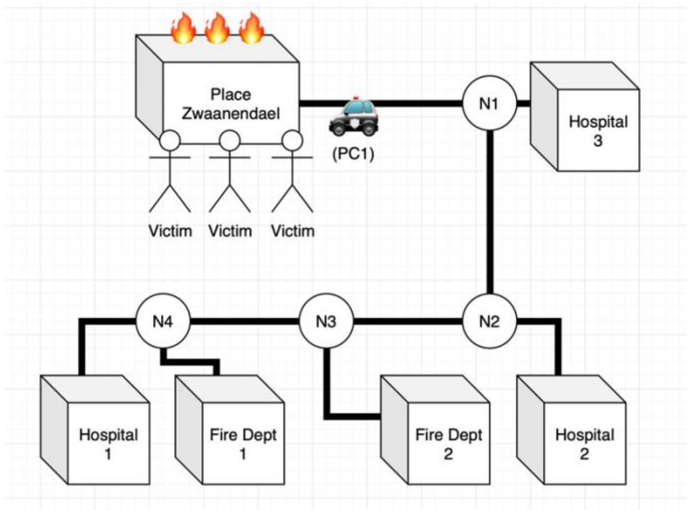
CISC 489
Homework 2
April 11, 2022
James Villemarette

Problem 1



In this scenario, we must reference the rules of the Contract Net from the textbook, page 156, section 8.2.1 “Task sharing in the Contract Net”.

First, let’s follow the general protocol for task allocation detailed in figure 8.3 of the book. Police Car 1 (PC1) would recognize the problem, and then start formulating a task.

Victims die faster when a place is on fire, and a police car only needs a fire fighter agent present at the place to start treating victims. Because of these rules of this scenario, the PC1 agent should formulate a task for fire fighters, first.

In the second stage of the CNET protocol, the PC1 agent will do a task announcement to the Fire Departments (1 & 2) that the Place Zwaanendael is on fire. In the third stage, each department will calculate the node traversal distance. Fire Department 2 (FD2) would have to travel 3 nodes (N3 —> N2 —> N1 —> Place Zwwanendael), and FD1 would have to travel 4 nodes.

So in the fourth stage of the CNET protocol, PC1 agent will likely award FD2 the contract.

As FD2 is underway to Place Zwaanendael to put out the fire, PC1 agent will start another CNET protocol process to request an ambulance for the victims.

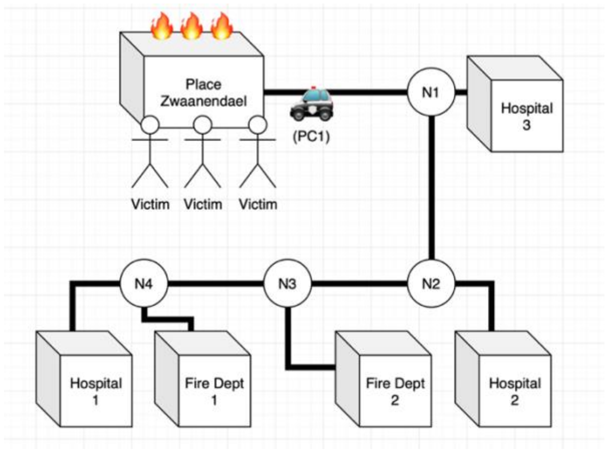
What is tricky in this CNET protocol is the lack of information sharing and checking that it affords. With my best interpretation, I think how the process for contracting the ambulance would go something like this:

PC1 would do the first stage of the CNET protocol, and recognize the problem as three victims, and evaluate their severities (state). In the second stage, PC1 would announce the task to all three hospitals (H1, H2, H3). In the third stage, H3 would probably say that it can move all three victims in 12 moves (going from H3 —> N1 —> Place Zwaan. —> N1 —> H3 —> …), H2 in 18 moves, H1 in 30 moves. The hospitals might take more than just the distance into accounting for the cost that it would take to move all the victims (cost of ambulance, driver salary, hospital capacity, hospital state, hospital capabilities, Kaiju destruction/not-yet-destructed, etc.) In the fourth stage, PC1 would probably award H3 the contract.

I just asked Professor Decker in his office hours (April 6, 12:04 pm), and he said the other contingencies to account for here are that Contract Net can be more flexible than originally stated. That the flow of problem recognition to task announcement to awarding to bidding does not have to be linear and can be interrupted and restarted. With that in mind, and assuming our PC1 agent is stateful, and can remember history, then we could more dynamically and cleanly allocate (contract) resources to address this problem.

Problem 2

I will be using the formal outline of the FIPA ACL language using page 140, section 7.2.2 “The FIPA agent communication language” of the textbook.



Again, I am referencing my diagram here. The four agents that I will be

- Police Car 1 (PC1) at the scene of the incident, Place Zwaanendael
- Ambulance 3 (A3) which is at Hospital 3
- Ambulance 2 (A2) which is at Hospital 2
- Fire Engine 2 (FE2) which is at Fire Department 2

Additionally, there will be

- Fire Engine 1 (FE1) which is at Fire Department 1

Note: Many examples of FIPA ACL seem to use XML for structuring data. I will do so, here.

