An Empirical Evaluation of the Effectiveness of Local Search for Replanning

Steve Chien, Russell Knight, and Gregg Rabideau

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
{firstname.lastname}@jpl.nasa.gov

Abstract. Local search has been proposed as a means of responding to changes in problem context requiring replanning. Iterative repair and iterative improvement have desirable properties of preference for plan stability (e.g., non-disruption, minimizing change), and have performed well in a number of practical applications. However, there has been little real empirical evidence to support this case. This paper focuses on the use of local search to support a continuous planning process (e.g., continuously replanning to account for problem changes) as is appropriate for autonomous spacecraft control. We describe results from ongoing empirical tests using the CASPER system to evaluate the effectiveness of local search to replanning using a number of spacecraft scenario simulations including landed operations on a comet and rover operations.

Introduction

In recent years Galileo, Clementine, Mars Pathfinder, Lunar Prospector, and Cassini have all demonstrated a new range of robotic missions to explore our solar system. However, complex missions still require large teams of highly knowledgeable personnel working around the clock to generate and validate spacecraft command sequences. Increasing knowledge of our Earth, our planetary system, and our universe challenges NASA to fly large numbers of ambitious missions, while fiscal realities require doing so with budgets far smaller than in the past. In this climate, the automation of spacecraft commanding becomes an endeavor of crucial importance.

Automated planning is a key enabling technology for autonomous spacecraft. Recent experiences indicate the promise of planning and scheduling technology for

· This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

space operations. Use of the DATA-CHASER automated planning and scheduling system (DCAPS) to command the DATA-CHASER shuttle payload reduced commanding-related mission operations effort by 80% and increased science return by 40% over manually generated sequences (Chien et al. 1999). This increase was possible because short turn-around times (approximately 6 hours) imposed by operations constraints did not allow for lengthy, manual optimization. And the Remote Agent Experiment (ARC, JPL et al. 1999) demonstrated the feasibility of flying AI software (including a planner) to control a spacecraft.

Local iterative algorithms have been successfully applied to planning and scheduling (Minton et al. 1994, Zweben et al. 1994, Chien et al. 1999, Chien et al. 2000b) for a wide range of space applications. Local iterative repair has been proposed as a means of providing a fast replanning capability to enable response to environmental changes (Smith 1994, Chien et al. 2000a). This paper describes an empirical evaluation of the effectiveness of local search in such a replanning context.

The remainder of this paper is organized as follows. First, we briefly describe our approach to local iterative repair. Next we describe how it is applied in a replanning context. We then describe a Comet Nucleus Sample Return (CNSR) scenario and simulation and empirical tests evaluating the effectiveness of local search in finding solutions. Finally, we describe future work, related work and conclusions.

Plan Conflicts and Repair

We now describe the overall approach to iterative repair planning and scheduling in the ASPEN system (Chien et al. 2000b, Rabideau et al. 1999). The ASPEN planning and scheduling system is able to represent a wide range of constraints including:

- Finite enumeration state requirements and changers (e.g., camera ON, OFF);
- Depletable (e.g., fuel) and non-depletable (e.g., 20W power) resource constraints;
- Task decompositions (e.g., Hierarchical Task Networks);
- Complex functional relationships between activity parameters (e.g., downlink time is data/rate + startup); and
- Metric time constraints (e.g., calibrate the camera 20-30s before the observation).

ASPEN supports a suite of search engines to support integrated planning and scheduling to produce activity plans that satisfy the constraints. The remainder of this section briefly outlines the most commonly used search strategy - local iterative repair.

We define a conflict as a particular class of ways to violate a plan constraint (e.g., over-use of a resource or an illegal state transition). For each conflict type, there is a set of repair methods. The search space consists of all possible repair methods applied to all possible conflicts in all possible orders. We describe an efficient approach to searching this space. During iterative repair, the conflicts in the schedule are detected and addressed one at a time until no conflicts exist, or a user-defined time limit has been exceeded. A conflict is a violation of a parameter dependency, tempo-

ral or resource constraint. Conflicts can be repaired by means of several predefined methods. The repair methods are: moving an activity, adding a new instance of an activity, deleting an activity, detailing an activity, abstracting an activity, making a reservation of an activity, canceling a reservation, connecting a temporal constraint, disconnecting a constraint, and changing a parameter value. The repair algorithm first selects a conflict to repair then selects repair method. The type of conflict being resolved determines which methods can repair the conflict. Depending on the selected method, the algorithm may need to make additional decisions. For example, when moving an activity, the algorithm must select a new start time for the activity.

