

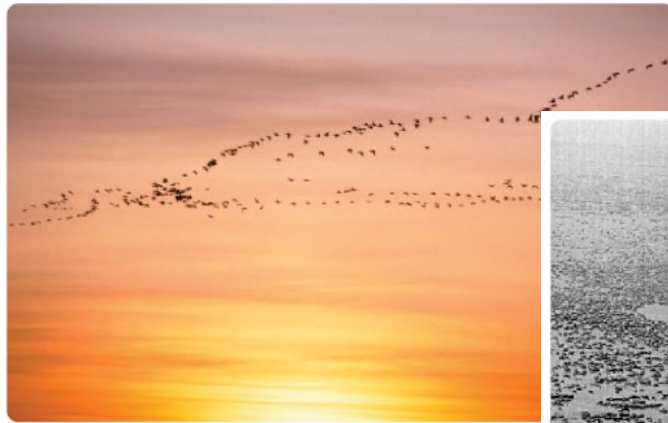
# **Structure and Synthesis of Robot Motion**

## **Multi-robot Coordination and Task Allocation**

**Subramanian Ramamoorthy**  
**School of Informatics**

**12 March, 2012**

# Motion Problems with Many Agents



*What kind of knowledge does any one agent have?  
How does the local knowledge get utilized in a global control strategy?*

# Utility of Self-Interest

- Consider scenarios where it makes sense to have large numbers of relatively simple robots
  - ‘large’ is necessarily subjective
- Although we have discussed ways to control such groups (e.g., potential functions), we have taken a fairly centralized view to problem formulation and asked how that goal can be achieved using factored computation
- A more general approach to decentralization is to allow each agent to be ‘self-interested’
  - What does this mean?
  - What is the effect on the decision making process?

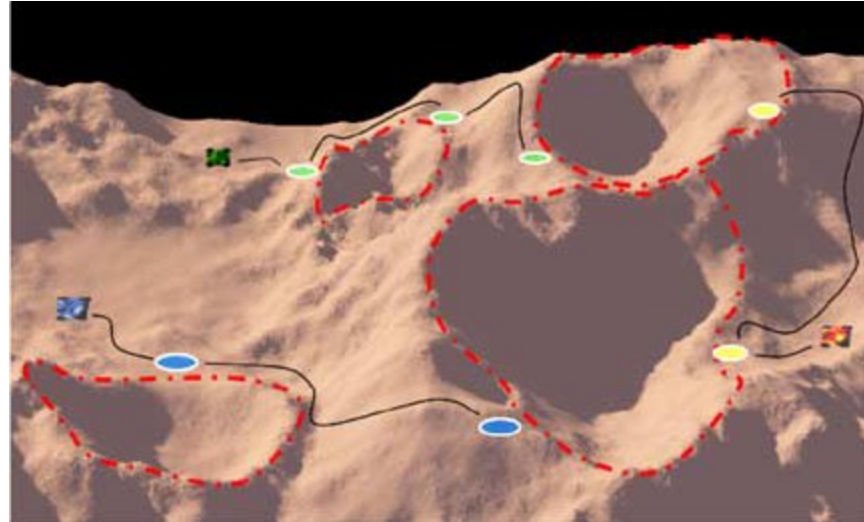
# Market Based Approaches

Overview article:

M.B. Dias, R. Zlot, N. Kalra, A. Stentz, Market-based multirobot coordination: A survey and analysis. *Proc. IEEE* 94(7) : 1257 –1270, 2006.

The following slides are excerpted from a tutorial presentation (at ICRA/AAMAS 2006) based on the same.

# Motivating Example: Robots Exploring on Mars



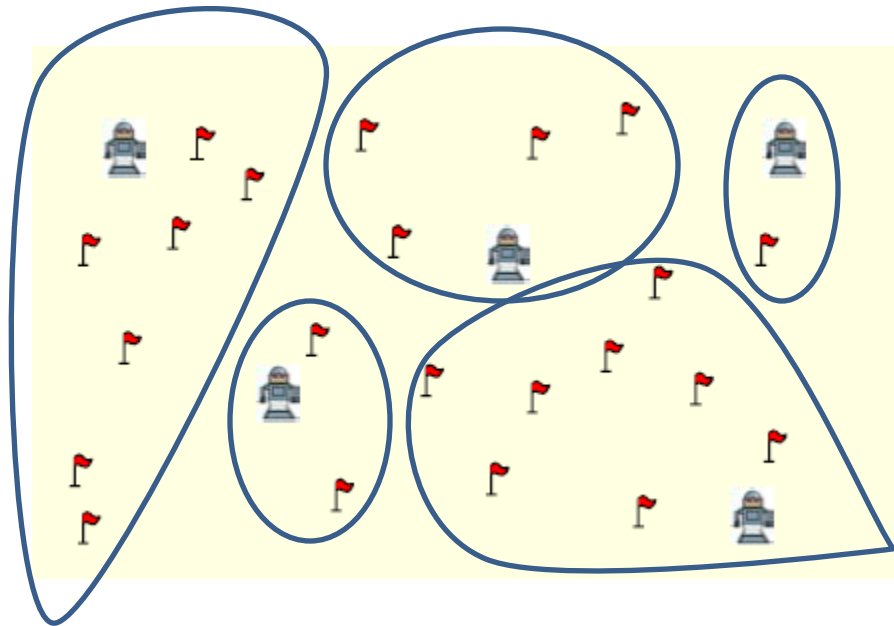
## **Multi-robot routing:**

A team of robots has to visit given targets spread over some known or unknown terrain. Each target must be visited by one robot.

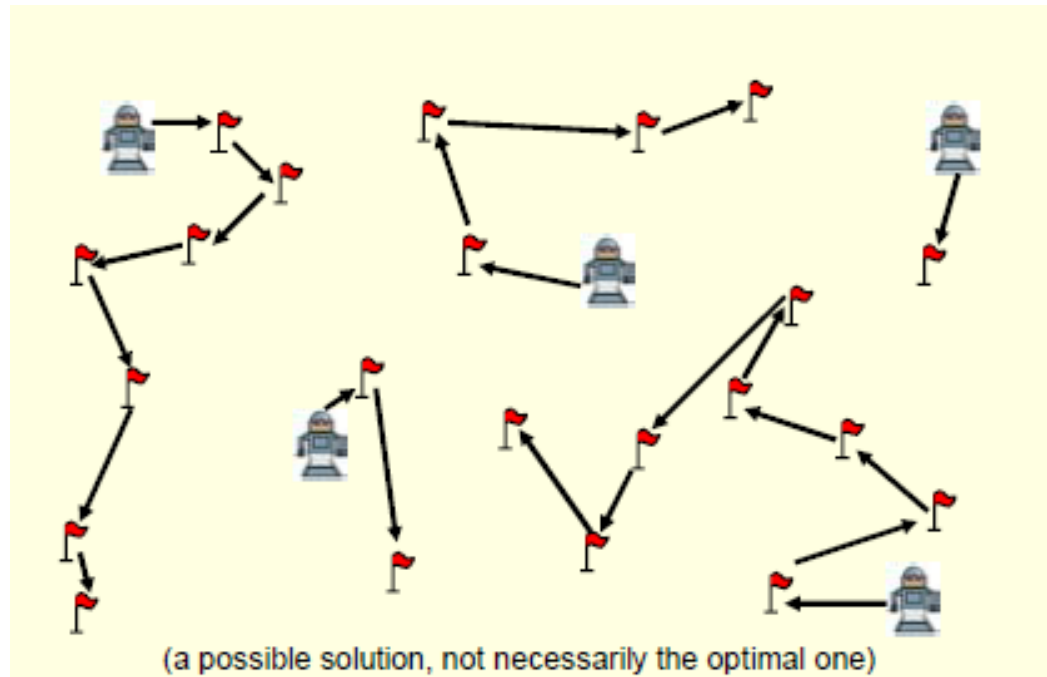
# Multi-robot Routing: Assumptions

- The robots are identical.
- The robots know their own location.
- The robots know the target locations.
- The robots might not know where obstacles are.
- The robots observe obstacles in their vicinity.
- The robots can navigate without errors.
- The path costs satisfy the triangle inequality.
- The robots can communicate with each other.

# Multi-robot Routing

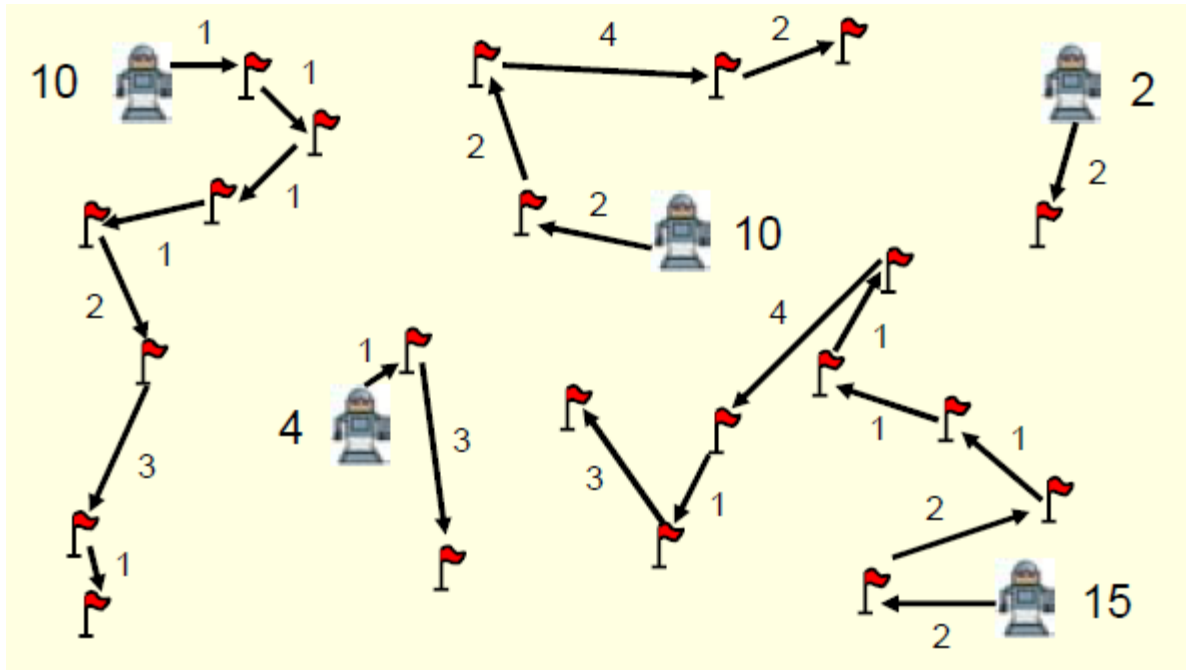


# Multi-robot Routing





# Routing: Minimum Sum Team Objective



$$10+10+2+4+15 = 41$$

# History of Coordination Problem

Multi-robot routing is related to ...

- ... Vehicle/Location Routing Problems

- ... Traveling Salesman Problems (TSPs)

- ... Traveling Repairman Problems

except that the robots ...

- ... do not necessarily start at the same location

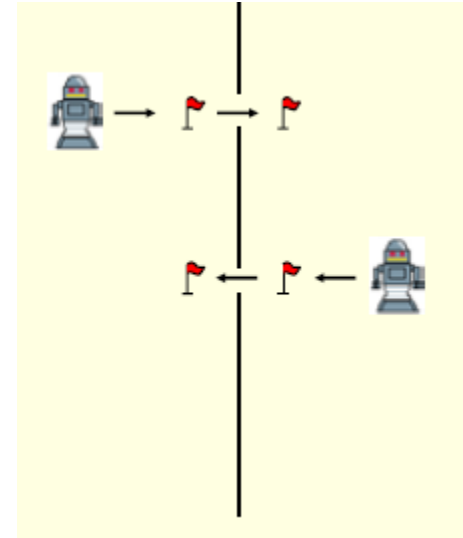
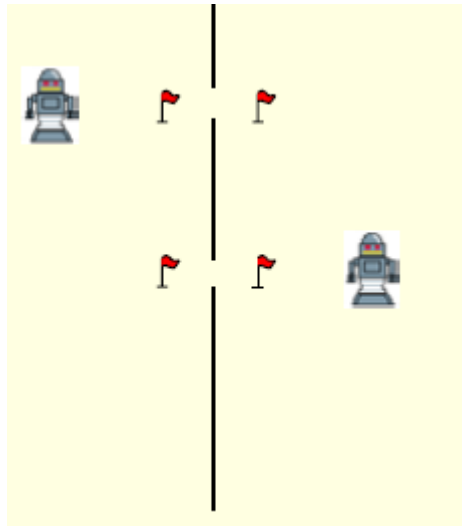
- ... are not required to return to their start location

- ... do not have capacity constraints

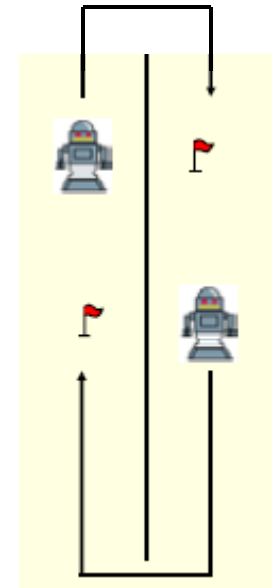
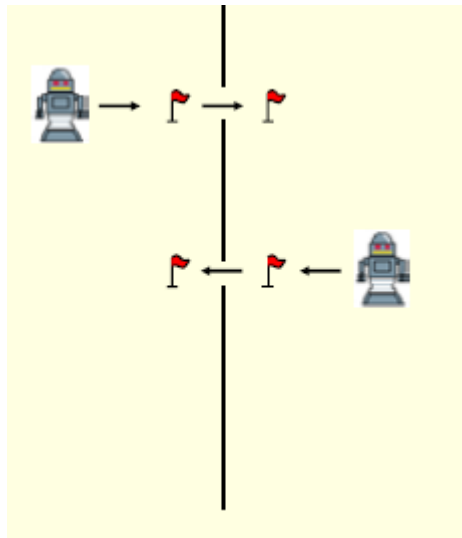
# Auctions for Multi-Robot Coordination

Agent coordination	Auctions
<ul style="list-style-type: none"><li>■ agents</li><li>■ tasks</li><li>■ cost</li></ul>	<ul style="list-style-type: none"><li>■ bidders</li><li>■ items</li><li>■ currency</li></ul>

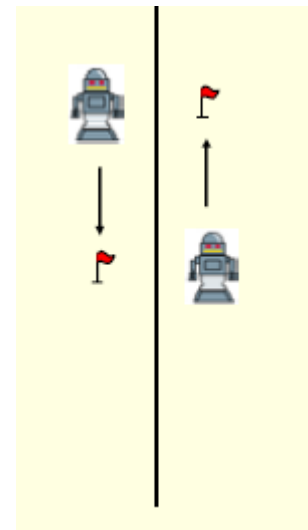
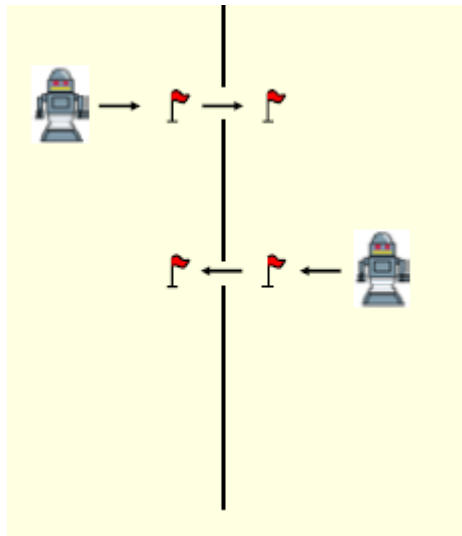
# Auctions for Agent Coordination: Known Terrain



