Introduction to Multi-Agent Programming

7. Working together

Coalitions and Role Assignment

Alexander Kleiner, Bernhard Nebel
Contents

• Introduction
• Coalition formation
  – Case study: ResQ Freiburg task allocation
• Dynamic Role Assignment
  – Case study: CS-Freiburg
• Summary
Coalition Formation

Introduction

• Necessary when tasks are more efficiently solved by a cooperating group of agents
  – E.g. ambulances can faster rescue victims if they are in a larger group
• Assignment of groups to tasks is necessary when tasks cannot be performed by a single agent
  – E.g. a single fire brigade cannot extinguish a large fire
• A group of agents is called a coalition
• A coalition structure is a partitioning of the set of agents into disjoint coalitions
• An agent participates in only one coalition
• A coalition may consist of only a single agent
• Generally, coalitions consist of heterogeneous agents
Coalition Formation
Example
Coalition Formation
Example
Applications for coalition formation

• In e-commerce, buyers can form coalitions to purchase a product in bulk and take advantage of price discounts (Tsvetovat et al., 2000)

• In Real Time Strategy (RTS) games groups of heterogeneous agents can jointly attack bases of the opponent. Mixture of agents has to be according to the defence strategy of the opponent

• Distributed vehicle routing among delivery companies with their own delivery tasks and vehicles (Sandholm 1997)

• Wide-area surveillance by autonomous sensor networks (Dang 2006)

• In Rescue, team formation to solve particular sub-problems, e.g. larger robots deploy smaller robots within confined spaces
Coalition Formation
Definition I

- Coalition formation includes three activities:
  - **Coalition structure generation**
    - Partitioning of the agents into exhaustive and disjoint coalitions
    - Inside the coalitions, agents will coordinate their activities, but agents will not coordinate between coalitions
  - **Solving the optimization problem** in each coalition:
    - pooling the tasks and resources of the agents in the coalition and solving the joint problem
    - The coalition objective could be to maximize the monetary value, or the overall expected utility
  - **Dividing the value** of the generated solution:
    - In the end, each agent will receive a value (money or utility) as a result of participating in the coalition
    - In some problems, the coalition value the agents have to share is negative, being a shared cost
Coalition Formation
Definition II

• A group of agents $S \subseteq A$ is called a coalition, where $A$ denotes the set of all agents and $S \neq \emptyset$
  – The coalition of all the agents is called grand coalition

• A coalition structure (CS) partitions the set of agents into coalitions
  – $CS^*$ is the social welfare maximizing coalition structure

• The value of each coalition $S$ is given by a function $v_S$
  – Each coalition value is independent of non-members actions
Coalition structure generation

• The value of a coalition structure is given by:

\[ V(CS) = \sum_{s \in CS} v_s \]

• The goal is to maximize the social welfare of the set of agents \( A \) by finding a coalition structure that satisfies:

\[ CS^* = \arg\max_{CS \in \text{Partitions}(A)} V(CS) \]
Special Coalition Values

• The coalition values are *super-additive* iff for every pair of disjoint coalitions $S, T \subseteq A$: $v_{S \cup T} \geq v_S + v_T$
  – If coalition values are super-additive, then the coalition structure containing the *grand coalition* gives the highest value
  – Agents cannot do worse by coordination

• The coalition values are *sub-additive* iff for every pair of disjoint coalitions $S, T \subseteq A$: $v_{S \cup T} < v_S + v_T$
  – If coalition values are sub-additive, then the coalition structure $\{\{a\} | a \in A\}$ in which no agent cooperates gives the highest value

• Is the *ambulance rescue task* in the RoboCup Rescue domain super-additive, sub-additive, or none of both?
Coalition structure generation

Example

The input is all possible coalitions and their values:

\[ A = \{1, 2, 3, 4\} \]

\[
\begin{array}{c|c|c|c|c|c|c|c}
 CL1 & v_s & CL2 & v_s & CL3 & v_s & CL4 & v_s \\
 \{1\} & 92 & \{1, 2\} & 189 & \{1, 2, 3\} & 316 & \{1, 2, 3, 4\} & 395 \\
 \{2\} & 96 & \{1, 3\} & 210 & \{1, 2, 4\} & 297 & & \\
 \{3\} & 87 & \{1, 4\} & 203 & \{1, 3, 4\} & 335 & & \\
 \{4\} & 105 & \{2, 3\} & 171 & \{2, 3, 4\} & 272 & & \\
 \{2, 4\} & & & 215 & & & & \\
 \{3, 4\} & & & 182 & & & & \\
\end{array}
\]

