On Task Recognition and Generalization in Long-Term Robot Teaching

(Extended Abstract)

Guglielmo Gemignani*
Sapienza University of Rome
gemignani@dis.uniroma1.it

Steven D. Klee*
Carnegie Mellon University
sdklee@cmu.edu

Manuela Veloso
Carnegie Mellon University
veloso@cs.cmu.edu

Daniele Nardi
Sapienza University of Rome
nardi@dis.uniroma1.it

ABSTRACT

Several research efforts address the challenge of having users incrementally teach or demonstrate a task to a robot. We are interested in an autonomous robot that persists over time and the problem of teaching it an *additional* task. We believe that the assumption that a user would know all the tasks previously taught to the robot does not hold. We hence investigate the problem of recognizing when a user is teaching a task similar to one the robot already knows and performing task autocompletion. In this extended abstract, we briefly discuss our approach, and report the results of an experiment run with human teachers.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms

Keywords

Robot Planning and Plan Execution, Robotic Agent Languages and Middleware for Robot Systems

1. INTRODUCTION

The CoBots are autonomous mobile service robots that operate in office settings [1]. Baxter is a manipulator robot used for factory automation, especially in environments with people. We use the Instruction Graph [3] framework to teach tasks to both robots through natural language interaction. This framework represents hierarchical tasks with conditionals and loops as a combination of robot actuation and sensing primitives, conditionals, and loop structures.

We are interested in the problem of teaching an *additional* task to these robots. Specifically, we investigate how task generalization and autocompletion can be applied to assist users who do not remember all of the tasks a robot has previously acquired. We first discuss our approach to generalize graph-based tasks with similar structures of robot-primitives, but possibly different parameters. Then, we explain our algorithm for task autocompletion.

First, we generalize over structurally similar tasks, containing the same primitives instantiated with different parameters. Since our task representation is graph-based, generalization requires us to find frequently occurring common subgraphs, which is NP-Hard. We relax the problem by creating a tree representation of each task. We then use frequent labeled subtree mining to extract generalized *parametric* tasks from the trees.

In future teaching episodes, the robot checks if the task it is learning is similar to a generalized task. If so, the robot proposes the remaining steps to the user, performing task autocompletion. In other domains, this form of autocompletion has helped users when they have trouble exactly expressing a command [5].

Some other work has been done to allow a user to teach complex tasks by combining pre-existing robot action primitives [4, 3]. As an alternative to generalization, some approaches [2] allow the user to specify parameterized tasks. However, they require the user to describe tasks at an abstract level. Instead, our approach creates parameterized tasks by learning from specific instances.

In the following sections we first briefly describe our approach to task generalization and autocompletion. Then, we report on an experiment run with human teachers.

2. APPROACH

Our tasks are represented as Instruction Graphs (IGs) [3]. Vertices represent action and sensing primitives of the robot, conditionals, or loops. Edges define the transitions between these vertices during execution. Each graph G is a tuple $\langle V, E\rangle$, where each vertex $v \in V$ is a tuple:

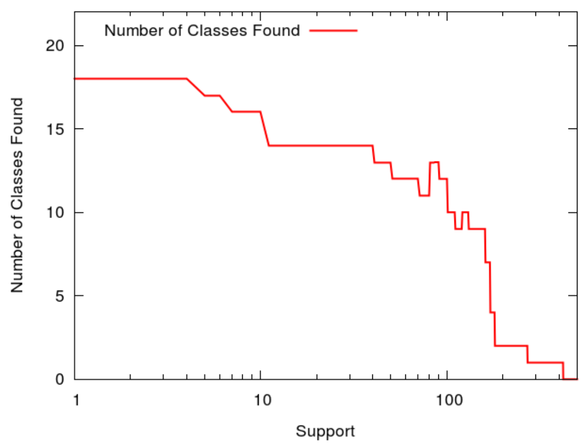
$$v = \langle id, InstructionType, f, P\rangle$$

where id and $InstructionType$ are parameters of the task execution framework. Together, the function f and the set of parameters P represent a robot primitive.