# Auctions for Agent Coordination: Unknown Terrain Plan 1



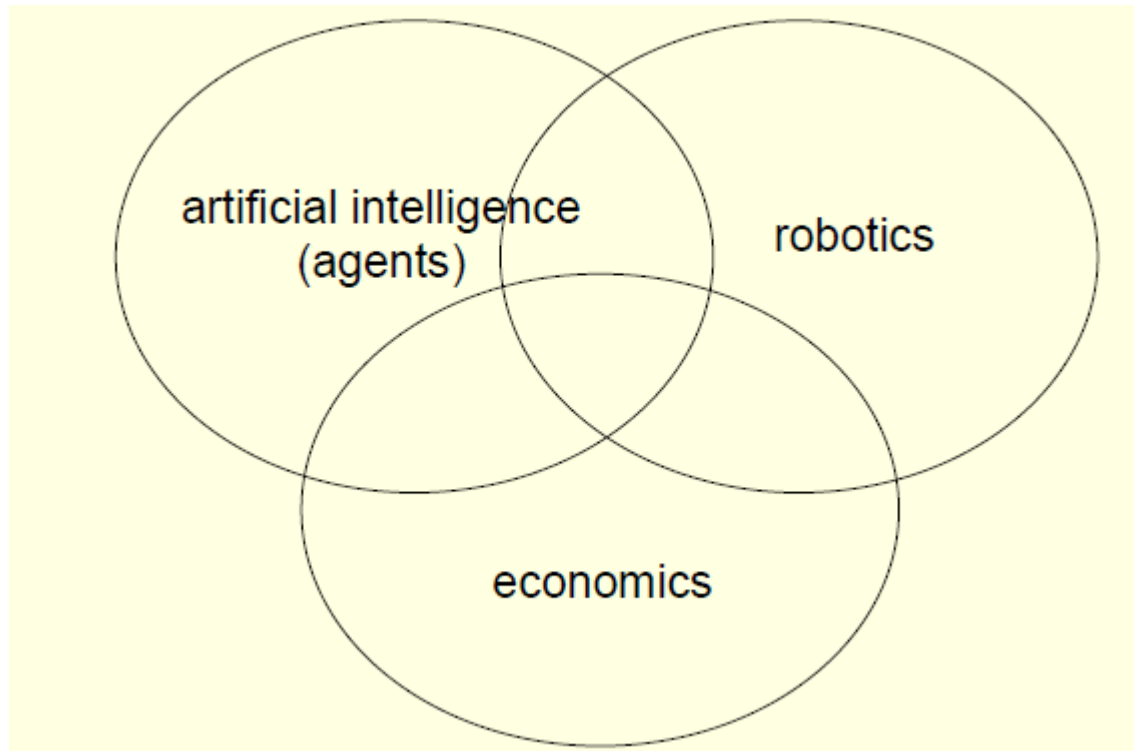
# Auctions for Agent Coordination: Unknown Terrain Plan 2



# Auctions for Agent Coordination

- Auctions are an effective and practical approach to agent-coordination.
- Auctions have a small runtime.
  - Auctions are communication efficient:
    - information is compressed into bids
  - Auctions are computation efficient:
    - bids are calculated in parallel
- Auctions result in a small team cost.
- Auctions can be used if the terrain or the knowledge of the robots about the terrain changes.

# Auctions for Agent Coordination





# What is an Auction?

Definition [McAfee & McMillan, JEL 1987]:

**a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants.**

Examples:

- eBay
- NASDAQ
- Sothebys

# Key Feature: Pricing Mechanism

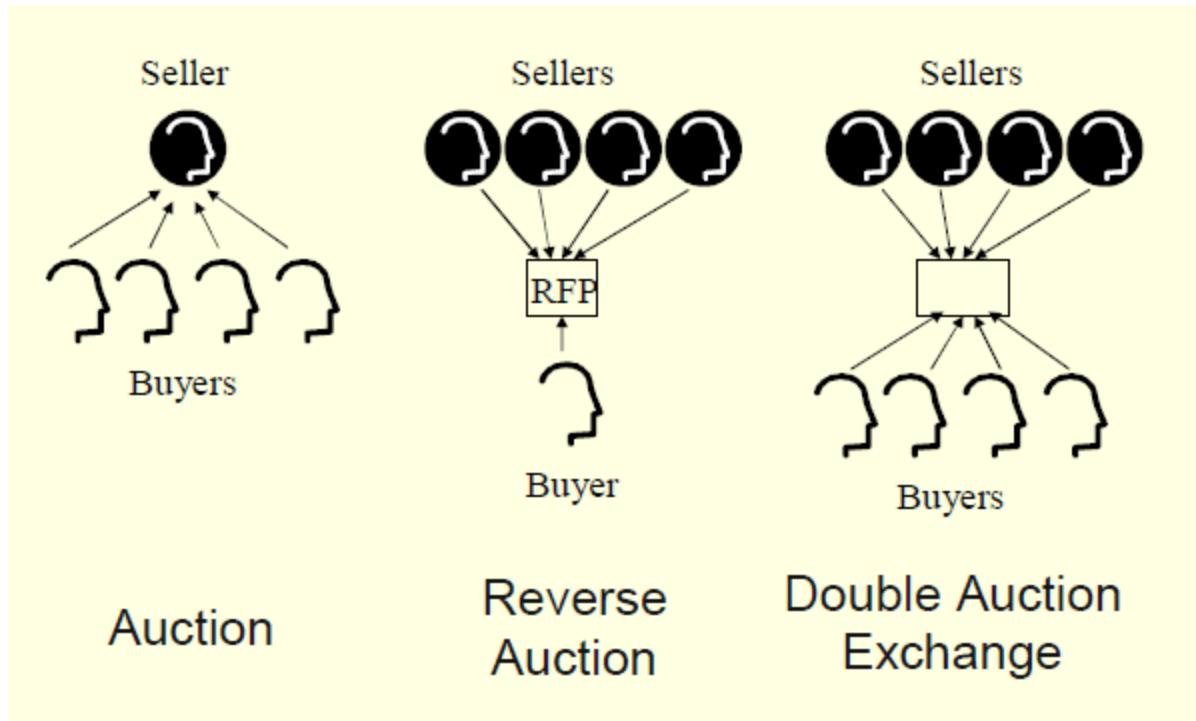
- Posted prices
  - Static
  - Dynamic
    - Change dynamically over time
    - Customized pricing
- Price discovery mechanisms
  - Auctions
  - Negotiations

# Why Auctions?

- For object(s) of unknown value
- Mechanized
  - reduces the complexity of negotiations
  - ideal for computer implementation
- Creates a sense of “fairness” in allocation when demand exceeds supply

*Can you think of robotics scenarios  
with the above characteristics?*

# Auction Formats



# Auction Formats

- What is being auctioned?
  - Private vs. Common valuations
- Who pays and what price do they pay?
  - Does only winner pay? Does she pay what she bid?
- What is the auction format?
  - Closed - Sealed bid - do not know bids of others when placing yours
  - Open - Can see what bids other people make
    - English Auction - has very nice properties
- If multiple units are being auctioned,
  - How are they bundled?
  - In which order are their sales sequenced?

# Key Features: Single vs. Double Sided

- Single-sided auctions
  - A single seller selling to multiple potential buyers.
    - Antique & art auctions, Real estate auctions
  - A single buyer buying from multiple potential sellers.
    - Bidding for government purchasing contracts
    - Carriers bidding for transportation contracts with shippers
    - Catering services bidding for university contracts
- Double-sided auctions
  - Multiple potential buyers and potential sellers are interacting.
    - Stock market
    - Internet exchanges, eg truckload transportation, container exchanges, airline tickets
    - Automobiles, Groceries at Priceline.com

# Key Features: Single vs. Multiple Units

- Single-unit auctions
  - Unique commodity being auctioned
    - Antiques and Art
    - Real estate (depending on situation)
    - Bidding for government purchasing contracts
- Multiple-unit auctions
  - Multiple units of a commodity being auctioned
    - Treasury bonds
    - Corporate stock
    - Electricity Power Exchange
    - Carriers bidding for transportation contracts with shippers
    - Automobile licenses in Singapore

# Key Features: Open vs. Sealed Bids

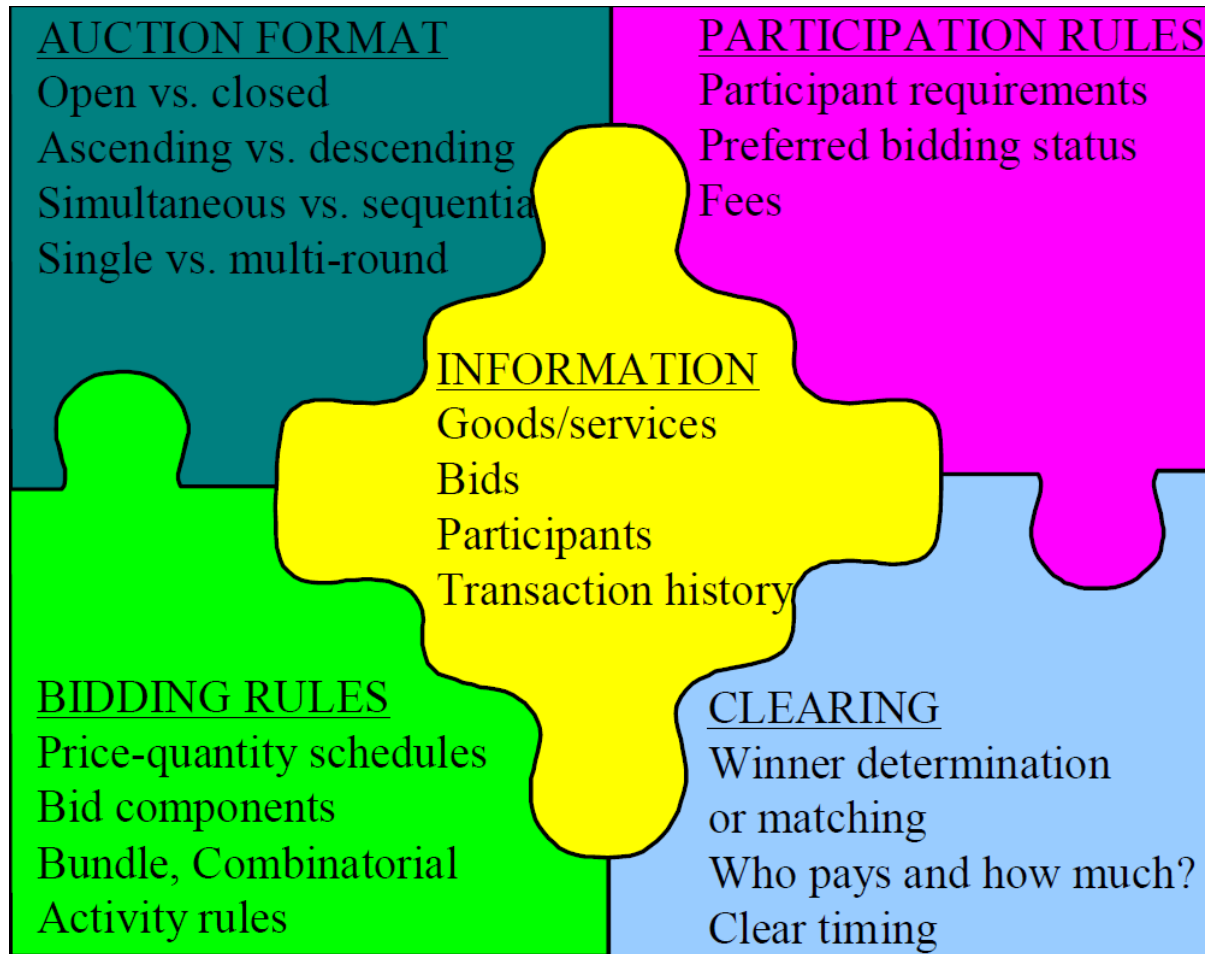
- Open auctions
  - All participants can observe other participants' bids as the bids are made.
    - English auction: Antiques and Art, Livestock, Real estate
    - Dutch auction: Flowers in Netherlands, Fish in Israel
    - Some Internet auctions
- Sealed-bid auctions
  - Participants cannot observe other participants' bids as the bids are made.
    - Bidding for government purchasing contracts
    - Bidding for mineral rights on government-owned land
    - Bidding in FCC spectrum auctions
    - Some internet auctions



# Key Features: Payments in Single Sided Auctions

- First-price auctions
  - Multiple buyers bidding: highest bidder pays the amount bid.
  - Multiple sellers bidding: lowest bidder is paid the amount bid.
- Second-price auctions
  - Multiple buyers bidding: highest bidder pays the amount bid by the second highest bidder (the highest losing bidder).
  - Multiple sellers bidding: lowest bidder is paid the amount bid by the second lowest bidder (the lowest losing bidder).

# Auction Mechanism Design: Major Research Area



# Auctions for Robot Coordination

- Auctioneer is selling a single task
- First-price auction
  - Protocol: Each bidder submits a bid containing a single number representing its cost for the task. The bidder with the lowest bid wins and is awarded the task, agreeing to perform it for the price of its bid.
- Vickrey (second-price) auction
  - Protocol: Same as above, but bidder with the lowest bid agrees to perform task for the price of the second-lowest bidder's bid.
  - Incentive compatible.
- Which mechanism?
  - Doesn't matter if robots bid truthfully. Why (we'll discuss...)

# Multi-Item Auctions

- Protocol: Auctioneer offers a set of  $t$  tasks. Each bidder may submit bids on some/all of the tasks. The auctioneer awards one or more tasks to bidders, with *at most one task awarded to each bidder*.
  - No multiple awards: bids do not consider cost dependencies.
- Protocol may specify a fixed number of awards, *e.g.:*
  1.  $m$  tasks awarded,  $1 \leq m \leq \#bidders$
  2. Every bidder awarded one task ( $m = \#bidders$ )
  3. The one best award ( $m = 1$ )
- For (2) assignment can be done optimally [Gerkey and Mataric 04]
  - Greedy algorithm common: Award the lowest bidder with the associated task, eliminate that bidder and task from contention, and repeat until you run out of tasks or bidders.