Note that resolving a conflict may cause another conflict so that complete resolution of a set of similar conflicts may require several steps. For example, disconnecting a violated temporal constraint would remove the temporal constraint violation but would introduce a new unconnected temporal constraint conflict. This would then be resolved by creating a new activity and linking in or by linking to an existing activity that correctly satisfies the temporal constraint. Likewise many conflicts can be resolved by abstracting the offending activity. However this causes an undetailed activity conflict and requires re-expansion of the abstract activity. The abstract activity can then be detailed again in a way that perhaps avoids the original conflict(s). Figure 1 shows an example situation for repair. On-board RAM is represented as a depletable resource. The shaded region shows a conflict where the RAM buffer has been oversubscribed. The science activities using the resource prior to the conflict are considered contributors. Moving or deleting one of the contributors can repair the conflict. Another possibility would be to create a new downlink activity in order to replenish the resource and repair the conflict.

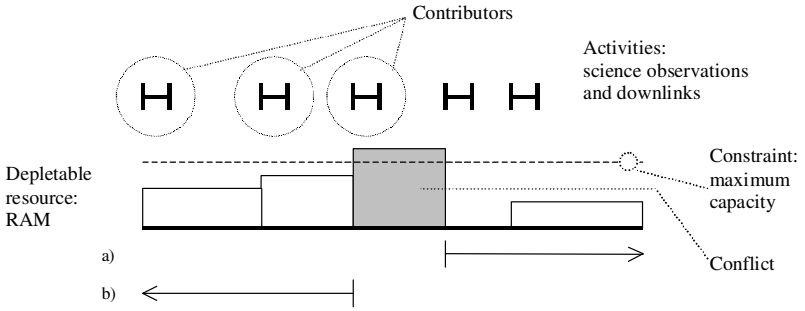

Fig. 1. Repairing a depletable resource conflict. The arrows show time intervals that resolve the conflict by a) moving a positive contributor or b) adding a negative contributor.

Integrating Planning and Execution

Traditionally, planning and scheduling research has focused on a batch formulation of the problem. In this approach, when addressing an ongoing planning problem, time is divided up into a number of planning horizons, each of which lasts for a

significant period of time. When one nears the end of the current horizon, one projects what the state will be at the end of the execution of the current plan (see Figure 2). The planner is invoked with: a new set of goals for the new horizon, the expected initial state for the new horizon, and the planner generates a plan for the new horizon. As an exemplar of this approach, the Remote Agent Experiment operated in this fashion (Jonsson et al 2000).

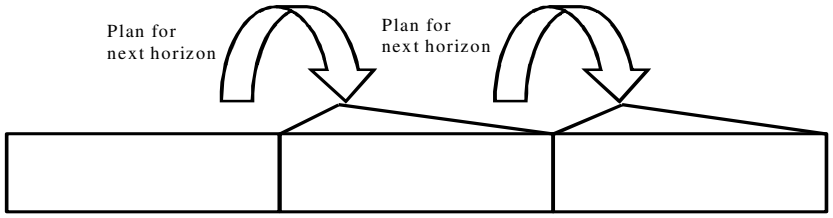

Fig. 2. Traditional Batch "Plan then Execute" Cycle

This approach has a number of drawbacks. In this batch oriented mode, typically planning is considered an off-line process which requires considerable computational effort and there is a significant delay from the time the planner is invoked to the time that the planner produces a new plan.[1] If a negative event occurs (e.g., a plan failure), the response time until a new plan may be significant. During this period the system being controlled must be operated appropriately without planner guidance. If a positive event occurs (e.g., a fortuitous opportunity, such as activities finishing early), again the response time may be significant. If the opportunity is short lived, the system must be able to take advantage of such opportunities without a new plan (because of the delay in generating a new plan). Finally, because the planning process may need to be initiated significantly before the end of the current planning horizon, it may be difficult to project what the state will be when the current plan execution is complete. If the projection is wrong the plan may have difficulty.

To achieve a higher level of responsiveness in a *dynamic planning* situation, we utilize a *continuous planning* approach and have implemented a system called CASPER (for Continuous Activity Scheduling Planning Execution and Replanning). Rather than considering planning a batch process in which a planner is presented with goals and an initial state, the planner has a current goal set, a plan, a current state, and a model of the expected future state. At any time an incremental update to

[1] As a data point, the planner for the Remote Agent Experiment (RAX) flying onboard the New Millennium Deep Space One mission (Jonsson et al 2000) takes approximately 4 hours to produce a 3 day operations plan. RAX is running on a 25 MHz RAD 6000 flight processor and uses roughly 25% of the CPU processing power. While this is a significant improvement over waiting for ground intervention, making the planning process even more responsive (e.g., on a time scale of seconds or tens of seconds) to changes in the operations context, would increase the overall time for which the spacecraft has a consistent plan. As long as a consistent plan exists, the spacecraft can keep busy working on the requested goals and hence may be able to achieve more science goals.

