ARL-MR-0926 ● MAR 2016

US Army Research Laboratory

Integration of Hierarchical Goal Network Planning and Autonomous Path Planning

by Nicholas C Fung

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-MR-0926 ● MAR 2016

US Army Research Laboratory

Integration of Hierarchical Goal Network Planning and Autonomous Path Planning

by Nicholas C Fung
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
March 2016	Final	01/2015–09/2015

4. TITLE AND SUBTITLE
Integration of Hierarchical Goal Network Planning and Autonomous Path Planning

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Nicholas C Fung

5d. PROJECT NUMBER
R.0013626.6.163.1

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CII-A
2800 Powder Mill Road
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-MR-0926

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Automated planning has become an increasingly influential area of research in the realm of artificial intelligence. Task-based planning algorithms provide a number of advantages, including the ease of human readability when creating mission-length plans. However, such planning algorithms are rarely implemented on real-world robotic systems. This report documents work to integrate a hierarchical goal network planning algorithm with low-level path planning. The system uses the Goal Decomposition with Landmarks (GoDeL) low-level path planner to create a plan, consisting of a sequence of actions, to attain the goals state. The domain is a robot operating in a known office environment with labeled rooms and doors that can be manipulated. The report goes on to discuss future improvement of the system with the goal of creating a robust system that can operate on a robotic platform in a dynamic environment.

15. SUBJECT TERMS
hierarchical task network, HTN, hierarchical goal network, HGN, planning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Nicholas C Fung
Unclassified	Unclassified	Unclassified	UU	16	19b. TELEPHONE NUMBER (Include area code) 301-394-3101

Contents

List of Figures

List of Tables

1. Introduction

Automated planning is a form of artificial intelligence that generates action sequences to accomplish a specific goal. Such techniques are often used when creating intelligent systems and are of particular interest for the implementation of autonomous robotic platforms. Within the field of autonomous planning, task planning has a number of advantages, including the ability to form long, human-readable plans. However, they have largely remained theoretical because of the challenges encountered when integrating them into a complex, autonomous system.

A hierarchical task network (HTN) is a structure used to create plans to accomplish specific goals. The graph structure is created by breaking down tasks into subtasks using actions and methods that are defined within the domain. A sample HTN in the blocks world domain can be seen in Fig. 1. A task defines an advancement of the system state to attain a specific subsequent state and is of the form task(literal1,literal2) for any number of literals. As an example, a task may be "pick-up(apple,kitchen table,right hand)" representing picking up the apple from the kitchen table with your right hand. A primitive task is a one that can be accomplished by a single operation. This operation is called an action.

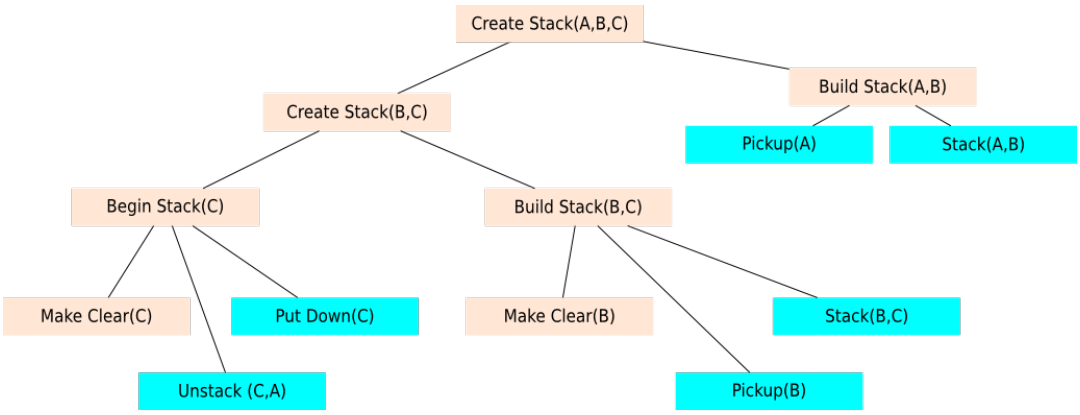

Fig. 1 HTN of a sample problem in the blocks world domain

A complex task is a one that requires more than 1 operation. A method is an operation that contains multiple subtasks, requiring the subsequent application of multiple actions or methods. Complex tasks can continually be broken down until only primitive tasks remain. A sequence of actions can be directly applied to the system in sequence in order to accomplish these primitive tasks. These sequence of actions is called a plan. Because each task and each literal can be named according to the user-defined domain documentation, the plan can easily be interpreted by a human reader. Similarly, a system state is equally readable, making it easy for a human to issue goals to the planner.