Dialogue:

```
(inform // The Police Car would inform all nearby agents of the scene
      :sender PC1
      :receiver A3, A2, FE2
      :language FIPA :ontology ACL
      :content
      <Scene>
            <Name>Place Zwaanendael</Name>
            <Victims>3</Victims>
            <Building>OnFire</Building>
      </Scene>
)

(cfp // The Police would ask the nearest fire engine for its proposal
      :sender PC1
      :receiver FE2
      :action “PutOutFire”
      :eventID 1001
      :condition “time < 10 mins”
      :condition “cost < $1000”
)
```

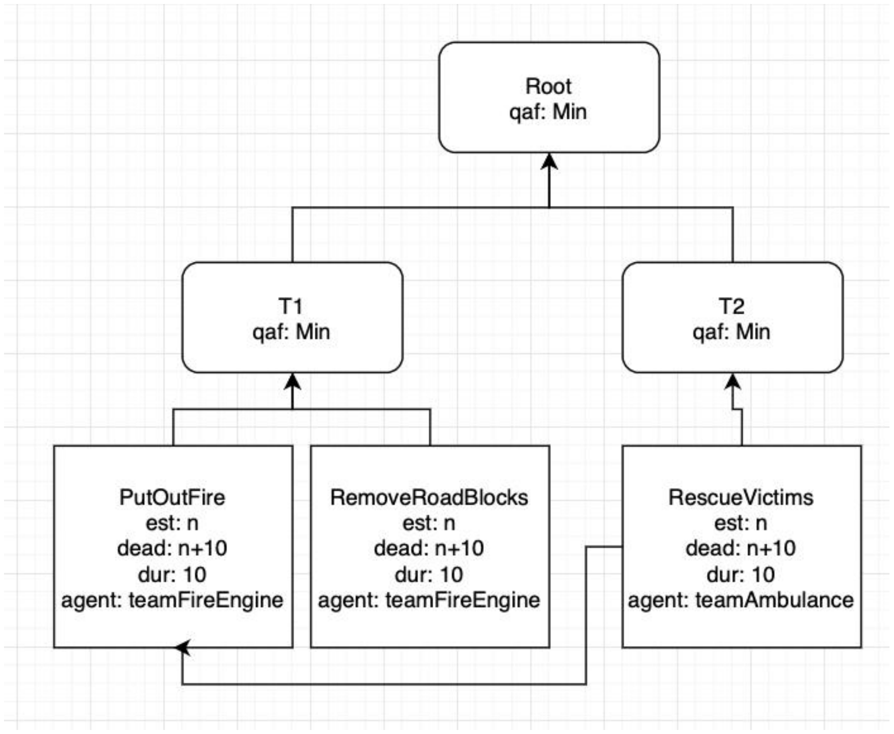
```
(propose // Fire Engine 2 would propose its resources
      :sender FE2
      :receiver PC1
      :eventide 1001
      :proposalID 302
      :time “8 mins”
      :cost $500
)

(accept-proposal // Police Car 1 would accept FE2’s proposal
      :sender PC1
      :receiver FE2
      :proposalID 302
)

(subscribe // Fire Engine 2 would listen to Police Car 1’s updates
      :sender PC1
      :receiver FE2
      :eventID 1001
)

(cfp // Police Car 1 would now call for proposals from the ambulances
      :sender PC1
      :receiver A3, A2
      :action “RescueVictims”
      :eventID 1001
      :condition “time < 10 mins”
      :condition “cost < $1000”
)

(propose // Ambulance 3 would propose its resources
      :sender A3
      :receiver PC1
      :eventID 1001
      :proposalID 303
      :time “4 mins”
      :cost $500
)
```

```
(propose // Ambulance 2 would propose its resources
	:sender A2
	:receiver PC1
	:eventID 1001
	:proposalID 304
	:time “9 mins”
	:cost $800
)

(reject-proposal // Police Car 1 would reject A2’s worse proposal
	:sender PC1
	:receiver A2
	:eventID 1001
	:proposalID 304
)

(accept-proposal // Police Car 1 would accept A3’s better proposal
	:sender PC1
	:receiver A3
	:proposalID 302
)

(subscribe // Ambulance 3 would listen to Police Car 1’s updates
	:sender PC1
	:receiver A3
	:eventID 1001
)

(request // Police Car 1 now requests Fire Engine 2 to the scene
	:sender PC1
	:receiver FE2
	:place Zwaanendael
	:time now
)

(request // Police Car 1 now requests Ambulance 3 to the scene
	:sender PC1
	:receiver A3
	:place Zwaanendael
	:time now
)
```

Additional updates may include:

- failure: If a fire engine or ambulance discovers a tree in its path
- inform-if: From the Police Car 1 to all responding agents to inform Police Car 1 if there's a tree in its path
- propagate: For a bigger network of agents
- proxy: Communication is somehow limited to only be 3 nodes max distance

Problem 3



To the left is the simple model in TÆMS that I have constructed. I am only considering the three basic problems presented by Professor Decker's scenario:

1. Put out the fire
2. Remove any road blocks
3. Rescue the victims to the hospital

To answer the questions "what are the cost, quality, and durations of tasks", it would be highly dependent on the specifics of the scenario that have not been exactly presented to us. I would assume that in order of time use, I would say its RemoveRoadBlocks > PutOutFire > RescueVictims. That is, removing the roadblocks takes the most time, putting out the fire less time, and rescuing the victims the least time. In the <Quality, Cost, Duration> format of TÆMS task structure, it may look like what I put below:

<PutOutFire, 500, 8>	// Putting out the fire is expensive, and time consuming
<RemoveRoadBlocks, 100, 10>	// Removing the road blocks is less expensive, more time
<RescueVictims, 700, 4>	// Rescuing the victims costs the most, and the least time

In this simple example, I would say that the Min, Max, Sum, and XOR operations cover the quality accumulations, nicely. We are generally looking for just the lowest (min) time and cost of moving units. I would not need to define any new or other subtask quality accumulation functions, I just don't believe this problem demands it.

