Autonomously Constructing Hierarchical Task Networks for Planning and Human-Robot Collaboration

Bradley Hayes[1] and Brian Scassellati[1]

Abstract— **Collaboration between humans and robots requires solutions to an array of challenging problems, including multi-agent planning, state estimation, and goal inference. There already exist feasible solutions for many of these challenges, but they depend upon having rich task models. In this work we detail a novel type of Hierarchical Task Network we call a Clique/Chain HTN (CC-HTN), alongside an algorithm for autonomously constructing them from topological properties derived from graphical task representations. As the presented method relies on the structure of the task itself, our work imposes no particular type of symbolic insight into motor primitives or environmental representation, making it applicable to a wide variety of use cases critical to human-robot interaction. We present evaluations within a multi-resolution goal inference task and a transfer learning application showing the utility of our approach.**

I. Introduction

Combining user-friendly skill acquisition and robust task-level planning in real-world systems is critical to the widespread adoption of collaborative robots. While demonstration-based training techniques inherently afford ease of use, their informality comes at the expense of not necessarily producing complete specifications of their skills' intentions or environment effects [1]. On the other hand, flexible planning systems are able to operate at varying levels of abstraction to quickly produce efficient solutions [2]. These systems typically require either manually annotated hierarchical structure or well-specified preconditions and effects of their atomic-level planning components [3]. Just as humans do not rely on explicit annotations or out-of-scenario interventions, we expect competent robot partners both to learn skill execution policies and to infer hierarchical task structure from observation or experience.

In addition to producing acceptable sequences of skills, planning systems represent an agent's awareness of the task's underlying structure, allowing for adjustment in case of unexpected deviation. While robots operating in isolation may seek to minimize a cost function specified in terms of effort or time required, it may be desirable for robot teammates to instead choose to optimize for flexible execution or maximally supportive roles [4]. As such, collaborative robots require a deeper understanding of tasks than their isolated counterparts, and must be capable of reasoning about and planning around potentially complex subtask assignment and resource management constraints [5]. Therefore a robot becomes increasingly capable of adapting to changing circumstances by deferring decisions, mitigating both its own failures and those of its partners as it learns more about the underlying structure of a task. This is known as *least-commitment planning*, an important concept in multi-agent domains with uncertainty [6].

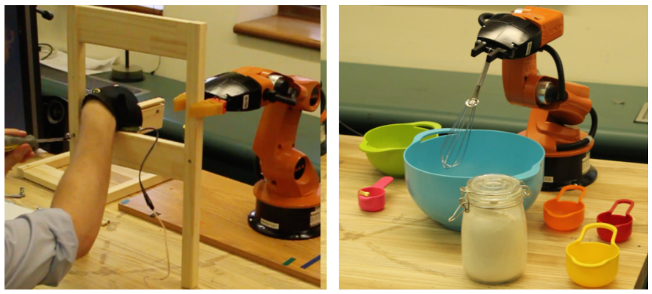


Fig. 1: Collaboration relies on complex planning and intention recognition capabilities. For many tasks, especially within the assembly and cooking domains we base our evaluations on, hierarchical models are required to identify and solve tractable sub-problems.

Effectively accomplishing tasks in collaborative domains demands a high level of execution flexibility that can only emerge through task comprehension. Manually providing a robot with this kind of knowledge is difficult and time consuming and does not scale well with increasing deployment of robots in the workforce or home. Mechanisms that allow robots to build their own hierarchical interpretations of tasks are central to addressing this issue. Thus, to build collaborative, user-friendly, trainable robots, we require a method of deriving hierarchical task structure. We present an approach that autonomously does so by leveraging ordering constraints and contextual knowledge from previously known tasks to learn multiple levels of abstraction for arbitrary task networks. We hold a minimal set of assumptions about skill representations, allowing our work to be applied even to systems trained solely with skills that lack rigidly defined preconditions or effects (or other such formalization). The primary contributions of this work are:

- Clique/Chain Hierarchical Task Networks (CC-HTNs), a new type of HTN that encapsulates topological properties of a task's action space.
- An algorithm for autonomously constructing CC-HTNs from task graphs
- Evaluations of CC-HTNs within state estimation and knowledge transfer for reinforcement learning domains

*This work supported by ONR Grant #N00014-12-1-0822, "Social Hierarchical Learning"

[1]Department of Computer Science, Yale University, 51 Prospect St, New Haven, CT, USA
{bradley.h.hayes,brian.scassellati}@yale.edu

II. Background and Related Work

Our contribution is targeted towards enabling robots to model relationships between skills to facilitate high level task reasoning. We provide a brief survey of Learning from Demonstration, Markov Decision Processes, and Hierarchical Task Networks to familiarize the reader with the techniques that form the basis of our approach.

A. Learning from Demonstration

The acquisition of skills and task solutions from novice users via demonstration, Learning from Demonstration (LfD), is widely considered to be a simple and natural method of robot training [7]. LfD is particularly useful due to its vast applicability, as demonstrations can be effectively provided via teleoperation [8], observation [9], and kinesthetic teaching [10] to achieve task or skill proficiency. Recent work has extended skill generalization within LfD, a highly desirable property, by leveraging Bayesian non-parametric models and dynamic movement primitives to extract flexible skills from unstructured demonstrations [11]. This represents one potential opportunity for applying our presented work, providing deeper task-level structure that may not otherwise be emergent from the segmented trajectories produced by similar approaches. Notably, autonomously obtaining complete and relevant symbolic groundings for LfD-acquired skills is difficult, making such skills ineligible for proper inclusion into traditional symbolic planners. Regardless, even if all effects or object properties relevant to a motor primitive cannot be precisely specified, these skills can still be utilized within policy-based learning techniques, a primary motivation for our work. Our algorithm contributes toward bridging this gap between symbolic abstraction and low-level policy learning.

