TDDE13 LE2 HT2023 - Distributed AI

Fredrik Heintz

Dept. of Computer Science
Linköping University

fredrik.heintz@liu.se
@FredrikHeintz

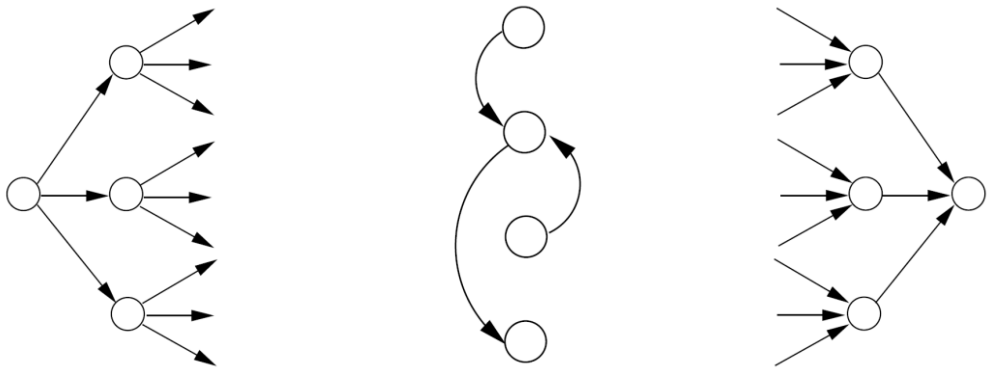
Outline:

- Distributed Problem Solving
- Task Allocation
- UAS Case Study

LINKÖPING UNIVERSITY

Cooperative Problem Solving

- How does a group of agents work together to solve problems?
- If we "own" the whole system, we can design agents to help each other whenever asked. In this case, we can assume agents are benevolent: our best interest is their best interest.
- Problem-solving in benevolent systems is *cooperative distributed problem solving* (CDPS).
- There are three stages:
  - Problem decomposition
  - Sub-problem solution
  - Answer synthesis



decomposition solution synthesis

Distributed Constraint Solving

Distributed Constraint Solving

- CSP: (X, D, C)
  - $X = \{x_1, x_2, \ldots, x_n\}$ *variables*
  - $D = \{d_1, d_2, \ldots, d_n\}$ *domains* (finite)
  - $C = \{c_1, c_2, \ldots, c_r\}$ *constraints*

  For each $c \in C$
  - $var(c) = \{x_i, x_j, \ldots, x_k\}$ *scope*
  - $rel(c) \in d_i \times d_j \times \ldots \times d_k$ *permitted tuples*
- Solution: total assignment satisfying all constraints
- DisCSP: (X, D, C, A, ϕ)
  - $A = \{a_1, a_2, \ldots, a_k\}$ *agents*
  - ϕ: $X \rightarrow A$ maps variables to agents
  - c is known by agents owning $var(c)$

Distributed Constraint Solving

Common assumptions:

- Agents communicate by *sending messages*
- An agent can send messages to others, *iff it knows their identifiers*
- The *delay* transmitting a message is finite but random
- For any pair of agents, *messages are delivered in the order they were sent*
- Agents *know the constraints in which they are involved*, but not the other constraints
- Each agent owns a *single variable (agents = variables)*
- Constraints are *binary* (2 variables involved)

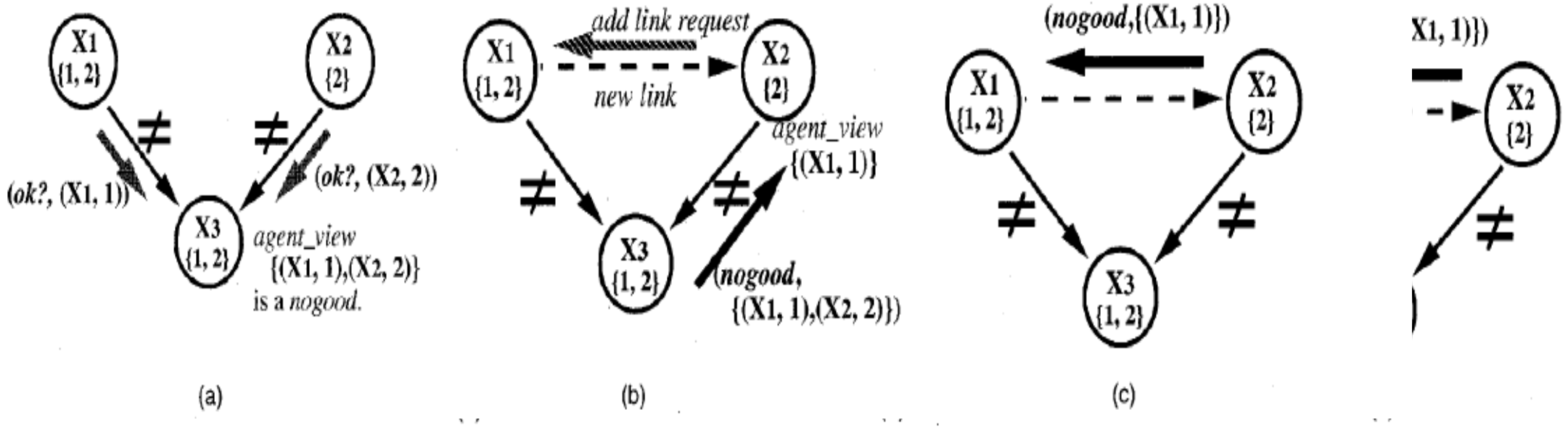
Asynchronous Backtracking

- Each agent starts with instantiated variables, and knows all constraints that concern it
- Agent graph is connected, but not necessarily fully connected. Each agent has a set of values for the agents connected to it by incoming links (agent view)
- Agents can change their values or message agents that are linked to them
- Messages are either *Ok?* or *noGood*

Asynchronous Backtracking

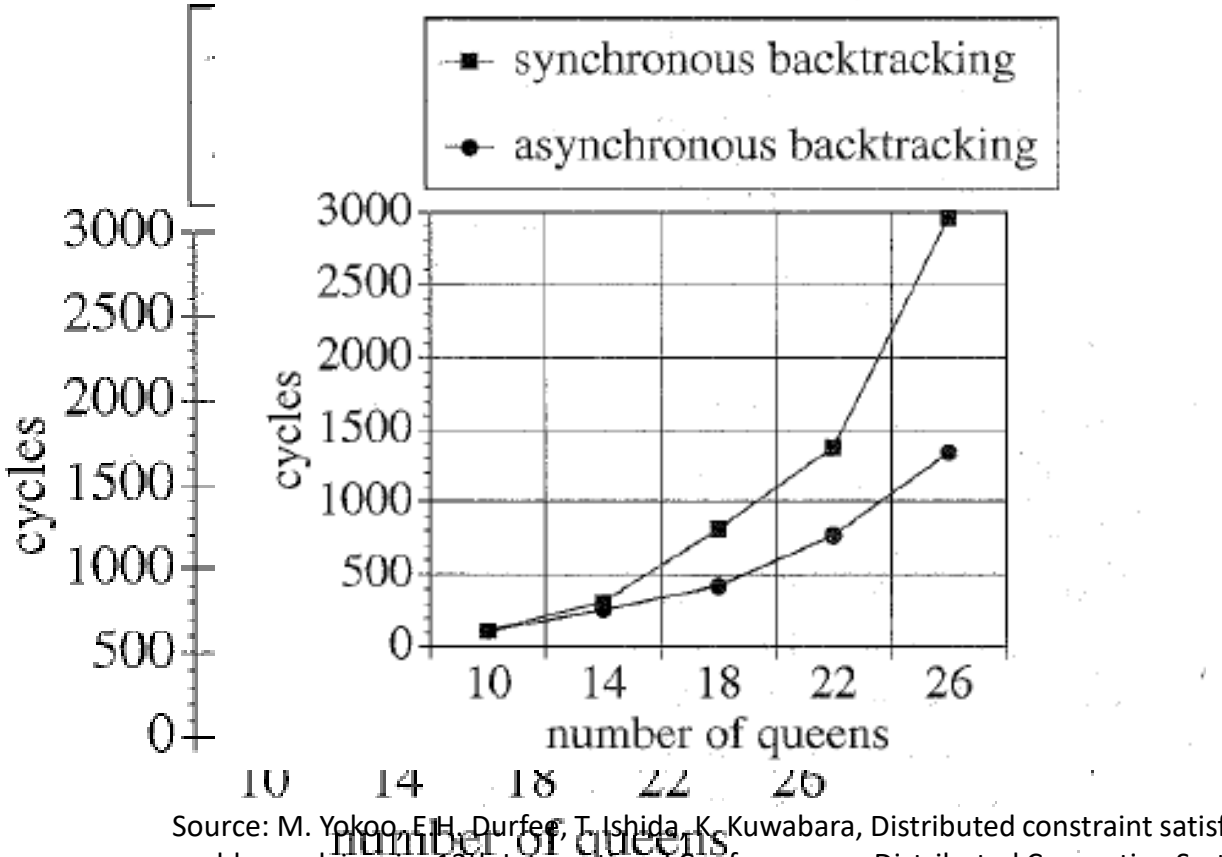
- Agent view: the values of all agents linked to a particular agent
- Message Handling
  - *Ok?* -> Agent wants to know if it can assign a certain value to itself, so it asks another agent
    - Receiving agent updates agent view and checks for consistency, makes sure updated agent view is not a “noGood”
    - Oks only sent to lower priority agents
  - *NoGood* -> in evaluating an Ok? Message, an agent cannot find a value for itself that is consistent, then its updated agent view is noGood and a NoGood (backtracking) message is sent to another agent.
    - Nogoods only sent to higher priority agents
- NoGoods can be seen as derived constraints
- Preventing infinite loops by having a total order among agents for communication
  - Only need to know order of agents that one agent is linked to

Example: Asynchronous Backtracking



Source: M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, Distributed constraint satisfaction for formalizing distributed problem solving, in: 12th International Conference on Distributed Computing Systems (ICDCS-92), 1992, pp. 614–621.

Comparison



Source: M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, Distributed constraint satisfaction for formalizing distributed problem solving, in: 12th International Conference on Distributed Computing Systems (ICDCS-92), 1992, pp. 614–621.

Asynchronous Weak-Commitment Search (AWC)

- Improvement over asynchronous backtracking
- Uses local dynamic priority values rather than static global ordering
- When an agent generates a nogood value, it promotes itself within its local network
- In ABT, an agent backtracks at dead-ends by sending a *nogood* to a higher priority agent
- in AWC, an agent gives up the attempt to satisfy its constraints and delegates the problem to other agents by raising its own priority

Comparison

COMPARISON BETWEEN ASYNCHRONOUS BACKTRACKING AND ASYNCHRONOUS WEAK-COMMITMENT SEARCH (DISTRIBUTED N-QUEENS)

	asynchronous backtracking		asynchronous backtracking with min-conflict heuristic		asynchronous weak-commitment	
n	ratio	cycles	ratio	cycles	ratio	cycles
10	100%	105.4	100%	102.6	100%	41.5
50	50%	325.4	56%	326.8	100%	59.1
100	14%	510.0	30%	504.3	100%	50.8
1000	0%	—	16%	323.8	100%	29.6

Source: M. Yokoo, E.H. Durfee, T. Ishida, K. Kuwabara, Distributed constraint satisfaction for formalizing distributed problem solving, in: 12th International Conference on Distributed Computing Systems (ICDCS-92), 1992, pp. 614–621.

Distributed Constraint Solving

- Exact algorithms (DisCSP and DCOP)
  - Asynchronous Backtracking (ABT),
  - Asynchronous Weak-Commitment Search (AWCS),
  - Asynchronous Distributed Optimization (ADOPT, BnB-ADOPT),
  - Distributed Pseudotree Optimization Procedure (DPOP)
- Approximate Algorithms with quality guarantees
  - k-optimality,
  - Bounded max-sum
- Approximate Algorithms without quality guarantees
  - Distributed Stochastic Algorithm (DSA),
  - Max-Sum

Task Allocation

Example Scenario: Search and Relief

Mission: First scan Area for survivors, then deliver emergency supplies to the survivors.

Multi Robot Task Allocation

- Given
  - n tasks,
  - m robots, and
  - a global objective function

  allocate the tasks so that the objective function is maximized (or minimized).
- Gerkey and Matarić (2004) classified multi robot task allocation along three dimensions:
  - Single-task robots (ST) vs. multi-task robots (MT)
  - Single-robot tasks (SR) vs. multi-robot tasks (MR)
  - Instantaneous assignment (IA) vs. Time-extended assignment (TA)

Multi Robot Task Allocation: ST-SR-IA

- Given
  - n independent tasks,
  - $m \geq n$ robots, and
  - a utility function $u(i,j)$ representing the utility for robot j doing task i

  assign every task to exactly one robot so that the total utility is maximized.
- Optimal Assignment Problem which can be solved in $O(mn^2)$ time by Kuhn's Hungarian method (1955).
- Example: m UAVs delivering $n \leq m$ boxes.

Multi Robot Task Allocation: ST-SR-TA

- Given
  - n independent tasks,
  - $m<n$ robots, and
  - a cost function $c(i,j)$ representing the cost (time) for robot j doing task i,

  create a schedule of tasks for each robot so that the total cost is minimized.
- Scheduling Problem $R||\sum w_j C_j$ which is NP-hard.
- Using an optimal assignment for the first m tasks and a greedy assignment for the rest as robots finish their tasks produces a 3-competitive solution.
- Example: m UAVs delivering $n>m$ boxes.

Multi Robot Task Allocation: ST-MR-IA

- Given
  - n independent tasks,
  - m robots, and
  - a utility function $u(i, \{j_1, \ldots, j_k\})$ representing the utility for the coalition consisting of robots $j_1, \ldots, j_k$ together doing task i,

  find a set of mutually exclusive coalitions maximizing the utility.
- Set Partition Problem which is NP-hard.
- Assumes that the utility for each coalition is known. Coalitional game theory is a very active research area.
- Example: m UAVs delivering n boxes where some boxes require several UAVs to be carried.

Multi Robot Task Allocation Summary

Problem	Optimization Problem (Gerkey & Matarić)	Complexity
ST-SR-IA	Optimal Assignment	$O(mn^2)$
ST-SR-TA	Scheduling: $R\|\|\sum w_j C_j$	NP-hard
ST-MR-IA	Coalition + Set Partitioning	NP-hard
ST-MR-TA	Coalition + Scheduling $MPTm\|\|\sum w_j C_j$	NP-hard
MT-SR-IA	Coalition + Set Partitioning	NP-hard
MT-SR-TA	Coalition + Scheduling $MPTm\|\|\sum w_j C_j$	NP-hard
MT-MR-IA	Coalition + Set Covering	NP-hard
MT-MR-TA	Coalition + Scheduling $MPTmMPMn\|\|\sum w_j C_j$	NP-hard

- IA problems correspond to assignment problems while TA problems correspond to scheduling problems.
- MR/MT problems also involve a coalition forming problem.

