Distributed Constraint Reasoning

-a paradigm for effective coordination in multiagent systems-

Jörg Denzinger[1], Adrian Petcu[2], Marius C. Silaghi[3], Makoto Yokoo[4]

[1]Computer Science Department, University of Calgary

[2]Ecole Politechnique Federale de Lausanne

[3]Computer Sciences Department, Florida Institute of Technology

[4]Information Science Department, Kyushu University

Foreword

This tutorial aims on the one hand side to provide an introduction into distributed constraint reasoning and on the other side to highlight the current research trends in the area. The tutorial draws from earlier tutorials by Makoto Yokoo (CP98), Jörg Denzinger (IJCAI01) and Yokoo, Denzinger and Marius Silaghi (IJCAI03, AAMAS04, IJCAI05) and from the PhD dissertations of Silaghi and Adrian Petcu. The previous tutorials were created as half-day tutorials and this was also the planned time frame for this tutorial. Consequently, we had the hard decisions to make which of the concepts to present and where to put foci. We decided to focus the introduction part on asynchronous methods and to highlight new approaches to constraint optimization and the whole topic of semi-cooperative agents with its connections to issues like privacy and reigning in the competitiveness of agents. To make up for this rather subjective selection, we include a structured bibliography of works on distributed knowledge-based search and distributed constraint reasoning with some notations of where to place the particular works.

Calgary, Lausanne, Melbourne (FL), Kyoto, September 2007
Jörg Denzinger, Adrian Petcu, Marius C. Silaghi, Makoto Yokoo

1. Introduction: Constraint Reasoning (I)

Given:

- $X = \{x_1,\ldots,x_m\}$ set of variables
- $D = \{D_1,\ldots,D_m\}$ set of domains for the variables, i.e. $x_i \in D_i$
  If D_i is finite, let $D_i = \{v_{i,1},\ldots,v_{i,d(i)}\}$
- $C = \{c_1,\ldots,c_k\}$ set of constraints over X;
  the constraint c_i is represented as a predicate $p_i(y_1,\ldots,y_j)$, $\{y_1,\ldots,y_j\} \subseteq X$, that checks the possible value assignment combinations for $y_1,\ldots,y_j$ if they fulfill the particular constraint

Constraint Reasoning (II)

Possible Goals:

- Find a value assignment for the variables that fulfills all constraints
  - ☛ constraint satisfaction
- Find a value assignment for the variables that optimizes a function goal: $D_1 \times \ldots \times D_m \rightarrow R$
  - ☛ constraint optimization
- Generate new constraints $c_{k+1},\ldots$ that are consequences of C (and fulfill additional conditions)
  - ☛ constraint deduction

Why Constraint Reasoning?

- A lot of problems can be transformed in either constraint satisfaction or constraint optimization problems:
  - Planning problems
  - Scheduling problems
  - ...
- Many negotiation approaches can be seen as a constraint reasoning process
  ☛ important for Multi-Agent Systems

Distributed Constraint Reasoning (I)

Given:

- Ag = {A_1,...,A_n} set of agents
- Each agent has a set of (private, intra-agent) constraints (although this set can be empty for a particular agent)
- For some type of problems, there is an additional set of (public, inter-agent) constraints. Restrictions can also be placed on the variables for which each agent may propose assignments (modifiers).
- The combination of variables of all agents and all their intra- (and eventual inter-) agent constraints form a constraint reasoning problem

Agents can exchange messages

Distributed Constraint Reasoning (II)

Possible goals:

Similar to goals for constraint reasoning (of combined problem)

Plus fulfilling/conforming to

- Additional individual goals of agents
  - Privacy
  - Individual interests, etc
  - ☛ Semi-cooperative agents
- Additional priorities for (dominance of) certain agents
- Additional limits/restrictions on communication between agents

What are Agents in this Context?

- Entities that act in a shared environment and have their own
  - goals,
  - desires and
  - beliefs
- But: in Distributed Constraint Reasoning we also have
  - at least one common goal: find a solution to the given problem instance

Why Distributed Constraint Reasoning?

- Often problem instances come already distributed without a way to bring all the information together into one place (☞ naturally distributed problems)
- Multi-agent conflict resolution can be modeled as distributed constraint reasoning problems
- Distribution can increase efficiency:
  - Each agent solves smaller problems
  - Each agent can be on its own processor
  - But: there can be a large communication and cooperation overhead!

Examples:

- Many graph problems ☞ artificial distribution (mostly), both constr. satisfaction and optimization
- N-queens problem ☞ artificial distribution, constr. satisfaction
- Meeting scheduling ☞ naturally distributed, either constr. satisfaction or optimization
- Job-shop scheduling ☞ naturally distributed (mostly), constr. optimization
- Resource assignments to agents ☞ naturally distributed, either constr. satisfaction or optimization

An example in more detail: meeting scheduling

Problem: a group of managers (agents) have to conduct several meetings as a whole group or as subgroups. They are situated in different cities and where a meeting takes place is part of the decision making. And these managers have busy schedules.

Task: find appropriate places and meeting times to have all meetings!

Meeting scheduling (II)

Constraint satisfaction:

Goal: find just one assignment of meetings to (time,place) pairs that works

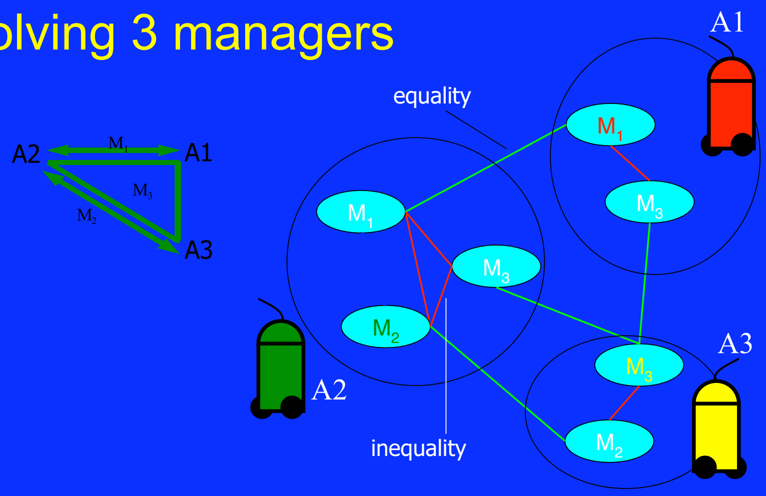
Formulation as constraint satisfaction problem:

Each meeting represented by a variable with domain being all (time,place) pairs

Constraints: manager schedules, meetings overlapping in attending persons cannot be scheduled at the same time, travel to a meeting must be possible for attendees

DisCSP variant: assign a manager to each meeting with the task of scheduling it

Problem: set up 3 meetings involving 3 managers



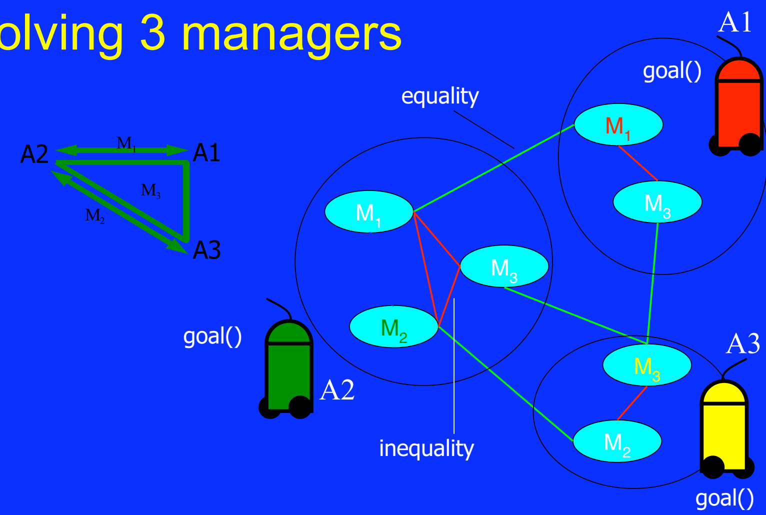
Meeting scheduling (III)

Constraint optimization:

For each combination of (time,place) pairs and meetings there is a cost of holding the particular meeting at that time and place (☞ function goal).

Goal: find an assignment of meetings to (time,place) pairs with minimal combined cost that fulfills all constraints

Problem: set up 3 meetings involving 3 managers



Meeting scheduling (IV)

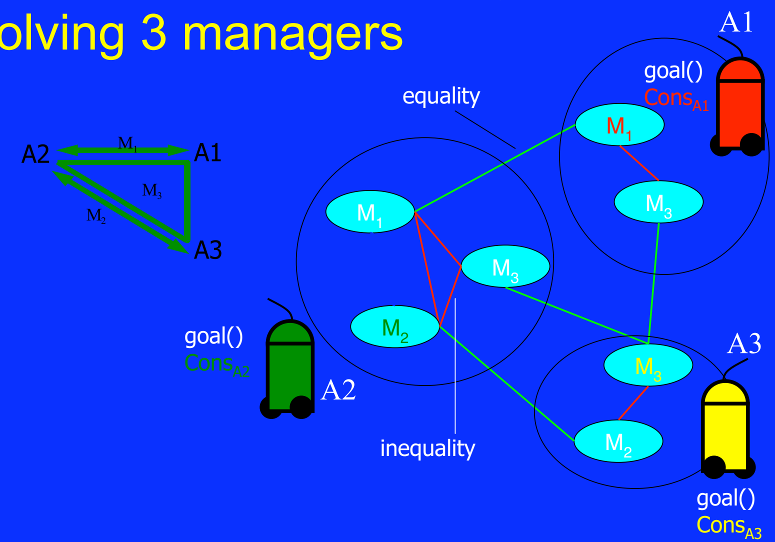
1st extension: privacy:

Some managers have (time,place) pairs that they should be available for, but that they do not want to use ☞ additional constraints

But: they do not want their bosses to know that!

Goal: find assignment of meetings to (time,place) pairs fulfilling all the constraints without revealing at least the additional constraints

Problem: set up 3 meetings involving 3 managers



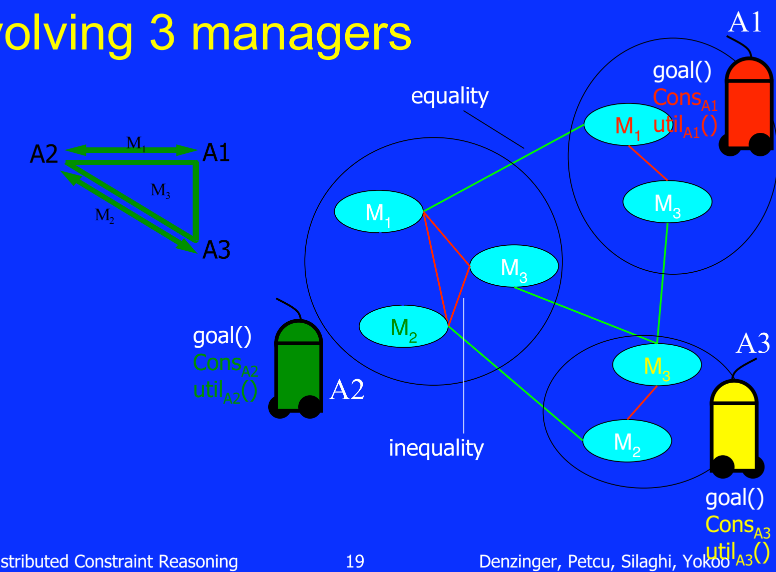
Meeting scheduling (V)

2nd extension: semi-cooperative agents

The managers have individual utility functions measuring the meetings they have to attend and the (time,place) pairs assigned to it.

Goal: minimize the costs of the assignment, while fulfilling all constraints and achieve good values of the managers' utility functions

Problem: set up 3 meetings involving 3 managers



Solving Constraint Reasoning Problems

- Every type of search-based solution technique has been applied to solving constraint reasoning problems:
  - Set-based methods:
    evolutionary alg., simulated annealing, tabu-search, etc.
  - Tree- and graph-based decision sequences
  - Simplex method
  - and so on
  - Either with global views or local views

Structure of the rest of the tutorial

1. Introduction
2. Classifying DisCR approaches
3. Top-Down Approaches
4. Bottom-Up approaches
5. Including privacy reasoning

2. Classifying DisCR approaches

There are many dimensions that can be used to classify the existing approaches for solving distributed constraint reasoning problems:

- Based on the cooperation paradigm used:
  - Dividing problem instance into subproblems
  - Working on a common search state
  - Improving on the competition approach
  - ☞ Naturally distributed problems favor dividing problem instance into subproblems

Classification dimensions (II)

- Agent communication:
  - Synchronous: agents can make assumptions about the view of all other agents when composing messages
  - Asynchronous: at no moment an agent knows the stability of the views of other agents
    ☞ often increases robustness of system
  - In the middle: concepts of epochs [ArmDur97], more on the synchronous side

Classification dimensions (III)

- Rank ordering of agents or not
- With or without central control
- Consensus building:
  - Top-down: higher rank agents decide first, lower rank agents have to find ways to comply
  - Bottom-up: agents build solutions starting with small pieces that get bigger and bigger

Classification dimensions (IV)

Dimensions inherited from sequential constraint reasoning:

- Complete vs. incomplete methods
- Basic search technique used
- Type of reasoning:
  - Satisfaction
  - Optimization
  - Deduction

3. Top-Down Approaches

Selected examples:

- Asynchronous, complete, without central control but ordering on agents, naturally distributed, agent=variable, doing satisfaction (with a little deduction): Asynchronous Backtracking
- Asynchronous, complete, without central control but dynamic ordering, naturally distributed, agent=variable, doing satisfaction (with a little deduction): Asynchronous weak commitment

Selected examples (II)

- Asynchronous, complete, control via ordering of agents, agent can be responsible for one or several variables, but still based on dividing problem instance into subproblems, doing optimization: ADOPT

3.1 Asynchronous Backtracking [YokDurIshKuw *92*]

Characteristics:

- Each agent acts asynchronously and concurrently without any global control.
  - Each agent communicates its tentative value assignment to related agents, then negotiates if constraint violations exist.

Merit:

- no communication/processing bottleneck, parallelism, privacy/security

Problems to solve:

- guaranteeing the completeness of the algorithm
  - avoiding infinite processing loops
  - escaping from dead-ends

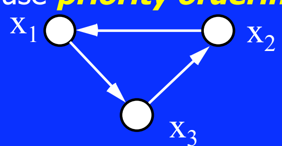
Avoiding Infinite Processing Loops

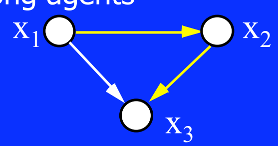
Cause of infinite processing loops:

- ***cycle*** in the constraint network
- If there exists no cycle, an infinite processing loop never occurs.

Remedy:

- directing links without creating cycles
- use ***priority ordering*** among agents





Escaping from Dead-Ends (I)

When there exists no value that satisfies constraints:

Sequential backtracking: change the most recent decision

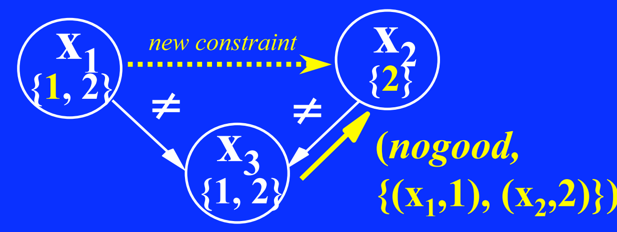
- simple control is inadequate under asynchronous changes

Asynchronous backtracking: derive/communicate a new constraint (***nogood***)

- other agents try to satisfy the new constraint; thus the nogood sending agent can escape from the dead-end
- can be done concurrently and asynchronously

☞ Constitutes constraint deduction component

Escaping from Dead-Ends (II)



Completeness of Asynchronous BT

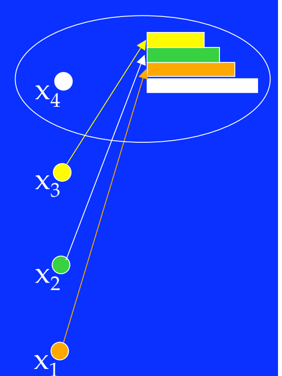
- An agent never stops its processing (changing value assignments/sending messages) unless it satisfies all constraints among higher priority agents or an empty set becomes a nogood.
- There exists no infinite processing loop (by induction)
  - The highest priority agent x_1 will not be in an infinite processing loop.
  - If x_1 to x_{k-1} are stable, then x_k cannot be in an infinite processing loop.

Space Complexity of ABT

- An agent might need to record an exponential number of nogoods, however,
  - An agent can throw away obsolete nogoods (that does not match the current states of other agents)
  - If an agent creates a new, smaller nogood based on communicated nogoods, communicated nogoods can be throw away (subsumption).
  - We can guarantee the completeness using only polynomial space for each agent.

Phenomenal/Operational Analysis of ABT

- Operationally, each agent performs only: backtracking.
- Phenomenological, each assignment is followed by immediate adjustment of connected future domains: forward checking.

