Task Decomposition and Delegation Algorithms for Coordinating Unstructured Multi Agent Systems

António Luís Lopes
"We, the Body, and the Mind" Research Lab of ADETTI – ISCTE, Lisbon, Portugal
antonio.lopes@we-b-mind.org

Luís Miguel Botelho
"We, the Body, and the Mind" Research Lab of ADETTI – ISCTE, Lisbon, Portugal
luis.botelho@we-b-mind.org

Abstract

This paper describes the initial PhD research work for developing an innovative process for agent coordination in unstructured distributed environments. In this proposal, an agent that receives a request will use an AI-based planning algorithm to decompose the received request into several subtasks; it will handle those for which it has know-how and available resources; and it will use decentralized networks computing methods to delegate the rest of the request to other agents that can possibly handle it. These, in turn, will recursively apply the same process until the initial request is completely solved or until some specified termination condition holds. The work contributions to the state-of-the-art are: the creation of an AI-based planning algorithm capable of partial task decomposition with only partial knowledge of available planning operators; and the creation of a coordination infrastructure that enables agents to delegate subtasks to other agents in totally unstructured decentralized environments.

1. Introduction

The main goal of the research described in this paper is to develop a robust agent architecture that enables agents to freely participate in distributed problem solving in open scalable agent societies. In such societies, agents receiving requests from other agents would be capable of using their own capabilities to handle the parts of the problems for which they have competence and resources, and deliver the remaining parts to other agents that can possibly handle them more appropriately. Agent architectures with such mechanisms would enable the development of more autonomous robust artificial agents capable of acting in open flexible, dynamic and scalable agent societies.

This work combines artificial intelligence, distributed problem solving and computational paradigms from unstructured decentralized networks in order to address its two main challenges: partial task decomposition and coordinated task delegation over unstructured decentralized environments.

One of the objectives is to overcome the limitation of current planning algorithms by developing a planning algorithm that is capable of creating plans containing action slots for undetermined actions, enabling an agent that is trying to solve a problem, to delegate the part for which it cannot contribute, to other agents. All known planning algorithms must know all available planning operators (all possible actions) to generate their plans. However we need an algorithm that generates a plan in which part of the planned actions are known to the algorithm whereas others are only action slots for yet to be determined actions.

In order to provide an infrastructure in which agents can cooperate in this kind of environments, this work also addresses the problem of coordinated task delegation over unstructured decentralized networks. The objective is to create a task distribution algorithm that guarantees the consistency of the distributed problem solving. Current work in this area [10] involves either centralized components (which compromise scalability) or the excessive use of communication for synchronizing agents. Another objective of this work is to avoid unnecessary communication used in synchronization (either to avoid having tasks being performed twice or to avoid execution errors in a specific sequence of actions) through optimization of task delegation based on previous interactions.

In order to better understand the need for this kind of research, we present a comparison with related work in section 2 and a motivating example in section 3.

Section 4 describes the technical approach taken in this research. Section 5 concludes the paper by describing the current status of the work and the future steps to be taken in the research.

2. Related Work

Distributed problem solving has been defined as the cooperative process of solving problems by a decentralized collection of knowledge sources located in a number of different processor nodes [2]. The domain can be characterized as a collection of agents, each with different skills and limited knowledge, which acts to solve problems in a distributed fashion. The main challenges are [6]: describing the problem; decomposing and allocating sub-problems to agents, designing mechanisms for agents to interact; and ensuring coherence, that is, making sure that the agents make decisions and perform tasks that get them closer to the solution.

The focus of this research is on the decomposition of tasks and corresponding coordination mechanisms that ensure the means for agents to cooperate in problem solving, using only their own capabilities.

Planning algorithms use methods to decompose tasks recursively into smaller subtasks, until they reach primitive tasks that can be performed directly. STRIPS-like [5] representations have been the most common logic-based representations for discrete planning problems.

Another well-known representation and planning algorithm is based on Hierarchical Task Networks (HTN) [4, 3], which introduced powerful ideas such as task decomposition. However, HTN planning systems require full knowledge of the available planning operators (all agents' capabilities) and involve centralized solutions such as service composition agents.

