How I Entered Constraints (and Met Ugo at the Doorway): Some of the Early Milestones

Rina Dechter

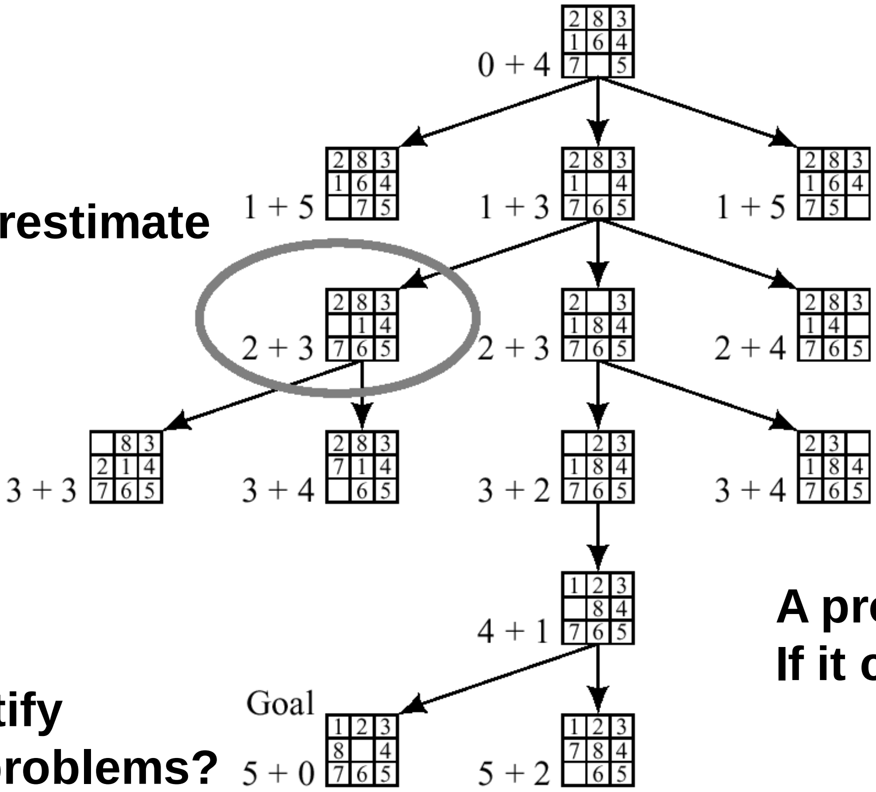
UC-Irvine

Outline

- How I met “Networks of Constraints”
- Networks of Constraints:
  - Sections 1&2 – Motivation and definitions
  - Section 3 – Networks of constraints
  - Section 4 – Path consistency
  - Section 5 – Tractable classes
- Early research after “Networks of Constraints”
- Current research in graphical models and yet another Montanari’s paper.

Mechanical Heuristic Generation

Observation: People generate heuristics by consulting simplified/relaxed models.
Context: Heuristic search (A*) of state-space graph (Nillson, 1980)
Context: Weak methods vs. strong methods
Domain knowledge: Heuristic function



h(n):Heuristic underestimate the best cost from n to the solution

A problem is simple If it can be greedily solved

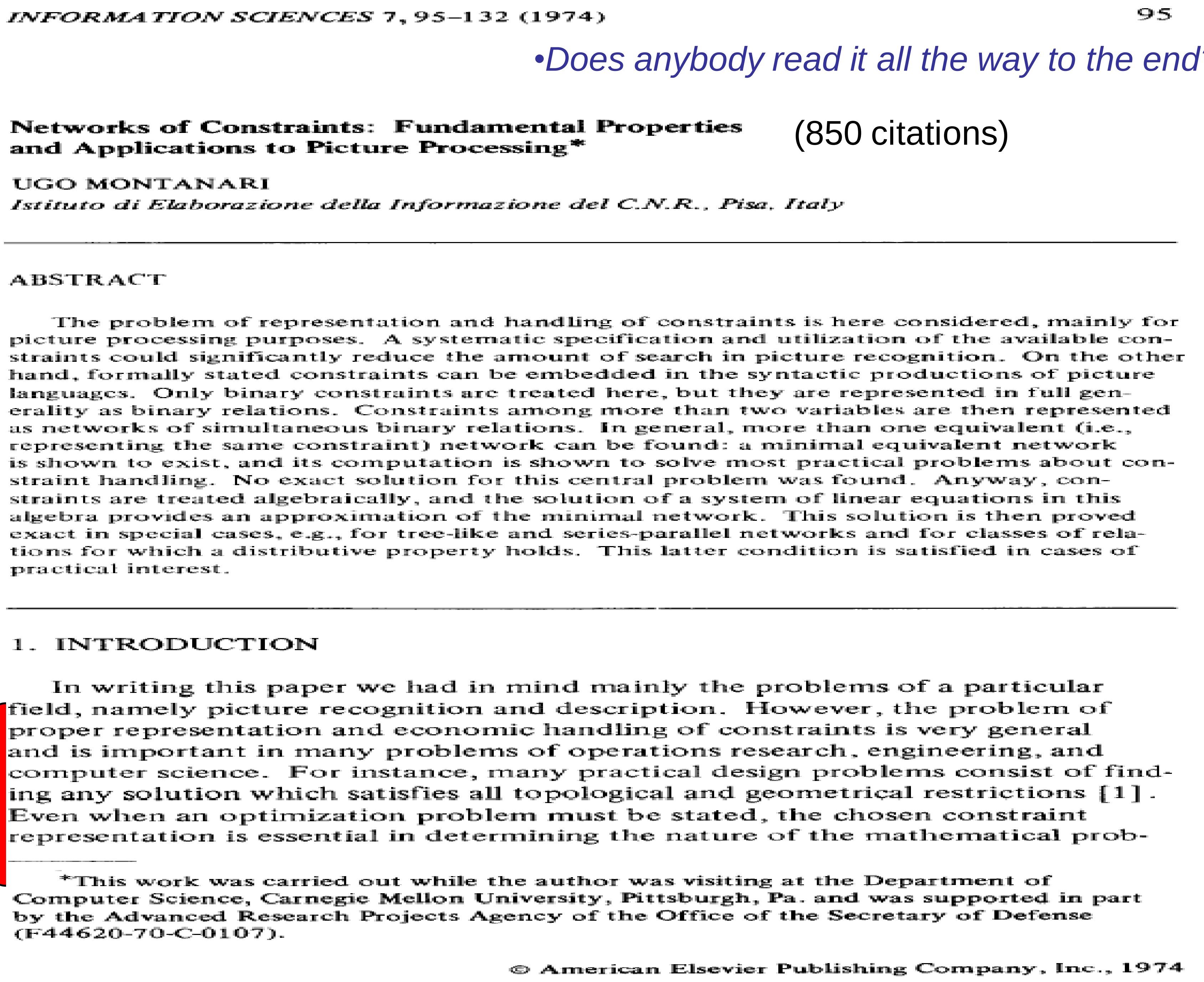
How can we identify Greedily solved problems?

Mechanical Generation of Heuristics pursuit lead to

- Question: How do we identify a greedily solved problems?
- Breakthrough: *A sufficient condition for backtrack-free solution" by Gene Freudeder (JACM, 1982)" (pages 24-32)*
- *A. Mackworth "Consistency in networks of relations" 1977 Pages 99-118, Artificial Intelligence*
- *"Montanari, U. Networks of constraints: Fundamental properties and applications to picture processing" Information Science, 1974."*

•Does anybody read it all the way to the end?

(850 citations)



Networks of Constraints: Fundamental Properties and Applications to Picture Processing*

UGO MONTANARI
Istituto di Elaborazione della Informazione del C.N.R., Pisa, Italy

ABSTRACT

The problem of representation and handling of constraints is here considered, mainly for picture processing purposes. A systematic specification and utilization of the available constraints could significantly reduce the amount of search in picture recognition. On the other hand, formally stated constraints can be embedded in the syntactic productions of picture languages. Only binary constraints are treated here, but they are represented in full generality as binary relations. Constraints among more than two variables are then represented as networks of simultaneous binary relations. In general, more than one equivalent (i.e., representing the same constraint) network can be found: a minimal equivalent network is shown to exist, and its computation is shown to solve most practical problems about constraint handling. No exact solution for this central problem was found. Anyway, constraints are treated algebraically, and the solution of a system of linear equations in this algebra provides an approximation of the minimal network. This solution is then proved exact in special cases, e.g., for tree-like and series-parallel networks and for classes of relations for which a distributive property holds. This latter condition is satisfied in cases of practical interest.

1. INTRODUCTION

In writing this paper we had in mind mainly the problems of a particular field, namely picture recognition and description. However, the problem of proper representation and economic handling of constraints is very general and is important in many problems of operations research, engineering, and computer science. For instance, many practical design problems consist of finding any solution which satisfies all topological and geometrical restrictions [1]. Even when an optimization problem must be stated, the chosen constraint representation is essential in determining the nature of the mathematical prob-

*This work was carried out while the author was visiting at the Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pa. and was supported in part by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-70-C-0107).

Section 2: Mathematical Notation

- Constraints as relations
- Characteristic matrix,
- union, intersection **Composition:**

x2	x1
a	b
a	c
b	a
c	a

Scene Labeling Constraint Network

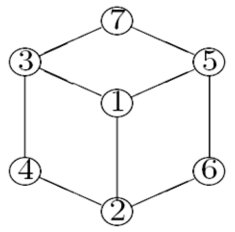
(**Waltz, 1972. Generating semantidescriptions from drawings of scenes with shadows.**)

$$R_{21} = \begin{array}{c} \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \quad R_{31} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad R_{51} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_{13} = R_{12} \cdot R_{23} \text{ iff } R_{13,rs} = \bigvee_{t=1}^{N_2} R_{12,rt} \wedge R_{23,ts}.$$

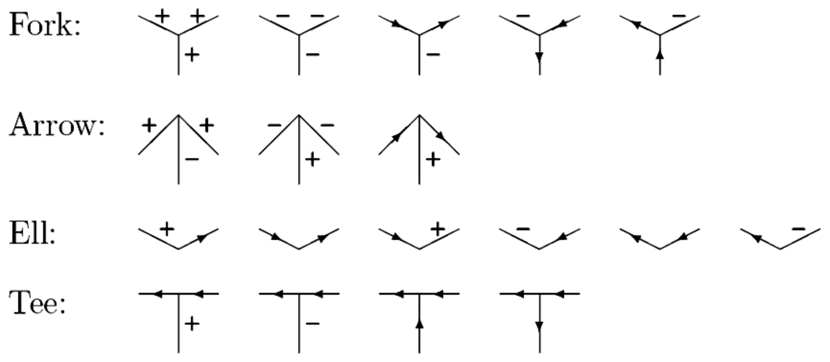
Note that composition, in matrix notation, is just binary matrix multiplication. For example, we may have

$$R_{12} = \begin{vmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \end{vmatrix}; R_{23} = \begin{vmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{vmatrix}; R_{13} = R_{12} \cdot R_{23} = \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}.$$



$$R_{24} = R_{37} = R_{56} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$R_{26} = R_{34} = R_{57} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$



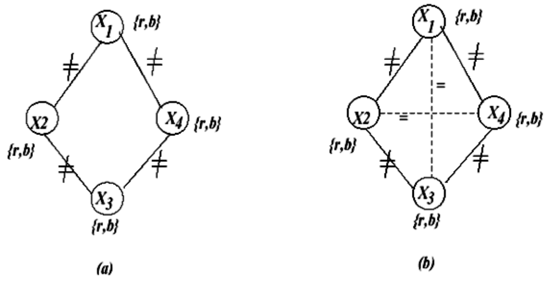


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

Constraint Networks

A

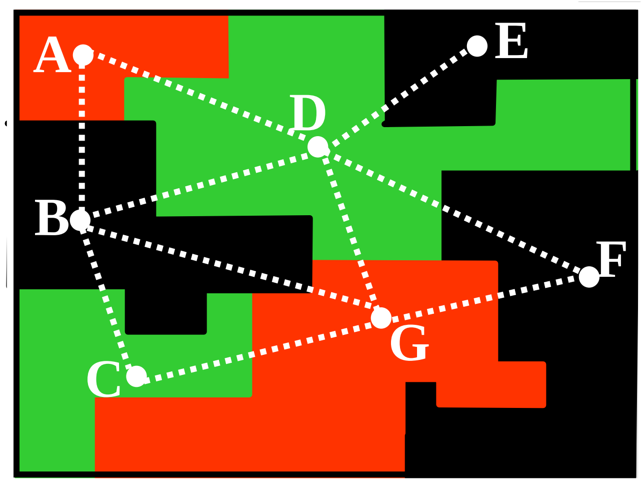
Example: map coloring

Variables - countries (A,B,C,etc.)