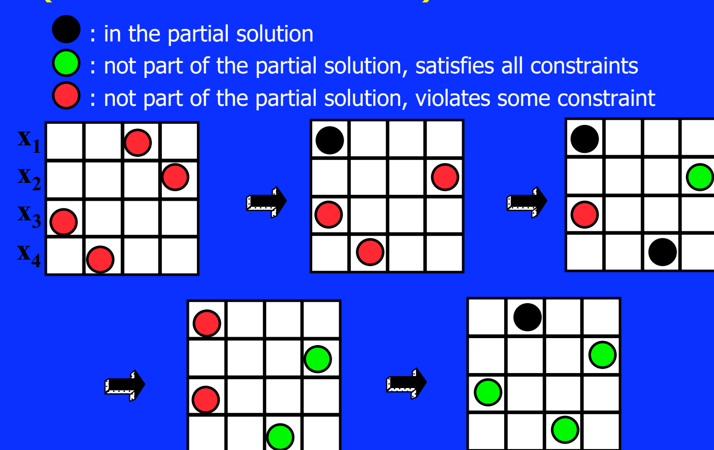


3.2 Weak-commitment Search [Yokoo *94*]

- Each variable has a tentative initial value.
- A consistent partial solution is constructed.
- When no consistent value exists for a variable with the partial solution, the ***whole partial solution is abandoned***.
- The ***search process is restarted*** using the current assignment of variable values as new tentative initial values.
- The abandoned partial solution is ***recorded as a new nogood***.

☛ ***complete, can be efficient***

Example of Algorithm Execution (weak-commitment)



Asynchronous Weak-commitment Search [Yokoo *95*]

Main cause of inefficiency of asynchronous backtracking:

- Convergence to a solution becomes slow when the decisions of higher priority agents are poor; the decisions cannot be revised without an exhaustive search.

Remedy:

- introduce dynamic change of the priority order, so that agents can revise poor decisions without an exhaustive search:
  - ☞ If an agent becomes a dead-end situation, the priority of the dead-end agent becomes higher.

Dynamically Changing Priority Order

- Define a non-negative integer value (***priority value***) representing the priority order of a variable/agent.
  - A variable/agent with a larger priority value has higher priority.
- Ties are broken using alphabetical order.
- Initial priority values are 0.
- The priority value of a ***dead-end agent is changed*** to *max+1*, where *max* is the largest priority value of related agents.

Completeness of AWC

- The priority order is changed if and only if a new nogood is found.
  - The priority order cannot be changed infinitely.
  - If the priority order is stable, this algorithm is equivalent to the asynchronous backtracking.
- The worst-case space complexity is exponential in *n*.
  - can be restricted in practice: forgetting older nogoods
  - If the number of recorded nogoods is sufficiently large, an infinite processing loop rarely occurs.

3.3 ADOPT [ModTamShYok02]

Constraint optimization requires comparisons of the quality of different solutions.

How can we achieve these comparisons, if the information about the solution is distributed over the agents and if the whole system represents at each moment at best one solution candidate?

☞ Adopting ABT to deal with quality of subsolutions information

ADOPT (II)

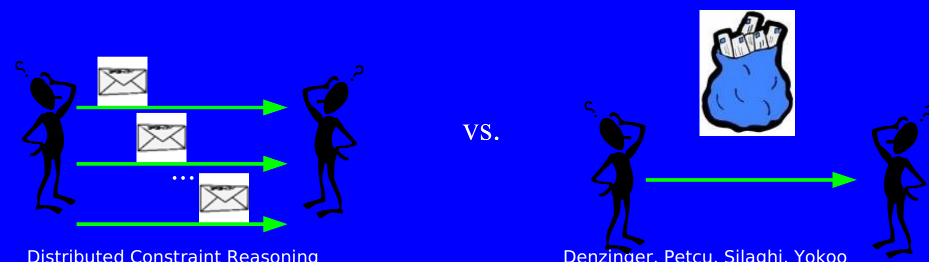
- Order agents in a tree hierarchy
- Higher level agents assume values, propagate them to lower level agents that propagate back lower bound on solutions achievable with values of higher-level agents
- Backtrack (via nogood) when lower bounds get too high (not when quality of best solution of subproblem is determined
  - ☞ requires threshold value that an agent gets from higher level agent

ADOPT: Problems and solutions

- How to order agents?
  - ☞ between levels: there is a constraint between the two agents
  - ☞ within level: no constraints between agents allowed
- Backtracking may lead to flip-floping which leads to need for storing/recomputing previous solutions!
  - ☞ Thresholds limit this: only flip-flop if over threshold (only then store a nogood representing previous subsolution and its cost)

PART 3: Bottom-up Approaches

DPOP: Dynamic Programming Optimization Protocol

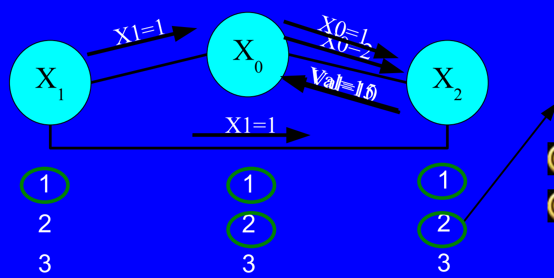


Overview of Part 3

- Dynamic Programming vs. Search
- DPOP algorithm
- DPOP evaluation
- DPOP extensions

Distributed Solving 1: backtrack search

- Assumes an ordering of the variables
- Variables are instantiated sequentially
- After (complete) assignment, backtrack



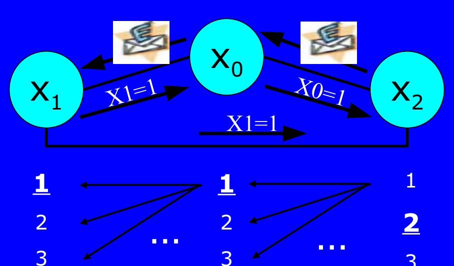
The utility of this assignment

Linear sized messages

Exponential # of messages

Distributed Solving 2: dynamic programming

- Assumes an ordering of the variables
- Incrementally compute all partial solutions; when they are complete, pick the best one



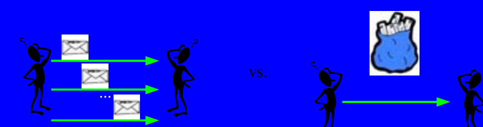
The whole set of utilities one for each combination (exponentially many)

Exponential message size

Linear # of messages

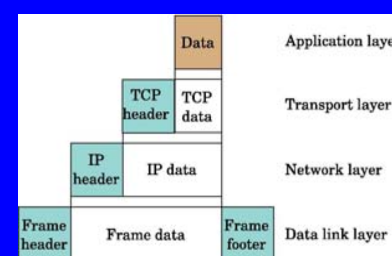
Dynamic Programming vs. Search

- DP sends a linear number of messages
- … but message size is exponential
- Search (e.g. ABT, ADOPT) sends linear size messages
- … but number of sequential messages is exponential in depth (> width)
- Messages have large overhead (packet/e-mail)



Why not many small messages

- Example: X_1 changes its value: $X_1=1$
  - Local assignment (let's say it's free)
  - Inform neighbors (once for each one):
    - *Identify recipient (DNS lookup)*
    - *Open connection with recipient*
    - Send data packet (X_1's value) – typical packet size: 100's to 1000's bytes (for just a single value: X_1=1!!)
    - *Acknowledgment*
    - *Close connection*
- Problems:
  - Important overheads
  - Large latency



DPOP: distributed dynamic programming

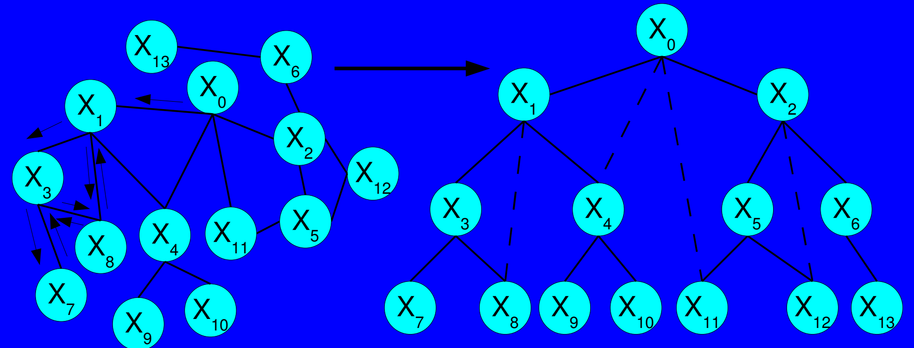
[Petcu&Faltings @ IJCAI'05]

- Non-serial dynamic programming [Bertele&Brioschi'72] Bucket Elimination [Dechter'99], BTE [Shenoy'97] [Kask et al '04]
- Complete algorithm, linear # of messages:

DPOP: distributed dynamic programming

[Petcu&Faltings @ IJCAI'05]

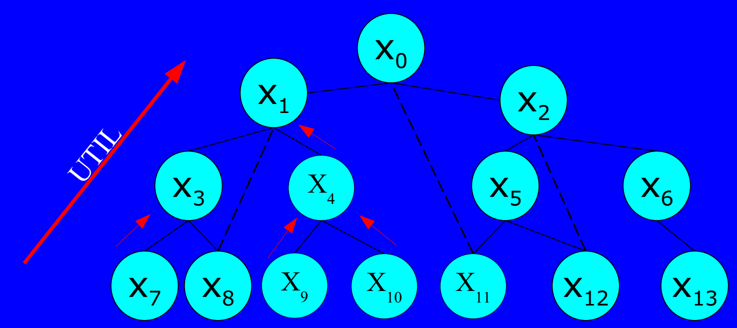
- Complete algorithm, linear # of messages:
  - 1: **DFS:** distributed depth first traversal



DPOP: distributed dynamic programming

[Petcu&Faltings @ IJCAI'05]

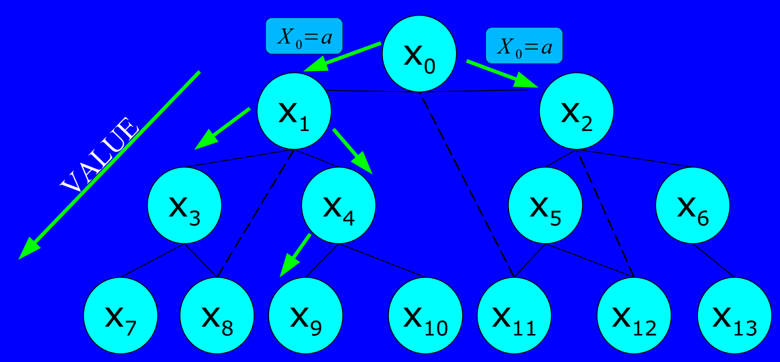
- Complete algorithm, linear # of messages:
  - 1: **DFS:** distributed depth first traversal
  - 2: **UTIL** messages bottom-up on the DFS



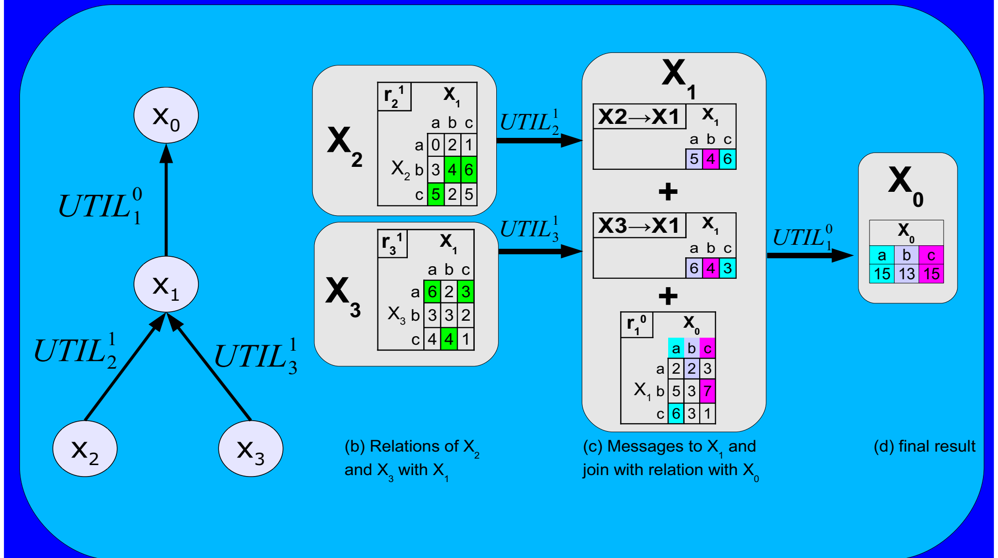
DPOP: distributed dynamic programming

[Petcu&Faltings @ IJCAI'05]

- Complete algorithm, linear # of messages:
  - 1: **DFS:** distributed depth first traversal
  - 2: **UTIL** messages bottom-up on the DFS
  - 3: VALUE assignments go top-down



DPOP: an example

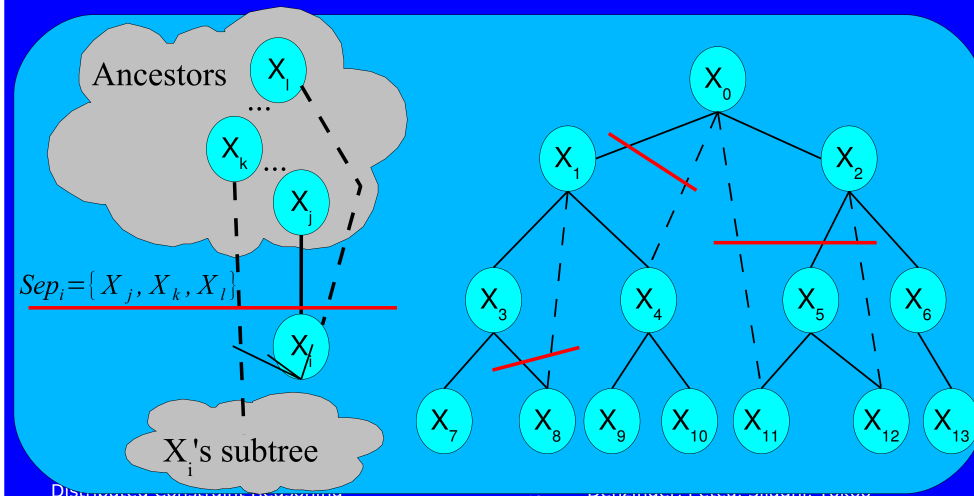


DPOP complexity

- Number of messages: linear in # of agents
- Complexity = size of messages and memory

DPOP complexity: separators

- Separator of X_i: ancestors linked to X_i or its descendants
- Size of messages: exponential in separator size



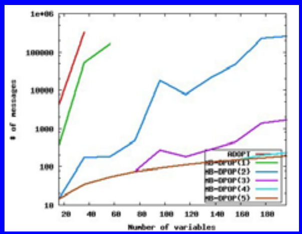
DPOP complexity

- Number of messages: linear in # of agents
- Size of messages and memory:
  - exponential in separator size
  - largest separator size = **induced width**
  - space is O(exp(width))

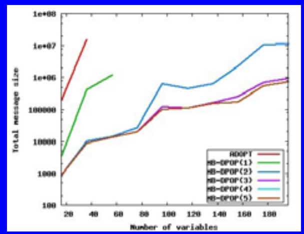
Experiments on meeting scheduling

- Problems with 10-100 agents, resulting in 16-196 variables
- ADOPT (search) vs. MB-DPOP (versions of DPOP)

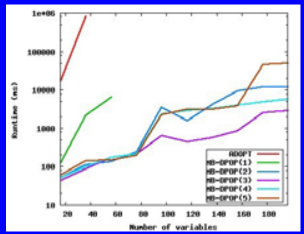
Number of messages



Total information sent



Runtime (ms)



Problem size: from 16 to 196 variables

- Adopt (in red) does not scale beyond 40 variables
- **Up to 100,000 times less messages and information exchange**

DPOP: distributed dynamic programming

[Petcu&Faltings @ IJCAI'05]

- **Complete algorithm**
- **Linear # of messages** (little overhead!)
- **Predictable**: before execution, each node knows:
  - Memory needed
  - Communication requirements
  - Computational power needed
- Works well on sparse problems
- Limitation: message size = exp(induced width)

DPOP vs ADOPT

	DPOP	ADOPT
Complete	YES	YES
No. msgs	linear	exponential
Memory	exp in width	linear
Predictable memory	YES (exp in width)	YES (linear)
Predictable computation	YES (exp in width)	no
Predictable communication	YES (exp in width)	no
Strength	linear no. msgs	linear memory
Recommended if:	low width problems	high width and very loose/tight problems

DPOP extensions for efficiency

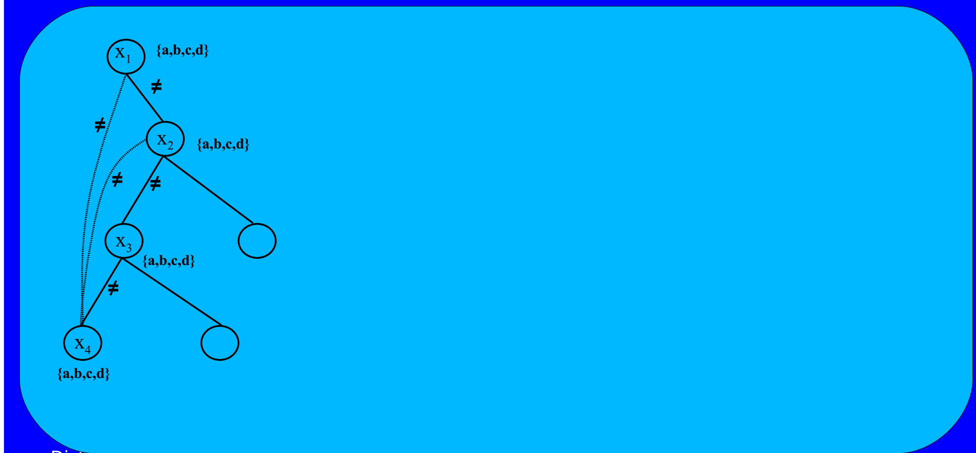
- Complete:
  - H-DPOP: pruning with CDDs [Kumar,Petcu, Faltings, DCR'07]
  - MB-DPOP: memory-bounded [Petcu and Faltings, IJCAI'07]
  - O-DPOP: incremental preference elicitation [P&F AAAI'06]
- Incomplete:
  - A-DPOP: bounded-error approximations [P&F, IAT'07]
  - LS-DPOP: local search hybrid [Petcu and Faltings, IAT'07]
- PC-DPOP: partial centralization [P&F, Mailler, IJCAI'07]

