Multiagent Systems (BE4M36MAS)

Distributed Constraint Programming

Branislav Bošanský and Michal Pěchouček

Artificial Intelligence Center,
Department of Computer Science,
Faculty of Electrical Engineering,
Czech Technical University in Prague

branislav.bosansky@agents.fel.cvut.cz

November 29, 2016

Previously ... on multi-agent systems.

1. Coalitional/Cooperative Game Theory

Constraint Reasoning

Many real-world problems can be specified as constrained programming.

Real-world scenarios are naturally constrained by limited resources (money, time, energy).

The agents *cooperatively* seek optimal plan with respect to these constraints. Examples:

- scheduling and planning
- sensor placement
- solving coordination and optimization problems in MAS

Set of agents must come to some agreement, typically via some form of negotiation, about which action each agent should take in order to jointly obtain the best solution for the whole system.

Constraint Reasoning in/for MAS

In multi-agent systems: Distributed Constraint Reasoning Problems

Each agent negotiates locally with just a subset of other agents (usually called neighbors) that are those that can directly influence his/her behavior.

Leads to a distributed and a robust solution.

Particularly useful, when agents are lightweight/simple and there is an open system (think Internet of Things).

Constraint Network

Definition

A constraint network $\mathcal{N}$ is formally defined as a triple $\langle X, D, C \rangle$, where:

- $X = x_1, \ldots, x_n$ is a set of variables;
- $D = \{D_1, \ldots, D_n\}$ is a set of variable domains, which enumerate all possible values of the corresponding variables; and
- $C = \{C_1, \ldots, C_m\}$ is a set of constraints; where a constraint C_i is defined on a subset of variables $S_i \subseteq X$ which comprise the scope of the constraint ($r_i = |S_i|$ is the arity of constraint i)

Hard vs. Soft Constraints

Hard constraint C_i^h is a Boolean predicate P_i that defines valid joint assignments of variables in the scope

$$P_i : D_1^i \times \ldots \times D_{r_i}^i \rightarrow \{F, T\}$$

Soft constraint C_i^s is a function F_i that maps every possible joint assignment of all variables in the scope to a real value

$$F_i : D_1^i \times \ldots \times D_{r_i}^i \rightarrow \mathbb{R}$$

Binary Constraint Networks

Binary constraint networks are those where each constraint (soft or hard) is defined over two variables.

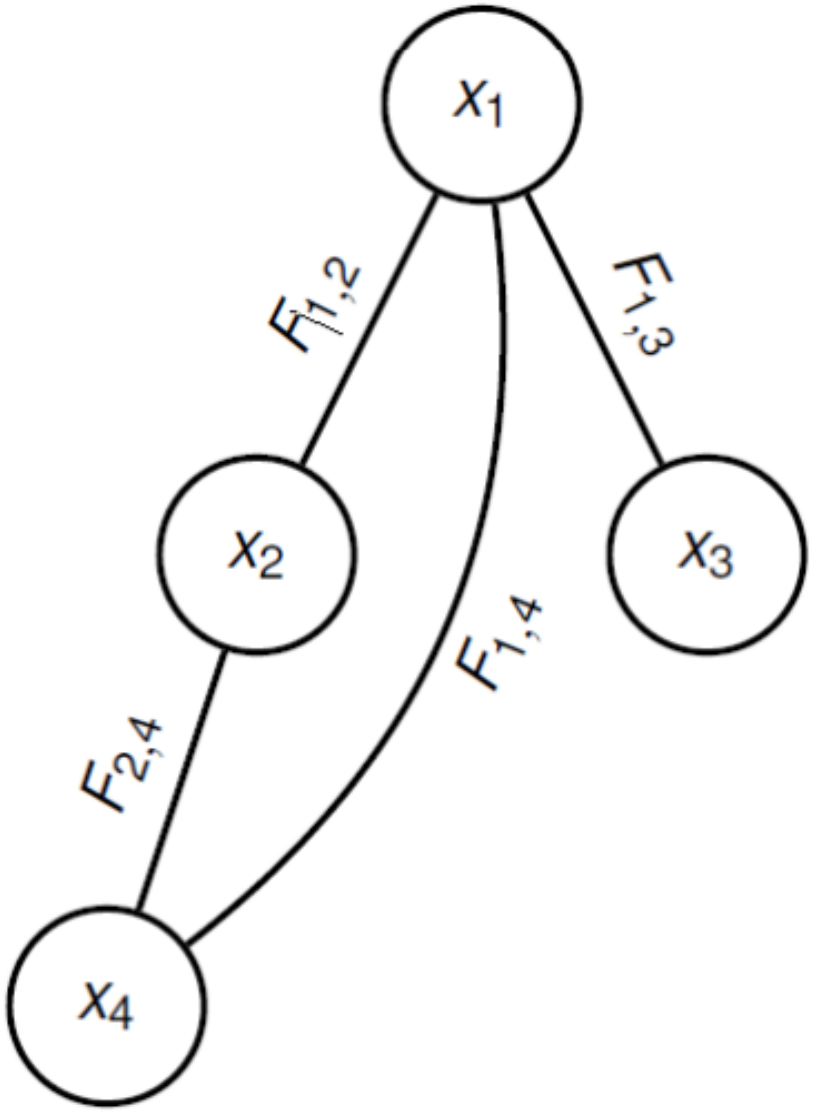
Binary constraint networks can be represented by a constraint graph.