B. Markov Decision Processes

Graphical task representations are desirable for their expressivity. Markov Decision Processes (MDPs) provide a convenient and efficient framework for planning.

The MDP framework, represented by the 4-tuple (S, A, R, P), defines an environment-centric directed multigraph with environmental knowledge encoded in vertices representing possible states ($s \in S$). Directed edges between vertices (transitions) are labeled with an action ($a \in A$). $R(s'|s, a)$ is the reward achieved by transitioning to state s' from state s via action a. $P(s'|s, a)$ characterizes the dynamics of the system, indicating the probability of arriving in state s' from state s when executing action a.

Semi-MDPs (SMDPs) are a variant that permits the creation of flexible, arbitrarily complex options: closed-loop policies describing action sequences [12]. The temporal abstraction (e.g., allowing actions have differing durations) and generality of representation inherent to SMDPs make them a popular method of representing knowledge about actionable skills and tasks. As such, we represent tasks as SMDPs throughout this work.

C. Hierarchical Task Networks

Hierarchical representations, particularly Hierarchical Task Networks (HTNs) [13], have been developed to manage the exponential complexity growth of realistic planning problems [14]. This highly effective approach consists of identifying abstractions present within collections of primitive skills or subtasks to reduce planning difficulty [15]. Primitives are typically represented in a manner reminiscient of STRIPS [16], containing symbolic representations of environmental preconditions and effects.

HTNs scale more readily and are more expressive than STRIPS-style planners [17]. HTN planners have been successful in a wide variety of rich and complex problem domains, including multi-agent team coordination in robot soccer [18] and mobile robot exploration [19]. As such, they are highly desirable task representations that serve a multitude of important uses. In particular, AND/OR graphs [20] (a type of HTN) have emerged as a popular choice within multi-agent assembly domains [21], [22].

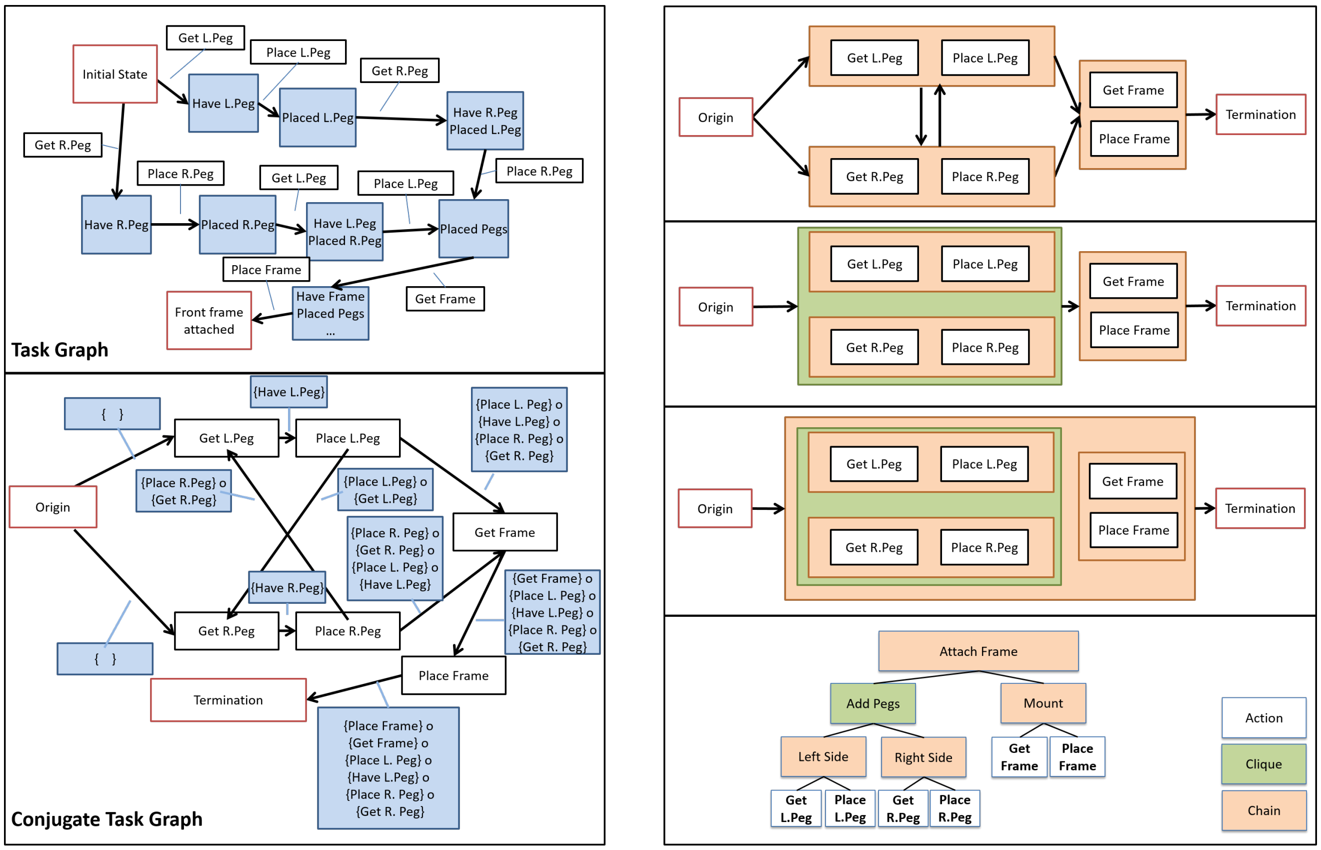
However, these hierarchical networks are typically assumed to be manually specified or assume the availability of rich domain knowledge and planning operator (action) descriptions to facilitate their discovery [23], [24]. We present a novel bottom-up approach to HTN generation without prior manual intervention, building on the causality analysis principles of Nehta et. al [25] and subgoal discovery of Menache et. al [26] to find and abstract flexible subtasks.

