University of Southampton ORCHID

Theory and Practice of Decentralised Coordination Algorithms exploiting the Generalised Distributive Law

Dr. Francesco M. Delle Fave
University of Southern California
fmdf08r@ecs.soton.ac.uk

USC University of Southern California

ALADDIN autonomous learning agents for decentralised data and information networks

---

This seminar is about coordination problems and algorithms

I. Motivation
II. Case Study of Coordination on Unmanned Aerial Vehicles
III. Partially Ordered Distributed Contraint Optimisation Problems (PO-DCOPs)
   a. Problem and Algorithms Definition
   b. Multi Objective Distributive Constraint Optimisation Problems (MO-DCOPs)
   c. Risk Aware Distributive Constraint Optimisation Problems (RA-DCOPs)
IV. Conclusions and Future Work

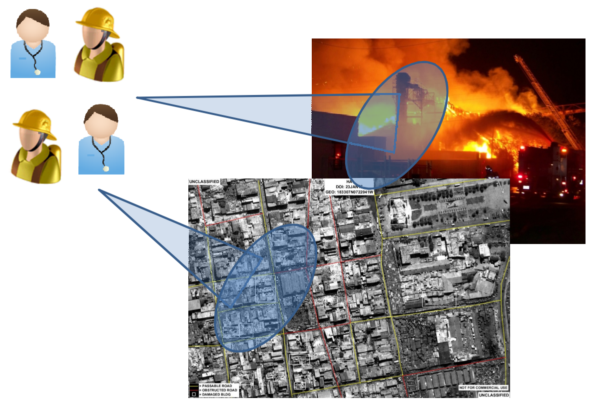
2

This seminar is about coordination problems and algorithms

I. Motivation
II. Case Study of Coordination on Unmanned Aerial Vehicles
III. Partially Ordered Distributed Contraint Optimisation Problems (PO-DCOPs)
   a. Problem and Algorithms Definition
   b. Multi Objective Distributive Constraint Optimisation Problems (MO-DCOPs)
   c. Risk Aware Distributive Constraint Optimisation Problems (RA-DCOPs)
IV. Conclusions and Future Work

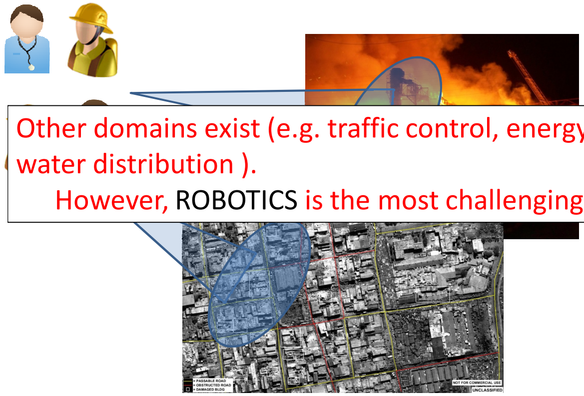
3

We study problems related to robotics for disaster response:



4

We study problems related to robotics for disaster response:



Other domains exist (e.g. traffic control, energy or water distribution).
However, ROBOTICS is the most challenging

5

First Responders (FRs) at the scene of a disaster require accurate Situational Awareness

Situation Awareness[Endsley 2000]: The ability to make sense, and predict what is happening within an environment

6

This information is necessary to prioritise intervention

7

Such information is necessary to prioritise intervention

More information will allow the first responders to discover (and save) more casualties

8

FRs request imagery using Personal Digital Assistants (PDAs)

9

Unmanned Vehicles (UVs) can acquire accurate information

UVs can collect and process more information than humans

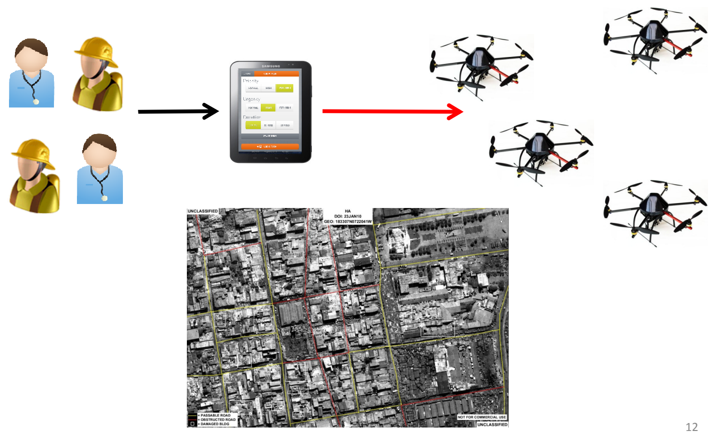
10

Unmanned Vehicles (UVs) can acquire accurate information

UVs can reach places dangerous to humans

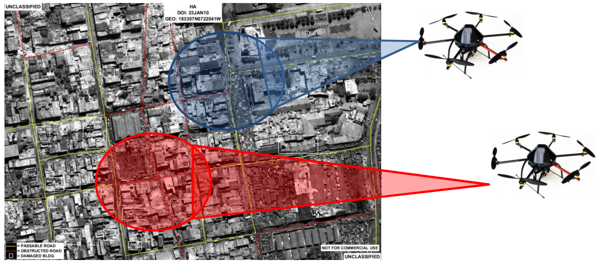
11

Imagery is provided by a team of Unmanned Aerial Vechicles (UAVs)



12

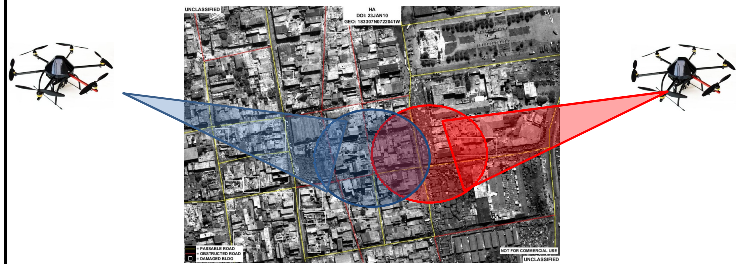
Within this setting, coordination is necessary to improve the performance



By coordinating, the UAVs can explore more areas

13

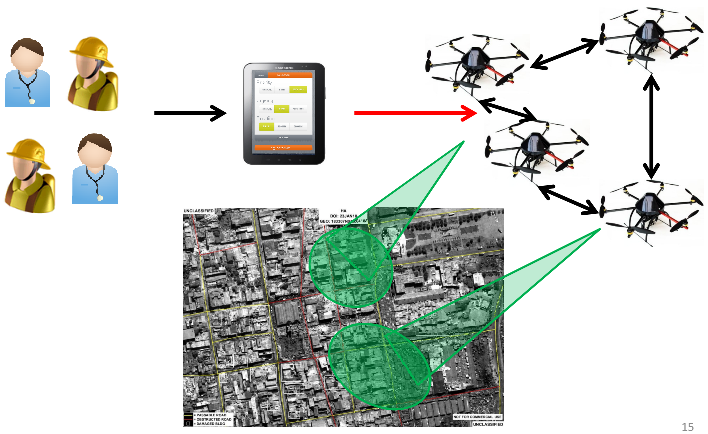
Within this setting, coordination is necessary to improve the performance



By coordinating, the UAVs can also decide to join forces in exploring a vast area

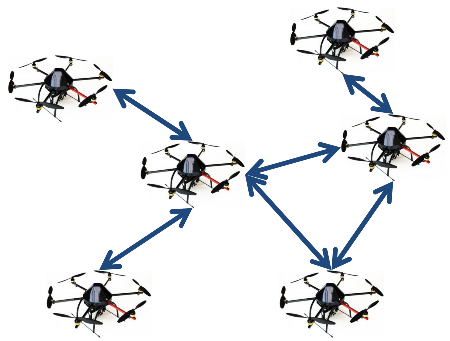
14

UAVs coordinate to attend the imagery tasks



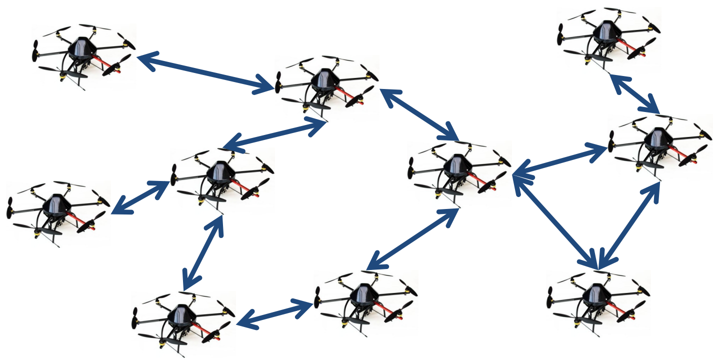
15

This coordination should happen in a decentralised fashion



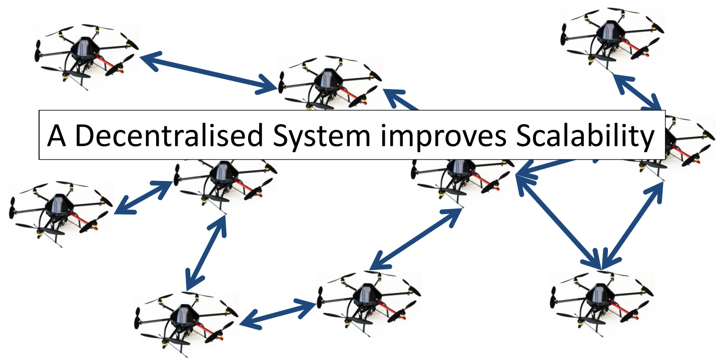
16

This coordination should happen in a decentralised fashion



17

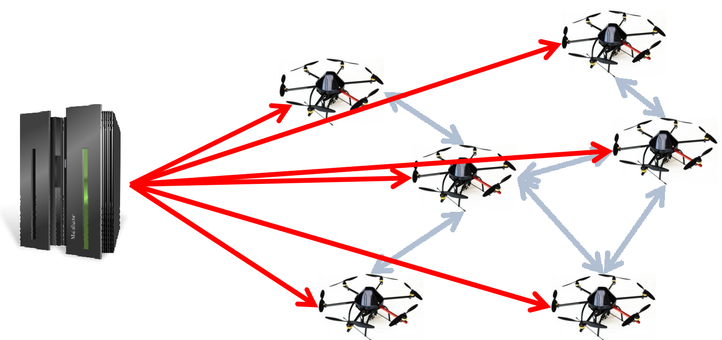
This coordination should happen in a decentralised fashion



