ABD3 Multi-Agent Systems Theorie

Peter Brouwer

Hoofdstuk 2 Distributed Optimization

- Contract Net

Contract net is een mogelijke idee/aanpak voor het verdelen van een globaal probleem onder een groep van agents met verschillende skills/specialisaties. Elk agent heeft een functie die de kost berekent om een set taken te volbrengen en de agents onderhandelen met elkaar om de taken zo optimaal mogelijk toe te wijzen en zo de totale kost te minimaliseren.

Contract net is geen een specifiek algoritme maar een fundamentele aanpak voor het verdelen van een probleem.

- Auction Algorithms

Auction algoritmes zijn verschillende soorten algoritmes die gebruikt worden om de beste soort bod van een groep bieder agents in een veiling te bepalen.

round	p_1	p_2	p_3	bidder	preferred object	bid incr.	current assignment
0	0	0	0	1	x_2	2	$(1, x_2)$
1	0	2	0	2	x_2	2	$(2, x_2)$
2	0	4	0	3	x_3	1	$(2, x_2), (3, x_3)$
3	0	4	1	1	x_1	2	$(2, x_2), (3, x_3), (1, x_1)$

- Linear Programming met het assignment problem.

Het assignment problem gaat over het verdelen van taken aan op een manier dat elke agent een taak krijgt op de meest efficiënte manier. Linear programming is een aanpak om het assignment problem op te kunnen lossen. Met Linear programming wordt er gekeken welke variables kan worden gemaximaliseerd en geminimaliseerd zodat op logische wijze de beste oplossing wordt gekozen.

i	$v(i, x_1)$	$v(i, x_2)$	$v(i, x_3)$
1	2	4	0
2	1	5	0
3	1	3	2

(Voorbeeld agent I met mogelijke aantal punten voor toewijzing taken voor elke agent)

- Integer Programming met het scheduling problem.

Het scheduling probleem gaat over het verdelen van tijdslots tussen verschillende agents die een gedeelde bron nodig hebben om hun taken te voltooien. Elke persoon heeft een deadline en er is een beperkt aantal tijdslots beschikbaar.

Het doel is om een planning te maken waarin elke persoon zijn taak kan uitvoeren en deze op tijd kan voltooien. Hierbij wordt er geprobeerd om de waarde die elke persoon hecht aan het op tijd voltooien van de taak te maximaliseren.

Met Integer Programming worden de beslissingen doorgaans weergegeven als gehele getallen, waardoor het model met discrete variabelen een scheduling problemen kan vastleggen.

job	length (λ)	deadline (d)	worth (w)
1	2 hours	1:00 P.M.	$10.00
2	2 hours	12:00 P.M.	$16.00
3	1 hours	12:00 P.M.	$6.00
4	4 hours	5:00 P.M.	$14.50

(Voorbeeld van het vastleggen in discrete variabelen)

Op het figuur hebben elke job 3 verschillende variabelen. Hieruit kan er gekeken worden welke taak als eerste wordt gedaan op basis van length, deadline en worth.

Opdracht

Situatie scheduling problem:

- Er kan maar 1 agent tegelijk op een food hotspot komen.
- Iedere agent is een willekeurig aantal stappen vandaan van de food hotspot
- Een agent kan maar een willekeurig aantal beurten wachten totdat de agent doodgaat van de honger
- Er bestaat maar 1 food hotspot

Agent	Afstand	Honger
1	2	9
2	4	7
3	10	15

List agent prio = []

Forall agent do:

Calculate agent effective waittime based on hunger and distance()

Add effective waittime to list agent prio with the assigned agent

Sort list on smallest effective waittime

For each item in List agent Prio do:

Agent move to food with distance

Done

Hoofdstuk 8 Communication

8.1 “Doing by talking” 1: cheap talk

1. Een speler in een communication game kan worden beïnvloed op de soorten incentives die worden meegegeven
2. 2 spelers hebben de mogelijkheid om elkaar tegen te spelen als de incentive gelijk of beter is dan de minst optimale winst
3. 2 soorten uitspraken self-commiting (een declaratie van een speler intent die zien wordt als de beste uitkomst) en een self-revealing (een declaratie van een speler om dit te doen maar liegt en gaat een andere actie kiezen als de incentive ervoor is)

Doing by talking gaat over de afhankelijkheid van acties door middel van communicatie die van tevoren is gedaan, dit wordt ook wel ‘cheap talk’ genoemd. Een voorbeeld van cheap talk is de Prisoners Dilemma game.