Enables and Facilities I think do help the non-local effects. The other agents facilitating the Police Car 1 agent's goal of saving the victims serves the overall goal and purpose of the emergency system.

I do not think this problem is complicated enough to demand new NLEs.

Problem 4

I will be using Lecture8.pdf as a reference for addressing this problem.

<u>Definitions</u>

Partial Global Planning (PGP) is where agents exchanged information to reach conclusions about problem solving plans. It is only partial because this is not delivering a complete solution, and it is global because agents are exchanging local plans to achieve a global solution (plan).

There are three stages:

- Each agent decides its own goal and local plan
- Agents exchange plans to determine interdependence
- Agents alter local plans for coordination

PGP will interleave planned activities. There is a scheduling phase that is highly repetitive to optimize ratings (using hill-climbing). PGP will repeat this computation on new plans arrival. It can be highly predictable, each agent can estimate the duration of the goals, and follow a prescribed order.

It has general mechanisms to generalize updating non-local viewing points, communicating results, handling redundancy (there is a lot of that here; foreshadow), and handling hard & soft coordination relationships.

STEAM is computationally tractable, and made to handle communication costs, uncertainty about other team members, handle single and multiple agent failures, and evolve a hierarchy of joint events. By forming joint intentions, it can establish organizational roles & dependencies.

<u>Compare & Contrast</u>

Compare
They're both trying to solve the coordination problem.
Some shared state or sharing of information needs to occur.
Shared uncertainty since no one agent knows the global state, only their own local states.

Contrast
STEAM is computationally tractable, while PGP appears to be very resource intensive.
STEAM has models of other agents, while PGP does not.
STEAM reduces communication use at all costs.
STEAM develops organizational roles & dependencies, while PGP does not.

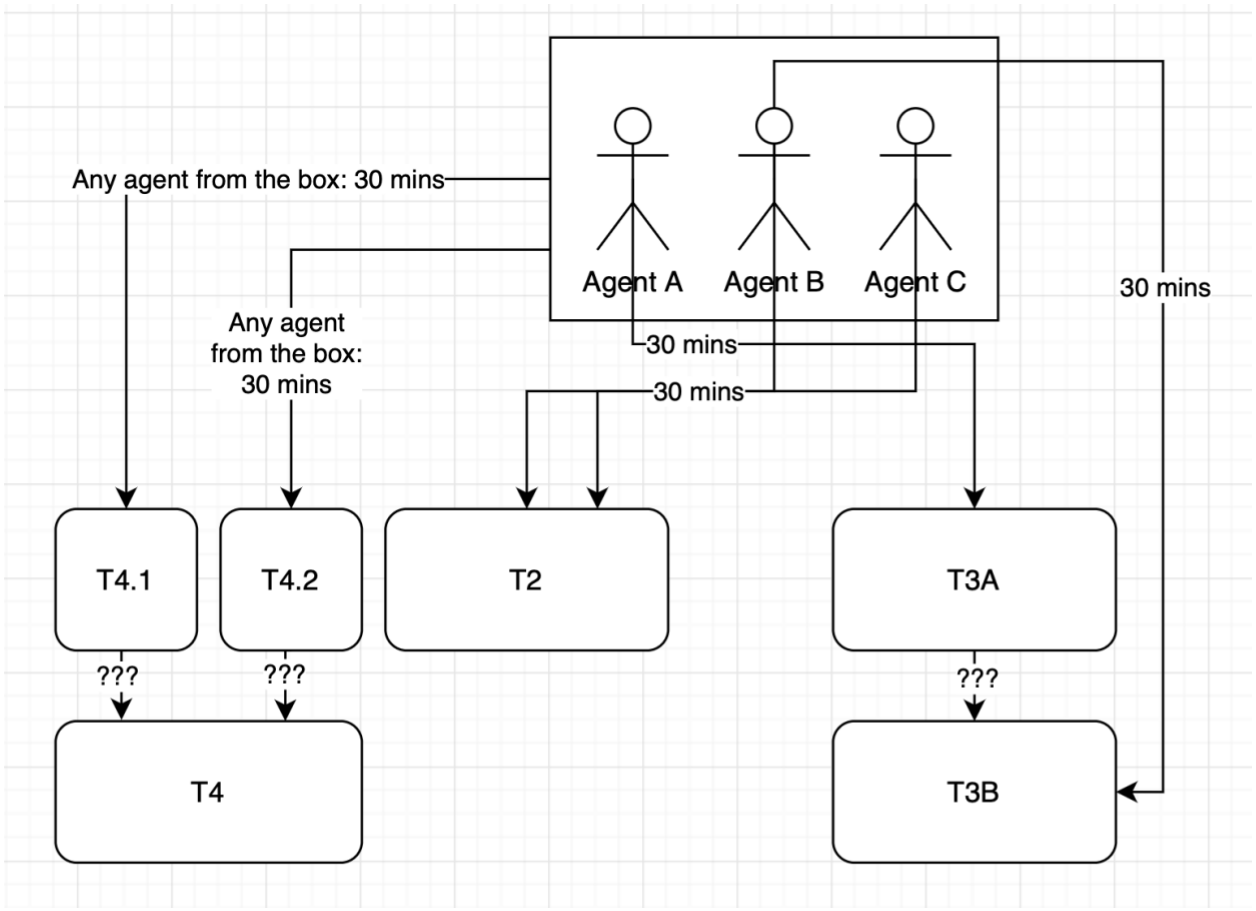
Problem 5

I will be using the DCOP-TAEMS.pdf as a reference in addressing this problem.