H-DPOP: pruning in dynamic programming

- Hard constraints rule out certain combinations
- DPOP still sends them, thus wasting space

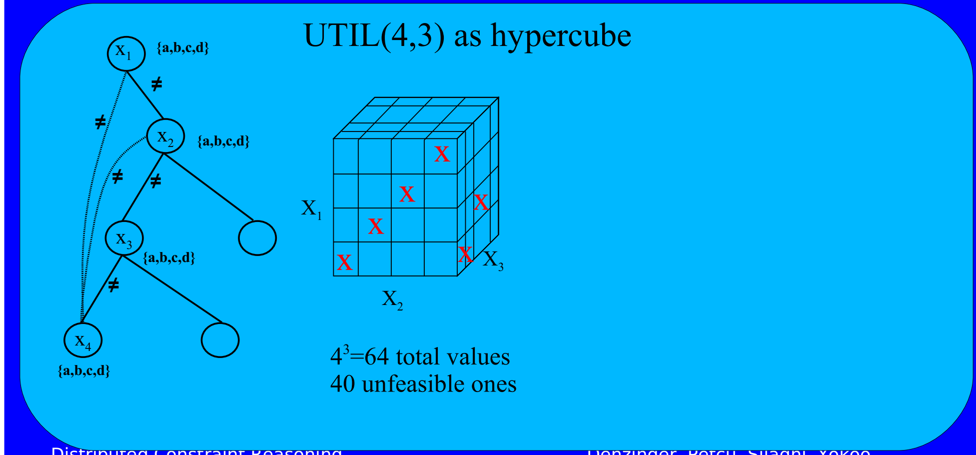
H-DPOP: pruning in dynamic programming

- Hard constraints rule out certain combinations
- DPOP still sends them, thus wasting space



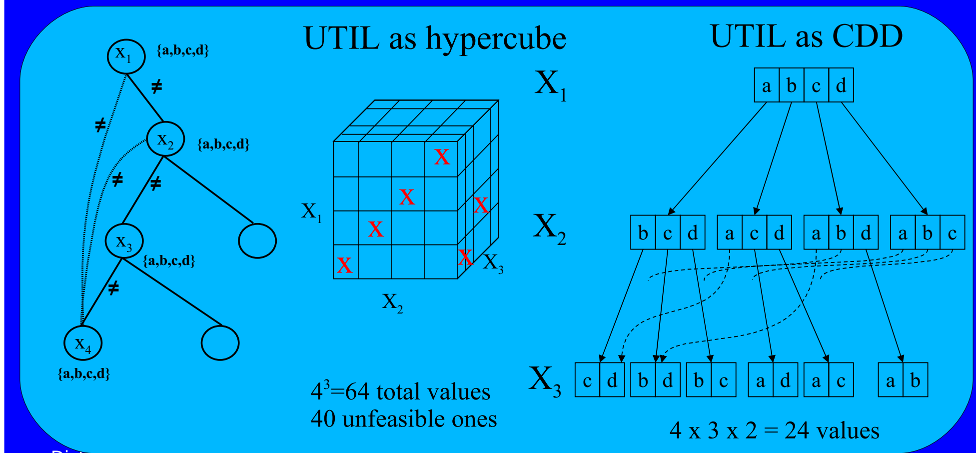
H-DPOP: pruning in dynamic programming

- Hard constraints rule out certain combinations
- DPOP still sends them, thus wasting space



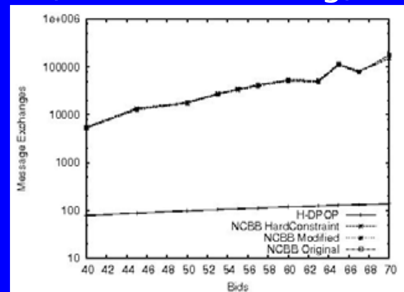
H-DPOP: pruning in dynamic programming

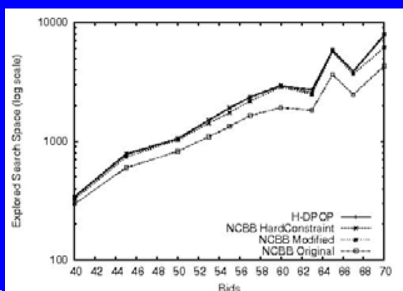
- Constraint Decision Diagrams [Cheng, Yap '05]
- H-DPOP uses CDDs to compact UTIL messages

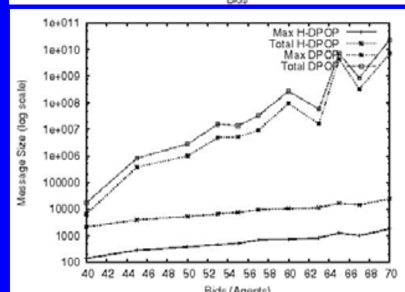


H-DPOP: experimental results

- Comparison w/ DPOP and NCBB (search+caching):
  - few messages, as DPOP
  - effective pruning, as NCBB



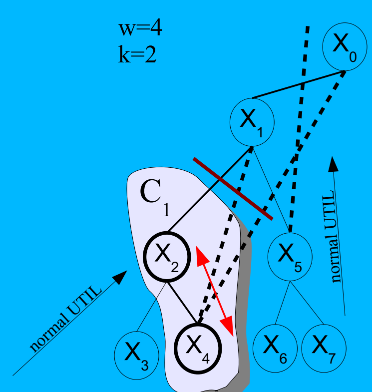




MB-DPOP: memory bounded DPOP

- Uses cycle cuts and bounded propagations
- Tradeoff memory vs number of messages

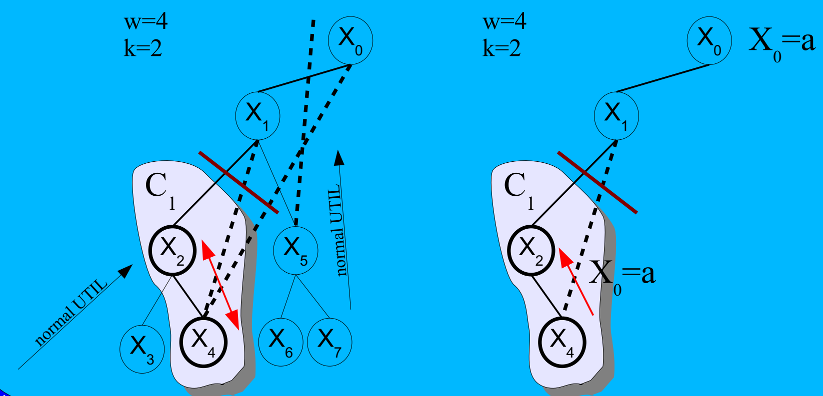
DFS arrangement, and a cluster of width>2



MB-DPOP: memory bounded DPOP

- Uses cycle cuts and bounded propagations
- Tradeoff memory vs number of messages

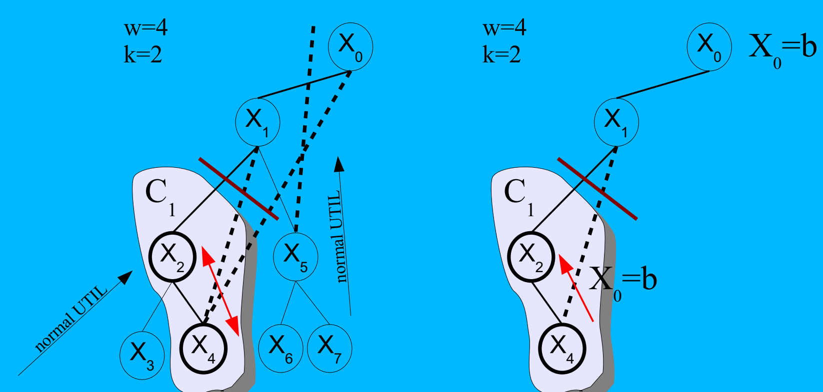
DFS arrangement, and a cluster of width>2



MB-DPOP: memory bounded DPOP

- Uses cycle cuts and bounded propagations
- Tradeoff memory vs number of messages

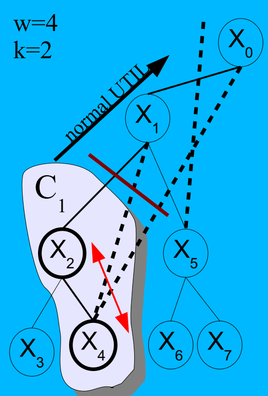
DFS arrangement, and a cluster of width>2



MB-DPOP: memory bounded DPOP

- Uses cycle cuts and bounded propagations
- Tradeoff memory vs number of messages

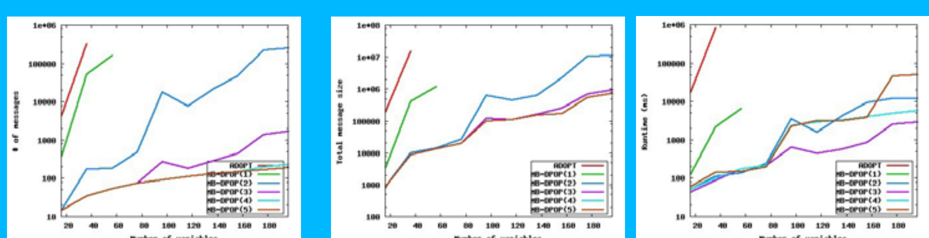
DFS arrangement, and a cluster of width>2



MB-DPOP: experimental results

- Compared against ADOPT on MS, GC, SN probs
- Outperforms ADOPT by up to 5 orders of mag

Number of messages | Total information sent | Runtime (ms)



Problem size: from 16 to 196 variables

- Adopt (in red) does not scale beyond 40 variables
- **Up to 100,000 times less messages and information exchange**

Wrap-up part 3

- Dynamic Programming (DPOP & co):
  - performs well for problems of low width
  - where applicable, orders of magnitude less overhead than search
  - predictable execution - each agent knows in advance:
    - computation
    - memory
    - communication
  - extensions for efficiency:
    - complete: H-DPOP, MB-DPOP, O-DPOP
    - incomplete: LS-DPOP, A-DPOP

4. Privacy for DCR

- Why are Distributed Constraint Problems distributed?
  - Because of the size of the problem?
  - Because of dynamic changes and openness?
  - Because of privacy?

Motivation for distribution (strength)

- Why are Distributed Constraint Problems distributed?
  - Because of the size of the problem?
    - If the problem is too large, communication in solvers may also be too expensive.
  - Because of dynamic changes and openness?
    - Announcing dynamic changes to a centralizing server may be faster than propagating changes through many agents.
  - Because of privacy?
    - It is agreed as the main reason for DCR

Object of privacy in DCR (approaches)

- The existence of a variable (internal to an agent)
  - communicate only about inter-agent constraints
- The existence of a constraint (between known var.)
  - declare that all constraints are present (costly)
  - shuffle variables (identity of variables is hidden)
- The feasibility of a tuple in a constraint (finite, ∞)
  - Scarce communication, aggregation with other values, shared encryption
- The lack of solutions in an area of the search space

Object of privacy in DCR

- The existence of a variable (internal to an agent)
- The existence of a constraint (between known var.)
- The value of a tuple in a constraint (finite, ∞)
- The lack of solutions in an area of the search space

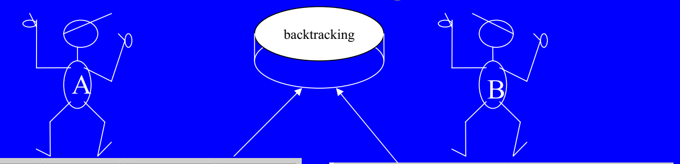
Privacy Properties

- Distributed Constraint-based Algorithms allow problems to be solved without necessarily revealing all the details of the constraints [FreMinWal 01,SilFal 02,WalSil 04]
- The alternative is:
  - Using a trusted server where the problem is centralized and solved (often more efficient, but less acceptable security)
- Newer approaches are based on cryptographic techniques
  - Using a set of untrusted/semi-trusted servers that together simulate the server (less efficient but more secure) [YokSuzHir 02]
  - Using threshold cryptographic schemes. Even less efficient. No server needed [Silaghi 03a, Silaghi04]

Privacy in constructive search

- Reveal only a bit of information at the time, by choosing the assignments/nogoods that reveal the least sensitive data.
- Exchanged information is aggregated with other data known by the agent, further precluding leaks
- Each communication is only between 2 agents
- Asynchronous schemes allow agents to:
  - Delay answering a sensitive question hoping for new information that precludes the need for answering.

Solution and orderings [AAMAS'04]



place/time	July	August	September
NY	☑	☑	☐
Melbourne	☑	☑	☑
Lausanne	☐	☑	☑

place/time	July	August	September
NY	☐	☐	☑
Melbourne	☐	☑	☐
Lausanne	☐	☑	☑

You know:
- the algorithm: backtracking
- the variables order: place, time
- the order on values
- the result: Melbourne, August

What can Alice infer about the constraints of Bob?

What if values are reordered?

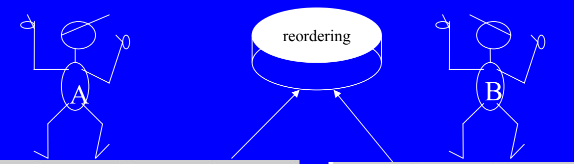
Spying on other's constraints

- Statistic learning
  - Agents can infer bounds on tuple values from the constraints of others, using so called "shadow CSPs"
- Exact learning
  - If the data in messages of an agent are not result of aggregation (e.g. nogood/cost messages from a terminal agent), then secrets are harvested easily
    - Solution: The terminal agent can be an abstract cryptographic computation
  - The solution can reveal secrets. Solution: shuffling

Absence of Solution (Silaghi 05b)

- If there is no solution to the problem then everybody learns that all sets of assignments are unsatisfiable (a large leak of secret information that may be useless).
  - One may prefer to have an answer "don't know" rather that "unsatisfiable."
- An answer "don't know" implies that solutions may exist but are lost.
  - This can be achieved if the solver can forget the solution with some probability *p* (before revealing the answer).

The effect of policies [AAMAS'04]



place/time	July	August	September
NY	☑	☑	☐
Melbourne	☑	☑	☑
Lausanne	☐	☑	☑

place/time	July	August	September
NY	☐	☐	☑
Melbourne	☐	☑	☐
Lausanne	☐	☑	☑

You know:
- the algorithm: backtracking
- the variables order: place, time
- P(Mel, Aug)>P(Lau,Aug)

Can Alice infer something?

Call it:
Non-uniform requested t-privacy!

Bob: #(Melbourne)<#(Lausanne)

What if variables are reordered?
(Lau,Sep) -> probably not (Mel,Sep)
What if all tuples are shuffled?

Can obtain *requested t-privacy*!

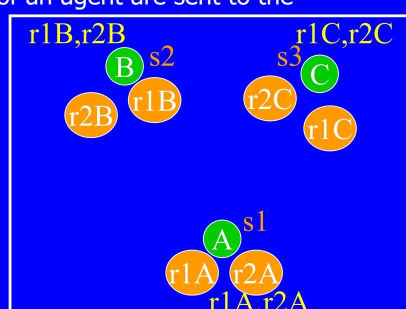
Secure evaluation of arithmetic circuits

- Arithmetic Circuit: Any function whose definition consists only of a fix set of additions and multiplications.
- It was shown that efficient computations of comparison, max, ∂ are possible this way.
- The output of any arithmetic circuit with secret inputs can be computed securely by running for each of these operations a corresponding protocol.