Complex Task Allocation

- Interrelated utilities
  - The utility depends on all tasks allocated to a robot or even on the allocation to other robots.
  - Example: When delivering boxes, the time it takes depends on the location of the UAV at the start of the task. And this depends on the previous task of the UAV.
  - Combinatorial optimization problem.
- Task constraints
  - There are dependencies between tasks such as precedence constraints, timing constraints and communication constraints.
  - Example: First deliver box1 and then within 10 minutes deliver box2.
  - Constraint satisfaction/optimization problem

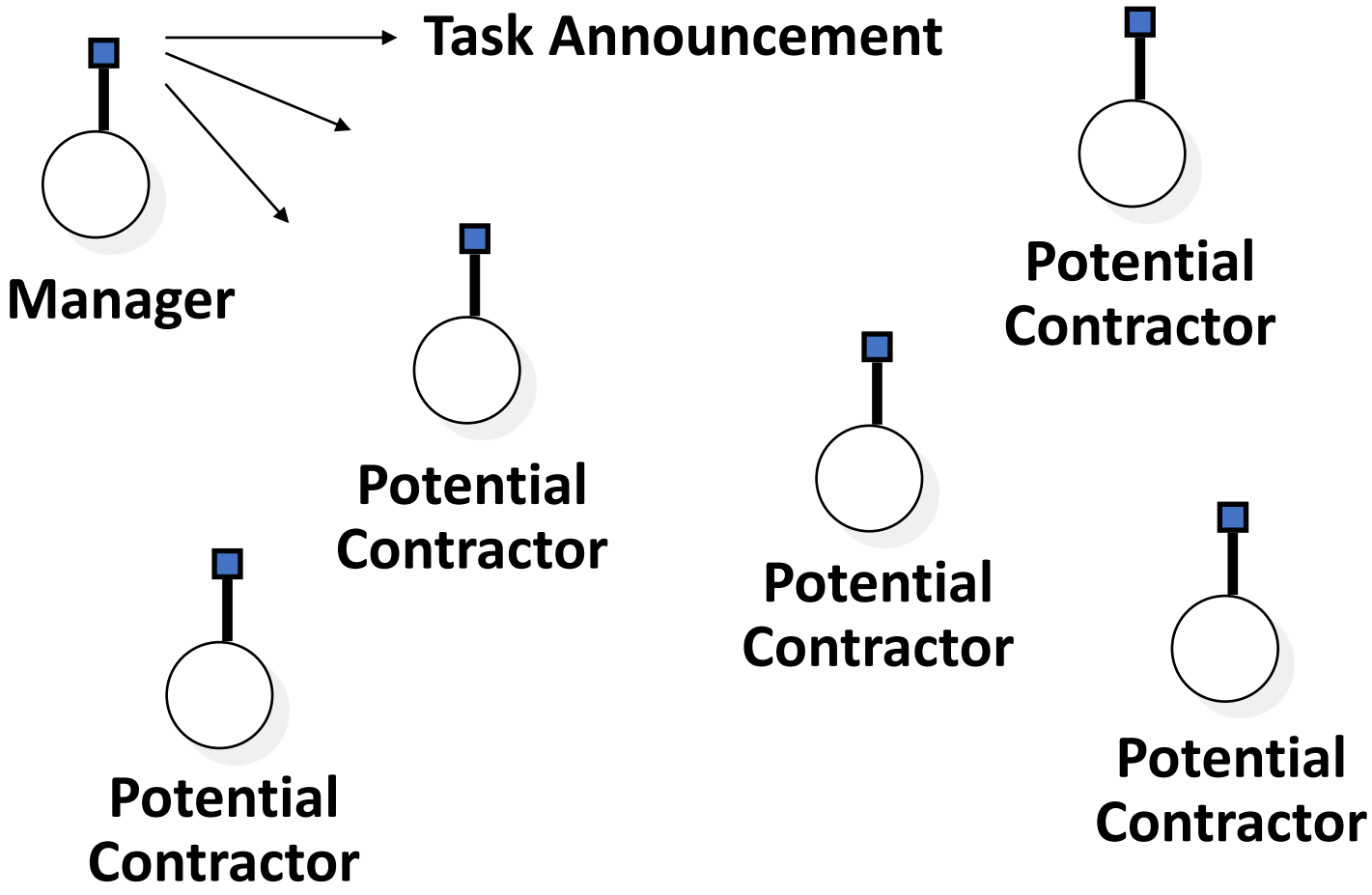
Complex Task Allocation

- Complex tasks
  - Tasks can be achieved in many possible ways.
  - Example: To deliver boxes a UAV can either deliver them directly or use a carrier which can load several boxes.
  - Task decomposition problem (planning problem)
- Uncertainty
  - The actual utility or cost might not be known.
  - Example: UAV1 needs between 8 and 12 minutes to deliver box2.
  - Stochastic optimization
- Multi-dimensional cost and utility functions
  - Example: Maximize the utility of the mission while minimizing the resource usage.
  - Multi-criteria optimization

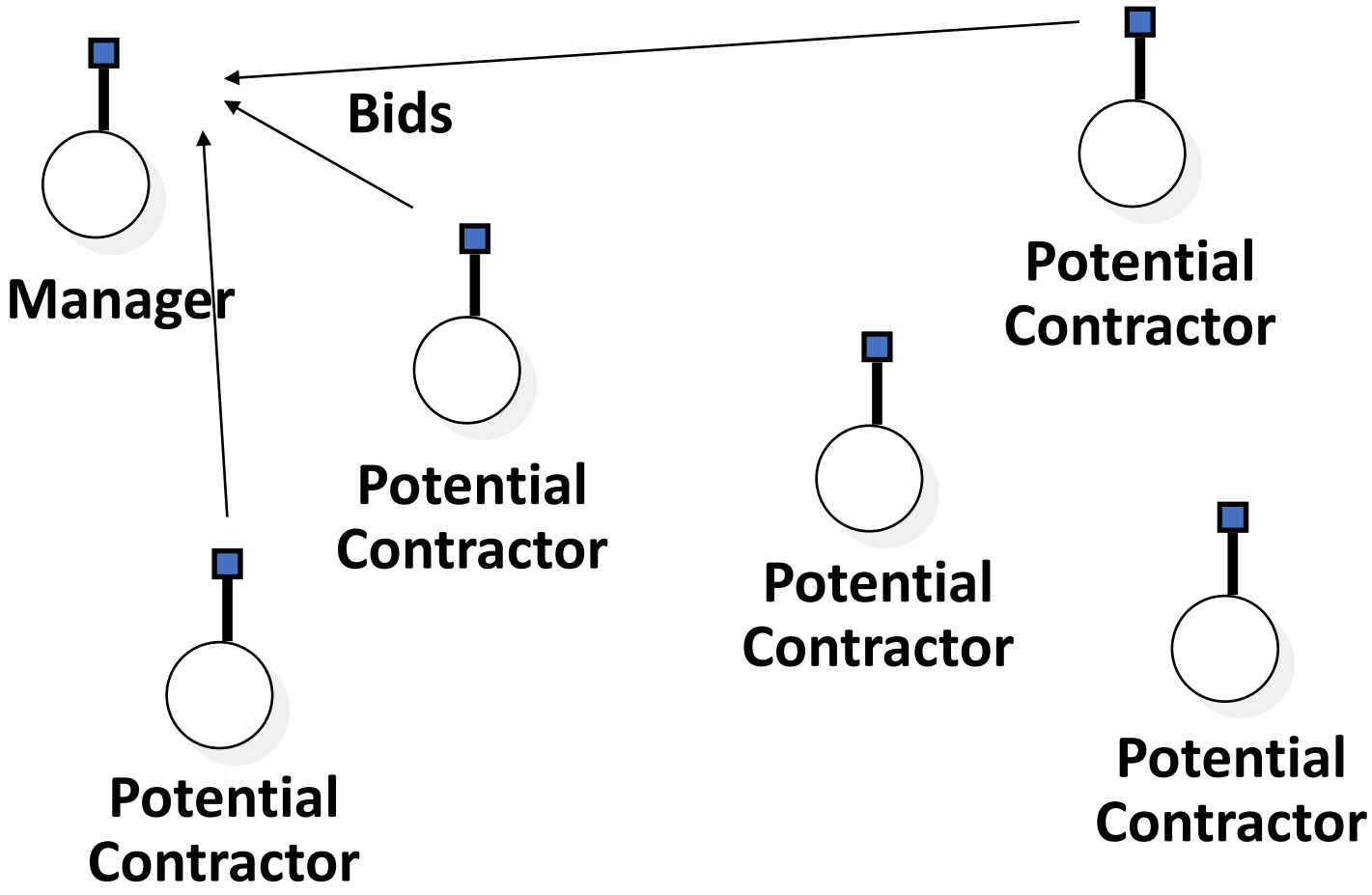
Market-Based MRTA Approaches

- The general idea is to create a market where tasks can be traded in such a way that a global objective function is optimized.
- Each robot has an individual utility function that specifies that robot's preferences based on information available to the robot. For example, maximize the revenue minus the cost for each task. This is the basis for the bidding rule of a robot.
- The auctioneer determines who is awarded a task based on the robots' bids (winner determination).
- Auctions are communication and computation efficient.
- The objective of the system designer is to engineer the costs, revenues, and auction mechanism in such a way that individual self-interest leads to globally efficient solutions.
- Many market mechanisms follow the Contract Net Protocol.