Hierarchical goal networks (HGNs) differ from HTNs in that they do not make use of tasks. While HTN planners break down tasks into subtasks and actions, an HGN breaks down goals into subgoals. This structure requires different definitions of actions and methods. For HTN planners, actions are applied to complete primitive tasks and methods are applied to decompose complex tasks into subtasks. In HGN planners, actions achieve a specific goal and methods are applied to decompose a specific goal into multiple subgoals. The planner iteratively plans on each goal in the graph structure. Each subgoal is assigned an action to achieve this subgoal or is decomposed into more subgoals. When each subgoal can be accomplished through the execution of a single action, this sequence of actions is a valid plan to attain the overall goal.

One of the difficulties in applying HTN or HGN planners to complex domains found in real-world scenarios is that the domain must be explicitly defined. Tasks, actions, and methods must be defined with preconditions and expected outcomes so that the planner can determine which operations can be applied in a specific state and what the outcome of the operation will produce. This can require a large amount of a priori work by the user. An important distinction between HTN and HGN planners is that methods in HGN planners can be more loosely defined. While executing a method operation within an HGN planner may involve the transition of many state variables, the definition may explicitly include many or few of these transitions. Methods with more detailed state variable transitions produce more directed searches of the state space.

One of the primary difficulties in implementing an HTN- or HGN-based planner on a robotic platform is in translating the actions generated by the planner. Because tasks and how they are decomposed into action, methods, and subtasks are user defined specifically for a particular domain, they are typically complex concepts. For example, an action may be “pick up box A”. While such instruction is easily understandable by a human, this requires numerous motor control instructions for a robot to execute. If, instead, actions are defined at the motor control level, the action definitions would be too complex for a human author.

The report documents efforts to integrate an HGN planner with a navigation system on a robotic platform within a simulation environment. The HGN planner is used at a high level to form an overall mission plan. The navigation system uses a low-level search-based action planner in order to form the motor control signals.

2. System

2.1 GoDeL Planner

The Goal Decomposition with Landmarks (GoDeL) planner, developed by Shivashankar,[1] is an HGN planner that uses landmarks to break down goals into subgoals. GoDeL operates as an HGN planner as described in Section 1 with additional rules to aid the planner under sparse domain information. As in generalized HGN planners, the algorithm assigns actions to achieve goals and breaks down goals into subgoals using methods when applicable. However, when a goal cannot directly be attained by an action or decomposed by a method, GoDeL will work to decompose the goal using landmarks. A landmark is a specific state that must be satisfied in any valid plan to attain a specific goal. For example, any plan to put an object down must at some point contain the state of holding the object. GoDeL uses the LAMA[2] algorithm to identify landmarks for a given subgoal. If the algorithm encounters a goal that does not have a relevant action or method and cannot be decomposed into landmarks, the state space is searched through nondeterministic action chaining. Through this progression, GoDeL uses whatever information it has to limit the search space and run more efficiently. However with limited access to useful methods, the algorithm will still search through the entire search space and will thus find a valid solution plan if it exists.

2.2 Waypoint Navigation

The waypoint navigation system works to provide semi-autonomous capabilities to a robotic platform. The system takes in a goal orientation, known as a waypoint, consisting of a map coordinate and pose. A path to this goal orientation, consisting of a series of poses, is then generated. The robot attempts to follow the path as closely as possible in order to obtain the goal pose.