# Why/when not Auctions?

- Time complexity (amount of computation)
  - bid valuation in a single auction
  - winner determination in a single auction
  - number of auctions required to sell all tasks
- Communication complexity (message bandwidth)
  - call for bids
  - bid submission
  - awarding tasks to winners
    - may or may not inform losers in addition to winners

# Time Complexity of Auctions

Auction type	Bid valuation	Winner determination	Number of auctions
Single-item	$v$	$O(r)$	$n$
Multi-item (greedy)	$O(n \cdot v)$	$O(n \cdot r \cdot m)$	$\lceil n/m \rceil$
Multi-item (optimal)	$O(n \cdot v)$	$O(r \cdot n^2)$ [6]	$\lceil n/m \rceil$
Combinatorial	$O(2^n \cdot V)$	$O((b + n)^n)$ [5]	1

$n$  = # of items

$r$  = # of bidders

$b$  = # of submitted bid bundles (combinatorial auctions)

$m$  = max # of awards per auction (multi-item auctions),  $1 \leq m \leq r$

$v / V$  = time required for item/bundle valuation (domain dependent)

# Communication Complexity of Auctions [worst case message bandwidth]

Auction type	Auction call	Bid submission	Award	Award (+ losers)
Single-item	$O(r)$	$O(r)$	$O(1)$	$O(r)$
Multi-item	$O(r \cdot n)$	$O(r \cdot n)$	$O(m)$	$O(r)$
Combinatorial	$O(r \cdot n)$	$O(r \cdot 2^n)$	$O(n)$	$O(r + n)$

$n$  = # of items

$r$  = # of bidders

$m$  = max # of awards per auction (multi-item auctions),  $1 \leq m \leq r$

“winners” = auctioneer only informs the winners of auctions

“winners + losers” = auctioneer also informs the losers that they’ve lost

# How exactly does this process work?

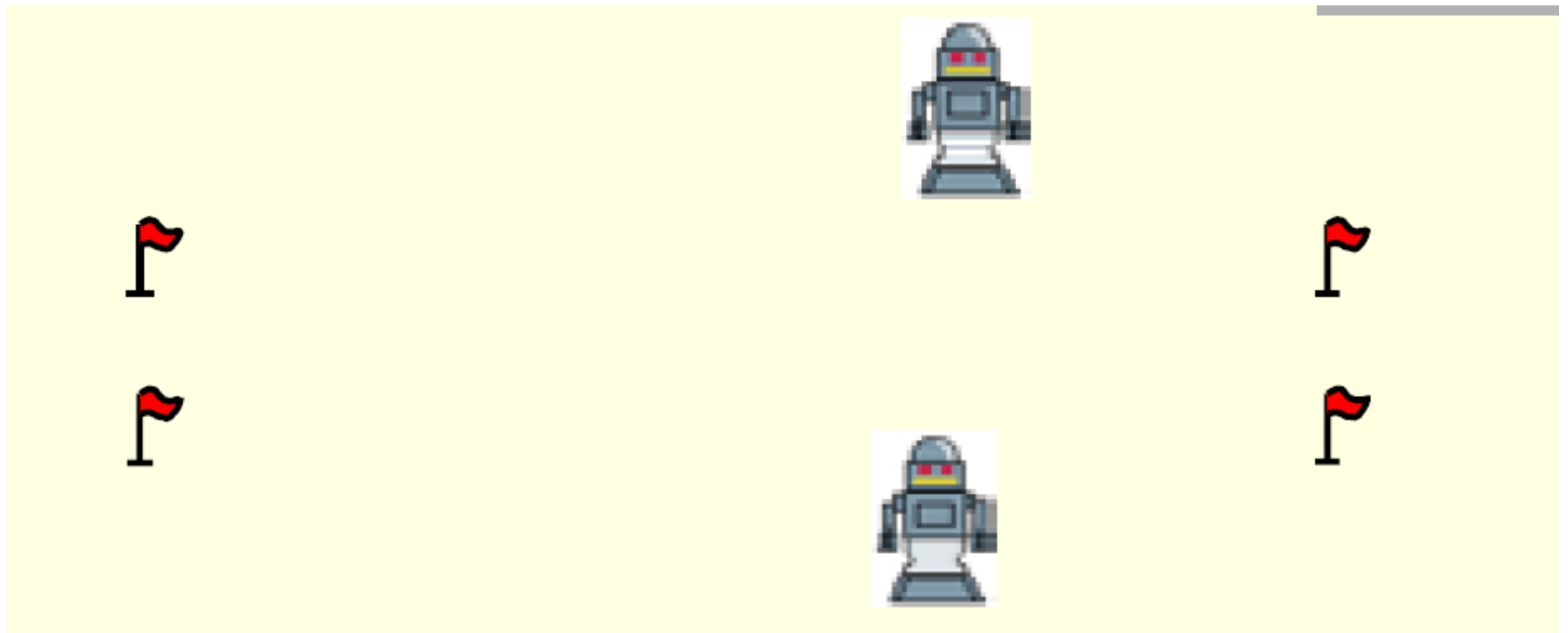
Let us look at some scenarios:

- Parallel auctions
  - Each robot bids on each target in independent and simultaneous auctions.
  - The robot that bids lowest on a target wins it.
  - Each robot determines a cost-minimal path to visit all targets it has won and follows it...
- Sequential auctions
- Combinatorial auctions



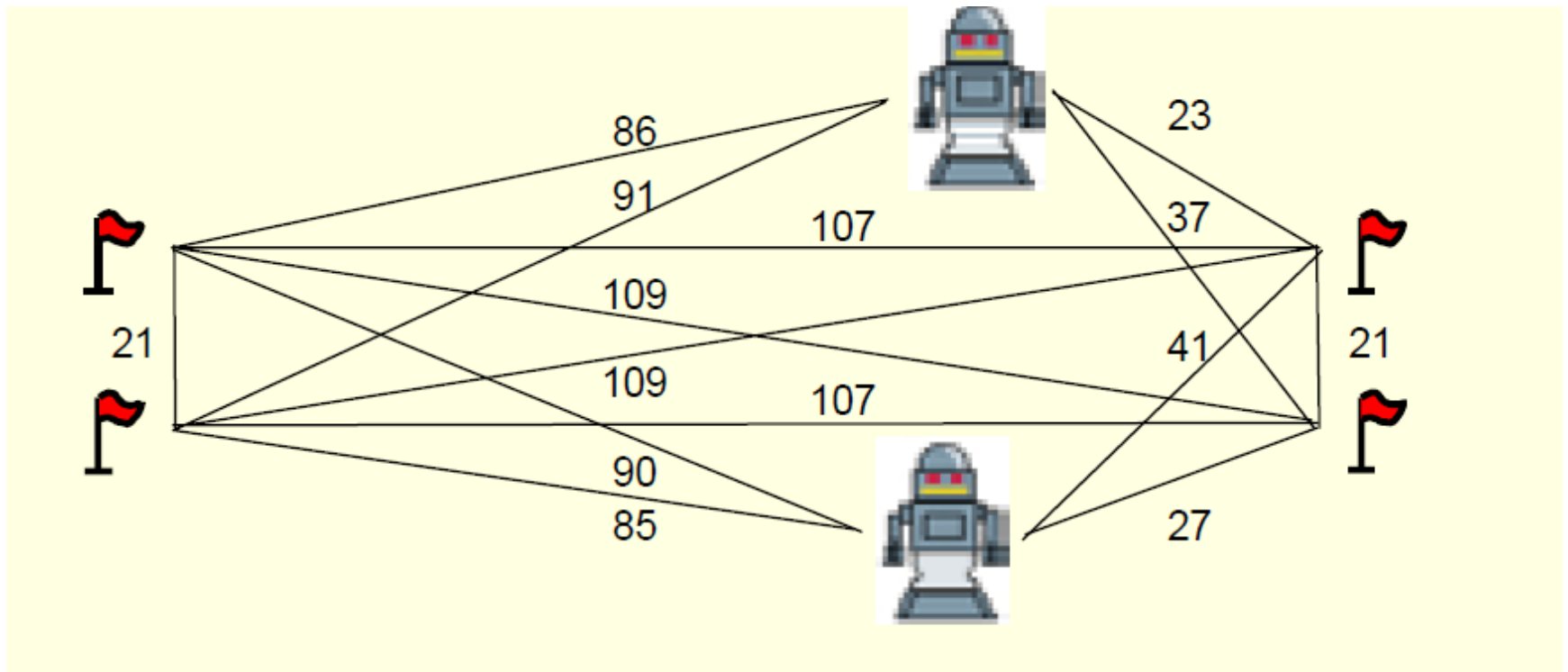
# Parallel Auctions

Each robot bids on a target the minimal path cost it needs from its current location to visit the target.




# Parallel Auctions

Each robot bids on a target the minimal path cost it needs from its current location to visit the target.

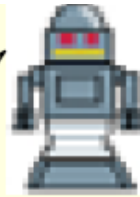


# Parallel Auctions

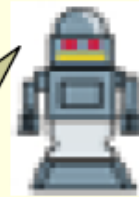
A 


B 


Bid on A: 86  
Bid on B: 91  
Bid on C: 23  
Bid on D: 37



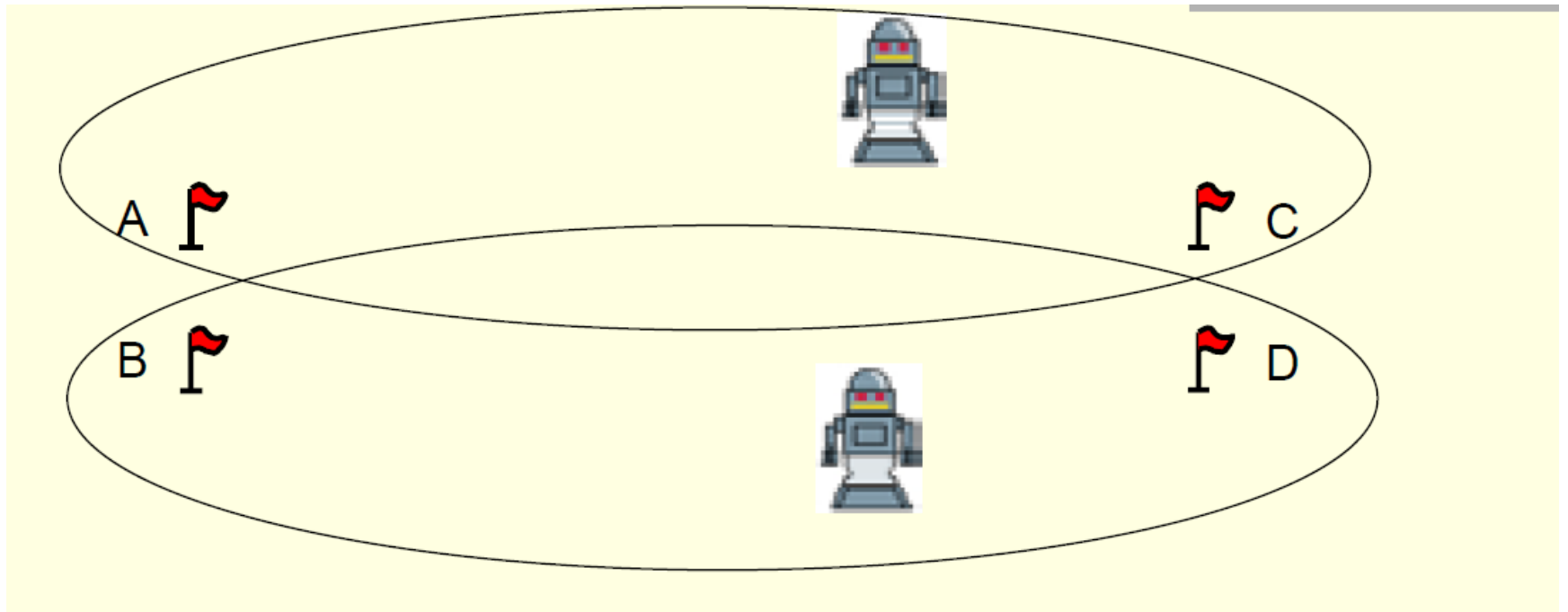
Bid on A: 90  
Bid on B: 85  
Bid on C: 41  
Bid on D: 27



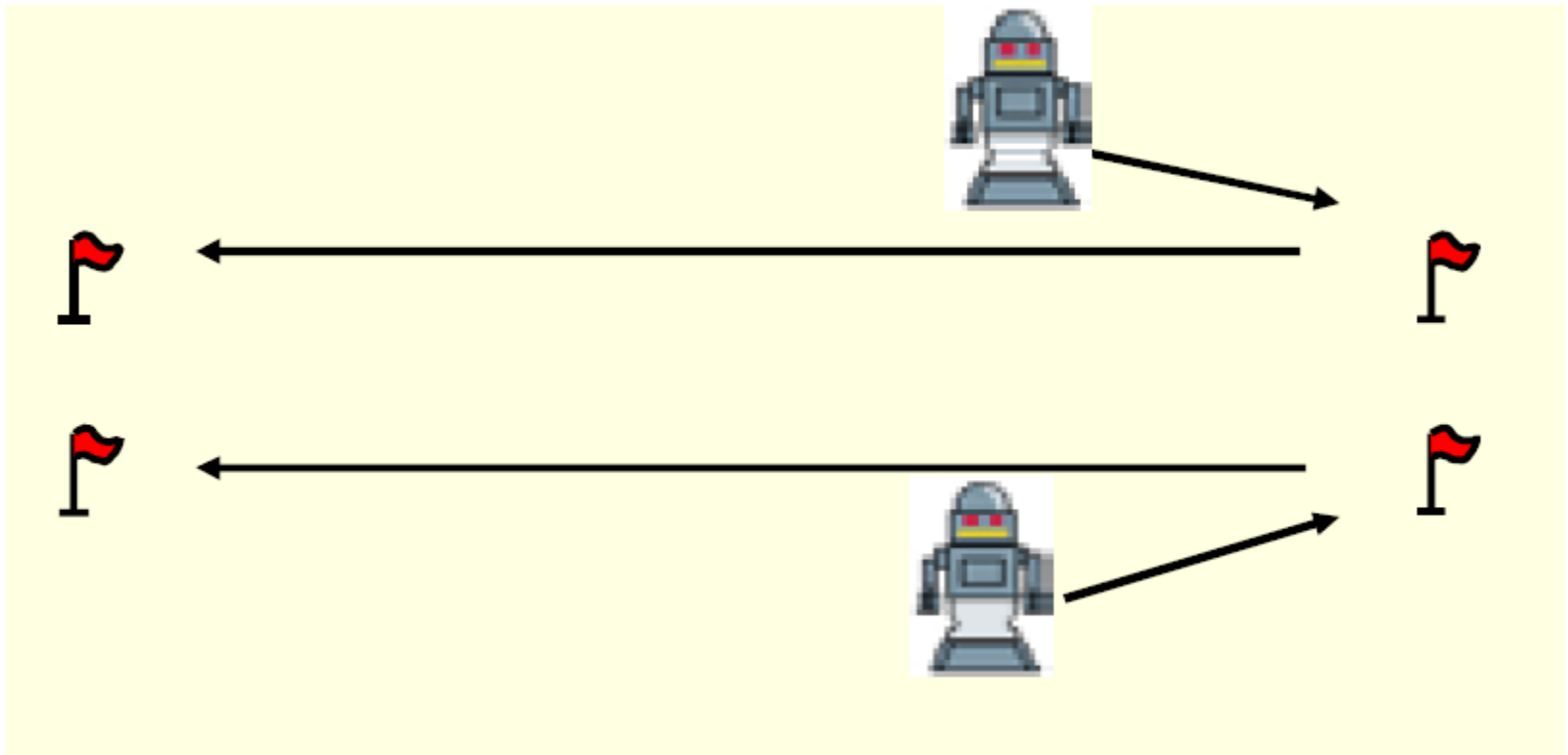
 C

 D

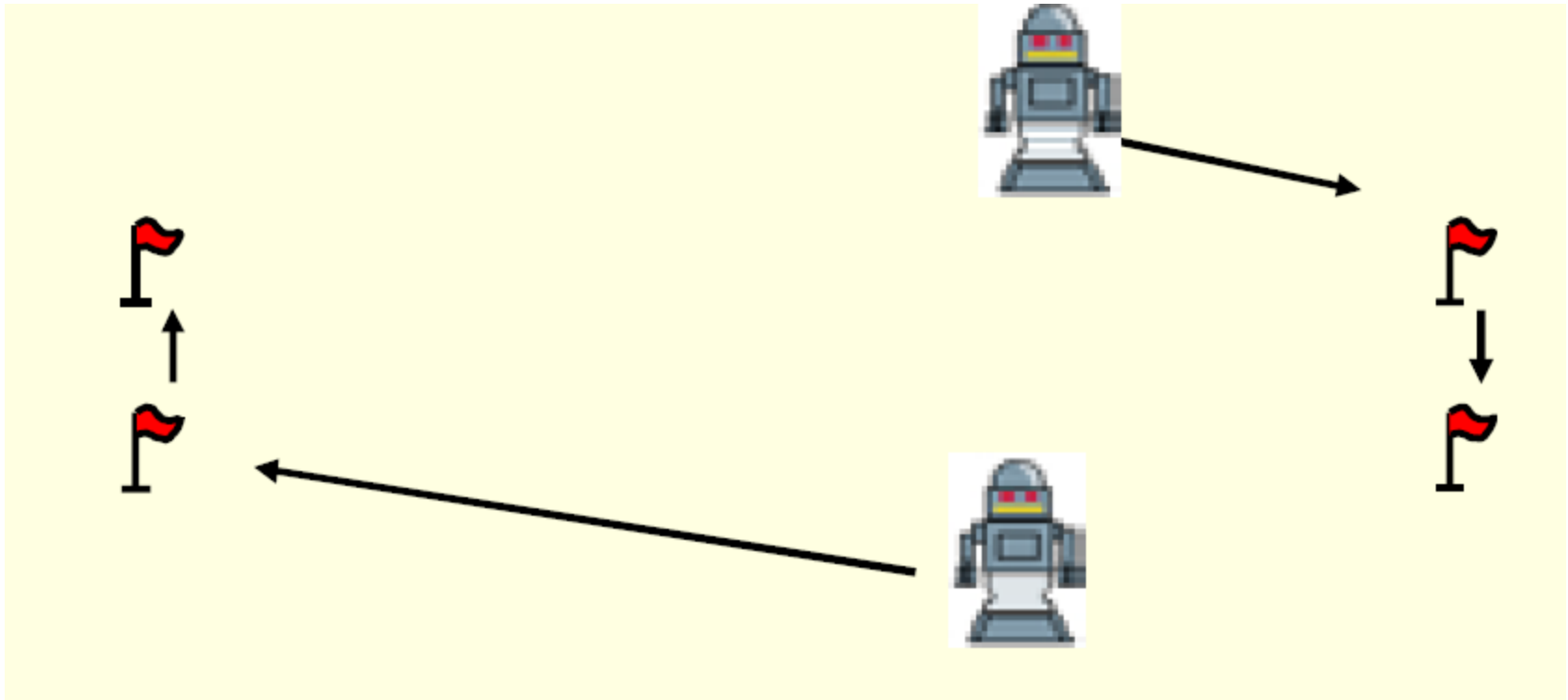
# Parallel Auctions



# Generated Plan

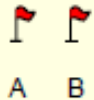


# Limitations of Parallel Auctions



- Minimal team cost (above) is not achieved.
- The team cost resulting from parallel auctions is large because they cannot take synergies between targets into account.