For N agents the number of possible coalitions is \(2^{N-1}\) but the number of possible coalition structures is \(N^{N/2}\).
Coalition graph

- For 4 agents: $A = \{1, 2, 3, 4\}$

- Nodes represent coalition structures

- Arcs represent either merges (downwards) or splits (upwards)
Coalition Structure Search I

- To search the whole coalition graph for the optimal coalition is intractable (in practice up from $|A| > 15$)
- Can we approximate the search by visiting only a subset of $L$ nodes?

\[
CS_L^* = \arg \max_{CS \in L} V(CS)
\]

- One requirement is to guarantee that the found coalition structure is within a worst case bound from optimal:

\[
k^* V(CS_L^*) \geq V(CS^*)
\]
Coalition Structure Search II

• **Theorem**: to bound $k$ for some subset $N$ of the coalition structures, it suffices to search the lowest two levels of the coalition structure graph
  
  - With this search, the bound is $k = |A|$, this bound is tight, and the number of nodes searched is $n = 2^{|A|-1}$
  
  - No other search algorithm (than the one that searches the bottom two levels) can establish a bound $k$ while searching only $n = 2^{|A|-1}$ nodes or fewer

• Intuition:
  
  - The lowest two levels of the coalition graph are the only two levels in which all possible coalitions occur
    
    • A level $l$ consists of coalition structures containing $l$ coalitions
    
    • Hence, if $l > 2$, the largest coalition in the level contains $|A| - l + 1$ agents since the smallest possible coalition contains 1 agent
Coalition Structure Search III

• Algorithm:
  – Search the **bottom** two levels of the coalition structure graph
  – Continue with breadth-first search from the **top** of the graph as long as there is time left, or until the entire graph has been searched
  – Return the coalition structure that has the **highest welfare** among those seen so far

• Note the search can be **distributed** among self-interested agents
Case study: ResQ Freiburg task allocation

- Problem description:
  - N ambulance teams have to rescue M civilians after an earthquake
  - Civilians are characterized by Buriedness, Damage and Hit-points
    - Buriedness is proportional to the required resources (ambulance cycles)
    - As more hit-points as more likely the civilian dies
    - The amount of damage increases the growth of hit-points, i.e. accelerates the time of death
  - Costs are the time to rescue a civilian, composed of the coalition’s joint travel time to reach the victim, and the time needed for the rescue
  - The overall utility is the number of rescued civilians (the civilians brought to a refuge)
- We considered the ambulance rescue task as super-additive
  - The rescue operation itself is super-additive
  - Assumption: travel costs are the same for every agent
  - However, consider the situation of 2 victims at two different locations that could both be rescued by a single agent but will die within a short amount of time
  - Maybe not the optimal solution!
The problem reduces to assign a sequence $R$ of rescue tasks to the entire set of agents $A$ (here the ambulances):

- $R = <r_1, r_2, \ldots, r_N>$ where $r_i$ denotes a rescue task and $i$ the position in the sequence

$U(R)$ denotes the predicted utility (the number of survivors) when executing sequence $R$

Hence, the problem is find the optimal sequence from the set of all possible sequences

- $R^* = \text{arg max} \ U(R)$

Enumerating all possible sequences is impossible within limited time (the world model changes frequently, altering the current sequence)

Greedy solutions

- Prefer victims that can be rescued fast (small burialness)
- Prefer urgent victims (high damage)
ResQ Freiburg task allocation
Implementation

• Non-allocated agents (e.g. police & fire brigades) continuously search unexplored locations and update information (e.g. buridness, health) about known victims

• The ambulance station (agent)
  – predicts for each known victim the lifetime and costs for rescue
  – simulates rescue sequences, selected by a genetic algorithm, over the set of known victims
  – When a better sequence has been found, the rescue sequence of agents in the field is altered

• Life time prediction
  – Learning of a decision tree for the classification of victims into will die and will survive
  – Adaptive Boosting (Ada Boost) for the regression learning of the life time prediction (previously on data sets)
  – Calculation of confidence values with respect to the age of information (e.g. as older the information as more unreliable the prediction)
ResQ Freiburg task allocation
Genetic Optimization

• Local search, i.e. **hill climbing**, that continuously improves the current best solution (**selection**)
• Solutions are represented by **strings** (DNA) that are locally modified for finding better outcomes (**mutation**)
  – For example 543261 → 534261
• Offsprings are generated by a **crossing** operation
  – For example “one-point crossover”
• **Genetic pool** is initialized with greedy solutions (e.g. prefer urgent victims or prefer victims that can be rescued fast)
• **Elitism**: Keep best two solutions in the genetic pool
• **Anytime execution**:
  – Number of genetic pool generations can be adjusted according to CPU usage
  – Optimization can anytime be stopped at current best solution
ResQ Freiburg task allocation
Results RoboCup 2004 cont.

