Game theory and Personality Traits in Task Scheduling
Problem of Swarm Robot Systems

Jieke SHI
School of Information Engineering, Yangzhou University
10 Jan 2020
Outline

- Preliminaries
- Modeling Using Game Theory
- Personality Traits
- Convergence Proof
- Simulation Results
- Future Work
Preliminaries

- Continuous damage of tasks
- Centralized scheduling strategy
- Real-time decentralized scheduling strategy
Centralized scheduling strategy

A dynamic auction approach for differentiated tasks under cost rigidities (DAACR)

\[
\begin{align*}
\min & \quad \sum_{i \in B} \sum_{j \in T_2} q_{ij} c_{ij}^i \\
\text{s.t.} & \quad \sum_{j \mid (i, j) \in D} q_{ij} = 1, \forall i = 1, 2, \ldots, n \\
& \quad \sum_{i \mid (i, j) \in D} q_{ij} = 1, \forall j = 1, 2, \ldots, n \\
& \quad q_{ij} = 0, 1, \forall (i, j) \in D \\
\end{align*}
\]

\[
|p_j - c_{ij}^i - \max_{k \in D(i)} \{p_k - c_{ij}^k\}| \leq \epsilon
\]

Fig. 1 Time cost comparison between Auction algorithm and Hungarian algorithm
Centralized scheduling strategy

Step 3: bidding phase All robots bid for $j_i$ which is the most gainful task, the bidding price of the robot is determined as:

$$a_{ij_i} = p_{ji} - u_i + v_i - \epsilon = cost^j_i + v_i - \epsilon$$  \hspace{1cm} (15)

Step 4: allocation phase For each task $j$, if the $B(j)$ is not empty, we update the price of the task $j$ to the highest bid price:

$$p_j = \max_{i \in B(j)} a_{ij}$$  \hspace{1cm} (16)

The task is assigned to the highest bidding robot $i_j$, and at the same time, the two-tuples related to the robot $i_j$ and the task $j$ are removed from the $A$ which is an infeasible allocation, and the new two-tuples ($i_j, j$) are added to the set $A$. 
Centralized scheduling strategy

We define that the optimal solution will minimize the total damage suffered by the system during the whole progress. This can be represented as an integer programming, the formula is that:

\[
\min \sum_{i=0}^{t_a} \sum_{j=1}^{M} H(w_j^t)
\]

s.t. \( \sum_{j=1}^{M} C_j^t \leq N \)

\[
\sum_{j=1}^{M} \sum_{i=1}^{c_j^t} c_i \leq \sum_{j=1}^{M} w_j^t
\]

\[
w_j^{t+1} = w_j^t - \sum_{i=1}^{c_j^t} c_i
\]

c_i is a no-negative constant, \( \forall i \),

\( C_j^t \) is a no-negative integer, \( \forall j, \forall t \)

According to Definition 1, the local optimal solution can be determined by computing the following integer programming.

\[
\min \sum_{j=1}^{M} H(w_j^t)
\]

s.t. \( \sum_{j=1}^{M} C_j^t \leq N \)

\[
\sum_{j=1}^{M} \sum_{i=1}^{c_j^t} c_i \leq \sum_{j=1}^{M} w_j^t
\]

\( C_j^t \) is a no-negative integer, \( \forall j, \forall t \)
Centralized scheduling strategy

The optimal scheduling for every time unit $t$, that is, the minimization of the damage $H(w_j^t)$, can eventually lead to the global optimal solution, when $\frac{\partial}{\partial t}H(w_j^t) \geq 0$.