We will represent this in the Constraint Satisfaction Formalism approach, that is the Constraint Satisfaction Problem (CSP) form, of <X, D, C>

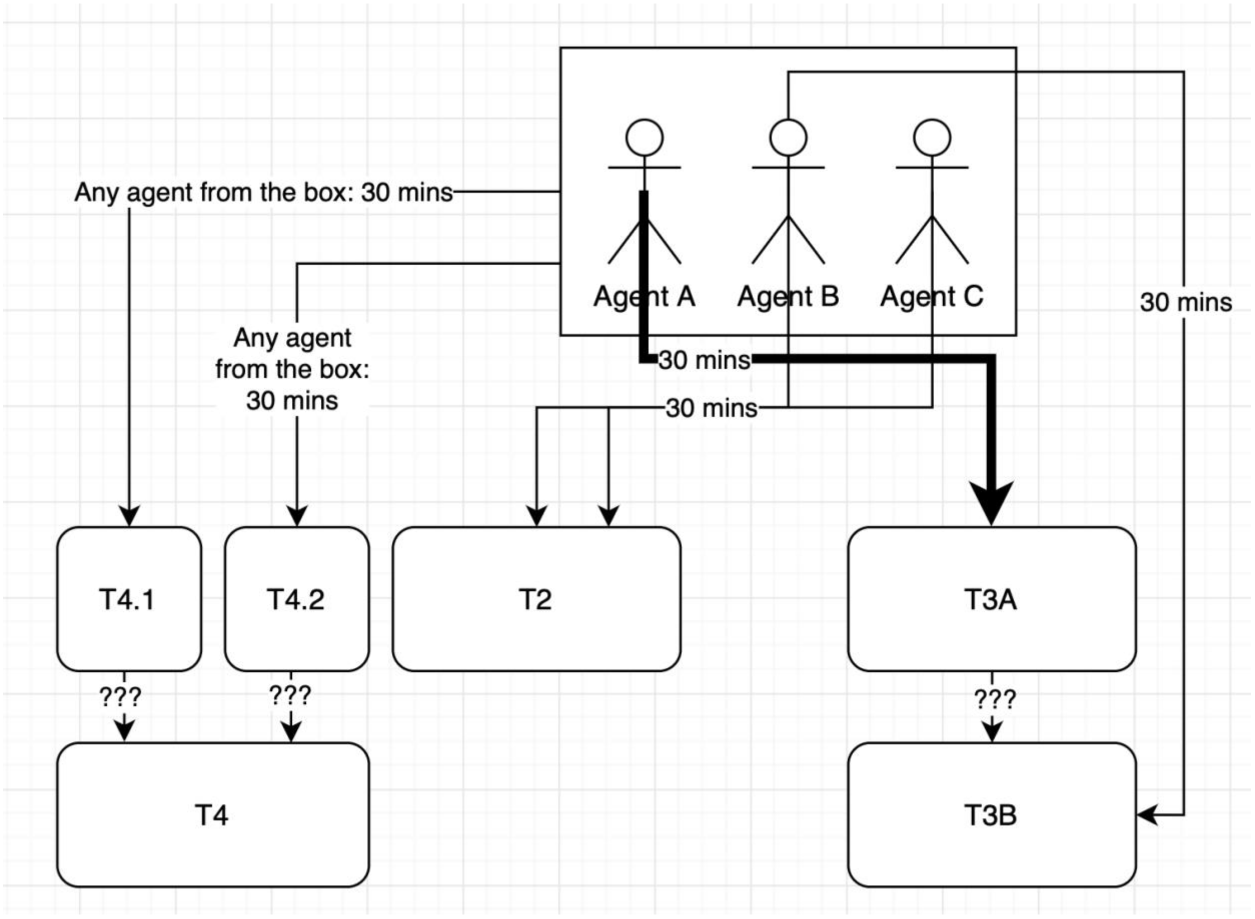
Our X is to be defined as our variables, in this case, our tasks: T2, T3A, T3B, T4.1, T4.2, and T4.
Our D is to be defined as our domains, in this case, our agents: A, B, and C.
Our C is our set of constraints. In this case, it is the constraints that Professor Decker has defined in the instructions.

The constraint diagram of this problem looks like so. Creating this diagram was somewhat recommended by Professor Decker in the CISC 489 Discord, channel #489-689_mas_homework on 04/08/2022.