The path is obtained through the use of the Search-Based Planning Library.[3] This library contains a generic set of domain-independent graph search planners, meant to be applicable to a variety of systems. This waypoint navigation implementation makes use of the ARA* anytime planner.

2.3 Test Bed Implementation

The integration of the planner and navigation system was implemented in a simulation environment under the robot operating system (ROS). Within this environment, an iRobot PackBot platform navigates its way through a known area. The PackBot is a versatile robot platform that has been used by the US Army

Research Laboratory for a number of different research and development systems. The area is an experimental floor plan meant to simulate a typical apartment.

In this scenario, doors that may be open or closed are implemented so that the waypoint navigation planner is not sufficient by itself. While the waypoint navigation planner can create paths between open rooms, closed doors may block a valid path from being created. In order to create a complete plan, the simulated robot must make use of an additional action to open doors. Although the system does not include motor control functions for the PackBot to open doors, the simulated robot was given this ability to increase the complexity of the problem. This could have a real-world analogy of using a remote control to open an electronic door or signaling a person to open the door for the robot.

The test domain is specifically kept to a small number of actions because this system is specifically designed as a proof of concept. The available actions are *move(robot,location1,location2)* and *open(robot,door).* A method *traverse(robot,location1,location2)* is also included in the domain and is designed to address the movement of the robot across several rooms with closed doors. The method is applied to a location goal (e.g., *robot-at(robot,location)*) and breaks the goal down into subgoals of the robot at intermediate rooms and open doors between these rooms. GoDeL uses these actions and methods to create an action plan consisting of *move* and *open* commands. This queue of action commands are interpreted by C++ code contained in a ROS node. The behaviors of this node is tailored by the user for this specific domain. It translates *move* actions into waypoints to be sent to the waypoint navigation system. The node executes *open* actions by directly manipulating the environment, clearing the path between 2 rooms, simulating the act of opening a door.

In this manner, the system demonstrates additional capabilities that are added to the waypoint navigation planner. The HGN planner is able to create *move* and *open* commands to establish a valid plan. These commands are applied through the system into autonomous selection of waypoints and door opening commands. The HGN planner is also extended in its ability to carry out actions. While a command to *move(robot1,location1)* is abstract in the literal sense, the system is able to translate this into a waypoint. It can then use the waypoint navigation planner to provide motor control actions to the robot.

3. Results

The system was demonstrated to operate under the simulation environment. The HGN planner is able to create valid plans consisting of move and open actions. A

sample plan can be seen in the following Table. These actions are then executed by the robot in order to attain specific goals. A screenshot of the simulator visualized in rviz can be seen in Fig. 2. This figure demonstrates the robot moving along a path created by the navigation system to a waypoint selected by the HGN planner. This system successfully provides a proof of concept that can serve to justify continued work in this direction.

Table. Sample plan from simulation

Plan:
1:move(robot1,loc1,loc2)
2:open-door(robot1,door1,loc2,loc3)
3:move(robot1,loc2,loc3)
4:move(robot1,loc3,loc4)

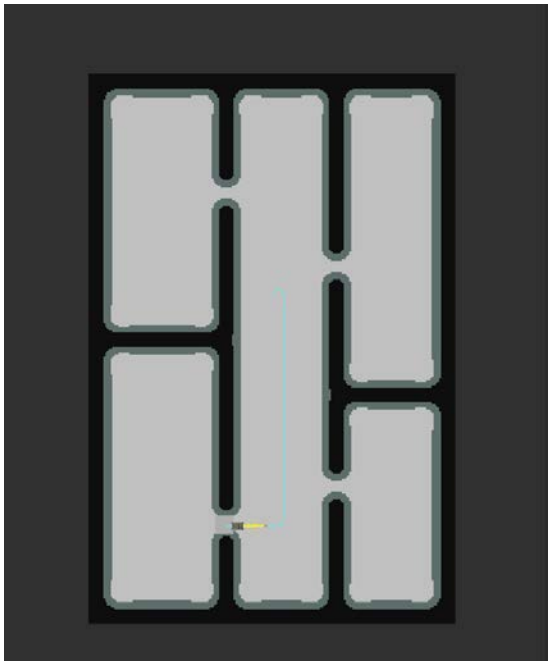

Fig. 2 Screen capture of the simulation in progress

4. Future Work

4.1 Plan Repair

To make the system more robust, plan repair capabilities will be included. Operation of the robotic platform in a real-world environment introduces dynamic events and nondeterministic outcomes. Dynamic events could include a door shutting without robot interaction. Nondeterministic outcomes of an action include

