Massachusetts Institute of Technology

"Enabling Fast Flexible Planning through Incremental Temporal Reasoning with Conflict Extraction"

I'shiang Shu, Robert Effinger, Prof. Brian Williams

Model-Based Embedded and Robotic Systems Group
Massachusetts Institute of Technology

ICAPS 2005

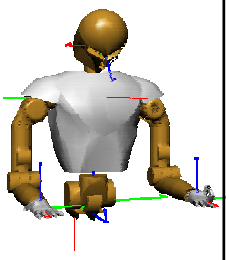
Model-based Embedded & Robotic Systems

CSAIL

We Need Fast, Flexible, and Reliable Planning

ATRV Rover Testbed

Robonaut Simulator



Kirk Planner

combines:

Contingency Planning

- **Redundant Methods**
  TPN (Kim, Williams, Abrahmson 01)
- **Conflict-directed Plan Repair**
  Dynamic Backtracking (Ginsberg 96)

Temporally Flexible Planning

- **Flexible Execution Times**
  STN (Dechter, Meiri, Pearl, 91)
- **Incremental Reasoning**
  (Cesta, Oddi 96) and many others
- **Conflict Extraction**
  (Shu, Effinger, Williams 05)

Kirk Planner

combines:

Contingency Planning

Temporally Flexible Planning

Kirk Planner

- **Redundant Meth**
  TPN (Kim, Williams,
- **Conflict-directed**
  Dynamic Backtracking

tion Times
ri, Pearl, 91)
asoning
nd many others
tion
iams 05)

Kirk Planner

combines:

Contingency Planning

Temporally Flexible Planning

Kirk Planner

- **Flexible Execution Times**
- **Incremental Reasoning**
- **Conflict Extraction**

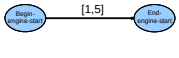
ITC Algorithm

- **Redundant Methods**
- **Conflict-directed Plan Repair**

Flexible Execution Times

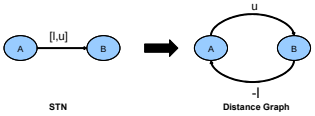
(Dechter, Meiri, Pearl 91)

Simple Temporal Network (STN):



$$T_j - T_i \in [1,5]$$

Equivalent Distance Graph Representation:

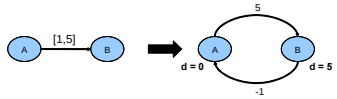


$$T_j - T_i \in [l,u] \Rightarrow T_j - T_i \le u \cap T_i - T_j \le -l$$

Flexible Execution Times

(Dechter, Meiri, Pearl 91)

A Simple STN:



Consistent !

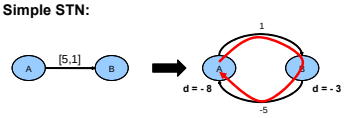
Determine STN consistency:

- **Calculate the Single Source Shortest Path (polynomial-time algorithm)**

Flexible Execution Times

(Dechter, Meiri, Pearl 91)

A Simple STN:



Inconsistent STN !

Determine STN consistency:

- **Calculate the Single Source Shortest Path (polynomial-time algorithm)**
- **A continually looping negative cycle indicates an inconsistency in STN**

Two methods to detect a continually looping negative cycle

1.) Check for any d-value to drop below –nC. (most space efficient)
2.) Keep an acyclic spanning tree of support, and terminate when a self-loop is formed. (Cesta, Oddi 96) (most time efficient)

Kirk Planner: Overview

combines:

Contingency Planning

Temporally Flexible Planning

Kirk Planner

ITC Algorithm

- Flexible Execution Times
- Incremental Reasoning
- Conflict Extraction
- Redundant Methods
- Conflict-directed Plan Repair

ITC Algorithm

- Basic Idea:

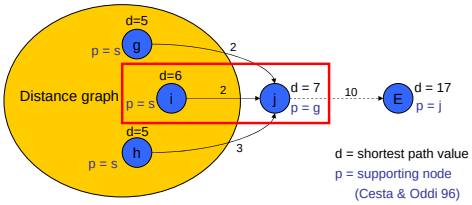
  1.) Keep dependency information for each shortest-path value in the distance graph (Cesta, Oddi 96)

  2.) Use incremental update rules to localize necessary changes to the distance graph.
  a.) 3 Update Rules to change a consistent distance graph.
  b.) 3 Update Rules to repair an inconsistent distance graph.