Number of saved civilians by greedy and genetic sequence optimization on different maps
### ResQ Freiburg task allocation
Results RoboCup 2004

<table>
<thead>
<tr>
<th></th>
<th>ResQ</th>
<th>Damas</th>
<th>Caspian</th>
<th>BAM</th>
<th>SOS</th>
<th>SBC</th>
<th>ARK</th>
<th>B.Sheep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final-VC</td>
<td>42</td>
<td>43</td>
<td>52</td>
<td>34</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Final-Random</td>
<td>32</td>
<td>25</td>
<td>29</td>
<td>16</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Final-Kobe</td>
<td>46</td>
<td>45</td>
<td>46</td>
<td>30</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Final-Foligno</td>
<td>66</td>
<td>54</td>
<td>50</td>
<td>29</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Semi-VC</td>
<td>18</td>
<td>15</td>
<td>17</td>
<td>12</td>
<td>11</td>
<td>12</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>Semi-Random</td>
<td>22</td>
<td>26</td>
<td>16</td>
<td>14</td>
<td>20</td>
<td>14</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Semi-Kobe</td>
<td>57</td>
<td>47</td>
<td>54</td>
<td>52</td>
<td>20</td>
<td>39</td>
<td>34</td>
<td>44</td>
</tr>
<tr>
<td>Semi-Foligno</td>
<td>37</td>
<td>46</td>
<td>44</td>
<td>43</td>
<td>42</td>
<td>28</td>
<td>29</td>
<td>24</td>
</tr>
<tr>
<td>Round2-Kobe</td>
<td>57</td>
<td>37</td>
<td>43</td>
<td>50</td>
<td>43</td>
<td>35</td>
<td>28</td>
<td>43</td>
</tr>
<tr>
<td>Round2-Random</td>
<td>52</td>
<td>48</td>
<td>39</td>
<td>45</td>
<td>47</td>
<td>44</td>
<td>50</td>
<td>37</td>
</tr>
<tr>
<td>Round2-VC</td>
<td>31</td>
<td>33</td>
<td>32</td>
<td>24</td>
<td>37</td>
<td>51</td>
<td>N/A</td>
<td>34</td>
</tr>
<tr>
<td>Round1-Kobe</td>
<td>45</td>
<td>51</td>
<td>47</td>
<td>43</td>
<td>47</td>
<td>31</td>
<td>25</td>
<td>34</td>
</tr>
<tr>
<td>Round1-VC</td>
<td>62</td>
<td>62</td>
<td>55</td>
<td>57</td>
<td>N/A</td>
<td>51</td>
<td>54</td>
<td>44</td>
</tr>
<tr>
<td>Round1-Foligno</td>
<td>53</td>
<td>53</td>
<td>37</td>
<td>33</td>
<td>37</td>
<td>41</td>
<td>30</td>
<td>23</td>
</tr>
</tbody>
</table>

**# wins:** 9
**Σ TOTAL:** 620
**Σ SEMI+PREM** 434

Number of rescued civilians
Task Allocation For Fire Brigades

• Fires have to be clustered in order to define tasks
  – For each cluster a utility has to be computed, e.g. # of victims nearby, # of neighboring houses
  – For each cluster the # of needed fire brigades has to be computed

• Problem: How to assign fire brigades to fire clusters efficiently?
  – Auctions are problematic due to communication constraints of the domain
  – Coalition formation
    • Is the problem is super additive?
    • Plays the sequence an important role?

• Some more problems:
  – Some fires are more dangerous than others due to their fireyness
  – Some fires can be much faster extinguished than others due to size and material of the building
  – It is advantageous to prefer “border fires” in order to stop fire spread
  – Logistics: How to optimally place fire brigades around fires in order to avoid that they block each other?