# Parallel Auctions: Good and Bad

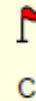
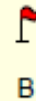


Smallest path cost to visit A: 5  
Smallest path cost to visit B: 4  
Smallest path cost to visit A and B: 5

smallest path cost to visit A and B  
<

smallest path cost to visit A + smallest path cost to visit B

(example: a cake is worth more than the sum of its ingredients)



Smallest path cost to visit B: 4  
Smallest path cost to visit C: 4  
Smallest path cost to visit B and C: 12

smallest path cost to visit B and C  
>

smallest path cost to visit B + smallest path cost to visit C

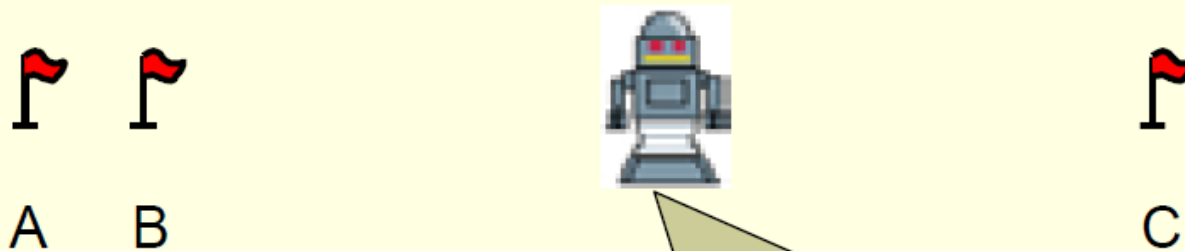
(example: two cars are worth less than the sum of the individual cars)

# Combinatorial Auctions

- Each robot bids on all bundles (= subsets) of targets.
- Each robot wins at most one bundle, so that the number of targets won by all robots is maximal and, with second priority, the sum of the bids of the bundles won by robots is as small as possible.
- Each robot determines a cost-minimal path to visit all targets it has won and follows it.

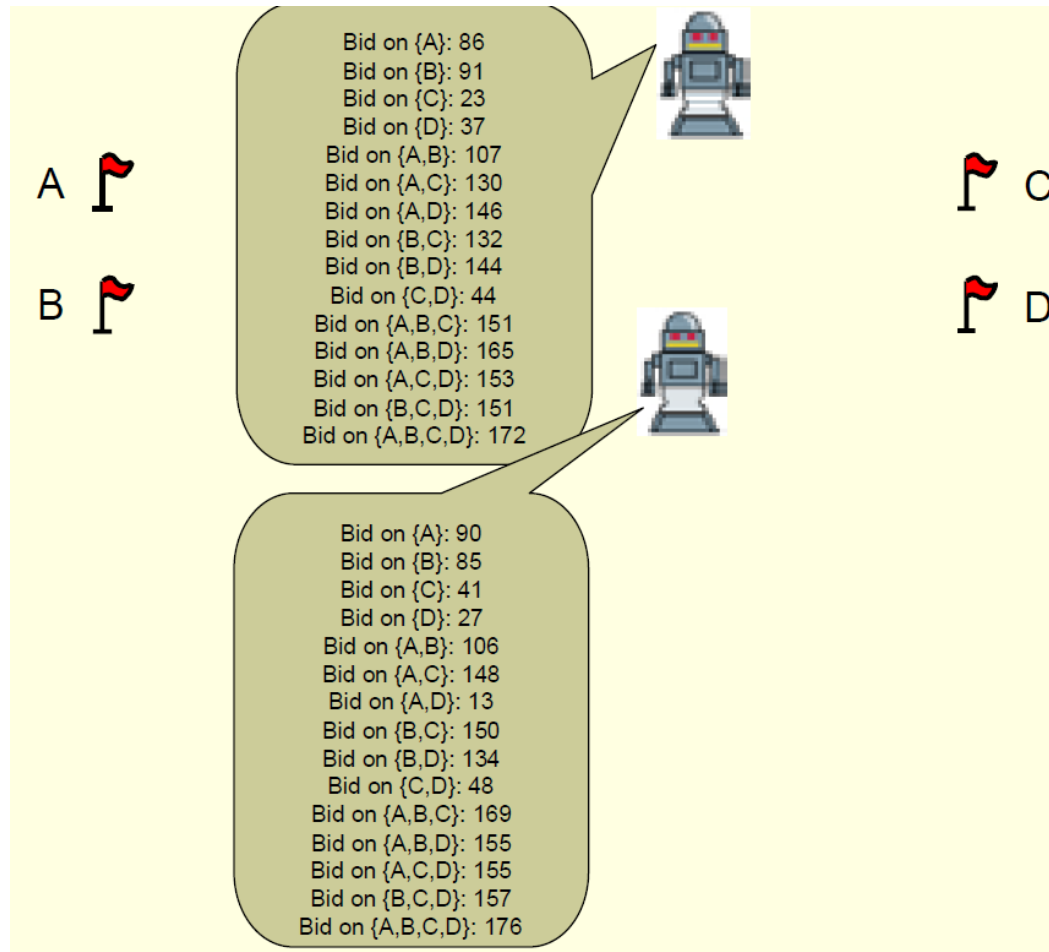


# Exploiting Synergies via Combinatorial Auctions

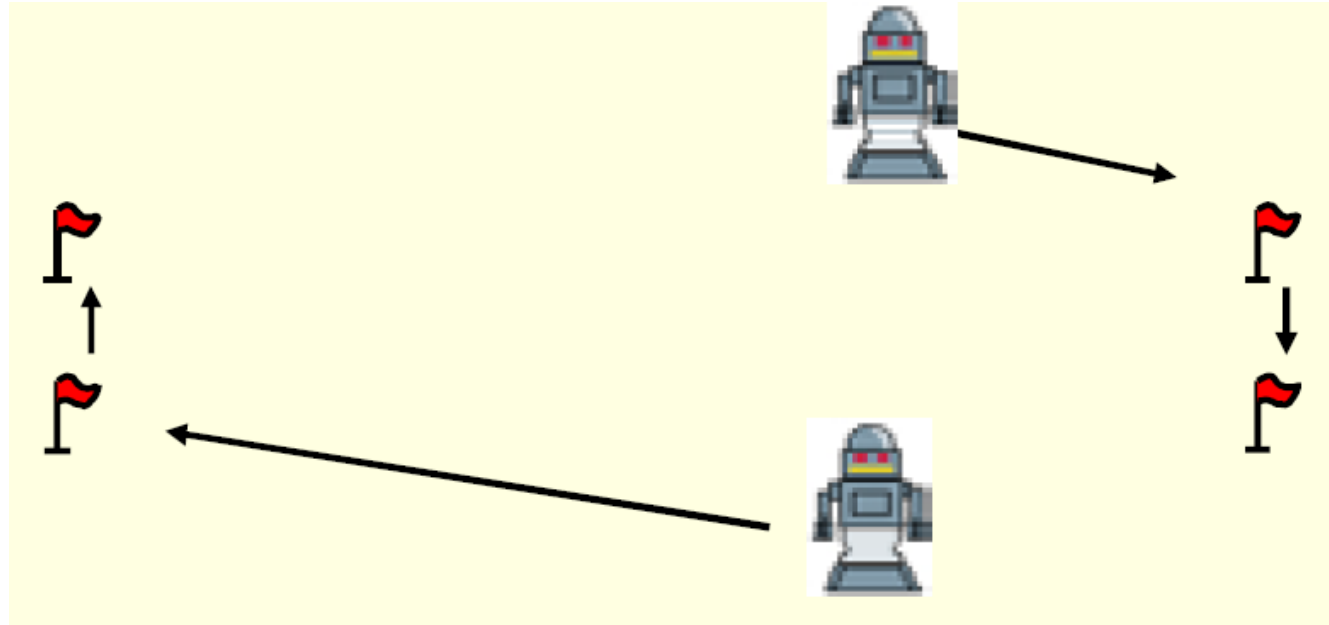


Each robot bids on a bundle the minimal path cost it needs from its current location to visit all targets that the bundle contains.

# Multi-robot Combinatorial Auction



# Combinatorial Auction Result



The team cost resulting from ideal combinatorial auctions is minimal since they take all synergies between targets into account, which solves an NP-hard problem. The number of bids is exponential in the number of targets. Bid generation, bid communication and winner determination are expensive.

# Bidding Strategies in Combinatorial Auctions

- Which bundles to bid on is mostly unexplored in economics because good bundle-generation strategies are domain dependent. For example, one wants to exploit the spatial relationship of targets for multi-robot routing tasks.
- Good bundle-generation strategies
  - generate a small number of bundles
  - generate bundles that cover the solution space
  - generate profitable bundles
  - generate bundles efficiently

# Combinatorial Auctions: Domain-independent Bundle Generation

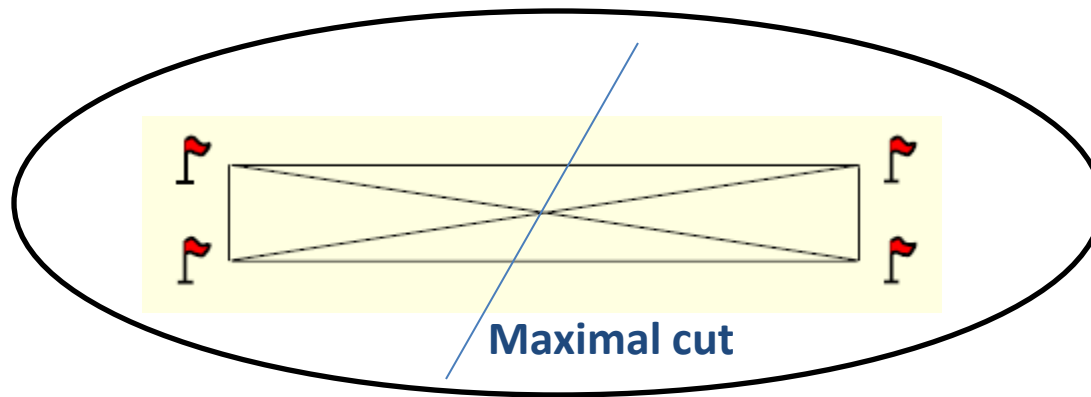
- Dumb bundle generation bids on all bundles (sort-of).
- THREE-COMBINATION
  - Bid on all bundles with 3 targets or less

Note: It might be impossible to allocate all targets.

# Domain Dependent Bundle Generation

- Smart bundle generation bids on clusters of targets.
- GRAPH-CUT
  - Start with a bundle that contains all targets.
  - Bid on the new bundle.
  - Build a complete graph whose vertices are the targets in the bundle and whose edge costs correspond to the path costs between the vertices.
  - Split the graph into two sub graphs along (an approximation of) the maximal cut.
  - Recursively repeat the procedure twice, namely for the targets in each one of the two sub graphs.

# Domain Dependent Bundle Generation

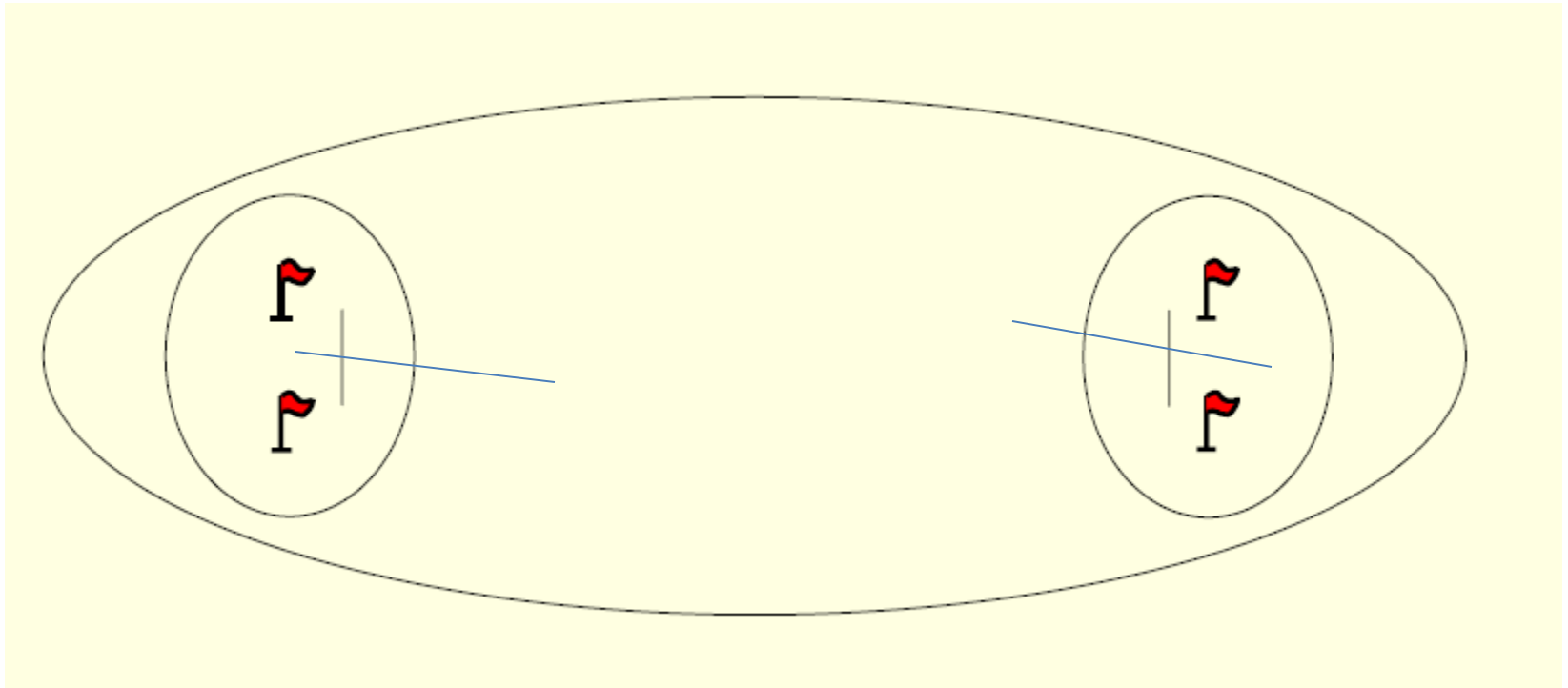


Cut = two sets that partition the vertices of a graph

Maximal cut = maxcut = cut that maximizes the sum of the costs of the edges that connect the two sets of vertices