	C	D
C	$-1,-1$	$-4,0$
D	$0,-4$	$-3,-3$

8.2 ”Talking by doing”: signaling games

Soms zijn acties luider dan woorden.

1. Signaling games zijn 2 speler games waarin **Nature** de game aan maakt gebaseerd op bekende distributie. Speler 1 is geïnformeerd op de gemaakte game die **Nature** heeft gemaakt en kiest een actie op basis van die informatie. Speler 2 weet niks op van de game van **Nature** maar wel van de keuze van Speler 1 en baseert zijn keuze op basis van deze informatie

	L	R
U	$4,-4$	$1,-1$
D	$3,-3$	$0,0$

	L	R
U	$1,-1$	$3,-3$
D	$2,-2$	$5,-5$

2. Bayes Theorem van toepassing op de berekening van Speler 2 beslissing
3. Speler 1 kan verschillende strategieën toepassen op basis van de gegeven Nature om een betere Payoff te krijgen (is te berekenen op basis van Bayes Theorem)
4. Signaling games valt onder asymmetrisch informatie spellen

Opdracht

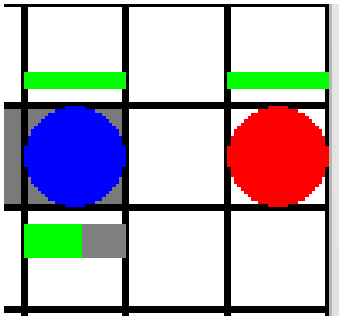
Voor een mogelijke toepassing in Hunger games simulatie heb ik op basis van de informatie vergaderd het volgende:

Ik wil met gebruik van “Talking by Doing” het toepassen op de choose_direction() functie in de agent class.

```python
def choose_direction(self, tiles, amount_tiles):
    """
    Look at all directions and determine how many tiles are not explored yet. Decide based on this which direction to go to.
    :param tiles: The entire grid
    :param amount_tiles: Amount of tiles the Agent can see ahead of him
    :return: Direction with the most amount of tiles that aren't explored yet
    """
    directions = {"north": 0, "east": 0, "south": 0, "west": 0}
    latest_direction = None
    directions = self.enemy_avoidance_based_on_memory(directions, tiles, amount_tiles)
    x, y = self.index()
    for i in range(x - 1, max(x - amount_tiles - 1, -1), -1):  # Check west direction
        # if not tiles[i][y].objects:
        if self.map_has_visited[i][y] == 0:
            directions["west"] += 1
    for i in range(x + 1, min(x + amount_tiles + 1, len(tiles))):  # Check east direction
        if self.map_has_visited[i][y] == 0:
            directions["east"] += 1
    for j in range(y - 1, max(y - amount_tiles - 1, -1), -1):  # Check north direction
        if self.map_has_visited[x][j] == 0:
            directions["north"] += 1
    for j in range(y + 1, min(y + amount_tiles + 1, len(tiles[0]))):  # Check south direction
        if self.map_has_visited[x][j] == 0:
            directions["south"] += 1

    return max(directions, key=directions.get)
```

Deze functie “kijkt” in elke richting vanuit zijn huidige positie en telt op punten op de posities die de agent het minst heeft bezocht. Dit is dan bijvoorbeeld Player 1 in de signaling game. Player 2 is dan een andere agent die Player 1 heeft gezien op het veld. Player 1 heeft een beweging actie genomen. Player 2 ziet dat en met deze informatie kan wordt er samen met de choose direction() functie een pay-off kunnen worden berekend.



Het doel hiermee is om een soort achtervolging gedrag toe te kunnen voegen in de agents als er voldoende incentive ervoor is om een andere agent te achtervolgen of te vermijden.