the goals, current state, or planning horizon (at much smaller time increments than batch planning)[2] may update the current state of the plan and thereby invoke the planner process. This update may be an unexpected event or simply time progressing forward. The planner is then responsible for maintaining a consistent, satisficing plan with the most current information. This current plan and projection is the planner's estimation as to what it expects to happen in the world if things go as expected. However, since things rarely go exactly as expected, the planner stands ready to continually modify the plan. From the point of view of the planner, in each cycle the following occurs:

- changes to the goals and the initial state first posted to the plan,
- effects of these changes are propagated through the current plan projections (includes conflict identification)
- plan repair algorithms[3] are invoked to remove conflicts and make the plan appropriate for the current state and goals.

This approach is shown in below in Figure 3.

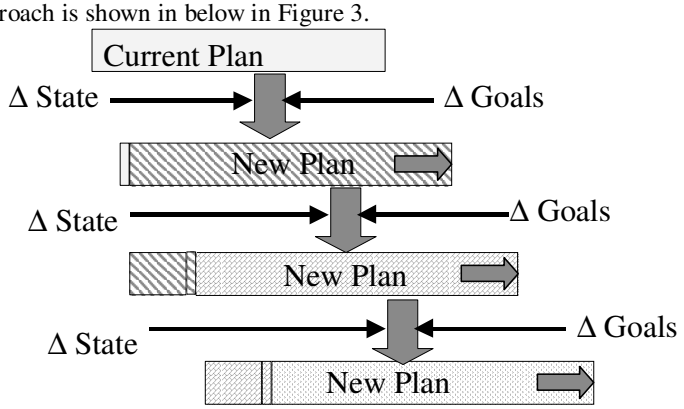

Fig. 3. Continuous Planning Incremental Plan Extension

At each step, the plan is created by using incremental replanning from:

- the portion of the old plan for the current planning horizon;
- the change (Δ) in the goals relevant for the new planning horizon;
- the change (Δ) in the state; and
- the new (extended) planning horizon.

[2] For the spacecraft control domain we are envisaging an update rate on the order of tens of seconds real time.

[3] In this paper we do not focus on the state/resource representation or the repair methods, for details see (Rabideau et al. 1999).

The key assumption of this approach is the belief that:
IF
the change in the goals is small (Δ goals small)
and the change in state is small (Δ state small)
THEN
the change in the plan required to bring the plan consistent with the state and goals will be small (Δ plan small).

Of course, there is no guarantee that this hypothesis will be met. Indeed, if the goals in the domain are highly interacting (e.g., a small change in the combination of goals may require major changes to the plan). However, an important point to note is that the use of varying abstraction levels can aid greatly in making this incremental property hold. Often a plan modification that would require a large number of changes at a lower level of abstraction can be represented as a single change at the more abstract level (e.g., moving a group of activities associated with a single observation could be performed as a single move operation).

This incremental fast replanning approach as embodied in the CASPER system is being used in a range of applications (Chien et al. 2000b) including: onboard a research prototype rover, planned for flight in several space missions, high level flight control and weapons management in an Unmanned Aerial Vehicle prototype, and Ground Communication Station control.

CNSR Landed Operations Testbed

The CNSR scenario represents landed operations of a mission to a comet (see Figure 4). The lander will use a one-meter long drill to collect samples and then feed them to a gas chromatograph/mass spectrometer onboard the lander. This instrument will analyze the composition of the nucleus collected from various depths below the surface. The lander will also carry cameras to photograph the comet surface. Additional instruments planned onboard the lander to determine the chemical makeup of the cometary ices and dust will include an infrared/spectrometer microscope and a gamma-ray spectrometer. After several days on the surface, the lander will bring a sample back to the orbiter for return to Earth.

Fig. 4. Artist depiction of ST4 lander landing on Comet

In this test scenario the planner has models of 11 state and resource timelines, including drill location, battery power, data buffer, and camera state. The model also includes 19 activities such as uplink data, move drill, compress data, take picture, and perform oven experiment.

The nominal mission scenario consists of three major classes of activities: drilling and material transport, instrument activity including imaging and in-situ materials experiments, and data uplink. Of these, drilling is the most complex and unpredictable.

The mission plan calls for three separate drilling activities. Each drilling activity drills a separate hole and acquires samples at three different depths during the process: a surface sample, a 20 cm deep sample, and a one-meter deep sample. Acquiring a sample involves five separate "mining" operations after the hole has been drilled to the desired depth. Each mining operation removes 1 cm of material. Drilling rate and power are unknown a priori, but there are reasonable worst-case estimates available. Drilling can fail altogether for a variety of reasons.