III. Constructing Task Hierarchies

For an arbitrary task network, we present a general method that builds hierarchical abstraction without manual intervention or reliance on particular symbolic representations.

A. Constraint Analysis

As the learner discovers successful action sequences or invalid state transitions within a task, insight into the structure of the task being performed increases. Given a graph $G = \{V, E\}$, with vertices corresponding to environment states and directed edges indicating available transitions between them, we define a task execution x as a directed graph induced by the path on G that is followed during the execution of a particular valid sequence of motor primitives. We define the set X as that consisting of all possible, efficient (having no superfluous actions) task execution paths that terminate in success (i.e., assumptions similar to [25]). Thus, $x = \{V_x, E_x\} \in X$ describes a sequence of actions, or in other words a graph formed from a single path through a set of vertices in $V_x \subseteq V$ connected via edges in $E_x \subseteq E$ that are labeled with actions $a \in A$, where A is a dictionary of known primitive skills (options). Learning from Demonstration provides a means to acquire these x quickly, via instructor-provided examples. Alternatively, these x can be generated by an automated planner with some degree of stochasticity or exploration. More plainly, each graph x is a chain of vertices linked by actions that results in task completion. This forms a subgraph in the complete task space defined by all possible states and action connections,



(a) Task graph (top) and its conjugate (bottom). In the task graph, environment states are encoded in vertices and edges are labeled with their corresponding actions. In the conjugate, subgoals are represented as vertices and edges are labeled with environmental prerequisites.

(b) Incremental execution of Algorithm 2, from CTG (top) to CC-HTN (bottom). For clarity, we omit edge prerequisite conditions. The internal node labels on the final CC-HTN were manually added for clarity, illustrating the implied logic behind the action grouping.

Fig. 2: The graph types used for constructing CC-HTNs, applied to part of a furniture assembly task from Figure 1.

originating from a vertex that is an initiation state of G and terminating in a goal state of G.

We define the task structure as being knowable when the learner's task graph can be reduced to a subgraph containing only the vertices and edges of successful (or in a planner's case, satisfying) task executions. Given each possible task solution $x_i \in X$, we define a task being iteratively learned by the robot as the weakly connected directed multigraph $T = \bigcup_{i=1}^{|X|} x_i$, describing a Markovian graph consisting of all known valid task execution paths. This construction is especially useful when using low-repetition, demonstration-based training common to deployed robotics scenarios.

In entirely demonstration or observation driven scenarios, our approach imposes the requirement of autonomous task failure detection, which can be difficult in LfD-driven, real-world scenarios. As a consequence of this, we qualify our approach as 'weakly supervised' for that particular domain, while remaining fully autonomous for more well specified applications such as symbolic planning.

B. *Task Encoding and Action Representation*

For generality, we encode tasks with the traditional SMDP representation $\{S, A, R, P\}$. Due to the lack of assumptions regarding symbolic skill knowledge, we represent environment states as compositions of functions $f_{a_n} \circ f_{a_{n-1}} \circ \ldots \circ f_{a_1}(x)$, where x is the initiation state at the time of execution and $f_{a_i} : S \rightarrow S$ represents the effect that a_i has on the environment. This function composition serves to map environment states satisfying preconditions of a_1 to environment states satisfying postconditions of a_n. Each of these f_{a_i} is taken directly from the action sequence followed by the agent through the state space to reach the current state. We assume the capability to identify commutativity between f_{a_i}, as these compositions will result in equivalent final states.

C. *Conjugate Task Graph*

We desire a representation that reveals easily exploitable relationships between skills to help discover logical groupings of actions to abstract into subtasks, free of execution context. In providing this abstraction, we sacrifice optimality in favor of tractability. Environment-centric representations of Markovian graphs (such as SMDPs) do not readily convey features facilitating the discovery of a task's underlying structure. To overcome this, we augment the task graph via a transformation function (Algorithm 1) to produce a

