CSCE 475/875

Handout 3: Some Real World Examples

January 14, 2020

In this handout, I provide some real world examples that we could see the application of multiagent-related methodologies and strategies.

Rainwater Harvesting

In India, for example, fresh water is at places a very scarce resource. These days, there is a campaign to help harvest rainwater to provide more water to the inhabitants. So, that is the central goal. One harvesting method is local-based. The official goes to a village, surveys for the most suitable place, and consults with the villagers to come up with a local solution. Each solution is unique, making use of the local resources, native resources, natural terrains, manpower, and special needs. Thus, we see distributed solutions. Do you see why this is a good example where multiagent methodologies are applied? Resources and expertise are different at different villages (or agents). One solution that works for one village may not work for another. Thus, it is just impossible to have only one plan, one solution. Naturally, the solution set has to be distributed, as long as each local solution achieves the overall goal.

For more information on Rainwater Harvesting, see Montaigue, F. (2002). Water Pressure, *National Geographic*, **202**(3):2-33.

Great Wall

The Great Wall in China is one of the greatest man-made structures in the world. It extends thousands of miles. Researchers have found out that the bricks used to build the Great Wall were of different materials at different regions: well-made clay bricks, mud bricks, rocks, straw bricks, and so on. Why is that? Different regions had different resources. To the northwest, at the time that portion of the Great Wall was built, the technology, materials, manpower, and transportation were not as advanced and adequate as those in the East. As a result, once again, the builders made use of local resources and technologies such as making bricks out of compact straw! So, we see here an application of distributed solutions for a centralized goal.

Musical Band

Why did the Beatles break up? Why did Simon and Garfunkel break up? Why did many musical bands break up, right after they have achieved some sort of stardom? If we look at this from the viewpoint of multiagent systems, it is actually quite straightforward. Let us take a basic musical band: a singer, a drummer, a guitarist, and a bassist. These are four agents. Each is an expert in a particular area. In the beginning, the four agents work together to try to achieve stardom, a centralized goal. Thus, they cooperate. Now, suppose the band achieves stardom. Now, the band members (agents) start to have distractions. The guitarist wants the songs he writes to be on the album. The singer thinks about a solo career. The bassist wants to tour the world. The drummer tries to start a career in acting. Now, the centralized goal is no more. Now, the multiagent system has distributed goals. So, we have a distributed planning for distributed plans situation. And it becomes difficult for the agents to carry out their own plans without conflicting each other's. In other words, it takes more effort for the agents to cooperate, for the band members to stay as a band. This example should make you realize that distributed planning for distributed plans is not a trivial task.

Chess

In the game of Chess, each player has 16 pieces: a Queen, a King, two Bishops, two Knights, two Rooks, and eight pawns. If the chessboard is a multiagent environment, and each chess piece is an agent, will the agents exhibit coherent behaviors?

Traditionally, the AI approach to chess-playing software has been centralized, with only one decision maker controlling all 16 chess pieces. Why?

Let us suppose that we want to build a multiagent chess-playing software so that every piece is an agent. So, we have a Queen agent, a King agent, 2 Bishop agents, and so on. Suppose we tell each agent: "Your goal is to help win the game." Now, what are the pieces going to do to achieve that goal? Does each piece monitor a portion of the chessboard? But, it is almost always that before moving each piece, one needs to know the positions of all pieces on the board. So, each piece needs to monitor the entire chessboard. Now, before a piece can move, does it need to know about what other pieces are doing? Well, at every step, only one piece can move. So, the situation is not that dynamic. However, for every step, many pieces can move and one has to choose the best move to make. So, the Queen agent needs to know all possible moves and pick the best move to take. So does the King agent, and so on. Wait a minute! If we use this design, then we would repeat the move selection for all the pieces. That is not very efficient. Okay, what about every agent only considers the moves that its piece can make? For example, the Queen agent enumerates all the moves that the Queen can make, and picks the best. So does the King agent, and so on. Then, out of the best moves, a controller picks the best out of the best moves. Is this multiagent system? Have we defeated the purpose of agents here? What we have just described is actually a parallelization strategy. If I have a computer for each piece, then I can enumerate many moves in a parallel. But do we really need agents? The answer is no.

We do not need agents in chess-playing software. It is counter-productive. The above discussions should make it clear to you why.