One of the three drilling operations is used to acquire material for sample-return. The other two are used to supply material to in-situ science experiments onboard the lander. These experiments involve depositing the samples in an oven, and taking data while the sample is heated. Between baking operations the oven must cool, but there are two ovens, allowing experiments to be interleaved unless one of the ovens fails.

The replanning capability was tested using a stochastic version of the CNSR simulation described above. This simulation had a number of random variables, which are described below.

- Compression - we model the compression for science data as a normal random variable with a mean of 0.9 and a standard deviation of 0.25*0.9. This has the effect of forcing the planner to respond to buffer over-runs (as described above) and buffer under-runs (to optimize the plan).
- Drilling Time - we model the amount of time to drill in minutes as a random variable with mean of 30 and standard deviation of 3.
- Drilling power - we model the actual power consumption from drilling in watts as a normal random variable with mean 40 and standard deviation 4.

- Oven Failure - we model oven failure occurrence as Poisson distributed with each oven having a 50% chance of failure over the entire mission horizon.
- Data Transmission Rate: we model the time to transmit data in kilobits per second as a normal random variable with a mean of 100 and a standard deviation of 10. This is intended to model the variability in communications to the orbiter.
- Oven Warming and Cooling Times: we model the amount of time to heat up the sample and for the oven to cool down in minutes as random variables with means of 30 and 120, and standard deviations of 3 and 12, respectively. This is intended to model the unknown thermal properties of the samples.

To illustrate the operation of CASPER in the CNSR domain consider the following example of CASPER replanning in response to execution-time resource usage updates. In the CNSR domain, the data collected during the sample activities is compressed and then stored in the data buffer of the lander. This data is uplinked to the orbiting spacecraft at a later time. The planner uses estimates of the amount of data compression to plan when uplink activities are necessary. Because the compression algorithms are content dependent, these estimates may significantly deviate from actual achieved compression.

In this example, the actual data generated by the second sample activity is greater than expected because the compression achieved is less than originally estimated. The planner realizes that it will not have sufficient buffer memory to perform the third sample activity. This results in an over-subscription of the data buffer depletable resource. The planner knows that such a conflict can be repaired by: 1) removing activities that contribute to resource usage or 2) adding an activity that renews the resource. In this case these two options correspond to deleting the third sample activity or adding an uplink activity. (The uplink activity renews the buffer resource by uplinking data to the orbiter.) The planner resolves this conflict by adding an uplink activity after the second sample activity, freeing memory for the third sample activity (see Figure 5).