The scheduling scheme obtained by the greedy strategy is $C[0, t_d] = \{C_0, C_1, \ldots, C_{t_d}\}$. Assume there exists a better solution $C_*[0, t_d] = \{C_0^*, C_1^*, \ldots, C_{t_d}^*\}$, which differs from the greedy solution. This means that at some time $t_0$ the better solution must assign agents differently than the greedy solution. The scheduling scheme of our greedy algorithm at time $t_0$ is $C^{t_0} = (C_0^{t_0}, C_2^{t_0}, \ldots, C_M^{t_0})$, the better scheduling is $C_*^{t_0} = (C_0^{t_0}, C_2^{t_0}, \ldots, C_M^{t_0})$. By definition of the greedy algorithm, we know $\sum_{j=1}^M H(w_j^{t_0}) \leq \sum_{j=1}^M H_*(w_j^{t_0})$. Thus we prove by induction that $\sum_{t=0}^{t_d} \sum_{j=1}^M H(w_j^{t}) \leq \sum_{t=0}^{t_d} \sum_{j=1}^M H_*(w_j^{t})$.

If $\sum_{j=1}^M H(w_j^{t_0}) < \sum_{j=1}^M H_*(w_j^{t_0})$, then there must exist $\sum_{t=0}^{t_d} \sum_{j=1}^M H(w_j^{t}) < \sum_{t=0}^{t_d} \sum_{j=1}^M H_*(w_j^{t})$. If $\sum_{j=1}^M H(w_j^{t_0}) = \sum_{j=1}^M H_*(w_j^{t_0})$, there also exists $\sum_{t=0}^{t_d} \sum_{j=1}^M H(w_j^{t}) < \sum_{t=0}^{t_d} \sum_{j=1}^M H_*(w_j^{t})$.

So, according to the greedy strategy, we can conclude $\sum_{j=1}^M H(w_j^{t_0+1}) \leq \sum_{j=1}^M H_*(w_j^{t_0+1})$ at the next unit time $t_0 + 1$. 

![Fig. 1. Shows the total damage comparison among three methods.](image1)

![Fig. 2. Shows the time comparison among two methods.](image2)
Real-time decentralized scheduling strategy

- $Ag = \{a_1, a_2, \ldots, a_n\}$ is the set of agents who participate in a negotiation. The negotiation host is not included in $Ag$.
- $Ac = (Ac_1, Ac_2, \ldots, Ac_n)$ is a tuple in which $Ac_i$ represents the action set of negotiation agent $i \in Ag$.
- $W$ is a non-empty set of states. $w_0 \in W$ is the initial state of a negotiation. $T \subset W$ is the set of terminal states $w \in W$.

- $L = (L_1, L_2, \ldots, L_n)$ is a tuple where each $L_i : W \setminus T \rightarrow 2^{Ac_i}$ is the legal function for $a_i$. It determines the actions $a_i$ can take at some state.
- A so-called state update function is defined as $u : W \times Ac_i \rightarrow W$ to represent at what condition, a state will change to another.
- $Agr$ is the set of all possible negotiation agreements. The function $Q : T \rightarrow Agr$ maps each terminal state to an agreement.

---

Algorithm 1: A real-time decentralized scheduling algorithm

```
repeat
  $U(1) = U(-1) = \emptyset$;
  for each $i \in Ag$ do
    propose($u_i(-1), u_i(1)$);
    $U(1) = U(1) \cup u_i(1)$;
    $U(-1) = U(-1) \cup u_i(-1)$;
  repeat
    $i = \text{arg min } U(-1)$;
    delete $i$ from $U(1)$ and $U(-1)$;
    $j = \text{arg max } U(1)$;
    if $u_i(-1) < u_j(1)$ then
      agent $i$ gives a task to agent $j$;
    else
      delete $j$ from $U(1)$ and $U(-1)$;
  until Algorithm Stops
until true
```
Real-time decentralized scheduling strategy

Fig. 4. Comparing the algorithms across three maps under unrestricted communication.

Fig. 7. Comparing the algorithms across Berlin map under restricted communication.
Outline

- Preliminaries
- **Modeling Using Game Theory**
- Personality Traits
- Convergence Proof
- Simulation Results
- Future Work
One of the major successes in the field of economics and social sciences in the past decades has been the application of Game Theory to the modeling of social interactions of rational entities for the prediction of outcomes of conflicts among them.
Prisoner’s dilemma game is a typical model of such interesting scenario. In a prisoner’s dilemma game, each of the players has a set $A = \{C, D\}$ of two available actions, where $C$ denotes Cooperation and $D$ denotes Defection.