Multi-party Computation of a sum of three secret numbers

- A has s1, B has s2, C has s3
- Each of A,B,C computes two random numbers: (r1A, r2A), (r1B, r2B), (r1C, r2C)
- The computed random numbers of an agent are sent to the other two participants

A:(s1,r2B,r1C) B:(s2,r1A,r2C) C:(s3,r2A,r1B)



- A: kA=r2B+r1C+(s1-r1A-r2A)
- B: kB=r1A+r2C+(s2-r1B-r2B)
- C: kC=r2A+r1B+(s3-r1C-r2C)
- kA+kB+kC=s1+s2+s3

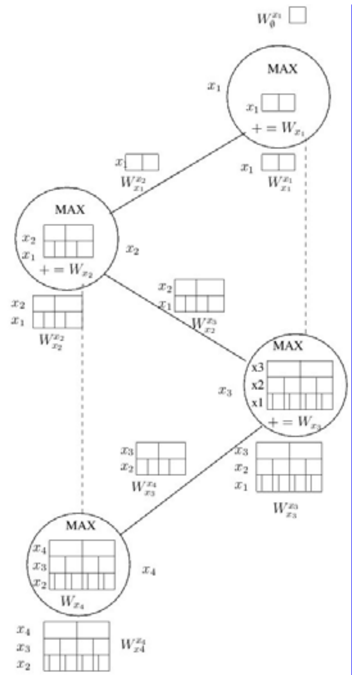
Distributed Constraint Optimization --- Application with Privacy [AiMath06] ---

- Auctions (e.g., Generalized Vickrey Auctions)
  - Allocation is computed with the formula: $\arg\max_{A=\{a_1,\ldots,a_N\}} \sum_{k=1}^{N} u_k(a_k)$
  - Prices are computed with (if A^i, a_k^i stand for solution without bidder A_i)
    - *for all i:* $p_i = \left(\max_{A^i=\{a_1^i,\ldots,a_N^i\}} \sum_{k\neq i} u_k(a_k^i) \right) - \sum_{k\neq i} u_k(a_k)$
  - Therefore, N+1 optimizations are needed, as intermediary steps
    - *$argmax_A(\Sigma_k u_k(a_k))$, $max_{A^i}\Sigma_k u_k(a^i_k)$*
  - The results of these intermediary computations should be secret, and only final results, p_i, a_i, should be known to A_i.
  - Still everybody should be convinced that the computation was correct given bidder's inputs.
- Distributed Meeting Scheduling [Wallace, Tambe,…]

Arithmetic Circuit for Upward DPOP

procedure *Upward*(x_i) **do**
- **foreach** ($y \in S_{x_i}$) **do**
  - // recursive call for each child;
  - *Upward*(y);
- **foreach** *tuple* $\varepsilon \in \Gamma_{G_{x_i} \cup \{x_i\}}$ **do**
  - // compose costs for the same tuple;
  - $W_{x_i}[\varepsilon] = W_{x_i}^{x_i}[\varepsilon_{|\{x_i\} \cup P_{x_i}}] + \sum_{y \in S_{x_i}} (W_{x_i}^{y}[\varepsilon_{|G_y}]$;
- **foreach** *tuple* $\varepsilon \in \Gamma_{G_{x_i}}$ **do**
  - // project composed constraint onto induced parents;
  - $W_{F_{x_i}}^{x_i}[\varepsilon] = \max_{v \in D_i}(W_{x_i}[\varepsilon \cup \langle x_i, v \rangle])$;

S_x – children of x
G_x – induced parents of x
P_x – neighbor ancestor of x
F_x – parent of x
W_x^y – constraint of x coming from y
Γ – search space



procedure *Downward* (*COP, W,* $V_{root}=W_0^{root}$) ***do***
- **foreach** *variable* x_i *in the COP* **do**
  - $\zeta_{x_i}[\emptyset] = 1$;
- **while** $x_i \leftarrow$ *get_In_PreOrder_Next(DFS(COP))* **do**
  - **foreach** $v \in D_i$ **do** // set $d_{xi}[v]=1$ for the first right v
    - $h_{x_i}[v] = 1 - \sum_{k=1}^{v-1} d_{x_i}[k]$; // remove 2nd and later solutions
    - $d_{x_i}[v] = h_{x_i}[v] \sum_{\varepsilon \in \Gamma_{G_{x_i} \cup \{x_i\}}, \varepsilon_{|\{x_i\}}=v} \zeta_{x_i}[\varepsilon_{|G_{x_i}}] \delta_K(V_{x_i}, W_{x_i}[\varepsilon])$;
  - **foreach** $y \in S_{x_i}$ **do**
    - // Compute $\zeta_y[\varepsilon]$ with formula 1 or 2
    - **if** *cost*($|G_y| - 2$ *multiplications*) $>$ *cost*($d^{|G_{x_i} \setminus G_y|}$ *additions*) **then**
      - **foreach** $\varepsilon \in \Gamma_{G_y}$ **do** // computing with a recurrence formula
        - $\zeta_y[\varepsilon] = d_{x_i}[\varepsilon_{|\{x_i\}}]((G_{x_i} == \emptyset)?1 : \sum_{\varepsilon' \in \Gamma_{G_{x_i}}, \varepsilon'_{|G_y}=\varepsilon} \zeta_{x_i}[\varepsilon'])$;
    - **else**
      - **foreach** $\varepsilon \in \Gamma_{G_y}$ **do** // computing with the basic formula
        - $\zeta_y[\varepsilon] = \prod_{p \in G_y} d_p[\varepsilon_{|p}]$;
    - $V_y = \sum_{\varepsilon \in \Gamma_{G_y}} \zeta_y[\varepsilon] W_{x_i}^{y}[\varepsilon]$;

$d_x[v]$ – is $x=v$?; V_x – weight of subtree with root x; $\zeta_x[\varepsilon]$ – is ε selected?

Secure DPOP

- The three mentioned problems are fixed by:
  - Shuffling and un-shuffling domains prior to solving
    - (to guarantee that each optimal solution has a chance)
  - Design an arithmetic circuit doing the decoding but avoiding to reveal the assignments.
    - Select values by assigning secret variables (one per value)
      - 2 alternative formulas → select the cheapest in each node!
    - Have this values under form of unary constraints to allow un-shuffling

    Therefore, an independent secret value is introduced for each element of the extensive representation in these constraints.

Distributed CSP with Privacy

- **Definition: "A Distributed CSP is defined by a set of participants willing to reach a solution to the CSP (X,D,C), defined by their private problems. The distribution of constraints and the rights to assign variables are defined by the privacy requirements."**
- A constraint c in C is formulated as a function on assignments to variables in X, returning 1 for accepted tuples and 0 for rejected tuples.

  c: D→{0,1}

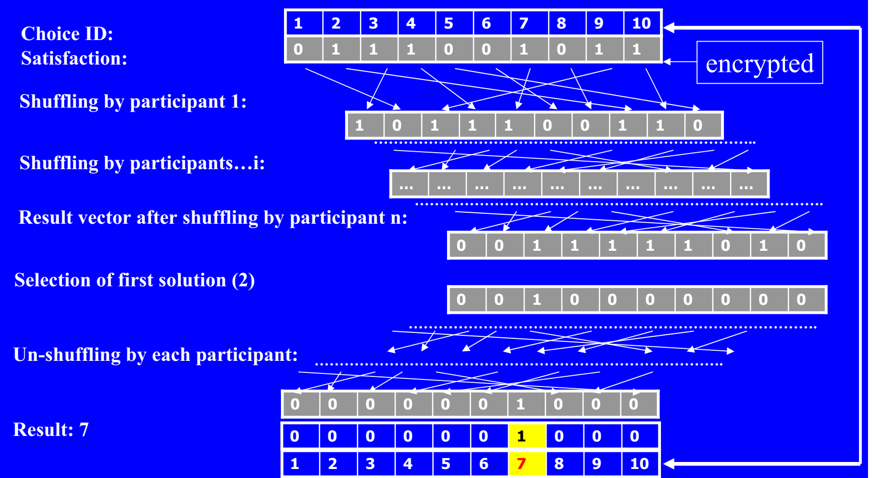
Finding all solutions of a DisCSP

- For each possible tuple *t* of assignment of values to variables, compute securely the acceptability of *t*.

$acceptability(t)=\prod_{c\in C} c(t)$

- The secret value of c in each point is shared by the participant knowing it.
- The product is computed using the BGW computation in section 4.

Uniformly Random (MPC-DisCSP4)



Example finding all meeting possibilities

- Alice, Bob, and Carol want to find all possible meetings,
  - either Monday or Tuesday,
  - either in Melbourne, Orlando, or Tampa
- Distributed CSP A={Alice, Bob, Carol}:
  - X={day,location}
  - D={{M,T},{M,O,T}}
  - C={Alice: c_A={(M,O),(T,M)},
    Bob: c_B={(M,M),(M,O),(T,M),(T,T)},
    Carol: c_C={(M,O),(M,T),(T,M)}}

Possible tuples t are	(M,M)	(M,O)	(M,T)	(T,M)	(T,O)	(T,T)
c_A	0	1	0	1	0	0
c_B	1	1	0	1	0	1
c_C	0	1	1	1	0	0
acceptability(t)=$c_A*c_B*c_C$	0	1	0	1	0	0

Uniformly Random one-Solution finding

1. As in the All-Solutions algorithm, compute the vector "S" with the value of *acceptable*(*t*) for each candidate *t*.
2. Shuffle S[1..K] using the mix-net algorithm (or some equivalent arithmetic circuit [Silaghi'04,'05]).
3. Select one solution in S with:
   h[1]=1;
   for k=2 to K
   1. h[k]=h[k-1]*(1-S[k-1])
   2. S[k]=h[k]*S[k]
4. Un-shuffle S, with the mix-net performing the reverse permutations.
5. Reveal S (or an extraction of the remaining tuple of S)

t-privacy

The following algorithms present *t*-privacy, if (*t+1*,*n*)-sharing schemes are use:

***t*-privacy**. For any group of t participants:

P(Secrets | Answer, Public-Knowledge, Computation)
=P(Secrets | Answer, Public-Knowledge)

For MPC-DisCSP4 [Silaghi'04'05]

P(Secrets | Solution, Public-Knowledge, Algorithm, Computation)
=P(Secrets | Solution, Public-Knowledge)

Non-uniform requested t-privacy

- *non-uniform requested t-privacy* of an algorithm
  - *t-privacy*
  - $P(\Sigma|\alpha^*,\Gamma)<1 \Rightarrow P(\Sigma|\alpha,\Gamma,\Pi,A)<1$
    - A – algorithm
    - α* – requested data
    - σ – secret constraints
    - Σ – an element of σ
    - α – answer
    - Γ – prior knowledge
    - Π – computation process

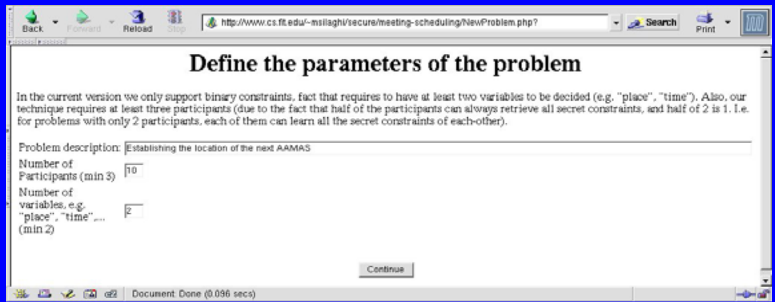
t-privacy

- *t-privacy* for a computation process [BGW'88]. For any group of t participants:
  - $P(\sigma|\alpha,\Gamma,\Pi) = P(\sigma|\alpha,\Gamma)$
    - σ – secret constraints
    - α – answer
    - Γ – prior knowledge
    - Π – computation process
- *requested t-privacy* of an algorithm [Sil DCR05]
  - $P(\sigma|\alpha,\Gamma,\Pi,A) = P(\sigma|\alpha^*,\Gamma)$
    - A – algorithm
    - α* – requested data [$\alpha^*=argmax_A(\Sigma_k u_k(a_k))$, $\underline{\alpha}=max_{A'}\Sigma_k u_k(a'_k)$]

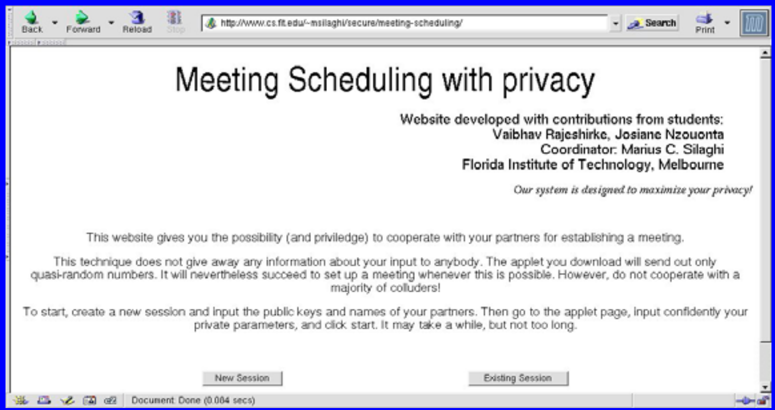
Secure DisCSP Solvers

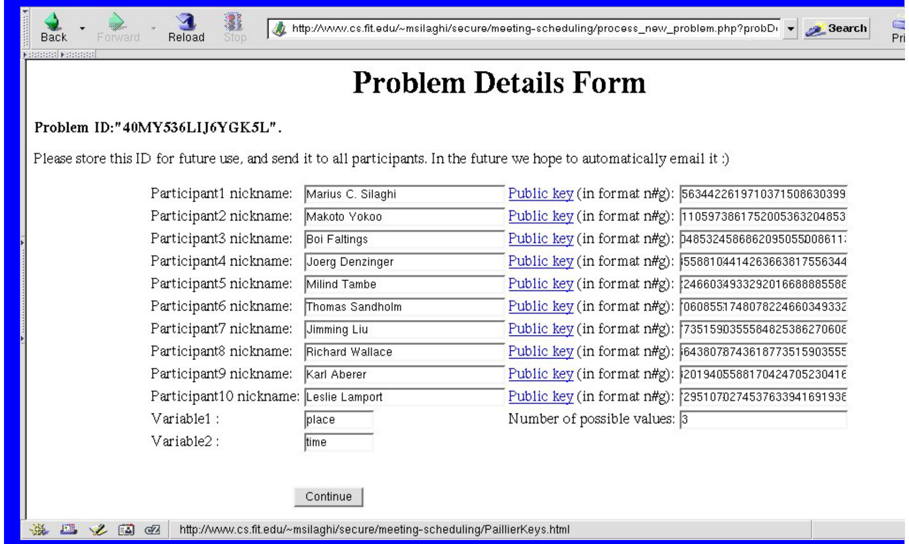
- Principles [SilRaj'04][Sil'05b]
- Finding all solutions [Herlea.et.al'01]
- Secure algorithms guaranteeing t-privacy
  - Minimizing information leaked by solution [Sil'05b]
  - Polynomial space algorithm [Silaghi'02'03]
  - Optimization [SilMit'04]
  - Stochastic search [Sil.et.al'05]
- Secure algorithms with cryptography
  - Complete algorithm [Yokoo.et.al'02'05]
  - Stochastic optimization (Simulated Annealing) [Sil.et.al'05]
- Solvers

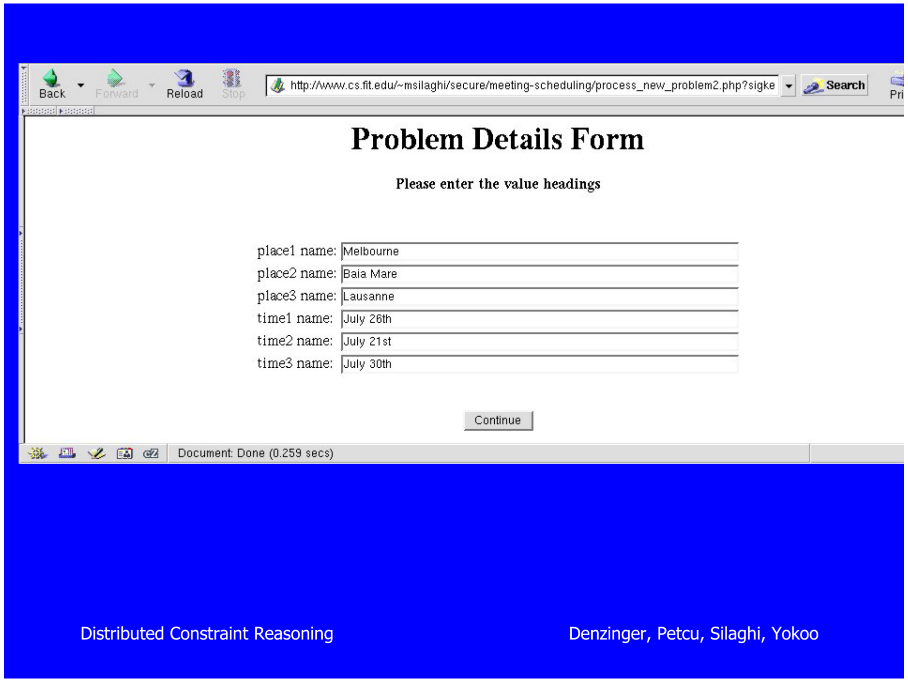
DEMO: System for secure solving of DisCSPs