Values - colors (red, green, blue)

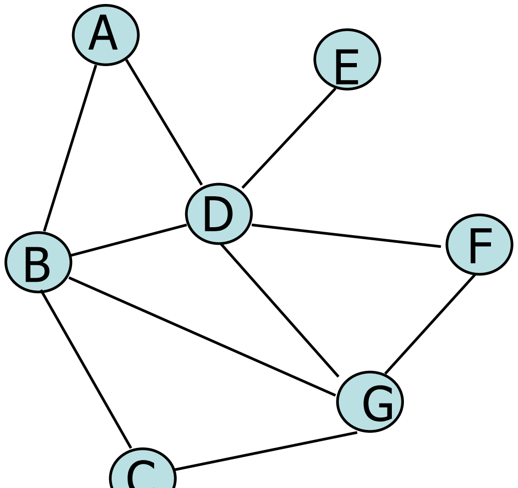
Constraints: $A \neq B$, $A \neq D$, $D \neq E$, *etc.*

A	B
red	green
red	yellow
green	red
green	yellow
yellow	green
yellow	red



Semantics: set of all solutions

Constraint graph



Section 3: Networks of Constraints

The section defines constraint networks, showing that:

- **Expressivness**: Not every relation can be expressed as a network of binary constraints
- Defining The **projection network**
- The **minimal network**
- The intricate concept of **decomposability**
- The central problem: **Computing the minimal network**.

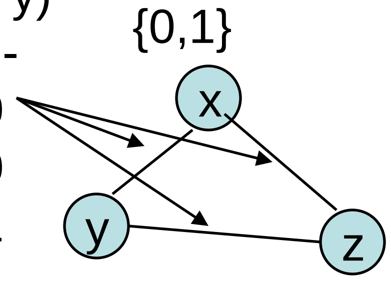
Section 3: Networks of Constraints

The section defines constraint networks, showing that:

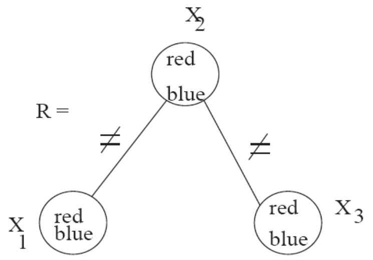
- **Expressiveness**: Not every relation can be expressed with binary constraints.
- The **projection network** is the best binary network approximation.
- The **minimal network** is the intersection of all equivalent constraint networks
- The **minimal network is unique** and identical to the projection network and thus the most explicit representation,
- **Decomposability**: a relation may be represented by a network of binary constraints but its projection may not. (Namely, it can never become backtrack-free with binary constraints only).
- **The central problem:** Computing the minimal network.

x y z
0 0 0
1 0 1
0 1 1
1 1 0

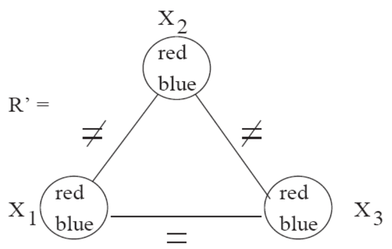
R(x y)
0 0
1 0
0 1
1 1



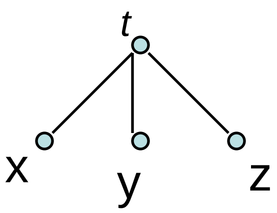
x y z
0 0 0
1 0 0
0 1 0
0 0 1
1 0 1
0 1 1
1 1 0
1 1 1



a



b

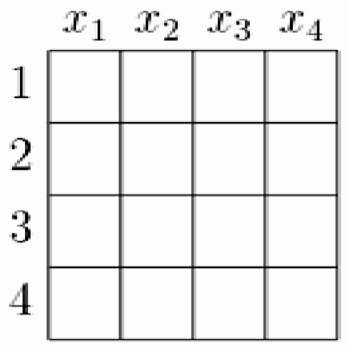


$$\rho = \begin{pmatrix} x & y & z & t \\ \hline a & a & a & a \\ a & b & b & b \\ b & b & a & c \end{pmatrix}$$

$$\pi_{xyz}\rho = \begin{pmatrix} x & y & z \\ \hline a & a & a \\ a & b & b \\ b & b & a \end{pmatrix}$$

a b a

The 4-Queen Problem
(specification (a) and minimal network (b))

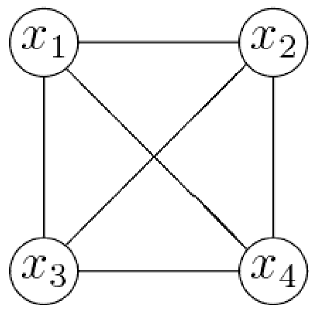


$R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$
$R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$
$R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)$
$(4,2), (4,3)\}$
$R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$
$R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$
$R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$

(2,4,1,3)
(3,1,4,2)

(a)

(b)

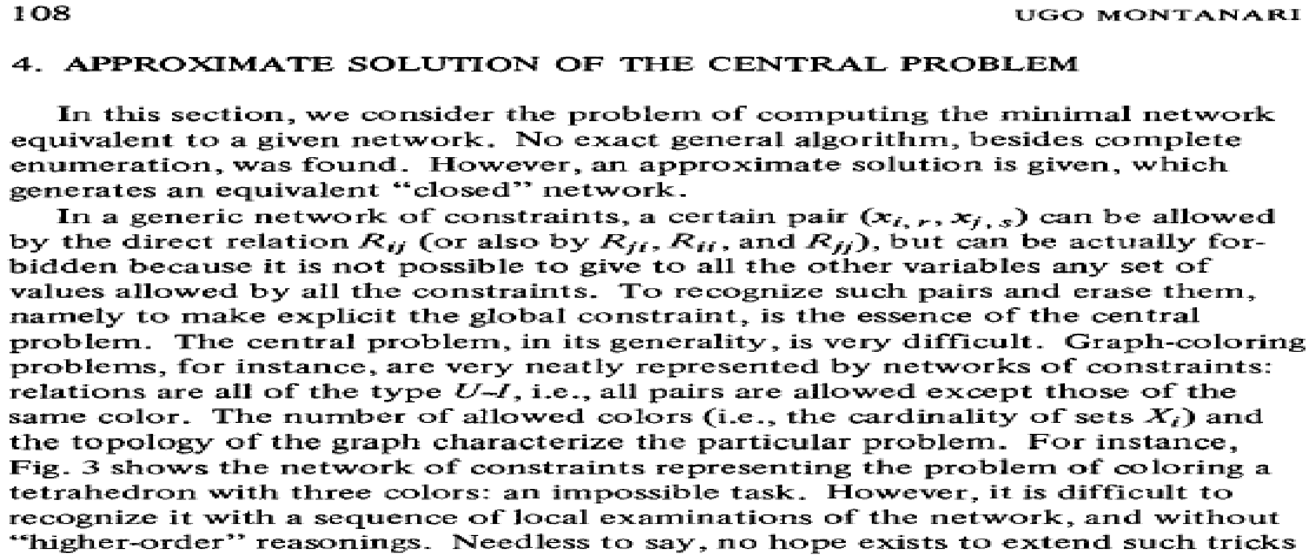


$M_{12} = \{(2,4), (3,1)\}$
$M_{13} = \{(2,1), (3,4)\}$
$M_{14} = \{(2,3), (3,2)\}$
$M_{23} = \{(1,4), (4,1)\}$
$M_{24} = \{(1,2), (4,3)\}$
$M_{34} = \{(1,3), (4,2)\}$

$D_1 = \{1,3\}$
$D_2 = \{1,4\}$
$D_3 = \{1,4\}$
$D_4 = \{1,3\}$

(a) (b) (c)

Section 4: Approximate Solution of the Central Problem



4. APPROXIMATE SOLUTION OF THE CENTRAL PROBLEM

In this section, we consider the problem of computing the minimal network equivalent to a given network. No exact general algorithm, besides complete enumeration, was found. However, an approximate solution is given, which generates an equivalent "closed" network.

In a generic network of constraints, a certain pair $(x_{i,r}, x_{j,s})$ can be allowed by the direct relation R_{ij} (or also by R_{ji}, R_{ii}, and R_{jj}), but can be actually forbidden because it is not possible to give to all the other variables any set of values allowed by all the constraints. To recognize such pairs and erase them, namely to make explicit the global constraint, is the essence of the central problem. The central problem, in its generality, is very difficult. Graph-coloring problems, for instance, are very neatly represented by networks of constraints: relations are all of the type U–I, i.e., all pairs are allowed except those of the same color. The number of allowed colors (i.e., the cardinality of sets X_i) and the topology of the graph characterize the particular problem. For instance, Fig. 3 shows the network of constraints representing the problem of coloring a tetrahedron with three colors: an impossible task. However, it is difficult to recognize it with a sequence of local examinations of the network, and without "higher-order" reasonings. Needless to say, no hope exists to extend such tricks