A Decentralised System improves Scalability

18

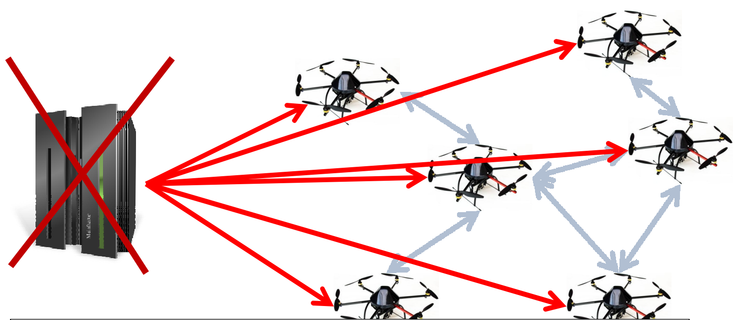
This coordination should happen in a decentralised fashion



A Decentralised System improves Robustness

19

This coordination should happen in a decentralised fashion

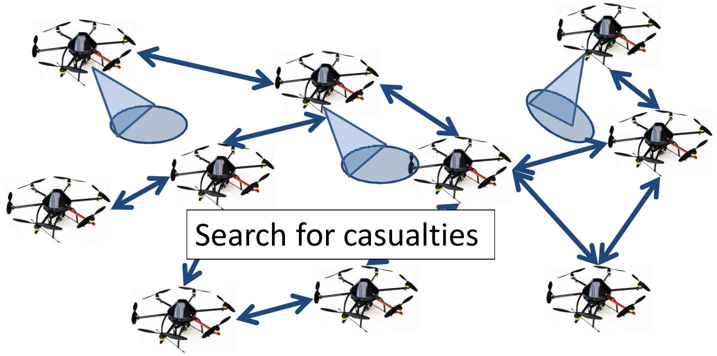


A Decentralised System improves Robustness

No central point of failure

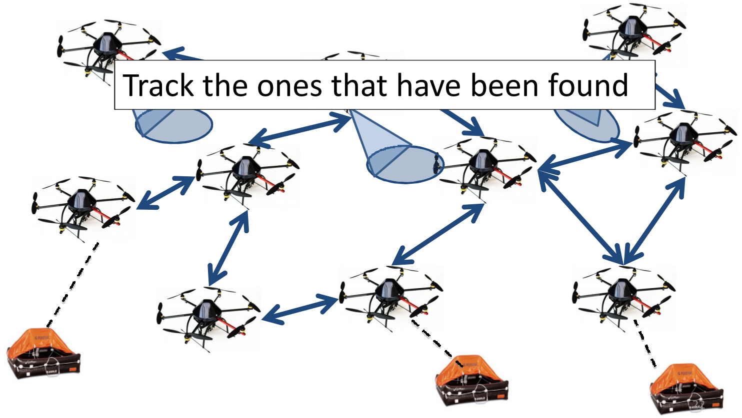
20

This coordination should take multiple objectives into account

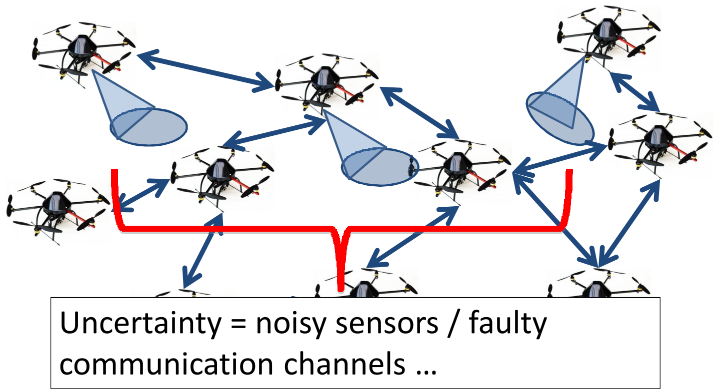


21

This coordination should take multiple objectives into account

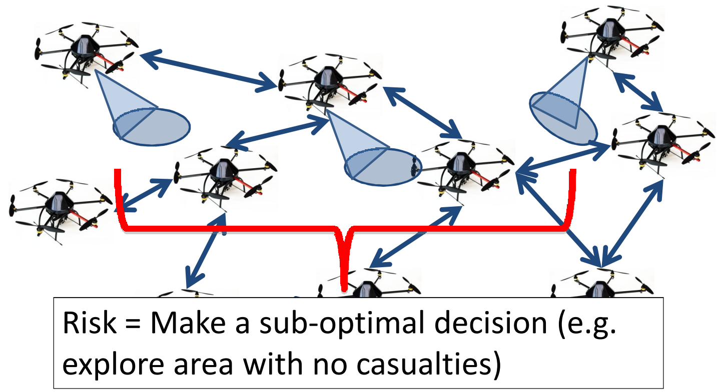


This coordination should take risk and uncertainty into account



23

This coordination should take risk and uncertainty into account

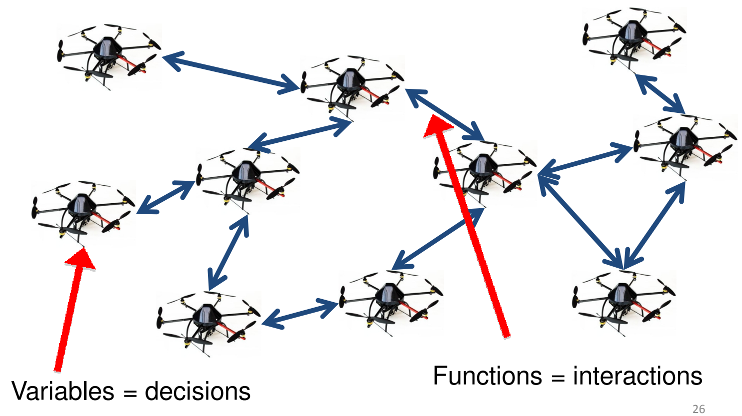


24

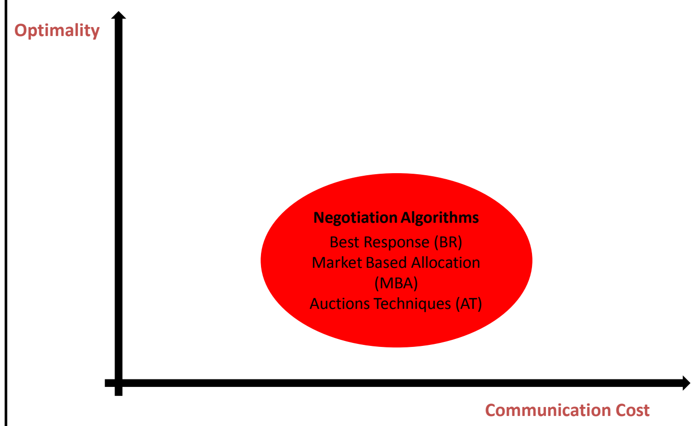
Coordination problems are often represented as Distributed Constraint Optimisation Problems (DCOP)

- A DCOP is a tuple <`A`, `X`, `D`, `U`> s.t.:
  - `A` is a set of agents
  - `X` is a set of variables (typically one per agent)
  - `D` is a set of discrete domains (one per variable)
  - `U` is a set of constraint functions defined over the variables
- To solve a DCOP the agents maximise the sum of the constraints in `U`.

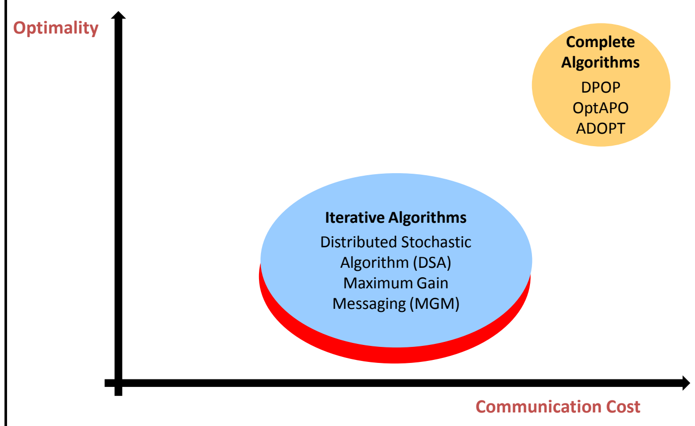
Variables and constraints are useful to encompass the sparse agents interactions



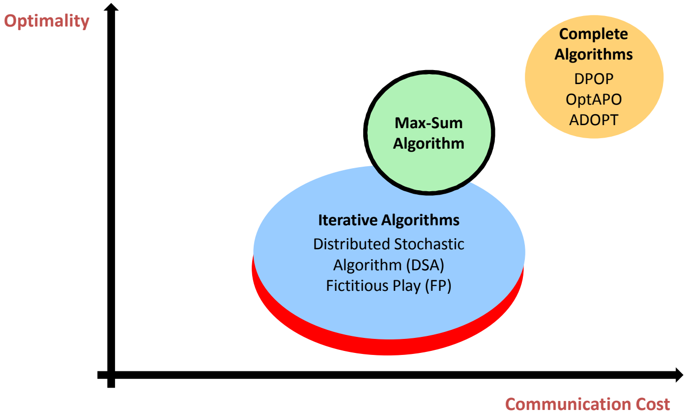
Multiple decentralised coordination algorithms exist in literature:



Many algorithms provide quality guarantees at the cost of a higher computation



Max-Sum appears as a good compromise between feasibility and solution's quality



29

Max-Sum has been applied to multiple Situational Awareness problems

- Monitoring Spatial Phenomena [Stranders, IJCAI'09]
- Patrolling / Pursuit Evasion [Stranders and Delle Fave, AAAI' 10]
- Resource Allocation in Disaster Response Scenarios [Mac Arthur and Ramchurn AAAI' 11]
- Target Search [Delle Fave, AAMAS' 10]
- Dynamic Task Assignment for UAVs [Delle Fave IAAI' 12, ICRA' 12]
- Target Tracking [Rogers, AAMAS' 08]

30

Max-Sum has been applied to multiple Situational Awareness problems

- Monitoring Spatial Phenomena [Stranders, IJCAI'09]
- Patrolling / Pursuit Evasion [Stranders and Delle Fave, AAAI' 10]

However:
All this work has been tested only in simulation

- Target Search [Delle Fave, AAMAS' 10]
- Dynamic Task Assignment for UAVs [Delle Fave ICRA' 12]
- Target Tracking [Rogers, AAMAS' 08]

31

Max-Sum has been applied to multiple Situational Awareness problems

- Monitoring Spatial Phenomena [Stranders, IJCAI'09]
- Patrolling / Pursuit Evasion [Stranders and Delle Fave, AAAI' 10]

Moreover:
There exists no systematic methodology to apply max-sum to all these different problems.