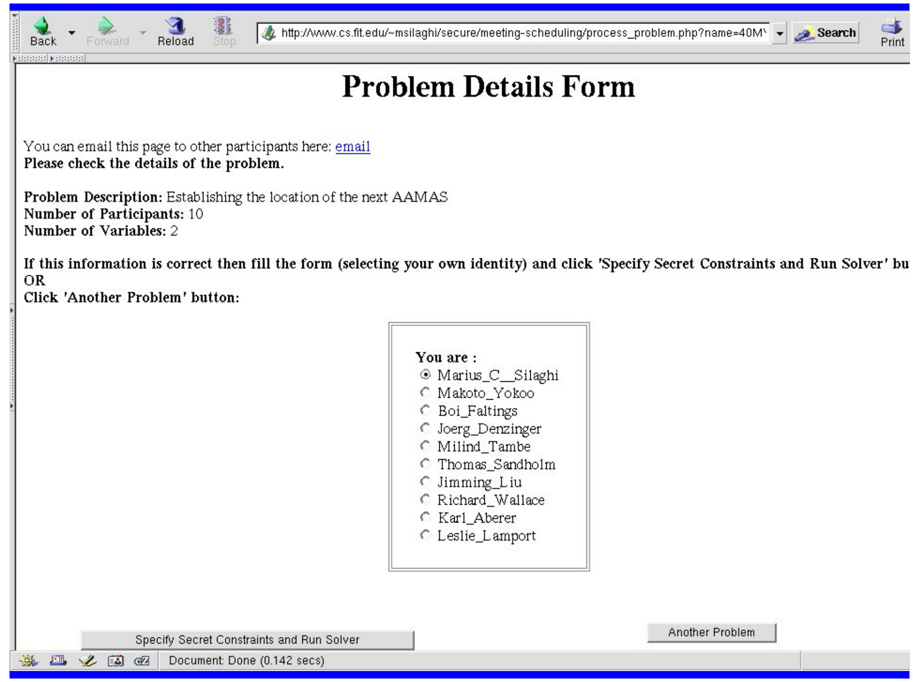


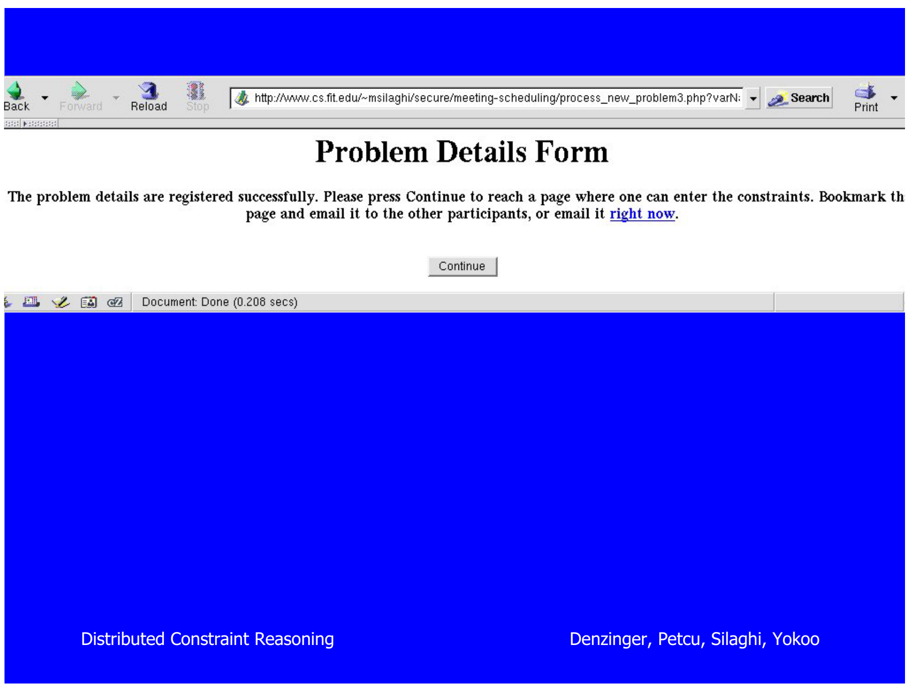
MPC-DisCSP1

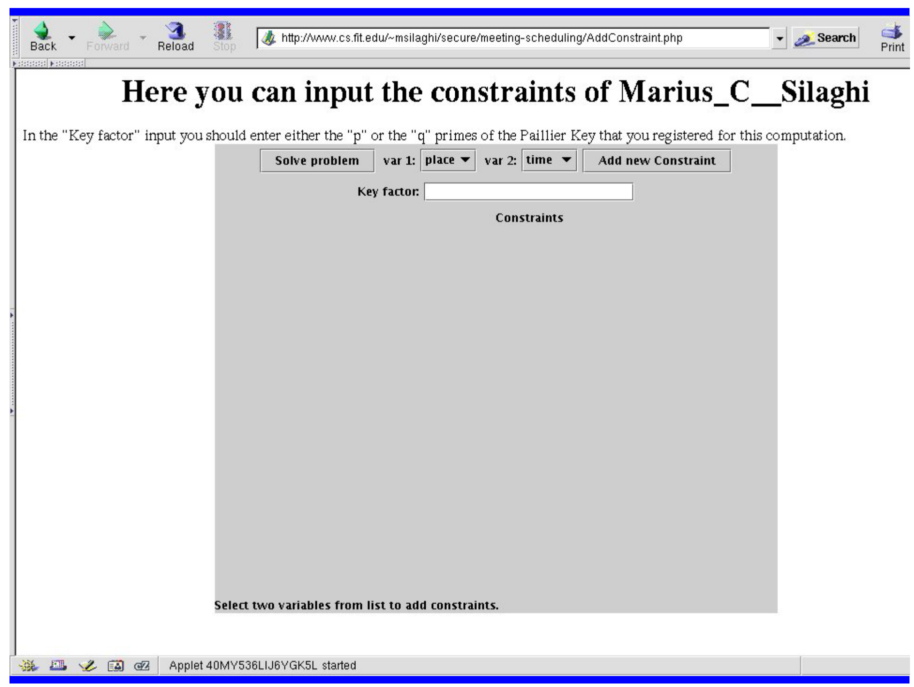


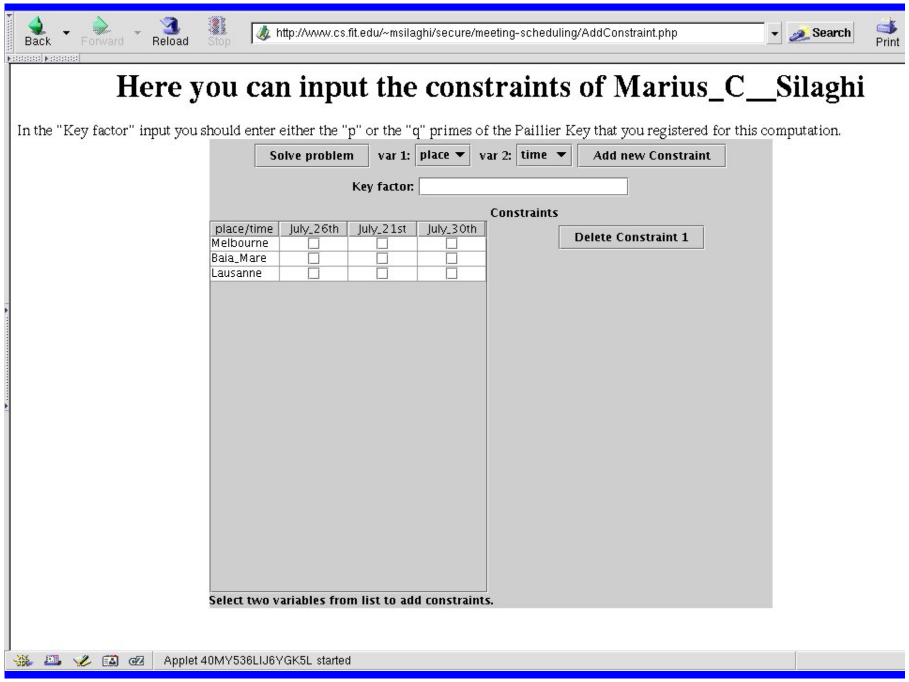


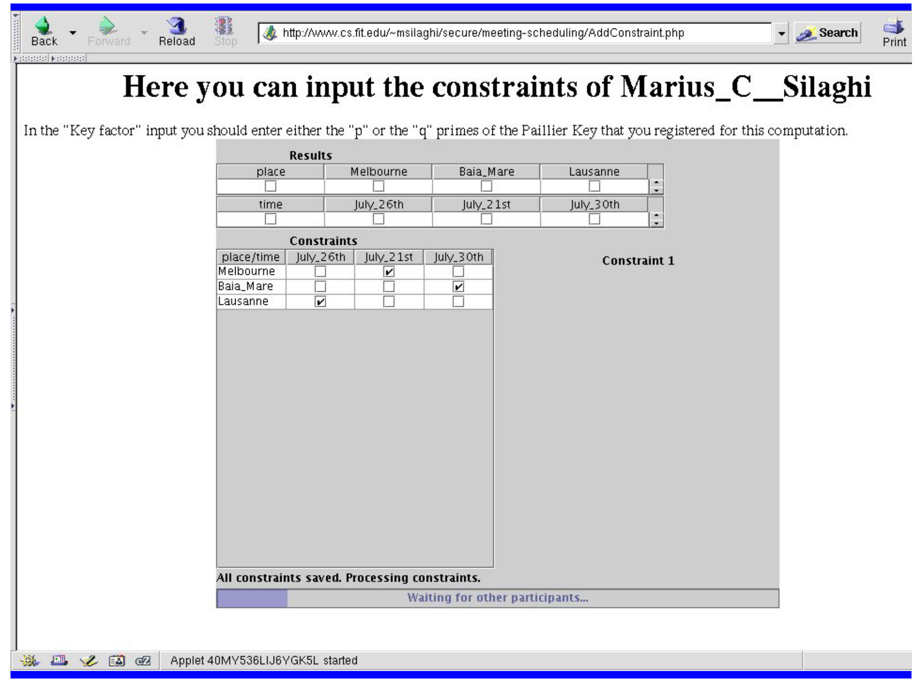

Back
Forward
Reload
Stop
http://www.cs.fit.edu/~msilaghi/secure/meeting-scheduling/process_new_problem2.php?sigke
Search
Problem Details Form
Please enter the value headings
place1 name: Melbourne
place2 name: Baia Mare
place3 name: Lausanne
time1 name: July 26th
time2 name: July 21st
time3 name: July 30th
Continue
Document: Done (0.259 secs)
Distributed Constraint Reasoning
Denzinger, Petcu, Silaghi, Yokoo

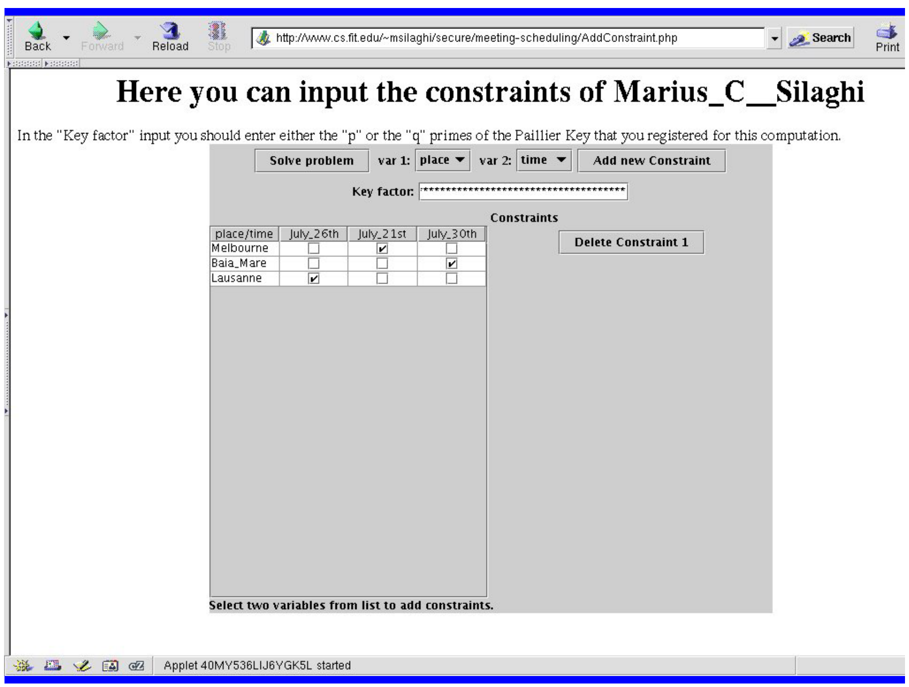

Back
Forward
Reload
Stop
http://www.cs.fit.edu/~msilaghi/secure/meeting-scheduling/process_problem.php?name=40M
Search
Print
Problem Details Form
You can email this page to other participants here: email
Please check the details of the problem.
Problem Description: Establishing the location of the next AAMAS
Number of Participants: 10
Number of Variables: 2
If this information is correct then fill the form (selecting your own identity) and click 'Specify Secret Constraints and Run Solver' bu
OR
Click 'Another Problem' button:
You are :
Marius_C__Silaghi
Makoto_Yokoo
Boi_Faltings
Joerg_Denzinger
Milind_Tambe
Thomas_Sandholm
Jimming_Liu
Richard_Wallace
Karl_Aberer
Leslie_Lamport
Specify Secret Constraints and Run Solver
Another Problem
Document: Done (0.142 secs)

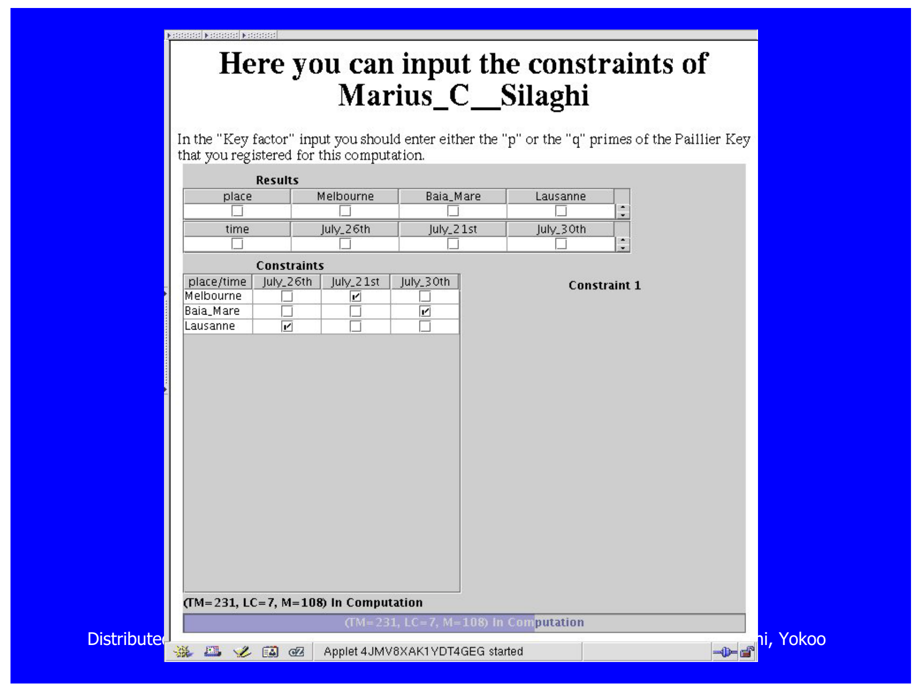

Back
Forward
Reload
Stop
http://www.cs.fit.edu/~msilaghi/secure/meeting-scheduling/process_new_problem3.php?varN
Search
Print
Problem Details Form
The problem details are registered successfully. Please press Continue to reach a page where one can enter the constraints. Bookmark th
page and email it to the other participants, or email it right now.
Continue
Document: Done (0.208 secs)
Distributed Constraint Reasoning
Denzinger, Petcu, Silaghi, Yokoo

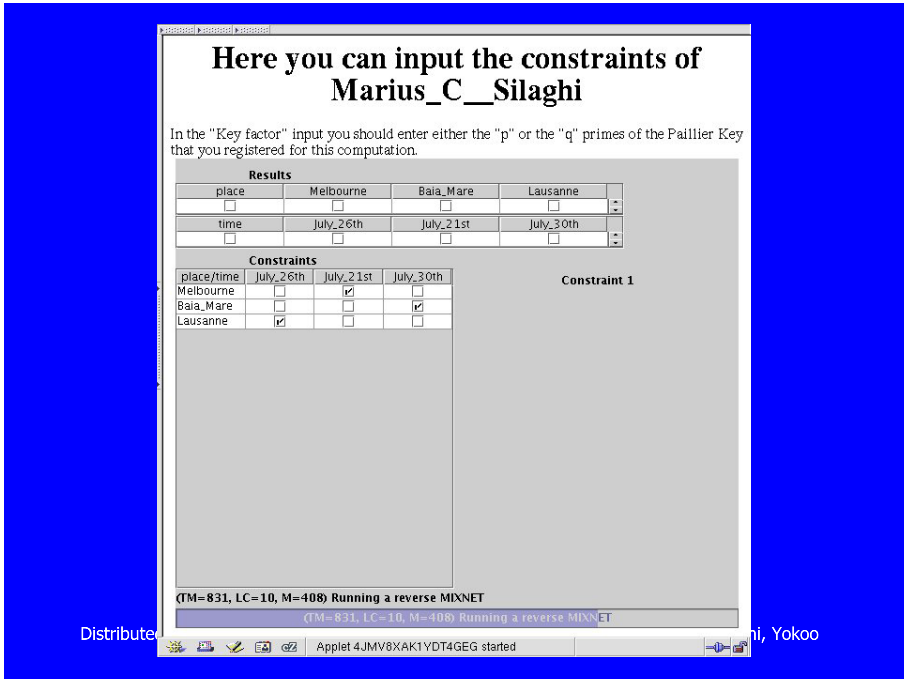

Back
Forward
Reload
Stop
http://www.cs.fit.edu/~msilaghi/secure/meeting-scheduling/AddConstraint.php
Search
Print
Here you can input the constraints of Marius_C__Silaghi
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Solve problem
var 1: place
var 2: time
Add new Constraint
Key factor:
Constraints
Select two variables from list to add constraints.
Applet 40MY536LIJ6YGK5L started

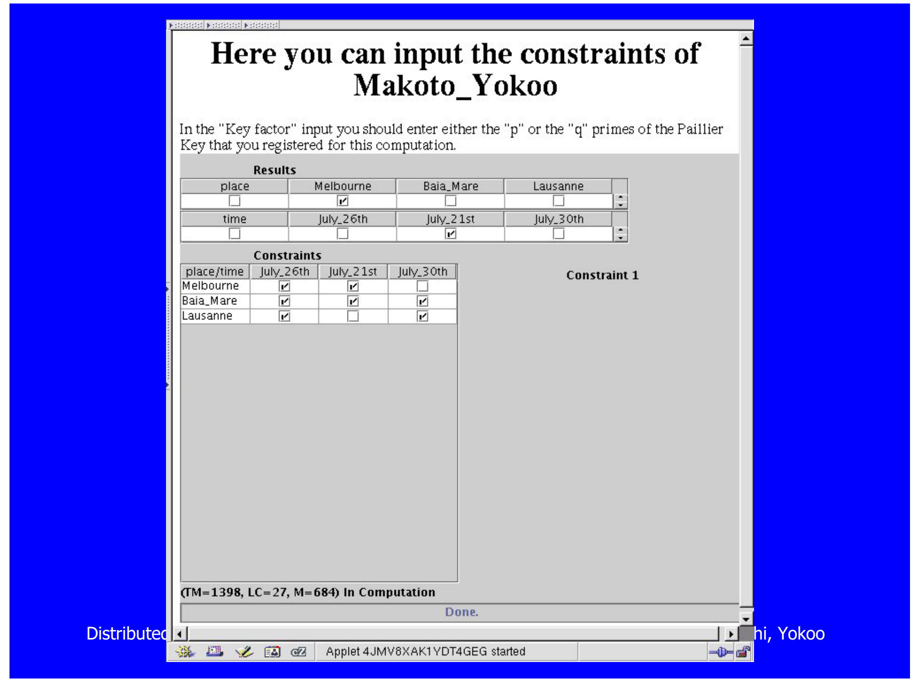

http://www.cs.fit.edu/~msilaghi/secure/meeting-scheduling/AddConstraint.php
Here you can input the constraints of Marius_C__Silaghi
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Solve problem
var 1: place
var 2: time
Add new Constraint
Key factor:
Constraints
Delete Constraint 1
place/time July_26th July_21st July_30th
Melbourne
Baia_Mare
Lausanne
Select two variables from list to add constraints.
Applet 40MY536LIJ6YGK5L started