For both players, defection $D$ is the only dominant strategy for any rational agent. Thus, mutual defection $(D, D)$ is the unique Nash equilibrium. However, mutual cooperation $(C, C)$ maximizes social welfare, the sum of utilities of both players. Therefore, out of the two possible actions $C$ and $D$ in a PD game, choosing cooperation $C$ is socially beneficial, yet there is always a lure for any agent to exploit its opponent’s cooperation $C$ via playing defection $D$. 
Modeling Using Game Theory

In the case of mixed strategies, the payoff functions are the expectations of the players. The formulation of the problem is a linear system of inequalities and characterizes a linear programming problem. More specifically, the problem becomes linear forms in the probabilities with which the various players play their pure strategies.
Outline

- Preliminaries
- Modeling Using Game Theory
- Personality Traits
- Convergence Proof
- Simulation Results
- Future Work
One of the suggestions of the Theory of Evolution is that animals have emotions. Moreover, these emotions are shared by the same species. Also, traits of personality (term that is used interchangeably with emotions) are important to the maintenance of objectives and collaboration.


**Personality Traits**

**Definition 1.** For an robot $i$, we define its personal traits as

$$\alpha_i = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$$

(1)

and for each one of the personality traits, the reward functions are represented by the vector

$$\beta_i = [\beta_1, \beta_2, \ldots, \beta_n]^T$$

(2)

The reward functions can be arbitrary functions that represent the extent to which personality traits affect the action of the robot. When the action chosen by the agent is $\gamma$, its utility function can be formulated as:

$$U(s, \alpha, \gamma) = f\left(\sum_{i=1}^{n} \alpha_i \beta_i(s_t, \gamma, t)\right)$$

(3)

In addition, we need to consider the evolution process of some personality traits, the dynamics of the personality trait vector can be described by the following general difference equation,

$$\alpha_{t+1} = \alpha_t + \eta \mathcal{H}(\alpha_t, \beta_t)$$

(4)

where the function $\mathcal{H}(\alpha_t, \beta_t)$ depends on the application under consideration.
Outline

- Preliminaries
- Modeling Using Game Theory
- Personality Traits
- Convergence Proof
- Simulation Results
- Future Work
**Definition 2.** Define \( x = [x_1, x_2, \ldots, x_n] \), \( n \) denotes the numbers of strategies.

\( \Delta(n) \) denotes the simplex in \( \mathbb{R}^n \), i.e., \( \{ x \in \mathbb{R}^n | x_i \geq 0, \forall i = 1, \ldots, n; \text{ and } \sum_{i=1}^{n} x_i = 1 \} \).

\( \text{Int}(\Delta(n)) \) denotes the set of interior points of a simplex, i.e., \( \{ x \in \Delta(n) | x_i > 0, \forall i = 1, \ldots, n; \} \).

\( v_i \in \Delta(n) \) denotes the \( i_{th} \) vertex of the simplex \( \Delta(n) \), i.e., the vector whose \( i_{th} \) term equals 1 and remaining terms equal 0, \( \{ x \in \Delta(n) | x_i = 1 \text{ and } x_j = 0, \forall j \neq i \} \).
For a one zero-sum game, the utility function of player $P_1$ is

$$U_1(p_1, p_2) = p_1^T M p_2$$

(5)

For player $P_2$,

$$U_2(p_2, p_1) = -p_2^T M^T p_1$$

(6)

Define the best response mapping for player $P_1$ as

$$\varepsilon_1(p_2) = \arg \max_{x \in \Delta(n)} U_1(p_1, p_2)$$

(7)

For player $P_2$,

$$\varepsilon_2(p_1) = \arg \max_{x \in \Delta(m)} U_2(p_2, p_1)$$

(8)

