LECTURE 28: TASK ALLOCATION 1

TEACHER: GIANNI A. DI CARO
(Centralized) Models of Task Allocation

Team Mission

Decomposition in sub-tasks

Who does what? (and when, how)
Optimizing team performance

Team resources and status

Dependencies (tasks, agents)
Task Allocation in Robots

A set of $n$ mobile Robots: $R$

$A: T \rightarrow R$

A set of $nt$ surveillance tasks: $T$
**Example: Customer Service**

Routing
(performance metric + constraints)

Customer Assignment
(performance metric + constraints)
TA ↔ Division of labor, Specialization, Social organization, Role switching
DIVISION OF LABOR IN SOCIAL INSECTS

Queen: reproduction

Workers: everything else

Age polyethism: age-dependent division of labor

Age-related activities of workers

First 2 days:
- cell cleaning

3-10 days:
- queen care
- nursing (feeding young)
- wax work

15-20 days:
- wax work
- nectar processing
- guarding
- undertaking

21-35 days:
- foraging (water, nectar, pollen, propolis)
- colony defense (soldiering)
RECRUITMENT, COALITION-MAKING

Coalition:
Group recruitment

Convergent stigmergy

Mass recruitment

Waggle dance:
Recruitment for foraging
**INTENTIONAL VS. EMERGENT**

- **Explicit/intentional TA:** agents explicitly cooperate and tasks are explicitly assigned to agent.

- **Emergent TA:** tasks are assigned *as the result* of local interactions among the agents and with the environment.
MRTA: A Formal Definition (Optimization)

Given:
✓ A set of tasks, \( T \)
✓ A set of robots / agents, \( R \)
✓ \( R = 2^R \) is the set of all possible robot sub-teams
  E.g., \((r_1 = 0, r_2 = 0, r_3 = 1, r_4 = 0, r_5 = 1)\)
✓ A robot sub-team utility (or cost) function: \( U_\rho : 2^T \times R \rightarrow \mathbb{R} \cup \{\infty\} \) (the utility/cost sub-team \( \rho \) incurs by handling a subset of tasks)

✓ An allocation is a function \( A : T \rightarrow R \) mapping each task to a subset of robots. \( R^T = (2^R)^T \) is the set of all possible allocations of tasks to subset of robots

Find:
✓ The allocation \( A^* \in R^T \) that maximizes (minimizes) a global, team-level utility (objective) function \( U : R^T \rightarrow \mathbb{R} \cup \{\infty\} \)
Utility function

- Utility function for a pair \((\text{robot}, \text{task})\)

\[ U_{rt} = \begin{cases} Q_{rt} - C_{rt} & \text{if } r \text{ is capable of executing } t \\ -\infty & \text{otherwise} \end{cases} \]

- \(Q\) and \(C\) are somehow estimates of Quality and Cost that account for all uncertainties, missing information, ...

- **Optimal allocation**: Optimal based on all the available information
  \(\rightarrow\) Rational decision-making

- For some problems, an agent’s (sub-team’s) utility for performing a task is independent of its utility for performing any other task.

- In general, this is not always true, utility depends, for instance, on the order performing the tasks

- Our basic definition fails capturing dependencies 😞
EXAMPLE: CUSTOMER SERVICE

Routing
(performance metric + constraints)

Customer Assignment
(performance metric + constraints)

Order / Routing matters!
Basic Taxonomy

(Gerkey and Mataric, 2004)

- **Task type**: does the task requires one or more robots?
- **Robot type**: is the robot capable of executing one or more tasks at a time?
- **Task information type**: is the available information only permitting an instantaneous allocation of tasks to robots, or can we manage to set-up time-extended schedules for each robot accounting for what will happen (in terms of tasks to deal with) in the future?
Simplifying assumptions (we will try to relax them a bit later on):

- Individual tasks have independent robot utilities
- Tasks have no dependencies
- Tasks have no time-dependencies (i.e. schedule of task execution is not constrained, task assignments can be sets, unordered)
**Basic Taxonomy: IA vs. TA**