the robot slipping in its movements. Both of these additional factors result in the possibility that a plan could be invalidated during execution, either because it cannot be executed or because the outcome will no longer satisfy the goal. When such an event occurs, the system must work to repair the plan. In the worst case, this will require a completely new plan to be created. However, computation time can be saved if parts of the original plan can be reused.

Ayan and Kuter have created the Hierarchical Ordered Task Replanning in Dynamic Environments (HOTRiDE) algorithm[4] in order to address this problem. This system uses the structure of an HTN planner to enact limited replanning on an invalidated plan while retaining as much of the original plan as possible. As the HTN structure is being created during the initial planning, HOTRiDE keeps track of causal links. A causal link is created between the outcome of one action and the preconditions of another action. If the first action produces an unexpected outcome, the preconditions of the linked action must be checked to ensure that it is still valid. In addition, the plan as a whole must be checked to ensure that the overall goal is still satisfied. HOTRiDE then uses these causal links to evaluate the plan when an action has an unexpected outcome. It searches the graph to find the *minimal failed task*. This task, and any other tasks it supports through causal links, must be replanned through a state space search.

To continue the work of this report, a parallel of HOTRiDE that can be applied to HGN planners such as GoDeL is being created. This new algorithm looks to take advantage of the structure of the HGN as opposed to that of an HTN. Because each node of the HGN is a goal, the repair algorithm can take advantage of the original search space. While HOTRiDE must look to complete each subtask, this new replanning algorithm can look to replan specific subgoals. Subgoals that were added through inefficient state space searching can be included in the replan in order to try and form a more efficient plan.

4.2 Dynamic Domain and Goals

As an additional area of research, a complete planning system for a dynamic system can include dynamic domain redefinitions. This capability is especially important for operation of the system with incomplete knowledge of the domain. For example, a robotic system working to explore an unknown building may find an unexpected door. A robust system would be able to add this door to the domain and invoke a replanning procedure. This may result in a more efficient plan to accomplish the goal state.

Similarly, the system could have cause to dynamically alter its goals. In the context of a search and rescue mission after a natural disaster, a robot could encounter an

injured person. The system would have need to dynamically add a goal in order to rescue this person or call for aid with a high priority. A robust planning system would be able to add this goal and invoke a replanning procedure that would look to achieve this high priority goal as quickly as possible before continuing with the original plan.

5. References

1. Shivashankar V, Alford R, Kuter U, Nau D. The GoDeL planning system: a more perfect union of domain-independent and hierarchical planning. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence; IJCAI 2013. 2013 Aug 3–9; Beijing, China. p. 238.

2. Richter S, Westphal M. The LAMA planner: guiding cost-based anytime planning with landmarks. J Artificial Intel Res. 2010;39(1):127–177.

3. Cohen BJ, Chitta S, Likhachev M. Search-based planning for manipulation with motion primitives. In: ICRA 2010. Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA); 2010 May 3–7; Anchorage, AK. p. 2902–2908.

4. Ayan NF, Kuter U, Yaman F, Goldman RP. Hotride: hierarchical ordered task replanning in dynamic environments. In: Planning and Plan Execution for Real-World Systems–Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop; ICAPS 2007. 2007 Sep 22–26; Providence, RI.

1 (PDF)	DEFENSE TECH INFO CTR DTIC OCA
2 (PDF)	US ARMY RSRCH LAB IMAL HRA MAIL & RECORDS MGMT RDRL CIO LL TECHL LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	US ARMY RSRCH LAB RDRL CII A N FUNG

INTENTIONALLY LEFT BLANK.