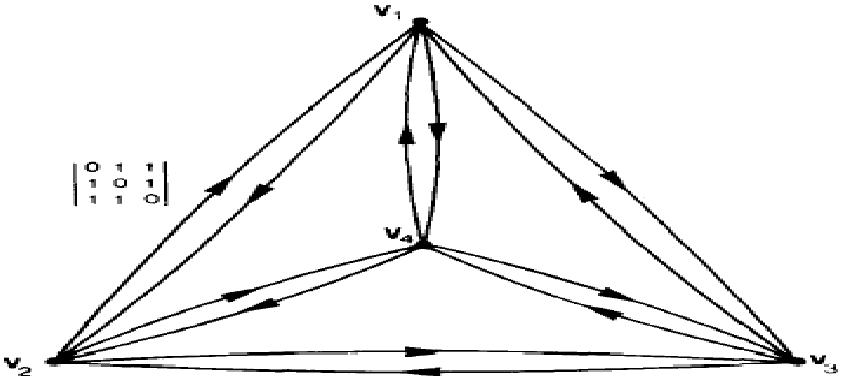
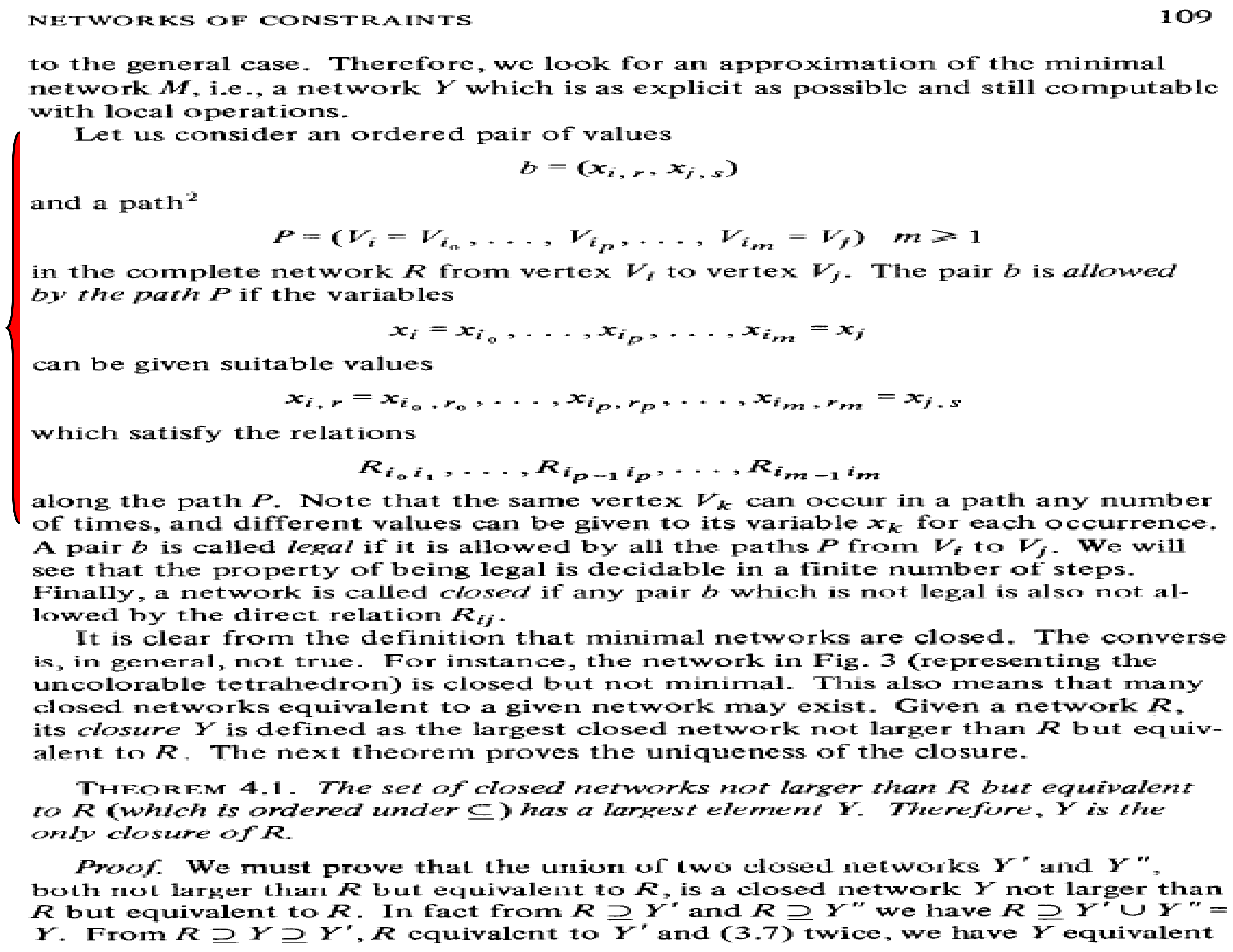


Fig. 3. In this network, the relation $\begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{vmatrix}$ is associated to every arc. This network represents the impossible problem of coloring a four-vertex complete graph with three colors. This network is symmetric and closed but not minimal.

"*Therefore we look for an approximation of the minimal network which is as explicit as possible and still computable with local operations*"

Defining Closure (e.g., Path-consistency)



to the general case. Therefore, we look for an approximation of the minimal network M, i.e., a network Y which is as explicit as possible and still computable with local operations.

Let us consider an ordered pair of values

$$b = (x_{i,r}, x_{j,s})$$

and a path[2]

$$P = (V_i = V_{i_0}, \ldots, V_{i_p}, \ldots, V_{i_m} = V_j) \quad m \geqslant 1$$

in the complete network R from vertex V_i to vertex V_j. The pair b is *allowed by the path P* if the variables

$$x_i = x_{i_0}, \ldots, x_{i_p}, \ldots, x_{i_m} = x_j$$

can be given suitable values

$$x_{i,r} = x_{i_0,r_0}, \ldots, x_{i_p,r_p}, \ldots, x_{i_m,r_m} = x_{j,s}$$

which satisfy the relations

$$R_{i_0 i_1}, \ldots, R_{i_{p-1} i_p}, \ldots, R_{i_{m-1} i_m}$$

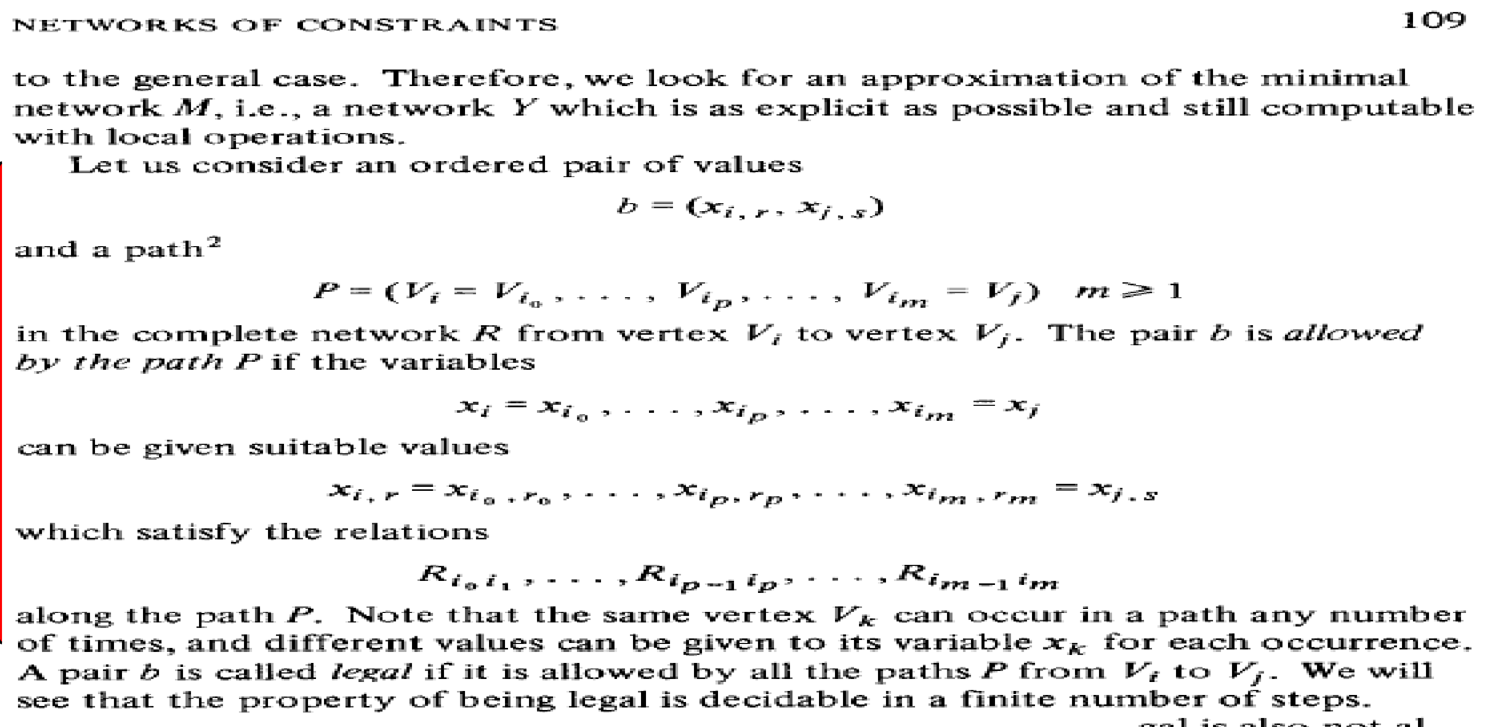
along the path P. Note that the same vertex V_k can occur in a path any number of times, and different values can be given to its variable x_k for each occurrence. A pair b is called *legal* if it is allowed by all the paths P from V_i to V_j. We will see that the property of being legal is decidable in a finite number of steps. Finally, a network is called *closed* if any pair b which is not legal is also not allowed by the direct relation R_{ij}.

It is clear from the definition that minimal networks are closed. The converse is, in general, not true. For instance, the network in Fig. 3 (representing the uncolorable tetrahedron) is closed but not minimal. This also means that many closed networks equivalent to a given network may exist. Given a network R, its *closure* Y is defined as the largest closed network not larger than R but equivalent to R. The next theorem proves the uniqueness of the closure.

THEOREM 4.1. *The set of closed networks not larger than R but equivalent to R (which is ordered under $\subseteq$) has a largest element Y. Therefore, Y is the only closure of R.*

Proof. We must prove that the union of two closed networks Y' and Y'', both not larger than R but equivalent to R, is a closed network Y not larger than R but equivalent to R. In fact from $R \supseteq Y'$ and $R \supseteq Y''$ we have $R \supseteq Y' \cup Y'' = Y$. From $R \supseteq Y \supseteq Y'$, R equivalent to Y' and (3.7) twice, we have Y equivalent

Defining Closure (e.g., Path-consistency)



to the general case. Therefore, we look for an approximation of the minimal network M, i.e., a network Y which is as explicit as possible and still computable with local operations.

Let us consider an ordered pair of values

$$b = (x_{i,r}, x_{j,s})$$

and a path[2]

$$P = (V_i = V_{i_0}, \ldots, V_{i_p}, \ldots, V_{i_m} = V_j) \quad m \geqslant 1$$

in the complete network R from vertex V_i to vertex V_j. The pair b is *allowed by the path P* if the variables

$$x_i = x_{i_0}, \ldots, x_{i_p}, \ldots, x_{i_m} = x_j$$

can be given suitable values

$$x_{i,r} = x_{i_0,r_0}, \ldots, x_{i_p,r_p}, \ldots, x_{i_m,r_m} = x_{j,s}$$

which satisfy the relations

$$R_{i_0 i_1}, \ldots, R_{i_{p-1} i_p}, \ldots, R_{i_{m-1} i_m}$$

along the path P. Note that the same vertex V_k can occur in a path any number of times, and different values can be given to its variable x_k for each occurrence. A pair b is called *legal* if it is allowed by all the paths P from V_i to V_j. We will see that the property of being legal is decidable in a finite number of steps.

gal is also not al-

osed. The converse
presenting the
means that many
en a network R,
r than R but equiv-
osure.