Define the empirical frequency, $q_i(k)$. The empirical frequency of player $P_i$ can be calculated as the running average of the actions of player $P_i$, that is

$$q_1(k) = q_1(k-1) + \frac{1}{k}(v_{a_1(k-1)} - q_1(k-1))$$

(9)

And,

$$q_2(k) = q_2(k-1) + \frac{1}{k}(v_{a_2(k-1)} - q_2(k-1))$$

(10)

Mapping personality traits to the strategy space by transaction matrix $A$,

$$q_1 = A_1 \alpha_1$$

(11)

$$q_2 = A_2 \alpha_2$$

(12)
The functions used to update the personality traits are

\[ H_1(\alpha_1, \beta_1) = \beta_1 - \alpha_1 \]  
(15)

\[ H_2(\alpha_2, \beta_2) = \beta_2 - \alpha_2 \]  
(16)

Then the personality dynamics are

\[ \tilde{\alpha}_1 = A_1^T (A_1 A_1^T)^{-1} \varepsilon_1(p_2) - \alpha_1 \]  
(17)

\[ \tilde{\alpha}_2 = A_2^T (A_2 A_2^T)^{-1} \varepsilon_2(p_1) - \alpha_2 \]  
(18)

Using these definitions, we can find the strategy dynamics for player \( P_1 \) is

\[ \tilde{q}_1(t) = A_1 \tilde{\alpha}_1(t) \]
\[ = A_1 (A_1^T (A_1 A_1^T)^{-1} \varepsilon_1(p_2) - \alpha_1) \]  
(19)

\[ = \varepsilon_1(q_2(t)) - q_1(t) \]

In the same way, the strategy dynamics for player \( P_2 \) is

\[ \tilde{q}_2(t) = \varepsilon_2(q_1(t)) - q_2(t) \]  
(20)
We can know the best strategy is that
\[
\max_{x \in \Delta(n)} U_1(x, q_2) = (\varepsilon_1(q_2))^T M q_2
\]  \hspace{1cm} (22)

So, we can get
\[
U_1(q_1, q_2) = (\varepsilon_1(q_2) - q_1)^T M q_2 \hspace{1cm} \text{(23)}
\]

In the same way,
\[
U_2(q_2, q_1) = -(\varepsilon_2(q_1) - q_2)^T M^T q_1 \hspace{1cm} \text{(24)}
\]

And note that by definition $U_1(q_1, q_2) \geq 0$ and $U_1(q_2, q_1) \geq 0$ with equality if and only if $\varepsilon_2(q_1) = q_2$ and $\varepsilon_1(q_2) = q_1$.

**Lemma 1.** Define $\tilde{U}_1(t) = U_1(q_1(t), q_{-i}(t))$, then $\tilde{U}_1(t) = -\tilde{U}_1(t) + \tilde{q}_1^T M \tilde{q}_2$, $\tilde{U}_2(t) = -\tilde{U}_2(t) + \tilde{q}_2^T M^T \tilde{q}_1$.

**Theorem 1.** The solutions of equ.19 and equ.20 satisfy
\[
\lim_{t \to \infty} (q_1(t) - \varepsilon_1(q_2(t))) = 0 \quad \text{and} \quad \lim_{t \to \infty} (q_2(t) - \varepsilon_2(q_1(t))) = 0.
\]

**Proof 2.** By Lemma 1, we can get $\tilde{U}_1(t) \leq -\tilde{U}_1(t) + \tilde{q}_1^T M \tilde{q}_2$, $\tilde{U}_2(t) \leq -\tilde{U}_2(t) + \tilde{q}_2^T M^T \tilde{q}_1$. Define function $U_{12}(t) = \tilde{U}_1(t) + \tilde{U}_2(t)$. Taking its derivative, we can get $\dot{U}_{12}(t) = \dot{\tilde{U}}_1(t) + \dot{\tilde{U}}_2(t) = -\tilde{U}_1(t) - \tilde{U}_2(t)$. Since $\tilde{U}_2 \geq 0$ and $\tilde{U}_2 \geq 0$ with equality only at the equilibrium point of equ.19 and equ.20, which means $\dot{U}_{12}(t) \leq 0$ is with equality only if and only if at the equilibrium point of equ.19 and equ.20, $U_{12}$ is a Lyapunov function and this theorem follows from standard Lyapunov arguments.
Outline

- Preliminaries
- Modeling Using Game Theory
- Personality Traits
- Convergence Proof
- Simulation Results
- Future Work
Algorithm 1 Zero-sum game.