- IA vs. TA distinction is a bit blurred
- IA is the case when the robots are only concerned with the task(s) they are handling at the moment and need not / cannot plan for future task allocations
- TA is the case when global information is available, such that a plan for future allocations can be built by defining a schedule
- In the IA case, no restrictions on how many tasks the robot will handle, overall, can be included in the reasoning for task assignment
- Since information about future tasks is not available, in the IA case it makes sense to distribute the available tasks (the load) among all robots
- This is also a way to avoid to reason on the ‘capacity’ of the robot handling multiple tasks over time
- In the TA case, since we have a precise view of what is and will happen in terms of tasks to deal with, limitations on what a robot can do in terms of performing multiple tasks can and need to be taken into account
Why a Taxonomy?

- A lot of “different MR scenarios”
- A lot of “different” MRTA methods
- Analysis and comparisons are difficult!

- Taxonomy → Single out core features of a MRTA scenario
- Allow to understand the complexity of different scenarios
- Allow to compare and evaluate different approaches
- A scenario is identified by a 3-vector (e.g., ST-MR-TA)
### Assignment

- Assign $n$ jobs to $n$ agents minimizing the overall cost of the assignment
- Perfect matching in a weighted bipartite graph
- **1-1 Task / Job / Area / Partner ... allocation**

\[
\begin{array}{c|cccc}
& 1 & 2 & \ldots & n \\
\hline 
1 & d_{11} & d_{12} & \ldots & d_{1n} \\
2 & d_{21} & d_{22} & \ldots & d_{2n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
n & d_{n1} & d_{n2} & \ldots & d_{nn} \\
\end{array}
\]

\[
\begin{align*}
\text{min} & \quad z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \\
& \quad \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \\
& \quad x_{ij} \in \{0, 1\}
\end{align*}
\]

Polynomial solution (Hungarian algorithm)
If $|R| = |T|$ the task assignment problem becomes a **linear assignment** and a polynomial-time solution does exist!

$$\max_{x_{rt}} \sum_{r=1}^{R} \sum_{t=1}^{T} U_{rt} x_{rt}$$

subject to

$$\sum_{r=1}^{R} x_{rt} = 1 \quad t = 1, \ldots, |T|$$

$$\sum_{t=1}^{T} x_{rt} = 1 \quad r = 1, \ldots, |R|$$

$x_{rt} \in \{0, 1\}$

The Hungarian algorithm has complexity $O(|T|^3)$

- In a **centralized architecture**, with each robot sending its $|T|$ utilities to the controller, $O(|T|^2)$ messages are needed

Assignment with hundreds/thousands of robots in $< 1s$

- Note that we **enforce that each robot gets one task**: this may not be necessarily the solution with the best utility or even best time span, but it allows to balance the load, optimize robustness, favor parallelism, and solve the problem in polynomial time 😊
Tasks can be prioritized, meaning that they are priority weights $w_i, i = 1, \ldots, |T|$ that define the importance of a task.

The weight can be added as a coefficient in the objective function:

$$\max \sum_{r=1}^{\lvert R \rvert} \sum_{t=1}^{\lvert T \rvert} R_{rt} x_{rt} w_t$$

s.t. $$\sum_{r=1}^{\lvert R \rvert} x_{rt} = 1 \quad t = 1, \ldots, \lvert T \rvert$$

$$\sum_{t=1}^{\lvert T \rvert} x_{rt} = 1 \quad r = 1, \ldots, \lvert R \rvert$$

$x_{rt} \in \{0, 1\}$
ST-SR-IA: $|R| \neq |T|$ 

- What if $|R| \neq |T|$?

- To preserve polynomial time solution, “dummy” robots or tasks can be included in a two-step process: two IA batches under the same assumptions as before.

- If $|R| < |T|$: $(|T|-|R|)$ dummy robots are added and given very low utility values with respect to all tasks, such that that their assignment will not affect the optimal assignment of $|R|$ tasks to the “real” robots.

- The remaining $|T|-|R|$ tasks (i.e., assigned to the dummy robots) can be optimally assigned in a second round, which will likely feature # of robots greater than the # of tasks.