Algorithm 1: Conjugate Task Graph Transform

```
Input: Markovian Graph $G = \{V, E\}$
Output: Conjugate Task Graph $C$
1  $C \leftarrow$ empty Conjugate Task Graph {W,F};
2  origin_vertex $\leftarrow$ new empty vertex;
3  terminal_vertex $\leftarrow$ new empty vertex;
4  Add origin_vertex and terminal_vertex to $W$;
5  foreach unique $a \in \{action(e) | \forall e \in E\}$ do
6      Add vertex {action = $a$} to $W$;
7  foreach edge $e \in E$ do
8      if from$(e) \in initiation\_states(G)$ then
9          Add edges {from: origin_vertex, to: $[v \in W |$
           $action(v) = action(e)]$, prerequisites: $\emptyset$} to $F$;
10     if to$(e) \in termination\_states(G)$ then
11         Add edges {from: $[v \in W |$
           $action(v) = action(e)]$, to: terminal_vertex,
           prerequisites: environment_state(to($e$))} to $F$;
12     else
13         foreach edge $f \in outbound\_edges(to(e))$ do
14             Add edges {from: [vertex $v \in W |$
               $action(v) = action(e)]$, to: $[u \in W |$
               $action(u) = action(f)]$, prerequisites:
               environment_state(to($e$))};
15 foreach $\{a, b \in F \mid a \neq b\}$ do
16     if from(a) = from(b) and to(a) = to(b) and
       prerequisites(a) $\subseteq$ prerequisites(b) then delete $b$;
```

constraint network from its conjugate, resulting in a graphical representation with parameterized actions encoded in vertices as subgoals and environmental prerequisites encoded on its edges as compositions of subgoal completions. We refer to this graph as the *Conjugate Task Graph (CTG)* (Figure 2a).

To create a CTG, we follow the procedure described in Algorithm 1. As input, we require a sequential manipulation problem represented as a task graph (obtainable by teleoperation, demonstration, or exploration), with options specified at the level of subgoals. Adapting arbitrary primitive actions to serve as subgoal specifications may involve utilizing a HI-MAT hierarchy [25] or learning options that achieve the 'bottleneck states' discovered via a state space segmentation algorithm like Q-Cut [26] to derive higher-level grounded options. In the IKEA assembly domain, these options are shown in Figure 2a.

Recall that the goal of this algorithm is to create a graph where subgoals are represented as vertices and environmental constraints are represented on edges.

In lines 8-11, the initiation and goal sets are transformed into the new representation. For any task graph edge e originating at an initiation state, a corresponding edge is created in the conjugate graph connecting the $origin\_vertex$ to the vertex labeled with the action represented in e. Similarly, for any graph edge e terminating at a goal state, a corresponding edge is made connecting the conjugate vertex with the same action label to the graph's $terminal\_vertex$.

Algorithm 2: Construct CC-HTN

```
Input: Conjugate-Task-Graph $G$
Output: CC-HTN $H$
1  $H \leftarrow$ Copy($G$)
2  while $|H_{vertices}| > 1$ do
3      $h\_size \leftarrow |H_{vertices}|$
4      foreach maximal clique $c = \{V, E\} \in H$ do
5          Compact $\{V \in c\}$ into single metanode $m$
6      foreach maximal chain $c = \{V, E\} \in H$ do
7          Compact $\{V \in c\}$ into single metanode $m$
8      if $h\_size == |H_{vertices}|$ then break;
```

In lines 13-14, the internal edges of the conjugate graph are populated by looking at pairs of edges (e, f) that share a common vertex in the original task graph. Edges are added to the conjugate graph from the vertex matching the action on e to the vertex matching the action on f. Finally, in lines 15-16 we remove redundant edges in the graph, contributing towards the HTN only enabling plans with forward justified steps [27].

D. CC-HTN Generation

In this section we introduce the Clique/Chain Hierarchical Task Network (*CC-HTN*), a novel HTN based on subtask ordering constraints. Our method draws its utility from the observation that the basis of a task network's structure is embedded in the restrictions placed on allowable permutations of subgoal sequences during execution. These restrictions can be characterized as independently applying to subsequences of goals or motor primitives in the task's hierarchy, indicating ordering constraints on a subtask. Our approach to hierarchy construction classifies subgraphs into two fundamental types: unordered sequences (cliques) and ordered sequences (chains).

Composing the CTG into collections of ordered and unordered subsequences provides a mechanism by which a task hierarchy can be derived solely from known transitions. To finish the process of converting a task graph into a CC-HTN (Algorithm 2), we perform a series of alternating, contracting graph operations on the CTG. These operations are performed until either the task is condensed into a single vertex, indicating a successful and complete hierarchical abstraction, or until the graph does not change as a result of the actions, indicating that the graph is either incomplete or cannot be condensed further. This process is visually represented in Figure 2b, with failure contingencies explained in the following section.

A *clique* in a CTG has at least one set of inbound edges such that one member edge terminates at each internal vertex of the clique. These edges must all share a common prerequisite list on their label. There must also exist a set of edges originating within the clique, with origins at each clique member, terminating at the same external node. These

Algorithm 3: Resolve Chain Ambiguity

```
Input: Conjugate Task Graph $G = \{V, E\}$
Output: Conjugate Task Graph $H = \{W, F\}$ or 'No ambiguity'
1  $C_G \leftarrow$ all sets of commutative subgoals in $G$;
2  $C_E \leftarrow$ all edges between subgoals in $C_G$;
3  $W \leftarrow V$;
4  $F \leftarrow E \setminus C_E$;
5  $new\_chains \leftarrow []$;
6  foreach maximal chain $c = \{X, I\} \in H$ do
7      Compact $\{X, I\} \in c$ into single chain metanode $m$;
8      $new\_chains.append(m)$
9  if $|new\_chains| == 0$ then return 'No ambiguity';
10 foreach Edge $e \in C_E$ do
11     if $to(e)$ is the head of a chain $c \in new\_chains$ then
12         $e.prerequisites.append($ Members of $c$ not in e.prerequisites);
13         $F = F \bigcup e$;
14 Remove redundant edges from $F$;
```

