

Toward coordination languages for distributed computation

Adrian Pearce

joint work with Ryan Kelly, Susannah Soon and Terence Law

Intelligent Agent Laboratory & NICTA Victoria Laboratory
Department of Computer Science and Software Engineering
University of Melbourne

28th August 2006



Talk outline

The coordination problem

Coordination languages

Coordination graphs

Distributed constraint satisfaction

Graph decomposition

Summary



The coordination problem

When computational devices have overlapping or common objectives or when agents rely on one-another to complete joints tasks,

- ▶ There is a lack of coordination languages and algorithms available in the literature,
- ▶ The problem is especially challenging where agents have limited knowledge about, and control over, other agents.
- ▶ There is a clear need for capturing and utilising richer dependency information in distributed computation tasks in ways that guarantee more effective and efficient solutions.



The coordination problem

When computational devices have overlapping or common objectives or when agents rely on one-another to complete joints tasks,

- ▶ There is a lack of coordination languages and algorithms available in the literature,
- ▶ The problem is especially challenging where agents have limited knowledge about, and control over, other agents.
- ▶ There is a clear need for capturing and utilising richer dependency information in distributed computation tasks in ways that guarantee more effective and efficient solutions.



The coordination problem

When computational devices have overlapping or common objectives or when agents rely on one-another to complete joints tasks,

- ▶ There is a lack of coordination languages and algorithms available in the literature,
- ▶ The problem is especially challenging where agents have limited knowledge about, and control over, other agents.
- ▶ There is a clear need for capturing and utilising richer dependency information in distributed computation tasks in ways that guarantee more effective and efficient solutions.



The coordination problem

When computational devices have overlapping or common objectives or when agents rely on one-another to complete joints tasks,

- ▶ There is a lack of coordination languages and algorithms available in the literature,
- ▶ The problem is especially challenging where agents have limited knowledge about, and control over, other agents.
- ▶ There is a clear need for capturing and utilising richer dependency information in distributed computation tasks in ways that guarantee more effective and efficient solutions.



Coordination languages

- ▶ High level program execution
- ▶ Controlled nondeterminism: *Golog* (Levesque, Reiter and Lesperance, Journal of logic programming, 1997)
- ▶ *Congolog* (De Giacomo, Lesperance and Levesque, Artificial Intelligence 2000)
- ▶ *MindiGolog* (Towards High-level programming for distributed problem solving, Kelly and Pearce, in submission)



Coordination languages

- ▶ High level program execution
- ▶ **Controlled nondeterminism: *Golog*** (Levesque, Reiter and Lesperance, Journal of logic programming, 1997)
- ▶ *Congolog* (De Giacomo, Lesperance and Levesque, Artificial Intelligence 2000)
- ▶ *MindiGolog* (Towards High-level programming for distributed problem solving, Kelly and Pearce, in submission)



Coordination languages

- ▶ High level program execution
- ▶ Controlled nondeterminism: *Golog* (Levesque, Reiter and Lesperance, Journal of logic programming, 1997)
- ▶ *Congolog* (De Giacomo, Lesperance and Levesque, Artificial Intelligence 2000)
- ▶ *MindiGolog* (Towards High-level programming for distributed problem solving, Kelly and Pearce, in submission)



Coordination languages

- ▶ High level program execution
- ▶ Controlled nondeterminism: *Golog* (Levesque, Reiter and Lesperance, Journal of logic programming, 1997)
- ▶ *Congolog* (De Giacomo, Lesperance and Levesque, Artificial Intelligence 2000)
- ▶ ***MindiGolog*** (Towards High-level programming for distributed problem solving, Kelly and Pearce, in submission)



Coordination graphs

- ▶ **Constraint graphs**, that capture agent dependencies and task constraints,

Conte and Sichman, *Dependence graphs: dependence within and between groups*, Computational and mathematical organisation theory, 2002

- ▶ and **organisational structures**, that simplify operations that lead to the formation of joint goals and actions and reduce the cost of inter-agent communication.