**Input:** $n = 0.1$, $t = 0.01$

**Output:** Mixed strategy

1: Set the number of personality traits for player $A$ and player $B$;
2: Initialize the personality traits $\alpha$
3: Define the payoffs and mappings from the personality traits spaces to the strategies spaces of the game.
4: Players initialize the empirical frequency as $0$.
5: $\textbf{for } i = 1 \textbf{ to } 300 \textbf{ do }$
6: \hspace{1em} Players calculates the strategy according to its probability distribution.
7: \hspace{1em} Both players play their actions and record the payoffs.
8: \hspace{1em} Update personality traits and empirical frequency of players.
9: $\textbf{end for}$

Example of a zero-sum game

<table>
<thead>
<tr>
<th></th>
<th>head</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>(-1, 1)</td>
<td>(1, -1)</td>
</tr>
<tr>
<td>tail</td>
<td>(1, -1)</td>
<td>(-1, 1)</td>
</tr>
</tbody>
</table>

![Graph showing mixed strategies over episodes](image)
Example of a Rock-Paper-Scissors Game

<table>
<thead>
<tr>
<th></th>
<th>Rock</th>
<th>Paper</th>
<th>Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>(0, 0)</td>
<td>(-1, 1)</td>
<td>(1, -1)</td>
</tr>
<tr>
<td>Paper</td>
<td>(1, -1)</td>
<td>(0, 0)</td>
<td>(-1, 1)</td>
</tr>
<tr>
<td>Scissors</td>
<td>(-1, 1)</td>
<td>(1, -1)</td>
<td>(0, 0)</td>
</tr>
</tbody>
</table>
Outline

- Preliminaries
- Modeling Using Game Theory
- Personality Traits
- Convergence Proof
- Simulation Results
- Future Work
Humans have norms, but do agents / robots have them?

A straightforward example of this is when two drivers arrive at an intersection simultaneously from neighboring streets. While each player has the incentive of not yielding, myopic decisions by both can lead to undesirable accidents. Both drivers yielding, however, also creates inefficiency. Ideally, we would like norms like “yield to the driver on right”, which serves all drivers in the long run.
Humans have norms, but do agents / robots have them?

Behavioral norms are key ingredients that allow agent coordination where societal laws do not sufficiently constrain agent behaviors. Whereas social laws need to be enforced in a top-down manner, norms evolve in a bottom-up manner and are typically more self-enforcing. While effective norms can significantly enhance performance of individual agents and agent societies, there has been little work in multiagent systems on the formation of social norms.
for a fixed number of epoch do
  repeat
    remove randomly agents $p_{row}$ and $p_{col}$ from the population ask each agent to select an action;
    send the joint action to $p_{row}$ and $p_{col}$ for policy update;
  until all agents have been selected during the epoch;

(a) social dilemma game

<table>
<thead>
<tr>
<th>$G$</th>
<th>$Y_L$</th>
<th>$G$</th>
<th>$Y_R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1, -1</td>
<td>3, 2</td>
<td>0</td>
<td>4, 4</td>
</tr>
<tr>
<td>2, 3</td>
<td>1, 1</td>
<td>-1, -1</td>
<td>4, 4</td>
</tr>
</tbody>
</table>

(b) coordination game

(c) and (d) illustrations of policy update
Game theory and Personality Traits in Task Scheduling Problem of Swarm Robot Systems

Jieke SHI
School of Information Engineering, Yangzhou University
10 Jan 2020