- Dynamic Task Assignment for UAVs [Delle Fave ICRA' 12]
- Target Tracking [Rogers, AAMAS' 08]

32

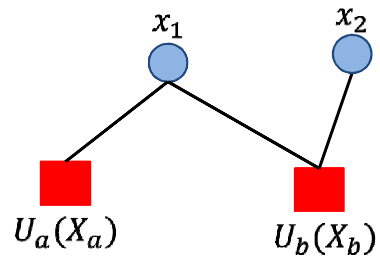
Max-Sum has been applied to multiple Situational Awareness problems

- Monitoring Spatial Phenomena [Stranders, IJCAI'09]
- Patrolling / Pursuit Evasion [Stranders and Delle Fave, AAAI' 10]
- Dynamic Task Assignment for UAVs [Delle Fave ICRA' 12]
- Target Tracking [Rogers, AAMAS' 08]

Moreover:
Max-Sum cannot address multiple objectives nor uncertainty.

33

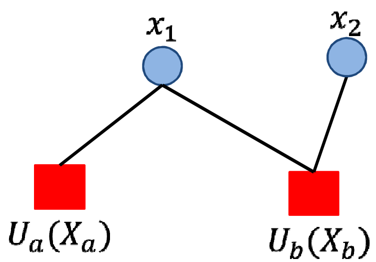
Max-Sum runs over a Factor Graph, a bipartite graph:



A factor graph contains two types of nodes: Variables and Function nodes

34

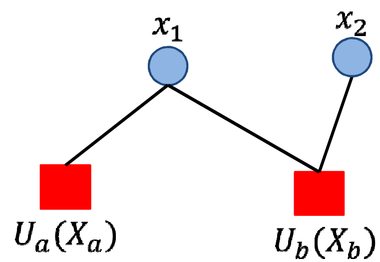
Max-Sum runs over a Factor Graph, a bipartite graph:



Variables represent agents decisions

35

Max-Sum runs over a Factor Graph, a bipartite graph:



Functions represent a DCOP's constraints (i.e. the agent's utility, the assignment of a task…)

36

Max Sum is an approximated message passing algorithm

- Messages flow between function and variable nodes of the factor graph
  - From variable to function

$$Q_{n\to m}(x_n) = \sum_{m'\in M(n)\backslash m} R_{m'\to n}(x_n)$$

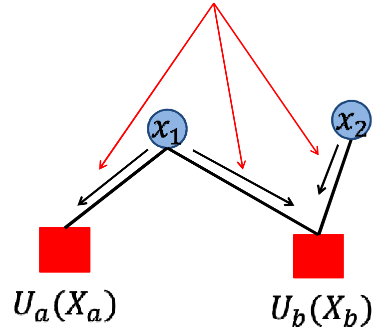
  - From function to variable

$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m\backslash n}\left(U_m(\mathbf{x}_m) + \sum_{n'\in N(m)\backslash n} Q_{n'\to m}(x_{n'})\right)$$

37

2 types of messages are passed between the nodes

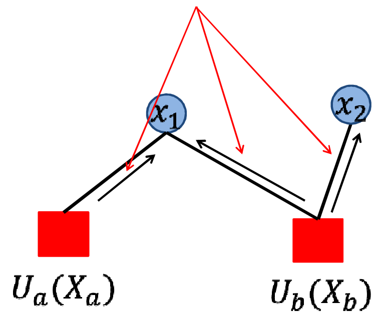
$$Q_{n\to m}(x_n) = \sum_{m'\in M(n)\backslash m} R_{m'\to n}(x_n)$$



38

2 types of messages are passed between the nodes

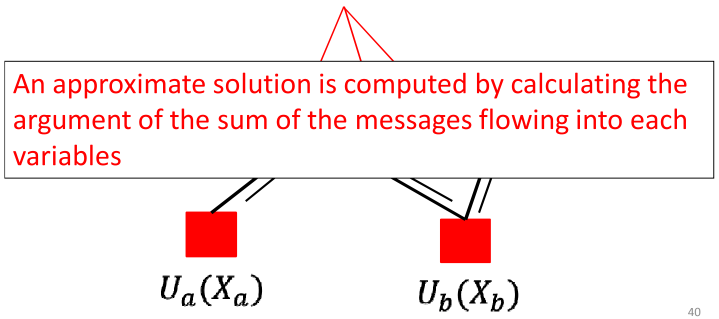
$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m\backslash n}\left(U_m(\mathbf{x}_m) + \sum_{n'\in N(m)\backslash n} Q_{n'\to m}(x_{n'})\right)$$



39

2 types of messages are passed between the nodes

$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m\backslash n}\left(U_m(\mathbf{x}_m) + \sum_{n'\in N(m)\backslash n} Q_{n'\to m}(x_{n'})\right)$$



An approximate solution is computed by calculating the argument of the sum of the messages flowing into each variables

40

Max-Sum belongs to a broader class of algorithms: the Generalised Distributive Law

- GDL algorithms proceed over 3 phases:
  - PHASE 1: transform the constraint graph so that no cycles are present.
    - 2 techniques: junction tree (DFS in DPOP)/ spanning tree

41

Max-Sum belongs to a broader class of algorithms: the Generalised Distributive Law

- GDL algorithms proceed over 3 phases:
  - PHASE 1: transform the constraint graph so that no cycles are present.
    - 2 techniques: junction tree (DFS in DPOP)/ spanning tree

However:
- a spanning tree yields an approximate solution (but bounded)
- a junction tree yields an exponential cost in terms of computation and communication.

42

Max-Sum belongs to a broader class of algorithms: the Generalised Distributive Law

- GDL algorithms proceed over 3 phases:
  - PHASE 1: transform the constraint graph so that no cycles are present.
    - 2 techniques: junction tree (DFS in DPOP)/ spanning tree
  - PHASE 2: run the “Max-Sum” message passing algorithms (util propagation in DPOP).
    - Solves the problem optimally because it is acyclic

43

Max-Sum belongs to a broader class of algorithms: the Generalised Distributive Law

- GDL algorithms proceed over 3 phases:
  - PHASE 1: transform the constraint graph so that no cycles are present.
    - 2 techniques: junction tree (DFS in DPOP)/ spanning tree
  - PHASE 2: run the “Max-Sum” message passing algorithms (util propagation in DPOP).
    - Solves the problem optimally because it is acyclic
  - PHASE 3: use Value Propagation to retrieve a consistent solution.

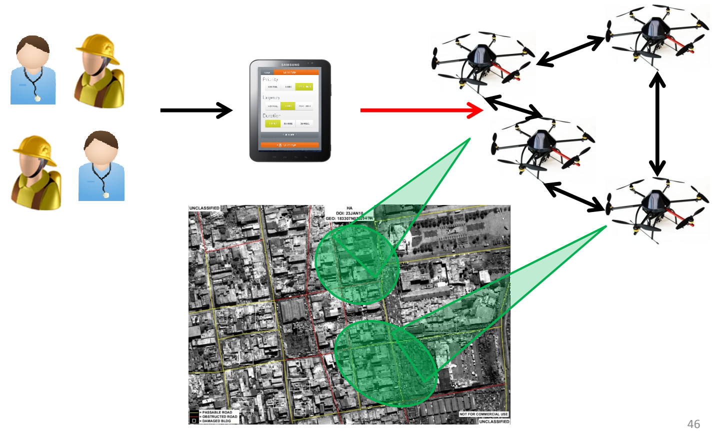
44

This seminar is about coordination problems and algorithms

I. Motivation
II. Case Study of Coordination on Unmanned Aerial Vehicles
III. Partially Ordered Distributed Contraint Optimisation Problems (PO-DCOPs)
   a. Problem and Algorithms Definition
   b. Multi Objective Distributive Constraint Optimisation Problems (MO-DCOPs)
   c. Risk Aware Distributive Constraint Optimisation Problems (RA-DCOPs)
IV. Conclusions and Future Work

45

We use GDL algorithms to build a system to present FRs with accurate SA using UAVs



46

To deploy max-sum we propose a 5 steps methodology

STEP 1 / 2: WHAT are the nodes?

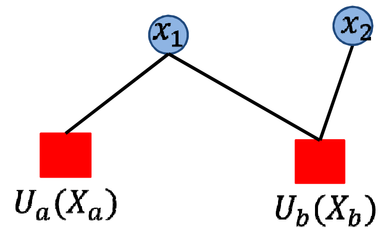
STEP 3: WHO controls the nodes?

STEP 4: WHEN are messages computed?

STEP 5: HOW do nodes know their neighbours?

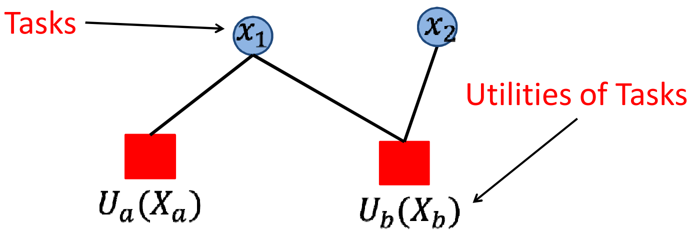
47

STEP 1 / 2: WHAT are the nodes?



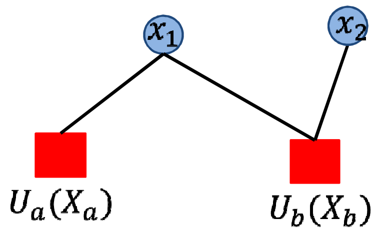
48

STEP 1 / 2: WHAT are the nodes?



