Task Tree Retrieval from Cooking Recipes for Robotic Cooking

Kotha Shiva Krishna Reddy
Department of Computer Science and Engineering
University of South Florida
shivakrishnareddy@usf.edu

Abstract — **The development of search algorithms for Functional Object-Oriented Networks (FOON) plays a pivotal role in advancing robotics and automation systems. The goal of this paper is to generate a sequence of actions for a robot to prepare a recipe successfully using different search algorithms. In this project, two search algorithms were implemented for generating a task tree of preparing a goal object using a given set of ingredients, functional units, and utensils in a kitchen environment. To solve the problem of figuring out the ideal flow of events for object preparation, the project makes use of artificial intelligence ideas. The project's primary objective is to develop and compare two search algorithms: Iterative Deepening Search and Greedy Best-First Search. For the Greedy Best-First Search method, the project employs two heuristic functions. This report provides insights into the implementation of these search algorithms and discusses the results and the performance of each algorithm in finding optimal paths to prepare goal objects.**

I. INTRODUCTION

Robotic cooking has become a very promising area of the field of robotics, offering noteworthy benefits like convenience and the potential for increased efficiency and accuracy in meal preparation. Task planning is the most important aspect of automating kitchen duties properly. This involves generating a sequence of actions for robot to complete a specific goal. The growing significance of information representation in robotics and automation systems has led to the creation of search algorithms for Functional Object-Oriented Networks (FOON). A key tool in these fields is FOON, a graph-based framework that simulates how objects might be used to carry out tasks and operations. To achieve automated task execution, it is essential to extract task trees from FOON, hence the goal of this study is to discuss the contributions to the area made by the development of search algorithms that make this process easier.

In the modern world, automation is a major force that is revolutionizing everything from industry to healthcare. Autonomous task performance has the potential to improve effectiveness, lower errors, and boost productivity. Robots and autonomous systems must have a thorough understanding of effective task execution and object manipulation to enable automation.

Knowledge regarding the functional connections between objects and actions can be represented in a structured and methodical way using FOON. Robots and autonomous systems can plan and carry out tasks more effectively with the help of FOON, which builds a graph that contains this information. In response to the expanding demand for efficient and automated task planning and execution, search algorithms for FOON have been developed.

The motivation derives from the conviction that utilizing FOON and intelligent search algorithms would fundamentally alter how robots and autonomous systems communicate with their surroundings. By enabling higher levels of autonomy and adaptation, these algorithms can empower robots to handle complex tasks in diverse settings. Robots can more effectively complete jobs, develop task trees, and determine the best routes to goals by utilizing the comprehensive data in FOON. Knowledge network consists of ingredients, utensils and the steps involved in the recipe preparation.

By carefully choosing the best search algorithm, the intended goals are achieved while ensuring efficient resource utilisation. The number of steps, functional units, or utilities required to execute a given task is used to evaluate how effective the task tree creation is. The outcomes illustrate which search algorithms are more useful for assignments with increased cost-efficiency.

II. FOON CREATION

A. Functional Object-Oriented Network

FOON consists of two different kinds of nodes in it: motion nodes and object nodes. The preconditions and effect of actions are represented by the directed edges that connect these nodes. The functional unit, which represents a single action, is the essential building block of FOON. It has only one motion node, one or more output nodes, and one or more input nodes. The output nodes represent the outcome of the action, whereas the input nodes explain the required condition of the objects prior to the activity. The motion node represents the action. A thorough and vivid picture of the actions is provided by functional units. Figure 1 shows two functional units of slicing an onion and placing onion to cooking pan.

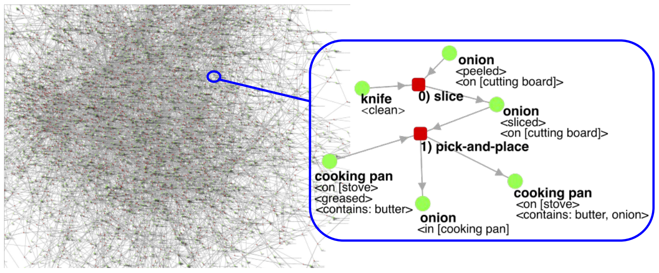

Fig. 1: Two functional units from FOON depicting slicing an onion and placing it to the cooking pan. Objects and motions nodes are denoted by green circle and red square respectively.