Every constraint network can be mapped to a binary constraint network:

- requires the addition of variables and constraints
- may increase the size of the model

Algorithms explained for binary constraints but can be extended to n-ary.

Binary Constraint Networks



Types of Constraint Reasoning Problems

Constraint Satisfaction Problem (CSP)

- Objective: find an assignment for all the variables in the network that satisfies all constraints.
- Extension to MaxCSP/MinCSP: Maximize the number of satisfied constraints/minimize the number of violated constraints.

Constraint Optimization Problem (COP)

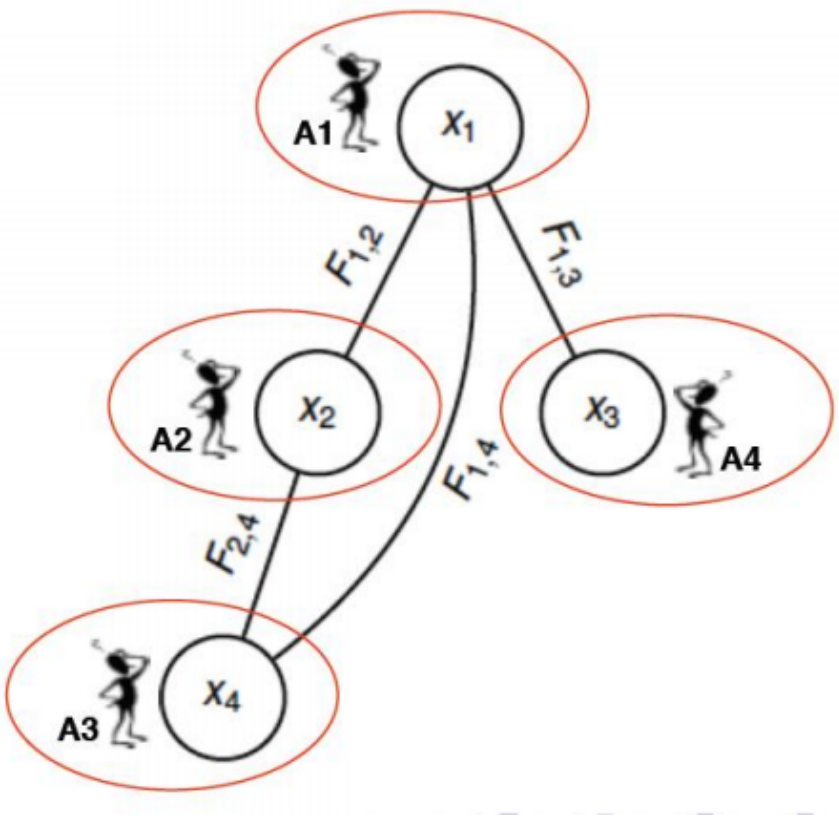
- Objective: find an assignment for all the variables in the network that satisfies all constraints and optimizes a global function.
- Global function = aggregation (typically sum) of constrain functions, i.e., $F = \sum F_i$