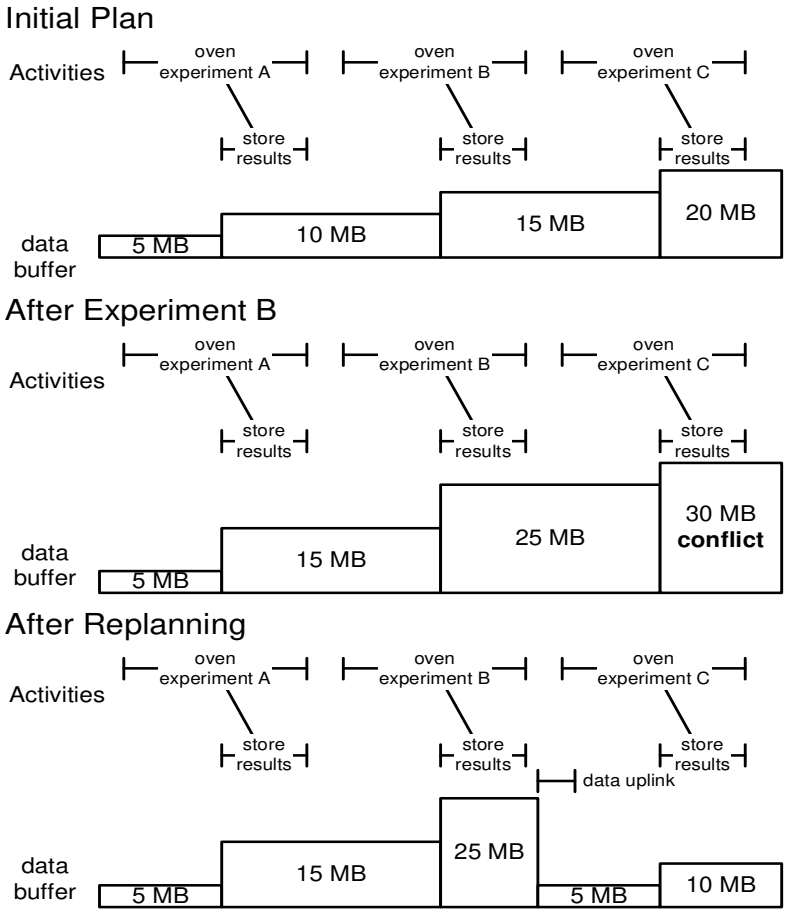

Fig. 5. Over-subscribed Data Buffer Example

Planetary Rover Operations Testbed

The CASPER system has been tested on a model and simulation for the Rocky-7 prototype Mars rover for long-range planetary science gathering (Volpe et al. 1997, Hayati & Arvidson 1997). The model consists of 18 shared resources, 12 state variables and 32 activity types. Resources and states include 3 digital cameras (at the front, rear, and on a mast), a deployable mast, a shovel, a spectrometer, solar arrays, batteries, and a RAM buffer. There are five activity groups that correspond to different types of science experiments: one for collecting spectrometer readings, and four for taking images at a location (a front image, a rear image, a panorama using the

mast, and a close-up image using the mast). Rover problems are sized by the number of hours of daylight (all operations require illuminated solar arrays). A series of science goals are generated corresponding to the number of hours of daylight, and the parameters for the goals are randomly generated (such as target locations). For each additional hour of daylight, ten additional goals are added spread over two new locations. Repair heuristics include traveling salesman heuristics, which attempt to order the rover traversals such that the total distance traveled is minimized.

The rover simulation contains a number of random variables including:

- Traverse speed is modeled as ~N(0.016, 0.0016) meters per second. This is intended to model unpredictable traverse speeds due to terrain (elevation, traction) effects or due to obstacle avoidance.
- Power usage of a traverse is modeled as ~N(4, 0.4) watts to model unpredictable power draw from the wheel motors due to terrain (elevation, traction) effects or due to obstacle avoidance.
- Duration of localization activity is modeled as ~N(120, 12) seconds to model the variability in the amount of time taken to deploy the mast (variances in mechanical actuation) as well as the variable amount of time to acquire images and run the localization algorithm.
- Power usage of localization activity is modeled as ~N(1, 0.1) watts to account for variances in mast movements and camera pointing required for localization.
- Rate of data transmission for data transfer activity is modeled as ~N(10, 1) bytes per second. This is intended to model communications errors and retransmissions that may be required for communications from the rover to a lander or orbiter.
- Power usage for data transfer activity is modeled as ~N(1, 0.1) watts - data transfer consists of transmission, waiting for acknowledgements of receipt of data, and retransmission. Transmission and retransmissions are power intensive, during the waits power consumption is reduced.
- Power usage of the spectrometer activity is modeled as ~N(1, 0.1) watts. The spectrometer activity includes deployment of the spectrometer to the surface of a rock, gathering data, and stowing the instrument. The deployment and stowage require movements of the arm which are dependent on the exact geometries of the surface being sample. These movements and feedback often cause variations in the amount of time and power draw for this activity.
- The amount of data generated (memory usage) for the spectrometer activity is modeled as ~N(0.2, 0.02) bytes per second to account for the effects of content dependent compression.
- Position estimation error rate of increase is modeled as ~N(0.02, 0.002). The position error is the rover's model of how well it knows its position from wheel sensors, etc. This increases at varying rates based on terrain variations as the rover moves.
- Actual rover movements are modeled as a ~N(0, 0.0025) meters per second wander away from the commanded path (0.5 meters away from target location is too far for a science experiment). The actual location vector state variable is used to determine actual distances to feed traverse activity parameters (such as travel

time). Variability here is caused by terrain effects and obstacle avoidance as well as position estimation error.

Empirical Results

In our tests we compare two different local search methods.

1. The CASPER approach - which uses the current plan as an initial seed and performs local, iterative repair to attempt to make the plan consistent with the current state.
2. Batch replanning on failure - in this approach after an update the plan cleared and then local, iterative repair is invoked to construct a plan to achieve the top-level goals from scratch. This differs from the CASPER algorithm in that the seed plan for local search is the null plan (e.g., no actions).

In our setup, CASPER was running on a Sun workstation Ultra 60 with a 359 MHz processor with 1.1 GB Memory. The experiment consisted of 200 runs each (*batch* and *repair*) on randomly generated problems. During each run, the simulator updates the plan an average of 18,000 times (most of these are battery power level updates). On average, around 100 updates result in conflicts that should be handled by the planner/scheduler.

In order gain insight into the effectiveness of the local search in replanning, we examine the number of plan operations and CPU time to either repair the current plan or to construct a new plan to reflect the real-time execution feedback. Table 1 shows the average number of CPU seconds and plan modifications required to repair the existing old plan and construct a new plan from scratch. This data clearly shows that it is more efficient to re-use the old plan than to construct a plan from scratch.