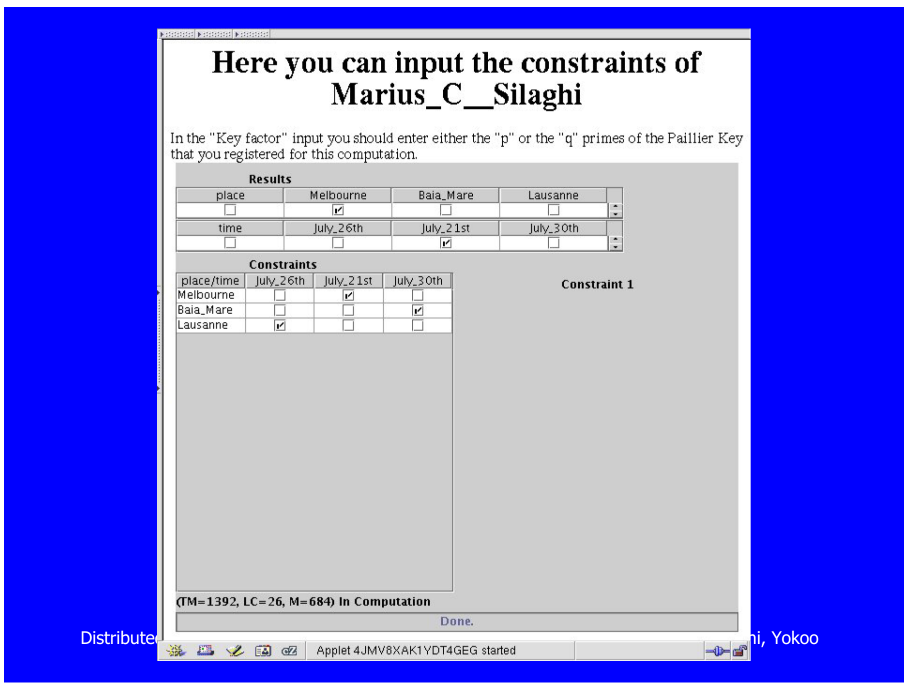

http://www.cs.fit.edu/~msilaghi/secure/meeting-scheduling/AddConstraint.php
Here you can input the constraints of Marius_C__Silaghi
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Results
place Melbourne Baia_Mare Lausanne
time July_26th July_21st July_30th
Constraints
place/time July_26th July_21st July_30th
Melbourne
Baia_Mare
Lausanne
Constraint 1
All constraints saved. Processing constraints.
Waiting for other participants...
Applet 40MY536LIJ6YGK5L started

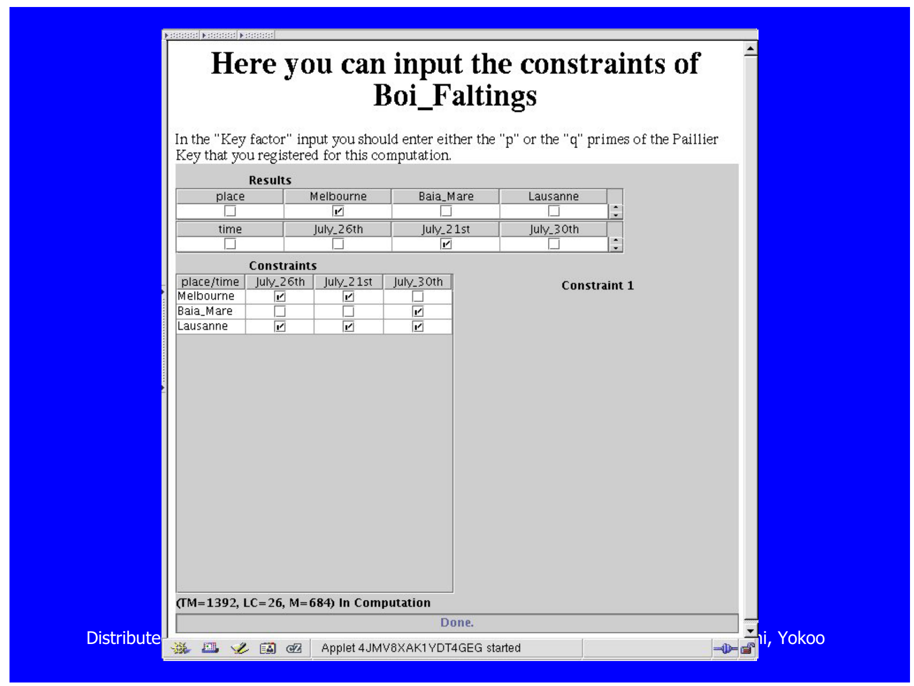

http://www.cs.fit.edu/~msilaghi/secure/meeting-scheduling/AddConstraint.php
Here you can input the constraints of Marius_C__Silaghi
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Solve problem
var 1: place
var 2: time
Add new Constraint
Key factor:
Constraints
Delete Constraint 1
place/time July_26th July_21st July_30th
Melbourne
Baia_Mare
Lausanne
Select two variables from list to add constraints.
Applet 40MY536LIJ6YGK5L started


Here you can input the constraints of Marius_C__Silaghi
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Results
place Melbourne Baia_Mare Lausanne
time July_26th July_21st July_30th
Constraints
place/time July_26th July_21st July_30th
Melbourne
Baia_Mare
Lausanne
Constraint 1
(TM=231, LC=7, M=108) In Computation
(TM=231, LC=7, M=108) In Computation
Applet 4JMV8XAK1YDT4GEG started
Distribute
i, Yokoo


Here you can input the constraints of Marius_C__Silaghi
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Results
place
Melbourne
Baia_Mare
Lausanne
time
July_26th
July_21st
July_30th
Constraints
place/time
Melbourne
Baia_Mare
Lausanne
Constraint 1
(TM=831, LC=10, M=408) Running a reverse MIXNET
Applet 4JMV8XAK1YDT4GEG started
Distribute
i, Yokoo


Here you can input the constraints of Makoto_Yokoo
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Results
place
Melbourne
Baia_Mare
Lausanne
time
July_26th
July_21st
July_30th
Constraints
place/time
Melbourne
Baia_Mare
Lausanne
Constraint 1
(TM=1398, LC=27, M=684) In Computation
Done.
Applet 4JMV8XAK1YDT4GEG started
Distributed
hi, Yokoo


Here you can input the constraints of Marius_C__Silaghi
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Results
place
Melbourne
Baia_Mare
Lausanne
time
July_26th
July_21st
July_30th
Constraints
place/time
Melbourne
Baia_Mare
Lausanne
Constraint 1
(TM=1392, LC=26, M=684) In Computation
Done.
Applet 4JMV8XAK1YDT4GEG started
Distribute
i, Yokoo


Here you can input the constraints of Boi_Faltings
In the "Key factor" input you should enter either the "p" or the "q" primes of the Paillier Key that you registered for this computation.
Results
place
Melbourne
Baia_Mare
Lausanne
time
July_26th
July_21st
July_30th
Constraints
place/time
Melbourne
Baia_Mare
Lausanne
Constraint 1
(TM=1392, LC=26, M=684) In Computation
Done.
Applet 4JMV8XAK1YDT4GEG started
Distribute
i, Yokoo

Wrap up

- Introduction to DCOP
- Search algorithms (top-down)
- Dynamic Programming algorithms (bottom-up)
- Privacy techniques
- Annex contains more:
  - incentive-compatibility and self-interest
  - dynamic problem solving
- (Rather) extensive bibliography

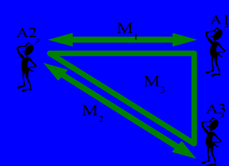
Annex

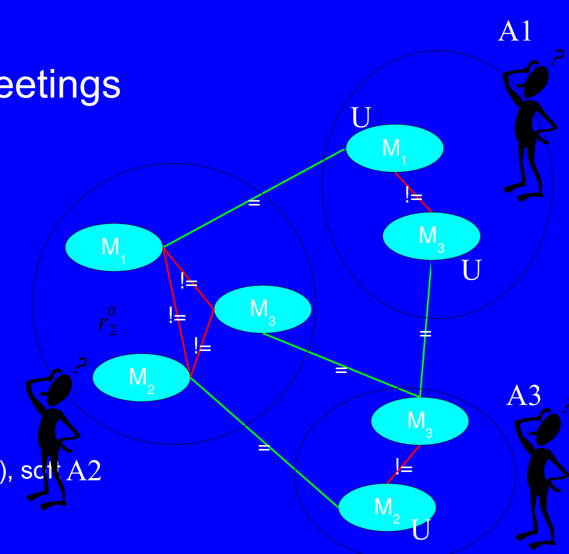
Conclusions

- MCOP is a powerful tool for many practical problems
- A lot of progress in recent years
- We need more:
  - Efficiency (less communication: DynProg, hybrids)
  - More realistic models (failures, message loss, etc)
  - Privacy
  - Incentives
- Growing community, increasing interest and visibility!
- Join us!

Example: Meeting Scheduling as DCOP

- Problem: setup 3 meetings
  -
  -
  -
  -
  - Variables=mtg time
  - Values=time slots
  - Constraints:
    - Hard (equality, inequality), soft

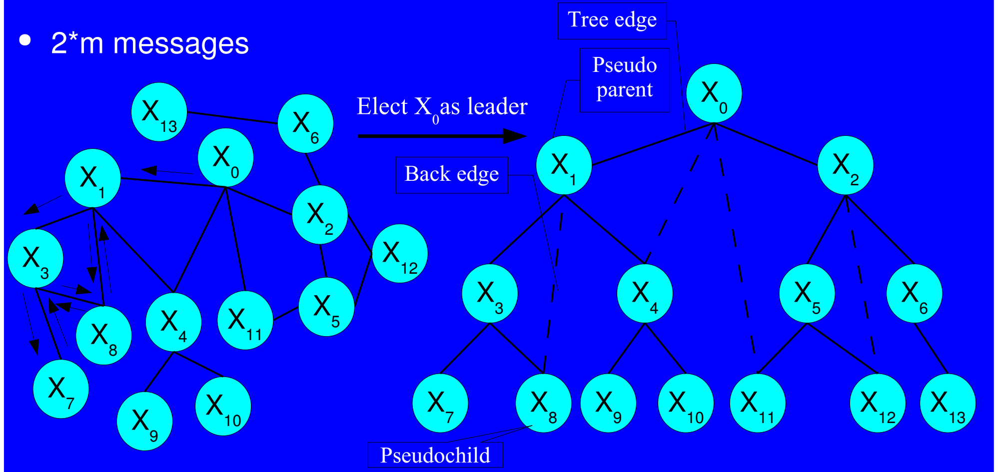




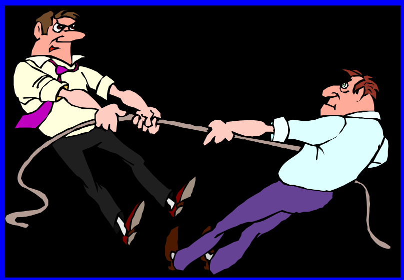
	U(M2)		
Time	8AM	9AM	10AM
Utility	2	5	10

Distributed depth first traversal

- *DFS tree*: neighbors lie in the same branch
- *Prop*: subtrees are independent (e.g. no link $X_8 - X_4$)
- 2*m messages



Self interested agents



Self-interested agents

- Agents have conflicting interests
- Agents can manipulate the algorithm:
  - exaggerate utilities
  - incorrect computation
  - forward bogus messages
- Optimization is meaningless
- Suboptimal outcomes

Incentive compatibility for self-interested agents

- Mechanism design: align incentives with system
- Vickrey-Clarke-Groves mechanism: [Ephrati & Rosenschein '91]
  - Each agent pays a tax:

    $\text{Pay}(A_i) = \sum_{j \neq i} R_j(V^*_{R \setminus R_i}) - R_j(V^*_R)$ (damage to others)
  - VCG is efficient, IC, IR.
- Distributed VCG difficult:
  - information revelation
  - computation
  - message passing

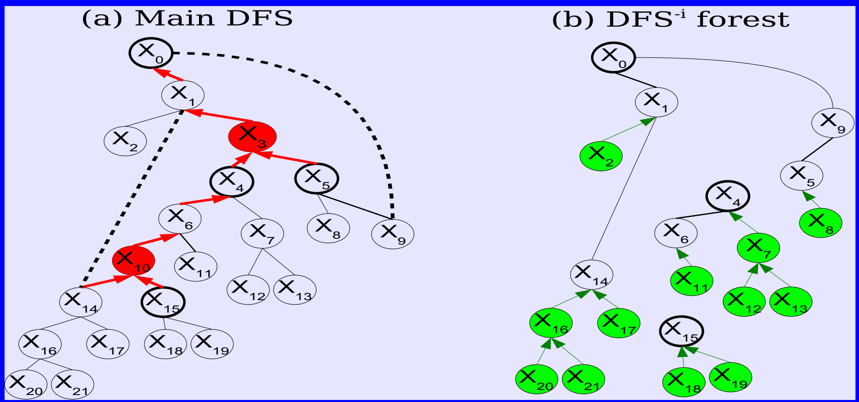
M-DPOP: the first distributed VCG mechanism

[Petcu, Faltings, Parkes '06]

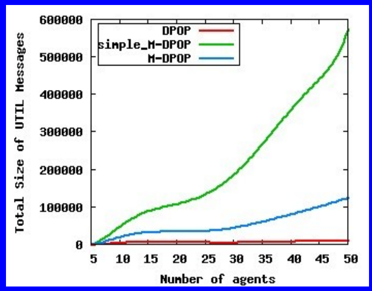
- First distributed implementation of VCG:
  - Only central entity required is the bank
  - All agents solve the main economy (using DPOP)
  - For each agent Ai:
    - the **other** agents solve MCOP(-i) (using DPOP)
    - the **other** agents report Ai's marginal effects
    - Bank fines Ai with sum of reports=VCG tax
  - Mechanism is faithful: truthfulness is ex-post Nash

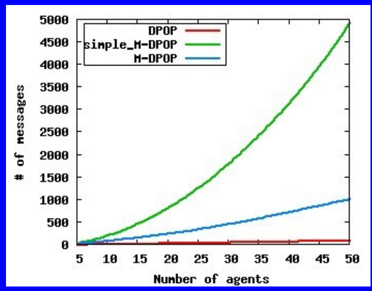
M-DPOP: reuse computation from main economy

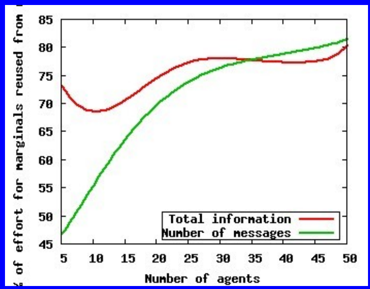
- reuses computation from MCOP(All) to MCOP(-i):
  - adapt DFS(-i) from DFS(All)
  - reuse only messages not influenced by Ai



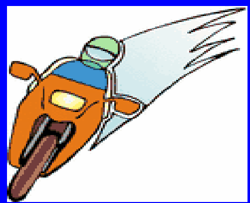
M-DPOP: reusing computation from main econ.





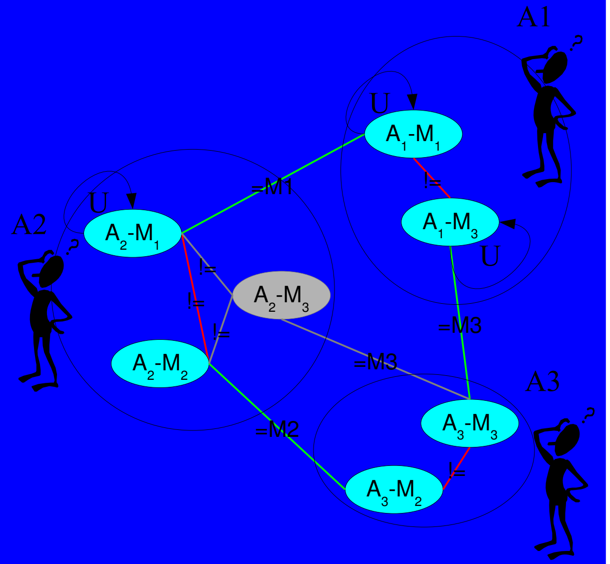


Dynamic Problems



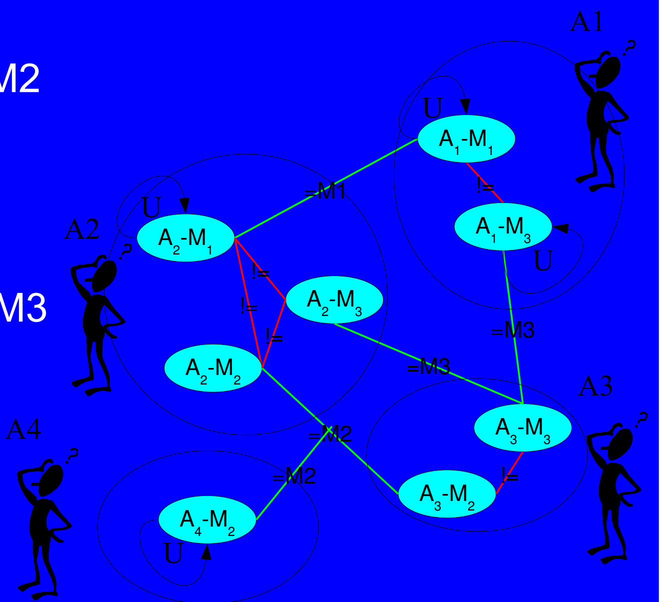
Question: what about dynamic problems?