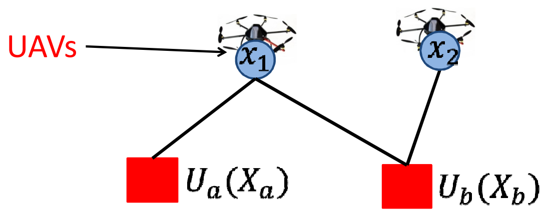
49

STEP 3: WHO controls the nodes?



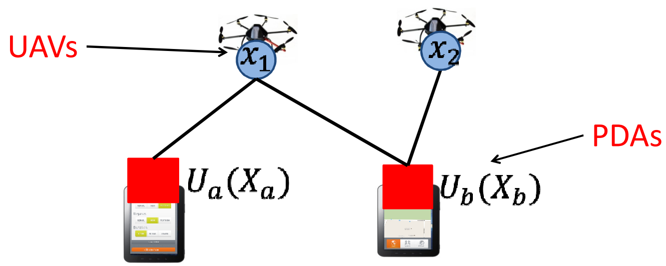
50

STEP 3: WHO controls the nodes?



51

STEP 3: WHO controls the nodes?

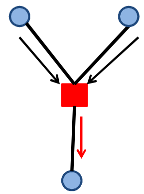


52

STEP 4: WHEN are messages computed?

STEP 4: WHEN are messages computed?

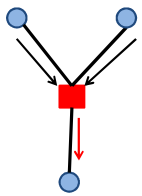
Synchronised



WAIT

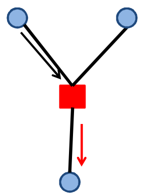
STEP 4: WHEN are messages computed?

Synchronised



WAIT

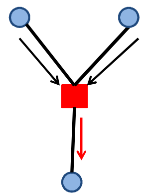
Reactive



When receive msg

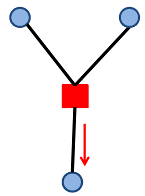
STEP 4: WHEN are messages computed?

Synchronised



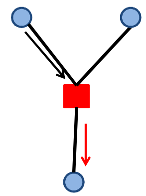
WAIT

Periodical



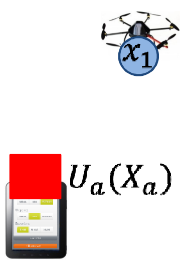
Every T seconds

Reactive

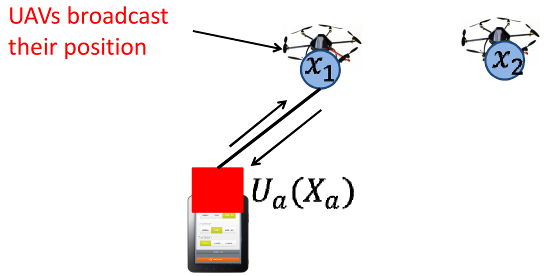


When receive msg

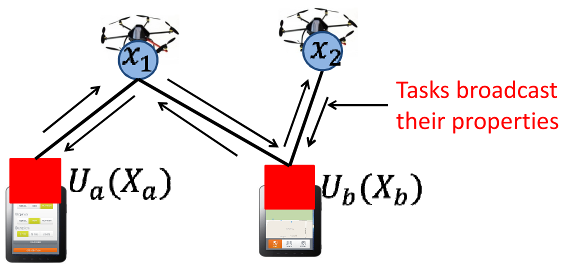
STEP 5: HOW do nodes know their neighbours?



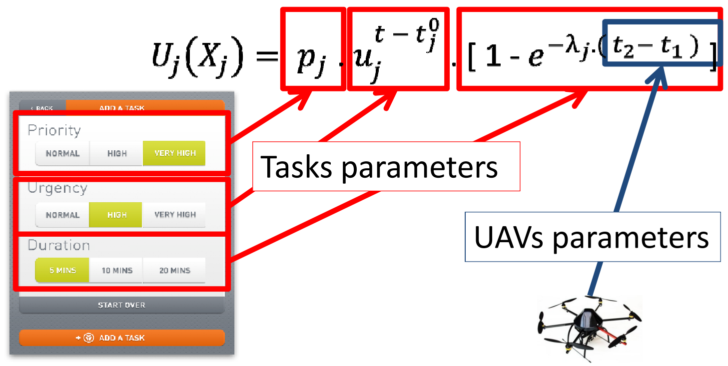
STEP 5: HOW do nodes know their neighbours?



STEP 5: HOW do nodes know their neighbours?

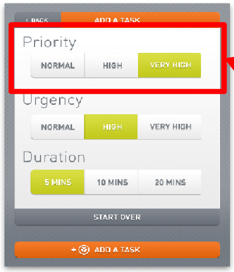


The task utility weights the problem's constraints to improve the allocation



The Priority represents a task's importance

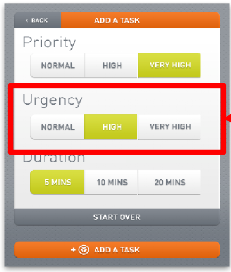
$$U_j(X_j) = \boxed{p_j} \,.\, u_j^{\,t - t_j^0} \,.\, [\, 1 - e^{-\lambda_j.(\,t_2 - t_1\,)}\,]$$



Task Priority

The Urgency prevents the tasks starvation

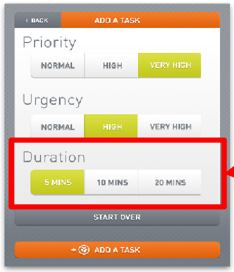
$$U_j(X_j) = p_j \,.\, \boxed{u_j^{\,t - t_j^0}} \,.\, [\, 1 - e^{-\lambda_j.(\,t_2 - t_1\,)}\,]$$



Task Urgency

The duration models the probability that the UAVs will complete the task

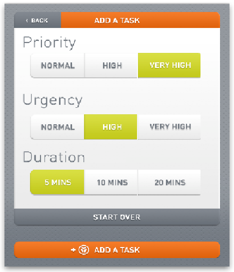
$$U_j(X_j) = p_j \,.\, u_j^{\,t - t_j^0} \,.\, \boxed{[\, 1 - e^{-\lambda_j.(\,t_2 - t_1\,)}\,]}$$



Task Termination:
Poisson Process
(details in the paper)

This probability allows to trade off between the UAVs that can attend the task

$$U_j(X_j) = p_j \,.\, u_j^{\,t - t_j^0} \,.\, [\, 1 - e^{-\lambda_j.\boxed{(\,t_2 - t_1\,)}}\,]$$



Time span between the UAV with the highest battery life and the UAV closest to the task

The allocation varies depending on the UAVs capabilities

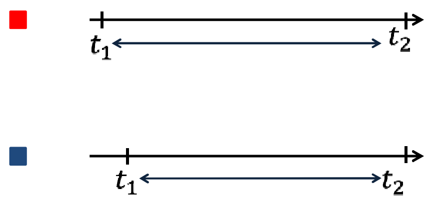
- Task 1 (LP, LU, LD)
- Task 2 (HP, LU, LD)
- UAV 1 (HB)
- UAV 2 (HB)

The UAVs can attend both the tasks

65

Why is this decision made?

$$U_j(X_j) = p_j \,.\, u_j^{\,t - t_j^0} \,.\, [\,1 - e^{-\lambda_j.(t_2 - t_1)}\,]$$

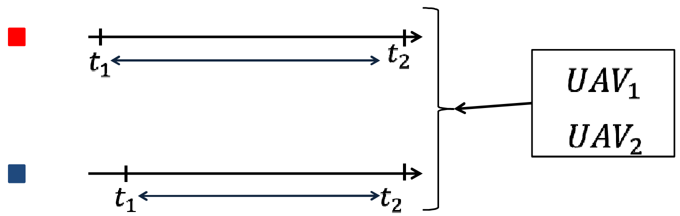


HB means that t_2 is very high for both the UAVs

66

Why is this decision made?

$$U_j(X_j) = p_j \,.\, u_j^{\,t - t_j^0} \,.\, [\,1 - e^{-\lambda_j.(t_2 - t_1)}\,]$$



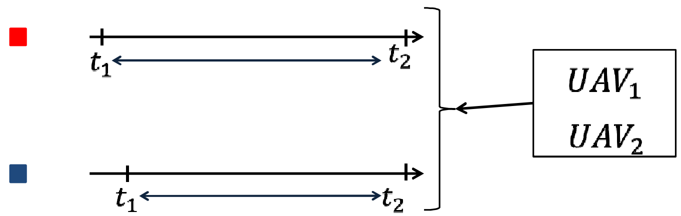
HB means that t_2 is very high for both the UAVs

Each UAV can complete both the tasks

67

Why is this decision made?

$$U_j(X_j) = p_j \,.\, u_j^{\,t - t_j^0} \,.\, [\,1 - e^{-\lambda_j.(t_2 - t_1)}\,]$$



Max-sum allocate each UAV to one different task so as to maximise the utility

68

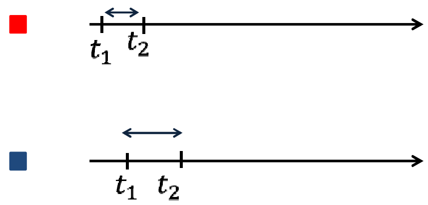
The allocation varies depending on the UAVs capabilities

- Task 1 (LP, LU, LD)
- Task 2 (HP, LU, LD)
- UAV 1 (LB)
- UAV 2 (LB)

The UAVs may not be able to attend any task -> they join their forces

Why is this decision made?