Domain	Average Planning Iterations		Average CPU Seconds	
	Repair Old Plan	Batch Plan	Repair Old Plan	Batch Plan
CNSR	19.73	67.60	2.55	8.10
Rover	15.9	33.6	1.80	3.43

Table 1. Summary of Data Comparing Average # of Plan Iterations and CPU Seconds required to Repair Old Plan vs. Construct New Plan from Scratch.

Table 2 shows, for the CNSR simulations, a histogram plot which indicates the frequency with which problems from feedback required a given number of plan operations to repair (i.e. the height of the column on the Y axis indicates the frequency of problems requiring the number of repair operations indicated on the X axis). Table 3 shows a similar plot for the CPU Time for the CNSR scenario. These plots indicate that a large number of solutions to the replanning problem lie in the neighborhood of the old plan. This validates our hypothesis that if the Δgoals and Δstate are small, that the Δplan should also be small. Additionally the histograms

clearly show that replanning from the old plan is more efficient than batch planning from scratch.

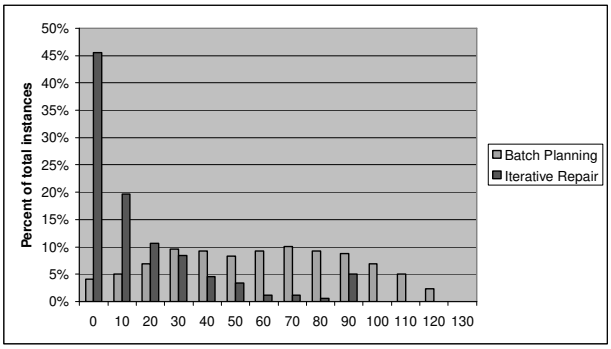

Table 2. CNSR – Number of planning operations

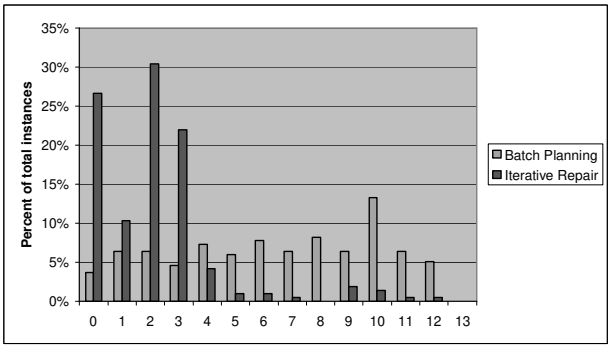

Table 3. CNSR – Time, In seconds

We also briefly present results in a rover operations domain (space constraints preclude a more lengthy presentation of these results). Tables 4 and 5 show the histogram plots for the plan operations and CPU time required to repair the old plan vs. constructing a new plan from scratch. Again the data indicates that not only are a large number of solutions quite close to the old plan but that re-using the old plan is significantly more efficient than replanning from scratch.

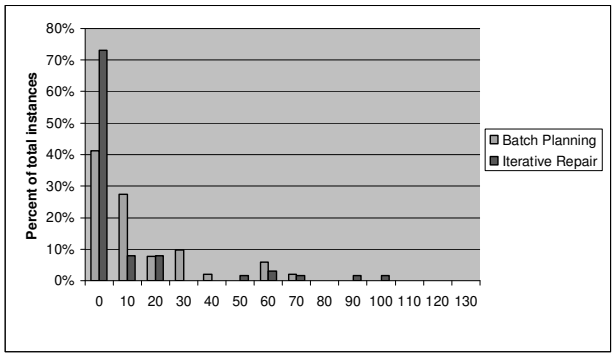

Table 4. Rover – Number of planning operations

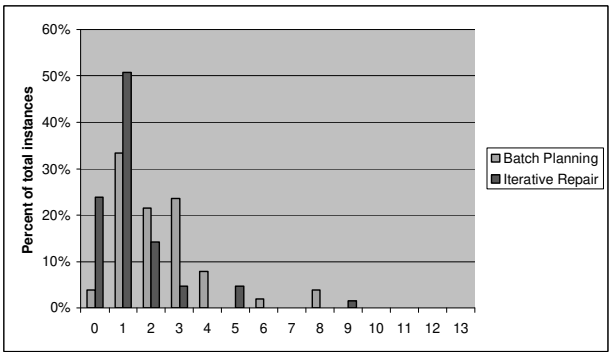

Table 5. Rover – Time, In seconds

Note that while the absolute differences might not seem large, the relative differences are quite significant. The CNSR domain shows a larger than 3x speedup and the Rover domain shows a speedup of almost 2x. These factors become critical when the running on a slower flight processor, which can typically increase CPU times by several orders of magnitude. The data on plan operations was provided to give some measure that is not CPU dependent. However, plan operations are specific to the chosen planner and CASPER's plan operations can vary significantly, with some requiring more time than others.