• Maybe a “task” for the exercises
ResQ Freiburg task allocation
Example Animation

Time: 191  Score: 85,881,927
Dynamic Role Assignment
Introduction

• Role assignment is a computational cheap mechanism to efficiently coordinate agents
  – Individual roles are assign according to the team formation
  – Can be applied in domains with $N$ pre-defined tasks and $M$ robots that can potentially be assigned to each task
  – Particularly suited in dynamic domains, such as robot soccer, where the optimal assignment depends on the current world state

• Example domain robot soccer:
  – The goal is to avoid swarm behavior and inference
    • do not attack your own team mates
    • do not get into the way of an attacking or defending robot
  – Task decomposition and task (re-)allocation
    • the player which is closest to the ball should go to the ball
    • If one player cannot do his task, another should take over
  – Joint execution: passing the ball
**Dynamic Role Assignment**

**General Algorithm**

- **Assumptions:**
  - There are $N$ available roles (not necessarily distinct)
  - There is a fixed ordering $\{1, 2, \ldots, N\}$ of the roles. Role 1 must be assigned first, followed by role 2, etc.
  - Each agent can be assigned to only one role
  - The utility $u_{ij}$ reflects how appropriate agent $i$ is for role $j$ given the current state

- **Role assignment algorithm:**

```plaintext
for all agents in parallel
  I := \emptyset; // Committed assignments with ordering
  for each role $j = 1, \ldots, N$
    compute utility $u_{i,j}$; // Own preference of agent $i$
    broadcast $u_{i,j}$; // To all other agents
  end;

  Wait until all $u_{i,j}$ are received //From all the other agents
  for each role $j = 1, \ldots, N$
    assign role $j$ to agent $i^* = \arg \max_{i \neq i} \{ u_{i,j} \}$;
    $I := I \cup \{ i^* \}$; // Add assignment
  end;
end.
```
Case Study: CS-Freiburg
Dynamic roles

- Each player can have one of four roles:
  - **goalie** (fixed)
    - special hardware setup → unable to change its role
  - **active player**: in charge of dealing with the ball
    - can approach the ball or to bring the ball forward towards the opponent goal
  - **strategic player**: defender
    - maintains a position back in its own half
  - **supporter**: serves the team
    - in defensive play it complements the team’s defensive formation
    - in offensive play it presents itself to receive a pass close to the opponents goal
Case Study: CS-Freiburg
Role Utilities

• Placement: each role has a preferred location, which depends on the situation:
  – ball position, position of team mates and opponents
  – defensive situation or attack
  – computed by potential fields

• Utility for each role:
  – “Negative utility (costs)” for reaching the preferred location of the role
  – Costs are computed from partial costs for distance \( u_d \), turn angle \( u_t \), objects on the path \( u_o \)
  – Weighted sum to ensure utilities between 0..1: \( U_{ij} = w_d u_d + w_t u_t + w_o u_o \)
Case Study: CS-Freiburg
Dynamic Role Assignment

• Each player computes the utility for each role and broadcasts it to the other players
• Given all utilities, each player tries to maximize the group utility
  – under the assumption that all team members do the same
• Group utility:
  – Consider all possible assignments and compute the summed utility from each agents’ individual utility for its assigned role
  – Take the assignment with the highest utility sum as solution
• Roles are reassigned only when
  – the role change is significant, i.e. the new utility >> old utility (hysteresis factor to avoid oscillation)
  – two players agree (by communication)
• Note that opinion about global position can differ (even with a global world model)
  – Agents might “lie” without intention
Attack against Osaka (Japan). The attacking robot is blocked by a defender and consequently replaced by an unblocked player.
Case Study: CS-Freiburg
Example for Role Switching II

Defense against Artisti Veneti (Italy).
The roles active and strategic player are switched a couple of times.
Case Study: CS-Freiburg
Joint Execution: A Pass . . . that was Unsuccessful

A pass in the semi-final against the Italian ART Italy team (RoboCup 1999). This was based on standard plan: “if it is not possible to score directly, wait until supporter arrives, then make the pass”
Case Study: CS-Freiburg
Demo Webplayer

See www.cs-freiburg.de
Summary

• Action selection and coordination are essential when acting in groups
  – If implemented efficiently, you can win a robotic soccer or rescue agent world championship

• Coalition formation is the process of finding the “social welfare” coalition structure among a set of agents
  – The search can be computational expensive when dealing with more than 15 agents
  – In practice, domain dependent heuristics are necessary for pruning the search tree (i.e. constraining the split and merge arcs)

• Dynamic role assignment is an efficient and cheap method for team coordination
  – However, the protocol requires truthful participants
  – Due to world model inconsistencies, this assumption can be violated
Literature


• Gerhard Weiss *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, pages 201-258
