Distributed Optimization

(Based on Shoham and Leyton-Brown (2008). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge.)

Leen-Kiat Soh

Introduction

How can agents, in a distributed fashion, optimize a global objective function?

- Distributed: coordination, communication
- Global: local, autonomy vs. coherence, resolution
- Optimal: complexity, how to compute optimality

Why is this difficult?

Introduction | Four Families of Approaches

- Distributed dynamic programming
  - as applied to path-planning problems
- Distributed solutions to Markov Decision Problems (MDPs)
- Optimization algorithms with an economic flavor
  - as applied to matching and scheduling problems
  - auctions and contract nets
- Coordination via social laws and conventions
  - Includes voting

Distributed Dynamic Programming | ADP

- **Asynchronous Dynamic Programming**
- Underlying strategy: **principle of optimality**
  - **If node *x* lies on a shortest path from *s* to *t*, then the portion of the path from *s* to *x* (or, respectively, from *x* to *t*) must also be the shortest paths between *s* and *x* (resp., *x* and *t*).**
  - This allows an incremental divide-and-conquer procedure, also known as *dynamic programming.*
- ***Notes*: It is **complete**, **optimal**, but **not scalable**.

Distributed Dynamic Programming | ADP 2

- The shortest distance from any node i to the goal g as $h^*(i)$.
- The cost for the link between nodes i and j is $w(i,j)$.
- The shortest distance from i to the goal g via a node j neighboring i is:
- $$f^*(i,j) = w(i,j) + h^*(j)$$
- $\boldsymbol{h^*(i) = \min_j f^*(i,j)}$ (by the principle of optimality)

Optimal at every step!

Distributed Dynamic Programming | ADP 3

procedure ASYNCHDP (node i)
if *i is a goal node* **then**
  $h(i) \leftarrow 0$
else
  initialize $h(i)$ arbitrarily (e.g., to ∞ or 0)
repeat
  forall *neighbors j* **do**
    $f(j) \leftarrow w(i, j) + h(j)$
  $h(i) \leftarrow \min_j f(j)$

Distributed Dynamic Programming | LRTA*

- **Learning Real-Time A***
- Here, the agent starts at a given node, performs an operation similar to that of asynchronous dynamic programming, and then ***moves*** to the neighboring node with the shortest estimated distance to the goal, and repeats
- ***Interleave planning and execution***

Why execute while still planning?

Distributed Dynamic Programming | LRTA* 2

procedure LRTA*
$i \leftarrow s$ // the start node
while *i is not a goal node* **do**
  foreach *neighbor j* **do**
    $f(j) \leftarrow w(i,j) + h(j)$
  $i' \leftarrow \arg\min_j f(j)$ // breaking ties at random
  $h(i) \leftarrow \max(h(i), f(i'))$
  $i \leftarrow i'$

Admissibility

Notes: h must be admissible: h never overestimates the distance to the goal, i.e.,. $h(i) \leq h^*(i)$. (WHY?)

- **Complete. Optimal** given enough trials.
- Multiple agents? (1) agents have different ways of breaking ties, and (2) all have access to a shared h-value table

Markov Decision Process (MDP)

- A Markov Decision Process (MDP) is a discrete time *stochastic* control process
  - At each time step, given state s, the decision maker chooses action a that is available in state s
  - With a state transition function, $p(s, a, s')$, the process *probabilistically* transitions into a new state s'
  - This transition gives a reward for that state-action decision: $r(s, a, s')$.
- Given s and a, it is ***conditionally independent*** of all previous states and actions
  - i.e., the state transitions of an MDP meet the Markov property

Why is the Markov property desired?

MDP | Action Selection via Value Iteration

- **Goal**: To maximize the total reward over time
- **Strategy**: By assigning the best possible action to each state
- **Method**: **Value iteration** is the most popular algorithm, to find control policies
- It recursively calculates the utility of each action relative to a reward function

$$Q^{\pi^*}(s,a) = r(s,a,s') + \beta \sum_{s'} p(s,a,s')\, V^{\pi^*}(s')$$

Q-value (utility) of the best policy for the state-action pair of (s,a)

reward

Discount factor

Value of the best policy π^* for s

- Then it updates:

$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s,a)$$

MDP | Action Selection via Value Iteration 2

- The algorithm

$$Q_{t+1}(s,a) \leftarrow r(s,a,s') + \beta \boxed{\sum_{\hat{s}} p(s,a,s')\, V_t(s')}$$

$$V_t(s) \leftarrow \max_a Q_t(s,a)$$

Notes: The future term is the *expected cumulative reward* of state *s*.

- In a multiagent MDP, any (global) action a is really a vector of local actions $(a_1, \dots, a_n)$, one by each of *n* agents.

Q-learning, reinforcement learning!

Think about asynchronous dynamic programming in path-finding

Optimization with Economic Flavor
Negotiations & Auctions

- **From Contract Nets to Auction-Like Optimization**
- A global problem is decomposed into subtasks, and distributed among agents; and each agent has different capabilities
- For each agent *i*, there is a function c_i such that for any set of tasks *T*, $c_i(T)$ is the cost for the agent to achieve all the tasks in *T*
- The agents then enter into a negotiation process which improves on the assignment, and hopefully, culminates in an optimal assignment, that is, one with minimal cost

Furthermore, the process can have a so-called *anytime property*; even if it is interrupted prior to achieving optimality, it can achieve significant improvements over the initial allocation

Optimization with Economic Flavor 2
Negotiations & Auctions

- **Contract net protocol**: contract host and bidders, auctions (Chapter 11)
- **Direct 1-to-N, or multiple 1-to-1 negotiations** (Advanced topics, if time permits)

Optimization with Economic Flavor 3

WHY?

