based model) of three different strategies (S_{low}-S_{low}, S_{low}-S_{high}, S_{high}-S_{high}) in two different problem settings. Empirical results of strategy performance in the previous section (Figure 5) show that, when 90% of all the agents are constrained, the performance of the strategies varies: while S_{low}-S_{high} was the best, S_{low}-S_{low} and S_{high}-S_{high} performed worse. Figure 7-a shows that the analytical results with the MTDP model match to the real experimental results at 90%: that is, the ordering of strategy (policy) performance in Figure 7-a is same with the empirical performance ordering in Figure 5. Figure 7-b shows the analytical results when 50% agents are locally constrained. Here, the analytical results also match to the empirical results in terms of performance ordering. Note that the transition probabilities for the actions in this 50% case are different from the ones used for the 90% case (Figure 7-a). Here, S_{low}-S_{low} (not S_{low}-S_{high}) was the winning strategy.

This result illustrates that MTDP-based model can be used to predict the right strategy to apply in a given situation (possibly with less computation overhead). That is, given a new domain, agents can analyze different strategies with the simple MTDP-based model and select the right strategy for the new domain without running a significant number of problem instances for each strategy to evaluate. Furthermore, this approach will enable agents to flexibly adapt their strategies to changing circumstances. More generally, this result indicates a promising direction for performance analysis in DCSP, and potentially other multiagent systems

Conclusion

While NASA is increasingly interested in multi-platform space missions, a key source of complexity in such missions is coordination and conflict resolution of multiple spacecraft plans. We proposed distributed constraint satisfaction (DCSP) techniques for automated coordination and conflict resolution of such multi-spacecraft plans. We introduced novel value ordering heuristics in DCSP to significantly improve the rate of conflict resolution convergence to meet the efficiency needs of multi-spacecraft missions. In addition, we introduced MTDP (Multiagent Team Decision Process) based techniques for DCSP convergence analysis, which facilitates automated selection of the most appropriate DCSP strategy for a given situation. While we focused on DCSP performance analysis, our approach could be applied to performance modeling and analysis for multiagent systems in general.

In this work we report on how to make an MTDP-based model given a grid of agents. It turns out that there are other topologies that motivate other structures. So far initial work for other topologies is promising, but more needs to be done. Also, generalizing to more complex variable types with ranges of continuous values are needed to appropriately represent spacecraft planning domains. While this explodes the possible value assignments in a DCSP problem, there are ways to circumvent the problem of policy selection by generating discrete abstractions, computing the transition probabilities in the MTDP-based model, analyzing the MTDP policies, and finally using the result to guide strategy selection in the continuous problem. While this approach makes intuitive sense, more work is needed for value assignment. Finding the most (or least) constraining variable assignment gets difficult as the number of possible assignments explodes.

References

E. Mettler and M. Milman, *Space Interferometer Constellation: Formation Maneuvering and Control Architecture*, SPIE Denver '96 Symposium.

S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, *Using Iterative Repair to Improve Responsiveness of Planning and Scheduling*, International Conference on Artificial Intelligence Planning and Scheduling, 2000.

A. Barrett, *From Rovers to Orbiters: Continuous Task Distribution Based Coordination*, NASA Intl. Workshop on Planning and Scheduling for Space, 2000.

M. Yokoo and K. Hirayama, *Distributed Constraint Satisfaction Algorithm for Complex Local Problems*, International Conference on Multi-Agent Systems, 1998.

D. Pynadath and M. Tambe, *Multiagent teamwork: Analyzing key teamwork theories and models*, International Joint Conference on Autonomous Agents and Multiagent Systems Conference, 2002.

S. Minton, M.D. Johnson, A. Philips, and P. Laird, *Solving Large-scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method*, National Conference on Artificial Intelligence, 1990