- ITC's Novel Claims:

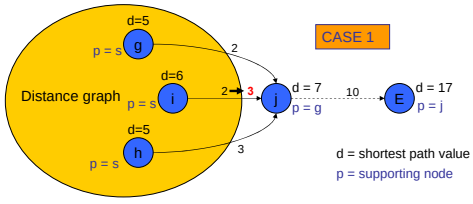
  1.) A conflict extraction mechanism to guide plan repair
  2.) Allow multiple arc-changes
  3.) Can repair inconsistent distance graphs incrementally

3 Update Rules to Change a Consistent Distance Graph



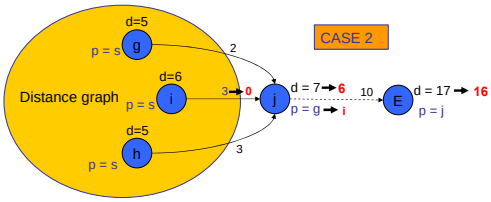
- Keep track of the support for each shortest path value
- Given a consistent STN, changing Arc(i,j)'s cost can have three possible effects on the shortest path.
  1. Arc(i,j) change does not affect the shortest path to node j.
  2. Arc(i,j) change improves the shortest path to node j.
  3. Arc(i,j) change invalidates the shortest path to node j.

1.) Arc(i,j) change does not affect shortest path



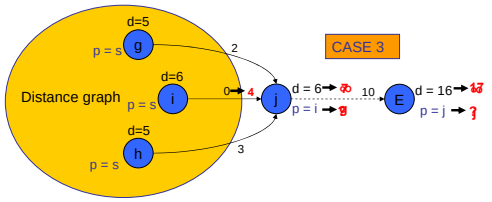
- The cost from node *i* to node *j* increases from *2* to *3*.
- No changes are needed.

2.) Arc(i,j) change improves shortest path to *j*



- The cost from node *i* to node *j* decreases from 3 to 0.
- Propagate the improved shortest path.

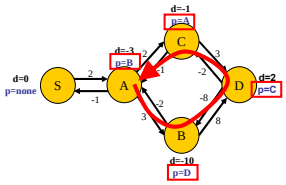
3.) Arc(i,j) change invalidates shortest path to *j*



- Increasing Arc(i,j) now invalidates node *j's* shortest path.
  - Reset node *j*
  - Recursively reset nodes dependent upon node j.
  - Insert node *j*'s parents into the queue so that a new path to node *j* can be found for node *j* and all other invalidated nodes.

3 Update Rules to repair an Inconsistent distance graph

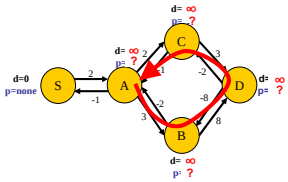
- ITC discovers an inconsistency (a negative cycle) by detecting cyclically dependent backpointers.



This must be a negative cycle cycle: (A,B,D,C,A)

3 Update Rules to repair an Inconsistent distance graph

- Now ITC must incrementally repair the inconsistency.

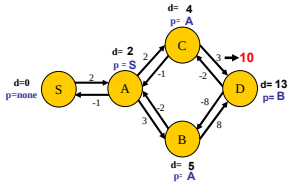


Negative cycle: (A,B,D,C,A)

- Three repair steps:
  1.) Reset all nodes in negative cycle.
  2.) Recursively reset all nodes that depend on the negative cycle nodes.
  3.) Put any parent of a reset node that was not also reset on the Q.

3 Update Rules to repair an Inconsistent distance graph

- Now ITC must incrementally repair the inconsistency.



Consistent !

- Change arc cost CD to 10.
- Propagate the new shortest path values

Kirk Planner: Overview

combines:

Contingency Planning

Temporally Flexible Planning

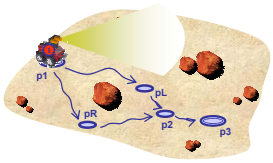
Kirk Planner

- Flexible Execution Times
- Incremental Reasoning
- Conflict Extraction

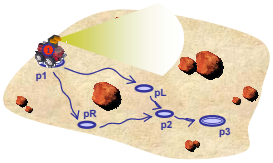
ITC Algorithm

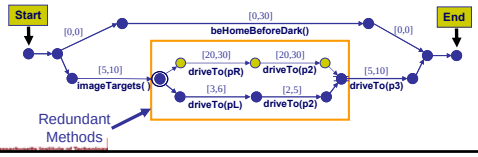
- Redundant Methods
- Conflict-directed Plan Repair

Temporal Plan Network (Kim, Williams, Abrahmson 01)