Symphony

In a symphony, there is a conductor, and many musical instruments specialists: violinists, pianists, cellists, woodwinds players, and so on. The conductor is the person who determines how a particular song should be played and directs the musicians. But can the conductor play the piano as well as the pianists in the symphony? Most likely not. So, you see a multiagent system with a hierarchy: the conductor is the super agent, and the musicians are agents taking instructions from the conductor. But each musician has his/her expertise and knows how to use it. (Think about this: A basketball coach may not be able to dunk a basketball, but he/she knows how to coach a team. A player knows how to dunk a basketball, but he/she may not know how to attack the weakness of the opponent team.) So this symphony is a very natural multiagent concept.

Suppose that the symphony is playing with some famous musicians (e.g., YoYo Ma, or Issac Stern). Now the conductor *collaborates* with them. Why is that? Because these famous musicians are extremely talented, very knowledgeable, and very experienced. They also have their own interpretations of how songs should be played, have their own goals in terms of emotional expressions, and so on. Thus, the conductor can no longer dominate. Thus, the multiagent has changed its makeup. And we can see that these collaborations are mostly successful. Why is that? Because the two agents are highly "intelligent" (goal-directed and autonomous)!! On the other hand, equal-part collaboration between an expert and a novice usually does not fare well because of the difference in expertise/intelligence.

So, what do we learn here? When we design our multiagent system, when we have to decide how to configure our agents, think about what the expertise/intelligence levels are for the agents.

Eco-Challenge

Eco-Challenge is an expedition race. Each team of four, comprising men and women, races non-stop for 6 to 12 days, 24-hours a day, over a rugged 300 mile course using mountain biking, river rafting, horseback riding, mountaineering and fixed ropes, kayaking and navigation skills.

The first team to cross the finish line together, in full complement, is the winner. If a team loses a member due to illness, fatigue, injury or a team disagreement, they are disqualified. Only teams that can work together as friends have any hope of reaching the finish line.

Reference: www.ecochallenge.com

This is a perfect real world example of a multiagent system. Each team member has a certain set of expertise, and they MUST work together as a team to achieve the global goal: to finish the race and to win the race. Because the race is so challenging, many teams have failed to finish. The team members must help each other, must cooperate, must agree on various strategies (for example, whether to continue throughout the night, whether to take a dangerous short cut, whether to rest, etc.), and must continuously re-evaluate their goals versus their health.

You are encouraged to check out the website and read about the various races, since 1995 in Utah to this year's Fiji expedition. Hopefully, this will give you a better sense of why multiagent systems are such a powerful AI paradigm.

An American Football Team

On an American football team, there are many players. During a game, the offensive team has 11 players on the field, and the defensive team has also 11 players on the field. On the offensive team, it has in general a Quarterback, one Tight End, 2-3 Receivers, 1-2 Running Backs, and 4-5 Offensive Linemen. Each is an agent in a multiagent system. Each plays a role and function within the multiagent system. Therefore, we say this multiagent system is heterogeneous. However, the multiagent system (an offensive team) is able to communicate and coordinate their plays. How? Through a synchronization mechanism that is known as the "huddle!" Before each play starts, the coach sends in the signal and the Quarterback gathers all the players in a huddle and relays the signal. Sometimes, the Quarterback is allowed to change the plays real-time based on his judgment of the situation. That is known as an "audible." Also, even though these agents are heterogeneous, they all share a single communication language and protocol as they all read the same "playbook!" Each team has its own codes to designate its collection of plays. As a result, the team members can communicate efficiently and effectively on and off the field. However, that also means that the team members must remember the "playbook" to avoid miscommunications. These are very true in any multiagent systems that we build.

What about the "no-huddle offense?" Sometimes, when the offensive team needs to score in a hurry, the players move from a play to the next without getting together. The Quarterback assesses the situation and quickly makes a decision, and the other players also assess the situation and quickly make their own decisions. As a result, they line up for their play without explicit coordination! How could this have worked? (It works sometimes, but not well. But it does work sometimes.) First, the players have been trained together and have been coached to know what one should do at a given situation. These are trainings in strategies. The players are supposed to be able to think and rationalize and observe what their teammates and opponents are doing and come up with the best action to take. Second, the opponents have to go through “no-huddle defense” as well. Thus, the defensive schemes may not be optimal since the opponents may not have enough time to observe the situation. So, a soon-enough, good-enough solution may be sufficient.