These two requirements have been deemed unacceptable in partially inaccessible large-scale environments. Partial task decomposition, i.e. the possibility for an agent to contribute only to a part of the proposed problem instead of having to contribute with a complete plan, provided within a distributed problem solving coordinating infrastructure, may be the answer to this challenge. The main advantage of this approach lies on the fact that the entire decomposition process emerges as the result of a collection of local decomposition processes and that an agent does not need to know the other agents' capabilities.

Efficient coordination among large numbers of heterogeneous agents promises to revolutionize the way in which some complex tasks can be performed. However, state-of-the-art coordination approaches [7, 8] are not capable of achieving efficient and effective coordination when a team is very large, since centralized discovery and composition processes need to be fuelled with agents' new capabilities in continuously growing environments, dramatically decreasing the performance of the entire system. Also, these approaches spend large amounts of time finding complete solutions that will fail, because they can only solve part of the problem.

Recent work focusing on scalable coordination [14] illustrates that exponential search spaces, excessive communication demands, localized views, and incomplete information of agents pose major problems for large scale systems. The main limitations are based on the fact that these approaches rely on partial, dynamic centralization [1, 7, 8], such as brokers [9, 11] or hyperdatabases [15], even though the replication of global information in local repositories can help detach this dependency on centralized components [16]. Distributed constraint-based algorithms [12, 13] have high communication requirements that get dramatically worse as the team size increases. Therefore, there is a clear need for alternative methods for large-scale coordination of agents' cooperation in distributed problem solving.

3. Motivating Example

In order to assess the need for this distributed problem solving approach based on task decomposition and delegation algorithms, let's consider a motivating example. This section of the paper describes a scenario which comprises a world with specific characteristics where some plan of action must be composed and executed by a team of agents with different capabilities, in order for a certain goal to be achieved.

The agents will collaborate to achieve the defined goal, using their own actions and planning capabilities. They will collaborate with each other in the creation of the distributed plan of action and by acting accordingly in the execution phase. At each moment in time, each agent has only a limited view of the world, but they can have, at all times, a complete view of the plan being collaboratively created, because they pass it to each other as they delegate tasks among them.

It is important to stress that agents are not aware of each other action capabilities. They are only aware of the actions they can perform and consequently they also cannot include the other agents' actions in the plan being collaboratively designed.

3.1. The Scenario's World

The world in this scenario (depicted in Figure 1) is a grid with *n x m* cells, where several different items are placed. In this grid there is a team of agents with different capabilities, which have to interact in order to create and execute a plan that will enable them to store specific items in a storing area.

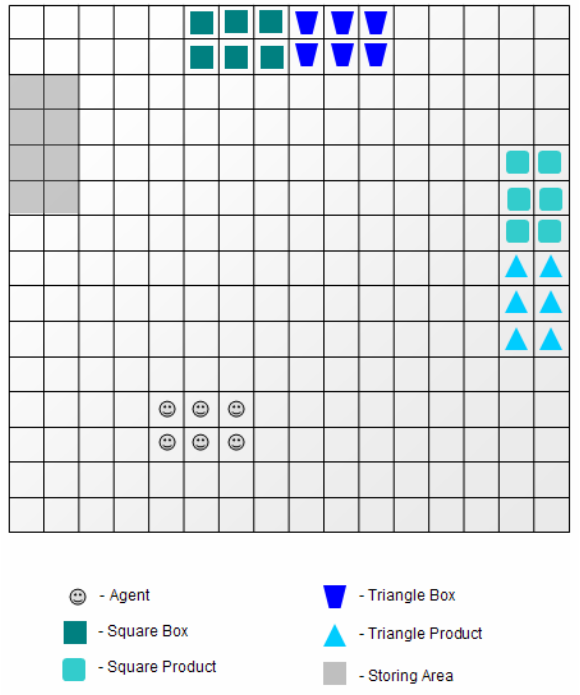

Figure 1. The Example World

These items can be:

- Square Products – A Square Product is a specific item which can only fit inside a Square Box
- Square Boxes – A Square Box is an item that can contain one Square Product inside, can be piled on top of another Square Box, and can be put anywhere on a free cell of the world
- Triangle Products – A Triangle Product is a specific item which can only fit inside a Triangle Box
- Triangle Boxes – A Triangle Box is an item that can contain one Triangle Product inside, can be piled on top of a Square Box, and can be put anywhere on a free cell of the world.