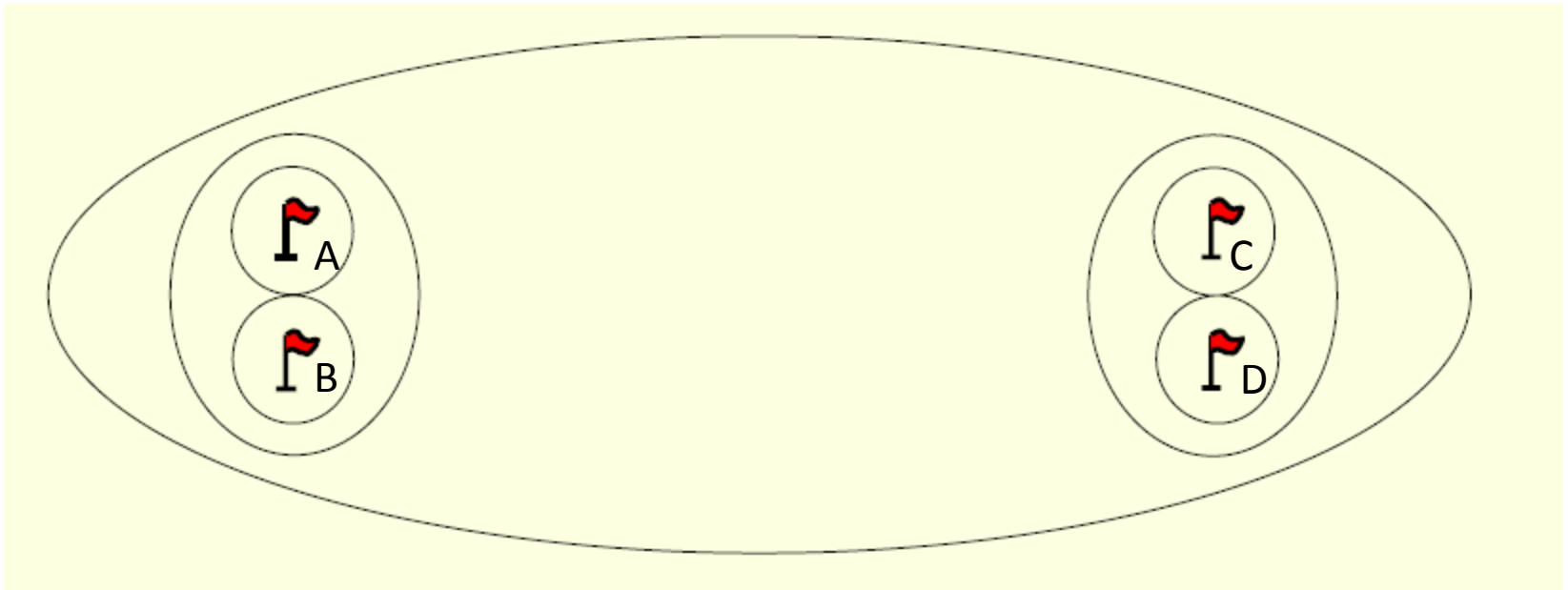
Finding a maximal cut is NP-hard and needs to get approximated.

# Domain Dependent Bundle Generation





# Domain Dependent Bundle Generation



Submit bids for the following bundles

- {A}, {B}, {C}, {D}
- {A,B}, {C,D}
- {A,B,C,D}

# Performance

	number of bids	SUM
parallel single-item auctions	635.1	426.5
combinatorial auctions with THREE-COMBINATION	20506.5	247.9
combinatorial auctions with GRAPH-CUT	1112.1	184.1
optimal (MIP) = ideal combinatorial auctions	N/A	184.4 (due to discretization issues)

**3 robots in known terrain with 5 clusters of 4 targets each (door are closed with 25 percent probability)**

# Combinatorial Auctions Summary

- Ease of implementation: **difficult**
- Ease of decentralization: **unclear** (form robot groups)
- Bid generation: **expensive**
  - Bundle generation: **expensive** (can be NP-hard)
  - Bid generation per bundle: **ok** (NP-hard)
- Bid communication: **expensive**
- Auction clearing: **expensive** (NP-hard)
- Team performance: **very good** (optimal)
  - many (all) synergies taken into account
- Use a smart bundle generation method.
- Approximate the various NP-hard problems.

# Parallel vs. Combinatorial Auctions

## Parallel Auctions

Ease of implementation: **simple**

Ease of decentralization: **simple**

Bid generation: **cheap**

Bid communication: **cheap**

Auction clearing: **cheap**

Team performance: **poor**

## Combinatorial Auctions

Ease of implementation: **difficult**

Ease of decentralization: **unclear**

Bid generation: **expensive**

Bid communication: **expensive**

Auction clearing: **expensive**

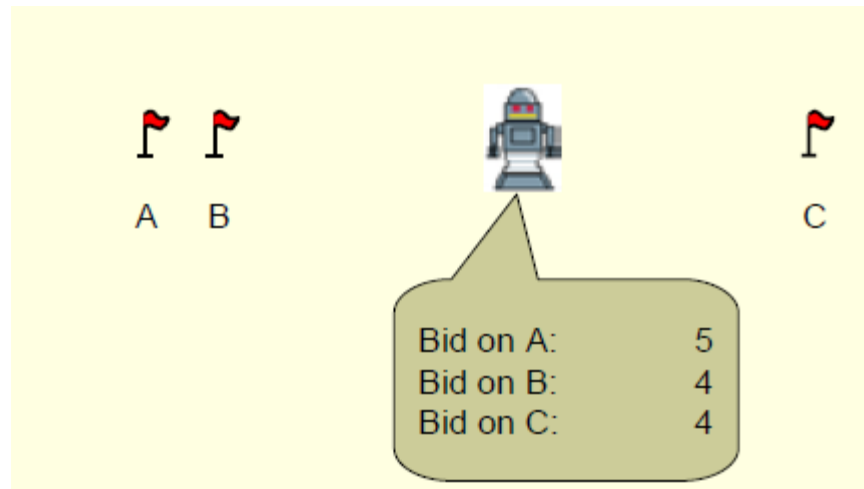
Team performance: **“optimal”**

**Sequential auctions provide a good trade-off between parallel auctions and combinatorial auctions.**

# Sequential Auction Procedure

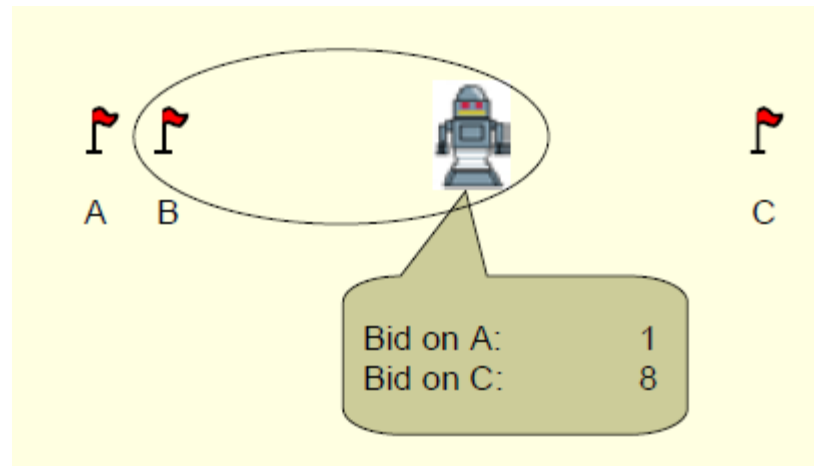
- There are several bidding rounds until all targets have been won by robots. Only one target is won in each round.
- During each round, each robot bids on all targets not yet won by any robot. The minimum bid over all robots and targets wins. (The corresponding robot wins the corresponding target.)
- Each robot determines a cost-minimal path to visit all targets it has won and follows it.

# Sequential Auctions: Synergy



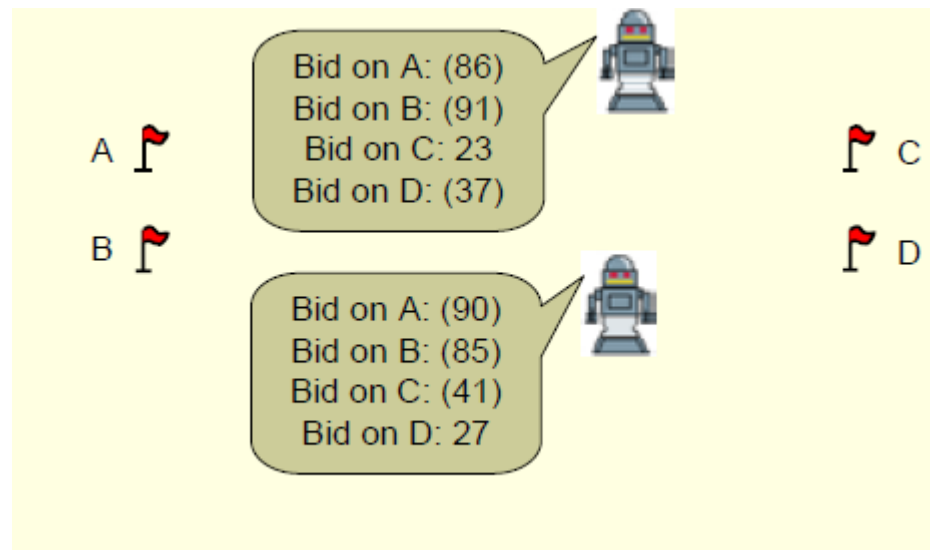
Each robot bids on a target the increase in minimal path cost it needs from its current location to visit all of the targets it has won if it wins the target (BidSumPath).

# Sequential Auctions: Synergy



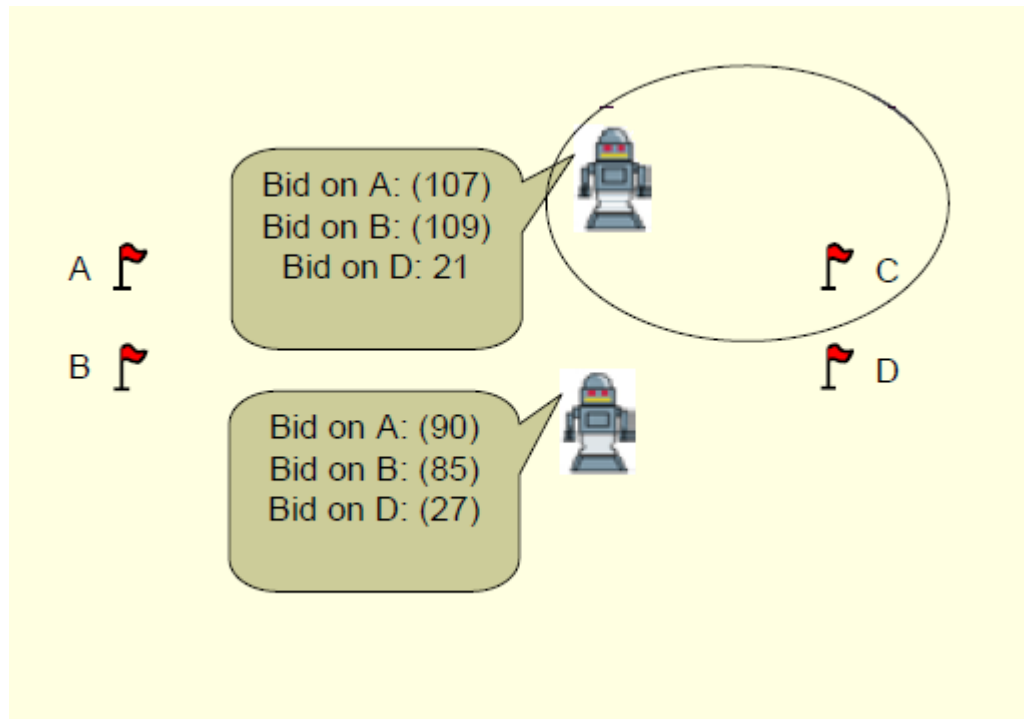
Each robot bids on a target the increase in minimal path cost it needs from its current location to visit all of the targets it has won if it wins the target (BidSumPath).

# Sequential Auctions with multiple robots

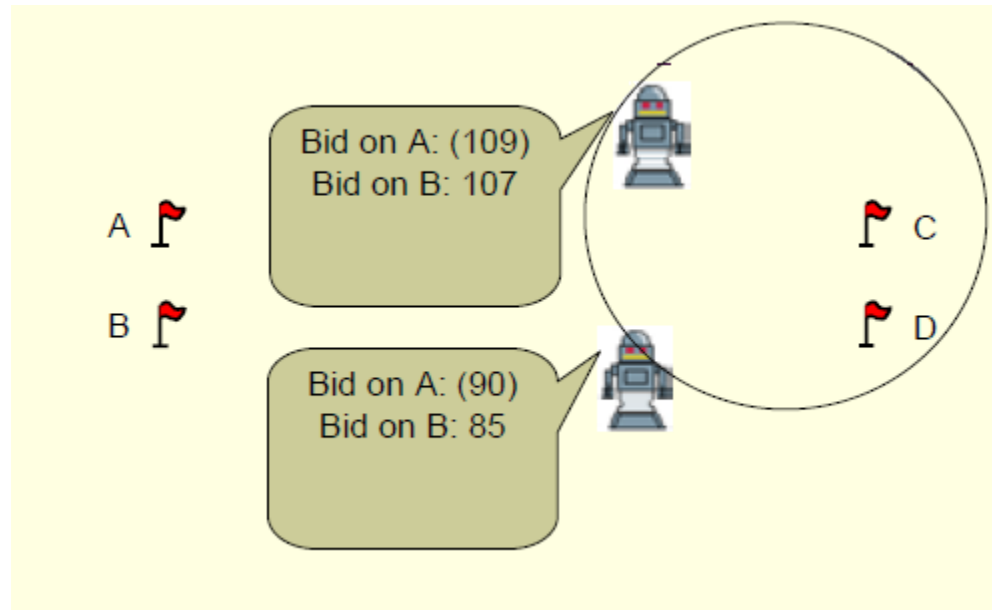




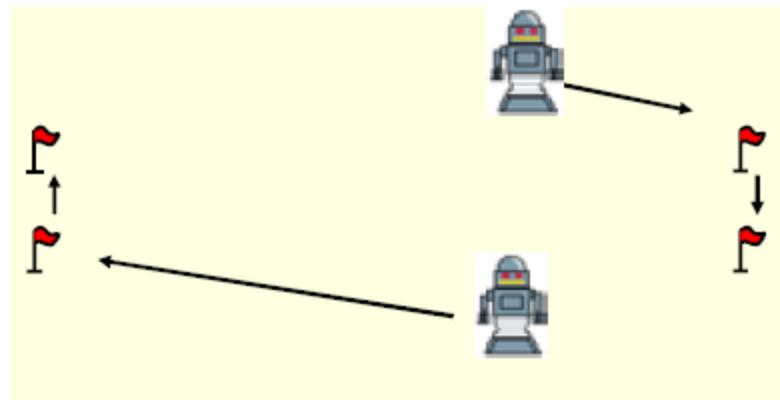
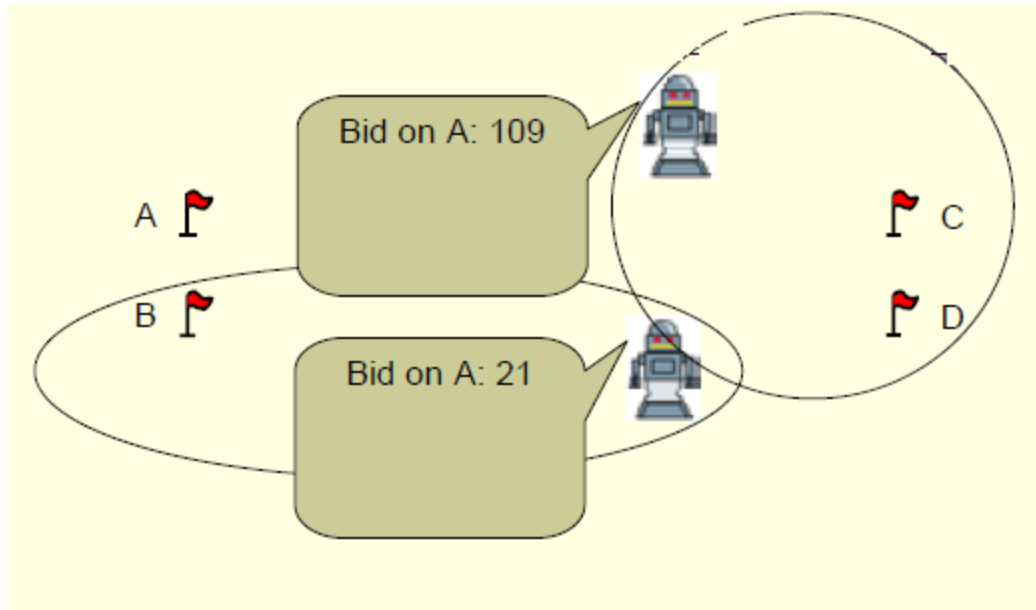
# Sequential Auctions with Multiple Robots