COP provides more modeling power on the expense of more complex solution algorithms.

Distributed Constraint Reasoning

When operating in a decentralized context:

- a set of agents control variables
- agents interact to find a solution to the constraint network



Distributed Constraint Reasoning

A distributed constraint reasoning problem consists of a constraint network $\langle X, D, C \rangle$ and a set of agents $A = A_1, \ldots, A_k$ where each agent:

- controls a subset of the variables $X_i \subseteq X$
- is only aware of constraints that involve variable it controls
- communicates only (locally) with its neighbors 1:1 agent-to-variable mapping assumed for algorithm explanation

Distributed Constraint Reasoning

Synchronous

- A few agents are active, most are waiting
- Active agents take decisions with up-to-date information
- Low degree of concurrency
- Poor robustness
- Algorithms: direct extensions of centralized ones

Asynchronous

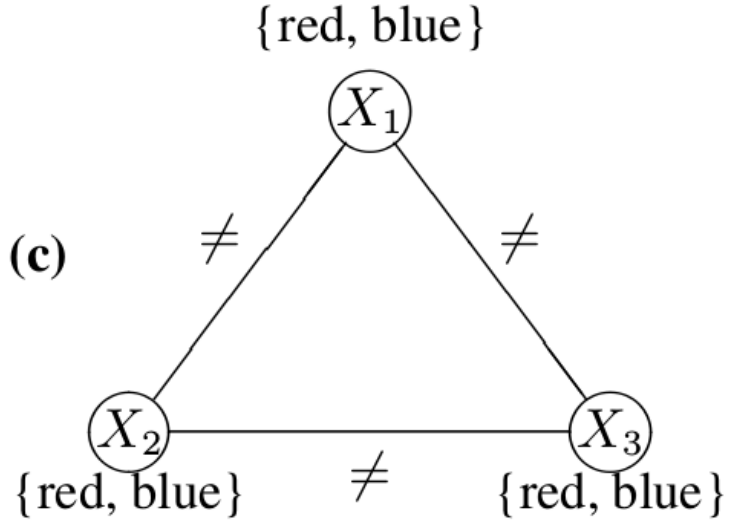
- All agents are active simultaneously
- Information is less updated, obsolescence appears
- High degree of concurrency
- High robustness
- Algorithms: new approaches

Filtering

Each node communicates its domain to its neighbors, eliminates from its domain the values that are not consistent with the values received from the neighbors, and the process repeats. Specifically, each node x_i with domain D_i repeatedly executes the procedure **Revise**(x_i, x_j) for each neighbor x_j.

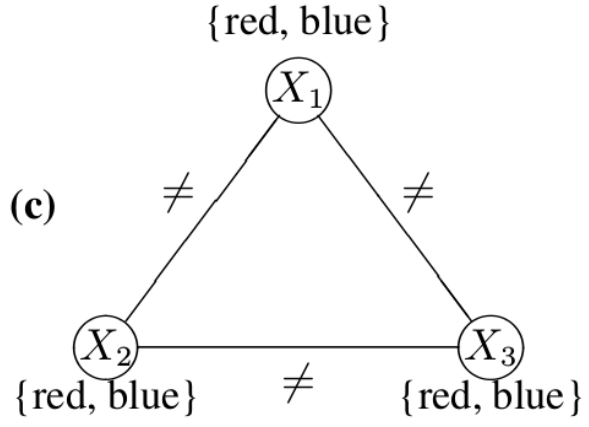
Distributed variant of *arc consistency* from CSPs.