R but equivalent
refore, Y is the

ks Y' and Y'',
k Y not larger than
ave $R \supseteq Y' \cup Y'' =$
have Y equivalent

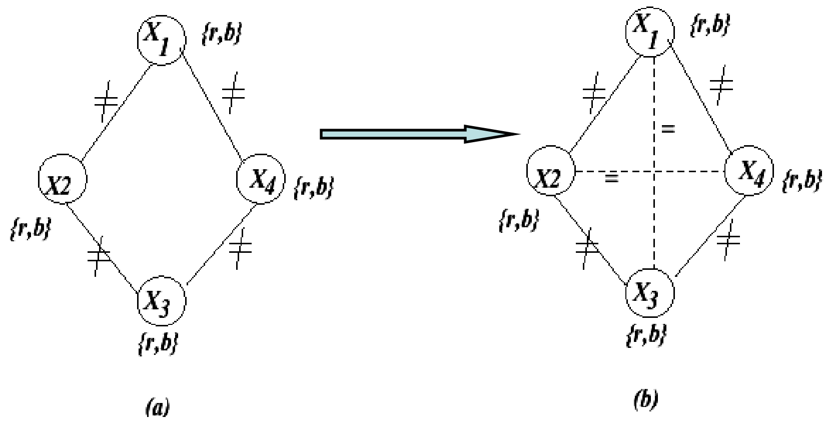


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

A Closure Algorithm (Path-Consistency)

Complexity:	
PC-1:	$O(n^5k^5)$
PC-2:	$O(n^3k^5)$
PC-4 optimal:	$O(n^3k^3)$

The closure Y of a network R can be characterized as being the solution of the following system of equations.

$$Y_{ij} = \sum_{k=1}^{n} R_{ik} Y_{kj} + d_{ij} \qquad (4.1)$$

where

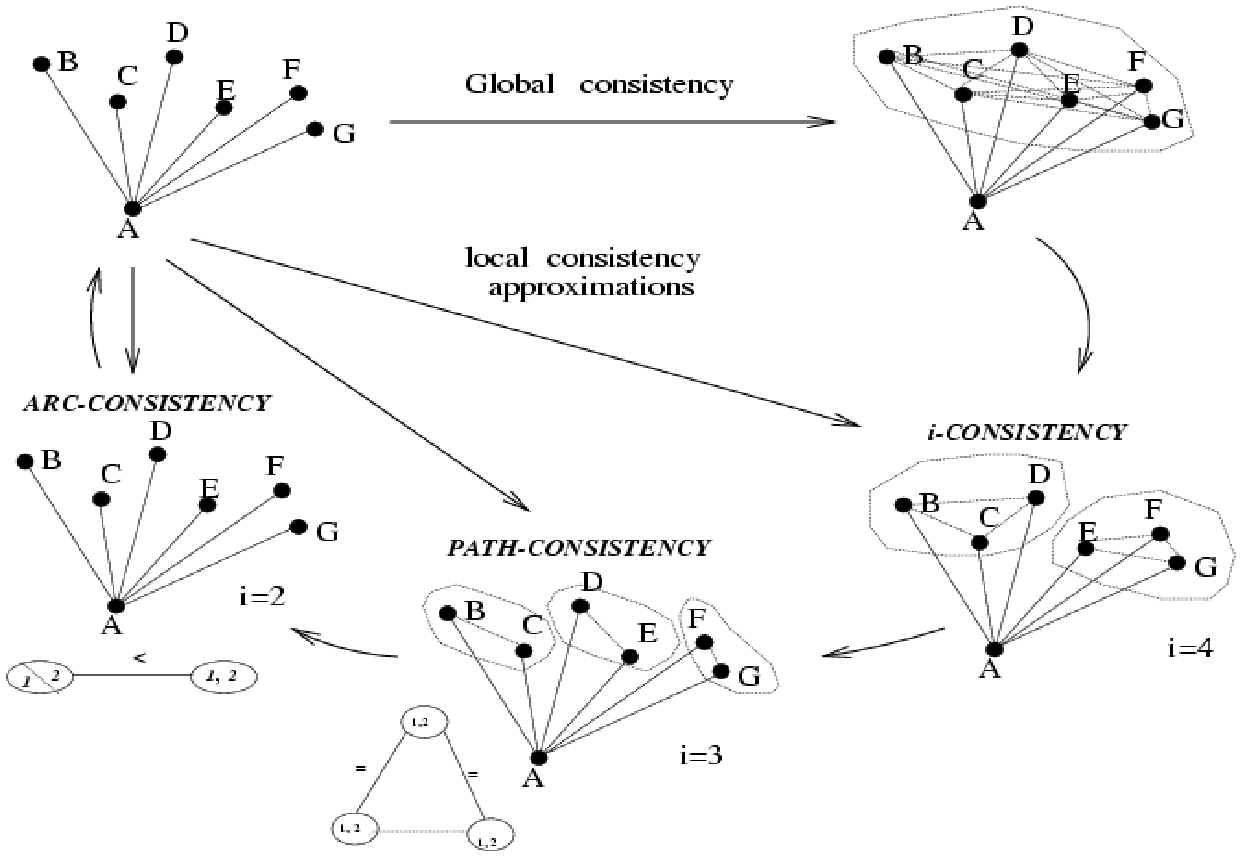
$$d_{ij} = I_{ij} \text{ if } i = j;\ d_{ij} = U_{ij} \text{ otherwise.}$$

Given a network R with n vertices we can give an algorithm for computing its closure Y.

```
Algorithm C
Step 1  $Y^0 = R$.
Step 2  Execute next step for $k = 1, \ldots, n$.
Step 3  $Y_{ij}^{k} = Y_{ij}^{k-1} + Y_{ik}^{k-1} Y_{kk}^{k-1} Y_{kj}^{k-1}$   $(i, j = 1, \ldots, n)$.   (4.4)
Step 4  If $Y^n \neq Y^0$ then let $Y^0 = Y^n$ and go to Step 2; else let $Y = Y^n$ and stop.
```

THEOREM 4.4. *Algorithm C computes the closure Y of R. In particular, if $Y_{ij,rs}^{n} = 1$ in the network Y^n obtained at the end of the first iteration, then pair $(x_{i,r}, x_{j,s})$ is allowed by all the paths from V_i to V_j in R.*

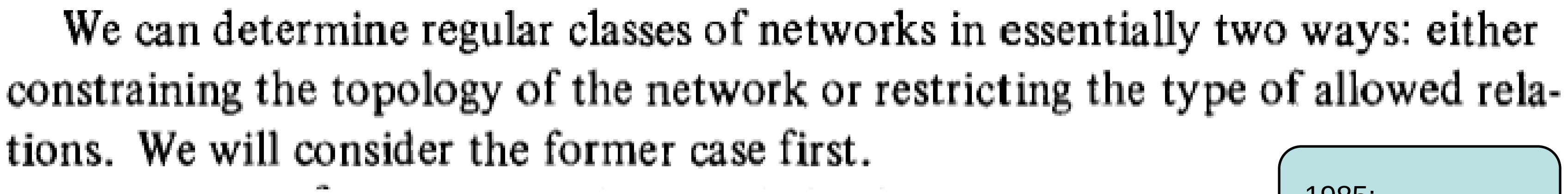
From Global to Local Consistency



Mackworth: Consistency in networks of relations (1977) (850 citations).
Freuder: Synthesizing Constraint Expressions" (1977) (420 citations).

Section 5: Exact Solutions for the Central Problem for Particular Classes of Networks

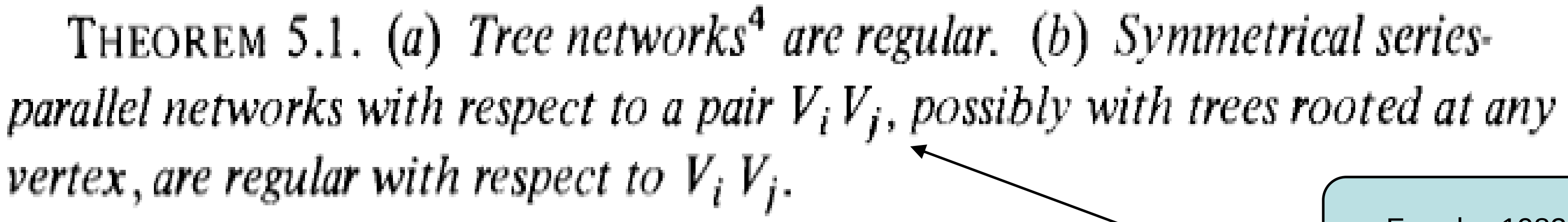
- **Regular networks:** when the path-consistent network is minimal



We can determine regular classes of networks in essentially two ways: either constraining the topology of the network or restricting the type of allowed relations. We will consider the former case first.

- **Topology-based:**

1985:
Mackworth & Freuder



THEOREM 5.1. (a) *Tree networks*[4] *are regular.* (b) *Symmetrical series-parallel networks with respect to a pair* $V_i V_j$, *possibly with trees rooted at any vertex, are regular with respect to* $V_i V_j$.

Freuder 1982
Dechter 1987

- **Restriction on allowed relations:**
  - **He defines distributed and star-distributed classes**

Distributive Networks (section 5)

- A network is **distributive** if composition is distributive over intersection.

$$R_{12}(R'_{23} + R''_{23}) \neq R_{12}R'_{23} + R_{12}R''_{23}.$$

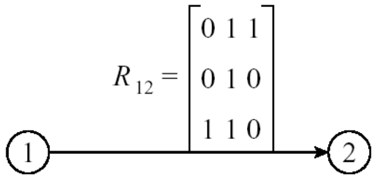
THEOREM 5.2. *A closed, distributive network Y is decomposable. Furthermore, its symmetrization*

$$Y'_{ij} = Y_{ij} + Y^T_{ji}\,(i, j = 1, \ldots, n)$$

is minimal. Thus in particular if Y is symmetric, it is minimal.

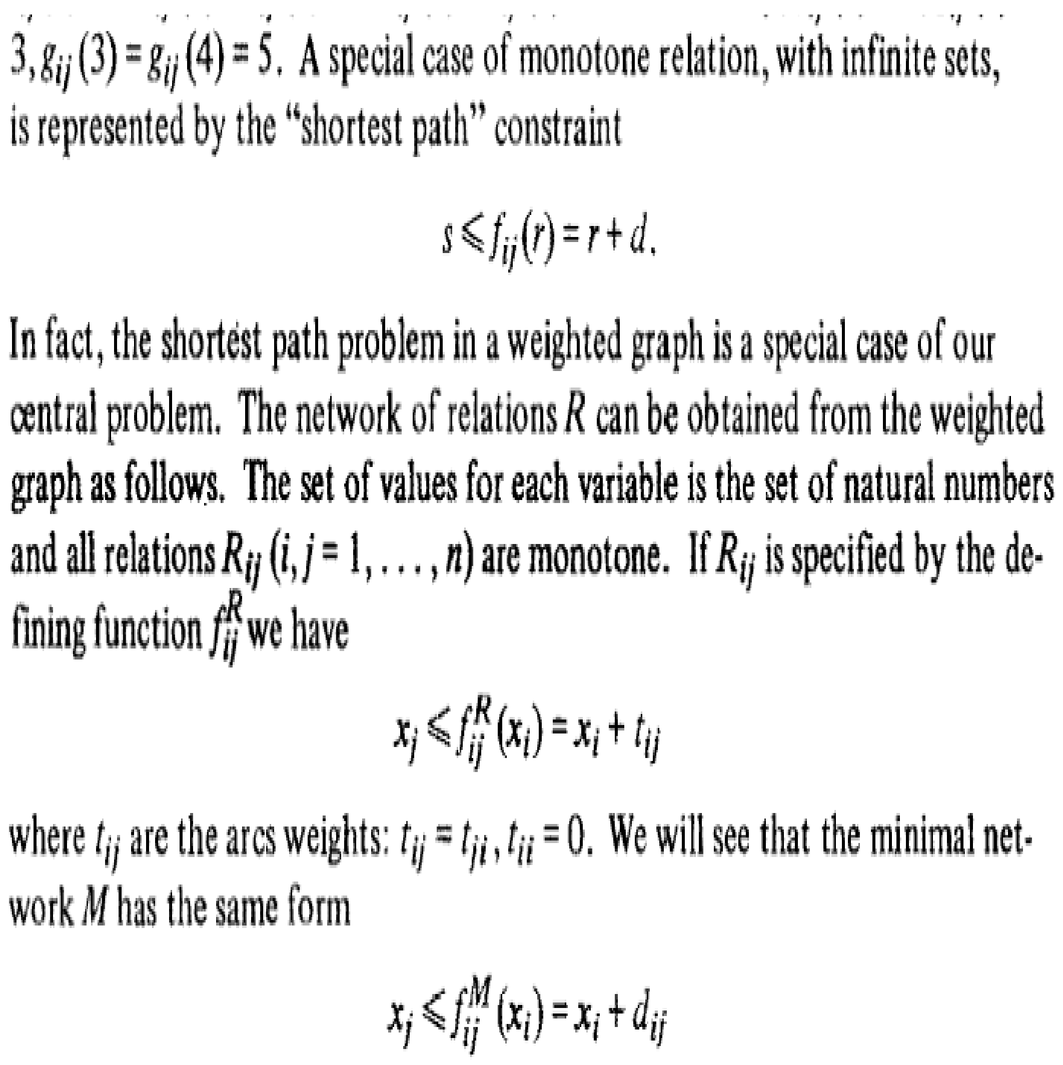
- **Theorem 5.3: "star distributivity" requires one iteration of closure algorithm.**
- Algorithm C (in case of star distributivity) is the Gausian elimination algorithm of the system equation (4.1)). Similar to the **Floyd Warshal algorithm** for all pairs shortest path in a graph