Algorithm 4: Resolve Clique Ambiguity

```
Input: Conjugate Task Graph $G = \{V, E\}$,
Edge set $Z$ = {Known subgoal transitions}
Output: Conjugate Task Graph $H = \{W, F\}$ or 'No ambiguity'
1  $H \leftarrow Copy(G)$;
2  $J = \{X, I\} \leftarrow$ Resolve-Chain-Ambiguity(G);
3  $C \leftarrow$ {set of chains in $J$} $\setminus$ {set of chains in $G$};
4  $Q \leftarrow$ Map {v: {subgoals commutative with $v \in W$}};
5  foreach Chain $c \in C$ do
6      foreach Edge $e \in$ Chain $c$ do
7          if ($to(e) \in c$ and $from(e) \in c$) then
               $F \leftarrow F \setminus e$;
8              $prerequisites(e) \leftarrow prerequisites(e) \cup Q[from(e)]$;
9              if (modified $e \notin Z$) then return 'No ambiguity';
10             $F \leftarrow F \bigcup$ {modified $e$};
11 $new\_cliques \leftarrow []$;
12 foreach maximal clique $q = \{X, I\} \in H$ do
13     Compact $\{X \in q\}$ into single metanode $m$;
14     $new\_chains.append(m)$;
15 if $|new\_cliques| == 0$ then return 'No ambiguity';
```

edges share the requirement that each edge's label must minimally match the postcondition set resulting from the execution of all skills represented within the clique. Due to the construction of the graph, it can be inferred that the environment-modifying functions of skills within a clique are commutative with respect to successful task completion. This can be seen in the "Add Pegs" node (Figure 2b), as the order doesn't matter when placing the left and right peg.

A *chain* is defined as a path of vertices fulfilling special connection criteria. A chain has a starting vertex with out-degree one and a termination vertex with in-degree one. All intermediate members of a chain must be on a path between the starting and ending vertex, with an in-degree and out-degree of one. This occurs in the "Mount" node (Figure 2b), as getting then placing the frame must occur in sequence.

By iteratively finding and replacing maximal chains and maximal cliques in the CTG with meta-vertices until either a single vertex remains or the graph remains unchanged, a hierarchical structure emerges from known (or hypothesized) graph transitions (Figure 2b). Hierarchies generated by this method have the benefit of being human interpretable (given sensibly named motor primitives).

E. Hierarchical Ambiguity

Three conditions exist in which a task does not cleanly abstract into a hierarchical structure with a single root node.

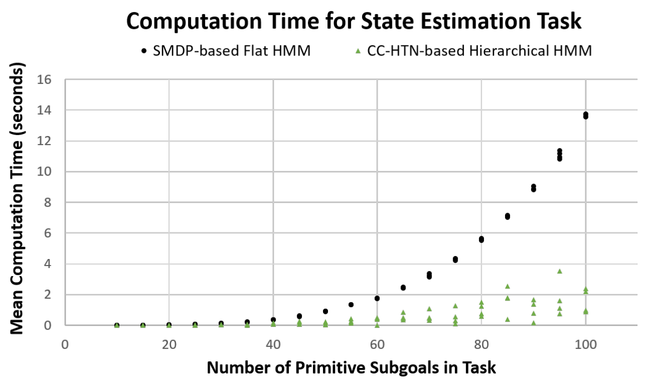
1) Redundancy or Equivalence: A maximally abstracted hierarchy cannot be guaranteed with our algorithm if actions tangential to the goal of the task are included. Any unnecessary actions must be identified and removed from the task graph. Similarly, if two or more skills perform equivalent functions from a goal fulfillment perspective, they must be classified as instantiations of the same skill (e.g., pressing a button with the top or side of a manipulator).

2) Incomplete Graph: If the task graph does not fully identify its transition dynamics, only a partial hierarchy can be provided. Autonomous exploration or active learning can be used to discover these ordering constraints [28]. Algorithms 1 and 2 can be applied online, providing more structure as new demonstrations are observed and the task graph's topology is discovered.

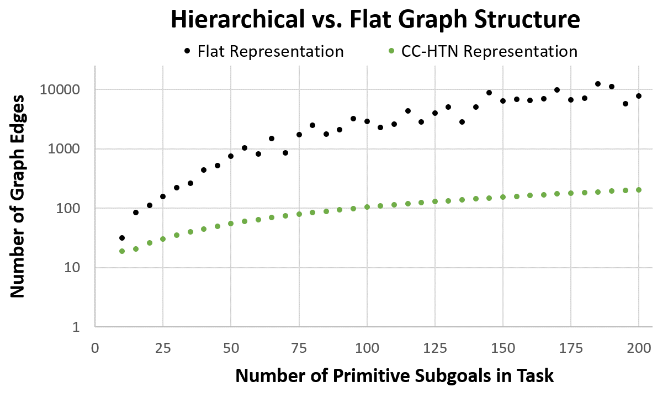
3) Multiple Valid Alternatives: Many subtasks have multiple valid constraint-based hierarchical representations. One such example is the task of placing two glasses on a table then filling them. If the possible actions were {Place Glass 1, Place Glass 2, Fill Glass 1, Fill Glass 2}, it is equally plausible that this can be abstracted to {Place Glasses, Fill Glasses} or {Place/Fill Glass 1, Place/Fill Glass 2}. These situations present themselves as vertices in the CTG that are both part of cliques and chains at the same level of abstraction. In practice, both hierarchies should be kept at each possible fork, with the final hierarchy decided by predicted execution policy reward (given the available agents, resources, etc.). In the event that Algorithm 2 does not converge to a single vertex, Algorithms 3 and 4 can be run afterwards to extract these alternative hierarchical representations if they exist.