The agent team includes agents of the following classes, according to their own action capabilities:

- Square Packer – a Square Packer is an agent that can only pack Square Products inside Square Boxes
- Triangle Packer – a Triangle Packer is an agent that can only pack Triangle Products inside Triangle Boxes
- Square Mover – a Square Mover is an agent that can move any Square item (Square Product or Box) around the world grid
- Triangle Mover – a Triangle Mover is an agent that can move any Triangle item (Triangle Product or Box) around the world grid
- Pile Pusher – a Pile Pusher is an agent that can pile up boxes on top of each other

The team of agents can have several agents with the same capabilities, but in order to increase the complexity of the planning environment, the number of agents with the same capabilities should be as minimal as possible (ideally, only one agent per class) and each agent can only have one of the mentioned capabilities.

Besides the individual capabilities, the agents have capabilities that are common to all the agents in the environment, such as moving around in the world grid with no restrictions.

Some rules can apply to the way items can be placed in the world grid and to the way agents can interact, so that the composition planning process is more complex. For example:

- In order to be packed, a Product (either Triangle or Square) has to be in an adjacent cell of the Box in which it will be packed into
- In order to be piled up, a Box (either Triangle or Square) has to be in an adjacent cell of the Box in which it will be piled onto
- Boxes (either Triangle or Square) can be piled up on top of Square Boxes, but there cannot be piles with more than 2 piled Boxes
- Triangle Boxes cannot have any items piled on top of them
- In order for a Box or Product to be moved, the agent which will move it has to be in an adjacent cell
- Products cannot be piled on top of Boxes even if they are of the same type
- If a Box is stored on a storing cell and a second Box is on top of the first one, then the second Box is considered to be stored on a storing cell as well

3.2. Planning and Executing the Example

Considering the scenario described in section 3.1, we can now envision a situation that would require a team of agents with different capabilities, interacting between them to achieve the final goal. In this case, we consider the following goal: all Products should be packed into Boxes, according to their types (Square or Triangle) and all Boxes should be stored in the Storing Area, according to the rules mentioned in section 3.1.

The planning process starts when this goal is received by one of the agents (for example a Square Packer Agent), which will analyze the goal by decomposing it into simpler tasks, according to its own capabilities. This agent will then create a plan that will contain some of its own capabilities (if these are relevant to the task at hand) and some empty slots for the other agents to fill.

In this example, we assume the agent creates the plan *(EmptySlot + [A1] + EmptySlot)*, where *A1* is one of the possible actions the agent can perform (for example the action "*pack a Square Box*") and each *EmptySlot* is an empty space in the plan that will be replaced by a sub plan when other agents find concrete actions that fit there. The used notation ("+") represents the concatenation of sub-plans.

In the next stage, the agent that has received the original goal will send the partially specified plan to the remaining agents on the team so that these can recursively apply the same process until a final plan is created. Each agent (for example a Square Mover Agent) analyzes the unrefined slots in the plan, using a task decomposition algorithm and considering its own capabilities in order to fill in the empty slots.

A possible plan in the second stage of this process would be something like *([A2, A3] + [A1] + [A3])*, where *A3* and *A2* are the actions that the agent that received the delegated task added to the initial plan (in this case, the Square Mover Agent filled the plan with its actions *A2 – "move a Square Product"* and *A3 – "move a Square Box"*).

Once a final plan is completed with no empty slots (in this case the plan would be *[A2, A3, A1, A3]*), the agents can execute it by following the corresponding sequence of actions in the plan.

Several options are available to implement a cooperation and delegation mechanism as the one described above. In the following section, we present and discuss the technical approach that can realize the described process.