Related Work

The question of repairability that this paper addresses is strongly linked to the notion of supermodels (Ginsberg & Parkes 1998). In supermodels, they examine the problem of finding a (m,n) SAT model which after having m bits flipped, can be made consistent by flipping a different n bits. They show that for specific classes of prob-

lems this can be reduced to a SAT problem. In contrast, we are interested in finding a plan that after being perturbed by real-world feedback (corresponding to the m bits flipped above) that using a bounded amount of computation (n bits flipped in response above) can be repaired to be consistent (be made a consistent SAT model). Because our perturbation space is much more rich (stochastic elements for states, resource usage, and activity duration) and our plan repair space more rich (add, move, delete, abstract activities) the problems are alike only at the most abstract level. However, the general approach of trying to generate robust plans is exactly the problem of interest and the study in this paper is aimed at evaluating the ability of local search to repair plans.

The high-speed local search techniques used in our continuous planner prototype are an evolution of those developed for the DCAPS system (Chien et al. 1999) that has proven robust in actual applications. In terms of related work, iterative algorithms have been applied to a wide range of computer science problems such as traveling salesman (Lin & Kernighan 1973) as well as Artificial Intelligence Planning (Biefeld & Cooper 1991, Chien & DeJong 1994, Zweben et al. 1994, Hammond 1989, Sussman 1973). Iterative repair algorithms have also been used for a number of scheduling systems. The GERRY system (Zweben et al. 1994) uses iterative repair with a global evaluation function and simulated annealing to schedule space shuttle ground processing activities. The Operations Mission Planner (OMP) (Biefeld & Cooper 1991) system used iterative repair in combination with a historical model of the scheduler actions (called chronologies) to avoid cycling and getting caught in local minima. Work by Johnston and Minton (Johnston & Minton 1994) shows how the min-conflicts heuristic can be used not only for scheduling but also for a wide range of constraint satisfaction problems.

The OPIS system (Smith 1994) can also be viewed as performing iterative repair. However, OPIS is more informed in the application of its repair methods in that it applies a set of analysis measures to classify the bottleneck before selecting a repair method. Excalibur (Nareyek, 1998) represents a general framework for using constraints to unify planning and scheduling constraints, uncertainty, and knowledge. This framework is consistent with the CASPER design, however in this paper we have focused on a lower-level. Specifically, we have focused on re-using the current plan using iterative repair.

Work on the PRODIGY system (Cox & Veloso 1998) has indicated how goals may be altered due to environmental changes/feedback. These changes would be modeled in our framework via task abstraction/retraction and decomposition for potentially failing activities. Other PRODIGY work (Veloso, Pollack, & Cox 1998) has focused on determining which elements of the world state need to be monitored because they affect plan appropriateness. In our approach we have not encountered this bottleneck, our fast state projection techniques enable us to detect relevant changes by noting the introduction of conflicts into the plan.

Work on CPEF (Continuous Planning and Execution Framework) (Myers 1998) uses PRS, AP, and SIPE-2, also represents a similar framework to integrating planning and execution. CPEF and CASPER differ in a number of ways. First, CPEF attempts to cull out key aspects of the world to monitor (as is necessary in general

open-world domains). They also suggest the use of iterative repair (they use the term conservative repairs). And their taxonomy of failure types is very similar to ours in terms of action failure and re-expansion of task networks (re-decomposition). However, in this paper we have focused on lower level issues in synchronization and timing.

Discussion and Conclusions

There are a great many areas for future work, the work presented in this paper is just beginning to analyze local search for replanning. Attempting to characterize better the cases under which problems are nearly decomposable would be extremely useful, as this property is strongly related to whether or not small changes in goal and state would require major changes to the plan. Understanding how abstraction could assist in making large plan changes more tractable is also important. Relating this work to other (non-planning) work in using old invalid solutions as a guide to solve new (slightly modified) problems is also important. For example, dynamic constraint satisfaction has investigated this area. Also, solution density and the phase transition are related to the notion of tractable local search for replanning - better understanding this relationship would be ideal. Finally, one interpretation of the results is that our planner heuristics are more well suited for repair (modification) of an existing plan than for construction of a plan from scratch. This is almost certainly true - but in general we believe that it is easier to construct good heuristics for repairing a plan than for from scratch plan construction. Benchmarking against other from scratch planning methods would be useful.

This paper has described an empirical evaluation of a local search approach to integrating planning and execution for spacecraft control and operations. In this empirical study we investigated the hypothesis that small perturbations in execution of a plan would be resolvable in an efficient fashion by local search. Empirical evidence from two space mission simulations supports the use of local search for this type of problem.

Acknowledgements

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

The authors would like to thank the anonymous reviewers for their many insightful comments that improved the readability of the paper and indicated many excellent areas for future work.