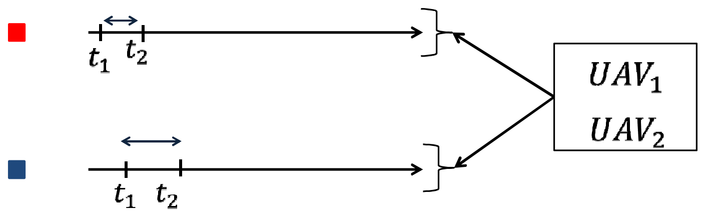
$$U_j(X_j) = p_j \, . \, u_j^{\,t - t_j^0} \, . \, [\, 1 - e^{-\lambda_j . (t_2 - t_1)} \,]$$



LB means that t_2 is very low for both the UAVs

Why is this decision made?

$$U_j(X_j) = p_j \, . \, u_j^{\,t - t_j^0} \, . \, [\, 1 - e^{-\lambda_j . (t_2 - t_1)} \,]$$

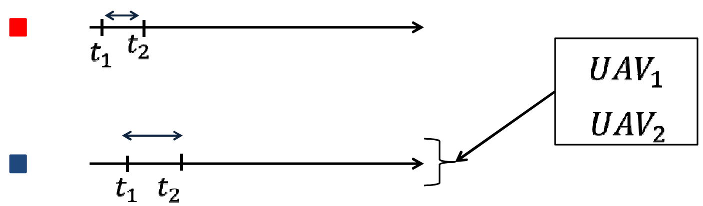


LB means that t_2 is very low for both the UAVs

The UAVs might not be able to complete one single task even working together

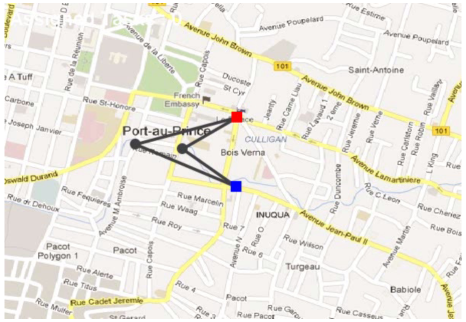
Why is this decision made?

$$U_j(X_j) = p_j \, . \, u_j^{\,t - t_j^0} \, . \, [\, 1 - e^{-\lambda_j . (t_2 - t_1)} \,]$$



Max-sum allocate both the UAVs to the HP task so as to maximise the utility

The allocation varies depending on the UAVs capabilities

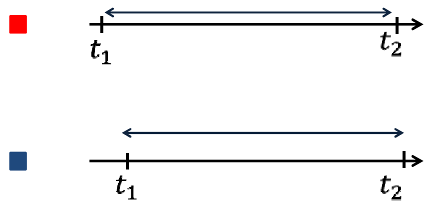


- Task 1 (LP, LU, LD)
- Task 2 (HP, LU, LD)
- UAV 1 (HB)
- UAV 2 (LB)

The UAVs may be able to attend both the tasks -> they revise their decisions

73

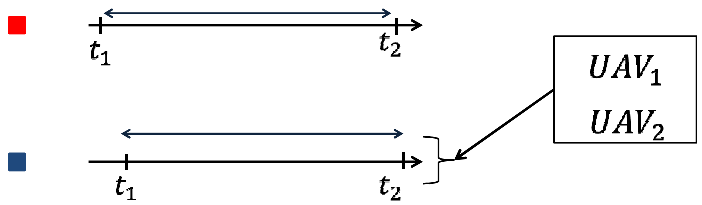
Why is this decision made?



HB for 1 UAV means that t_2 is very high for only 1 UAV

74

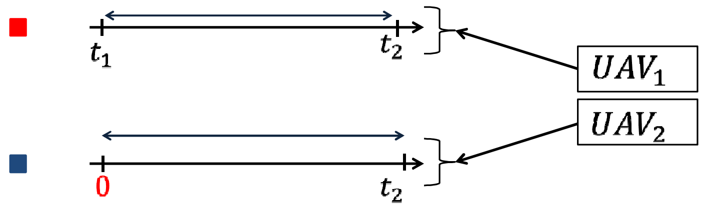
Why is this decision made?



HB for 1 UAV means that t_2 is very high for only 1 UAV → The UAVs will be able to complete the HP task if they work together

75

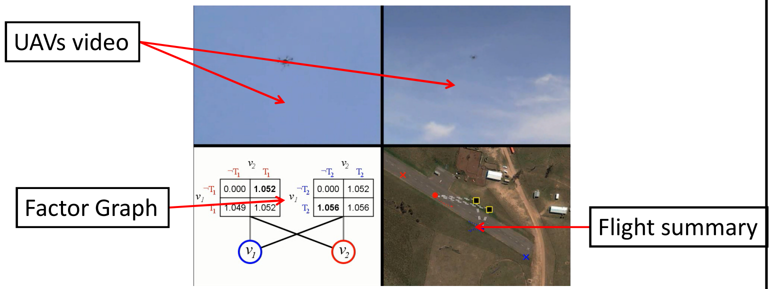
Why is this decision made?



When they both reach the HP task, t_1 is the same for both the UAVs → One UAV hands over the HP task and goes to complete the LP one

76

VIDEOs legend



This seminar is about coordination problems and algorithms

I. Motivation
II. Case Study of Coordination on Unmanned Aerial Vehicles
III. Partially Ordered Distributed Contraint Optimisation Problems (PO-DCOPs)
   a. Problem and Algorithms Definition
   b. Multi Objective Distributive Constraint Optimisation Problems (MO-DCOPs)
   c. Risk Aware Distributive Constraint Optimisation Problems (RA-DCOPs)
IV. Conclusions and Future Work

DCOPs are not sufficient to model complex interactions

Problem: the agents decisions cannot be represented considering a single scalar function

Example: mission objective + computation, communication, and battery life

Consequence: Standard DCOP do not encompass the complexity of the real world

81

There exist many type of complex interactions

Examples:

- Multiple conflicting objectives (search, track, avoid dangerous areas)
- Uncertainty and risk (risk to search dangerous areas or areas with no casualties because information is uncertain)

82

Such complex interactions can be modeled using partially ordered functions

Example:

- Bi-objective functions: (1,2), (2,1), (1,1)
- Mean and variance: (3,1.3), (5,2.5), (1,5.4)

83

Such complex interactions can be modeled using partially ordered functions

Which assignment is the best?

Example:

- Bi-objective functions: (1,2), (2,1), (1,1)
- Mean and variance: (3,1.3), (5,2.5), (1,5.4)

84

DCOPs + PO functions = new CLASS of problems: PO-DCOPs

- A PO-DCOP is a tuple <A, X, D, U> s.t.:
  - A is a set of agents
  - X is a set of variables (typically one per agent)
  - D is a set of discrete domains (one per variable)
  - U is a set of partially-ordered constraint functions defined over the variables

85

DCOPs + PO functions = new CLASS of problems: PO-DCOPs

- A PO-DCOP is a tuple <A, X, D, U> s.t.:
  - A is a set of agents
  - X is a set of variables (typically one per agent)
  - D is a set of discrete domains (one per variable)
  - U is a set of partially-ordered constraint functions defined over the variables

What is the solution of a PO-DCOP?

86

The solutions of a PO-DCOP are similar to those of a DCOP

- The solutions are all the assignments of the variables in X that optimise ($\oplus$ ≈ counting operator) the aggregation of the partially ordered functions in U ($\otimes$ ≈ aggregation operator)

87

The solutions of a PO-DCOP are similar to those of a DCOP

- The solutions are all the assignments of the variables in X that optimise ($\oplus$ ≈ counting operator) the aggregation of the partially ordered functions in U ($\otimes$ ≈ aggregation operator)

88

A PO-DCOP has multiple non-dominated solutions

Example:
- Bi-objective functions: (1,2), (2,1), (1,1)
- Mean and variance: (3,1.3), (5,2.5), (1,5.4)

The solutions of a PO-DCOP are similar to those of a DCOP

- The solutions are all the assignments of the variables in $\mathbb{X}$ that optimise ($\oplus$ ≈ counting operator) the aggregation of the partially ordered functions in $\mathbb{U}$ ($\otimes$ ≈ aggregation operator)

Can we use GDL algorithms to solve them?

PO-DCOPs structure allows to use GDL algorithms

- The abstract GDL framework uses two operators:
  - $\oplus$ for combining sets of values ("sum")
  - $\otimes$ for selecting values from a set ("max")
- Exploits the fact that $\oplus$ distributes over $\otimes$ to minimise computation

By changing $\oplus$ and $\otimes$ in the message passing algorithms we can instantiate new algorithms

The GDL solves PO-DCOPs using local message passing

- Messages flow between function and variable nodes of the factor graph
  - From **variable** to **function**

$$Q_{n\rightarrow m}(x_n) = \boxed{\sum}_{m'\in M(n)\setminus m} R_{m'\rightarrow n}(x_n)$$

$\otimes$

  - From **function** to **variable**

$$R_{m\rightarrow n}(x_n) = \boxed{\max}_{\mathbf{x}_m\setminus n}\left(f_m(\mathbf{x}_m) \boxed{+} \boxed{\sum}_{n'\in N(m)\setminus n} Q_{n'\rightarrow m}(x_{n'})\right)$$

$\oplus$

PO-DCOPs structure allows to use GDL algorithms

- The abstract GDL framework uses two operators:
  - $\oplus$ for combining sets of values ("sum")
  - $\otimes$ for selecting values from a set ("max")
- Exploits the fact that $\oplus$ distributes over $\otimes$ to minimise computation

Main Theorem: if the constraint graph representing a PO-DCOP is acyclic then GDL algorithms produce optimal solutions

93

PO-DCOPs structure allows to use GDL algorithms

- The abstract GDL framework uses two operators:
  - $\oplus$ for combining sets of values ("sum")
  - $\otimes$ for selecting values from a set ("max")

We can instantiate ALL GDL algorithms!

representing a PO-DCOP is acyclic then GDL algorithms produce optimal solutions

94

This seminar is about coordination problems and algorithms

I. Motivation
II. Case Study of Coordination on Unmanned Aerial Vehicles
III. Partially Ordered Distributed Contraint Optimisation Problems (PO-DCOPs)
   a. Problem and Algorithms Definition
   b. Multi Objective Distributive Constraint Optimisation Problems (MO-DCOPs)
   c. Risk Aware Distributive Constraint Optimisation Problems (RA-DCOPs)
IV. Conclusions and Future Work

95

We instantiate PO-DCOP to solve **multi-objective** problems

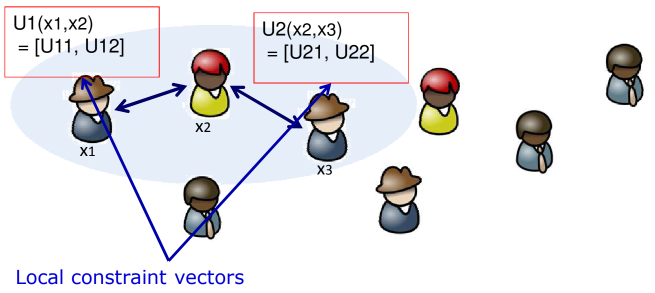
Problem: multiple (conflicting) objectives exist