- **What relations are (star) distributive?**
  - Monotone,
  - Row convex (van Beek 1995)



$R_{12} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$
1
2

Simple Linear Constraints



3, $g_{ij}(3) = g_{ij}(4) = 5$. A special case of monotone relation, with infinite sets, is represented by the "shortest path" constraint

$$s \leqslant f_{ij}(r) = r + d.$$

In fact, the shortest path problem in a weighted graph is a special case of our central problem. The network of relations R can be obtained from the weighted graph as follows. The set of values for each variable is the set of natural numbers and all relations R_{ij} $(i, j = 1, \dots, n)$ are monotone. If R_{ij} is specified by the defining function f_{ij}^R we have

$$x_j \leqslant f_{ij}^R(x_i) = x_i + t_{ij}$$

where t_{ij} are the arcs weights: $t_{ij} = t_{ji}$, $t_{ii} = 0$. We will see that the minimal network M has the same form

$$x_j \leqslant f_{ij}^M(x_i) = x_i + d_{ij}$$

Temporal Constraint Networks

(Dechter, Meiri and Pearl 1990, 971 citations)

$$a_{ij} \leq x_j - x_i \leq b_{ij}.$$

Alternatively, the constraint can be expressed as a pair of inequalities:

$$x_j - x_i \leq b_{ij}, \;\; and \;\; x_i - x_j \leq -a_{ij}.$$

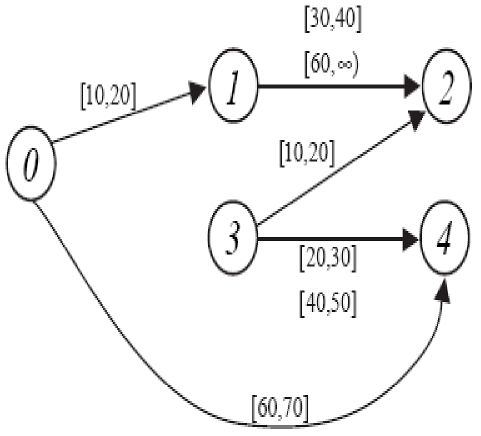


Figure 12.8: A constraint graph representing Example 12.2.1.

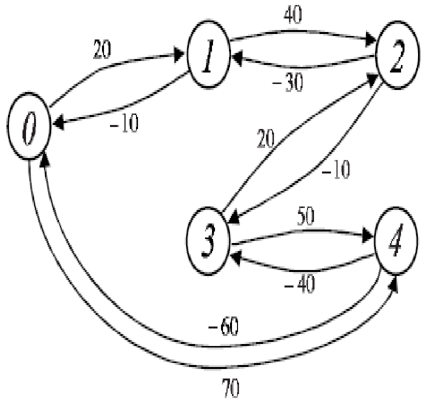


Figure 12.10: A distance graph representing a portion of Example 12.2.1.

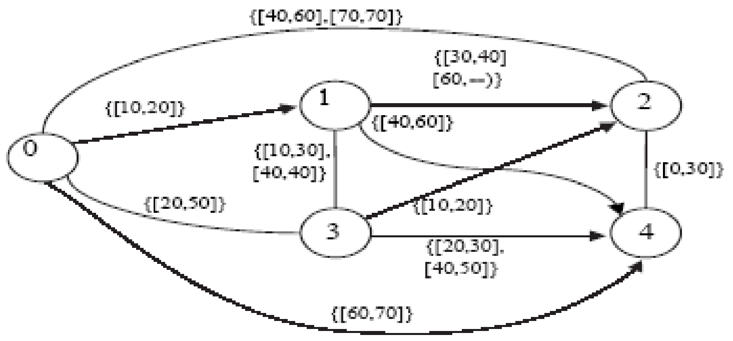


Figure 12.12: The minimal network of Example 12.2.1.

CONCLUSION

In this paper we have presented a formal treatment of networks of binary constraints. The main practical result was the discovery of an algorithm for adding to the direct constraint between each pair of variables the indirect constraints transmitted by all the paths in the network. In particular cases the resulting constraint was proved equivalent to the global constraint represented by the entire network as seen by that pair of vertices. This result allows the partial or total utilization of the global constraint structure for reducing the set of feasible values of a variable to be determined, when the values of other variables are known.

For the practical computer implementation of this method, the following requirements can be suggested:

(a) In the application under examination, most constraints must be reasonably represented or approximated by binary constraints or simple networks of binary constraints. Note that if we allow a constraint among m variables to be represented by a network of n vertices, with $n > m$, then the negative result of Section 3 no longer holds, and many representations of the constraint, trivial and not, can be found. For instance, the ternary relation (3.4) which is not representable with a 3-vertex network, can be represented by the 4-vertex network in Fig. 2, as seen from vertices V_1, V_2, and V_3.

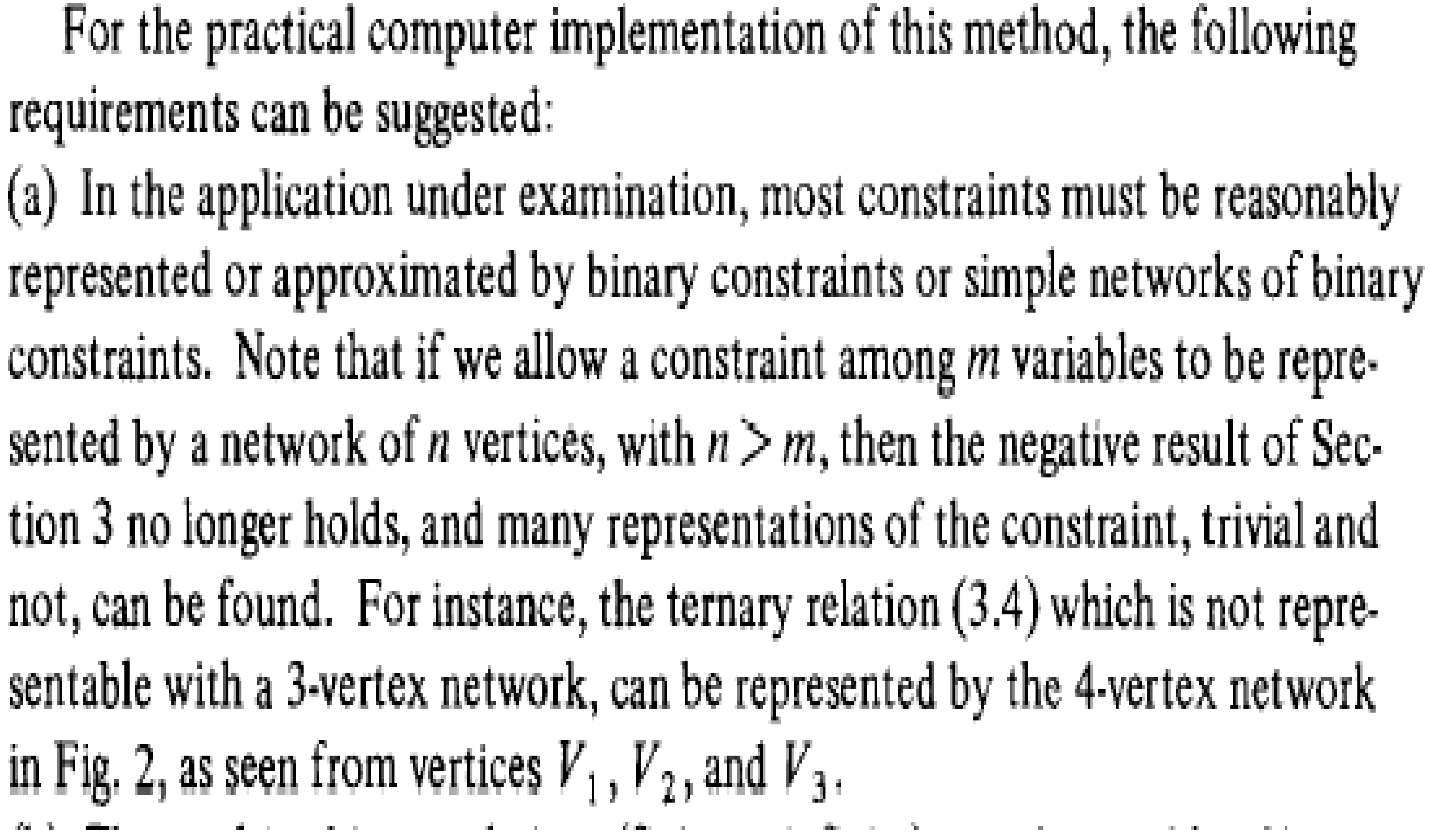
(b) The resulting binary relations (finite or infinite) must be capable of being stored in an economical way in a computer memory. For instance, if the variables are points of m-dimensional spaces, a relation R_{ij} could be stored representing the images in X_j of all elements[7] $x_{i,r}$ of X_i as m-dimensional domains. Known techniques of domain encoding can then be used. For instance, two given points are sufficient for determining a rectangular domain: this is often the meaning of functions $f_{ij}(r)$ and $g_{ji}(r)$ representing a monotone relation.

(c) The operations of intersection and composition must be easily definable in the chosen class of relations. In particular, this class must be closed under those two operations. For instance, this is the case of relations represented by domains, convex domains, domains enclosed by polygons or convex polygons, rectangular domains.

(d) The closed network is then obtained with algorithm C. The closed network should then be close to the minimal. For instance, we have coincidence for rectangular domains, and we expect reasonable closeness for convex domains. Bad results can be expected if the relations allow most pairs and forbide a few isolated pairs, like in graph-coloring problems. Anyway, if the addition of a further constraint destroys regularity (i.e., closed $\neq$ minimal), it is, nevertheless,

[7]Or just one, if all the other images can be obtained from it by a fixed procedure (e.g., translation).

Using Hidden variables



For the practical computer implementation of this method, the following requirements can be suggested:

(a) In the application under examination, most constraints must be reasonably represented or approximated by binary constraints or simple networks of binary constraints. Note that if we allow a constraint among m variables to be represented by a network of n vertices, with $n > m$, then the negative result of Section 3 no longer holds, and many representations of the constraint, trivial and not, can be found. For instance, the ternary relation (3.4) which is not representable with a 3-vertex network, can be represented by the 4-vertex network in Fig. 2, as seen from vertices V_1, V_2, and V_3.