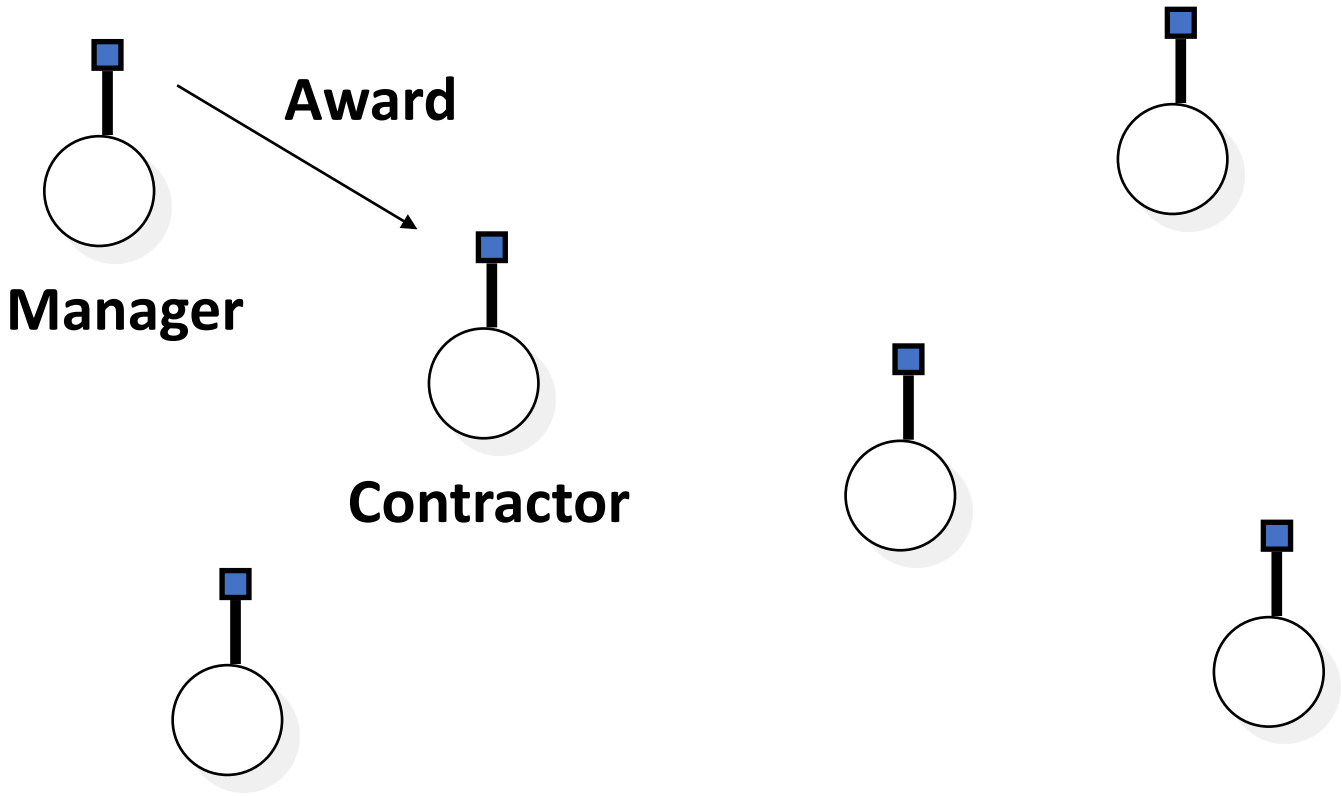
A Manager Announces a Task



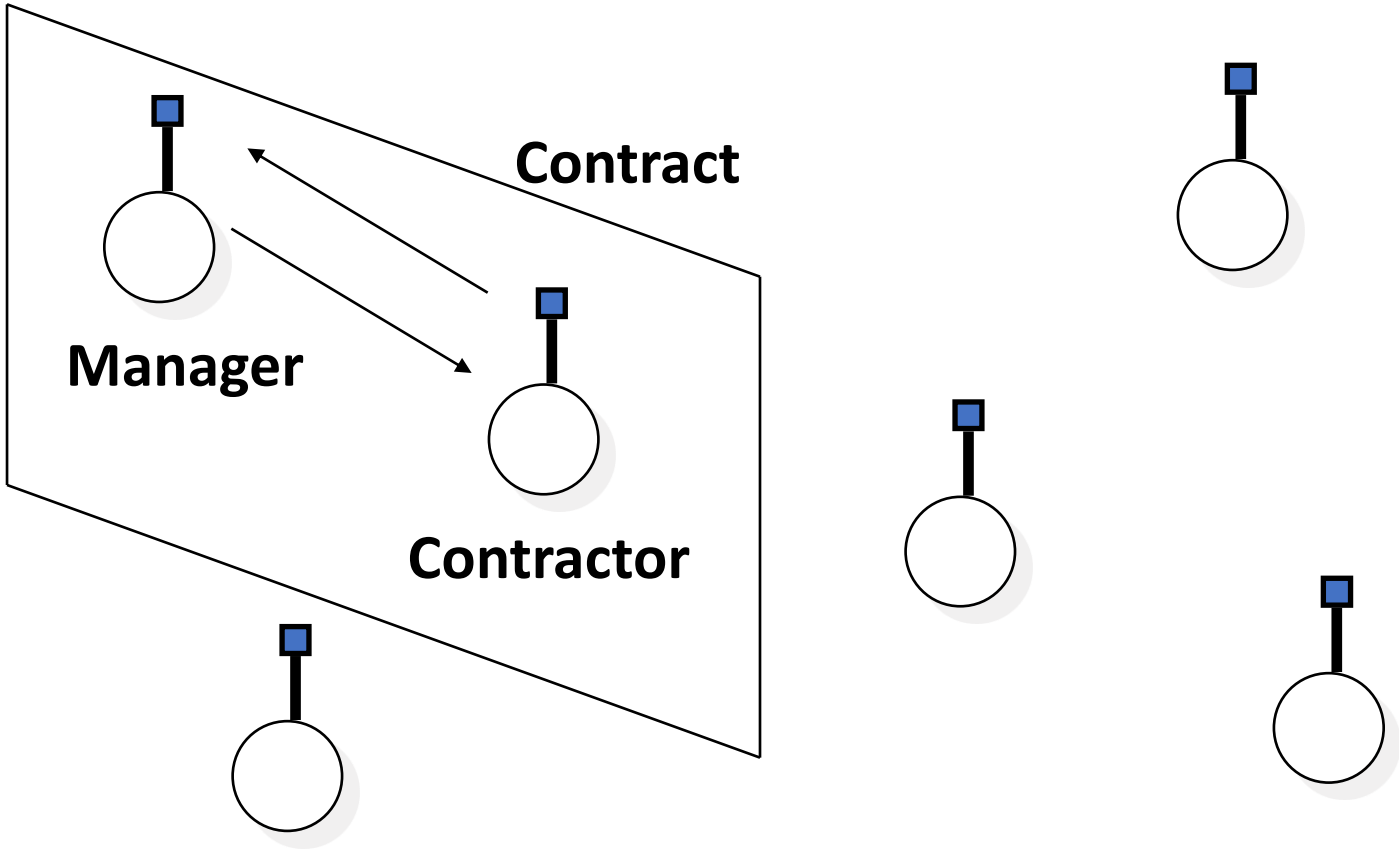
Potential Contractors Submit Bids



The Manager Awards the Contract



A Contract is Established

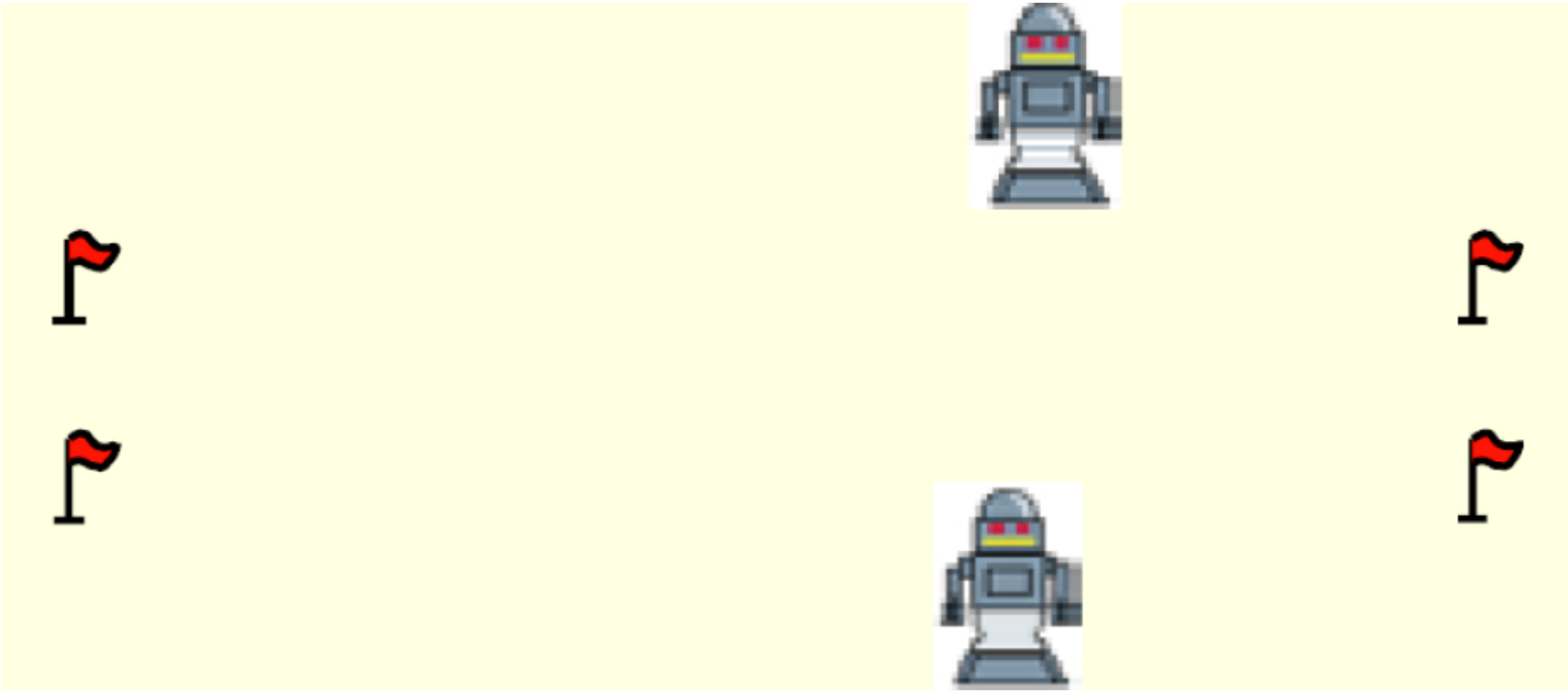


Market-Based MRTA: Single Item Auction

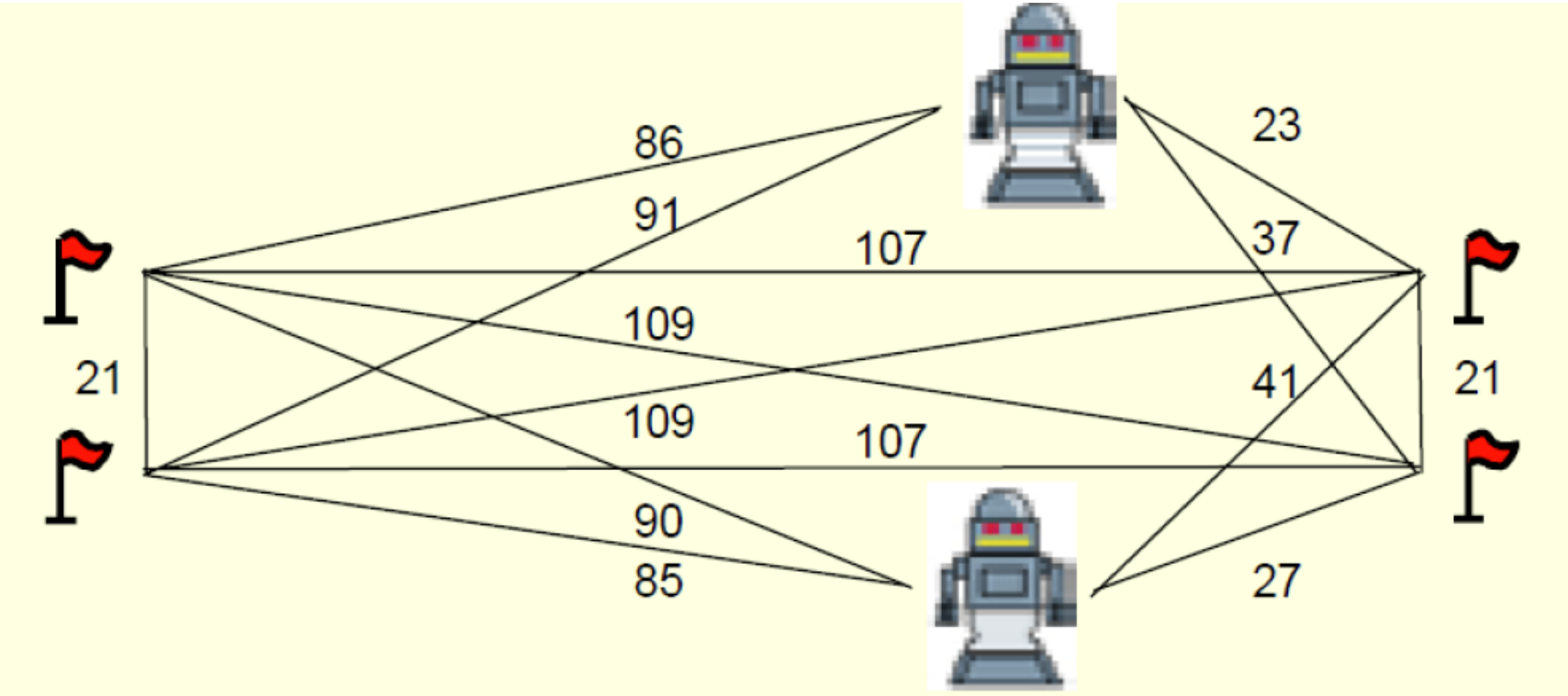
- Given
  - n tasks,
  - m robots,
  - and a bidding rule for individual tasks

  Auction out the tasks sequentially and allocate each task according to the best bid.
- Computational cost:
  - Bid valuation v (the cost to compute a single bid)
  - Winner determination $O(m)$
  - Number of auctions n
- An important question is how to design bidding rules to optimize a global objective function.