- Agent A2 looses interest in M3
- Variables and constraints are deleted
- Perhaps a better schedule can be found!



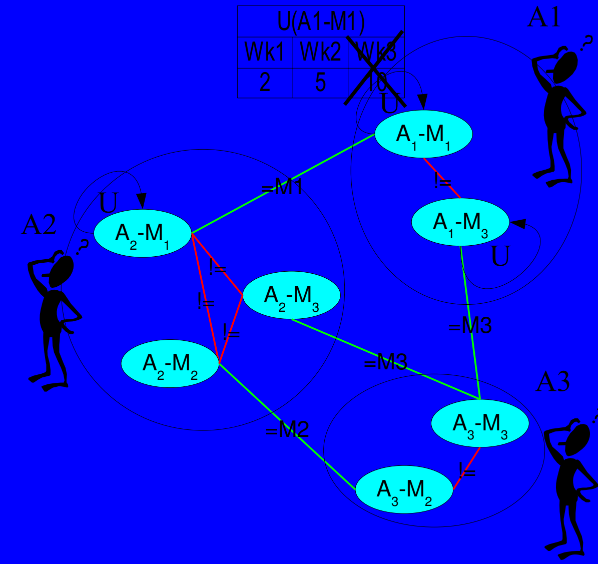
Question: what about dynamic problems?

- Agent A4 joins M2
- Variables and constraints are added
- Maybe M2 and M3 need to be rescheduled



Question: what about dynamic problems?

- Agent A1 goes on holiday from week 3
- Domains can change, constraints can change



Self-Stabilization for dynamic problems

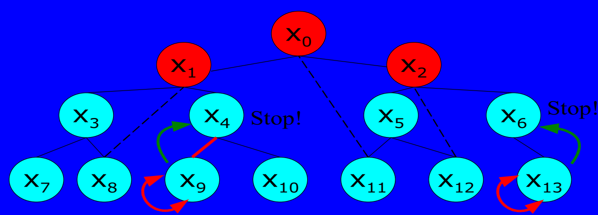
[Petcu & Faltings, AAAI'05]

- SS [Dijkstra '74]: always reach and maintain a stable state
  - In DCOP: stable state = optimal solution
  - Perturbations: everything can change: variables and relations can be added/deleted/changed
- S-DPOP: (self stabilizing DPOP)
  - run DPOP continuously
  - when changes occur, restart propagations
  - reuse previous computation
- Complete algorithm (given time between failures)

Self-stabilizing optimization: key ideas

[Petcu&Faltings @ AAAI'05]

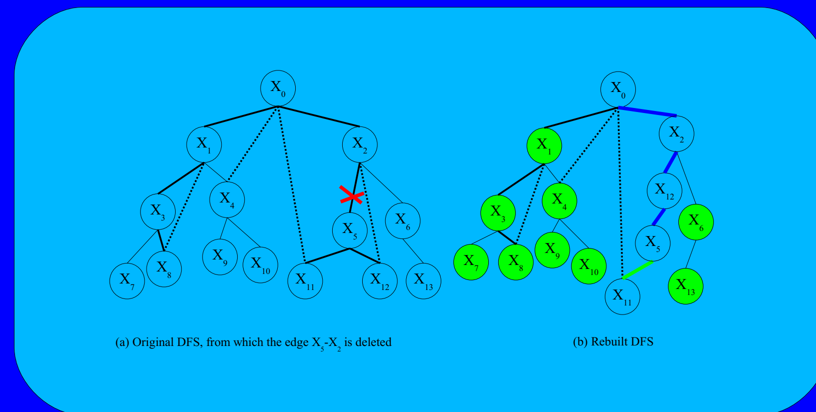
- Adjust DFS tree upon changes
- Restart propagations upon changes (ensures SS)
- Better than simply re-solving the problem:
  - Reuse previous computation whenever possible
  - Limiting the spread of new propagations (fault containment)



Self-stabilizing optimization: key ideas

[Petcu&Faltings @ AAAI'05]

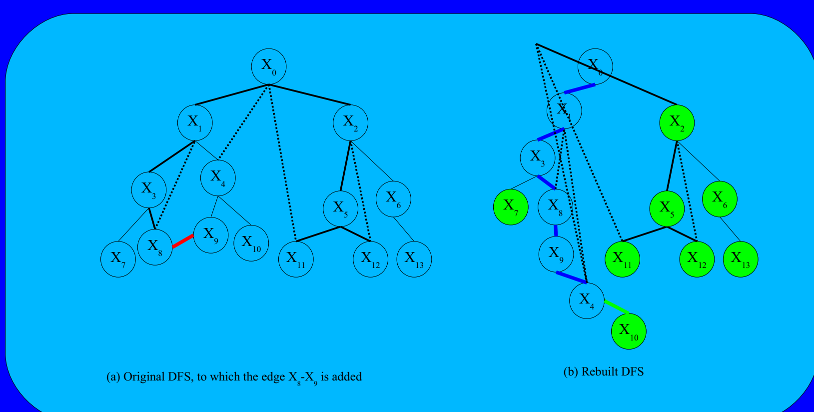
- Adjust DFS tree upon changes
- deleting an edge: green nodes are reusable



Self-stabilizing optimization: key ideas

[Petcu&Faltings @ AAAI'05]

- Adjust DFS tree upon changes
- adding a new edge: green nodes are reusable



8. Bibliography

In the following, we provide references to papers that either were referenced on the previous slides or that were a source for us when putting together this or previous tutorials. Please note that therefore this bibliography is in no way a complete overview of distributed knowledge-based search and distributed constraint reasoning. We first present the references (in a short form) clustered by general topics within constraint reasoning and then provide an alphabetical list.

Overviews of Distributed Constraint Reasoning:

- [Hamadi 99a] PhD thesis (in French)
- [Yokoo 01] book
- [Silaghi 02a] PhD thesis
- [Petcu 07c] PhD thesis
- [Silaghi&Yokoo 07] Encyclopaedia of AI. Chapter on Distributed Constraint Reasoning

Frameworks for Distributed Constraint Satisfaction

- [ZhaMac 91] Each agent is responsible for enforcing a part of the constraints
- [Yokoo 98] Each variable's domain is the secret of one agent
- [SilHaFal 00b] Each constraint's relation is the secret of one agent
- [MesJim 00] Agents stand for variables and each constraint's relation is the secret of one of the agents standing for a variable involved in it
- [Hannebauer 00] Separate agents knowing each constraint and each variable's domain
- [Silaghi 04] constraints not known to anybody, defined as functions on agents' secret inputs each agent gets the result of a function on inputs and a solution to the constraints

Overviews of Distributed Search:

- [Denzinger 00]

Distribution vs. Parallelization:

[Avenhaus et al. 02]

Distributed Search concepts based on a common search state:

[YokKit96] Central common state, or-graph-based search

[Bonacina 97] distributed common state, set-based search, specific application: theorem proving

[Schaeffer 89] distributed common state, and-or-tree-based search

[KumKan 84] central and distributed common state, and-tree-based and or-tree-based search

[Talukdar et al. 93] heterogeneous agents with central common states

Distributed search concepts based on dividing the problem into subproblems:

[Sycara et al. 91] homogeneous agents, static problem division, and-or-graph-based search

[Smith 79] homogeneous agents, dynamic problem division, and-tree-based search

[LanLes 92] heterogeneous agents, static problem division

[Fischer et al. 95] heterogeneous agents, dynamic problem division

Distributed search concepts based on improving on the competition approach:

[Bäck 94] homogeneous agents, no central control, set-based search

[Denzinger 95] homogeneous agents, central control, set-based search

[DenFuc 99] heterogeneous agents, no central control, application area: theorem proving

Synchronous Techniques

[ColDecKat 91/00] exploiting DFS trees, maintaining consistency in search

[YokDurIshKuw 92] synchronous backtracking

[SolGudMei 96] sync FC-CBJ, CFPA/CAPF modeling guidelines

[Tel 99] consistency maintenance improved

[MeiRaz 01] reordering heuristic in sync FC with parallel pruning

[PetFal 05] DPOP: dynamic programming in DCOP using a DFS tree ordering

Asynchronous Systematic Techniques for CSPs

[YokDurIshKuw 92] ABT (asynchronous search)

[Yokoo 93/95] AWC (reordering heuristic based on nogood generation in ABT)

[ArmDur 97] reordering heuristics in search with epoch-based synchronization

[YokHir 98] ABT for complex local problems

[SilHaFal 00b] AAS (ABT where several agents may modify the same variable)

[BesMaeMes 01] DisDB (i.e. ABT without add-link messages)

[Silaghi 02] AAS' (alternative to AAS)

[SilHaFal 01b] consistency maintenance in ABT

[SilHaFal 00a/01/01e] consistency maintenance in AAS

[SilHaFal 01/01c/01d] reordering and reordering heuristics for ABT and AAS

Consistency achievement (preprocessing)

[Kasif 90] intrinsic inefficiency of distribution of consistency achievement

[ZhaMac 91] simple iteration and exploiting graphs

[BauDev 97]/[NguDev 98] distributed AC4 and AC6 with message passing

[Hamadi 99] optimal distribution and corresponding optimal consistency

[MonRet 99] formalization for numeric domains

Consistency Maintenance in search

[SilHaFal 00a/01/01e] schemes for consistency maintenance in AAS

[ArbMon 00] distributed splitting with chaotic iteration

[SilHaFal 01b] consistency maintenance in ABT

[Silaghi 02a/03] consistency maintenance in Adopt and Asynch. PFC-MRDAC

[SilLanLarr 04] consistency maintenance with WAC for DisWCSPs with B & B and A*

Reordering in search

[Yokoo 93/95] AWC (reordering heuristic based on nogood generation in ABT)

[ArmDur 97] reordering heuristics in search with epoch-based synchronization

[MeiRaz 01] reordering heuristic in sync FC with parallel pruning

[SilHaFal 01/01c/01d] polynomial space reordering heuristics for ABT and AAS

[ZivMei 05]: AWC-like heuristic without self-reordering

[Silaghi 06b] Reordering frameworks

Forward Checking

[YokDurIshKuw 92] ABT (asynchronous search, FC pruning is done immediately after inst)

[LuoHenBuc 93] FC done hierarchically by receiving agents

[SolGudMei 96] sync FC-CBJ, CFPA/CAPF modeling guidelines

[MesJim 00] FC done locally by sending agent

[MeiRaz 01] reordering heuristic in sync FC with parallel pruning

Numeric problems

[MonRet 99] formalization of consistency achievement for numeric domains

[ArbMon 00] mixed consistency and splitting

[SilHaFal 01]/[SilSteHaFal 01] polynomial space solving for numeric CSPs

Add-link messages

[YokDurIshKuw 92] ABT (introduction of add link messages)

[BesMaeMes 01] DisDB (i.e. ABT without add-link messages)

[SilHaFal 01] One-Shot-Links (i.e. ABT with (temporary) single usage links)

[BesMaeMes 02] experimental comparison of ABT, One-Shot-Links, DisDB

Privacy

[YokDurIshKuw 98] privacy of domains mentioned as motivation for ABT

[SilHaFal 00c] privacy of constraints for distributed CSPs

[MesJim 00] mixed privacy of domains and partial privacy of constraints

[FreMinWal 01] privacy measure based on number of revealed constraint tuples

[SilFal 02] comparison of ABT, AWC, AAS, DMAC, DMAC-ABT,… with respect to privacy

[YokSuzHir 02] cryptographic technique based on external servers

[Silaghi 03a/04] cryptographic technique without external servers

[WalSil 04] search strategies to reduce privacy loss

[SilRaj 04] DisCSP formulation for reducing privacy loss

Openness

[JunTamZhaSh 00] openness support in AWC

[ModJunTamShKul 01] framework with openness

[SilFal 02a] openness in ABT, AAS, DMAC, …

[FalMac 02] open CSPs

[Mod 03] message exchange

[PetFal 06c] open DPOP

Optimization

[Yokoo 93] incremental relaxation

[YokKit96] [cooperative A*]

[YokHir 98] synchronous B&B

[SilHaCalFal 01] branch and bound for AAS and SAS

Self-interested agents

[PetFal 05]: a limited distributed implementation of the VCG mechanism

[PetFalPar 06]: M-DPOP: the first faithful mechanism for social choice problems

[FalParPetShn 06]: an application of M-DPOP to overlay networks

[SilFalPet 06]: secure optimization with guarantees of privacy

[PetFalParXue 07]: BB-M-DPOP: structural techniques for budget balance in VCG

Dynamic DCOP:

[ColDecKat 99]: a self stabilizing search algorithm for DisCSP

[PetFal 05b]: S-DPOP: a self-stabilizing extension of DPOP for dynamic problems

[PetFal 07]: R-DPOP: self-stabilizing extension of DPOP that provides solution stability

[PetFalDec 07]: an overview of DFS based algorithms, includes dynamic algorithms

Partial Centralization:

[MailLes 03]: OptAPO: asynchronous centralization of subproblems in mediation sessions

[PetFal 07b]: PC-DPOP identifies and centralizes difficult subproblems

Tradeoffs:

[PetFal 05b]: A-DPOP: approximation algorithm, tradeoff between effort and solution quality

[PetFal 06]: O-DPOP: open DPOP, tradeoff between message size and number of messages

[PetFal 07a]: MB-DPOP: tradeoff between memory requirements and number of messages

[PetFal 07c]: LS-DPOP: local search hybrid; tradeoff between effort and solution quality

[ModTamShYok 02] Adopt (A* in asynchronous search)

[Silaghi 02a] DVR-MAS (branch & bound for DMAC and extension with Adopt's A*)

[Silaghi 03] Asynchronous PFC-MRDAC (consistency+B&B+Adopt)

[SilLanLar 04] WAC with asynchronous B & B and A*

[PetFal 05] DPOP: dynamic programming in DCOP using a DFS tree ordering

[KumPetFal 07]: H-DPOP: uses Constraint Decision Diagrams for pruning in DPOP

Hill-climbing

[YokHir 96] Distributed Breakout

[Fabiuke 97] DSA: Distributed Stochastic Algorithm

[ZhaWanWit 02] Analysis of DSA and Distr. Breakout

[Liu et al 01] Hill climbing by dynamic changes in problem

[ArsSil 03/04] DSAN: Distrib. Simulated Annealing (async and synch)

[Chouiery et al 03] evaluation of hill climbing methods

[PetFal 07e] LS-DPOP: hybrid of local search and dynamic programming

Semi-cooperative search

[DenFed 06] Semi-cooperative agents using improvement on competition approach

Bibliography in lexicographical order

[ArmDur 97] Armstrong, A.; Durfee, E.F.:
Dynamic prioritization of complex agents in distributed constraint satisfaction problems,
15th IJCAI, 1997.

[ArbMon 00] Arbab, M.; Monfroy, E.:
Distributed splitting of constraint satisfaction problems,
Coordination, 2000

[ArsSil 03] Arshad, M.; Silaghi, M.C.:
Distributed Simulated Annealing
Frontiers in AI and Applications (Proc. of IJCAI03-DCR Workshop) IOS, 2003.

[Avenhaus et al. 02] Avenhaus, J. ; Denzinger, J. ; Küchlin, W. ; Sinz, C.:
Teamwork-PaReDuX: Knowledge-based Search with Multiple Parallel Agents,
Proc. MPCS-02, Ischia, 2002.

[Bäck 94] Bäck, T.:
Parallel Optimization of Evolutionary Algorithms,
Proc. Parallel Problem Solving from Nature III, LNCS, 1994, pp. 418-427.

[BauDev 97] Baudot, B.; Deville, Y.:
Analysis of distributed arc-consistency algorithms
TR RR-97-07, U. Catholique Louvain, 1997.

[Bessiere 91] Bessiere, C.:
Arc-consistency in dynamic constraint satisfaction problems
AAAI91, 1991.

[BesMaeMes 01] Bessiere, C.; Maestre, A.; Meseguer, P.:
Distributed dynamic backtracking
IJCAI DCR Workshop, 9-16, 2001.
[BesMaeMes 02] Bessiere, C.; Maestre, A.; Meseguer, P.:
The ABT family
JFNP, 2002.
[Bonacina 97] Bonacina, M.P.:
The Clause-Diffusion Theorem Prover Peers-mcd,
Proc. CADE-14, Townsville, LNAI 1249, 1997, pp. 53-56.
[ColDecKat 00] Collin, Z.; Dechter, R.; Katz, S.:
Self-stabilizing distributed constraint satisfaction
Chicago Journal of Theoretical Computer Science, 2000. (first version published in 1990)
[ConKuwLes 91] Conry, S.E.; Kuwabara, K.; Lesser, V.R.:
Multistage Negotiation for Distributed Constraint Satisfaction
IEEE Trans on systems, man, cybernetics 21(6):1426-1477, 1991.
[DenFed06] Denzinger, J.; Fedoruk, A.:
A General Framework for Multi-agent Search with Individual and Global Goals:
Stakeholder Search,
International Transactions on Systems Science and Applications (ITSSA), Vol 1(4), 2006, pp. 357-362.
[DenFuc 99] Denzinger, J.; Fuchs, D.:
Cooperation of Heterogeneous Provers,
Proc. IJCAI-99, Stockholm, Morgan-Kaufmann, 1999, pp. 10-15.
[Denzinger 95] Denzinger, J.:
Knowledge-Based Distributed Search Using Teamwork,
Proc. 1st ICMAS, San Francisco, CA, USA, 1995, pp. 81-88.