F. Algorithm Runtime Analysis

The dominating complexity factor throughout these algorithms is the maximal clique detection within the Conjugate-Task-Graph to HTN Transform (Algorithm 2) and clique ambiguity resolution (Algorithm 4), an NP-hard problem. This step can be accomplished in time $O(d|V|3^{d/3})$ using the Bron-Kerbosch algorithm [29], where d is the degeneracy of the graph and $|V|$ is the number of graph vertices,



(a) Mean computation times for estimating an agent's last completed subgoal using HMMs, as a function of task complexity. Timings were measured using a single thread of an Intel i7-3930K CPU. The responsiveness required in collaborative task execution mandates high frequency state estimation capabilities. In most cases, the CC-HTN model completed its computation faster than 1Hz.



(b) Average number of edges in generated sequential manipulation tasks as a function of subgoal count. The CC-HTN encapsulates much of the transition complexity in the task, causing a dramatic reduction in the amount of edges required to represent task substructure. This simplification of task dynamics allows for more rapid computations in role selection and intention recognition.

Fig. 3: Performance metrics evaluating our hierarchical approach in a task state estimation problem.

on a simplified (undirected) CTG to find maximal clique candidates. These candidates are then kept or eliminated based upon verification that edge prerequisite conditions are met in the original CTG. The process of finding chains is linear in the number of vertices within the graph, while the conjugate graph transform (Algorithm 1) and chain ambiguity resolution (Algorithm 3) run in time that is polynomial in the number of observed vertices, edges, and task actions.

In practice, a large number of conjugate task graphs in the sequential manipulation domain will have low degeneracy (indicating a majority of subtasks being ordered) compared to the number of vertices within them (total number of subgoals), which drives down the cost of clique detection.

IV. Applications and Evaluation

Two important applications of CC-HTNs are in goal inference and task planning. The challenge of interpreting another's actions to derive potential goals is critically important within human-robot collaboration. Intention recognition enables an agent to vastly expand its planning horizons and narrow its belief space about its teammates.

An agent that can rapidly re-plan task solutions to accommodate and work around the actions of its collaborators is far more valuable than one who cannot. As such, we conclude with an analysis of the CC-HTN's ability to accelerate task planning solutions by leveraging abstractions from prior experiences. This performance boost is obtained by employing a simple transfer of knowledge across tasks: internal nodes of CC-HTNs (each representing multiple motor primitives) are added to the agent's action dictionary, allowing for a planner to use larger building blocks in its search.

A. *State Estimation*

To showcase the effectiveness of the autonomously constructed CC-HTNs, we demonstrate their performance first within an online goal estimation task. In human-robot collaboration, there will always exist one or more agents that cannot be controlled by a central planner and cannot be assumed to regularly report their status explicitly. Thus, it is imperative that agents be able to infer each other's goals (e.g., "Agent 1 wants to attach the seat of the chair").

To perform this goal inference, we create hierarchical Hidden Markov Models that we assemble from CC-HTNs (created by Algorithm 2). As (hierarchical) HMMs have been shown to be a popular and effective approach for action and goal recognition [4], [30], we test the effectiveness of our hierarchies at segmenting tasks within this domain. HMM states represent task subgoals, while state transition probabilities are uniformly distributed across valid transitions within the HTN to create a naïve initial policy prior. We use Semantic Event Chains [31] to define the observation sets for each action within the HTN, with a naïve Bayes assumption of observation independence. Semantic Event Chains model actions as sequences of object-object and agent-object contact/non-contact events, thus the observation sets at each state in the HMM are these contact events. Combining this action representation with an HMM provides a probabilistic model for predicting the goals of an observed agent. The Semantic Event Chain action representation is conveniently independent from agent kinematics, and can be used for predicting both human and robot intent.

1) Task Definition: The tasks used in this evaluation were generated using encoded IKEA furniture assembly tasks, a domain previously explored in multi-agent collaborative scenarios [32], as a statistical basis for the task structure. These tasks had subtask commutativity/invariance, per-skill observation set overlap, and action set sizes sampled from distributions modeled after real assembly tasks. These scenarios serve as a convenient method of providing arbitrarily complex tasks within which to demonstrate the scaling capabilities of our work. Our dataset included 76 tasks, each with 4 randomly determined execution paths. These tasks ranged in size from 10 to 100 subgoals, consisting of parameterized pick, place, and fasten actions. Within the CC-HTNs, the average depth for a motor primitive ranged from 1.99 to 3.61, with a mean across all tasks of 2.84.