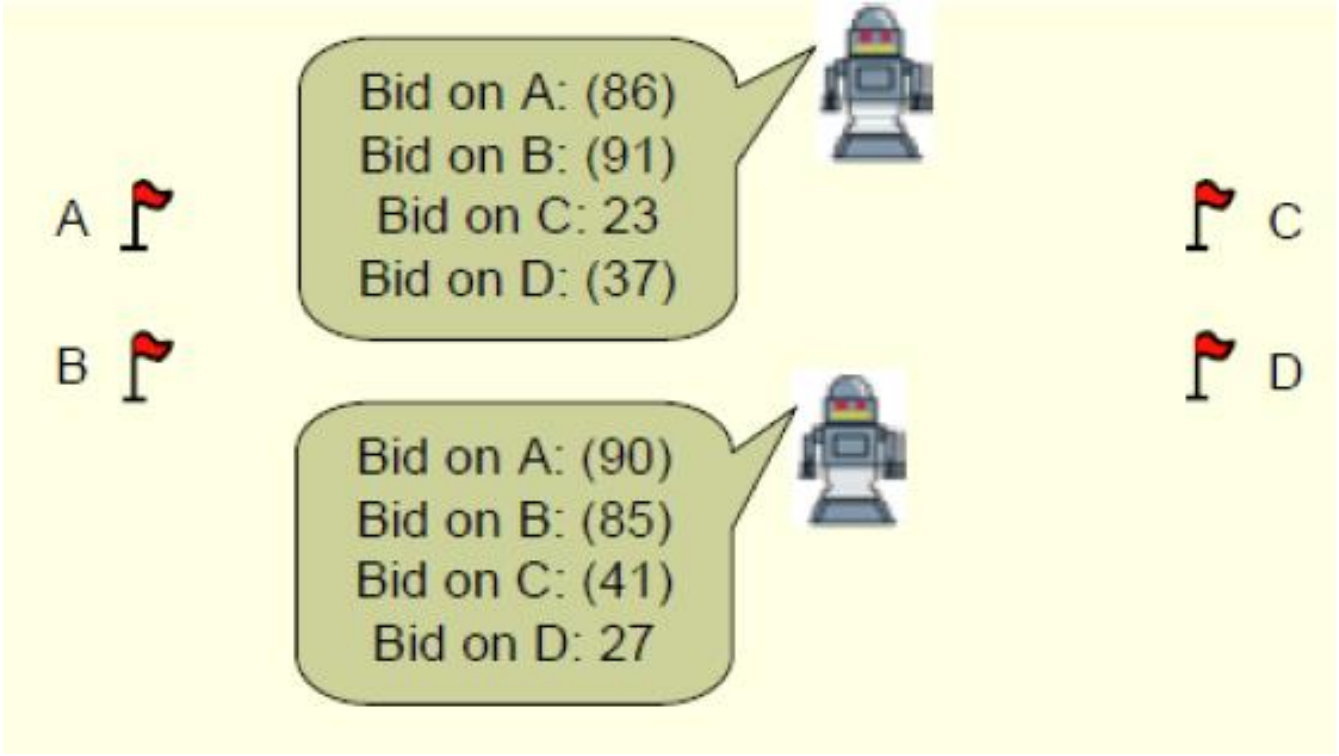
Market-Based MRTA: Single Item Auction



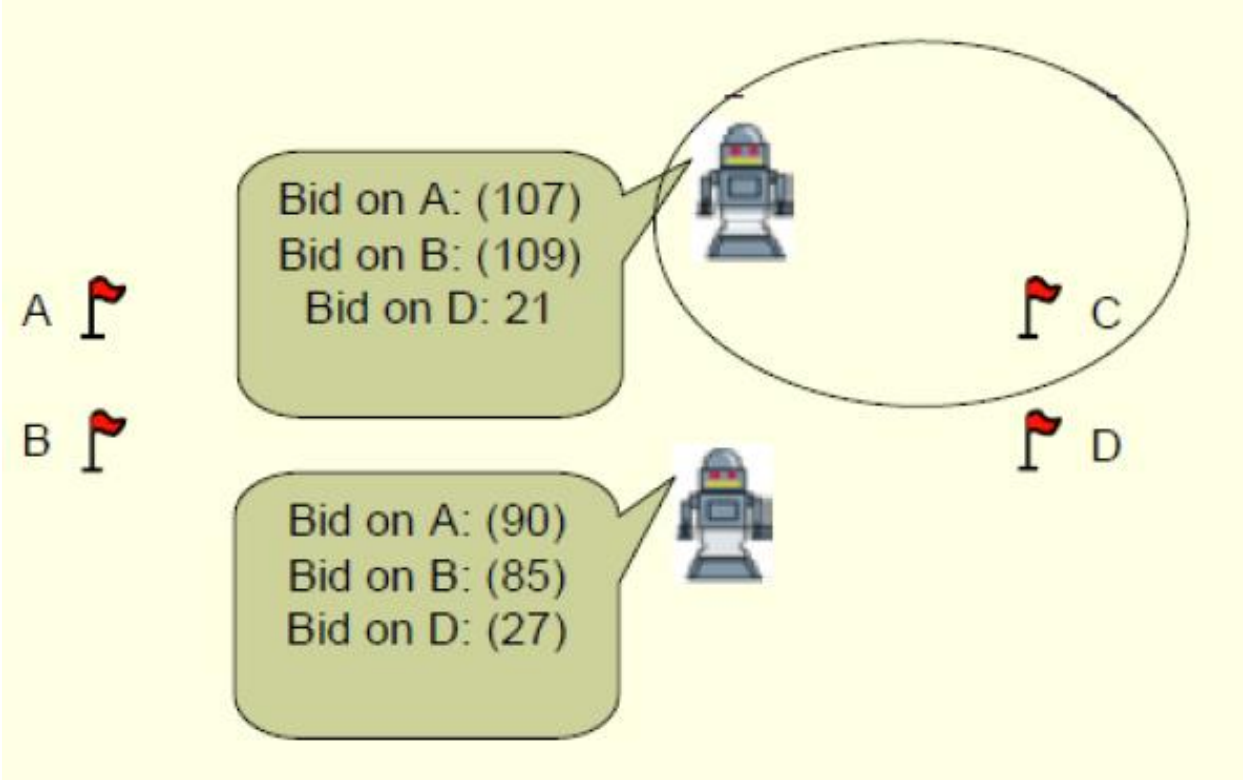
Market-Based MRTA: Single Item Auction



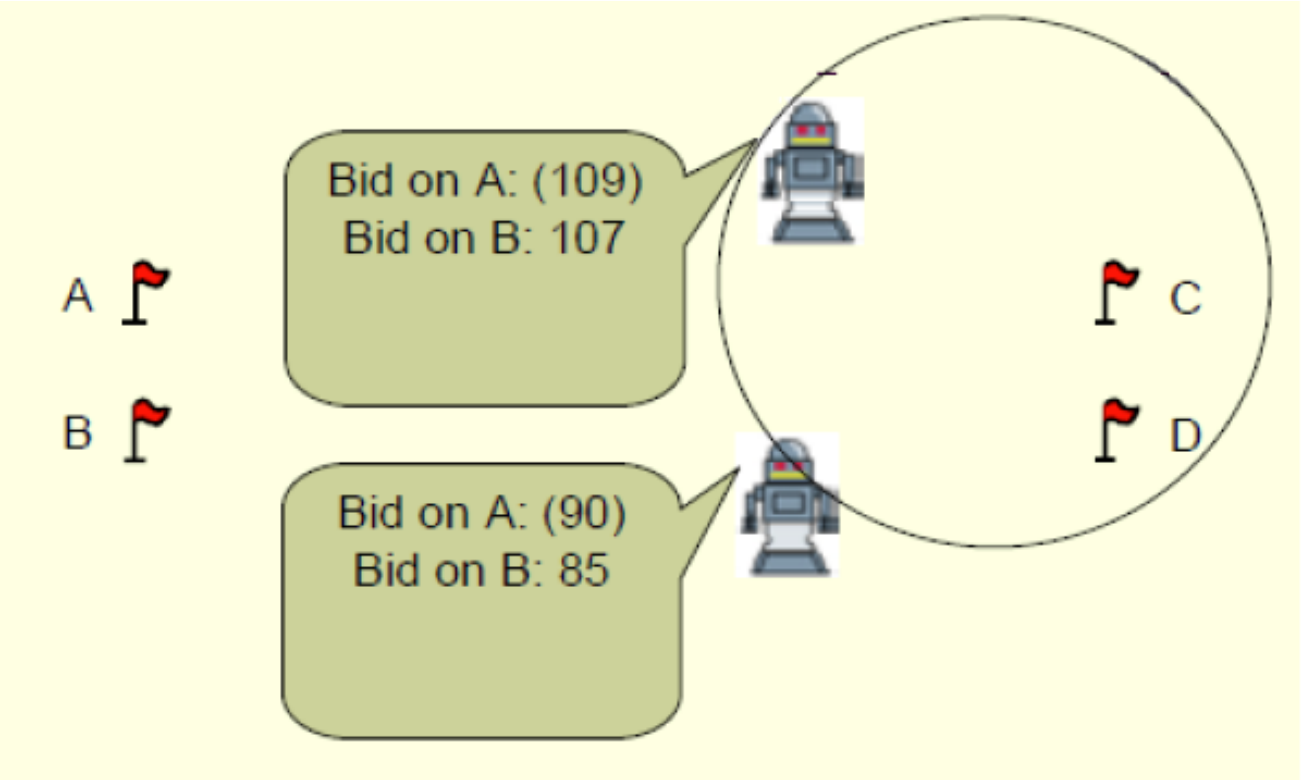
Market-Based MRTA: Single Item Auction



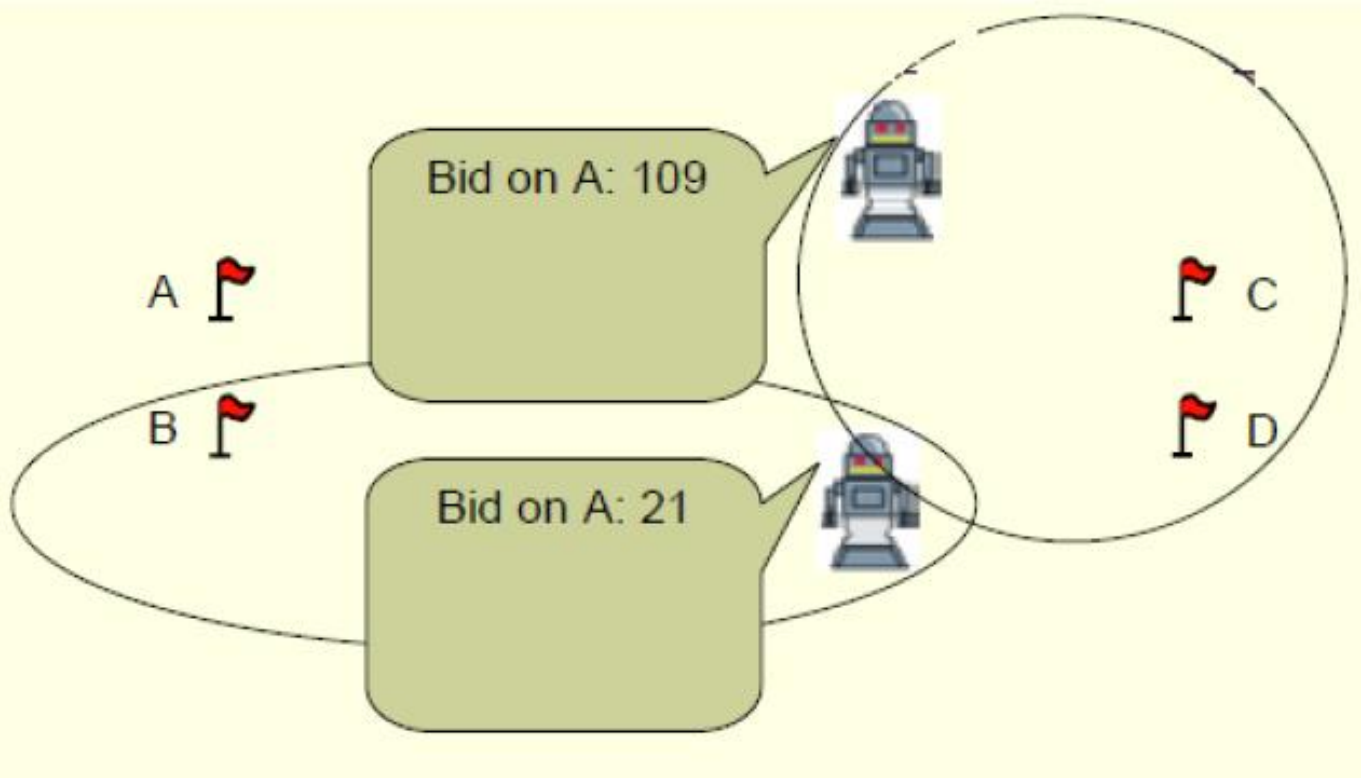
Market-Based MRTA: Single Item Auction



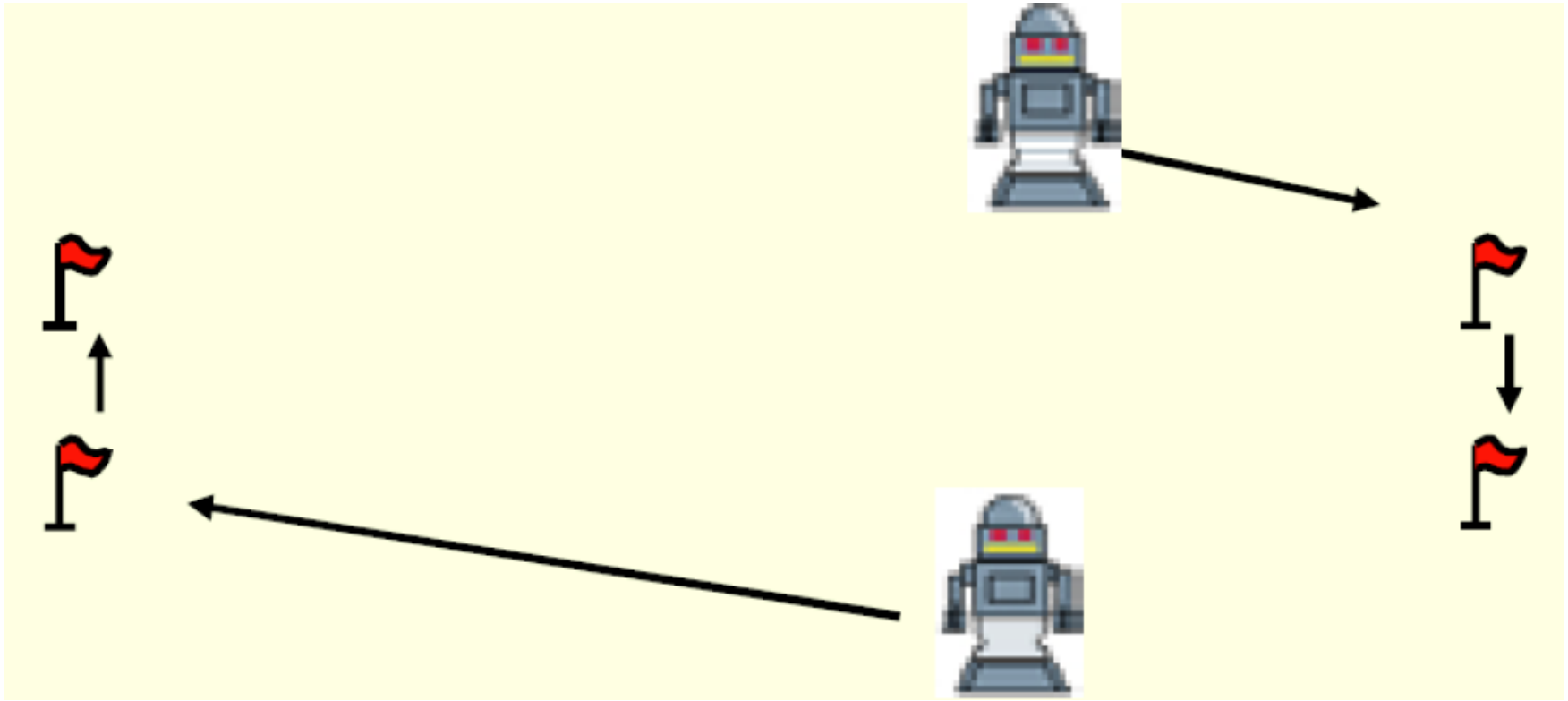
Market-Based MRTA: Single Item Auction



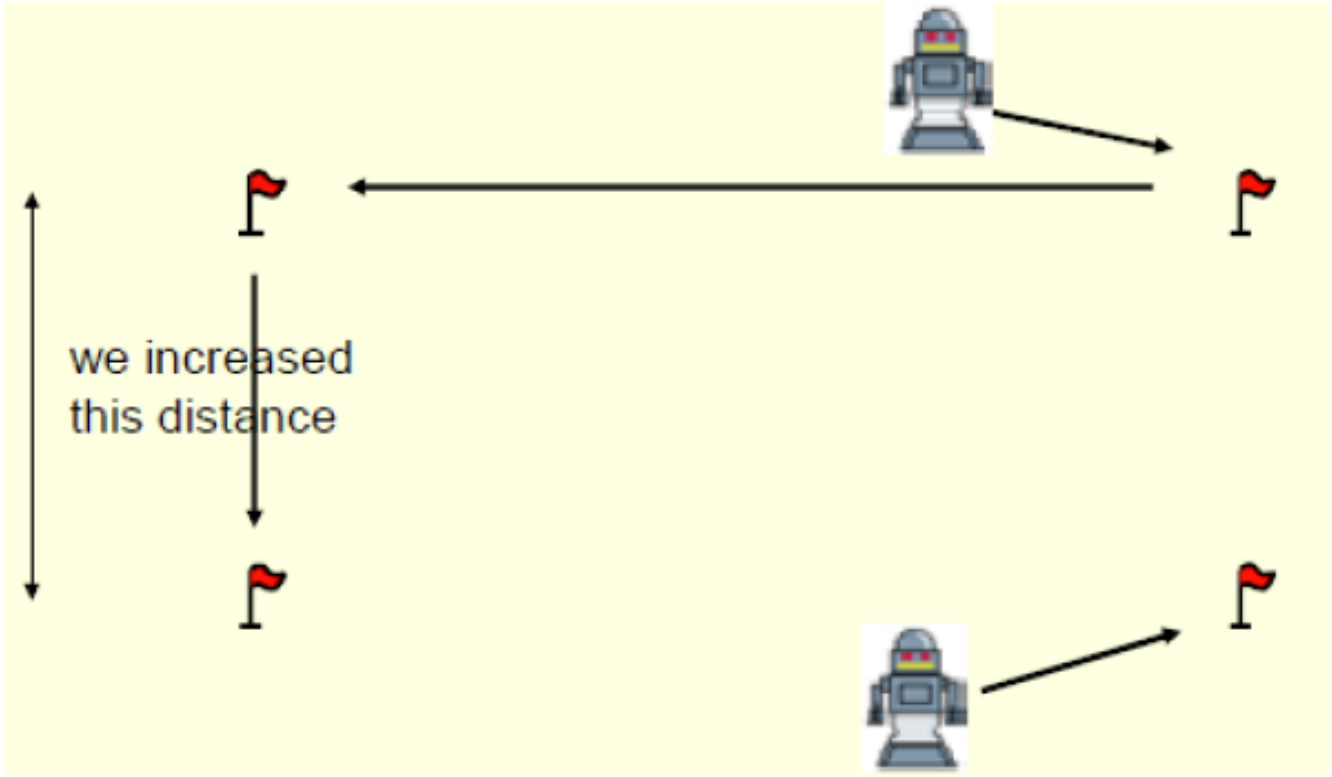
Market-Based MRTA: Single Item Auction



Market-Based MRTA: Single Item Auction



Market-Based MRTA: Single Item Auction



Market-Based MRTA: Single Item Auction

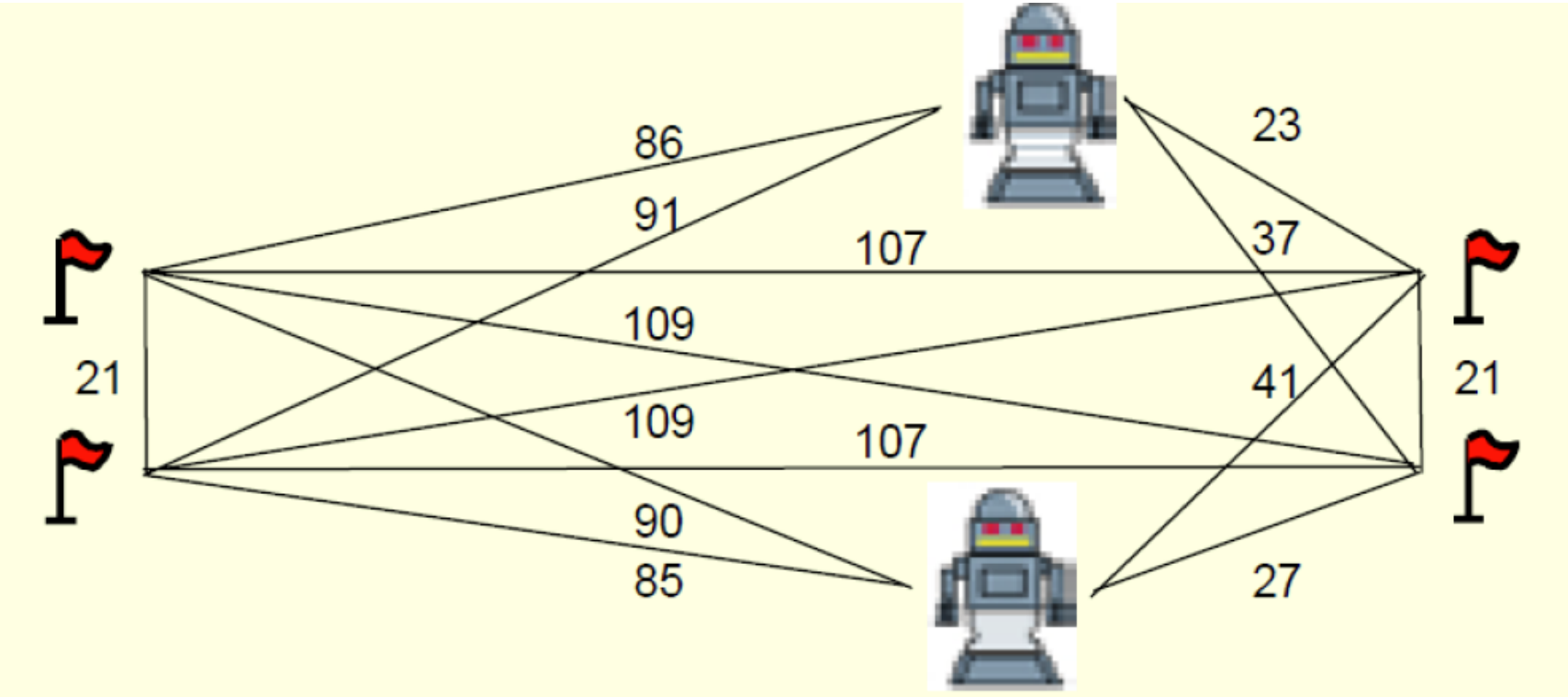
- **MiniSum**
  - Minimize the sum of the path costs over all robots
  - Minimization of total energy or distance
  - Application: planetary surface exploration
- **MiniMax**
  - Minimize the maximum path cost over all robots
  - Minimization of total completion time (makespan)
  - Application: facility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

Market-Based MRTA: Combinatorial Auction

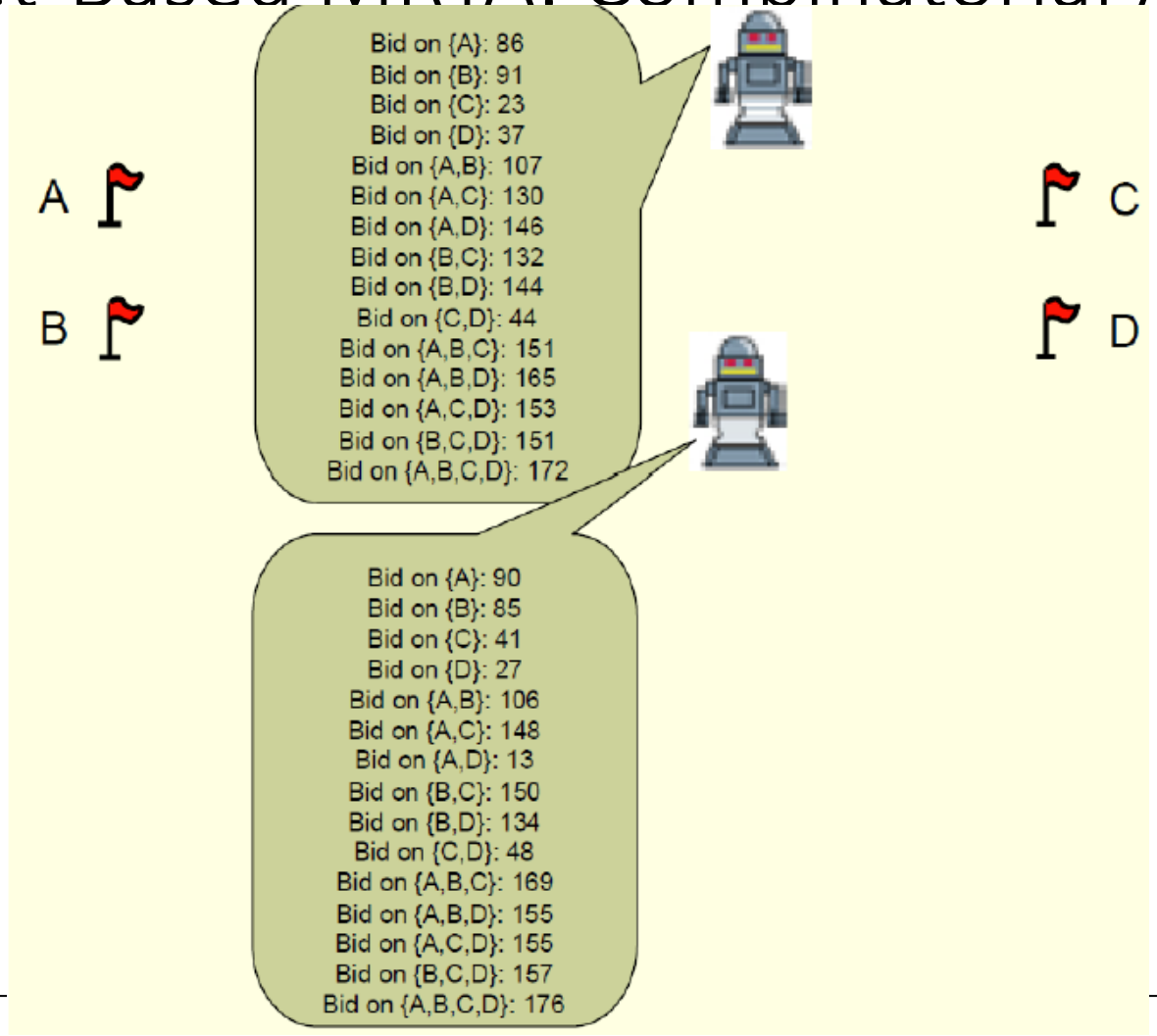
- A combinatorial auction is an auction where bidders are allowed to bid on combinations of items, or bundles, instead of individual items.
- Given
  - n tasks,
  - m robots,
  - and a bidding rule for bundles of tasks

  Auction out all tasks and allocate bundles to robots so that the sum of the bids is maximized.
- Computational cost:
  - Bid valuation $O(2^n v)$ (v is the cost to compute a single bid)
  - Winner determination $O((b+n)^n)$ (b is the number of bids)
  - Number of auctions 1