- We start with some global problem to be solved, but then speak about minimizing the total cost to the agents. What is the connection between the two?
    - *Think about autonomy and emergent behavior!*
- When exactly do agents make offers, and what is the precise method by which the contracts are decided on?
    - *Think about utility, future and current rewards, reinforcement learning!*
- Since we are in a cooperative setting, why does it matter whether agents “lose money" or not on a given contract?
    - *Think about incomplete and dynamic environmental properties and optimality!*

Think about why we need an agent-based, distributed solution in the first place!

Optimization with Economic Flavor
Assignment Problem

- A (symmetric) assignment problem consists of
  - A set *N* of *n* agents
  - A set *X* of *n* objects
  - A set $M \subseteq N \times X$ of possible assignment pairs
  - A function $v : M \to \mathbb{R}$ giving the value of each assignment pair
- A feasible assignment *S* is optimal if it maximizes $\sum_{(i,j)\in S} v(i,j)$

Optimization with Economic Flavor

Assignment Problem 2

Implication of this equilibrium?
What is an equilibrium?

- Imagine that each of the objects in *X* has an associated price; the price vector is $p = (p_1, \ldots, p_n)$, where p_j is the price of object *j*.
- Given an assignment $S \subseteq M$ and a price vector *p*, define the "utility" for an assignment *j* to agent *i* as $u(i, j) = v(i, j) - p_j$
- An assignment and a set of prices are in *competitive equilibrium* when **each agent is assigned the object that maximizes his or her utility given the current prices**

Definition 2.3.4. (Competitive Equilibrium). A feasible assignment S and a price vector p are in competitive equilibrium when for every pairing $(i, j) \in S$ it is the case that $\forall k \; u(i, j) \geq u(i, k)$.

Optimization with Economic Flavor

Assignment Problem and Auction

Naive Auction Algorithm

// Initialization:

$S \leftarrow \emptyset$

forall $j \in X$ **do**

  $p_j \leftarrow 0$

repeat

  // Bidding Step:

  let $i \in N$ be an unassigned agent

  // Find an object $j \in X$ that offers i maximal value at current prices:

  $j \in \arg\max_{k|(i,k)\in M}(v(i,k) - p_k)$

  // Compute i's bid increment for j:

  $\boxed{b_i \leftarrow (v(i,j) - p_j) - \max_{k|(i,k)\in M; k\neq j}(v(i,k) - p_k)}$

  // which is the difference between the value to i of the best and second-best objects at current prices (note that i's bid will be the current price plus this bid increment).

  // Assignment Step:

  add the pair (i,j) to the assignment S

  if *there is another pair* (i', j) **then**

    remove it from the assignment S

  increase the price p_j by the increment b_i

until S *is feasible* // that is, it contains an assignment for all $i \in N$

Potential problem?

Why increment this way?

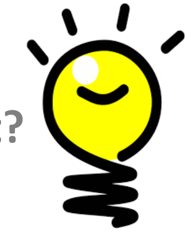
- *What if there are two or more objects offering maximal value for a given agent? The agent's bid increment will be zero*
  - *If these two items also happen to be the best items for another agent, they will enter into an infinite bidding war in which the price never rises*
- To remedy, add a small value:

$$b_i \leftarrow u(i,j) - \max_{k|(i,k)\in M; k\neq j} u(i,k) + \epsilon$$

Social Laws and Conventions

- **Would you drive if there weren’t any traffic rules?**

Why or why not?



Social Laws and Conventions 2

- Consider the task of a city transportation official who wishes to optimize traffic flow in the city. While he or she cannot redesign cars or create new roads, he or she can impose *traffic rules*
- A traffic rule is a form of a ***social law***: a restriction on the given strategies of the agents
  - A typical traffic rule prohibits people from driving on the left side of the road or through red lights
- For a given agent, a **social law presents a tradeoff; it suffers from loss of freedom (think: autonomy!)**, but can benefit from the fact that others lose some freedom
- **A good social law is designed to benefit *all* agents**

Social Laws and Conventions 3

- In general, agents are **free** to choose their own strategies, which they will do based on their guesses about the strategies of other agents
  - Sometimes the interests of the agents are at odds with each other, but sometimes they are not
  - If the interests are perfectly aligned, then the only problem is **coordination** among the agents
    - Traffic presents the perfect example; agents are equally happy driving on the left or on the right, provided everyone does the same
- A social law simply eliminates from a given game certain strategies for each of the agents, and thus induces a *subgame*
- **When the subgame consists of a *single* strategy for each agent, we call it a *social convention***

Can you think of any social convention?
Say, in this classroom?

Social Laws and Conventions 4

- **How** might one find a good social law or social convention?
- **Democratic perspective**
  - How conventions can emerge dynamically as a result of a learning process within the population
  - ***Note****: Learning and Teaching and Voting!*
- **Autocratic perspective**
  - Imagine a social planner imposing a good social law (or even a single convention)
  - The question is how such a benign dictator arrives at such a good social law
  - ***Note****: Mechanism design!*
- The general problem of finding a good social law (under an appropriate notion of “good”) can be shown to be **NP-hard**

Implications for MAS designers?

Connection to MAS?

Distributed: coordination, communication
Global: local, autonomy vs. coherence, resolution
Optimal: complexity, how to compute optimality

Stupid question: What if the “pedestrian crossing” button resets its timing after every time a person presses it?

Stupid question: What if the elevator always goes to the nearest floor on-demand?