1) Functional Unit

In a FOON, a functional unit is a cohesive group of actions and objects that work together to complete a specific task. These functional units serve as the fundamental building blocks of the FOON knowledge representation, offering a structured manner to express the relationships between actions and objects within the context of various tasks or functions. For instance, a functional unit for making a cup of tea might include actions like "stirring" and items like "a clean spoon" and "a cup of unsweetened tea.

2) Nodes

The core components of a FOON are nodes. They represent either objects or actions and form the building blocks of the network. Because each node is a distinct entity that can be a component of one or more functional units, they can be employed repeatedly in various contexts. Typically, objects nodes represent physical objects like "a cup," "a spoon," or "tea”. Action nodes, on the other hand, depict actions or activities like "stirring" or "pouring." The nodes act as the vertices of the FOON graph and are connected by edges to show their relationships.

3) Edges

In a FOON, edges are the connections between nodes, and they show the relationships between these nodes. There are many other kinds of connections that can exist between objects, such as object-object connections, object-action connections, and more. For example, to show that the spoon is participating in the stirring process, an edge may link the action node "stirring" to the object node "a clean spoon". These edges play a crucial role in specifying how nodes interact and depend on one another, allowing the FOON to record how objects and actions are integrated to complete tasks.

4) Task Tree

A task tree is a hierarchical organization of functional units and their relationships that depicts the approach or series of steps necessary to finish a given task. It is a crucial component of FOON that shows how functional units are connected to one another to accomplish a higher-level objective. In a task tree, the root node often represents the goal or target and branches into other sub-functional units and those actions and objects that go along with them. Understanding the processes and dependencies necessary to complete difficult jobs is made easier with the help of the task tree. It allows for a clear and structured representation of how different functional units collaborate to achieve a specific function.

Functional Units are established to represent groups of activities and objects, Nodes are established to represent the discrete elements inside these units, and Edges are established to show the connections between nodes. These components are then arranged into Task Trees, which, by placing the functional units hierarchically, provide a structured representation of the method for carrying out a certain task or function. Together, these terminologies lay the groundwork for FOON's knowledge representation, allowing for the modelling of how various activities and functions in robotics and automation systems connect to objects and actions.

B. Task Tree Creation

During this project, received pre-generated task trees for specific tasks. To ensure their accuracy and completeness, conducted the following checks:

- Checked the task trees to ensure that they are related to the standard FOON structure, where each functional unit should have motion, input, and output nodes connected by edges. Any deviation from this structure was noted and corrected.
- Cross-referenced the task trees with the textual instructions or recipes that were provided. This meant making sure that the task tree and each step in the instructions matched up exactly.
- Examined the connections between object nodes and action nodes within the task trees to make sure that they correctly reflected the logic of the task. Any inconsistencies were identified and addressed.
- Observed the logical flow of the task trees to ensure that the sequence of functional units and their associated actions and objects made sense and aligned with the desired task. Any deviations from a logical sequence were corrected.
- Verified the task trees using the subject expertise to make sure they corresponded to actual task execution processes. Any inconsistencies were fixed.
- The task trees were represented visually using the FOON visualization tool, and it was confirmed that the representations accurately mirrored the task structure. This allowed for a more intuitive and visual validation process.

In conclusion, these checks included examining the structure, cross-verifying with text instructions, making sure that object-action relationships were consistent, evaluating logical flow, validating with domain knowledge, and using visual representations to visually verify the accuracy and completeness of the given task trees. The goal of this careful checking procedure was to make sure that the task trees accurately depicted the approach used to complete the tasks within the FOON framework and were error-free.

C. Task Trees Generated Visualization

Figure 2.1 shows the visualization of sweet potato FOON, in which there are 3 motion nodes that depicts 3 functional units peel, pick-and-place, and cut. It denotes all the input nodes, motion nodes and output nodes for the recipe sweet potato.