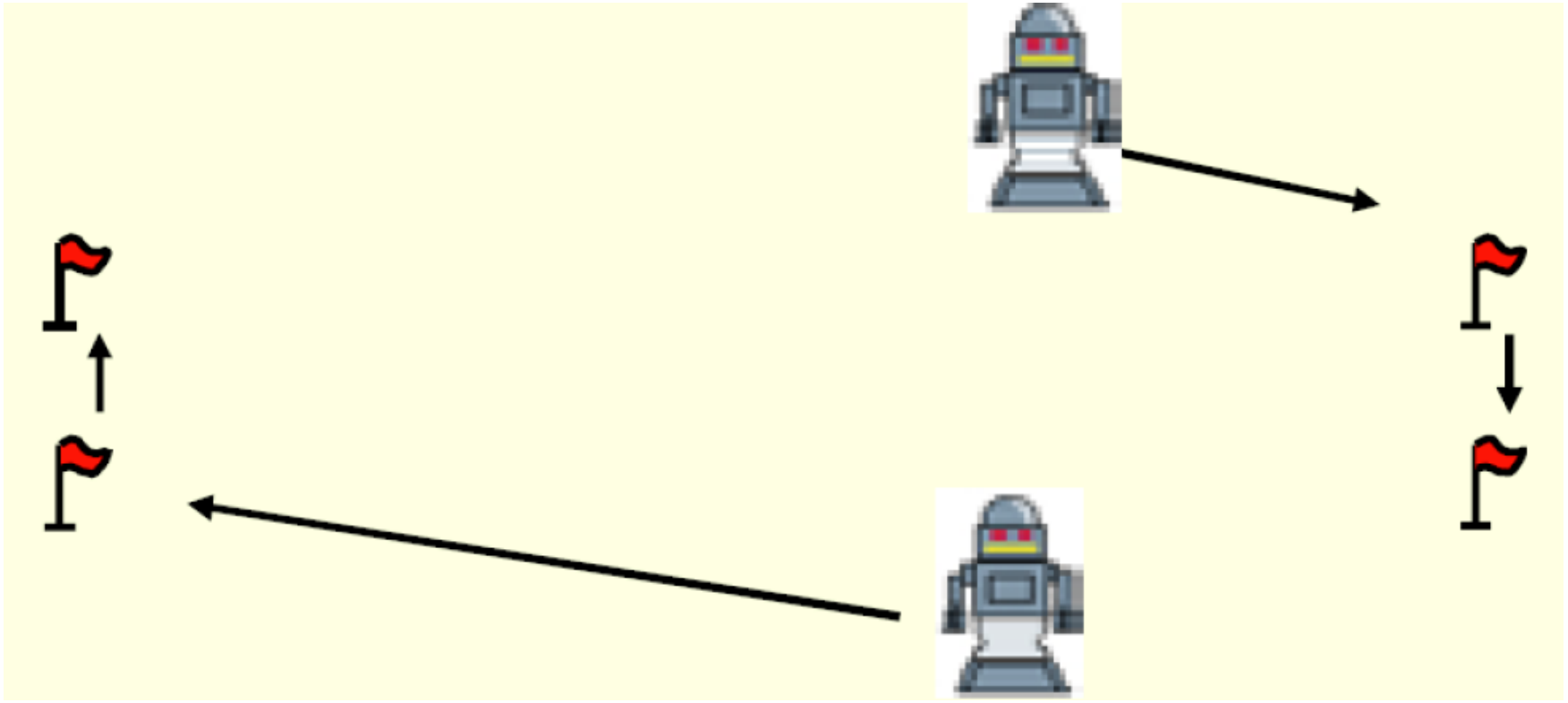
Market-Based MRTA: Combinatorial Auction



Market-Based MRTA: Combinatorial Auction

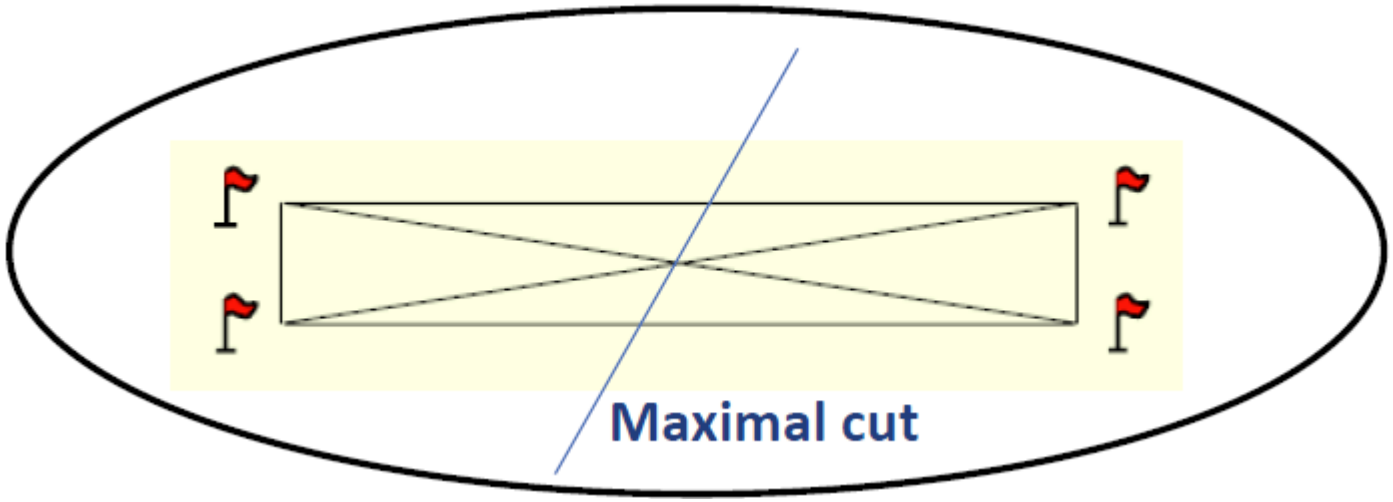


Market-Based MRTA: Combinatorial Auction

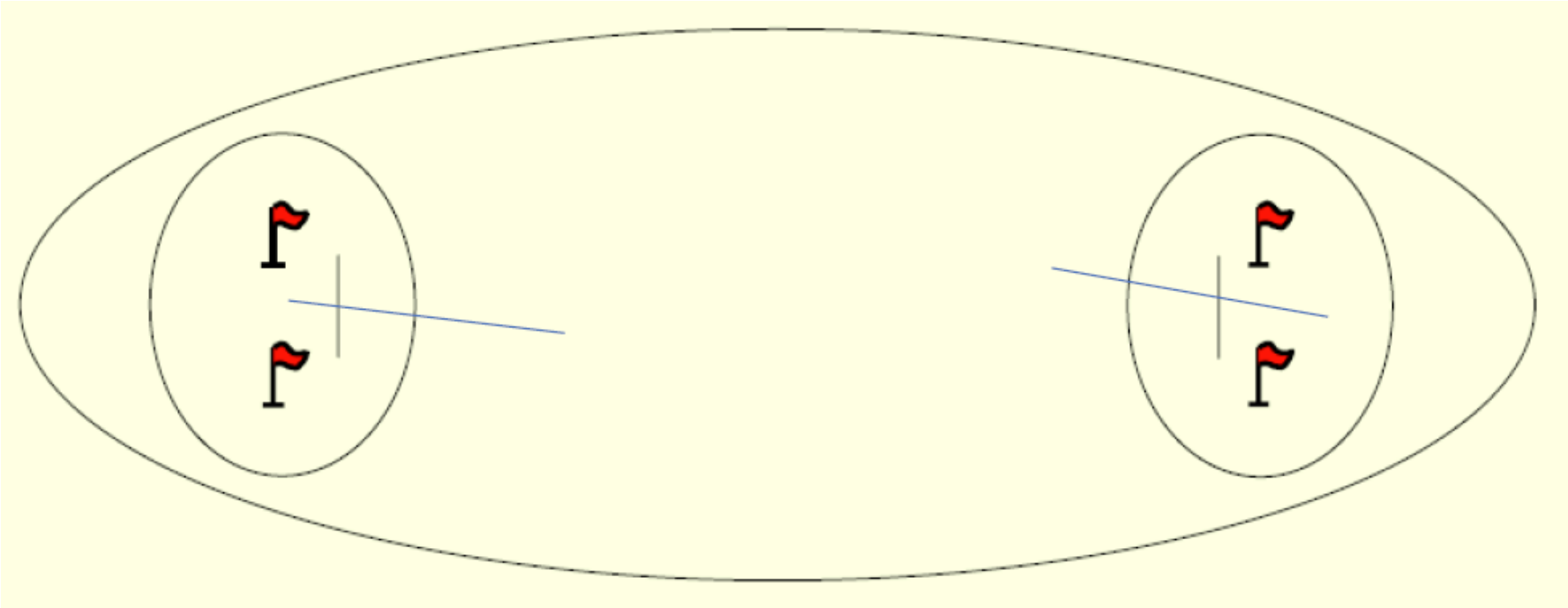


Market-Based MRTA: Combinatorial Auction

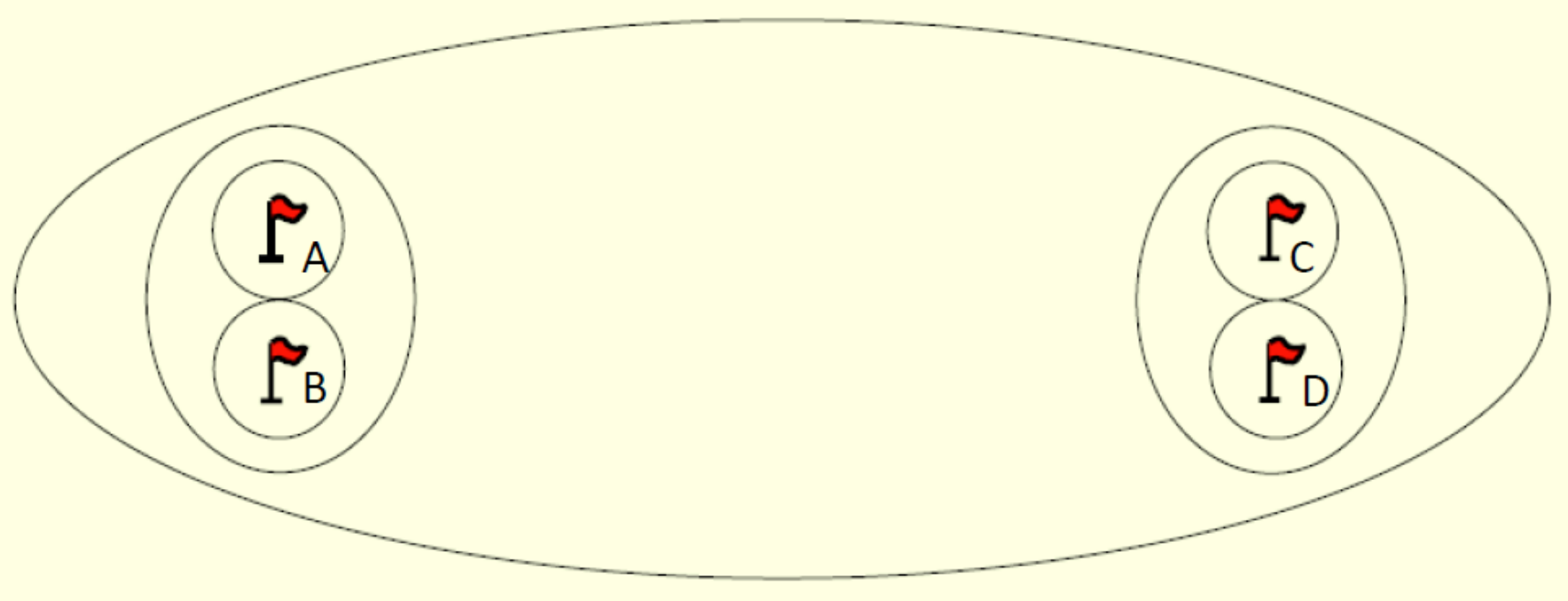
- Example bidding strategies for robot exploration
  - ***Single***: Each robot bids its surplus for a target (the reward for the target minus the travel cost from its current location).
  - ***Three-Combination***: Bid on all bundles with no more than *3* targets.
  - ***Graph-Cut***: Generate a complete undirected graph from the targets with the travel cost as the edge cost. Generate bundles by recursively using the max cut algorithm to split the graph into two connected subgraphs.



Market-Based MRTA: Combinatorial Auction

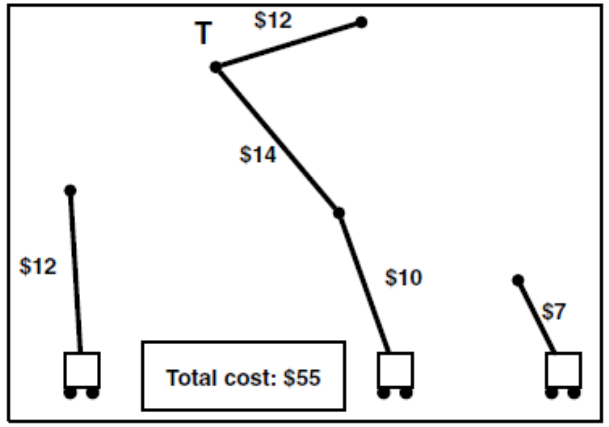


Market-Based MRTA: Combinatorial Auction



Market-Based MRTA: Peer to Peer Trading

- Given an initial allocation of tasks to robots, a robot may reallocate its tasks by creating an auction for them.
- Corresponds to local search: each exchange decreases the solution cost while maintaining a feasible solution.



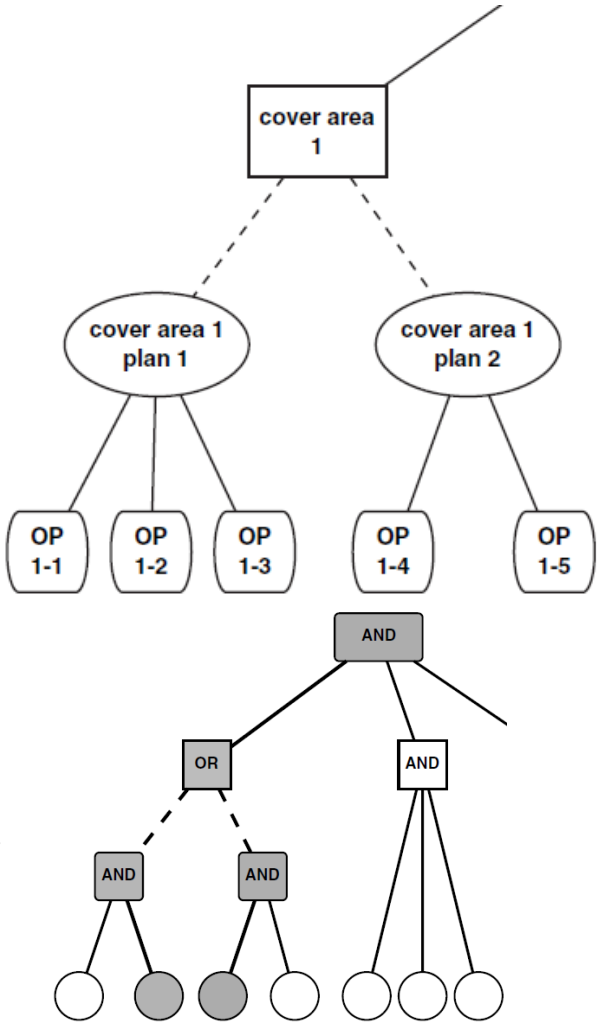
- Improves allocations when there is uncertain, incomplete or changing information.
- Sandholm (1998) proved that with a sufficiently expressive set of local search moves (single-task, multitask, swaps, and multiparty exchanges) the global optimum solution can be reached in a finite (although potentially large) number of steps.

Market-Based MRTA: Complex Tasks

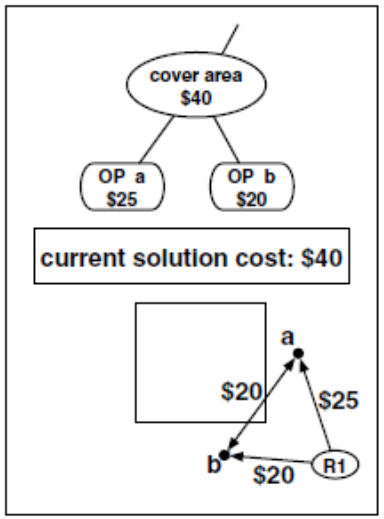
- Zlot and Stentz (2006) suggested an auction for complex task represented by *AND/OR* trees.
- Robots bid on nodes along any root-to-leaf path, which can branch at *OR* nodes.

  Bid on leaf: agree to execute a primitive task

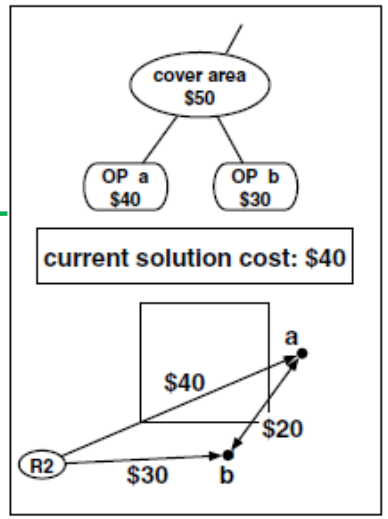
  Bid on *AND/OR*: agree to complete a complex task
- With some more restrictions an *O(LT)* winner determination algorithm is possible (L = #leaves in the tree and T = #tree nodes).
- Example bidding rule: Bid on the node with the highest surplus (bid price minus reserve price) at each level of the tree.
- They also presented a heuristic $O(LT^2)$ winner determination algorithm for the general case, where a robot bids on any or all nodes in a tree.