# Sequential Auctions with Multiple Robots



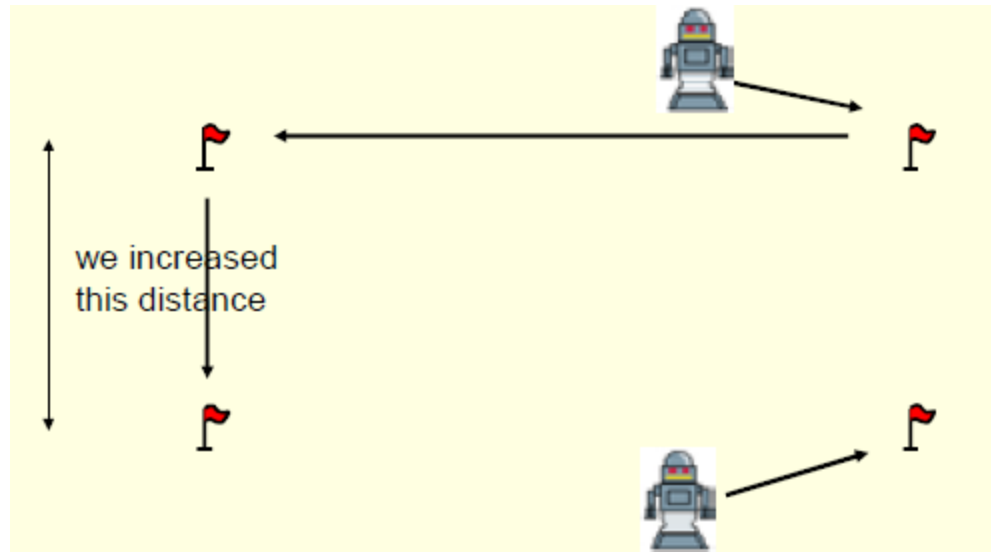
# Sequential Auctions with Multiple Robots



# Sequential Auctions Procedure

- Each robot needs to submit only one of its lowest bid.
- Each robot needs to submit a new bid only directly after the target it bid on was won by some robot (either by itself or some other robot).
- Thus, each robot submits at most one bid per round, and the number of rounds equals the number of targets. Consequently, the total number of bids is no larger than the one of parallel auctions, and bid communication is cheap.
- The bids that do not need to be submitted were shown in parentheses in the example.

# Sequential Auctions Example



The team cost resulting from sequential auctions is not guaranteed to be minimal since they take some but not all synergies between targets into account.

# Sequential Auctions: Summary

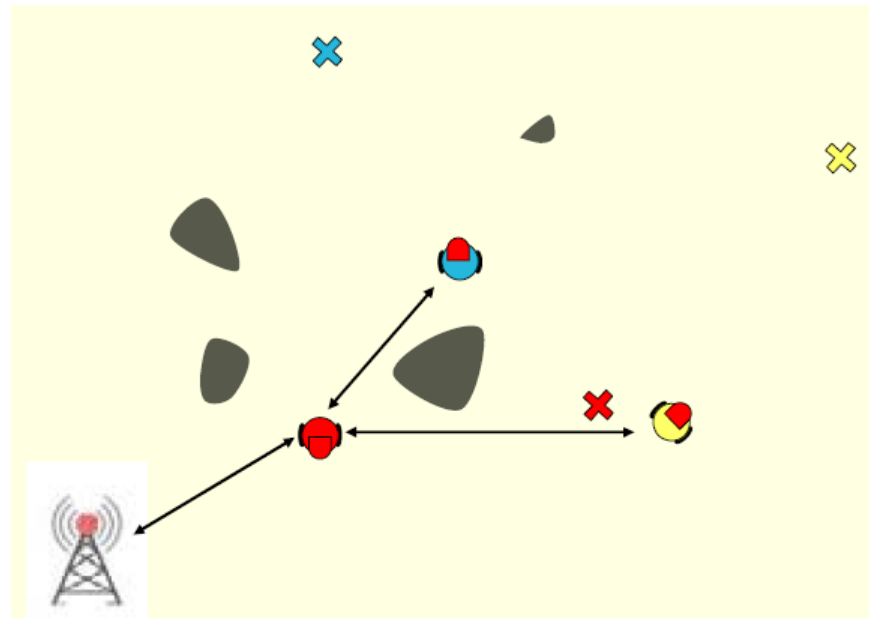
- Ease of implementation: relatively simple
- Ease of decentralization: simple
- Bid generation: cheap
- Bid communication: cheap
- Auction clearing: cheap
- Team performance: very good
  - some synergies taken into account

# Various Kinds of Path Bidding Rules

- **MiniSum**
  - Minimize the sum of the path costs over all robots
  - Minimization of total energy or distance
  - Application: planetary surface exploration
- **MiniMax**
  - Minimize the maximum path cost over all robots
  - Minimization of total completion time (makespan)
  - Application: facility surveillance, mine clearing
- **MiniAve**
  - Minimize the average arrival time over all targets
  - Minimization of average service time (flowtime)
  - Application: search and rescue

# Small Example of Coordinated Motion

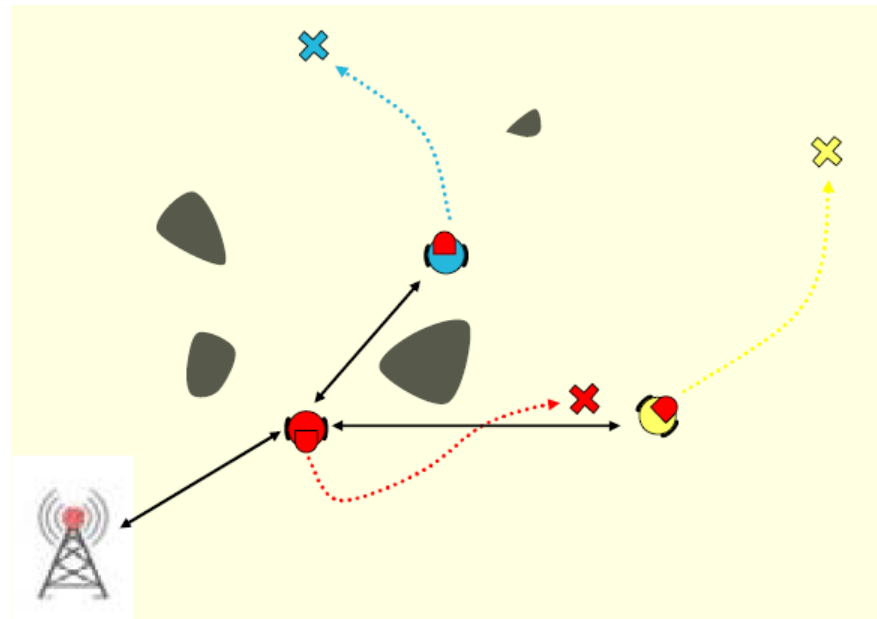
Setup: each robot must go to its goal target without losing contact with the radio tower. The cost of travel is relatively small compared to the high cost of LOS communication.





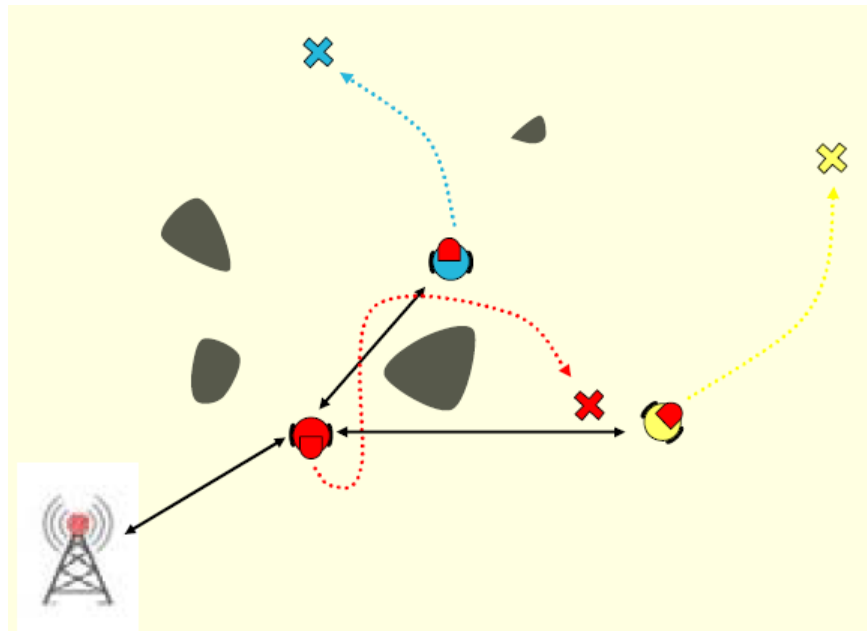
# Small Example of Coordinated Motion

Robots independently generate paths to their goals while considering their teammates' paths. The LOS between red and yellow will not break so they do not need to actively coordinate. But LOS will break between red and blue. Both red and blue will be penalized if they follow their current paths.



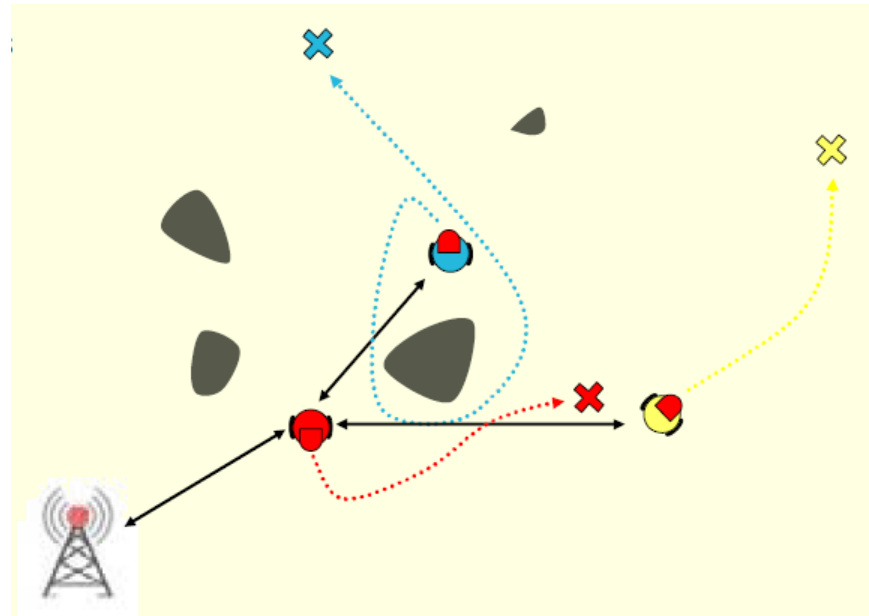
# Small Example of Coordinated Motion

The blue robot proposes this joint plan to the red robot and requests a bid from the red robot for its participation. Red's bid will be too expensive because the proposed plan causes LOS loss between red and yellow.



# Small Example of Coordinated Motion

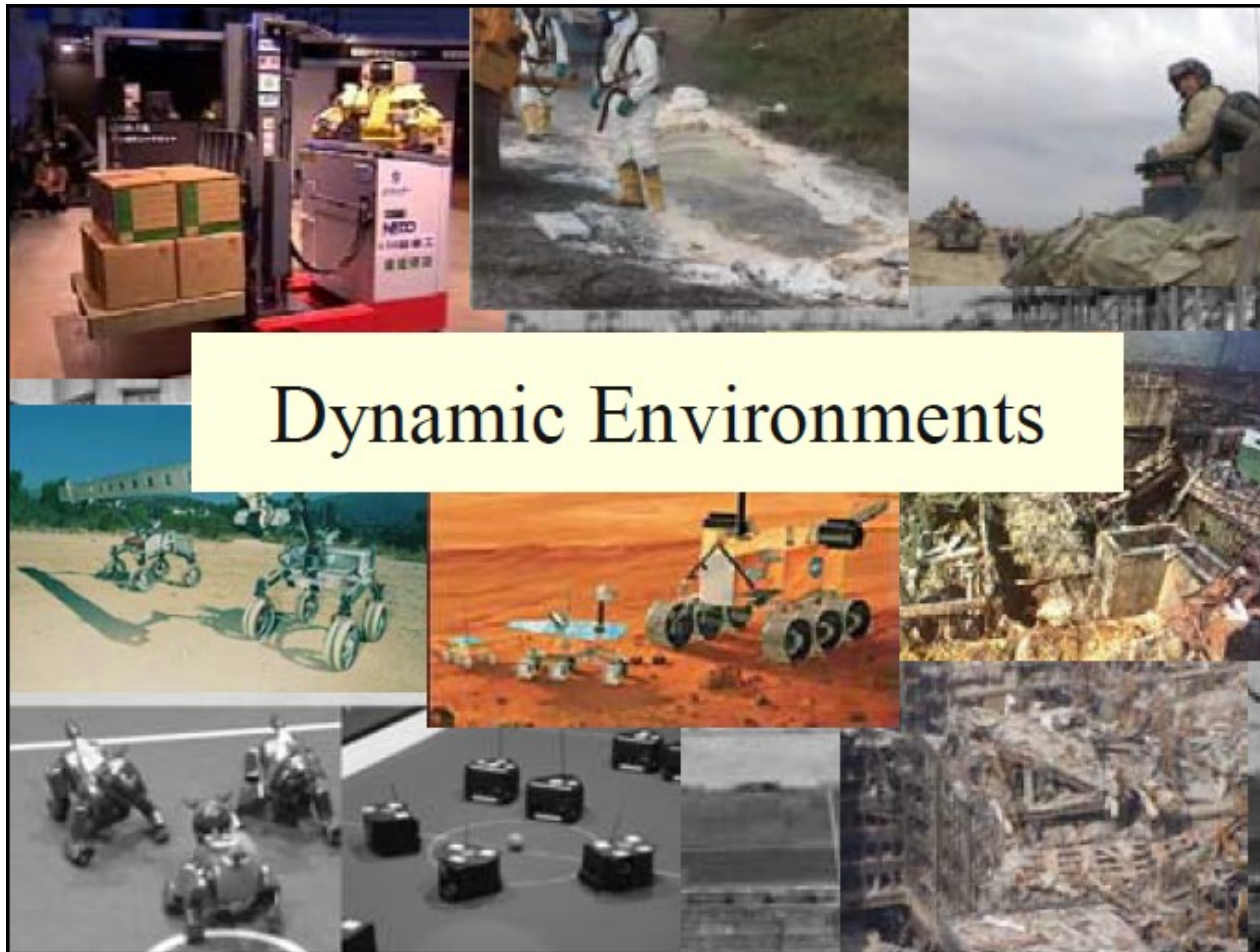
The red robot sends blue a counter offer of this joint plan to the blue robot and requests a bid from the blue robot. Although the path is long, blue's bid will be less costly because it will have communication with the tower. This path will be adopted by the two robots.



# Considerations when designing Coordination Mechanisms

- How dynamic is your environment?
- What are your requirements for robustness?
- How reliable is your information?
- How will you balance scalability vs. solution quality?
- What type of information will you have access to?
- What resources/capabilities does your team possess?
- What do you want to optimize?
- How often will your mission/tasks change?
- What guarantees do you require?