Grosz and Kraus, *Collaborative plans for complex group action*, Artificial Intelligence, 1996.

Tidhar, Rao and Sonenberg, *Guided team selection* ICMAS-96.

Tambe, *Towards flexible teamwork* JAIR 1997,

Durfee, *Practically coordinating* AI Magazine, 1999

Coordination graphs

- ▶ **Constraint graphs**, that capture agent dependencies and task constraints,

Conte and Sichman, *Dependence graphs: dependence within and between groups*, Computational and mathematical organisation theory, 2002

- ▶ and **organisational structures**, that simplify operations that lead to the formation of joint goals and actions and reduce the cost of inter-agent communication.

Grosz and Kraus, *Collaborative plans for complex group action*, Artificial Intelligence, 1996.

Tidhar, Rao and Sonenberg, *Guided team selection* ICMAS-96.

Tambe, *Towards flexible teamwork* JAIR 1997,

Durfee, *Practically coordinating* AI Magazine, 1999

Coordination graphs

- ▶ **Constraint graphs**, that capture agent dependencies and task constraints,

Conte and Sichman, *Dependence graphs: dependence within and between groups*, Computational and mathematical organisation theory, 2002

- ▶ and **organisational structures**, that simplify operations that lead to the formation of joint goals and actions and reduce the cost of inter-agent communication.

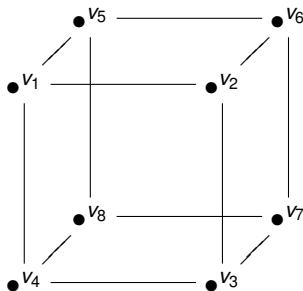
Grosz and Kraus, *Collaborative plans for complex group action*, Artificial Intelligence, 1996.

Tidhar, Rao and Sonenberg, *Guided team selection* ICMAS-96.

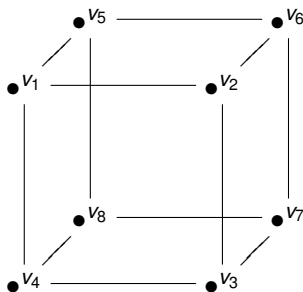
Tambe, *Towards flexible teamwork* JAIR 1997,

Durfee, *Practically coordinating* AI Magazine, 1999





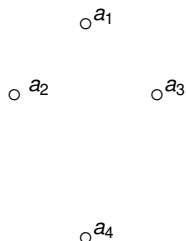
- ▶ A (simple) graph $G = (V, E)$, is defined by a set of vertices $V = \{v_1, \dots, v_n\}$ and an (unordered) set of edges $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \dots\}$
- ▶ Vertex identifiers, v_1, \dots, v_n , are simply unique identifiers for vertices, as distinct from
- ▶ value- or logic-based *attributes* or *labels* which can attribute either vertices and/or edges.



- ▶ A (simple) graph $G = (V, E)$, is defined by a set of vertices $V = \{v_1, \dots, v_n\}$ and an (unordered) set of edges $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \dots\}$
- ▶ Vertex identifiers, v_1, \dots, v_n , are simply unique identifiers for vertices, as distinct from
- ▶ value- or logic-based *attributes* or *labels* which can attribute either vertices and/or edges.

Example: asynchronous, distributed graph colouring

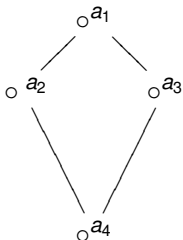
Simplest case: two colours; unrestricted is NP-complete



- ▶ Each agent is considered a vertex in the graph.
- ▶ Edges correspond to constraints between agents (assigned initial colours).
- ▶ Agents must solve colouring asynchronously, through computation and communication (using appropriate representations).

Example: asynchronous, distributed graph colouring

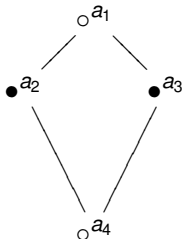
Simplest case: two colours; unrestricted is NP-complete



- ▶ Each agent is considered a vertex in the graph.
- ▶ Edges correspond to constraints between agents (assigned initial colours).
- ▶ Agents must solve colouring asynchronously, through computation and communication (using appropriate representations).