4. Technical Approach

4.1. Coordination Infrastructure for Task Delegation

One of the focuses of this research is the development of the coordination infrastructure that will enable agents to cooperate in an unstructured decentralized environment. Two possible approaches are considered for the task delegation coordination infrastructure:

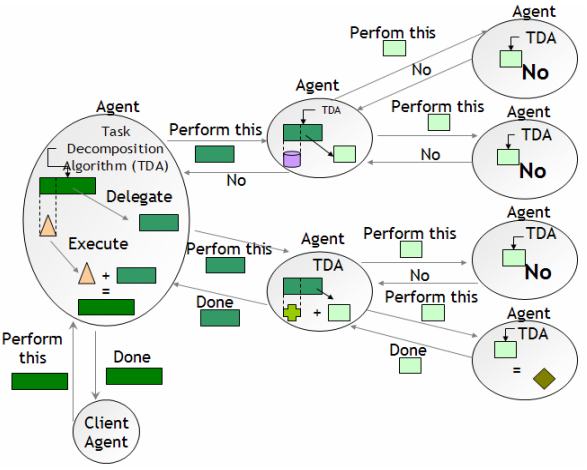

Figure 2. One-phase approach for the task delegation coordination infrastructure

One-phase-approach (see Figure 2): Delegated tasks will be decomposed into subtasks that will be immediately executed or recursively delegated until the whole task is completed.

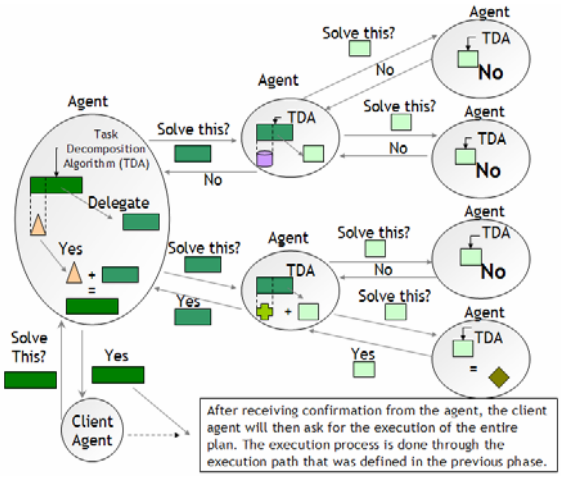

Figure 3. Two-phase approach for the task delegation coordination infrastructure

Two-phase-approach (see Figure 3): Previously, agents are asked if they will ensure the task proper execution. The initial request is decomposed into subtasks, some of which can be handled by the agent that receives the request. Then, the agent inquires other agents if they would be capable of executing the remaining subtasks. These, using an analogous process, will proceed until an execution path is found that ensures the completion of the whole initial task.

The first approach requires a great amount of synchronization to avoid having two agents performing the same task or tasks that depend on others that were not performed yet. Besides, it may and will happen that agents will perform subtasks that are part of a plan that will latter fail.

The second approach does not require the described synchronization but it will possibly have a longer execution time for the entire task. Besides, in highly dynamic environments, an agent that has previously ensured the execution of a given subtask may no longer be capable of executing it in a latter moment.

Both approaches can be used in non-cooperative environments, where agents that receive a task may first want to negotiate possible rewards for its decomposition, delegation and execution. If this happens, the agent that negotiates task delegation will have to choose the best deal.

We will analyse and implement both approaches so that comparative data can be acquired on their relative strengths and weaknesses.

4.2. Task Decomposition Algorithm

After deploying and testing the task delegation coordination infrastructure, the focus of the research will be on the definition and development of the task decomposition algorithm. The main challenge is to provide a way for agents to produce partially-specified plans, that is, plans that contain actions that are known to the agent, which it can perform and slots for actions yet to be determined, which will be delegated to other agents.

The generated plans will contain totally refined parts and parts yet to be refined. The major contribution of this algorithm is the ability to consider both the agent concrete capabilities, which will be known to the algorithm, and the other agents' abstract capabilities, which will not be known to the algorithm. That is, an algorithm capable of generating plans containing slots for actions that are not known to the algorithm.

The plan will only become totally refined by the recursive approach of delegating the non-refined parts to other agents, based on the coordinated task delegation infrastructure described in the previous sub-section.

5. Conclusion

This paper describes initial PhD work that is currently in its first year. The work focuses on the development of an innovative process for agent coordination in unstructured decentralized environments in which agents collaborate to solve a specific problem for which they can only partially contribute, without knowing the other agents' capabilities. This kind of approach can help overcome some of the limitations that exist in current systems, such as centralization and full knowledge of the planning space.