Example: In search and rescue, agents need to search, track, and maintain communications

96

We define MO-DCOPs: constraint functions become constraint vectors

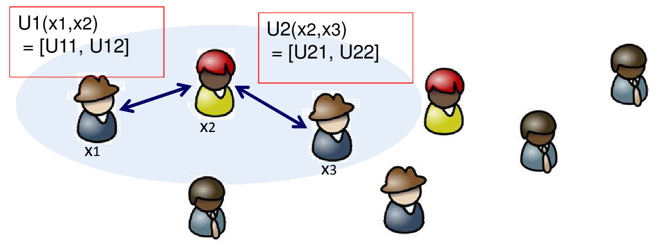
MO-DCOPs:



97

We define MO-DCOPs: the global function become a global vector

MO-DCOPs:

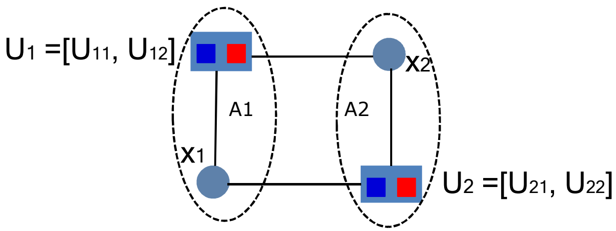


Global constraint vector

$$U(x) = \text{argmax} \sum U_{i(x_i)}$$
$$= \text{argmax} \sum [U_{i1}, U_{i2}]$$

98

MO-DCOPs can be directly encoded into a factor graph



99

MO-DCOPS have multiple optimal solutions which are non-comparable

x_1	x_2	$U_1 = [U_{11}, U_{12}]$	$U_2 = [U_{21}, U_{22}]$	$U = U_1 + U_2$
0	0	(1,2)	(2,0)	(3,2)
0	1	(2,1)	(0,2)	(2,3)
1	0	(0,0)	(4,3)	(4,3) dominates
1	1	(1,1)	(2,3)	(3,4)

100

MO-DCOPS have multiple optimal solutions which are non-comparable

x_1	x_2	$U_1 = [U_{11}, U_{12}]$	$U_2 = [U_{21}, U_{22}]$	$U = U_1 + U_2$
0	0	(1,2)	(2,0)	(3,2)
0	1	(2,1)	(0,2)	(2,3)
1	0	(0,0)	(4,3)	(4,3)
1	1	(1,1)	(2,3)	(3,4)

dominates

MO-DCOPS have multiple optimal solutions which are non-comparable

x_1	x_2	$U_1 = [U_{11}, U_{12}]$	$U_2 = [U_{21}, U_{22}]$	$U = U_1 + U_2$
0	0	(1,2)	(2,0)	(3,2)
0	1	(2,1)	(0,2)	(2,3)
1	0	(0,0)	(4,3)	(4,3)
1	1	(1,1)	(2,3)	(3,4)

Non-dominated vectors

MO-DCOPS have multiple optimal solutions which are non-comparable

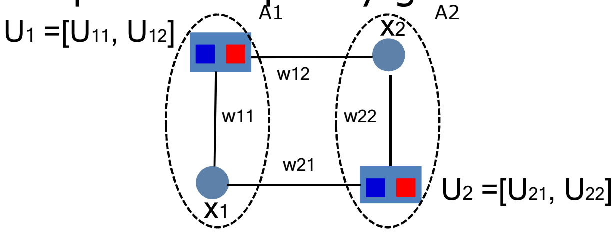
x_1	x_2	$U_1 = [U_{11}, U_{12}]$	$U_2 = [U_{21}, U_{22}]$	$U = U_1 + U_2$
0	0	(1,2)	(2,0)	(3,2)
0	1	(2,1)	(0,2)	(2,3)
1	0	(0,0)	(4,3)	(4,3)
1	1	(1,1)	(2,3)	(3,4)

Pareto optimal solutions:
it is not possible to increase the value of one objective without decreasing the value of another.

We instantiate Bounded Multi-Objective Max-Sum (B-MOMS)

- Extends the Bounded Max-Sum Algorithm
- Proceeds in 3 phases:
  1. Bounding phase
  2. Max-Sum phase
  3. Value Propagation phase

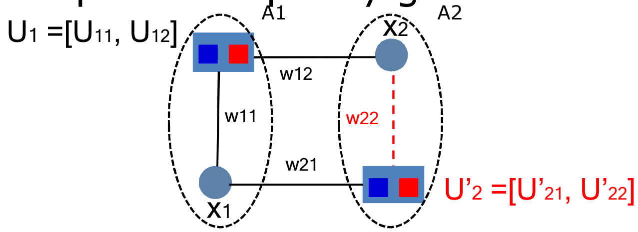
Phase 1: The Bounding Phase provides quality guarantees



- Prune the factor graph to a tree to guarantee convergence of the max-sum algorithm
- Remove edges with minimal impact on solution quality

105

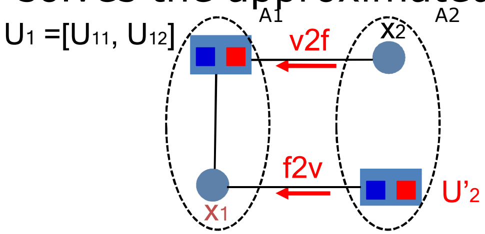
Phase 1: The Bounding Phase provides quality guarantees



- Prune the factor graph to a tree to guarantee convergence of the max-sum algorithm
- Remove edges with minimal impact on solution quality

106

Phase 2: The Max-Sum Phase solves the approximated problem



- Optimally solves the approximated problem
  (Main Theorem discussed earlier)

107

Phase 2: The Max-Sum Phase solves the approximated problem

- Messages flow between function and variable nodes of the factor graph
  - From variable to function

$$Q_{n\to m}(x_n) = \sum_{m'\in M(n)\setminus m} R_{m'\to n}(x_n)$$

  - From function to variable

$$R_{m\to n}(x_n) = \max_{\mathbf{x}_m\setminus n}\left(U_m(\mathbf{x}_m) + \sum_{n'\in N(m)\setminus n} Q_{n'\to m}(x_{n'})\right)$$

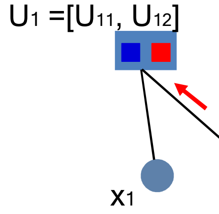
However, *max* and *+* operators are generalised to consider multiple objectives

108

Phase 2: Max-Sum messages now contain multiple non-dominated vectors

Example: computing the message from U_1 to x_1 :

x_1	x_2	$U_1 = [U_{11}, U_{12}]$
0	0	(1,2) + (0,1)
0	1	(2,1) + (1,0)
1	0	(0,0) + (0,1)
1	1	(1,1) + (1,0)



x_2	x_2 -> U_1
0	(0,1)
1	(1,0)

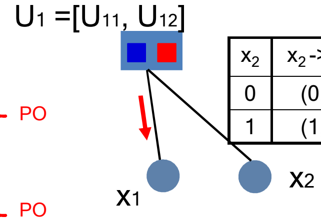
Phase 2: Max-Sum messages now contain multiple non-dominated vectors

Example: computing the message from U_1 to x_1 :

x_1	x_2	$U_1 + Q_{2\to1}$
0	0	(1,2) + (0,1)
0	1	(2,1) + (1,0)
1	0	(0,0) + (0,1)
1	1	(1,1) + (1,0)

PO

PO

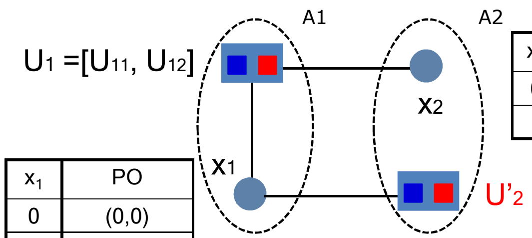


x_2	x_2 -> U_1
0	(0,1)
1	(1,0)

Result:

x_1	U_1 -> x_1
0	(1, 3), (3, 1)
1	(2,1)

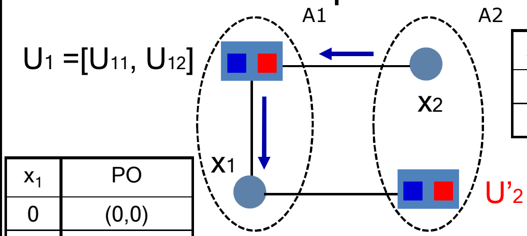
At the end of phase 2, each agent recovers its corresponding PO assignments



x_2	PO
0	(2,0)
1	(1,1)

x_1	PO
0	(0,0)
1	(2,0), (1,1)

Phase 3: Value Propagation selects one Pareto optimal assignment



x_2	PO
0	(2,0)
1	(1,1)

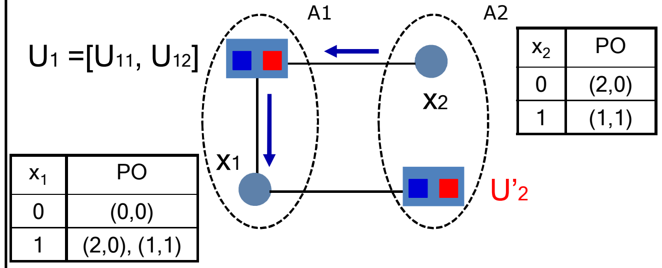
x_1	PO
0	(0,0)
1	(2,0), (1,1)

Two solutions: (1, 0) and (1, 1)