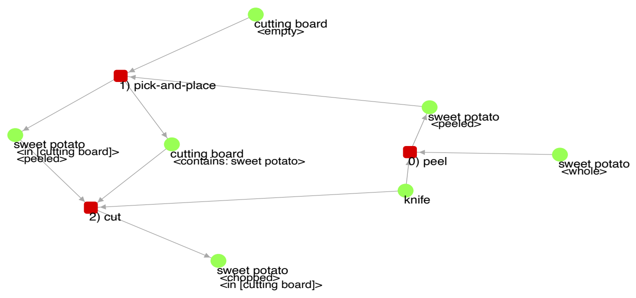

Fig. 2.1: FOON view of Sweet Potato task tree using IDS

As seen in the Fig. 2.1, the input nodes haves outgoing edges and output nodes have incoming edges. In the Fig. 2 Task tree for sweet potato, for first functional unit: whole sweet potato, knife are the input nodes, peel is the motion node and sweet potato peeled is the output node.

For second functional unit: peeled sweet potato, empty cutting board are input nodes, pick-and-place is the motion node and the peeled sweet potato in cutting board, cutting board having peeled sweet potato are output nodes.

And for the final third functional unit: output nodes of second functional unit along with knife are input nodes, cut is the motion node and the sweet potato chopped in the cutting board is the output node.

Similarly, the Fig. 2.2 Depicts the Functional Object-Oriented Network for the Greek Salad.

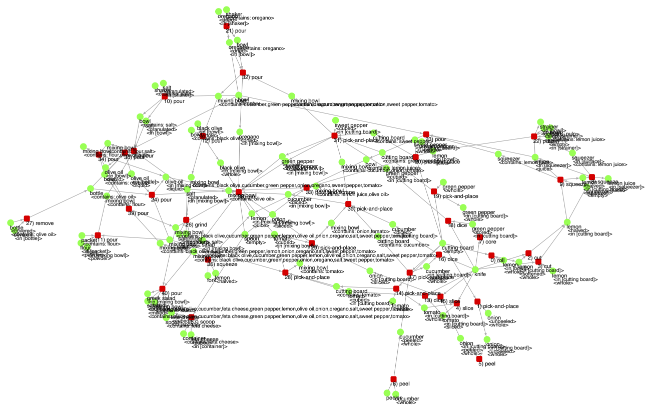

Fig. 2.2: FOON view of Greek Salad task tree using BFS_h1

Progress lines in Macaroni recipe

1. Water

off ---> turn on ---> Water(faucet) ---> fill ---> water (measuring cup) ---> pour ---> water(pot) ---> boil ---> boiling water(pot) ---> sprinkle ---> salted water(pot)

2. Macaroni

raw macaroni ---> pour ---> pot (macaroni, salt, water) ---> cook ---> cooked macaroni(pot)

III. METHODOLOGY

The development of search algorithms for FOON is critical to enable effective task planning and execution in robotics and automation systems. Iterative Deepening Search (IDS) and Greedy Best-First Search were both used to accomplish this. Additionally, developed two heuristic functions to guide the Greedy Best-First Search process. Additionally, two heuristic functions were created to direct the Greedy Best-First Search procedure. The goal is to create effective search algorithms that produce extremely precise and executable task trees for robotic activities.

1. Iterative Deepening Search (IDS)

Iterative Deepening Search is a depth-first search algorithm that systematically explores a FOON network to extract task trees. It is intended to find the best answer by gradually increasing the search depth.

IDS begins its search at a depth of 0. At the current depth level, it conducts a depth-limited depth-first search. In each iteration, IDS explores the FOON network up to a certain depth limit. After each depth-limited search, IDS determines if the desired goal object is present in the kitchen. A task tree is regarded as a solution and the search is considered to have succeeded if the objective item is located. If the target object cannot be located, IDS raises the depth limit and continues searching until a solution is found or the entire search space is explored. IDS returns the task tree associated with the first solution path it comes across.

The IDS algorithm is used to make sure that the search procedure is methodical and thorough, exploring every avenue while gradually deepening the search until a solution is identified. This approach is especially valuable when dealing with complex task trees and extensive FOON networks.

2. Breadth-First Search with Heuristic 1 (BFS with h1):

BFS with Heuristic 1 is a greedy search algorithm that prioritizes paths based on the success rate of the associated motion.

The search_BFS_heuristic1 function performs BFS with heuristic 1. It creates lists for objects to be searched (items_to_search) and items that have already been searched (items_already_searched).

The path with the best success rate is chosen after calculating the success rate of the motions connected to each path. Information about the success rate is taken from a given file. Moving towards the target object, the algorithm keeps exploring nodes based on motion success rates. When the goal item is attained, a task tree is returned.