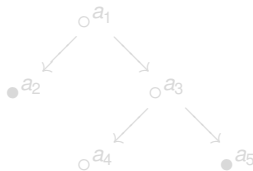
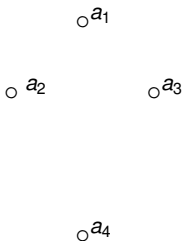
Example: asynchronous, distributed graph colouring

Simplest case: two colours; unrestricted is NP-complete



- ▶ Each agent is considered a vertex in the graph.
- ▶ Edges correspond to constraints between agents (assigned initial colours).
- ▶ Agents must solve colouring asynchronously, through computation and communication (using appropriate representations).

Algorithm 1: asynchronous and distributed



Organisational structure:

• Graphs hierarchy

• Graphs decomposition

• Graphs composition

• Graphs decomposition

Constraint graph:

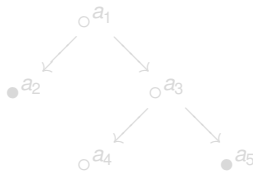
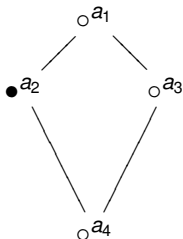
$G = (V, E)$, where

$V = \{a_1, \dots, a_n\}$

$E = \{\{a_1, a_2\}, \{a_1, a_3\}, \dots\}$



Algorithm 1: asynchronous and distributed



Organisational structure:

• *Constraint graph*

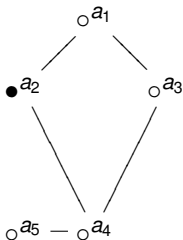
Constraint graph:

$G = (V, E)$, where

$V = \{a_1, \dots, a_n\}$

$E = \{\{a_1, a_2\}, \{a_1, a_3\}, \dots\}$

Algorithm 1: asynchronous and distributed



Organisational structure:

Asynchronous algorithm

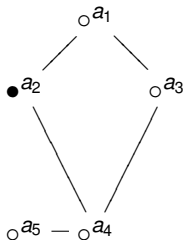
Constraint graph:

$G = (V, E)$, where

$V = \{a_1, \dots, a_n\}$

$E = \{\{a_1, a_2\}, \{a_1, a_3\}, \dots\}$

Algorithm 1: asynchronous and distributed

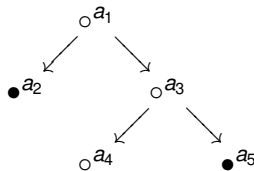


Constraint graph:

$G = (V, E)$, where

$V = \{a_1, \dots, a_n\}$

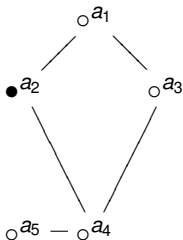
$E = \{\{a_1, a_2\}, \{a_1, a_3\}, \dots\}$



Organisational structure:

- ▶ Creates hierarchy over of agents,
- ▶ Costs propagate up asynchronously,
- ▶ Constraints propagate down.

Algorithm 1: asynchronous and distributed

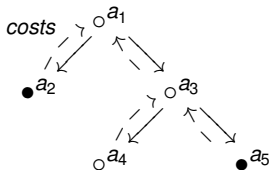


Constraint graph:

$G = (V, E)$, where

$V = \{a_1, \dots, a_n\}$

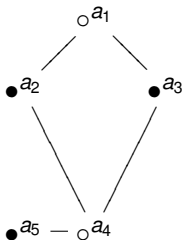
$E = \{\{a_1, a_2\}, \{a_1, a_3\}, \dots\}$



Organisational structure:

- ▶ Creates hierarchy over of agents,
- ▶ Costs propagate up asynchronously,
- ▶ Constraints propagate down.