ON THE EXPRESSIVENESS OF NETWORKS WITH HIDDEN VARIABLES

Rina Dechter [1]

Computer Science Department
Technion -- Israel Institute of Technology
Haifa, Israel, 32000
e-mail: dechter@techsel.bitnet

Abstract

This paper investigates design issues associated with representing relations in binary networks augmented with hidden variables. The trade-off between the number of variables required and the size of their domains is discussed. We show that if the number of values available to each variable is just two, then hidden variables cannot improve the expressional power of the network, regardless of their number. However, for $k \geq 3$, we can always find a layered network using k-valued hidden variables that represent an arbitrary relation. We then provide a scheme for decomposing an arbitrary relation, ρ, using $\frac{|\rho|-2}{k-2}$ hidden variables, each having k values ($k>2$).

1. Introduction

Hidden units play a central role in connectionist model, without which the model would not represent many useful functions and relations. In the early days of the Perceptrons [Minsky 1969] it was noted that even simple functions like the *XOR* were not expressible in a single layer perceptron; a realization that slowed research in the area until the notion of hidden units had emerged [Rumelhart 1988a, Hinton 1988]. Nevertheless, a formal treatment of the expressiveness gained by hidden units, and systematic schemes for designing systems with hidden units within the neural network paradigm are still not available.

Our intention is to investigate formally the role of hidden units and devise systematic schemes for designing systems incorporating hidden units. Specifically, we address the following task: given a relation on n variables, called visible, we wish to design a network having $n+h$ units whose stable patterns, (relative to the visible units) coincide with the original relation. This task is central to most applications of connectionist networks, in particular to its role as associative memory. The task will be investigated for a connectionist architecture which is different from classic connectionist networks in that it is based on **constraint networks.** The sequential constraint network model is defined next.

(1) This research was supported in part by NSF grant #IRI-8821444 and by an Air Force grant #AFOSR 88 0177 while the author was visiting the cognitive systems lab at UCLA.

A Network of binary constraints involves a set of n variables $X_1,...,X_n$, each represented by its domain values, $D_1, \ldots, D_n$, and a set of constraints. A **binary constraint** R_{ij} between two variables X_i and X_j is a subset of the cartesian product $D_i \times D_j$ that specifies which values of the variables are compatible with each other. A solution is an assignment of values to all the variables which satisfy all the constraints, and the **constraint satisfaction problems (CSP)** associated with these networks is to find one or all solutions. A binary CSP can be associated with a **constraint-graph** in which nodes represent variables and arcs connect pairs of variables which are constrained explicitly. Figure 1a presents a constraint network where each node represents a variable having values $\{a, b, c\}$ and each link is associated with a strict lexicographic order (where $X_i < X_j$ iff $i < j$). (The domains and the constraints explicitly indicated on some of the links.)

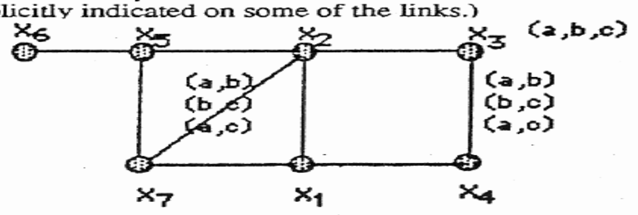


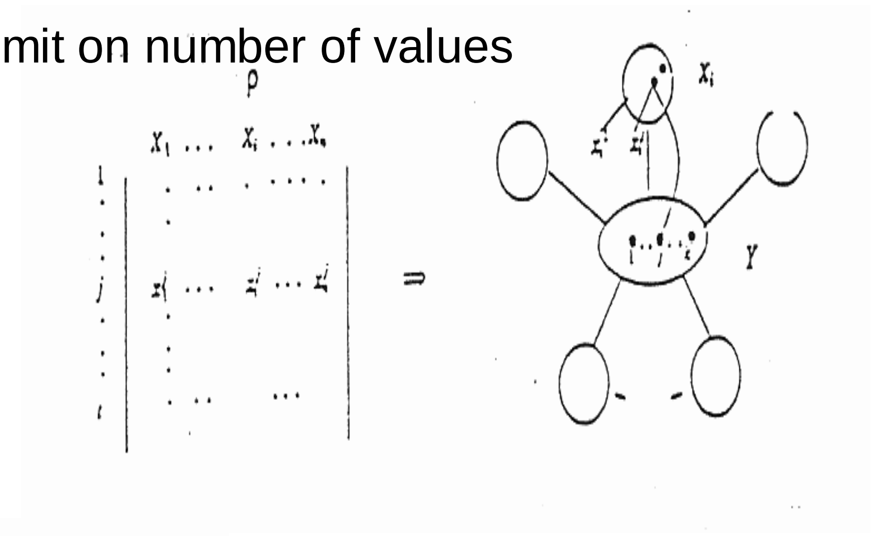
Figure 1: An example of a binary CN

Our constraint-based connectionist architecture assumes that each unit plays the role of a variable having k states, and that the links, representing the constraints, are quantified by compatibility relations between states of adjacent units. Each unit asynchronously updates its state

On the expressiveness of networks with hidden variables

Can a relation be expressed by a binary constraint networks with hidden variables?

Yes. If no limit on number of values



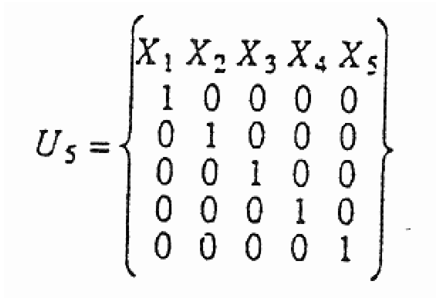
And, with limit?

$$U_5 = \left\{ \begin{array}{ccccc} X_1 & X_2 & X_3 & X_4 & X_5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right\}$$

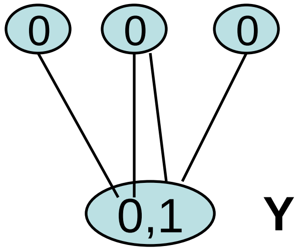
With 2 values?
With 3 values?
How many hidden
Variables?

On the expressiveness of networks with hidden variables

Theorem: Relations which are not binary network decomposable cannot be binary network decomposable by adding **any number of bi-valued** Hidden variables.
Proof: We want to exclude (x1,x2,x3)=(0,0,0) using a variable Y={0,1}. …



$$U_5 = \left\{ \begin{array}{ccccc} X_1 & X_2 & X_3 & X_4 & X_5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right\}$$



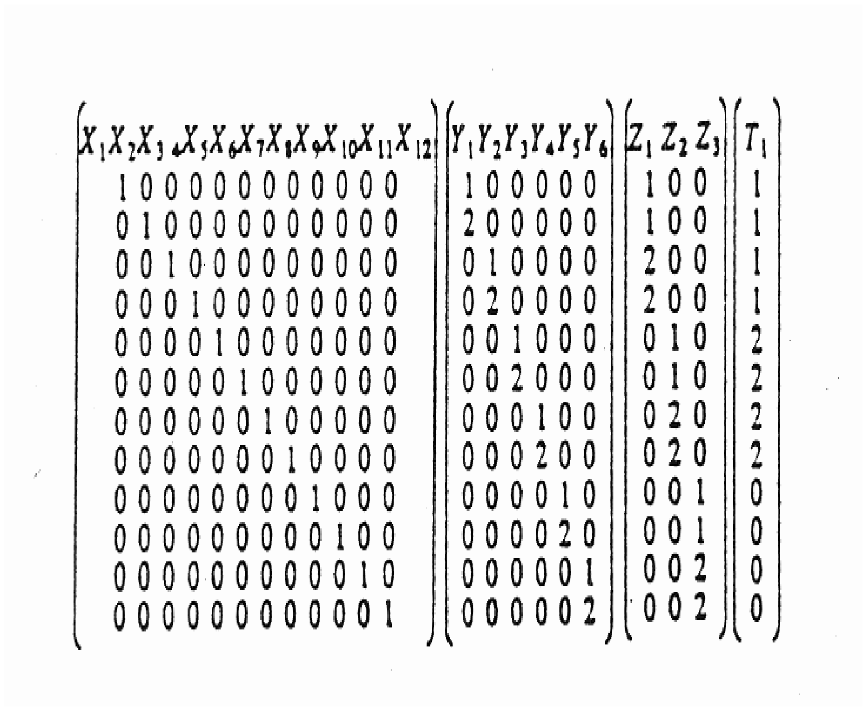
So, it is not possible that Y allows any pair but not triplets.

Reason: 3-consistent bi-valued binary networks are globally consistent

Theorem (Dechter, 1992): k-valued binary networks which are strong (k+1)-consistent are globally consistent (decomposable).

Semantic based tractability: row-convex constraints (van Beek 1995)
A whole major line of work by Jeavons and Cohen (1995-2007)
(Constraint Processing, chapter 10, 2003)

On the expressiveness of networks with hidden variables



X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Z_1	Z_2	Z_3	T_1
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	2	0	0	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	2
0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0	2
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	2	0	2
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0	0	0	2	0	2
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	2	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	0	2	0

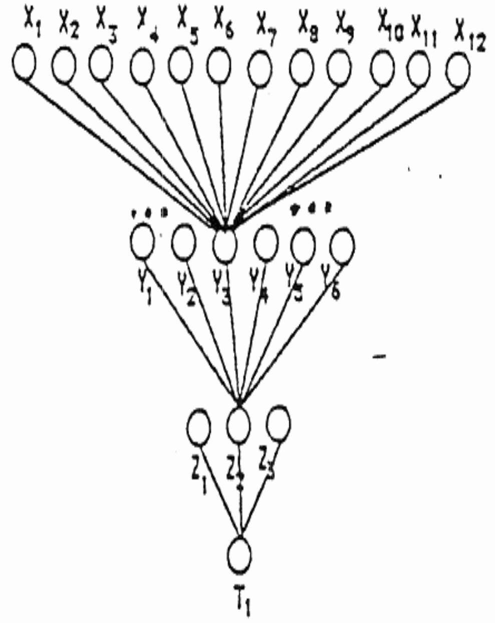


Figure 6: A layered decomposition of U^*_{12}

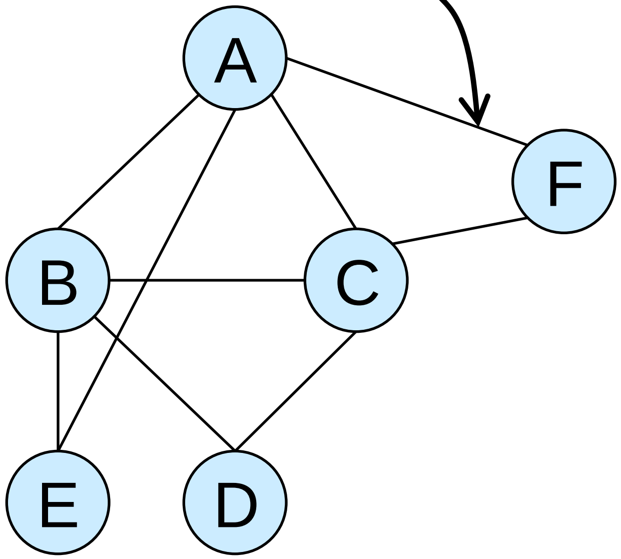
From Then On… Personal Perspective

Backjumping and no-good learning (1987-88)
(Wanted to understand TMS and Logic programming)
Sat-based Nonmon-reasoning (with **Ben-Eliyahu, 1990)**
(answer-set programming)
(Wanted to understand default logic, logic programming)
Temporal constraint networks (with **Meiri** and Pearl**, 1988-90)**
(Understanding what Dean and Macdermoth and James Allen were doing)
Distributed constraints (with **Collin and Katz, 1990)**
Neural networks hyped up again.
On the expressiveness of networks with hidden variables(Dechter 1990),
from local to global consistency (Dechter 1992)
Neural networks (will explain)
Identifiability of structures (trees) from relations (with **Meiri and Pearl, 1990)**
Learnability / PAC learning.
Bucket-elimination (Dechter, 96) (bringing treewidth/induced-width
to Bayesian networks)
Understanding probabilistic reasoning through VE
Mini-buckets, (with **Rish 1997)** finally Generating heuristics for real
(with **Kask, Marinescu, 2001, 2004)**
AND/OR search (with **Mateescu, 2004)**