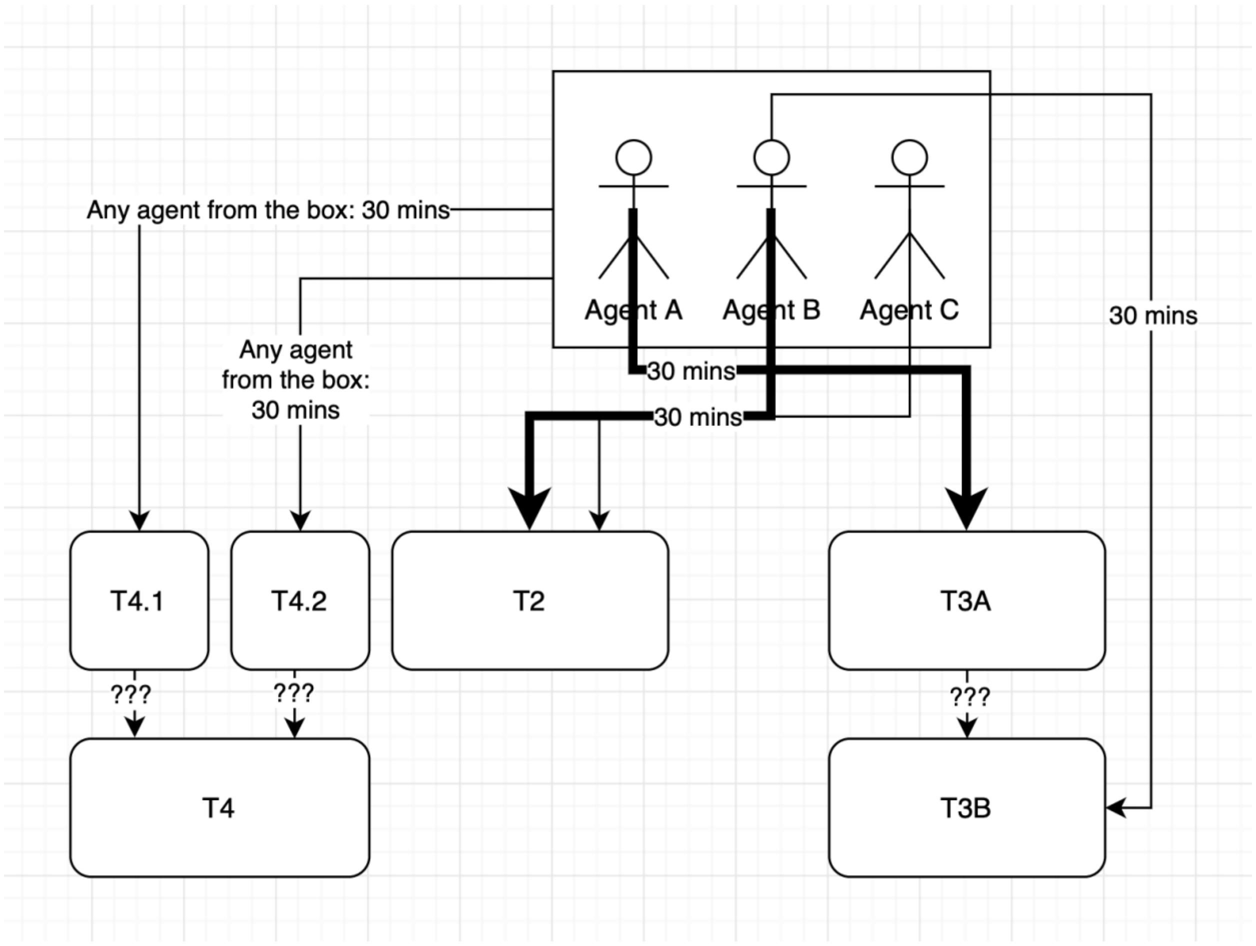


The time durations between task T4 and T4.1 and T4.2, and between T3A and T3B were not explicitly described. I assume they are 30 mins, since every other task is 30 mins, but I am just marking their ambiguity on my diagram.

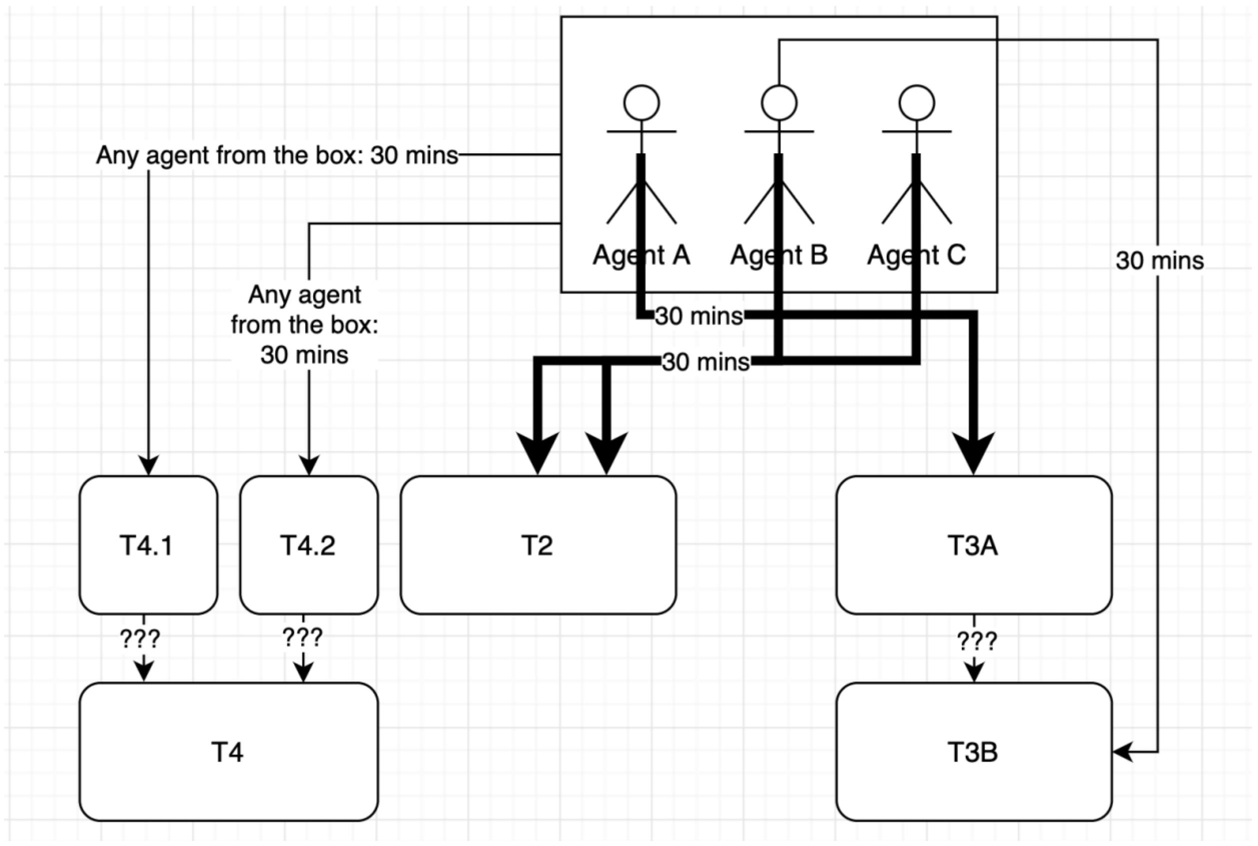
I am using a depth first search (DFS), or more generally, a backtracking search to find the optimal solution in this graph.