Algorithm 1: asynchronous and distributed

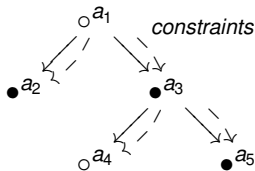


Constraint graph:

$G = (V, E)$, where

$V = \{a_1, \dots, a_n\}$

$E = \{\{a_1, a_2\}, \{a_1, a_3\}, \dots\}$



Organisational structure:

- ▶ Creates hierarchy over of agents,
- ▶ Costs propagate up asynchronously,
- ▶ Constraints propagate down.

Each algorithm has yielded approximately 10^2 speedup (each improvement uses more and more relational information)

- ▶ Yokoo, Hirayama *Algorithms for distributed constraint satisfaction*, AAMAS Journal, 2000
- ▶ Modi, Shen, Tambe and Yokoo *ADOPT: asynchronous distributed constraint optimisation*, Artificial Intelligence Journal, 2005
- ▶ Mailler and Lesser *Cooperative Mediation for solving distributed constraint satisfaction problems*, AAMAS04 conference, N.Y. 2004
- ▶ Law and Pearce *A multi-stage graph decomposition algorithm for distributed constraint optimisation* (submitted 2006).



Each algorithm has yielded approximately 10^2 speedup (each improvement uses more and more relational information)

- ▶ Yokoo, Hirayama *Algorithms for distributed constraint satisfaction*, AAMAS Journal, 2000
- ▶ Modi, Shen, Tambe and Yokoo *ADOPT: asynchronous distributed constraint optimisation*, Artificial Intelligence Journal, 2005
- ▶ Mailler and Lesser *Cooperative Mediation for solving distributed constraint satisfaction problems*, AAMAS04 conference, N.Y. 2004
- ▶ Law and Pearce *A multi-stage graph decomposition algorithm for distributed constraint optimisation* (submitted 2006).



Each algorithm has yielded approximately 10^2 speedup (each improvement uses more and more relational information)

- ▶ Yokoo, Hirayama *Algorithms for distributed constraint satisfaction*, AAMAS Journal, 2000
- ▶ Modi, Shen, Tambe and Yokoo *ADOPT: asynchronous distributed constraint optimisation*, Artificial Intelligence Journal, 2005
- ▶ Mailler and Lesser *Cooperative Mediation for solving distributed constraint satisfaction problems*, AAMAS04 conference, N.Y. 2004
- ▶ Law and Pearce *A multi-stage graph decomposition algorithm for distributed constraint optimisation* (submitted 2006).



Each algorithm has yielded approximately 10^2 speedup (each improvement uses more and more relational information)

- ▶ Yokoo, Hirayama *Algorithms for distributed constraint satisfaction*, AAMAS Journal, 2000
- ▶ Modi, Shen, Tambe and Yokoo *ADOPT: asynchronous distributed constraint optimisation*, Artificial Intelligence Journal, 2005
- ▶ Mailler and Lesser *Cooperative Mediation for solving distributed constraint satisfaction problems*, AAMAS04 conference, N.Y. 2004
- ▶ Law and Pearce *A multi-stage graph decomposition algorithm for distributed constraint optimisation* (submitted 2006).



Each algorithm has yielded approximately 10^2 speedup (each improvement uses more and more relational information)

- ▶ Yokoo, Hirayama *Algorithms for distributed constraint satisfaction*, AAMAS Journal, 2000
- ▶ Modi, Shen, Tambe and Yokoo *ADOPT: asynchronous distributed constraint optimisation*, Artificial Intelligence Journal, 2005
- ▶ Mailler and Lesser *Cooperative Mediation for solving distributed constraint satisfaction problems*, AAMAS04 conference, N.Y. 2004
- ▶ Law and Pearce *A multi-stage graph decomposition algorithm for distributed constraint optimisation* (submitted 2006).

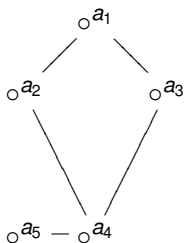
Problem: typically requires exponential communications

Example: wireless, sensor-actuation devices

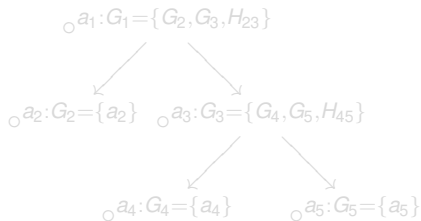
- ▶ Communicating one bit of information typically costs two orders of magnitude more than computing one bit of information.