Graphical models

- A graphical model (X,D,C):
  - $X = \{X_1,\ldots X_n\}$ variables
  - $D = \{D_1, \ldots D_n\}$ domains
  - $C = \{F_1,\ldots,F_t\}$ functions (constraints, CPTS, cnfs)

$$F_i := P(F \mid A, C)$$
$$F_i := F = A + C$$



Primary tasks

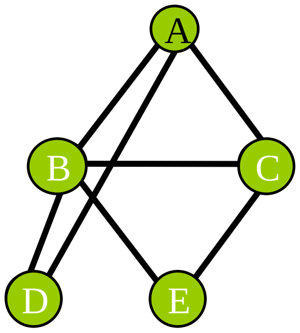
•Constraint satsifaction
•Constraint optimization
•Counting, belief updating
•Max expected utility

- All these tasks are NP-hard
- → identify special cases
- → approximate

Semiring-based constraint satisfaction and optimization (Bistareli, Montanari, Rossi JACM, 1997)

Finding

$$OPT = \min_{X_1,\ldots,X_n} \sum_{j=1}^{r} f_j(X)$$



Algorithm **elim-opt** (Dechter, 1996)
Non-serial Dynamic Programming (Bertele & Briochi, 1973)

$$OPT = \min_{a,e,d,c,b} F(a,b)+F(a,c)+F(a,d)+F(b,c)+F(b,d)+F(b,e)+F(c,e)$$

$$\underbrace{\min_b \sum}$$ ⟵ Elimination operator

bucket B: F(a,b) F(b,c) F(b,d) F(b,e)

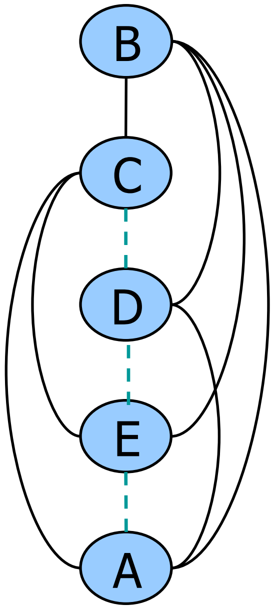
bucket C: F(c,a) F(c,e) ***$h^B(a,d,c,e)$***

bucket D: F(a,d) ***$h^C(a,d,e)$***

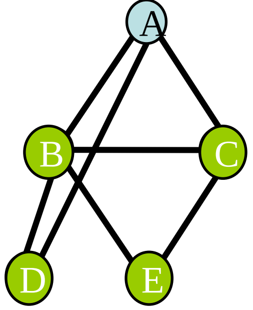
bucket E: ***$h^D(a,e)$***

bucket A: ***$h^E(a)$***

OPT



Generating the Optimal Assignment



5. $b' = \arg\max_{b} F(b,a') + F(d',b,a') + F(e',b,c')$

4. $c' = \arg\max_{c} F(c,a') + F(c,e') + h^{B}(a',d',c,e')$

3. $d' = \arg\max_{d} F(a',d) + h^{C}(a',d,e')$

2. $e' = \arg\max_{E} h^{D}(a,e)$

1. $a' = \arg\max_{a} h^{E}(a)$

B: F(a,b) F(b,c) F(b,d) F(b,e)

C: F(c,a) F(c,e) $h^{B}(a,d,c,e)$

D: F(a,d) $h^{C}(a,d,e)$

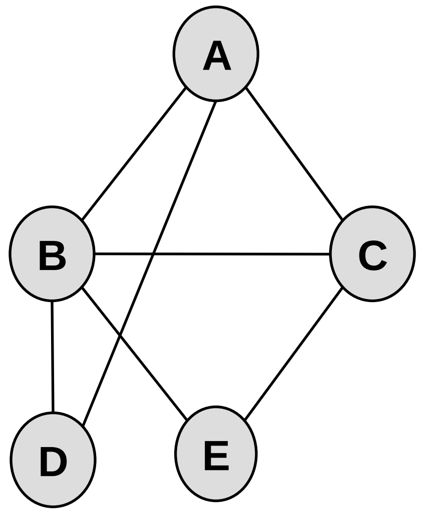
E: $h^{D}(a,e)$

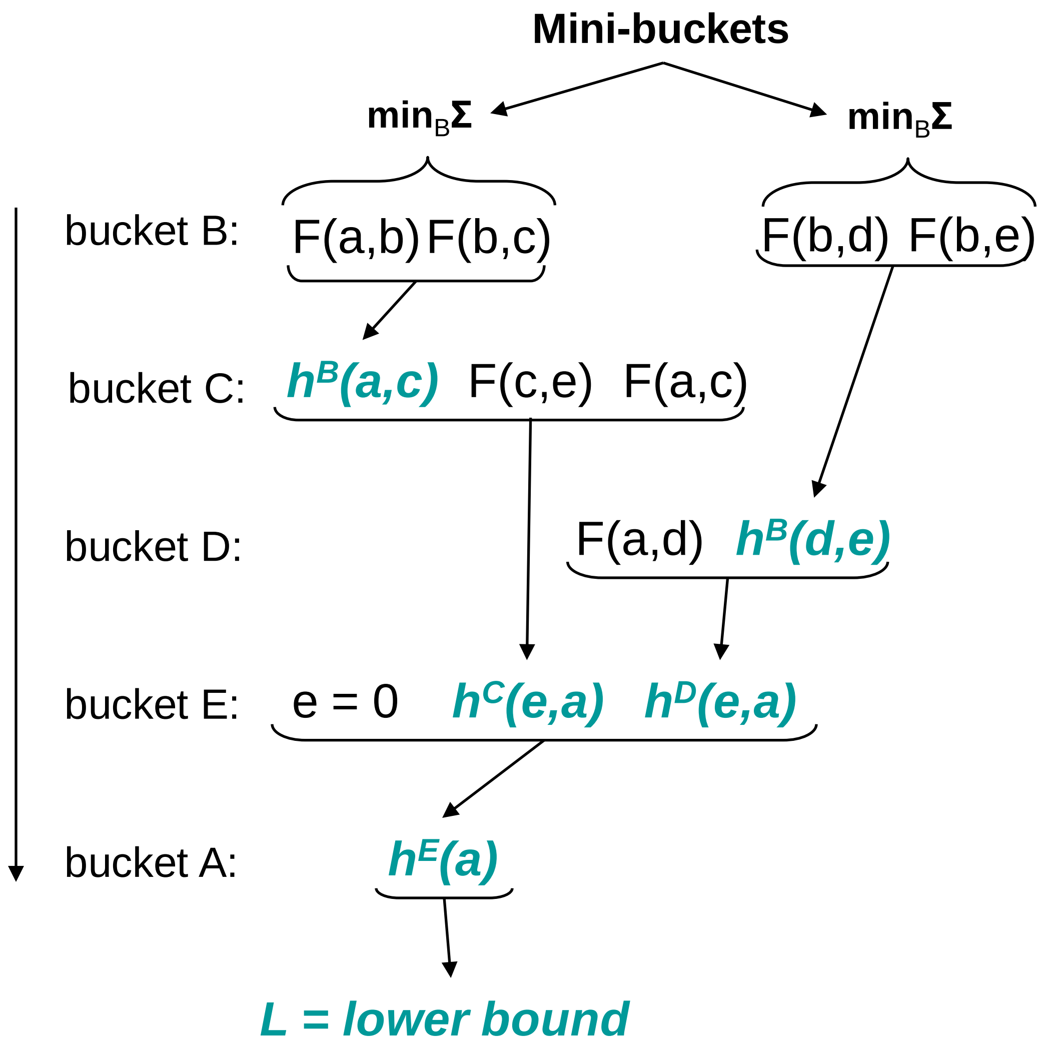
A: $h^{E}(a)$

Return (a',b',c',d',e')

Mini-Bucket Elimination

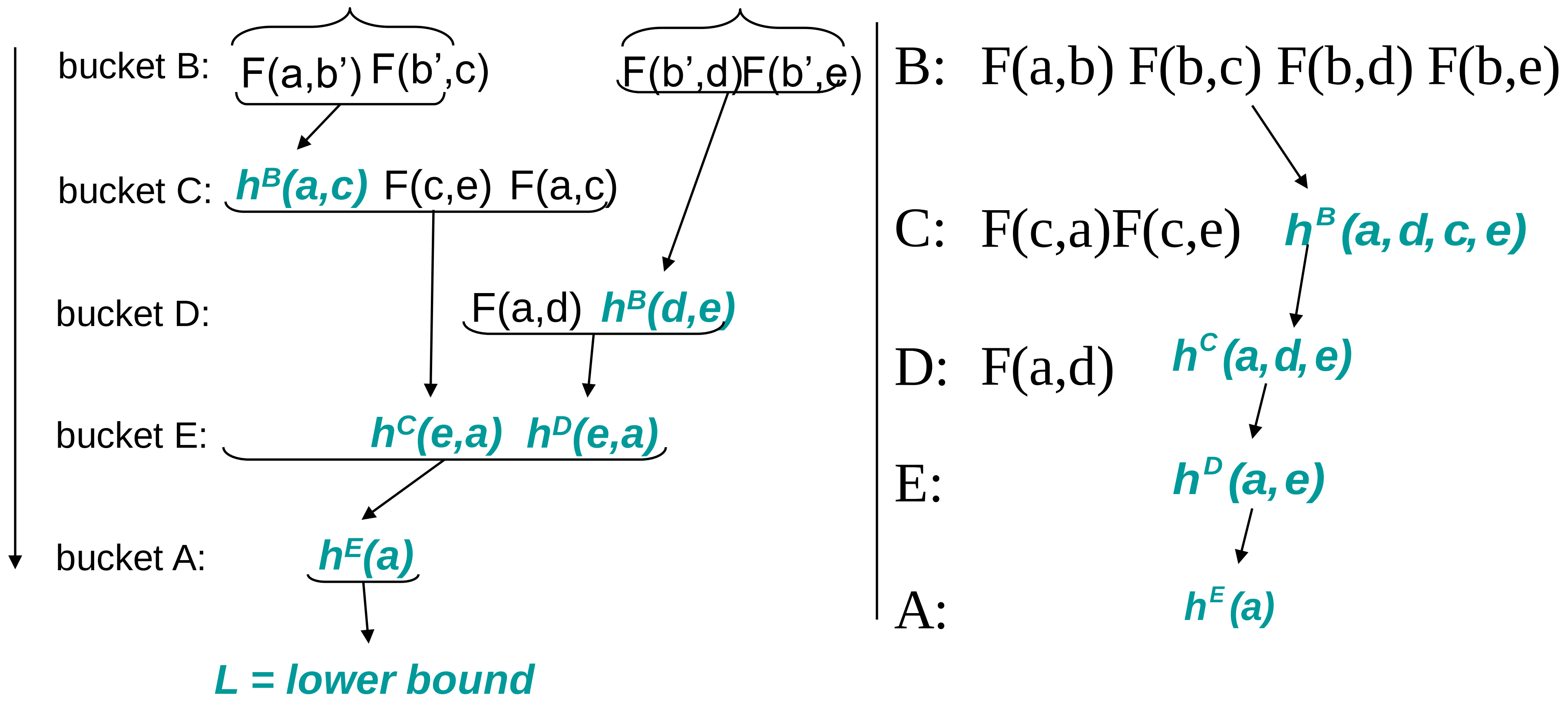
(Dechter and Rish 2003)