Recall: Is this sufficient?



Filtering

Filtering is a special case of inference rule termed *unit resolution*:



$$\frac{A_1 \qquad \neg(A_1 \wedge A_2 \wedge \ldots \wedge A_n)}{\neg(A_2 \wedge \ldots \wedge A_n)} \qquad \frac{x_1 = \text{red} \qquad \neg(x_1 = \text{red} \wedge x_2 = \text{red})}{\neg(x_2 = \text{red})}$$

Generates forbidden value combinations called *Nogoods*.

Hyper Resolution

Can be generalized to *hyper resolution*:

$$\frac{\begin{array}{r} A_1 \vee A_2 \vee \ldots \vee A_m \\ \neg(A_1 \wedge A_{1,1} \wedge A_{1,2} \wedge \ldots) \\ \neg(A_2 \wedge A_{2,1} \wedge A_{2,2} \wedge \ldots) \\ \vdots \\ \neg(A_m \wedge A_{m,1} \wedge A_{m,2} \wedge \ldots) \end{array}}{\neg(A_{1,1} \wedge \ldots \wedge A_{2,1} \wedge \ldots)}$$

Filtering using Hyper Resolution

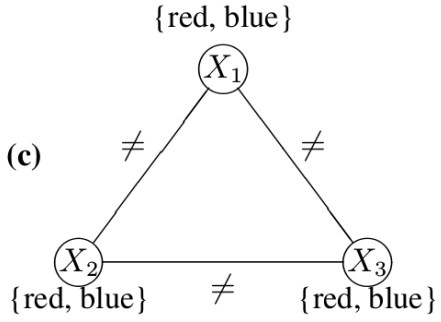
procedure ReviseHR(NG_i, NG_j^*)
repeat
  $NG_i \leftarrow NG_i \bigcup NG_j^*$
  let NG_i^* denote the set of new Nogoods that i can derive from NG_i and his domain using hyper-resolution
  if NG_i^* *is nonempty* **then**
    $NG_i \leftarrow NG_i \bigcup NG_i^*$
    send the Nogoods NG_i^* to all neighbors of i
    if $\{\} \in NG_i^*$ **then**
      stop
until *there is no change in* i*'s set of Nogoods* NG_i

Does the improved version help us?

Filtering using Hyper Resolution

Recall the example:

- x_1 derives $\neg(x_2 = red \wedge x_3 = blue)$ and $\neg(x_2 = blue \wedge x_3 = red)$
- x_2 derives $\neg(x_3 = blue)$ and $\neg(x_3 = red)$
- now $\emptyset$ can be derived in Nogoods of agent x_3



This algorithm generates all Nogoods that are possible.
This is intractable in general.
We need to employ search and backtracking.

Asynchronous Backtracking Algorithm (ABT) – assumptions

- Agents communicate by sending messages
- An agent can send messages to others, iff it knows their identifiers (directed communication / no broadcasting)
- The delay transmitting a message is finite but random
- For any pair of agents, messages are delivered in the order they were sent
- Agents know the constraints in which they are involved, but not the other constraints
- Each agent owns a single variable (agents = variables)
- Constraints are binary (2 variables involved)

Asynchronous Backtracking Algorithm (ABT)

High Level Idea:

- the algorithm makes an ordering on agents and assigns them priority numbers
- a higher-priority agent j informs a lower-priority agent k of its assignment
- lower-priority agent k evaluates the shared C_{jk} constraint with its own assignment
  - if constraints are satisfied with the current assignment $\rightarrow$ no action necessary
  - otherwise, agent k looks for a different value consistent with choice of agent j
  - if such a consistent value exists $\rightarrow$ agent j adopts this value and informs other low-priority agents
  - if such a consistent value does not exist, agent j updates Nogoods and sends the message to agent j