First, just starting randomly, Agent A would do start task T3A, for 30 mins.

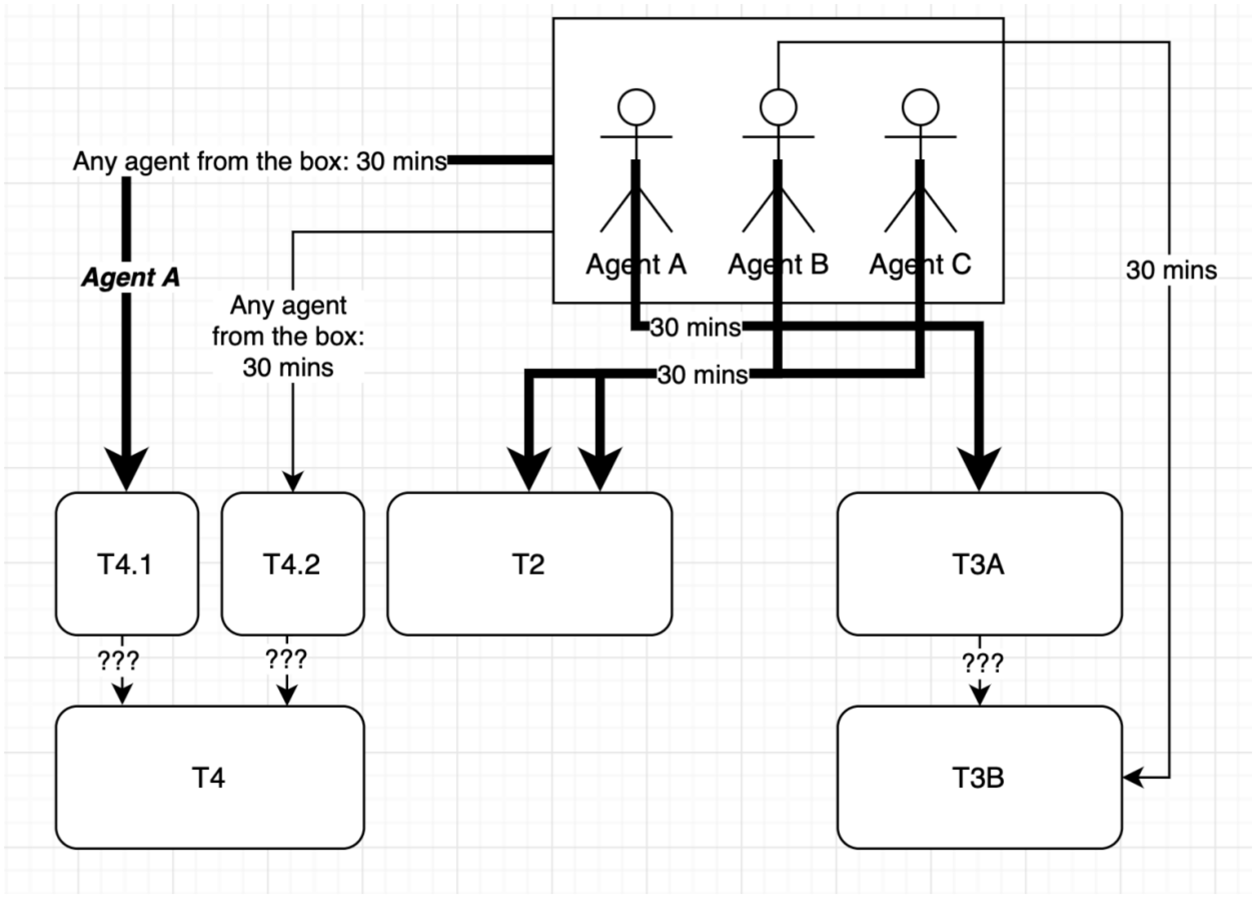


Then, secondly, Agent B would start task T2 for 30 mins. But T2 needs both Agents B & C to be working on it. So we would back track all the way, and start fresh again. I am not going to show all the intermittent steps, as that would be many, many screenshots. So, I am just going to keep pressing forward, assuming I have backtracked and restarted.