[Denzinger 00] Denzinger, J.:
Conflict Handling in Collaborative Search,
in Tessier, Chaudron, Müller (eds.): Conflicting Agents: Conflict management in multi-agent systems, Kluwer Academic Publishers, 2000, pp. 251-278.
[FalParPetShn 06] B. Faltings, D. Parkes, A. Petcu and J. Shneidman:
Optimizing Streaming Applications with Self-Interested Users using MDPOP.
COMSOC'06: International Workshop on Computational Social Choice , Amsterdam, The Netherlands, December, 2006.
[FalMac 02] Faltings, B.; Macho-Gonzales, S.:
Open Constraint Satisfaction Problems
CP, Ithaca, 2002
[FelFriJanSilZan 02] Felfernig,A.;Friedrich,G.;Jannach,D.;Silaghi,M.;Zanker,M.:
Distributed generative CSP framework for multi-site product configuration
ECAI Workshop on Configuration, 2002.
[FerBejKriGom 00] Fernandez, C.; Bejar, R.; Krishnamachari, B.; and Gomes, C.:
Communication and Computation in DCSP Algorithms
CP, Ithaca, September, 2002
[Fischer et al. 95] Fischer, K.; Müller, J.P.; Pischel, M.; Schier, D.:
A Model for Cooperative Transportation Scheduling,
Proc. ICMAS-95, San Francisco, AAAI-Press, 1995, pp. 109-116.
[FreMinWal 01] Freuder, E.C.; Minca, M.; Wallace, R.:
Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents
IJCAI DCR Workshop 63-72, 2001.
[Hamadi 99] Hamadi, Y.:
Optimal distributed arc-consistency
CP, 1999.

[Hamadi 99a] Hamadi, Y.:
Traitement des problèmes de satisfaction de contraintes distribues
PhD Dissertation, Universite Montpellier II, 1999.
[HaBe 98] Hamadi, Y.; Bessiere, C.:
Backtracking in distributed constraint networks
ECAI98, 219-223, 1998.
[Havens 97] Havens, W.:
Nogood caching in multiagent backtrack search
AAAI CA Workshop, 1997.
[JunTamZhaShe 00] Jung, H.; Tambe, M.; Zhang, W.; Shen, W.M.:
On modeling argumentation as distributed constraint satisfaction: Initial results
CP-DCS, 47-56, 2000.
[JusDebBoi 00]
Maintaining arc-consistency within dynamic backtracking
CP, Singapore, 2000
[KumKan 84] Kumar, V.; Kanal, L.N.:
Parallel Branch-and-Bound Formulations for AND/OR Tree Search,
IEEE Trans. Pattern Analysis and Machine Intelligence 6(6), 1984, pp. 768-778.
[KumPetFal 07] Kumar, A; Petcu, A; Faltings, B
H-DPOP: Using Hard Constraints to Prune the Search Space.
IJCAI'07 - Distributed Constraint Reasoning workshop, DCR'07, Jan, 2007.
[Lamport 78] Lamport, L.:
Time, clocks and the ordering of events in a distributed system
Communications of the ACM 21(7):558-565, 1978.

[LanLes 92] Lander, S.E.; Lesser, V.R.:
Customizing Distributed Search Among Agents with Heterogeneous Knowledge,
Proc. 1st Intern. Conf. on Information and Knowledge Management, Baltimore, 1992.
[LuoHenBuc 93] Luo, Q.; Hendry, P.; Buchanan, J.:
Heuristic search for distributed constraint satisfaction problems
TR KEG-6-93, Univ. of Strathclyde, 1993.
[MeiRaz 01] Meisels, A.; Razgon, I.:
Distributed Forward Checking with Dynamic Ordering
CP01 COSOLV Workshop, 21-27, 2001.
[MesJim 00] Meseguer, P.; Jimenez, M.A.:
Distributed Forward Checking
CP DCS Workshop, 2000
[MinJohPhiLai 90]
Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method
AAAI90, 17-24, 1990
[ModJunTamShKul 01] Modi,P.J.;Jung,H.;Tambe,M.;Shen,W.-M.;Kulkarni,S.:
Dynamic distributed resource allocation: A distributed constraint satisfaction approach
IJCAI DCR Workshop, 2001.
[ModTamShYok 03] Modi, P.J.; Tambe, M.; Shen, W.-M.; Yokoo, M.:
Asynchronous Distributed Optimization (ADOPT)
AAMAS, 2003.

[Mod 03] Modi, P.J.;
Distributed Constraint Reasoning under unreliable communication.
IJCAI03-DCR Workshop, 2003

[MonRet 99] Monfroy, E.; Rety, J.-H.:
Chaotic iteration for distributed constraint propagation
14th ACM SAC 19-24. 1999.

[Morris 93] Morris, P.:
The breakout method for escaping from local minima
AAAI93, 40-45, 1993.

[NguDev 98] Nguyen, T.; Deville, Y.:
A distributed arc-consistency algorithm
Science of Computer Programming 30(1-2):227-250, 1998.

[PetFal 05] A. Petcu and B. Faltings:
DPOP: A Scalable Method for Multiagent Constraint Optimization.
IJCAI 05, Edinburgh, Scotland, Aug, 2005, pp. 266-271.

[PetFal 05a] A. Petcu and B. Faltings:
S-DPOP: Superstabilizing, Fault-containing Multiagent Combinatorial Optimization.
Proceedings of the National Conference on Artificial Intelligence, AAAI-05, Pittsburgh, Pennsylvania, USA, July, 2005, pp. 449-454.

[PetFal 05b] A. Petcu and B. Faltings:
A-DPOP: Approximations in Distributed Optimization.
CP05 - workshop on Distributed and Speculative Constraint Processing, DSCP, Sitges, Spain, October, 2005.

[PetFal 05c] A. Petcu and B. Faltings:
Incentive Compatible Multiagent Constraint Optimization.
LNCS 3828: WINE'05 - Workshop on Internet and Network Economics, Hong Kong, Dec, 2005, pp. 708-717.

[PetFalPar 06a] A. Petcu, B. Faltings and D. Parkes:
M-DPOP: Faithful Distributed Implementation of Efficient Social Choice Problems.
AAMAS'06 - Autonomous Agents and Multiagent Systems, Hakodate, Japan, May, 2006, pp. 1397-1404.

[PetFal 06b] A. Petcu and B. Faltings:
O-DPOP: An algorithm for Open/Distributed Constraint Optimization.
Proceedings of the National Conference on Artificial Intelligence, AAAI-06, Boston, USA, July, 2006, pp. 703-708.

[PetFal 07a] A. Petcu and B. Faltings:
MB-DPOP: A New Memory-Bounded Algorithm for Distributed Optimization.
Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-07, Hyderabad, India, Jan, 2007, pp. 1452-1457.

[PetFalMai 07b] A. Petcu, B. Faltings, R. Mailler:
PC-DPOP: A New Partial Centralization Algorithm for Distributed Optimization.
Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-07, Hyderabad, India, Jan, 2007, pp. 167-172.

[PetFal 07c] A. Petcu and B. Faltings:
R-DPOP: Optimal Solution Stability in Continuous-Time Optimization.
IAT 2007 - International Conference on Intelligent Agent Technology, Fremont, CA, USA, Nov, 2007.

[PetFal 07d] A. Petcu and B. Faltings:
LS-DPOP: A Hybrid of Inference and Local Search for Distributed Combinatorial Optimization
IAT 2007 - International Conference on Intelligent Agent Technology, Fremont, CA, USA, Nov, 2007.

[PetFalPar 07] Petcu, A; Faltings, B; Parkes, D:
M-DPOP: Faithful Distributed Implementation of Efficient Social Choice Problems
submitted to JAIR (Journal of Artificial Intelligence Research), April 2007

[PetFalParXue 07] Petcu, A; Faltings, B; Parkes, D; Xue, W
BB-M-DPOP: Budget-Balance in Social Choice based on Problem Structure
Technical Report ID: LIA-REPORT-2007-002, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), April, 2007.

[PetFalDec 07] Petcu, A; Faltings, B; Dechter, R:
DFS-based Algorithms for (Dynamic) Distributed Constraint Optimization Problems
submitted to the Artificial Intelligence Journal, Aug. 2007

[Petcu 07] Adrian Petcu:
PhD Thesis, Ecole Polytechnique Federale de Lausanne
Lausanne, Switzerland, 2007

[ProConMul 92] Prosser, P.; Conway, P.; Muller, C.:
A constraint maintenance system for distributed allocation problems
1st IC on Intelligent Systems Engineering, 197-202, 1992.

[Puget 97] Puget, J.F.:
A fast algorithm for the bound consistency of alldiff constraints
AAAI98, 359-366, 1998.

[SabFre 94] Sabin. D.; Freuder, E.C.:
Contradicting conventional wisdom in constraint satisfaction
ECAI 94, 125-129, 1994.

[Schaeffer 89] Schaeffer, J.:
Distributed Game-Tree Searching,
J. of Parallel and Distributed Computing 6, 1989, pp. 90-114.

[SelLevMit 92] Selman, B.; Levesque, H.; Mitchell, D.:
A new method for solving hard satisfiability problems
AAAI92, 440-445, 1992.

[Silaghi 02] Silaghi, M.C.:
How to get AAS' when you just have AAS
CP2002-TRICS, Ithaca,US, 2002.

[Silaghi 02a] Silaghi, M.C.:
Asynchronously Solving Problems with Privacy Requirements
PhD Thesis, Swiss Federal Institute of Technology (EPFL), June, 2002.

[Silaghi 03] Silaghi, M.C.:
Asynchronous PFC-MRDAC±Adopt: Consistency-Maintainance in Adopt (±B&B)
IJCAI03 DCR Workshop, 2003.

[Silaghi 03a] Silaghi, M.C.:
Solving a distributed CSP with cryptographic multi-party computations, without revealing constraints and without involving trusted servers
IJCAI03 DCR Workshop, 2003.

[Silaghi 04] Silaghi, M.C.:
Desk-mates and Stable Marriages problems solved with n/2-privacy of human preferences
CP04 (subm.), 2004 (FIT TR 14/2004)

[Silaghi 06] Silaghi, M.C.
Framework for modeling reordering heuristics for asynchronous backtracking, IAT 2006

[SilFal 01] Silaghi, M.C.; Faltings, B.:
Parallel Proposals in Asynchronous Search
EPFL-TR-01/371, August, 2001.
[SilFal 02] Silaghi, M.C.; Faltings, B.:
A Comparison of Distributed Constraint Satisfaction Techniques with respect to Privacy
AAMAS-DCR(W12), Bologna, Italy, 2002
[SilFal 02a] Silaghi, M.C.; Faltings, B.:
Openness in Asynchronous Constraint Satisfaction Algorithms
AAMAS-DCR(W12), Bologna, Italy, 2002
[SilHaFal 00] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Fractionnement Intelligent de domaine pour CSPs avec domaines ordonnes
Proc. RFIA2000, III,439-448, Paris, Feb. 2000.
[SilHaFal 00a] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Maintaining Hierarchical Distributed Consistencies
CP2000 DCS Workshop, Singapore, 2000.
[SilHaFal 00b] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Asynchronous Search with Aggregations
AAAI2000, Austin, 2000.
[SilHaFal 00c] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Distributed Asynchronous Search with Private Constraints
AA2000, Bologna, April, 2000.
[SilHaFal 01] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Polynomial-Space and Complete Multiply Asynchronous Search with Abstractions
IJCAI-01 DCR Workshop, Seattle, 2001.
[SilHaFal 01a] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
ABT with Asynchronous Reordering
2nd Asia-Pacific IAT, Maebashi, Japan, 2001.

[SilHaFal 01b] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Maintaining Consistency in ABT
CP, Paphos, Cyprus, 2001.
[SilHaFal 01c] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Asynchronous Consistency Maintenance with Reordering
EPFL-TR-01/360, March, 2001.
[SilHaFal 01d] Silaghi, M.C.; Sam-Haroud, D.; Faltings, B.:
Hybridizing ABT and AWC into a polynomial space, complete protocol with reordering
EPFL-TR-01/364, May 2001
[SilHaFa 01e] Silaghi, M.C., Sam-Haroud, D.; Faltings, B.:
Asynchronous Consistency Maintenance
2nd Asia-Pacific IAT,Maebashi,Japan,2001
[SilHaCaFal 01a] Silaghi, M.C.; Sam-Haroud, D.; Calisti, M.; Faltings, B.:
Negotiations by Relaxation in Dynamic Distributed CSPs with Private Constraints
EPFL-TR-01/365: Submitted to 1st AAMAS, August 2001
[SilHaCaFal 01b] Silaghi, M.C.; Sam-Haroud, D.; Calisti, M.; Faltings, B.:
Generalized English Auctions by Relaxation in Dynamic Distributed CSPs with Private Constraints
IJCAI-01 DCR Workshop, Seattle, 2001
[SilLanLar 04] Silaghi, M.C.; Landwehr, J.; Larrosa Bondia, J.
Asynchronous Branch & Bound and A* for DisWCSPs,
with heuristic function based on Consistency Maintenance
Frontiers in Artificial Intelligence and Applications, IOS, 2004
[SilRaj 04] Silaghi, M.C.; Rajeshirke, V.:
The effect of policies for selecting solutions of a Distributed CSP, AAMAS04, 2004.

[Silagh&iYokoo 06] Silaghi. M.C.; Yokoo, M.:
Nogood-based Asynchronous Distributed Optimization, AAMAS 2006
[SilaghiYokoo 07b]
Dynamic DFS tree in ADOPT-ing,
AAAI 2007
[SilaghiYokoo 07a]
Distributed Constraint Reasoning,
Encyclopaedia of AI, Information Science Reference 2007
[Smith 79] Smith, R.G.:
A Framework for Distributed Problem Solving,
UMI Research Press, 1979.
[SolGudMei 96] Solotorevsky, G.; Gudes, E.; Meiseles, A.:
Algorithms for solving distributed constraint satisfaction problems (DCSPs)
AIPS96, 1996.
[Sycara et al. 91] Sycara, K.; Roth, S.; Sadeh, N.; Fox, M.:
Distributed Constrained Heuristic Search,
IEEE Trans. Systems, Man, and Cybernetics 21(6), 1991, pp. 1446-1461.
[Talukdar et al. 93] Talukdar, S.N.; de Souza, P.S.; Murthy, S.:
Organizations for Computer-based Agents,
Journal of Engineering Intelligent Systems, 1993.
[Tel 99] Tel, G.:
Chapter: Distributed Control Algorithms for AI
Multiagent Systems, A Modern Approach to Distributed AI. 539-580, MIT Press, 1999

[WaSil 04] Wallace, R.; Silaghi, M.:
Using privacy loss to guide decisions during distributed CSP search, FLAIRS04, 2004
[YokDurIshKuw 92] Yokoo, M.; Durfee, E.H.; Ishida, T.; Kuwabara, K:
Distributed constraint satisfaction for formalizing distributed problem solving, ICDCS, 614-621, 1992.
[YokDurIshKuw 98] Yokoo, M.; Durfee, E.H.; Ishida, T.; Kuwabara, K:
The distributed constraint satisfaction problem: Formalization and algorithms
IEEE Trans on KDE 10(5):673-685, 1988.
[Yokoo 93] Yokoo, M.:
Dynamic value and variable ordering heuristics for distributed constraint satisfaction
Workshop on Intelligent Agents, 1993
[Yokoo 93a] Yokoo, M.:
Constraint relaxation in distributed constraint satisfaction problem
ICDCS93, 56-63, 1993
[Yokoo 95] Yokoo, M.:
Asynchronous weak-commitment search for solving large-scale distributed constraint satisfaction problems.
ICMAS, 467-318, 1995
[YokHir 96] Yokoo, M.; Hirayama, K.:
Distributed breakout algorithm for solving distributed constraint satisfaction problems
ICMAS96, 401-408, 1996.

[YokHir 98] Yokoo, M.; Hirayama, K.:
Asynchronous backtracking with complex local problems
ICMAS98, 1998.
[YokKit 96] Yokoo, M. ; Kitamura, Y.:
Multiagent Real-Time-A* with Selection: Introducing Competition in Cooperative Search,
Proc. ICMAS-96, Kyoto, AAAI-Press, 1996, pp. 409-416.
[Yokoo 01] Yokoo, M.:
Distributed Constraint Satisfaction
Springer, 2001
[YokSuzHir 02] Yokoo, M.; Suzuki, K.; Hirayama, K:
Secure distributed constraint satisfaction: Reaching agreement without revealing private information
AAMAS02 DCR Workshop, 2002
[ZhaMac 91] Zhang, Y.; Mackworth, A.K.:
Parallel and distributed algorithms for finite constraint satisfaction problems
3rd IEEE Symposium on PDP 394-397, 1991.
[ZhaWanWit 02] Zhang, W.; Wang, G.; Wittenburg, L.:
Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance
AAAI-PAS Workshop, 2002.
[ZiMei 02] Zivan, R.; Meisels, A.:
Parallel Asynchronous Weak-Commitment Algorithms
AAMAS-DCR Workshop, 2002.
[ZiMei 02] Zivan, R.; Meisels, A.:
A Dynamic Reordering Heuristic for Distributed CSPs
CP, 2005.