# Why Auctions?



# Characteristics of Dynamic Environments

- Unreliable/incomplete information
- Changing/moving obstacles
- Changing task requirements
- Changing limited resources and capabilities
- Evolving ad-hoc teams

# A Team is Robust if it can...

- Operate in dynamic environments
- Provide a basic level of capability without dependence on communication, but improve performance if communication is possible
- Respond to new tasks, modified tasks, or deleted tasks during execution
- Survive loss (or malfunction) of one or more team members and continue to operate efficiently

# How do things go wrong?

- Communication Failure
  - Acknowledgements can help ensure task completion but delay task allocation
  - Tradeoff between repeated tasks & incomplete tasks
  - Message loss often results in loss in solution quality
- Partial Robot Malfunction
  - Identifying malfunction may be done as an individual or as team
  - Key advantage is that malfunctioning teammate can re-auction tasks it cannot complete
  - Possible new tasks can be generated to enable recovery from malfunction

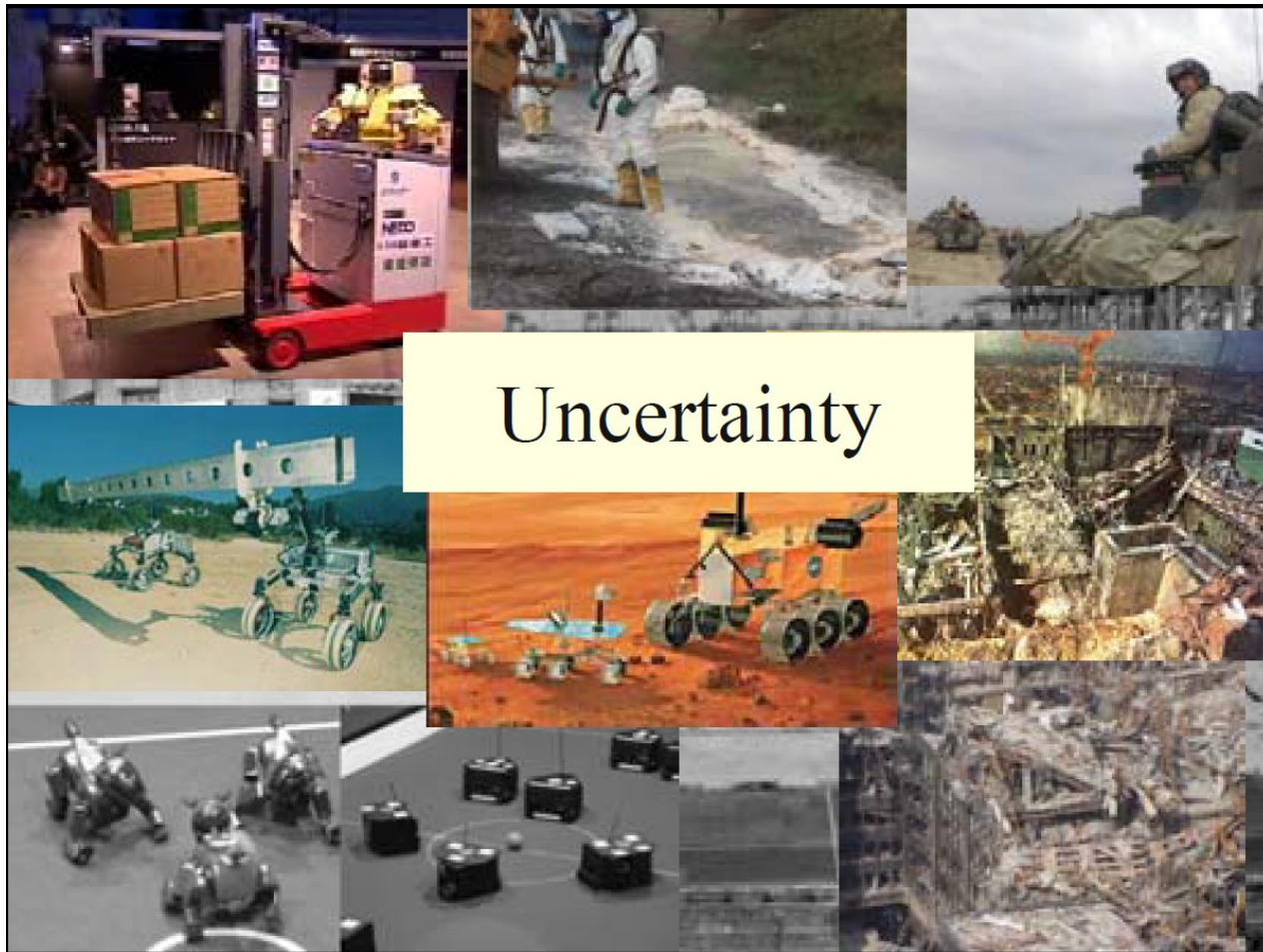


# How do things go wrong?

## Dealing with robot death

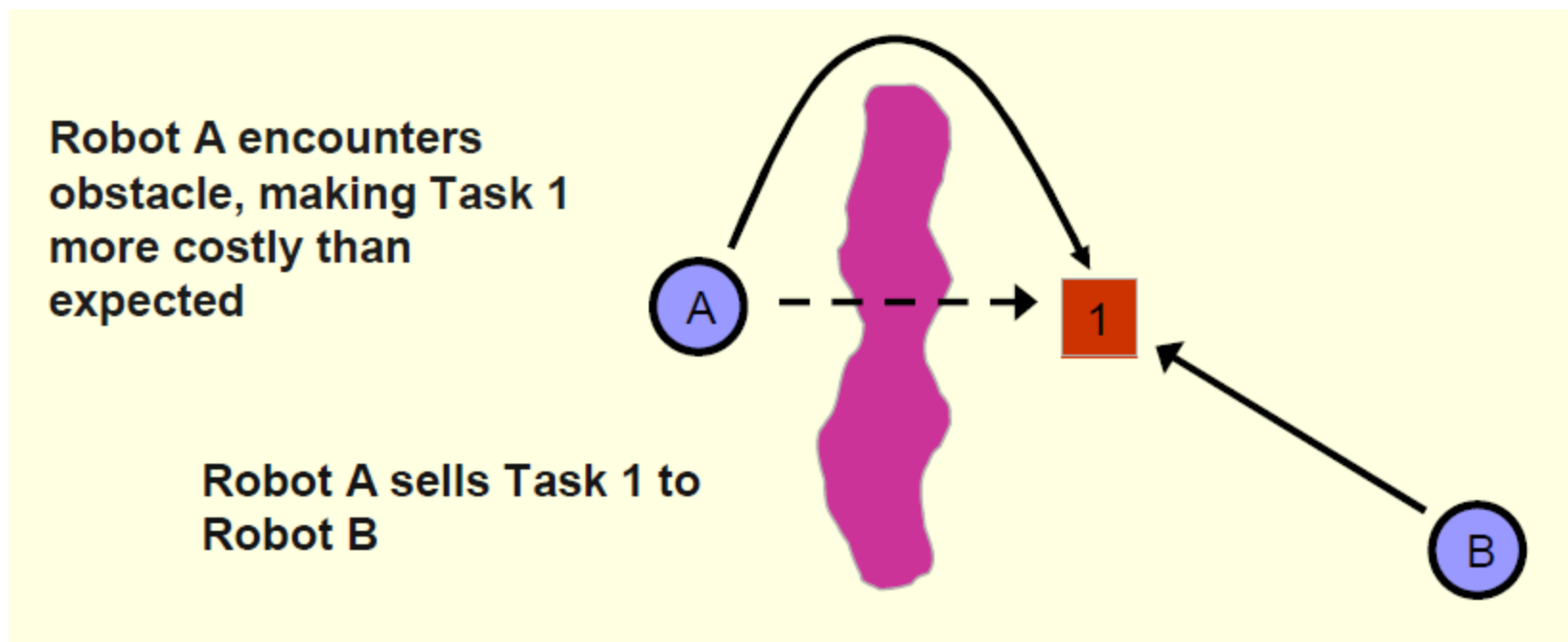
- Detecting the death must be done by the team
- Can detect potential deaths by keeping track of communication links
- Need to seek confirmation of suspected deaths
- Need to query other robots about tasks assigned to dead robot(s) and repair subcontract links
- If no new contract can be made, the owner of the task must complete it

# Why Auctions?



# In Uncertain and Changing Environments

- Robots discover that a task can't be executed for the bid cost
- Robots auction the task to another robot, default, or execute at a loss (learning to estimate better in the future)



# Task Allocation Problem

- **Given**

- a set of tasks,  $T$
- a set of agents,  $A$
- a cost function  $c_i: 2^T \rightarrow \mathbf{RU}\{\infty\}$  (*states the cost agent  $i$  incurs by **handling a** subset of tasks*)
- an initial allocation of tasks among agents  $\langle T_1^{init}, \dots, T_{|A|}^{init} \rangle$ ,  
*where  $\cup T_i^{init} = T$  and  $T_i^{init} \cap T_j^{init} = \emptyset$  for all  $i \neq j$*

- **Find**

the allocation  $\langle T_1, \dots, T_{|A|} \rangle$  that minimizes  $\sum c_i(T_i)$

# Task Allocation Problem: Another Formulation

## Given

- a set of tasks,  $T$
- a set of robots,  $R$
- $\mathcal{R} = 2^R$  is the set of all possible *robot subteams*
- a cost function  $c_r: 2^T \rightarrow \mathbf{R}^+ \cup \{\infty\}$  (**states the cost subteam  $r$  incurs by handling a subset of tasks**)
  - Then an allocation is a function  $A: T \rightarrow \mathcal{R}$  *mapping each task to a subset of robots*
  - Equivalently,  $\mathcal{R}^T$  is the set of all possible allocations

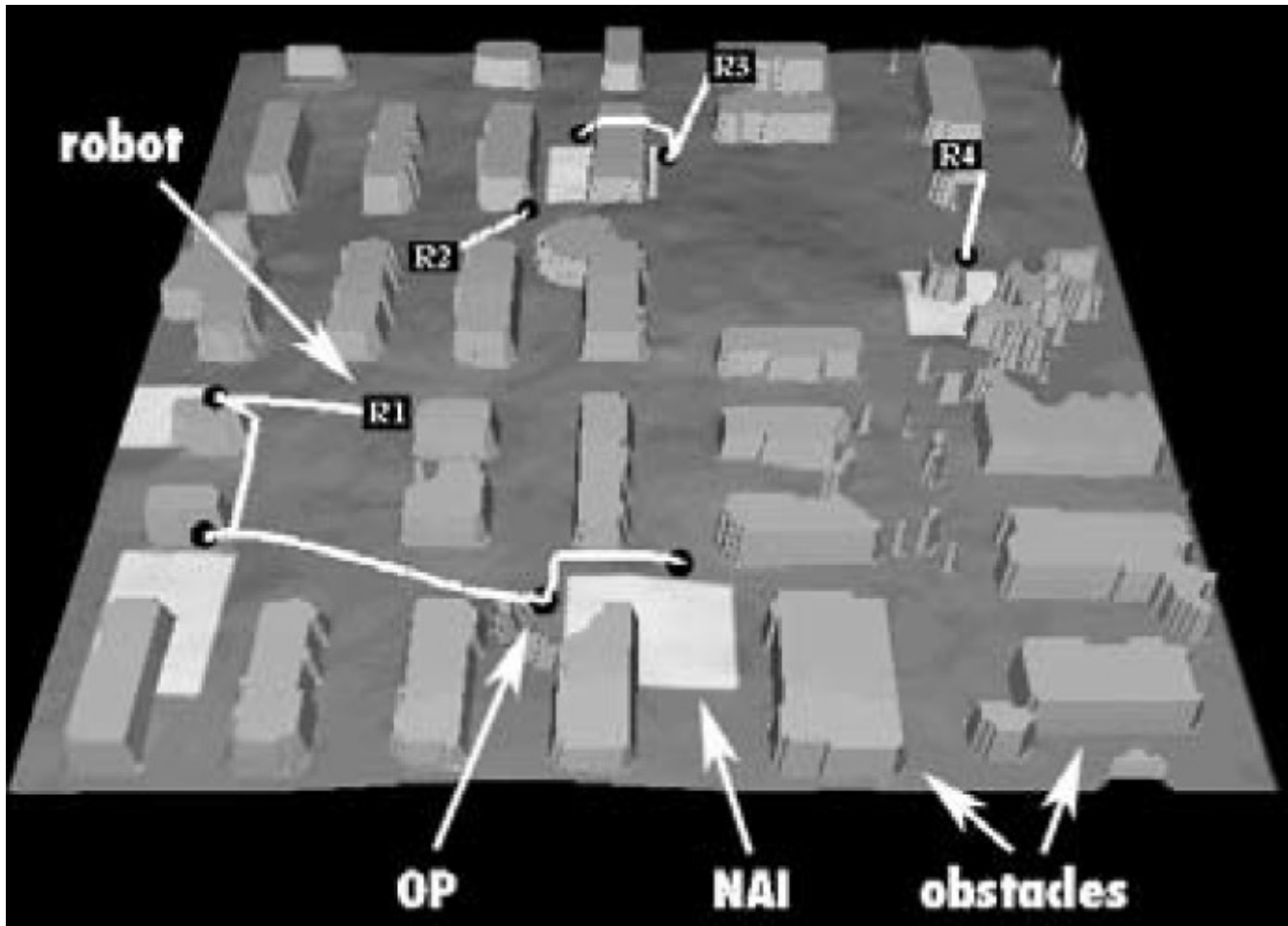
## Find

- the allocation  $A^* \in \mathcal{R}^T$  that minimizes a global objective function  $C: \mathcal{R}^T \rightarrow \mathbf{R}^+ \cup \{\infty\}$

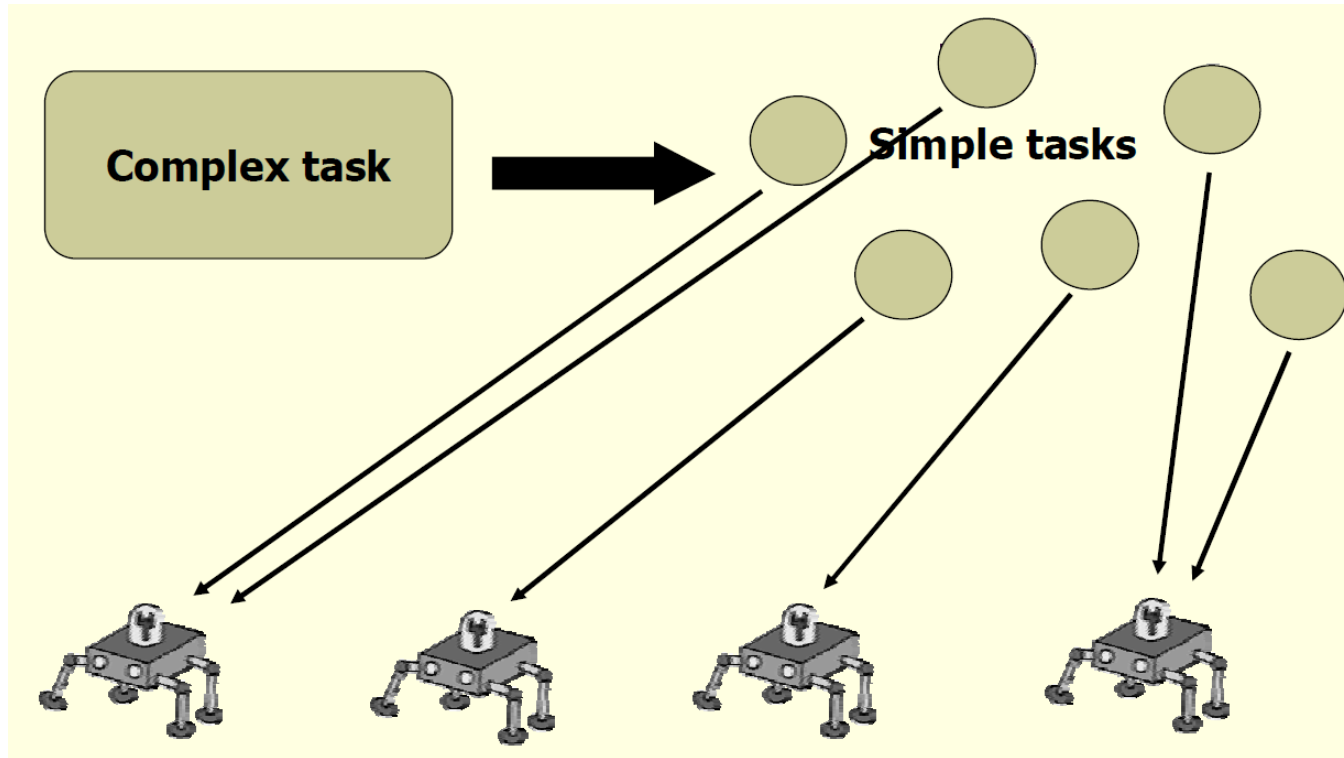
# Remarks about Formulation

- Tasks  $T$  and robots  $R$  may be changing over time
  - Can represent as  $T(t)$  and  $R(t)$
- Robots can only be in one subteam
  - Cost function of a subteam can change if one or more members are performing other tasks individually or as part of other subteams