References

NASA Ames & JPL, Remote Agent Experiment Web Page, http://rax.arc.nasa.gov/, 1999.

E. Biefeld and L. Cooper, “Bottleneck Identification Using Process Chronologies,” *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

S. Chien and G. DeJong, “Constructing Simplified Plans via Truth Criteria Approximation,” *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, Chicago, IL, June 1994, pp. 19-24.

S. Chien, G. Rabideau, J. Willis, and T. Mann, “Automating Planning and Scheduling of Shuttle Payload Operations,” *Artificial Intelligence Journal*, 114 (1999) 239-255.

S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proc. 5th International Conference on Artificial Intelligence Planning and Scheduling*, Breckenridge, CO, April 2000.

S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran , "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," Space Operations 2000, Toulouse, France, June 2000.

M. Cox & M. Veloso, "Goal Transformation in Continuous Pannning," in Proceedings of the AAAI Fall Symposium on Distributed Continual Planning, 1998.

B. Drabble, J. Dalton, A. Tate, "Repairing Plans on the Fly," Working Notes of the First International Workshop on Planning and Scheduling for Space, Oxnard, CA 1997.

A. Fukunaga, G. Rabideau, S. Chien, D. Yan, “Towards an Application Framework for Automated Planning and Scheduling,” *Proc. 1997 Int.l Symp. on Art. Int., Robotics and Automation for Space*, Tokyo, Japan, July 1997.

M. Ginsberg and A. Parkes, "Supermodels and Robustness," Proceedins of AAAI-98.

K. Hammond, “Case-based Planning: Viewing Planning as a Memory Task,” Academic Press, San Diego, 1989.

S. Hayati, and R. Arvidson, “Long Range Science Rover (Rocky 7) Mojave Desert Field Tests,” *Proceedings of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Tokyo, Japan, July 1997.

M. Johnston and S. Minton, “Analyzing a Heuristic Strategy for Constraint Satisfaction and Scheduling,” in *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 1994.

A. Jonsson, P. Morris, N. Muscettola, K. Rajan and B. Smith, “Planning in Interplanetary Space: Theory and Practice,” *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems.* Breckenridge, CO. April, 2000.

H. Kautz, B. Selman, “Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search,” *Proceedings AAAI96*.

S. Lin and B. Kernighan, “An Effective Heuristic for the Traveling Salesman Problem,” *Operations Research Vol. 21*, 1973.

S. Minton “Automatically Configuring Constraint Satisfaction Programs: A Case Study.” *Constraints* 1:1(7-43).1996.

N. Muscettola, B. Smith, S. Chien , C. Fry , K. Rajan, S. Mohan, G. Rabideau , D. Yan, “On-board Planning for the New Millennium Deep Space One Spacecraft,” *Proceedings of the 1997 IEEE Aerospace Conference*, Aspen, CO, February, 1997, v. 1, pp. 303-318.

K. Myers, "Towards a Framework for Continuous Planning and Execution", in Proceedings of the AAAI Fall Symposium on Distributed Continual Planning, 1998.

A. Nareyek, "A Planning Model for Agents in Dynamic and Unicertain Real-Time Environments," in Integrating Planning, Scheduling, and Execution in Dynamic and Uncertain Environments, AIPS98 Workshop, AAAI Technical Report WS-98092.

B. Pell, D. Bernard, S. Chien, E. Gat, N. Muscettola, P. Nayak, M. Wagner, and B. Williams, " An Autonomous Spacecraft Agent Prototype," *Autonomous Robots,* March 1998.

G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," Int Symp on Artificial Intelligence Robotics and Aut. in Space (ISAIRAS), Noordwijk, The Netherlands, June 1999.

R. Simmons, "Combining Associational and Causal Reasoning to Solve Interpretation and Planning Problems," Tech. Rep., MIT Artificial Intelligence Laboratory, 1988.

S. Smith, "OPIS: An Architecture and Methodology for Reactive Scheduling," in *Intelligent Scheduling*, Morgan Kaufman, 1994.

G. Sussman, "A Computational Model of Skill Acquisition," Technical Report, MIT Artificial Intelligence Laboratory, 1973.

M. Veloso, M. Pollack, M. Cox, "Rationale-based monitoring for planning in dynamic environments," Proceedings Artificial Intelligence Planning Systems Conference, Pittsburgh, PA, 1998.

R. Volpe, J. Balaram, T. Ohm, and R. Ivlev, "Rocky 7: A Next Generation Mars Rover Prototype," *Journal of Advanced Robotic*s, 11(4), December 1997.

M. Zweben, B. Daun, E. Davis, and M. Deale, "Scheduling and Rescheduling with Iterative Repair," in *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 1994.