$U(x_1 = 1, x_2=0) = (2,0)$

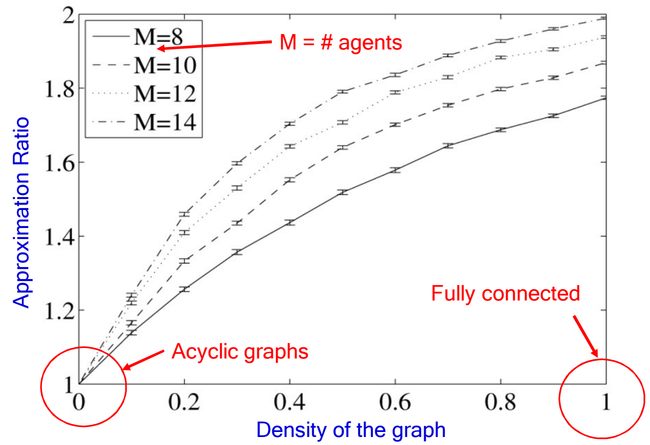
$U(x_1 = 1, x_2=1) = (1,1)$

Phase 3: Value Propagation selects a consistent Pareto optimal assignment

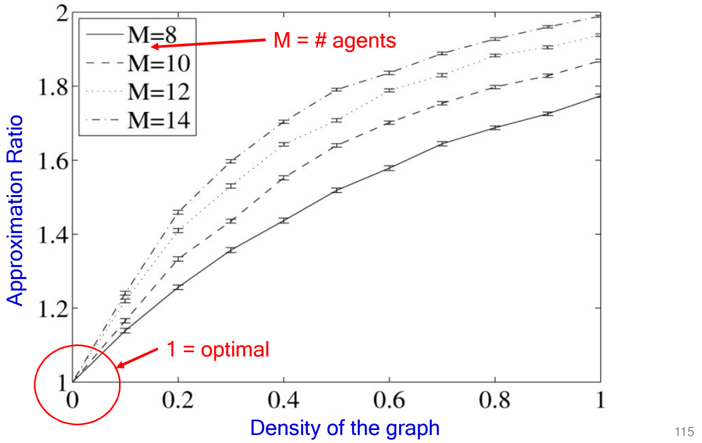


Either at **random**, or based on a **logical condition.**

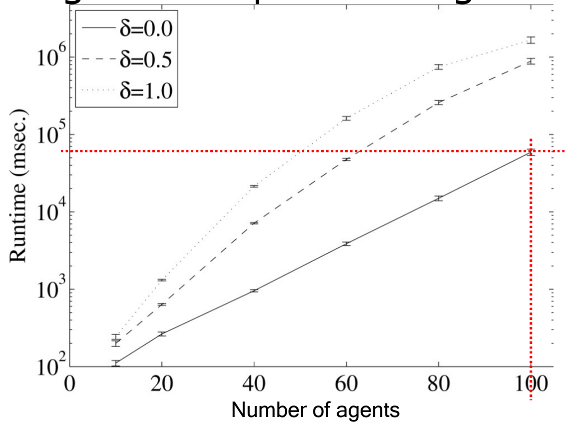
Our results: B-MOMS typically provides solutions within 50% of the optimal



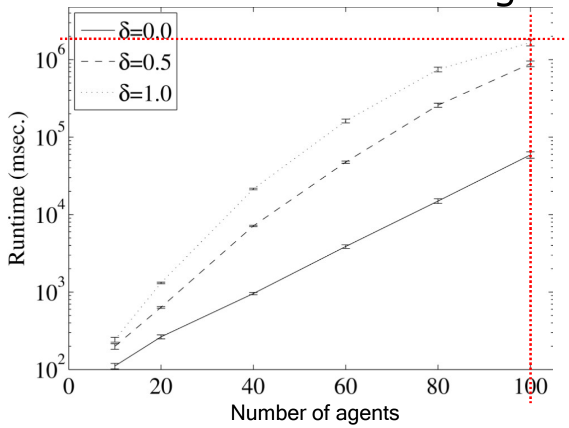
Our results: B-MOMS is optimal on acyclic graphs



Our results: On acyclic graphs, the runtime is, on average 0.5 sec per agent for up to 100 agents



Our results: Even on fully constrained problems, the runtime is < 30 minutes for 100 agents



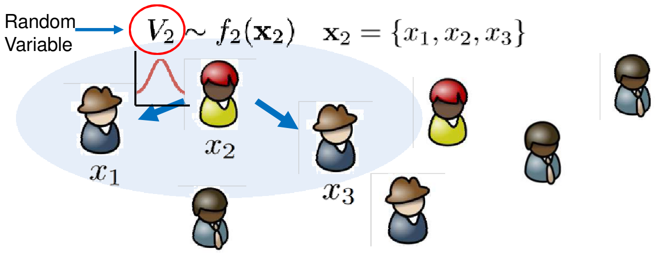
117

This seminar is about coordination problems and algorithms

I. Motivation
II. Case Study of Coordination on Unmanned Aerial Vehicles
III. Partially Ordered Distributed Contraint Optimisation Problems (PO-DCOPs)
   a. Problem and Algorithms Definition
   b. Multi Objective Distributive Constraint Optimisation Problems (MO-DCOPs)
   c. Risk Aware Distributive Constraint Optimisation Problems (RA-DCOPs)
IV. Conclusions and Future Work

118

So, local functions output probability distributions instead of scalars



119

The new objective is to maximise **expected utility**

Sum of local constraint values (= also random variable)

$$V = \sum_{i=1}^{m} V_i$$

Objective: maximise **expected utility** of the sum of values

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]$$

120

Why maximise **expected utility** instead of **expected value**?

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]$$

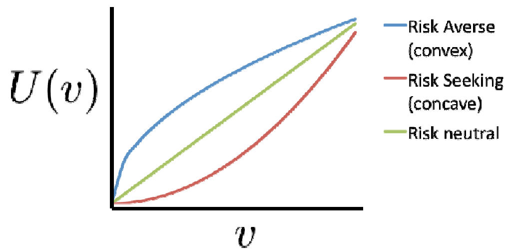
Instead of

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[\sum_{i=1}^{m} V_i\right]$$

121

Agents might not be risk neutral!
Utility function represents **risk** profile

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]$$

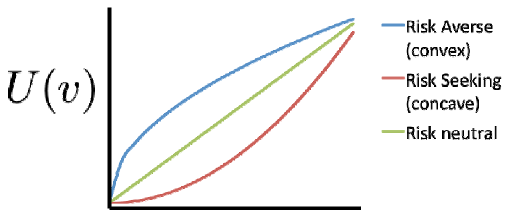


Standard risk theory

122

Agents might not be risk neutral!
Utility function represents **risk** profile

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]$$



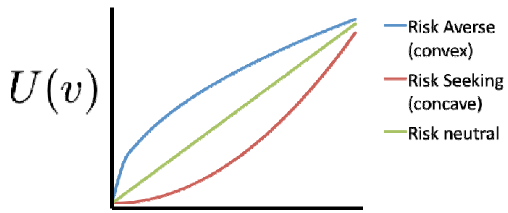
Standard risk theory

v = $$$$

123

Agents might not be risk neutral!
Utility function represents **risk** profile

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]$$



Standard risk theory

v = civilians saved

124

Risk Aware-DCOPs (RA-DCOPs) formalise **uncertainty** and **risk** in decentralised coordination problems

An **RA-DCOP** consists of:

1. Discrete decision variables	$\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$
2. Local functions expressing uncertain interactions between agents	$V_i \sim f_i(\mathbf{x}_i)$
3. A utility function mapping *value* to *utility*	$U : \mathbb{R} \to \mathbb{R}$
4. A global objective	$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]$

DCOP algorithms cannot solve RA-DCOPs

In general, an RA-DCOP can not be expressed as a sum of factors (U is not linear):

$$\underbrace{\boxed{\sum_{i=1}^{m} E[U(V_i)]}}_{\text{DCOP}} \neq \underbrace{\boxed{E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]}}_{\text{RA-DCOP}}$$

So, RA-DCOPs are not DCOPs!

What if we ignore uncertainty by using a DCOP algorithm to solve RA-DCOPs?

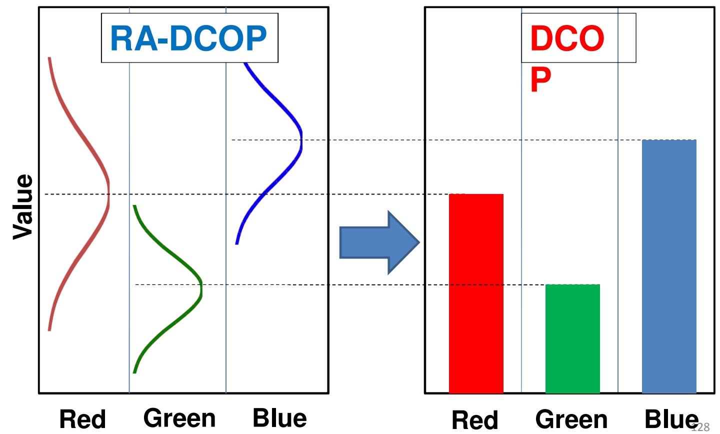
What happens if we transform a **RA-DCOP** into a **DCOP** by defining local functions as follows?

$$\boxed{f_i(\mathbf{x}_i) = E[U(V_i)]}$$

And thus maximise the left-hand side of:

$$\boxed{\sum_{i=1}^{m} E[U(V_i)]} \neq \boxed{E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]}$$

What if we ignore uncertainty by using a DCOP algorithm to solve RA-DCOPs?

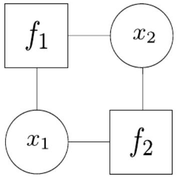


What if we ignore uncertainty by using a DCOP algorithm to solve RA-DCOPs?

An example:

$U(v) = \mu - \sigma$

x_1	x_2	f_1 μ	f_1 σ^2	f_2 μ	f_2 σ^2	f_1+f_2 μ	f_1+f_2 σ^2
0	0	9	8^2	10	15^2	19	17^2
0	1	3	5^2	10	12^2	13	13^2
1	0	15	7^2	5	24^2	20	25^2
1	1	2	4^2	2	3^2	4	5^2



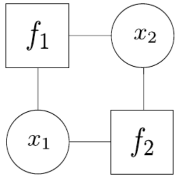
129

What if we ignore uncertainty by using a DCOP algorithm to solve RA-DCOPs?

An example:

$U(v) = \mu - \sigma$

x_1	x_2	f_1 μ	f_1 σ^2	f_2 μ	f_2 σ^2	f_1+f_2 μ	f_1+f_2 σ^2
0	0	9	8^2	10	15^2	19	17^2
0	1	3	5^2	10	12^2	13	13^2
1	0	15	7^2	5	24^2	20	25^2
1	1	2	4^2	2	3^2	4	5^2



Adding random variables: convolution operator

$V_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$

$V_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

$V_1 + V_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$

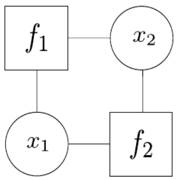
130

What if we ignore uncertainty by using a DCOP algorithm to solve RA-DCOPs?