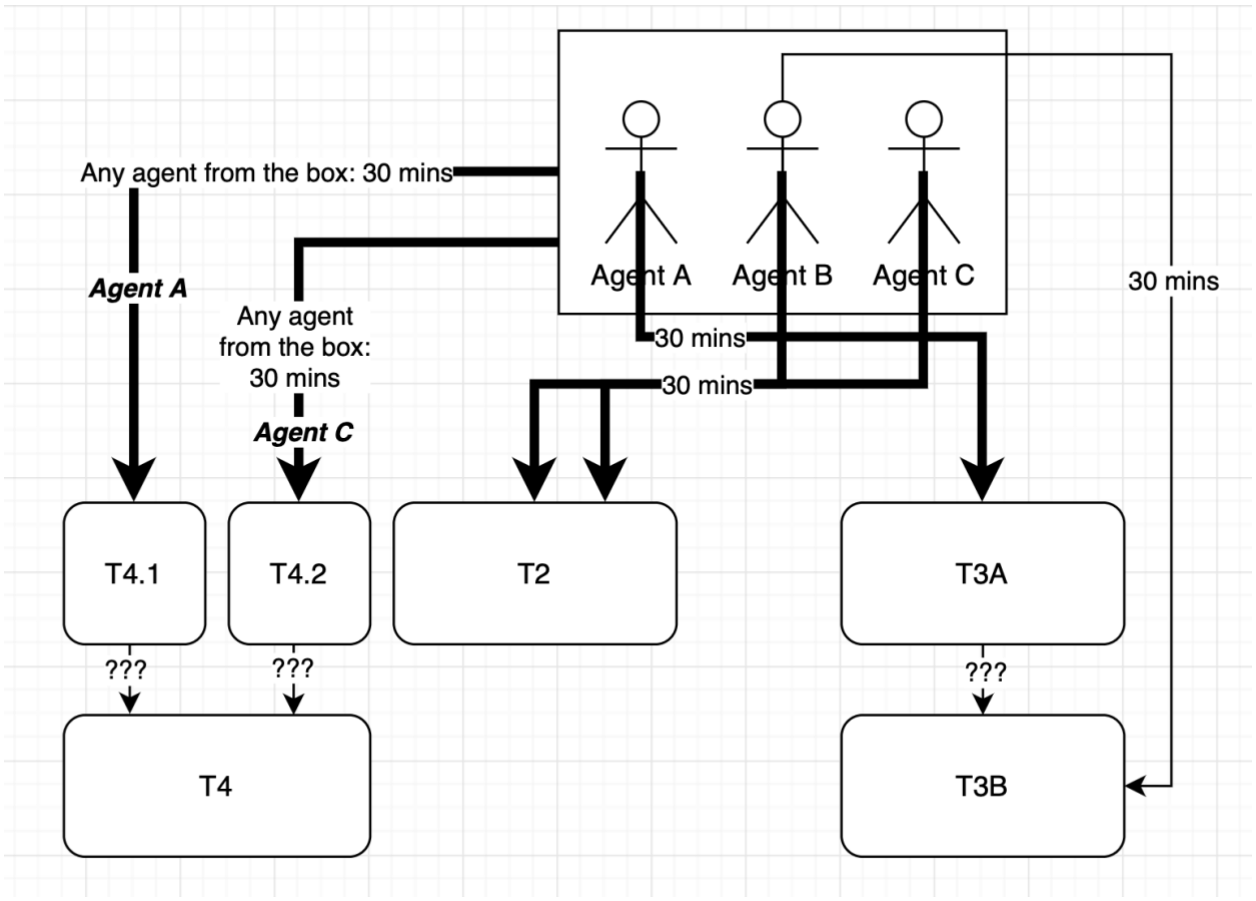


Agent C is now also assigned to start T2, for 30 mins.

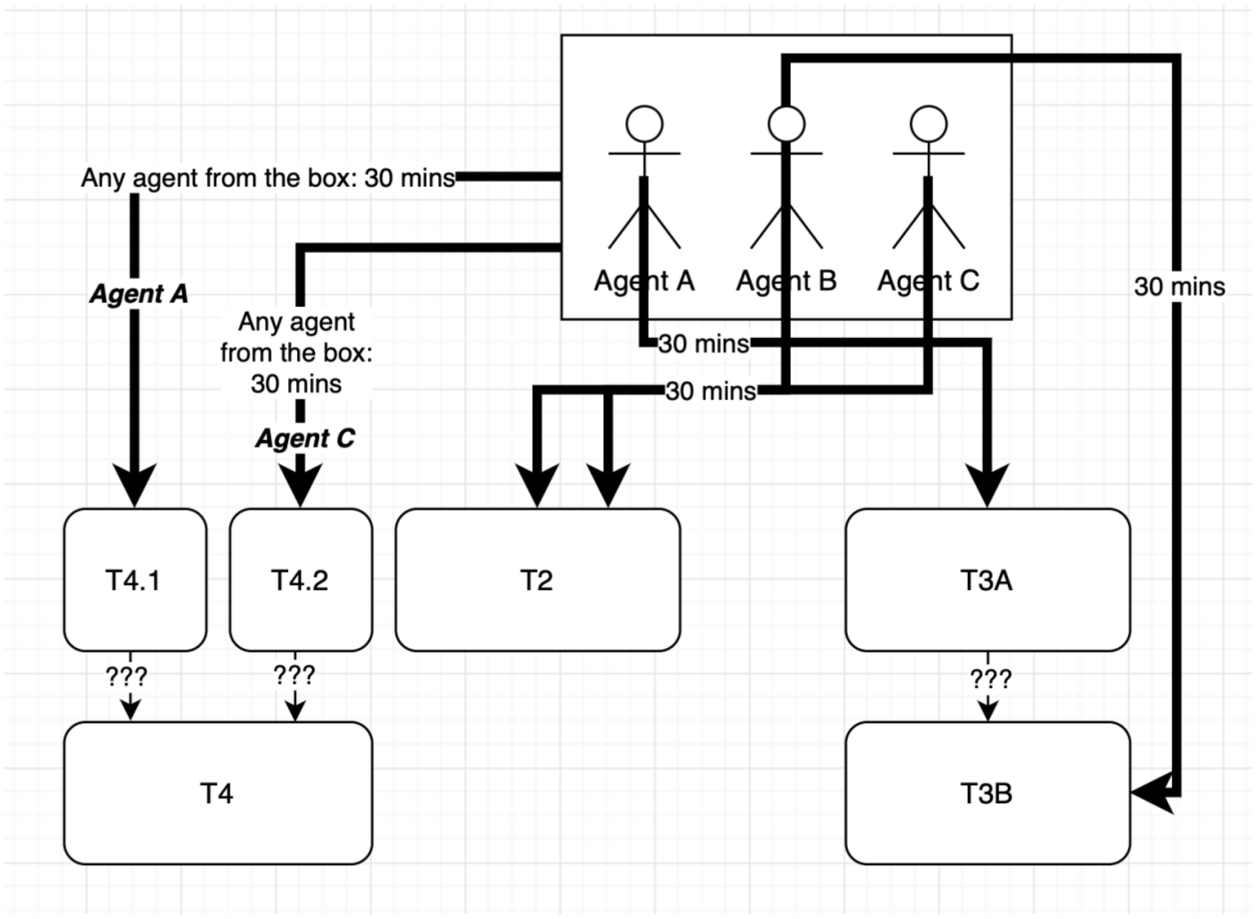
We now wait 30 mins, as all agents are assigned.