Algorithm 2: graph decomposition



$$G = \{G', G'', H\}$$



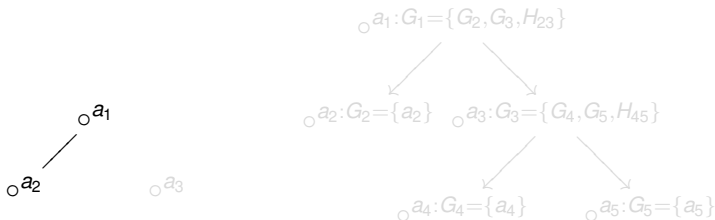
Graph decomposition:

Linear ordering a_1, a_2, a_3, a_4, a_5

Linear ordering a_1, a_2, a_3, a_4, a_5

Linear ordering a_1, a_2, a_3, a_4, a_5

Algorithm 2: graph decomposition



Graph decomposition:

1. Create ordering a_1, a_2, \dots, a_n

2. For $i = 1$ to n

3. $G_i = G - \{G_j, H_{ij} \mid j < i\}$

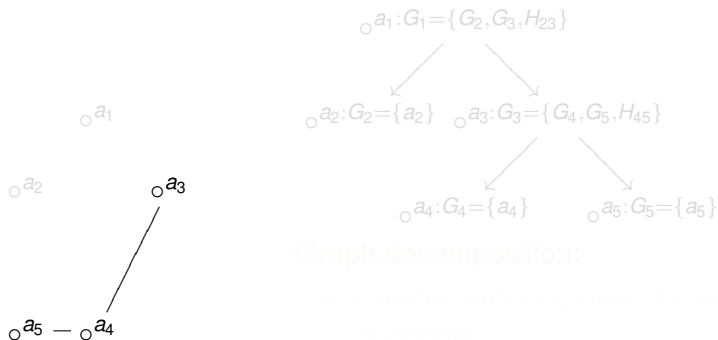
4. $H_i = \{H_{ij} \mid j < i\}$

5. $G = \{G_1, G_2, \dots, G_n, H_1, H_2, \dots, H_n\}$

a_5 a_4

$$G = \{\mathbf{G}', G'', H\}$$

Algorithm 2: graph decomposition



Graph decomposition:

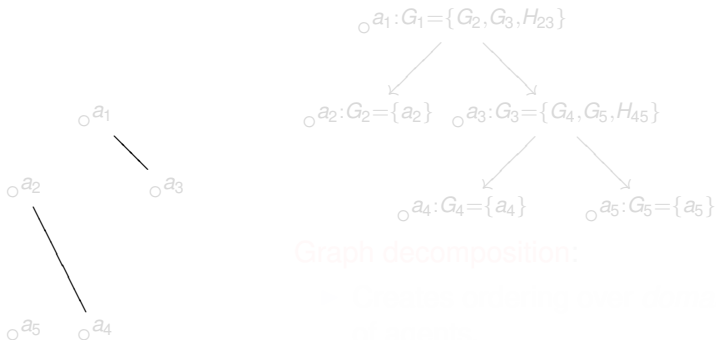
1. Create ordering a_1, \dots, a_n

2. $G_i = \{G_j, H_{ij} \mid j > i\}$

3. $G_i = \{G_j, H_{ij} \mid j > i\}$

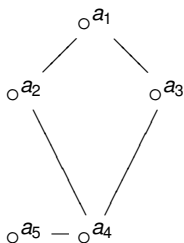
$$G = \{G', \mathbf{G}'', H\}$$

Algorithm 2: graph decomposition

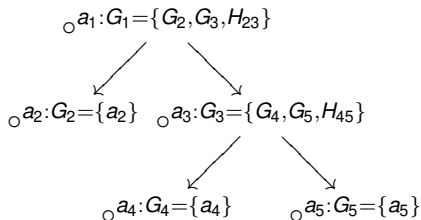


$$G = \{G', G'', H\}$$

Algorithm 2: graph decomposition



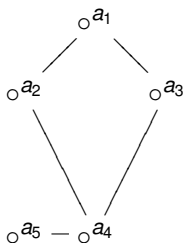
$$G = \{G', G'', H\}$$



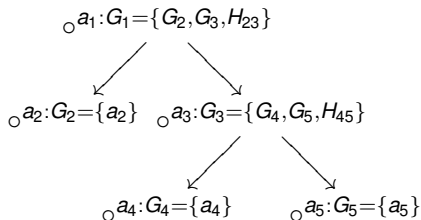
Graph decomposition:

- ▶ Creates ordering over *domain* of agents,
- ▶ Each agent solves for each (sub)graph,
- ▶ Partial solutions and/or *bound intervals* propagate up.

Algorithm 2: graph decomposition



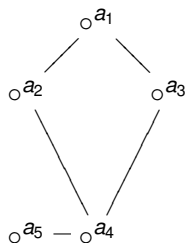
$$G = \{G', G'', H\}$$



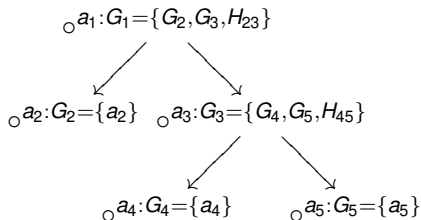
Graph decomposition:

- ▶ Creates ordering over *domain* of agents,
- ▶ Each agent solves for each (sub)graph,
- ▶ Partial solutions and/or *bound intervals* propagate up.

Algorithm 2: graph decomposition



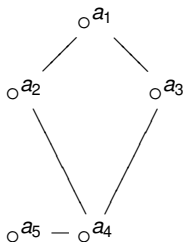
$$G = \{G', G'', H\}$$



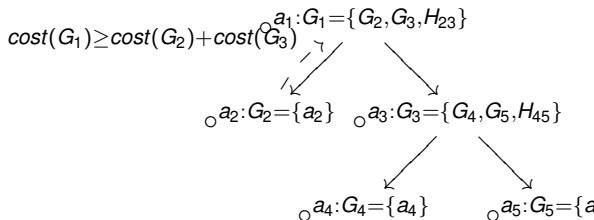
Graph decomposition:

- ▶ Creates ordering over *domain* of agents,
- ▶ Each agent solves for each (sub)graph,
- ▶ Partial solutions and/or *bound intervals* propagate up.

Algorithm 2: graph decomposition



$$G = \{G', G'', H\}$$



Graph decomposition:

- ▶ Creates ordering over *domain* of agents,
- ▶ Each agent solves for each (sub)graph,
- ▶ Partial solutions and/or *bound intervals* propagate up.

Summary

- ▶ Coordination language being developed, based on distributed, parallel version of situation calculus (*MindiGolog*).
- ▶ Parallel execution results in concurrency issues, managed via distributed constraint satisfaction (*distdecomp* algorithm).
- ▶ Epistemic aspects, as a result of partial observability, is addressed by incorporating knowledge into situation calculus formalism (Kelly and Pearce).

Summary

- ▶ Coordination language being developed, based on distributed, parallel version of situation calculus (*MindiGolog*).
- ▶ Parallel execution results in concurrency issues, managed via distributed constraint satisfaction (*distdecomp* algorithm).
- ▶ Epistemic aspects, as a result of partial observability, is addressed by incorporating knowledge into situation calculus formalism (Kelly and Pearce).



Summary

- ▶ Coordination language being developed, based on distributed, parallel version of situation calculus (*MindiGolog*).
- ▶ Parallel execution results in concurrency issues, managed via distributed constraint satisfaction (*distdecomp* algorithm).
- ▶ Epistemic aspects, as a result of partial observability, is addressed by incorporating knowledge into situation calculus formalism (Kelly and Pearce).