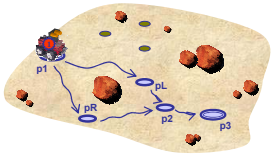


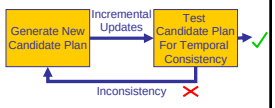
Temporal Plan Network (Kim, Williams, Abrahmson 01)

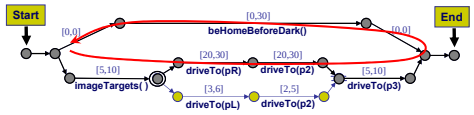




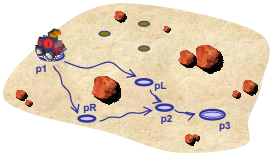
Conflict-Directed Plan Repair

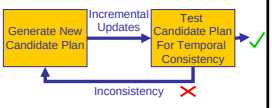


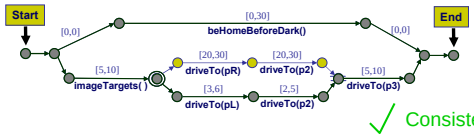




Conflict-Directed Plan Repair

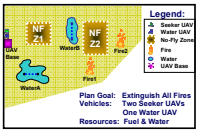


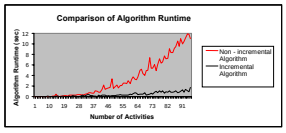




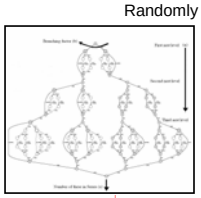
Performance Improvements

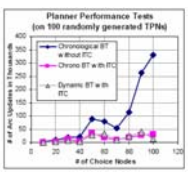
UAV Scenarios





Randomly Generated Plans





Conclusions

- ITC is an incremental shortest path algorithm that can repair distance graphs incrementally as the plan changes
- ITC's Novel Claims:
  1.) A conflict extraction mechanism
  2.) Allow multiple arc-changes at once
  3.) Can incrementally repair inconsistent distance graphs
- Shows an order of magnitude improvement over non - incremental planning
- Applicable to any plan representation that uses disjunctions of simple temporal constraints.

Any Questions?

References

Ahuja, R.; Magnanti T.; Orlin J, 1958. Network Flows: Theory, Algorithms, and Applications. Prentice Hall.

Cesta, A. and Oddi, A., 1996. Gaining Efficiency and Flexibility in the Simple Temporal Problem, *3rd Workshop on Temporal Representation and Reasoning.*

Dechter, R.; Meiri, I.; Pearl, J., 1991. Temporal Constraint Networks. *Artificial Intelligence*, 49:61-95.

Estlin, T.; et al., 2000. Using Continuous Planning Techniques to Coordinate Multiple Rovers. *Electronic Transactions on Artificial Inttligence,* 4:45-57.

Gelle, E. and Sabin M., 2003. Solving Methods for Conditional Constraint Satisfaction. In *IJCAI-2003.*

Gerevini, A., et al., 1996. Incremental Algorithms for Managing Temporal Constraints. *8th International Conference of Tools with Artificial Intelligence.*

Ginsberg, M.L., 1993. Dynamic backtracking. Journal of AI Research, 1:25-46.

Kim, P.; Williams, B.; and Abrahmson, M, 2001. Executing Reactive, Model-based Programs through Graph-based Temporal Planning. *IJCAI-2001.*

Koenig, S. and Likhachev. M., 2001. Incremental A*. In *Adv. in Neural Information Processing Systems 14.*

McAllester, D., 1991. Truth Maintenance. In *Proceedings of AAAI-90*, 1109-1116.

Muscettola N., et al., 1998. Issues in temporal reasoning for autonomous control systems. In *Autonomous Agents*

Prosser, P., 1993. Hybrid algorithms for the constraint satisfaction problem, *Comp. Intelligence. 3* 268-299.

References (continued)

Rabideau, G., et.al., 1999. Iterative Repair Planning for Spacecraft Operations in the ASPEN System. *ISAIRAS.*

Stergiou, K, and Koubarakis, M., 1998. Backtracking Algorithms for Disjunctions of Temporal Constraints. *15th Nat. Conference of Artificial Intelligence,* 81-117.

Tsamardinos, I., Vidal T., Pollack, M., 2003. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints.*, vol. 8, no. 4, pp. 365-388.

Williams, B. and Ragno, J., 2002. Conflict-directed A* and its role in model-based embedded systems. *Journal of Discrete Applied Math.*