Agent A is now assigned to do T4.1.

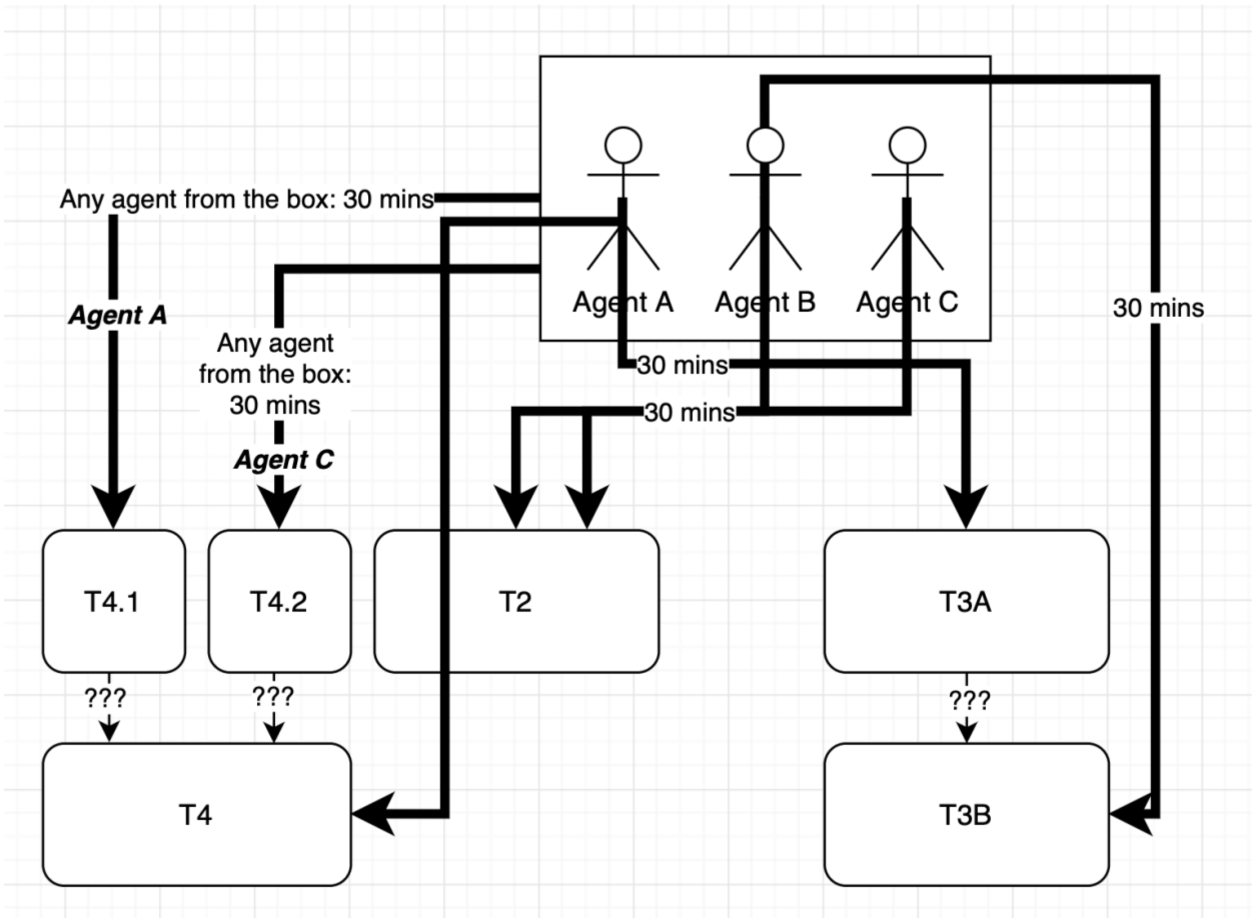


Agent C is now assigned to do T4.2.



Agent B is assigned to T3B.

We now must wait another 30 minutes, as all agents are assigned to 30 minute tasks. The total time is now 1 hr.



Agent A is now assigned to do the last task, T4, for 30 minutes.

In conclusion, 1 hour and 30 minutes (1:30) time passed to complete all the tasks. Considering that every task was 30 minutes, and there are 6 tasks in total, that would have meant 3 hours in total for one agent, I think the work was efficiently split between the 3 agents.

The slides say to represent the constraints with utility functions, and I would say my utility function would just return the time for the task to complete.

In which case, this problem could easily be represented as a search problem, and a search algorithm like DFS could solve it, as I have manually done in the screenshots above.

Considering that the slides on DCOP look pretty insane, I think this is a pretty reasonable approach to solving this problem.