The research work will be applied to strategic game playing, in order to determine if this kind of approach can help develop human-like behavior in strategic games' artificial opponents. In this kind of applications several challenges arise, which will have to be addressed in the future work of this PhD research:

- Strategic games are often limited in time, resources and/or budgets. This may force the execution process to start before the planning process ends, which may be an obstacle to using the two-phase approach described before. In this case, interleaved planning and execution has to be considered.
- There are quite a lot of (unknown) dependencies and relations between the available actions in this kind of applications, which makes the planning process much more complex, especially in environments where each agent doesn't know the other agents' capabilities.

6. References

[1] Davin, J., and Modi, P. J., 2005, "Impact of problem centralization in distributed constraint optimization algorithms". In Proceedings of the Fourth international Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1057-1063.

[2] Decker, K. S., 1987, "Distributed problem solving: A survey", IEEE Trans. on Systems, Man, and Cybernetics, 17(5):729-740.

[3] Erol, K., 1996, “Hierarchical Task Network Planning: Formalization, Analysis, and Implementation”, Doctoral Thesis, UMI Order Number: UMI Order No. GAX96-22054., University of Maryland at College Park.

[4] Erol, K., Nau, D. and Hendler, J., 1994, “HTN Planning: Complexity and Expressivity”, In Proceedings of the Twelfth National Conference on Artificial Intelligence, 1123–1128. Menlo Park, Calif.: American Association for Artificial Intelligence.

[5] Fikes, R. E. and Nilsson, N. J., 1971, “STRIPS: A new approach to the application of theorem proving”, Artificial Intelligence Journal, 2:189-208, 1971.

[6] Harandi, M. T. and Rendon, G., 1998, “Classification and organizational issues in distributed problem solving”, In Proceedings of the 1998 ACM Symposium on Applied Computing, pp. 65-69.

[7] Helin, H., Klusch, M., Lopes, A., Fernandez, A., Schumacher, M., Schuldt, H., Bergenti, F., and Kinnunen, A., 2005, “CASCOM: Context-Aware Service Co-ordination in Mobile P2P Environments”, Multiagent System Technologies, Lecture Notes in Computer Science, Vol. 3550 / 2005, ISSN: 0302-9743, pp. 242-243.

[8] Helin, H., Klusch, M., Lopes, A., Fernandez, A., Schumacher, M., Schuldt, H., Bergenti, F., and Kinnunen, A., 2005, “Context-aware Business Application Service Co-ordination in Mobile Computing Environments”, In Proceedings of the AAMAS Workshop on Ambient intelligence – Agents for Ubiquitous Environments, Utrecht, Netherlands.

[9] Hunsberger, L. and Grosz, B., 2000, “A combinatorial auction for collaborative planning”, In Proceedings of the Fourth International Conference on MultiAgent Systems, pp. 151–158.

[10] desJardins, M. E., Durfee, E. H., Ortiz, C. L. Jr., Wolverton, M. J., 1999, “Survey of research in distributed, continual planning”, Artificial Intelligence Magazine. Vol. 20, no. 4, pp. 13-22.

[11] Lopes, A., Botelho, L.M., 2005, “SEA: a Semantic Web Services Context-aware Execution Agent”, In Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web.

[12] Mailler, R. and Lesser, V., 2004, “Solving distributed constraint optimization problems using cooperative mediation”, In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems.

[13] Modi, P., Shen, W., Tambe, M. and Yokoo, M., 2003, “An asynchronous complete method for distributed constraint optimization”, In Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems.

[14] Scerri, P., Vincent, R. and Mailler, R., 2004, “Comparing three approaches to large scale coordination”, In Proceedings of the Workshop on Challenges in the Coordination of Large Scale Multi-Agent Systems of the Third International Joint Conference on Autonomous Agents and Multiagent Systems.

[15] Schek, H.-J., 2002, “Hyperdatabases: infrastructure for the information space”, Lecture Notes in Computer Science, ISSU 2287.

[16] Schuler, C, Weber, R., Schuldt, H., Schek, H.-J., 2003, “Peer-to-Peer Process Execution with Osiris”, Lecture Notes in Computer Science, ISSU 2910, pp. 483-498.