An example:

$U(v) = \mu - \sigma$

x_1	x_2	f_1 μ	f_1 σ^2	f_2 μ	f_2 σ^2	f_1+f_2 μ	f_1+f_2 σ^2
0	0	9	8^2	10	15^2	19	17^2
0	1	3	5^2	10	12^2	13	13^2
1	0	15	7^2	5	24^2	20	25^2
1	1	2	4^2	2	3^2	4	5^2



Not additive:

$\boxed{U(f_1(0,0)) = 9 - 8 = 1}$

$\boxed{U(f_2(0,0)) = 10 - 5 = -5}$

$\boxed{U(f_1(0,0) + f_2(0,0)) = 19 - 17 = -2}$

131

What if we ignore uncertainty by using a DCOP algorithm to solve RA-DCOPs?

$U(v) = \mu - \sigma$

$\boxed{\sum_{i=1}^{m} E[U(V_i)]}$ $\boxed{E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]}$

x_1	x_2	f_1 μ	f_1 σ^2	f_2 μ	f_2 σ^2	f_1+f_2 μ	f_1+f_2 σ^2	SEU	EUS
0	0	9	8^2	10	15^2	19	17^2	-4	**2**
0	1	3	5^2	10	12^2	13	13^2	-4	0
1	0	15	7^2	5	24^2	20	25^2	-11	-5
1	1	2	4^2	2	3^2	4	5^2	$\mathbf{-3}$	-1

132

What if we ignore uncertainty by using a DCOP algorithm to solve RA-DCOPs?

$$\boxed{\sum_{i=1}^{m} E[U(V_i)]} \qquad \boxed{E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]}$$

x_1	x_2	f_1 μ	f_1 σ^2	f_2 μ	f_2 σ^2	f_1+f_2 μ	f_1+f_2 σ^2	SEU	EUS
0	0	9	8^2	10	15^2	19	17^2	-4	**2**
0	1	3	5^2	10	12^2	13	13^2	-4	0
1	0	15	7^2	5	24^2	20	25^2	-11	-5
1	1	2	4^2	2	3^2	4	5^2	$\mathbf{-3}$	-1

-1 instead of 2!

What if we ignore uncertainty by using a DCOP algorithm to solve U-DCOPs?

Sub-optimality!

$$\boxed{\sum_{i=1}^{m} E[U(V_i)]} \qquad \boxed{E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]}$$

x_1	x_2	f_1 μ	f_1 σ^2	f_2 μ	f_2 σ^2	f_1+f_2 μ	f_1+f_2 σ^2	SEU	EUS
0	0	9	8^2	10	15^2	19	17^2	-4	**2**
0	1	3	5^2	10	12^2	13	13^2	-4	0
1	0	15	7^2	5	24^2	20	25^2	-11	-5
1	1	2	4^2	2	3^2	4	5^2	$\mathbf{-3}$	-1

-1 instead of 2!

Main challenge: what should $\oplus$ and $\otimes$ be to solve RA-DCOPs?

DCOP: $$\arg\boxed{\max_{\mathbf{x}}}\boxed{\sum_{m=1}^{M}} f_m(\mathbf{x}_m)$$

$\boxed{\oplus}$ $\boxed{\otimes}$

RA-DCOP: $$\arg\boxed{\max_{\mathbf{x}} E\left[U\right.}\left.\left(\boxed{\sum_{i=1}^{m}} V_i\right)\right]$$

How do we define $\oplus$ and $\otimes$ for RA-DCOPS?

$\boxed{\oplus}$ $\boxed{\otimes}$

$$\arg\boxed{\max_{\mathbf{x}} E\left[U\right.}\left.\left(\boxed{\sum_{i=1}^{m}} V_i\right)\right]$$

How do we define $\oplus$ and $\otimes$ for U-DCOPS?

$\otimes$ sums random variables: **convolution**

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-a)g(a)da$$

$$\arg \underbrace{\max_{\mathbf{x}} E}_{\oplus} \left[U \left(\underbrace{\sum_{i=1}^{m}}_{\otimes} V_i \right) \right]$$

137

How do we define $\oplus$ and $\otimes$ for U-DCOPS?

$\otimes$ sums random variables: **convolution**

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-a)g(a)da$$

$\oplus$ should select random variables that have the *potential* of maximising global expected utility

$$\arg \underbrace{\max_{\mathbf{x}} E}_{\oplus} \left[U \left(\underbrace{\sum_{i=1}^{m}}_{\otimes} V_i \right) \right]$$

138

$\oplus$ selects all random variables that are not dominated under $\succeq$

RA-DCOP

$$X_i \in \oplus(X_1, \ldots, X_n) \Leftrightarrow \nexists X_j \succ X_i$$

DCOP

$$x_i \in \max(x_1, \ldots, x_n) \Leftrightarrow \nexists x_j > x_i$$

139

To benchmark U-GDL, we compared against a GDL algorithm (max-sum)

Main question:
What happens if we ignore uncertainty?

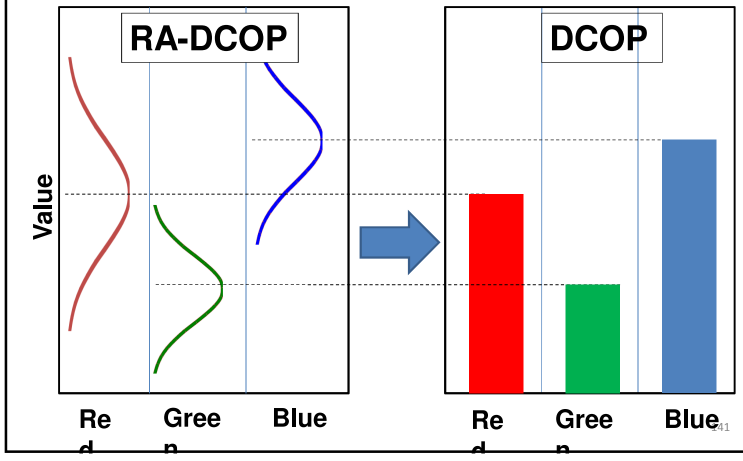
PO-GDL
RA-DCOP $\arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right]$

VS.

GDL
DCOP $\arg\max_{\mathbf{x}} \sum_{i=1}^{m} E[U(V_i)]$

140

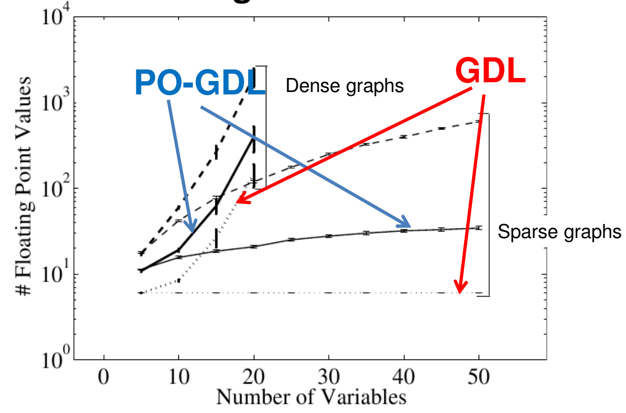
To benchmark U-GDL, we compared against a GDL algorithm (max-sum)



141

What happens if we ignore uncertainty?

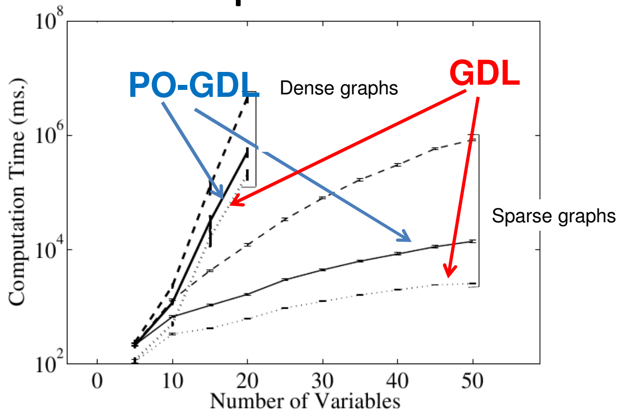
Messages are smaller



142

What happens if we ignore uncertainty?

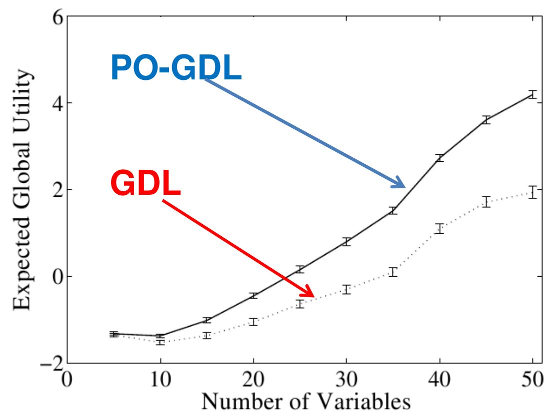
Less computation needed



143

What happens if we ignore uncertainty?

We lose optimality!



144

This seminar is about coordination problems and algorithms

I. Motivation
II. Case Study of Coordination on Unmanned Aerial Vehicles
III. Partially Ordered Distributed Contraint Optimisation Problems (PO-DCOPs)
   a. Problem and Algorithms Definition
   b. Multi Objective Distributive Constraint Optimisation Problems (MO-DCOPs)
   c. Risk Aware Distributive Constraint Optimisation Problems (RA-DCOPs)
IV. Conclusions and Future Work

145

To summarise:

- We presented a detailed study of coordination problems and GDL algorithms:
  - In practice: we presented a case study where max-sum is deployed in a multi-agent system for disaster response.
  - In theory: we extended the DCOP and the GDL frameworks to represent problems involving multiple interactions
    - We presented a study on multi-objective and on risk-aware coordination problems.

146

To summarise:

- Our initial empirical evaluation emphasizes that:
  - Considering the complexity of the problems the algorithms are efficient both in terms of computation and communication.
  - This complexity is, however, still not sufficient to deploy these techniques in the real world.

147

Future Work:

- We wish to study approximation techniques for these problems
  - Some questions:
    - Can we use standard max-sum?
    - Can we use pruning techniques to cut the search space or the message size?
    - Can we make these algorithms more efficient to solve dynamic problems?

148