3. Breadth-First Search with Heuristic 2 (BFS with h2):

BFS with Heuristic 2 is a greedy search algorithm that prioritizes paths based on the number of input objects required by a functional unit.

BFS with heuristic 2 is carried out using the search_BFS_heuristic2 function. Additionally, it keeps lists of both items that need to be searched and objects that have already been searched (items_already_searched and items_to_search). It determines the minimum amount of input objects each functional unit needs before choosing the best course of action. Based on this criterion, the algorithm keeps exploring nodes as it advances towards the target object. When the goal item is attained, a task tree is returned.

For extracting task trees from FOON networks, these approaches implement the search algorithms Iterative Deepening Search and Greedy Best-First Search. Greedy Best-First Search uses heuristic functions to direct the search process, whereas Iterative Deepening Search is a methodical strategy that systematically explores the search space. Based on motion success rates and input object efficiency, respectively, the two heuristic functions, h1(n) and h2(n), offer various viewpoints on path selection. Together, these algorithms and heuristics can effectively extract task trees, which helps automate and improve job planning and execution in robotics and automation systems.

IV. EXPERIMENTS AND RESULTS

Table [1]. shows the comparison of results of the task tree size for each goal node and with each search algorithm.

Goal Node	Search Algorithm	Task Tree Size
Whipped Cream	IDS	16
Whipped Cream	BFS_h1	16
Whipped Cream	BFS_h2	15
Greek Salad	IDS	37
Greek Salad	BFS_h1	41
Greek Salad	BFS_h2	41
Macaroni	IDS	9
Macaroni	BFS_h1	12
Macaroni	BFS_h2	9
Sweet Potato	IDS	3
Sweet Potato	BFS_h1	3
Sweet Potato	BFS_h2	3
Ice	IDS	1
Ice	BFS_h1	1
Ice	BFS_h2	1

Table [1]: Depicts the comparison of task tree size for each goal node with each search algorithm.

Goal Node: Whipped Cream

IDS: The task tree with 16 functional units was produced by the Iterative Deepening Search (IDS) algorithm.

BFS with Heuristic 1 (BFS_h1): This algorithm also generated a task tree with 16 functional units while taking the success rate of the motion into account.

BFS using Heuristic 2 (BFS_h2): A smaller task tree with 15 functional units was produced by the BFS search using Heuristic 2, which takes the quantity of input objects into account. This proves that BFS_h2 was able to reach this objective node via a more effective route.

Goal Node: Greek Salad

IDS: For the goal node "Greek Salad," the IDS algorithm generated a relatively large task tree with 37 functional units.

BFS with Heuristic 1 (BFS_h1): The BFS search with Heuristic 1 produced a larger task tree with 41 functional units. This suggests that BFS_h1 might consider less efficient paths, resulting in a larger task tree.

BFS with Heuristic 2 (BFS_h2): Similarly, BFS using Heuristic 2 also yielded a task tree with 41 functional units. Both BFS search algorithms with heuristics resulted in task trees of the same size for this goal node.

Goal Node: Macaroni

IDS: The IDS algorithm generated a task tree with 9 functional units for the "Macaroni" goal node.

BFS with Heuristic 1 (BFS_h1): The BFS search using Heuristic 1 produced a larger task tree with 12 functional units.

BFS with Heuristic 2 (BFS_h2): The BFS search with Heuristic 2 resulted in a smaller task tree with 9 functional units. This indicates that BFS_h2 found a more efficient path for this goal node, like the case of "Whipped Cream."

Goal Node: Sweet Potato

IDS: For the "Sweet Potato" goal node, the IDS algorithm generated a relatively small task tree with 3 functional units.

BFS with Heuristic 1 (BFS_h1): The BFS search with Heuristic 1 also produced a task tree with 3 functional units, indicating that it did not significantly impact the task tree size for this goal node.

BFS with Heuristic 2 (BFS_h2): Similarly, BFS using Heuristic 2 resulted in a task tree with 3 functional units. Both BFS search algorithms with heuristics and IDS yielded similar task tree sizes for this goal node.

Goal Node: Ice

IDS: The IDS algorithm generated the smallest task tree with only 1 functional unit for the "Ice" goal node.

BFS with Heuristic 1 (BFS_h1): The BFS search with Heuristic 1 also produced a task tree with 1 functional unit, matching the result of IDS.