Asynchronous Backtracking (ABT)

Asynchronous action; spontaneous assignment. Four operations:

1. Assignment:j takes value a, j informs all lower priority agents in the neighborhood (**ok?** message).
2. Backtrack (no good): k has no consistent values with higher-priority agents. k resolves nogoods and sends a backtrack (**nogood**) message to the higher-priority agent in the neighborhood with the lowest priority.
3. New links: j receives a nogood mentioning i, however, it is not connected with j. j asks i to set up a link
4. Stop: "no solution" (empty set in nogood) detected by an agent: stop

Solution: when agents are silent for a while (quiescence), every constraint is satisfied $\rightarrow$ solution (detected by specialized algorithms outside ABT)

ABT: Data Structures

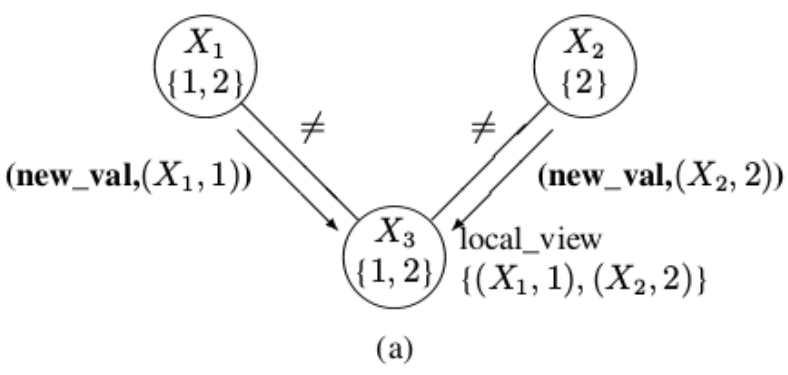
Current context / agent view: values of higher-priority constrained agents

NoGood store: each removed value has a justifying nogood

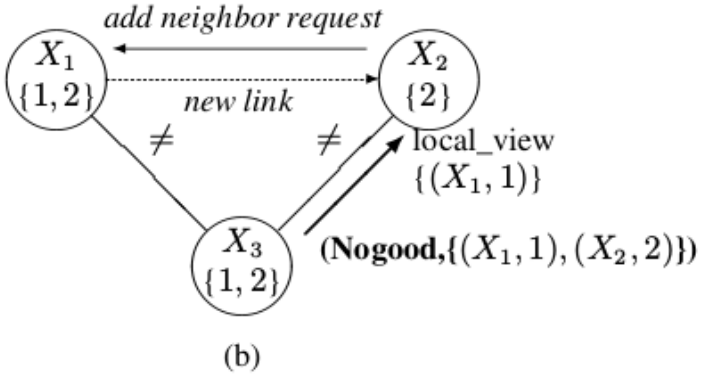
$$x_i = a \wedge x_j = b \Rightarrow x_k \neq c$$

- Stored nogoods must be active: left-hand side of the nogood satisfied in the current context
- If a nogood is no longer active, it is removed (and the value is available again)

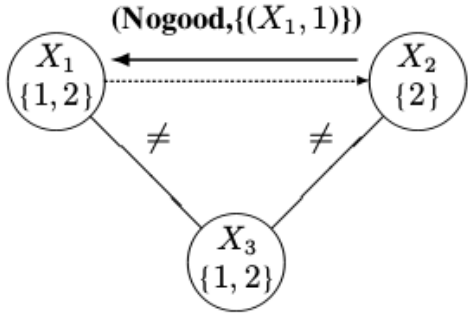
ABT Example



(a)



(b)



ABT Improvements

ABT problem: highly constraint variables can be assigned very late

Solution: Use dynamic priorities

Change **ok?** and **nogood messages** to include agents current priority

Use *min-conflict heuristic*: choose assignment minimizing the number of violations with other agents

We can sacrifice the completeness for the scalability and use locally optimal algorithms (hill-climbing, etc.).