2) Results: We compare the state estimation performance of CC-HTN-based hierarchical HMMs against traditional HMMs built from the same task. This provides a measure of utility for our generated hierarchies, namely how well it segments and abstracts the problem space. We evaluated performance across two measures: state estimation computation time and graph size.

Collaborative task execution imposes strong real-time computation constraints. Hierarchical structures mitigate the growth of computational time requirements with task size. If the generated hierarchies were mostly flat or did not segment the task well, we would expect the hierarchical HMM and flat HMM performance curves to be similar. As our computational performance results show (Figure 3a), CC-HTNs provide considerable computational benefits over flat task models despite being built from the same base representation. These outcomes strongly imply a useful abstractions were created. In practice, hierarchical state estimation accuracy benefits will be dependent on observation set overlap between subtasks, however for cases where there is suitable distinction, these observed computational benefits apply.

We also evaluate the representation of the hierarchical representation against that of the flat task graph (Figure 3b). The primary comptuational advantage of our approach is derived from the simplified structure afforded by encoding child ordering constraints within parent vertices. In particular for clique substructures, this reduces the number of edges required to represent subgoal relationships from being polynomial in the number of involved subgoals ($|E| = |V|^2 - |V|$) to linear ($|E| = |V|$). For each incoming and outgoing clique transition, edge count is reduced from linear in the number of clique members ($|E| = |V|$) to constant ($|E| = 1$).

Having a hierarchical task representation allows for state estimation at multiple resolutions. In the case of assembling an IKEA chair, a predicted internal node of the hierarchical structure may represent the goal of “attach rear left leg to chair”, encompassing all of its child primitives, providing broader context than merely identifying an agent attempting to “get wooden peg”. Being able to identify higher level goals provides essential context when providing assistance or planning one’s own actions. In many cases it is more important to recognize these abstract goals than the motor primitives themselves.

B. *Task Planning and Transfer Learning*

In addition to state estimation and intention prediction, the presented approach has utility within task planning, allowing an agent to leverage prior experiences (as macro actions) to learn more quickly in new contexts.

The ability to plan with high level actions acquired through normal operation is tremendously valuable for a collaborative robot. These meta-actions allow a task planner to take larger steps towards a goal state, acting as a valuable heuristic that

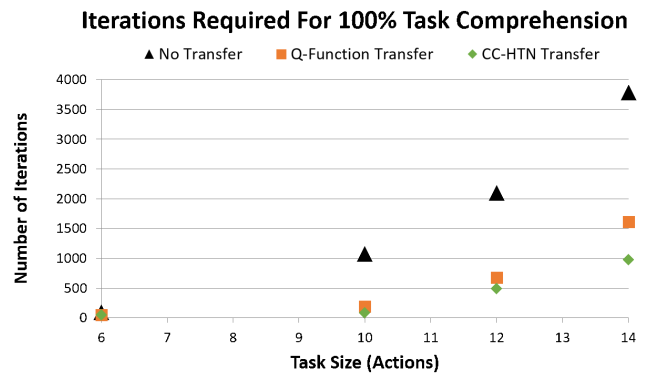


Fig. 4: Task MDP discovery results on a dataset of 25 food preparation tasks with a primitive action set size of 39. Q-function transfer aggregates Q-functions from known task MDPs to bias action selection during exploration. CC-HTN Transfer uses Q-function transfer with macro actions learned from the CC-HTNs of other tasks in the data set.

removes multiple levels of depth from the plan search each time one is utilized. Additionally, for applications where a human user is assisting the task planner by specifying action sequences manually, our abstractions can greatly reduce the effort and time required to formulate a solution. As our abstractions readily indicate intra-task dependencies and can identify parallel subtasks through comparisons of required resources at each subgoal, we also obtain benefits seen in AND/OR graph constructions for assembly tasks [22].

More generally, given a task with an action set of size $|A|$ and an optimal solution length of d, a standard breadth-first search will have complexity $\mathcal{O}(|A|^d)$. With an abstraction strategy that adds *useful actions* and follows the Downward Refinement Property [27], the average search complexity will improve exponentially with the layers of abstraction provided. Given k levels of abstraction, the average complexity reduces to $\mathcal{O}([k|A|]^{d/k})$. The addition of modern heuristics (such as FFRob [33]) can be expected to provide additional improvement. Thus, given the ability to determine topically similar tasks, the burdens incurred by increasing the size of the action dictionary are more than compensated for by reducing the necessary search depth. This strategy is particularly effective for robots designed to perform tasks that originate from the same domain (such as a cooking robot or a particular type of product assembly machine), but will decrease in effectiveness as the task corpus diverges.

We conclude with results comparing three approaches to a task exploration problem to quantify our contribution’s transfer learning benefits for real tasks (Figure 4). Using a data set of 25 food preparation tasks of between 6 and 14 steps encoded as SMDPs ($|A| = 39$), we measure the number of iterations required for an ϵ-greedy exploration policy to capture all structural insight (valid subgoal orderings) of a task. The macro actions produced by our algorithm achieved full task comprehension faster than either random exploration or flat Q-function transfer strategies in every trial. This indicates that autonomously built CC-HTNs are more effective than the implicit subsequence clustering of

Q-function transfer, demonstrating deeper structural insight.