# More Complex Tasks e.g., Area Reconnaissance



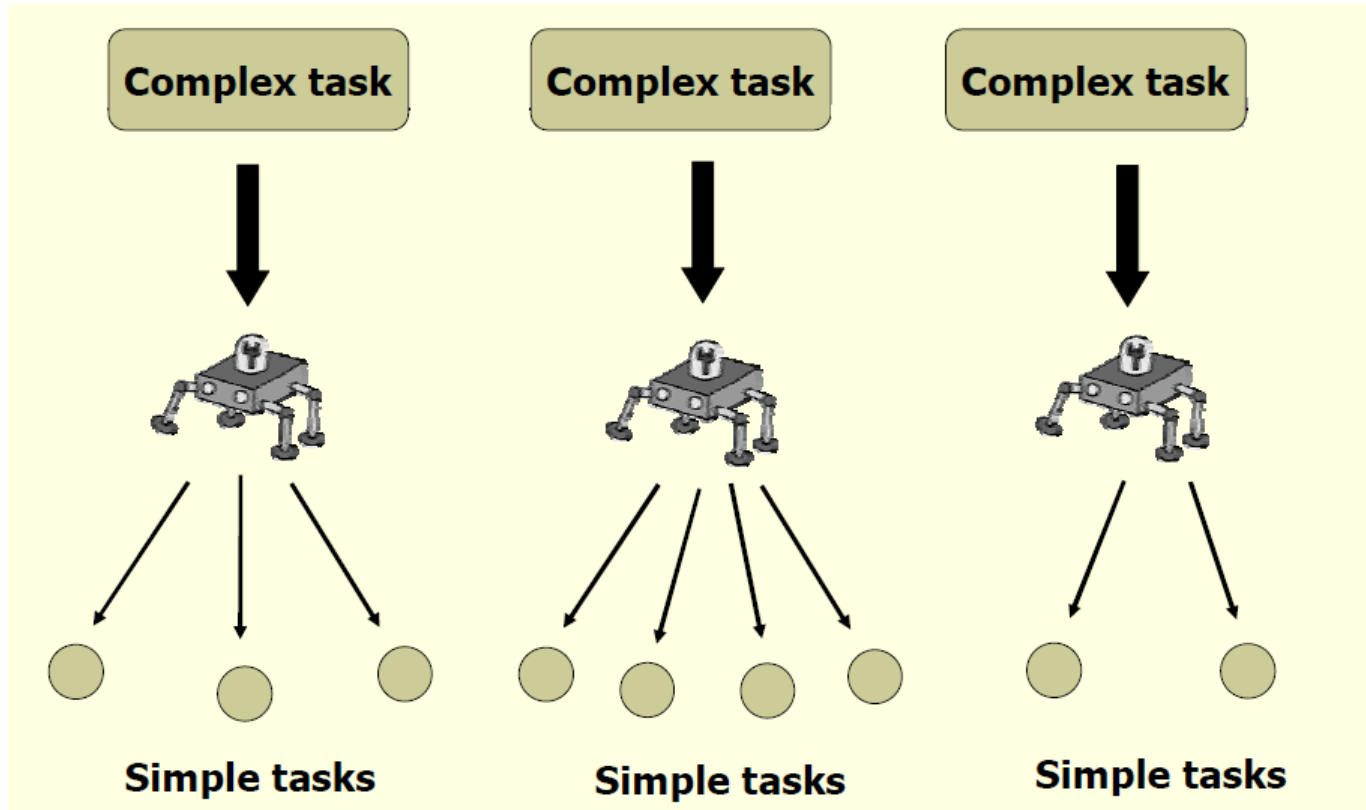
# The Complex Task Allocation Problem



**How can we know how to decompose the complex task(s) efficiently before we know which robots are going to be assigned the resulting simple tasks?**

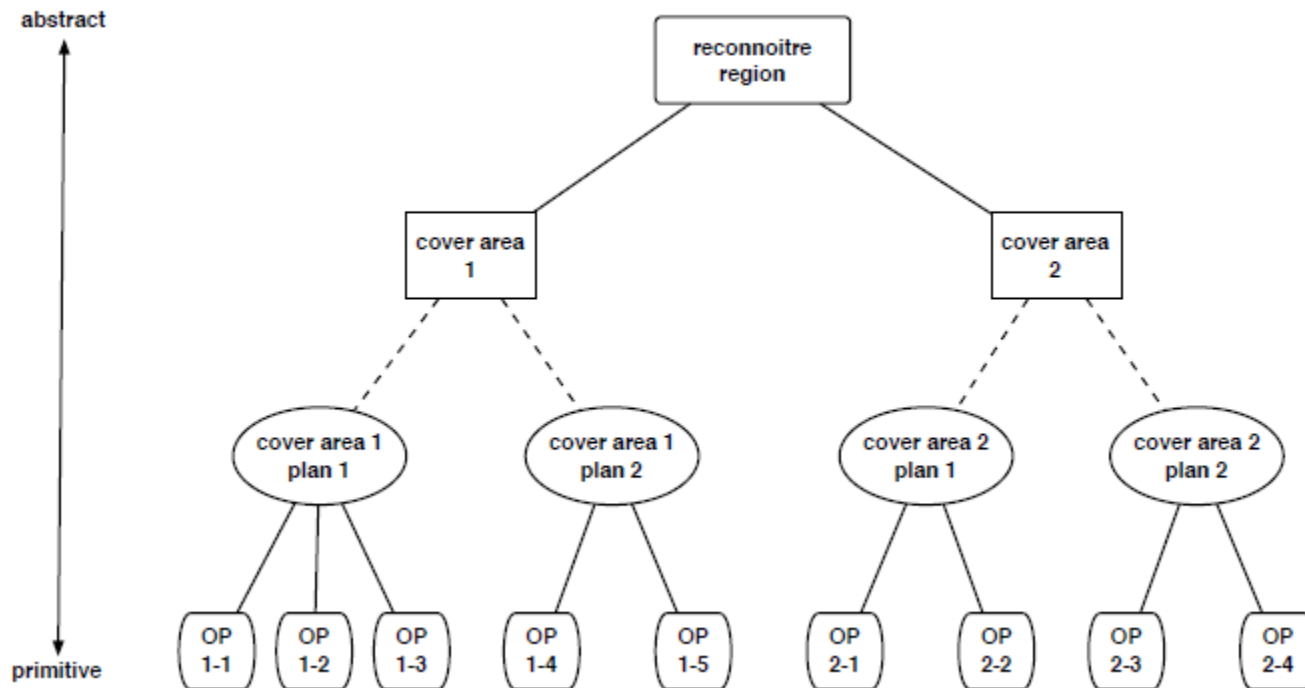


# Complex Task Allocation Problem



**How can we know how to best allocate the complex tasks if we don't yet know how they will be decomposed?**

# An Approach: AND/OR Task Tree



# Task Tree Auctions

- *Task trees are traded on the market*
- Bids are placed for tasks at any level of a task tree
- Bid on a leaf: an agreement to execute a task for a given price
- Bid on an interior node: agreement to complete a complex task
  - original tree decomposition
  - replanning
- Avoids premature commitment on allocation/decomposition decisions
- Mechanism enables:
  - Tasks can be reallocated or redecomposed
  - Robots can develop their own plans for complex tasks
  - Subtasks of a complex task can be shared by multiple robots

# Heterogeneous Team Scenarios

- Members of team are equipped differently, have different skills, or play different roles.
- Why heterogeneous teams?
  - For complex missions, many specialists better than a few generalists
    - For USAR (rescue), robots need different form factors and sensing modalities
  - Specialists often easier to design than generalists.
  - Enabling coordinated heterogeneous teams means easier reuse across applications

# Allocation for Heterogeneous Teams

- Allocation requires reasoning about different robots' capabilities.
- Markets well suited for allocation in these domains
  - Each bid can encapsulate a robot's ability to complete the task.
    - Robots need not bid if they can't do the task.
    - Individual robot needs only to be able to assess its own abilities and resources.
  - Auctioneer can award task only based on bids, not individual knowledge of individual capabilities.
- Valuation of different allocations difficult
  - For a visual inspection task should a very busy Binocular Roving-Eye bid lower or higher than an idle Pioneer with a web cam?

# On Valuation of Online Tasks

- Naïve valuation (current increased reward for each agent) produces reasonable results but can be highly non-optimal
  - Various kinds of market inefficiencies occur
    - Re-auctions can help
  - Does not take into account that task issue is online
- What might improve the valuation given that new tasks will be arriving?
  - Learning opportunity cost for Heterogeneous agents.

# Example: Learning for Heterogeneous Agents

Domain: Mars rovers investigating rocks

- Two types of rocks: A and B rocks
- Two types of rovers: AB rovers and A rovers
- Variable rewards offered for examining rocks; model reward decay as  $\gamma^t R$ , where  $\gamma$  is a discount rate,  $t$  is time since issue, and  $R$  is the maximum possible reward
- Tasks issued at fixed rate, and system oversubscribed



# Learning Opportunity Cost

- Opportunity cost per time unit (*OpCost*) *initialized to some value*
- Bidding process
  - When agents get a new task they compute additional reward *A as well as the difference in schedule length S*
    - *S represents additional time requirement*
  - Actual bid is  $A - (OpCost * S)$
- Learning opportunity cost
  - At set interval, all rovers of the same type set their *OpCost* to total received reward over total experiment time (average reward per unit time)



# Some Open Questions:

## Can we Incorporate Human Preferences?

- Instantiating human preference in an objective function can be difficult
- Many interactions between objective function and solution quality
  - Success of allocation strategy contingent on many factors
    - System load
    - Types of tasks (values and rates of decay)
    - Learning capabilities of agents
- (How) can we somehow incorporate user feedback?

# Challenge Problem: Ad Hoc Teams

- Imagine that you are in a foreign country where you do not speak the language, walking alone through a park. You see somebody fall off his bicycle and injure himself badly;
- There are a few other people in the area, and all of you rush to help the victim. There are several things that need to be done. Somebody should call an ambulance, someone should check that the victim is still breathing, and someone should try to find a nearby doctor or policeman.
- However, none of you know one another, and thus you do not know who has a mobile phone, who is trained in first aid, who can run fast, and so forth. Furthermore, not all of you speak the same language.
- Nonetheless, it is essential that you quickly coordinate towards your common goal of maximizing the victim's chances of timely treatment.

# Ad Hoc Autonomous Agent Teams

## The Challenge

---

To create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members.

P. Stone, G.A. Kaminka, S. Kraus, J.S. Rosenschein, Ad hoc autonomous agent teams: collaboration without precoordination, *AAAI* 2010.

# What is the Objective?

How does one evaluate performance in an open ended way, i.e., what does it mean to be functional in this setting?

---

Evaluate( $a_0, a_1, A, D$ )

---

- Initialize performance (reward) counters  $r_0$  and  $r_1$  for agents  $a_0$  and  $a_1$  respectively to  $r_0 = r_1 = 0$ .
  - Repeat:
    - Sample a task  $d$  from  $D$ .
    - Randomly draw a subset of agents  $B$ ,  $|B| \geq 2$ , from  $A$  such that  $E[s(B, d)] \geq s_{min}$ . ————— Performance level,  $s$
    - Randomly select one agent  $b \in B$  to remove from the team to create the team  $B^-$ .
    - increment  $r_0$  by  $s(\{a_0\} \cup B^-, d)$
    - increment  $r_1$  by  $s(\{a_1\} \cup B^-, d)$
  - If  $r_0 > r_1$  then we conclude that  $a_0$  is a better ad-hoc team player than  $a_1$  in domain  $D$  over the set of possible teammates  $A$ .
-

# A Simple Formulation

- Can agent A **lead** agent B?
- Robots are not all programmed by the same people, and may not all have the same communication protocols or world models
- They are likely to have heterogeneous sensing and acting capabilities that may not be fully known to each other
  - So, team strategies cannot be developed *a priori*
- A good agent must be:
  - prepared to cooperate with many types of teammates
  - able to generate actions that differ significantly depending on the characteristics of its teammates

# Simple Problem Formulation

Fully cooperative iterative normal-form game between two agents, *Agent A and Agent B*

<i>M1</i>	<i>b</i> <sub>0</sub>	<i>b</i> <sub>1</sub>	<i>b</i> <sub>2</sub>
<i>a</i> <sub>0</sub>	25	1	0
<i>a</i> <sub>1</sub>	10	30	10
<i>a</i> <sub>2</sub>	0	33	40

Agent B's actions,  $y$

Agent A's actions,  $x$

Immediate payoffs

Highest payoff,  $m^*$

What sequence of actions should *Agent A take so as* to maximize the team's undiscounted long-term payoff over iterative interactions using the identical payoff matrix?

- A solves this problem, but solution depends on B's strategy

# One Solution Method

- Assume B is a bounded memory best response agent
- A's problem is that of identifying an optimal action sequence  $S^*(a_i, b_j)$  given the matrix payoffs M
- This can be achieved using a variety of recursive search procedures
  - Here is one example:

---

## Algorithm 1 Find $S^*$ 's ( $M$ )

---

```

1: for  $i = 0$  to  $x - 1$  do
2:   for  $j = 0$  to  $y - 1$  do
3:      $S_0^*(a_i, b_j) = \begin{cases} [(a_i, b_i), (a_i, b_i), \dots] & \text{if } m_{i,j} = m^* \\ \omega & \text{if } m_{i,j} < m^* \end{cases}$ 
4:   end for
5: end for
6:  $len = 0$ 
7: repeat
8:    $len = len + 1$ 
9:   for  $i = 0$  to  $x - 1$  do
10:     $S_{len}^*(a_i, b_0) = S_{len-1}^*(a_i, b_0)$ 
11:    for  $act = 0$  to  $x - 1$  do
12:      $S' = S_{len-1}^*(a_{act}, b_{BR(a_i)})$ 
13:     if  $m^* - m_{i,0} + C(S') < C(S_{len}^*(a_i, b_0))$  then
14:       $S_{len}^*(a_i, b_0) = \text{PREPEND}((a_i, b_0), S')$ 
15:     end if
16:    end for
17:    for  $j = 1$  to  $y - 1$  do
18:      $S_{len}^*(a_i, b_j) = \text{REPLACEHEAD}(S_{len}^*(a_i, b_0), (a_i, b_j))$ 
19:    end for
20:   end for
21: until  $len = \text{UPPERBOUND}(\bar{L}(M))$ 

```

---

# Summary

- Auction mechanism as a way to take local information and – in combination with protocols – achieve global decisions
- Many kinds of problems:
  - Coordination
  - Allocation
- Auctions and mechanism design is a vibrant research area – trick is to pose robotics problems in a way that allows us to take advantage of all these ideas