BFS with Heuristic 2 (BFS_h2): Similarly, BFS using Heuristic 2 resulted in a task tree with 1 functional unit. All three algorithms produced the same, minimal task tree size for this goal node.

Execution time varies depending on the algorithm and goal node. IDS generally takes longer to execute but may find more comprehensive task trees.

V. DISCUSSION

The results of the experiments provided the insight into the advantages and disadvantages of the two search algorithms, Iterative Deepening Search (IDS) and Breadth-First Search with Heuristic 1 (BFS_h1) and Heuristic 2 (BFS_h2). These algorithms are useful for various contexts because each one has a unique set of benefits and drawbacks.

IDS is a comprehensive approach to search the FOON network, making sure to explore every possible path within a gradually increasing depth limit. This method, which seeks to identify the most extensive task tree, is very useful for large and complicated FOON networks. It worked well in situations like "Greek Salad" and "Whipped Cream," when the goal object needed a relatively complex series of operations. In contrast, it might not be the best option when the goal is a straightforward task, such as "Ice" or "Sweet Potato."

IDS may not be appropriate for applications demanding quick decisions or with limited computational resources because it can use more memory and take longer to execute.

The BFS_h1 and BFS_h2 heuristic-based algorithms provide two distinct viewpoints on path selection, considering, respectively, motion success rates and input object efficiency. They performed best in situations when the goal object had a reasonably simple path, such "Ice" and "Sweet Potato." In these examples, both methods generated task trees that were brief and effective.

However, both BFS algorithms produced larger task trees than IDS for more difficult problems, such as "Greek Salad," suggesting that they might take fewer effective routes.

These algorithms are appropriate for real-time applications with resource limitations because they perform well when speed and efficiency are critical factors.

VI. CONCLUSION

A crucial step in developing robotics and automation, especially in the context of robotic task planning and cooking, is the development of search algorithms for Functional Object-Oriented Networks (FOON). This paper introduced and compared two search algorithms, Iterative Deepening Search (IDS) and Breadth-First Search with Heuristic 1 (BFS_h1) and Heuristic 2 (BFS_h2), for extracting task trees from FOON networks.

The search algorithm chosen relies on the needs of the application. IDS is a thorough strategy appropriate for challenging and difficult jobs, although it could require more processing power. For tasks with straightforward and direct pathways, on the other hand, when speed and effectiveness are crucial, BFS algorithms with heuristics are preferred.

Future work in this field might concentrate on improving the heuristics used in BFS algorithms and investigating hybrid strategies that combine the best features of IDS and BFS. Moreover, the application of these algorithms in real-world robotic cooking scenarios could provide valuable insights into their practical utility.

In conclusion, the creation of search algorithms for FOON networks represents an exciting new direction for improving automation and robotics systems, one that has the potential to fundamentally alter how robots interact with their surroundings and carry out difficult tasks.

REFERENCES

[1] D. Paulius, Y. Huang, R. Milton, W. D. Buchanan, J. Sam, and Y. Sun. Functional Object-Oriented Network for Manipulation Learning. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2655–2662. IEEE, 2016.

[2] Md Sadman Sakib, Hailey Baez, David Paulius, and Yu Sun. Evaluating Recipes Generated from Functional Object-Oriented Network. *arXiv preprint arXiv:2106.00728*, 2021. (Featured in *18th International Conference on Ubiquitous Robots (UR 2021))*.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] FOON Website: Graph Viewer and Videos. http://www. foonets.com, 2021. Accessed: 2023-10-15.

[5] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[6] Md. Sadman Sakib, David Paulius, and Yu Sun. Approximate task tree retrieval in a knowledge network for robotic cooking. *IEEE Robotics and Automation Letters*, 7:11492–11499, 2022.

International Conference on Humanoid Robots, pages 529–536, 2011.

[7] Angel Andres Daruna, Devleena Das, and S. Chernova. Explainable knowledge graph embedding: Inference reconciliation for knowledge inferences supporting robot actions. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1008–1015, 2022.

[8] Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion: From natural language instructions to feasible plans. *arXiv preprint arXiv:2303.12153*, 2023.

[9] OpenAI. Chatgpt. https://openai.com/, 2021. Accessed on 15th October 2023.

[10] David Paulius, Alejandro Agostini, and Dongheui Lee. Long-horizon planning and execution with functional object-oriented networks. 2022.