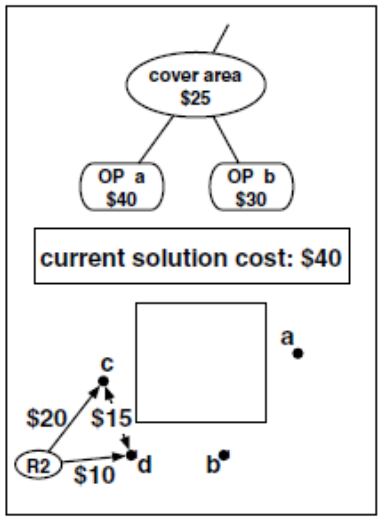
Market-Based MRTA: Complex Tasks



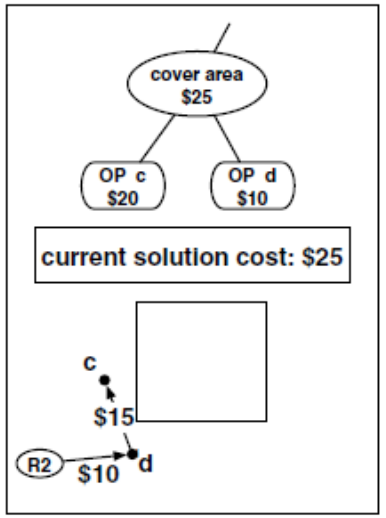
R2's cost given R1's decomposition



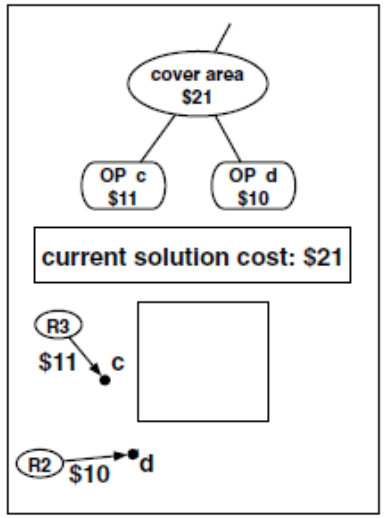
R2's cost given R2's decomposition



New decomposition and allocation



Old decomposition and new allocation



Summary Market-Based MRTA Approaches

Approach	Theoretical guarantees	Experimental results
Combinatorial auctions [16], [34]	Optimal (if all bundles are considered) [5]	Good solutions with limited number of task bundles [16], [34], [40], [41]
Central single task iterated auctions [9], [36]	Approximation bounds for 18 cases (3 objective functions, 6 bidding rules) [9]	Close to optimal results when using the appropriate bidding rules [36]
Central instantaneous assignment (IA) [7], [13]	Optimal possible; commonly used greedy algorithm is a 2-approximation; greedy algorithm for online version is 3-competitive [6]	
Peer-to-peer trading [17], [18], [19], [20], [35]	Optimal solution possible in a finite number of trades with a sufficiently expressive set of contract types [42]	In a limited number of rounds, a combination of single- and multi-task trades outperforms all other combinations of single-task, multi-task, swap, and multi-party contracts [12]; allowing non-individual rational trades can lead to better solutions [43]
Central multi-task auctions followed by peer-to-peer trading [44]		Increasing the maximum number of tasks awarded per multi-task auction results in poorer solution quality [44]

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

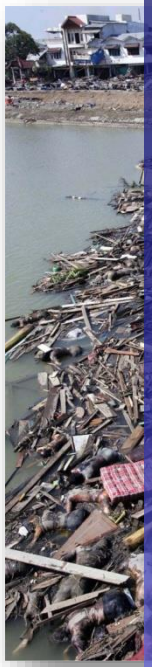
UAS Case Study

Collaborative Unmanned Aircraft Systems

A principled approach to building collaborative intelligent autonomous systems for complex missions.

Collaborative Unmanned Aircraft Systems

A principled approach to building collaborative intelligent autonomous systems for complex missions.



Challenges:

- Support humans and robots including legacy systems
- Support adjustable autonomy and mixed-initiative interaction
- Manage tasks and information on many abstraction levels
- Coordinating control, reaction and deliberation
- Coordination of systems, resources and platforms
- Incomplete information at design time and run time
- Inspection, monitoring, diagnosis and recovery on many abstraction levels

Autonomous Systems at AIICS, Linköping University

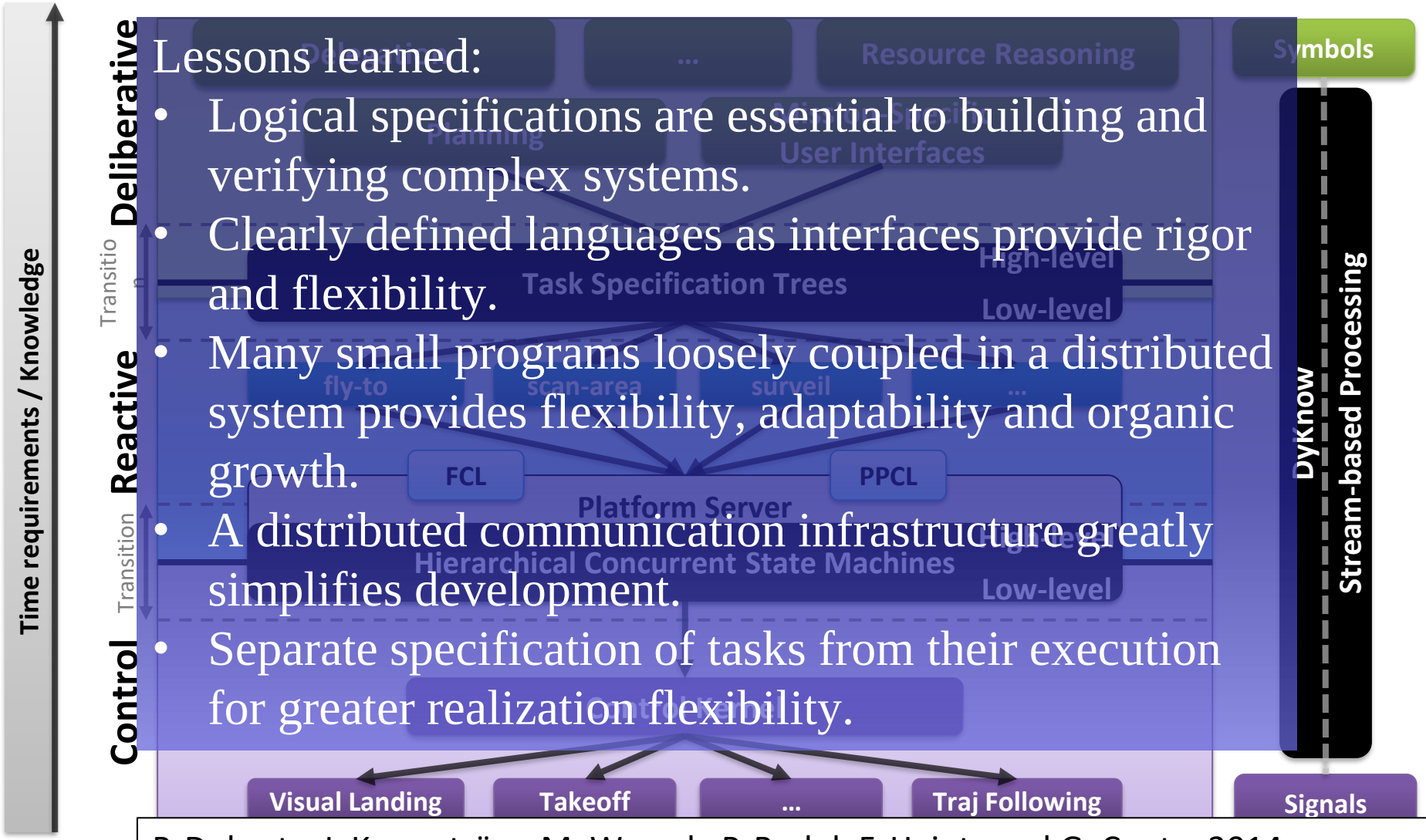
Micro UAVs
weight < 500 g,
diameter < 50 cm

Yamaha RMAX
weight 95 kg,
length 3.6 m

LinkQuad weight ~1 kg, diameter ~70cm


Selected Autonomous Functionalities
Linköping University, Sweden
Department of Computer and Information Science
Autonomous Unmanned Aerial Vehicle Technologies Lab

HDRC3: A Distributed Hybrid Deliberative/Reactive Architecture for Autonomous Systems



Lessons learned:

- Logical specifications are essential to building and verifying complex systems.
- Clearly defined languages as interfaces provide rigor and flexibility.
- Many small programs loosely coupled in a distributed system provides flexibility, adaptability and organic growth.
- A distributed communication infrastructure greatly simplifies development.
- Separate specification of tasks from their execution for greater realization flexibility.

LINKÖPINGS UNIVERSITET

P. Doherty, J. Kvarnström, M. Wzorek, P. Rudol, F. Heintz and G. Conte. 2014.
HDRC3 - A Distributed Hybrid Deliberative/Reactive Architecture for Unmanned Aircraft Systems.
In K. Valavanis, G. Vachtsevanos, editors, Handbook of Unmanned Aerial Vehicles, pages 849–952.

Example Scenario: Search and Relief

Searching for injured people and delivering food, medicine and other supplies are highly prioritized activities in disaster relief.

Example Scenario: Search and Relief

Mission: First scan Area for survivors, then deliver emergency supplies to the survivors.

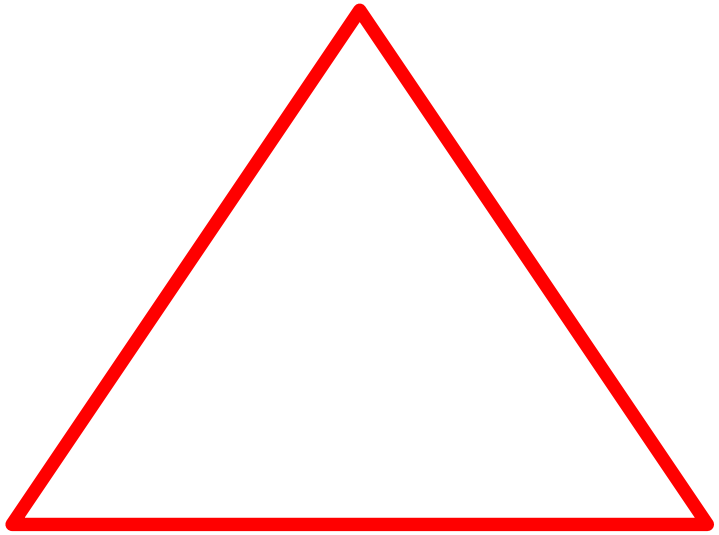
Example Scenario: Search and Relief

Mission: First scan Area for survivors, then deliver emergency supplies to the survivors.

Human-Robot Collaboration

Delegation

Delegate(A, B, task, constraints)



Adjustable Autonomy

By varying the task and constraints parameters the degree of autonomy allowed can be controlled.

Mixed-Initiative Interaction

Delegate(GOP, UAV, task, constraints)
Delegate(UAV, GOP, task, constraints)
Important: Safety, security, trust, etc.

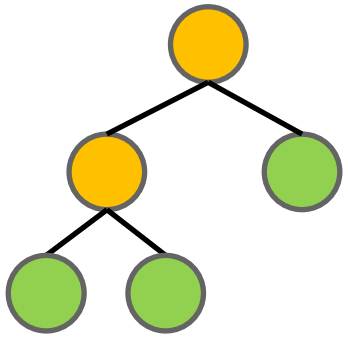
Patrick Doherty, Fredrik Heintz and Jonas Kvarnström. 2013.
High-level Mission Specification and Planning for Collaborative Unmanned Aircraft Systems using Delegation. Unmanned Systems, **1**(1):75–119. World Scientific.

Collaborative Tasks for UAS

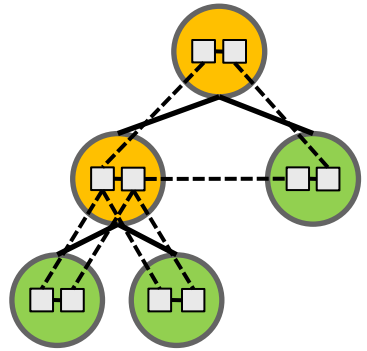
- Tasks need to be
  - general to cover the spectrum from high level goals to detailed plans (*task constraints*),
  - assigned to resource constrained physical platforms (*interrelated utilities*), and
  - expanded and modified as parts of tasks are recursively delegated (*complex tasks*).
- The task representation should
  - be highly flexible, distributed and dynamically extendible and
  - support dynamic adjustment of autonomy.

Task Specification Trees

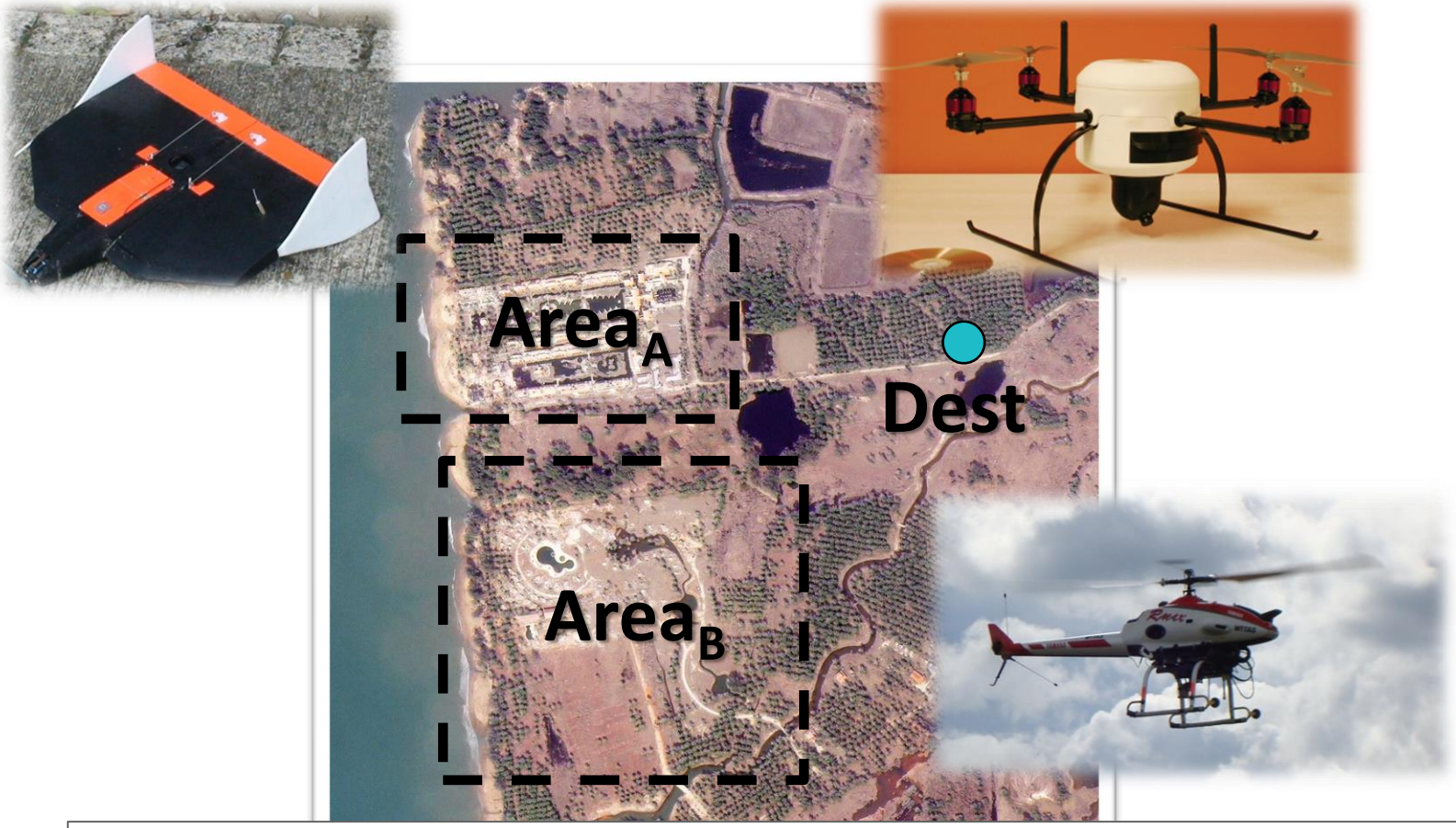
- A Task Specification Tree (TST) is a distributed data structure with a declarative representation that describes a complex multi-agent task.
- A node in a TST corresponds to a task. It has a node interface with parameters and a set of node constraints that restrict the parameters.
- There are currently six types of nodes: Sequence, concurrent, loop, select, goal, and elementary action.
- A TST is associated with a set of tree constraints expressing constraints between tasks in the tree.