V. Conclusion

In this work we introduced CC-HTNs, and provided a novel, bottom-up approach for autonomously deriving hierarchical task structure from graphical task representations, using SMDPs as a representative example. In doing so, we introduce the Conjugate Task Graph: a task representation with graphical properties that facilitate the identification of underlying structure. The approach we present is widely applicable, depending only upon the availability of primitive skill classification and ordering constraint knowledge within the target task. To make our contribution even further accessible, we identify and provide solutions to instances where ambiguity may arise in building the CC-HTN.

We evaluate our work in domains deeply relevant to human-robot teaming: collaborator goal inference via state estimation and task learning. Our results suggest that CC-HTNs are useful in segmenting complex tasks, reducing computation time and enabling inference at multiple levels of abstraction. Our macro actions find further utility in transfer learning settings, accelerating the comprehension of new tasks and enhancing the value of prior training data. Finally, our results demonstrate that CC-HTNs can reduce the average search complexity for agents that operate in use cases where task subgoals may be shared across activities.

References

[1] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning*, 2000, pp. 663–670.

[2] Q. Yang, "Formalizing planning knowledge for hierarchical planning," *Computational intelligence*, vol. 6, no. 1, pp. 12–24, 1990.

[3] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Constructing symbolic representations for high-level planning," in *AAAI Conference on Artificial Intelligence*, 2014.

[4] B. Hayes and B. Scassellati, "Effective robot teammate behaviors for supporting sequential manipulation tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015.

[5] P. Stone and M. Veloso, "Task decomposition and dynamic role assignment for real-time strategic teamwork," *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pp. 293–308, 1999.

[6] J. Shah, J. Wiken, B. Williams, and C. Breazeal, "Improved human-robot team performance using chaski, a human-inspired plan execution system," in *Proceedings of the 6th international conference on Human-robot interaction*. ACM, 2011, pp. 29–36.

[7] J. Chen and A. Zelinsky, "Programming by demonstration: Coping with suboptimal teaching actions," *The International Journal of Robotics Research*, vol. 22, no. 5, pp. 299–319, 2003.

[8] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[9] G. Konidaris, S. Kuindersma, R. A. Grupen, and A. G. Barto, "Autonomous skill acquisition on a mobile manipulator." in *AAAI Conference on Artificial Intelligence*, 2011.

[10] B. Akgun, M. Cakmak, J. Yoo, and A. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the 7th annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2012, pp. 391–398.

[11] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *The International Journal of Robotics Research*, 2014.

[12] R. Sutton, D. Precup, S. Singh, *et al.*, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[13] A. Tate, "Generating project networks," in *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc., 1977, pp. 888–893.

[14] G. Neumann and W. Maass, "Learning complex motions by sequencing simpler motion templates," in *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[15] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "Shop2: An htn planning system," *J. Artif. Intell. Res.(JAIR)*, vol. 20, pp. 379–404, 2003.

[16] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3, pp. 189–208, 1972.

[17] K. Erol, J. Hendler, and D. S. Nau, "Htn planning: Complexity and expressivity," in *AAAI*, vol. 94, 1994, pp. 1123–1128.

[18] O. Obst, "Using a planner for coordination of multiagent team behavior," in *Programming Multi-Agent Systems*. Springer, 2006, pp. 90–100.

[19] L.-J. Lin, "Hierarchical learning of robot skills by reinforcement," in *IEEE International Conference on Neural Networks*. IEEE, 1993, pp. 181–186.

[20] B. R. Fox and K. Kempf, "Opportunistic scheduling for robotic assembly," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 880–889.

[21] L. S. Homem de Mello and A. C. Sanderson, "And/or graph representation of assembly plans," *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 2, pp. 188–199, 1990.

[22] R. A. Knepper, D. Ahuja, G. Lalonde, and D. Rus, "Distributed assembly with and/or graphs," in *Proceedings of the Workshop on AI Robotics at the International Conference on Intelligent Robots and Systems*, 2014.

[23] N. Nejati, P. Langley, and T. Konik, "Learning hierarchical task networks by observation," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 665–672.

[24] A. Mohseni-Kabir, C. Rich, S. Chernova, C. L. Sidner, and D. Miller, "Interactive hierarchical task learning from a single demonstration," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2015, pp. 205–212.

[25] N. Mehta, S. Ray, P. Tadepalli, and T. Dietterich, "Automatic discovery and transfer of maxq hierarchies," *Proceedings of the 25th International Conference on Machine Learning*, pp. 648–655, 2008.

[26] I. Menache, S. Mannor, and N. Shimkin, "Q-cutdynamic discovery of sub-goals in reinforcement learning," in *Machine Learning: ECML 2002*. Springer, 2002, pp. 295–306.

[27] F. Bacchus and Q. Yang, "The downward refinement property," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, 1991, pp. 286–293.

[28] B. Hayes and B. Scassellati, "Discovering task constraints through observation and active learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[29] D. Eppstein and D. Strash, "Listing all maximal cliques in large sparse real-world graphs," in *Experimental Algorithms*. Springer, 2011, pp. 364–375.

[30] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. Bui, "Learning and detecting activities from movement trajectories using the hierarchical hidden markov model," in *CVPR 2005*, vol. 2. IEEE, 2005, pp. 955–960.

[31] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object–action relations by observation," *The International Journal of Robotics Research*, 2011.

[32] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 855–862.

[33] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: An efficient heuristic for task and motion planning," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.