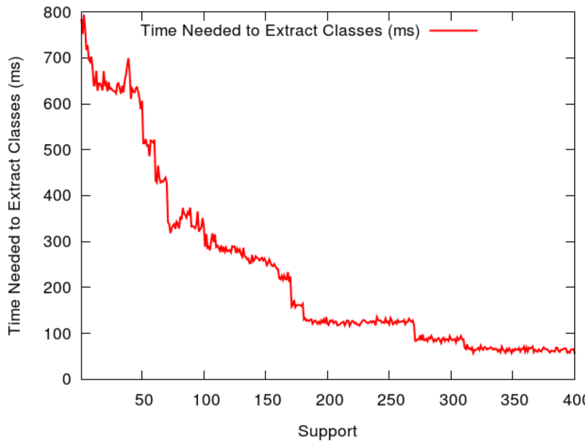
Our goal is to generalize a set of IGs to learn paramet-

*The first two authors contributed equally to this work.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*



(a)



(b)

Figure 1: (a) GIGs vs Support Threshold and (b) Time (ms) vs Support Threshold

ric tasks. We define these tasks as Generalized Instruction Graphs (GIGs). In GIGs, some parameters in P are *ungrounded*, with a type such as *Angle*, or *Distance*, but no value. Given a library of IGs, our goal is to extract a library of GIGs. We use this library to perform task autocompletion in future teaching episodes. Our approach consists of the following steps:

Frequent Subgraph Mining Since finding subgraph isomorphisms is NP-Hard, we first relax the problem by creating a labeled spanning tree for each IG. Each label corresponds to the vertex's robot-primitive and instruction type. We provide the trees to a frequent labeled subtree mining algorithm, which returns *tree patterns* with an occurrence frequency greater than a threshold. The occurrence frequency of a tree pattern is typically called the *support*. Each pattern corresponds to a subgraph that appears in multiple IGs, possibly with different robot primitive parameters.

General Task Creation Next, the tree patterns are filtered to eliminate any with corresponding IGs that are not executable. The remaining patterns are further filtered to remove any that are subtrees of one another. Without this filter, we note that for any tree pattern, all of its subtrees are also found because their support is at least as large. This leads to an explosion in the number of patterns found.

Finally, we create one GIG from each tree pattern, by determining which robot-primitive parameters should be instantiated, and which should be left ungrounded. A parameter is instantiated if it has the same value in a threshold percent of the IGs. Otherwise, it is left ungrounded, to be defined by the user at the task-teaching time.

Task Autocompletion During task teaching, the agent learns a task by building an IG through user interaction. After each interaction, the agent compares its *partial task* to the GIGs. If a predefined percentage of the two match, the robot requests permission to demonstrate the GIG. If the user accepts the suggestions, they are asked to instantiate any ungrounded parameters in the GIG. Finally, the instantiated GIG is appended to the partial task. If multiple GIGs match the same partial task, we break ties randomly.

3. SELECTED EXPERIMENTS

This task generalization and autocompletion approach has been run on a Baxter manipulator and a CoBot mobile base. Specifically the generalization algorithm has been applied to a library of 1000 tasks with 20 unique classes composed by 12 different primitives. The number of task examples for each class varied from 10 to 100. Users taught 2 to 10 tasks for each class, and the others were generated by taking the same structures and generating different parameters.

In this experiment we compared the number of GIGs found versus different minimum support thresholds. Figure 1 shows the results of the experiment. The algorithm extracted 18 out of 20 classes in less than 700ms on a quad-core intel i7 machine. Notably, for the optimal support, the algorithm was unable to generalize two classes, which were subclasses of other classes. This was an expected behavior due to the adopted filtering policy.

4. CONCLUSION

Our approach generalizes collections of tasks taught to a long-term deployed robot. In future teaching sessions, these tasks are proposed to the user to perform task autocompletion. This technique is scalable enough to run online with task libraries on the order of thousands of tasks in size.

5. REFERENCES

[1] J. Biswas and M. Veloso. Localization and navigation of the cobots over long-term deployments. *The International Journal of Robotics Research*, 2013.

[2] G. Gemignani, E. Bastianelli, and D. Nardi. Teaching robots parametrized executable plans through spoken interaction. In *AAMAS*, 2015.

[3] Ç. Meriçli, S. D. Klee, J. Paparian, and M. Veloso. An interactive approach for situated task specification through verbal instructions. In *AAMAS*, 2014.

[4] P. E. Rybski, K. Yoon, J. Stolarz, and M. Veloso. Interactive robot task training through dialog and demonstration. In *HRI*, 2007.

[5] D. Ward, J. Hahn, and K. Feist. Autocomplete as research tool: A study on providing search suggestions. *Information Technology and Libraries*, 2012.