Interface:
t_s, t_e, Dest
Speed

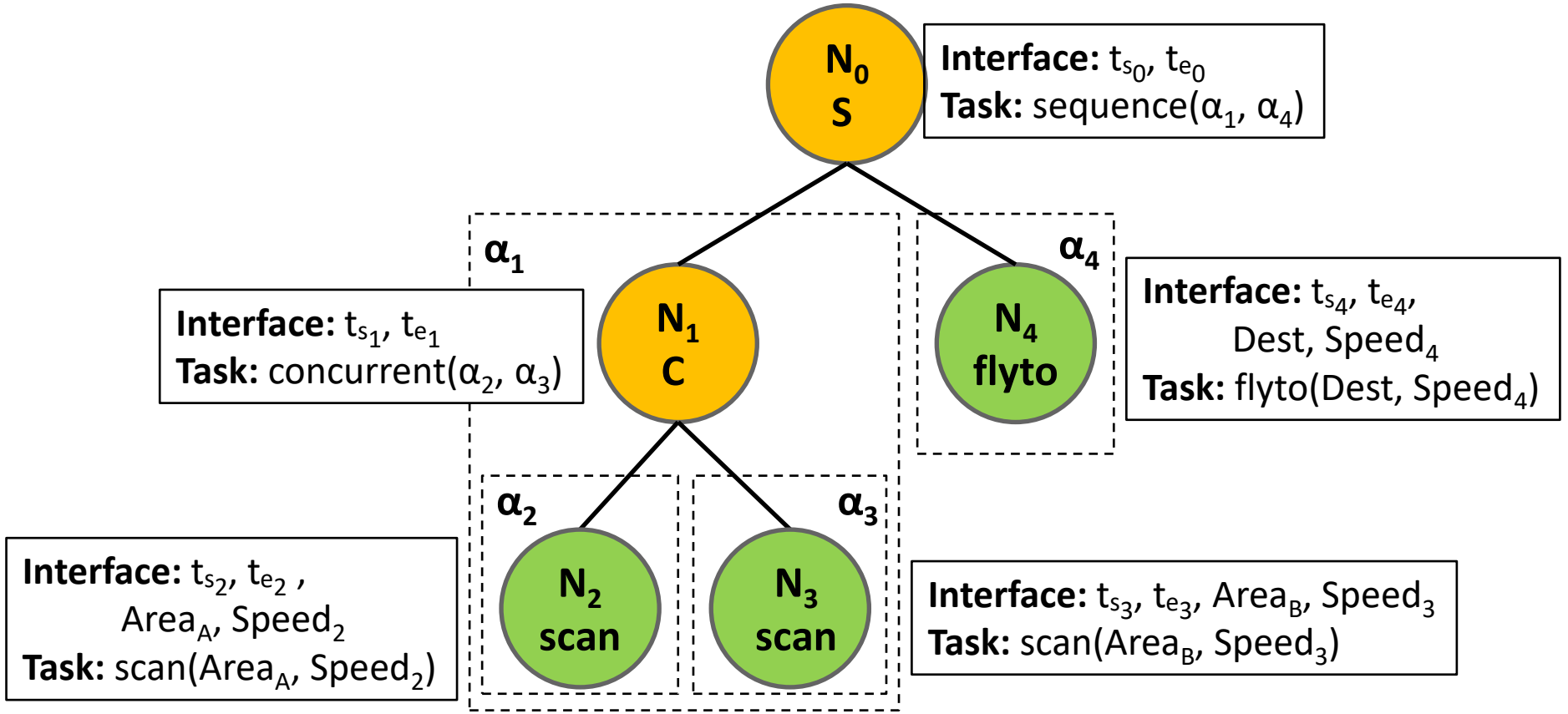


Example Scenario: Search and Relief



Mission: First scan Area$_A$ and Area$_B$, then fly to Dest.

Example TST

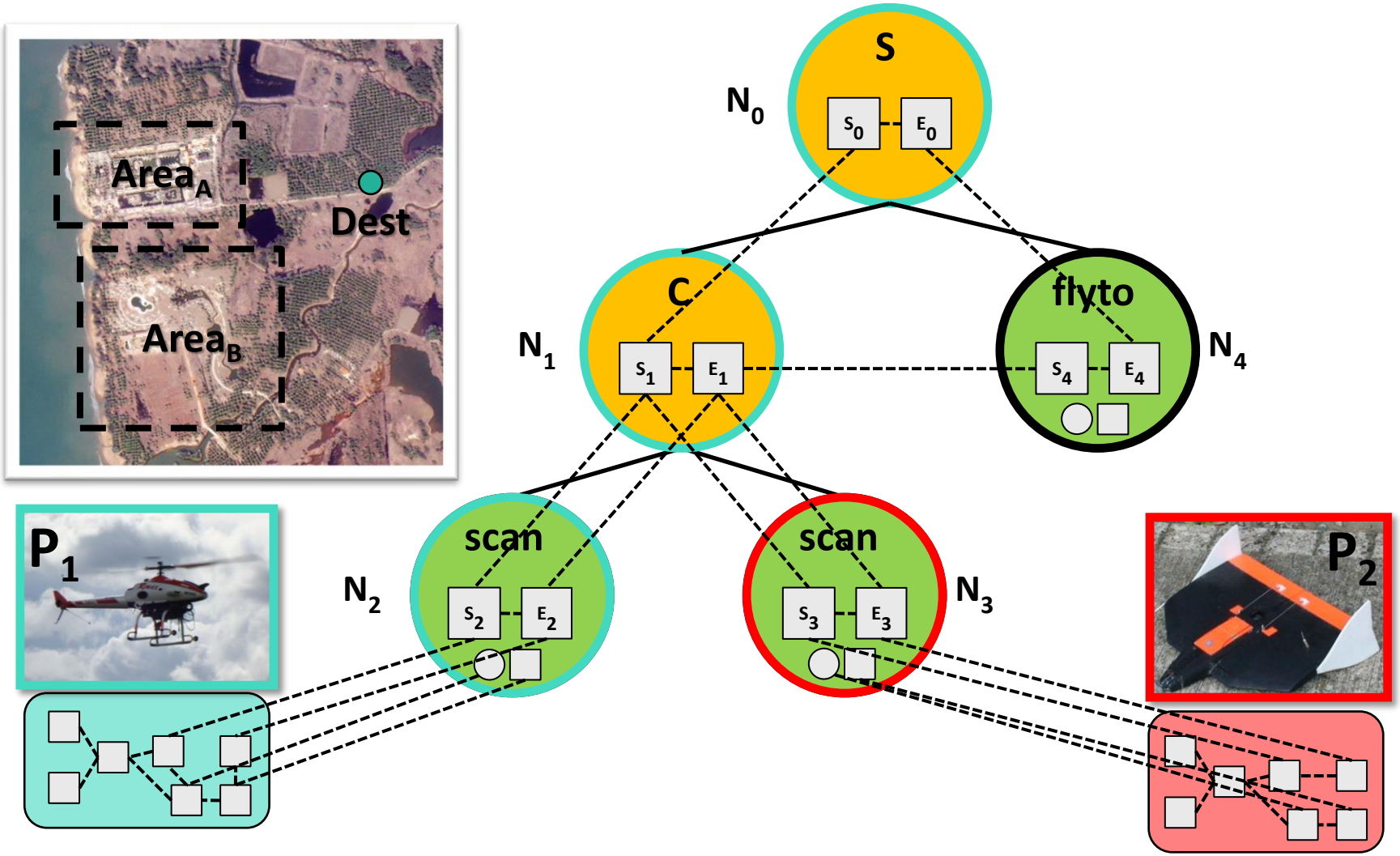


Mission: First scan $Area_A$ and $Area_B$, then fly to Dest.

Delegating TSTs

- What it means to be able to carry out a TST is defined in terms of the Can and Delegate predicates.
- Can(*B*, τ, [t_s, $t_{e, \ldots}$], *cons*) asserts that an agent *B* has the capabilities and resources for achieving a task τ in the interval [t_s, t_e] with the constraints *cons*.
- The semantics of control nodes is platform independent while the semantics of elementary action nodes are platform dependent.
  - Can(*B*, *S*(τ_1, …, τ_n), [t_s, $t_{e, \ldots}$], *cons*) holds iff *B* either can do or delegate each task τ_1, …, τ_n in the sequence so that the constraints are satisfied.

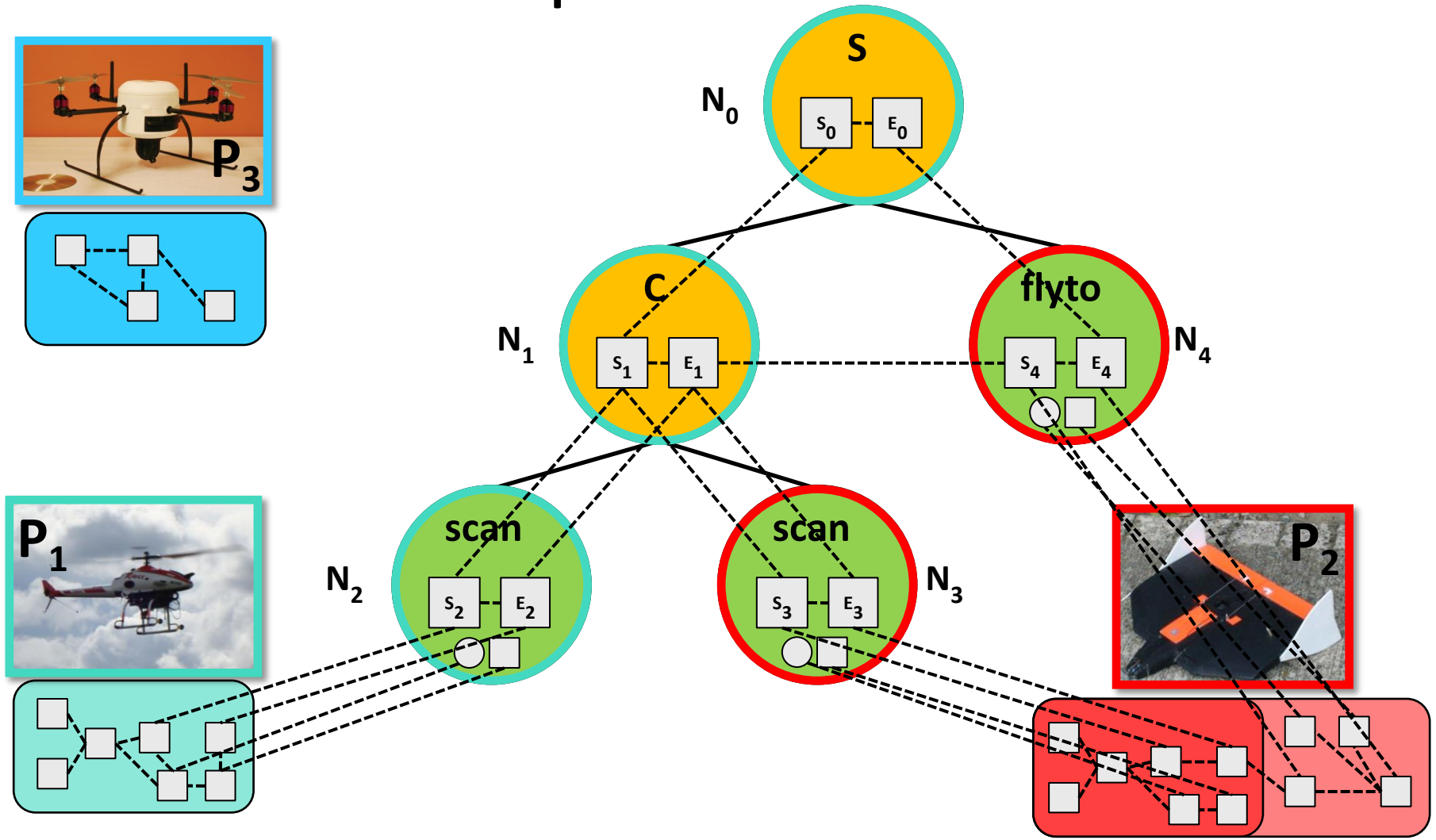
TST Delegation Example



Complex Task Allocation for CUAS

- The goal of the delegation process is to recursively find a set of platforms that can achieve a task specified as a TST.
- For a task to be achievable every node in the TST must be allocated to a platform such that the distributed constraint satisfaction problem corresponding to the semantics of the allocated TST is consistent.
- Hence we need to solve a complex task allocation problem.
- Our approach combines auction-based heuristic search for allocation and distributed constraint satisfaction for consistency checking partial allocations.