- If $|T| < |R|$: Dummy tasks with very low, flat, utilities are introduced such that their assignment will not affect the assignment of real tasks.
ST-SR-IA: Iterated Assignment

- Not always full/final task information and utility information is available since the beginning of the operations, but rather it gets revised over time.

- New / revised evidence (utility) \(\rightarrow\) Iterated assignment problem

- There are no new tasks, but their utility or properties change over time.

- Recompute from scratch to solve the assignment, or, adapt greedily:

- Broadcast of Local Eligibility (BLE, 2001), worst-case 50% opt: each robot keep broadcasting its utilities (that can change over time) and assigns itself to the best task match, if any task is still in need:

  1. If any robot remains unassigned, find the robot-task pair \((r, t)\) with the highest utility. Otherwise, quit.
  2. Assign robot \(r\) to task \(t\) and remove them from consideration.
  3. Go to step 1.

  - 2-competitive: \(U(\text{BLE}) \geq c \cdot U(\text{OptOffline}) - a, \quad c = 2\)
  - L-ALLIANCE (1998) can learn the best assignments over time.
Cooperative multi-robot observation of multiple moving targets (hard to predict): when presented with new sensor inputs (e.g., camera images) and consequent utility estimates (e.g., perceived distance to each target), the system must decide which robot should track which target.

- Homogeneous team $\rightarrow$ Robots are interchangeable $\rightarrow$ it is often advantageous to allow any player to take on any role within the team based on scenarios
- Iterated assignment problem in which the robots’ roles are periodically reevaluated, usually at a frequency of about 10 Hz.
Tasks are revealed one at-a-time
If robots can be reassigned, then solving each time the linear assignment provides the optimal solution, otherwise:

MURDOCH (2002)

- When a new task is introduced, assign it to the most fit robot that is currently available.

- Greedy
- 3-competitive

Performance bound is the best possible for any on-line assignment algorithm (Kalyanasundaram, Pruhs 1993): without a model of the tasks that are to be introduced, and without the option of reassigning robots that have already been assigned, it is impossible to construct a better task allocator than MURDOCH.
The “budget” constraints restrict the max number $T_r$ of tasks (or the total time/energy to execute them based on some cost parameter $c$) that can be assigned to robot $r$ (NP-hard).
ST-SR-TA: Generalized Assignment

\[
\begin{align*}
\max & \sum_{r=1}^{\mid R \mid} \sum_{t=1}^{\mid T \mid} U_{rt} x_{rt} \\
\text{s.t.} & \sum_{t=1}^{\mid T \mid} c_{rt} x_{rt} \leq T_r & r = 1, \ldots, \mid R \mid \\
& \sum_{r=1}^{\mid R \mid} x_{rt} = 1 & t = 1, \ldots, \mid T \mid \\
& x_{rt} \in \{0, 1\}
\end{align*}
\]

Approximated solution (not all tasks are jointly assigned):

1. Optimally solve the initial $R \times R$ assignment problem
2. Use the Greedy algorithm to assign the remaining tasks in an online fashion, as the robots become available.

Bound by 3-competitive greedy: as $(\mid T \mid - \mid R \mid)$ goes to zero, gets optimal
MT-SR-IA: Generalized Assignment

\[
\begin{align*}
\text{max} & \quad \sum_{r=1}^{\left| R \right|} \sum_{t=1}^{\left| T \right|} U_{rt} x_{rt} \\
\text{s.t.} & \quad \sum_{t=1}^{\left| T \right|} c_{rt} x_{rt} \leq T_r \quad r = 1, \ldots, \left| R \right| \\
& \quad \sum_{r=1}^{\left| R \right|} x_{rt} = 1 \quad t = 1, \ldots, \left| T \right| \\
& \quad x_{rt} \in \{0, 1\}
\end{align*}
\]

- The “capacity” constraint explicitly restricts the max number \( T_r \) of tasks that robot \( r \) can take, this time simultaneously.

- Not common in the literature instances from MRTA, but nowadays robots and robot architectures are more and more capable of supporting MT scenarios.

NP-hard!