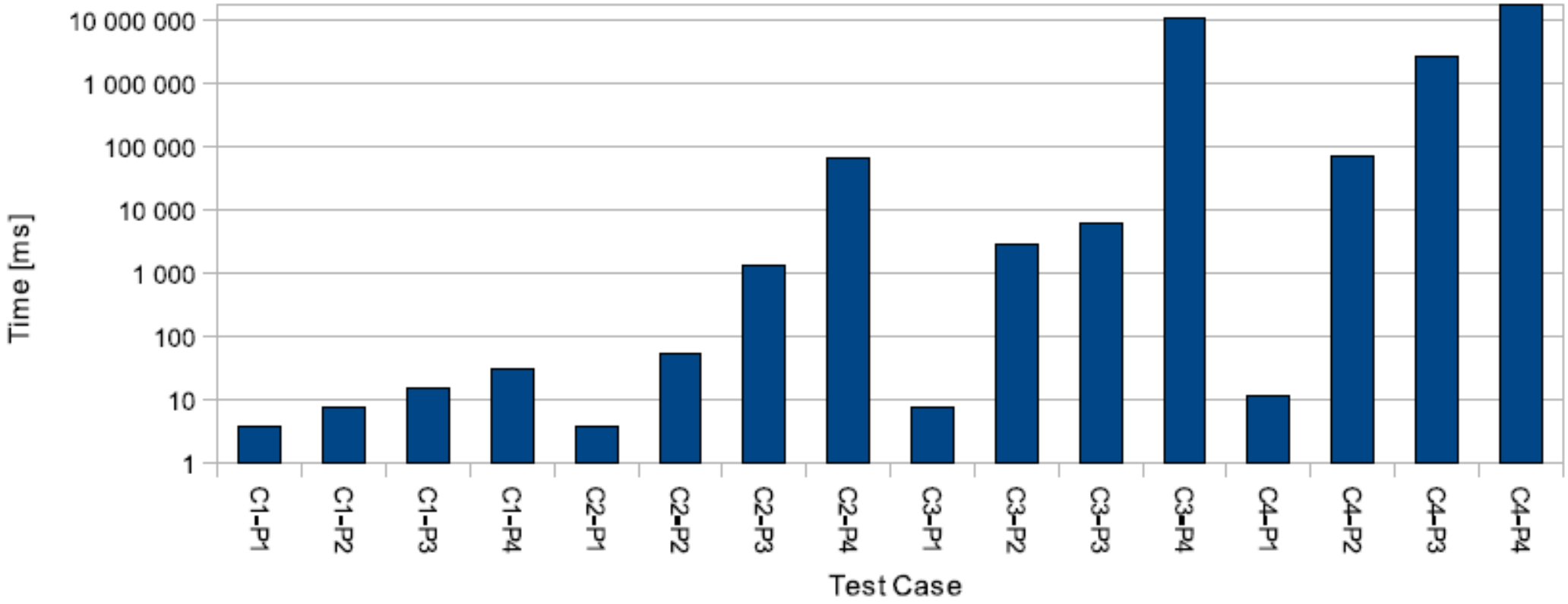
TST Allocation Example



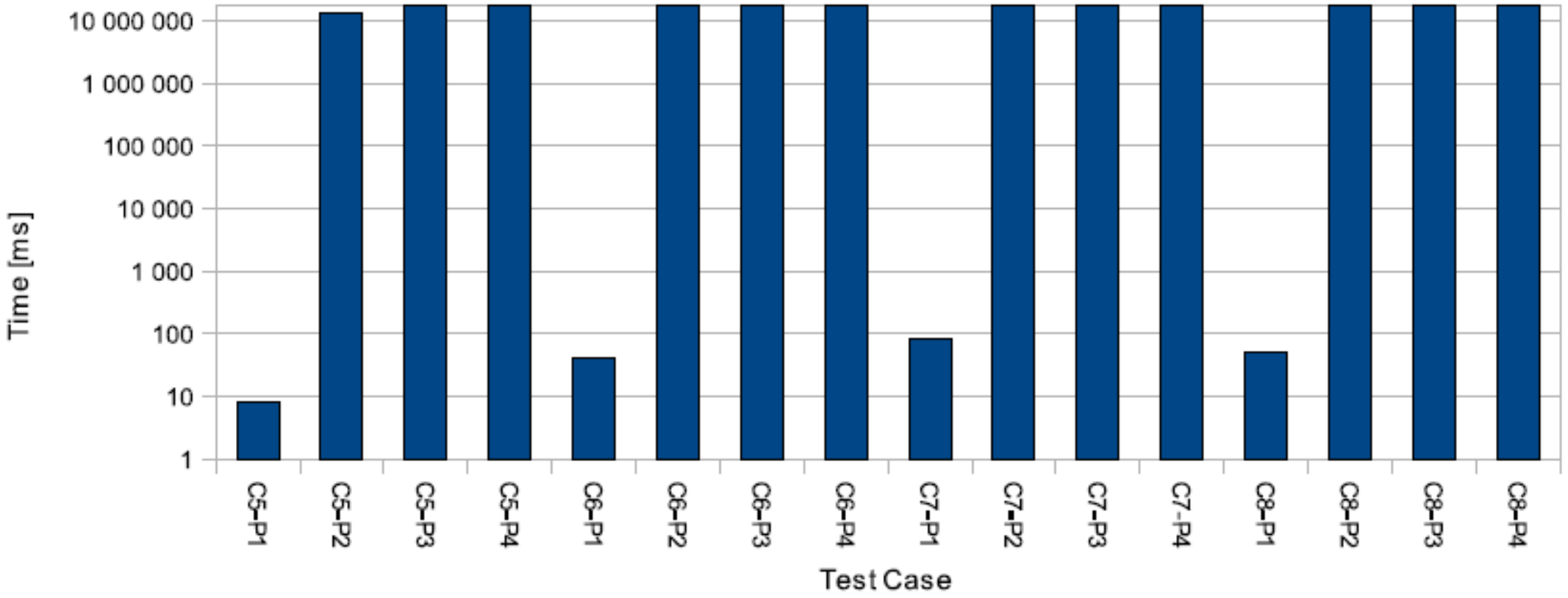
Size of CSP Formulation

	constants	variables	constraints
C1-P1	48	22	129
C1-P2	35	35	138
C1-P3	35	35	147
C1-P4	35	35	156
C2-P1	87	43	417
C2-P2	62	68	435
C2-P3	62	68	453
C2-P4	62	68	471
C3-P1	126	64	867
C3-P2	89	101	894
C3-P3	89	101	921
C3-P4	89	101	948
C4-P1	165	85	1479
C4-P2	116	134	1515
C4-P3	116	134	1551
C4-P4	116	134	1587

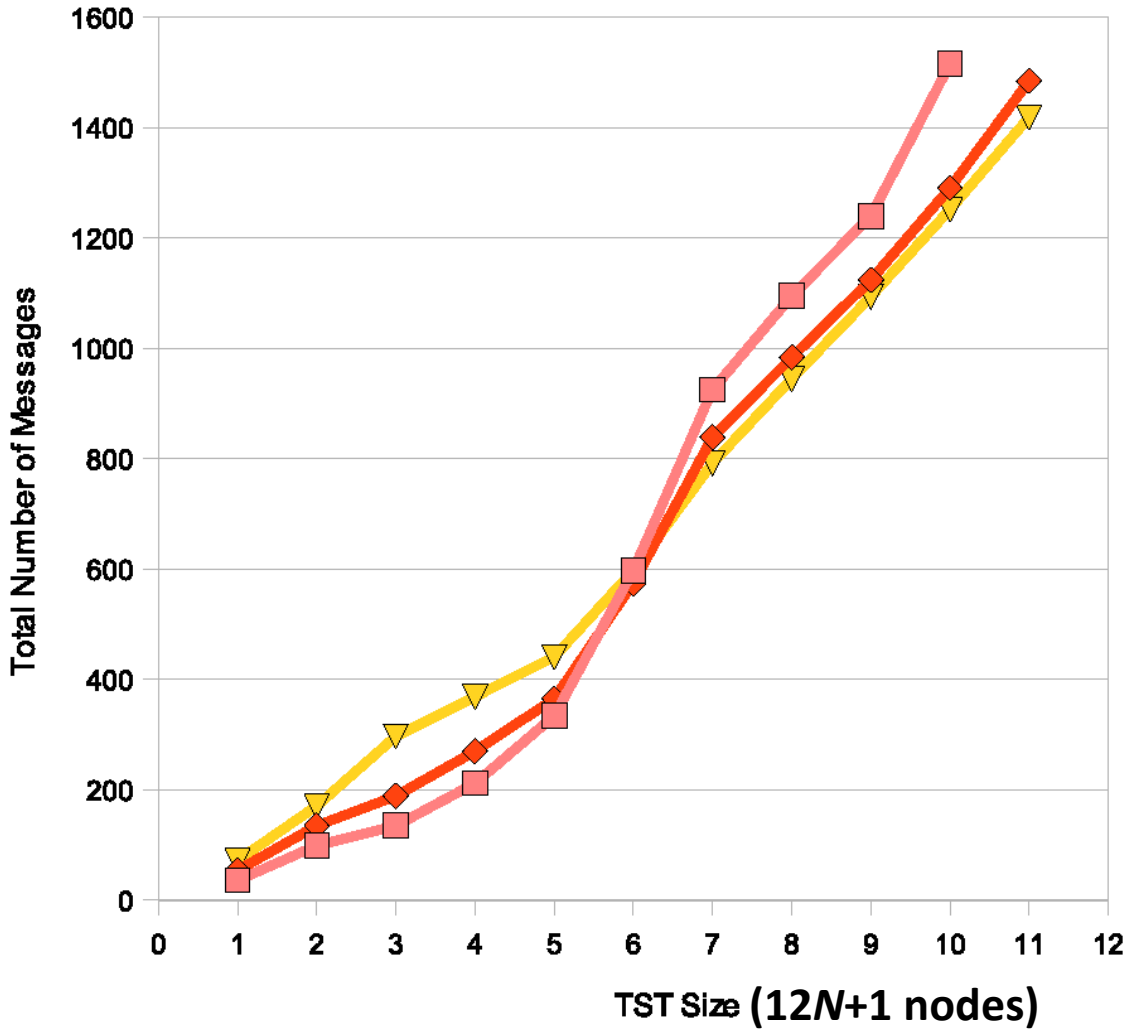
Results Centralized CSP Formulation

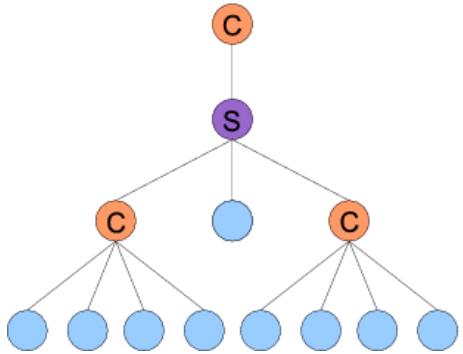


Results Centralized CSP Formulation

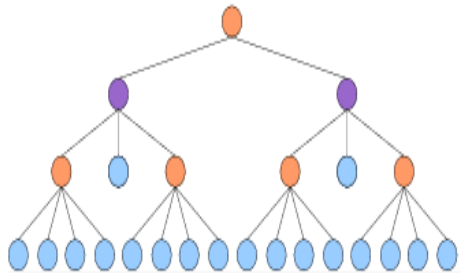


Results Distributed CSP Formulation



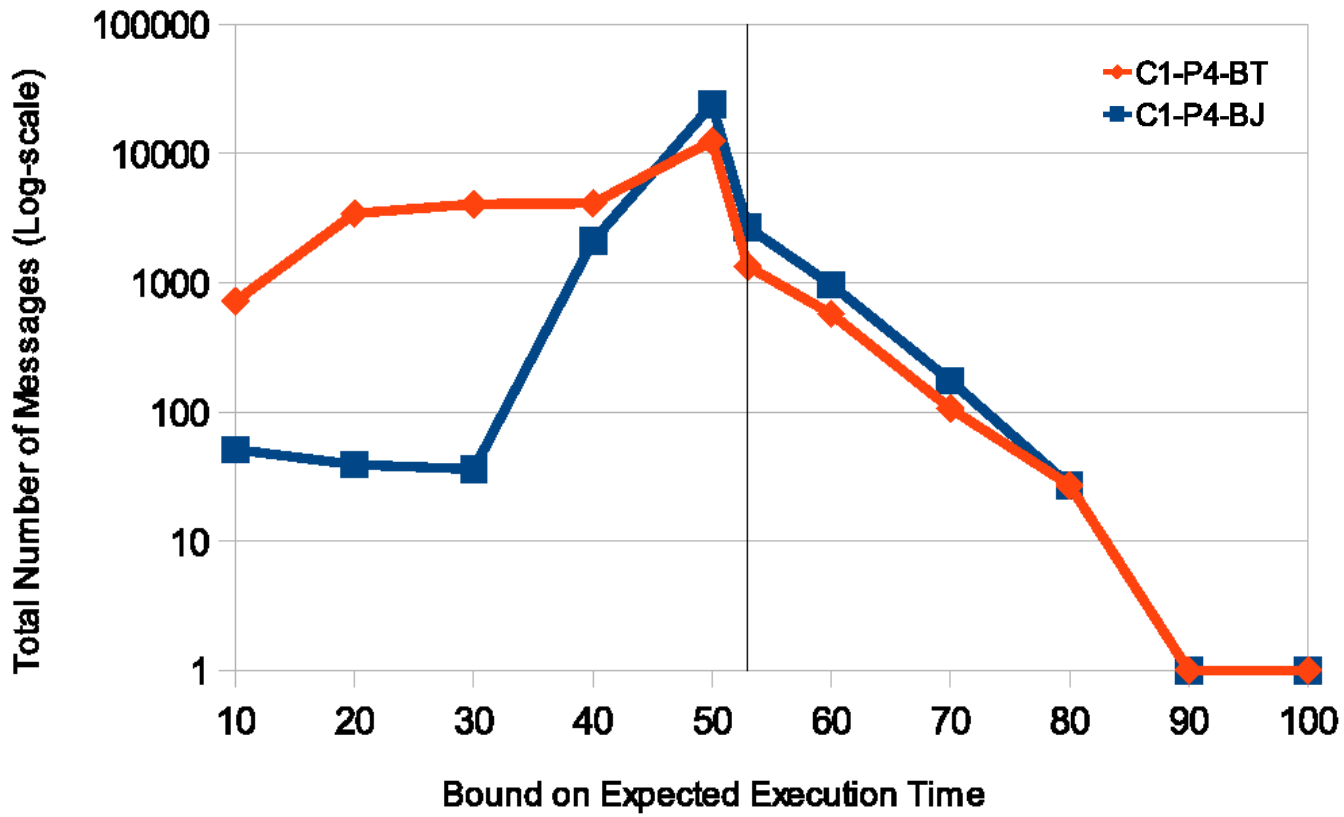


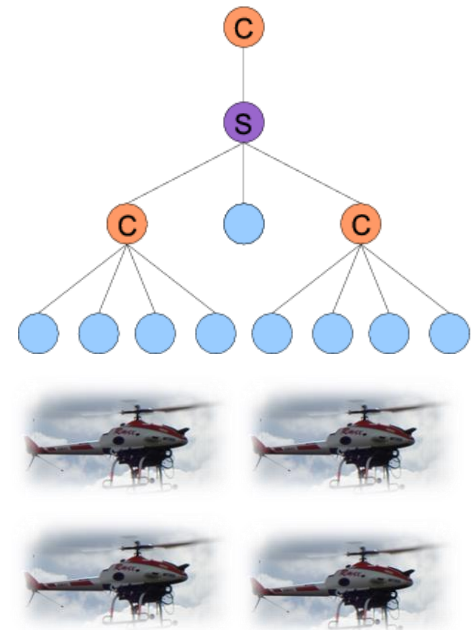
TST size 1 (13 nodes)



TST size 2 (25 nodes)

Results Distributed CSP Formulation



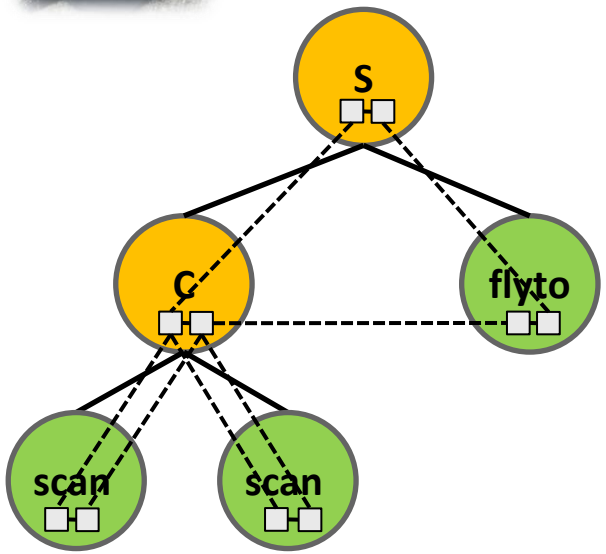


Discussion

- Integrate planning and allocation
- Improve the efficiency of allocating TSTs
  - Study approximating algorithms for allocating TSTs
  - Study heuristics for allocating TSTs
  - Explicitly trade-off quality and efficiency (e.g. anytime algorithms)
  - Study restrictions on TSTs that facilitate more efficient allocation algorithms
  - Develop more efficient distributed constraint solving algorithms for our specific type of problems
  - Further study the interaction between auctions and constraint reasoning to balance guarantees and efficiency
- Consider optimization criteria such as
  - Maximize robustness to deviations due to uncertainty
  - Minimize total execution time and resources usage
  - Minimize resource usage and maximize communication quality

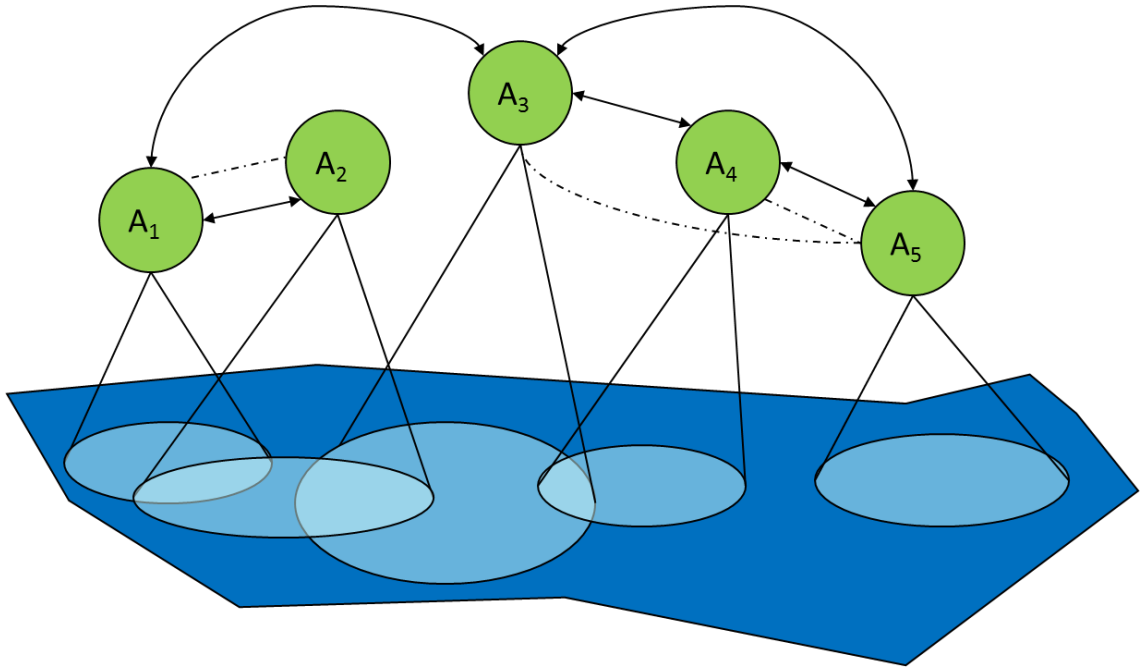
Summary

- Discussed complex task allocation for collaborative unmanned aircraft systems.
- Outlined a delegation-based collaboration framework which uses Task Specification Trees (TSTs) for specifying complex tasks.
  - The consistency of allocations of platforms to TST nodes can be checked using distributed constraint satisfaction techniques.
  - To delegate a TST a complex task allocation problem has to be solved for example using a market-based approach.
- The result is a very rich collaborative robotics framework which opens up for many interesting research questions.



Summary

- A multi-agent system (MAS) can be defined as a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver.
- Communication
- Game Theory
- Social Choice
- Teamwork
- Task Allocation



TDDE13 MAS LE2 HT2023:
Distributed Problem Solving
Task Allocation
UAS Case Study

www.ida.liu.se/~TDDE13

LINKÖPING UNIVERSITY