Mini-buckets
min_BΣ
min_BΣ
bucket B: F(a,b) F(b,c) F(b,d) F(b,e)
bucket C: h^B(a,c) F(c,e) F(a,c)
bucket D: F(a,d) h^B(d,e)
bucket E: e = 0 h^C(e,a) h^D(e,a)
bucket A: h^E(a)
L = lower bound

We can now generate a solution going up in the same manner
Yielding an upper bound

Bucket vs Mini-Bucket Elimination (Dynamic porgamming)


bucket B: F(a,b') F(b',c) F(b',d)F(b',e)
bucket C: hB(a,c) F(c,e) F(a,c)
bucket D: F(a,d) hB(d,e)
bucket E: hC(e,a) hD(e,a)
bucket A: hE(a)
L = lower bound
B: F(a,b) F(b,c) F(b,d) F(b,e)
C: F(c,a)F(c,e) hB(a,d,c,e)
D: F(a,d) hC(a,d,e)
E: hD(a,e)
A: hE(a)

$$\boldsymbol{h^{B}(a,d,c,e)} \leq \boldsymbol{h^{B}}(a,c) + h^{B}(d,e)$$

TO: Irina Rish

INFORMATION PROCESSING 71 – NORTH-HOLLAND PUBLISHING COMPANY (1972)

(1971)

ON THE OPTIMAL APPROXIMATION OF DISCRETE FUNCTIONS WITH LOW-DIMENSIONAL TABLES

Ugo MONTANARI
Istituto di Elaborazione dell'Informazione del C.N.R., Pisa, Italy

A discrete function, defined by a high-dimensional array is here approximated with a sum of functions, each dependent of a smaller number of variables. The tables defining these functions are computed by minimizing the mean square error. Three different types of constraints for this problem are considered. In the first type, the functional dependence of each term is given, while in the second type the "interaction graph" of the approximating sum is known. In the third type of problem, only the amount of the total available memory is given. In all these cases the solution of the problem can be obtained with simple algorithms. An important application of this approximation technique can be the approximate implementation of a dynamic programming procedure, where the intermediate stages need the memorization of tables of rapidly growing dimension. In fact, the objective function can be broken in the sum of lower-dimensional pieces after each stage.

1. INTRODUCTION

The practical problem of storing large high-dimensional arrays is often critical in numerical methods. For instance, the main limitation of dynamic programming optimization techniques [1] is the dimension of intermediate tables. If an approximate representation can be tolerated, many methods can be devised. For instance truncated expansions in terms of orthogonal functions can be a solution [2].

The method suggested by this paper consists in the optimal approximation (in the least square sense) of the given function with a sum of lower-dimensional functions. The advantages of this method are:

(1) The decoding process is very simple (a fixed number of summations).

(2) The compression ratio which we can obtain in this way is high, while the mean error can be small if the interaction among "separated" variables is limited.

(3) The approximating function has the form of a sum of terms and is therefore suitable for dynamic

assume as fixed the functional dependence of each term of the optimal approximating sum. An algorithm is then given for computing the actual value of the terms. Those values are essentially obtained using simple averaging techniques*. In section 3 the "interaction graph" of the approximating sum is given instead. This problem can be simply reduced to the precedent one. Finally, in section 4 we assume only the total available storage as given. The form of the approximating sum is here obtained by solving a linear programming problem (0,1) restricted.

2. APPROXIMATION WITH A SUM OF LOW-DIMENSIONAL FUNCTIONS

Let F be a function of n discrete variables (for simplicity all with the same definition domain):

$$F(x_1,\dots,x_n)\ ; \quad x_i = 1,\dots,N \quad (i=1,\dots,n)\,. \qquad (2.1)$$

Problem A

Given a function F and a characteristic function B for its lattice L, find a function $F_B \in S_B$ such that the error is minimal.

$$\epsilon = |F - F_B| = \sqrt{\sum_X (F - F_B)^2}$$

Example:

$$\overline{F}(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2, x_3) +$$

$$+ f_2(x_1, x_3, x_5) + f_3(x_3, x_4) + f_4(x_4, x_5)$$

Best approximation on a given set of scopes

Problem A: Given a function F and a characteristic function B for its lattice L, find a function $F_B \in S_B$ such that the error

$$\epsilon = |F - F_B| = \sqrt{\sum_X (F - F_B)^2}$$

is minimal.

THEOREM 2.2 (a) The average projections $h_i(X_i)$ computed by Step 1 of Algorithm A are the orthogonal projections of F on S_i *. (b) The union of proper spaces S_j, $X_j \subseteq X_i$, spans the entire space S_i. (c) Functions $k_i(X_j)$ computed by Step 4 are the orthogonal projections of F on S_i. (d) Function F_B computed by Step 5 is the solution of Problem A.

Algorithm A

Step 1. Compute the average projections

$$h_i(X_i) = \sum_{X - X_i} F(X)$$

of F on all the elements X_i of lattice L.

Step 2. Let $k_0(\phi) = h_0(\phi)$.

Step 3. Execute next step for $r = 1, \ldots, n$.

Step 4. For all the elements X_i of L having cardinality r, let

$$k_i(X_i) = h_i(X_i) - \sum_{X_j \subset X_i} k_j(X_j) \qquad (2.4)$$

where the summation is extended to all the elements X_j of L smaller than X_i.

Step 5. Compute the function

$$F_B = \sum_{B(X_i)=1} k_i(X_i) \qquad (2.5)$$

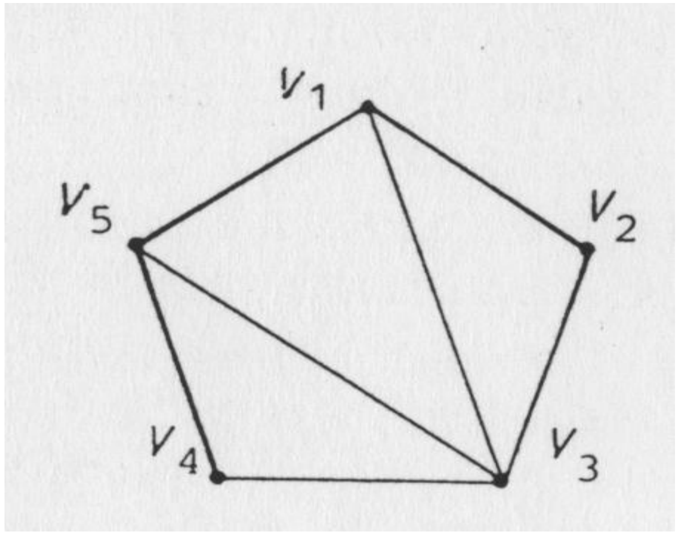
where the summation is extended to all the elements X_i of L such that $B(X_i) = 1$.

Optimal approximation with a given interaction graph

- Sum of terms:

$$\overline{F}(x_1,x_2,x_3,x_4,x_5)=f_1(x_1,x_2,x_3)+$$
$$+f_2(x_1,x_3,x_5)+f_3(x_3,x_4)+f_4(x_4,x_5)$$

Interaction graph:



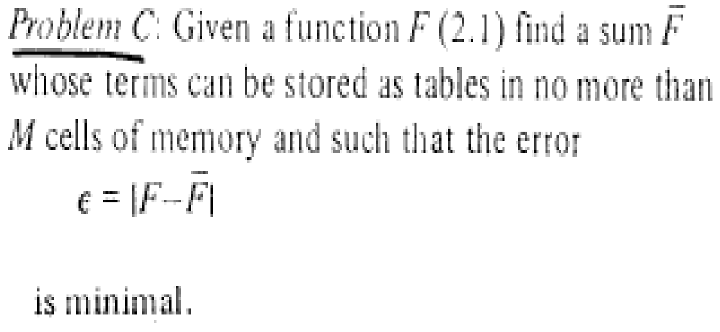
Alternative sum of

$$f_1(x_1,x_2,x_3)+f_3(x_1,x_5)+$$
$$+f_2(x_3,x_4,x_5)$$

Problem B reduces to:

1. Finding all complete subgraphs of graph G
2. Solving problem A

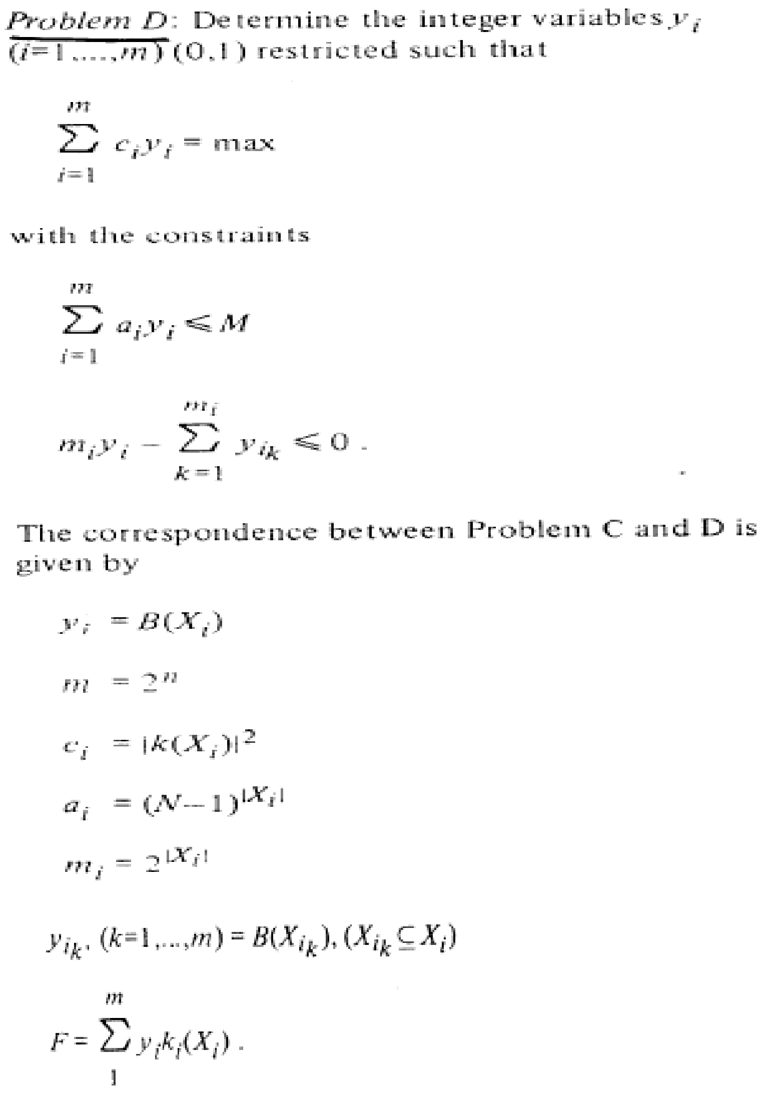
The Central Approximation Problem



Problem C: Given a function F (2.1) find a sum $\bar{F}$ whose terms can be stored as tables in no more than M cells of memory and such that the error

$$\epsilon = |F - \bar{F}|$$

is minimal.



Problem D: Determine the integer variables y_i $(i=1,\ldots,m)$ $(0,1)$ restricted such that

$$\sum_{i=1}^{m} c_i y_i = \max$$

with the constraints

$$\sum_{i=1}^{m} a_i y_i \leqslant M$$

$$m_i y_i - \sum_{k=1}^{m_i} y_{i_k} \leqslant 0 \, .$$

The correspondence between Problem C and D is given by

$$y_i = B(X_i)$$

$$m = 2^n$$

$$c_i = |k(X_i)|^2$$

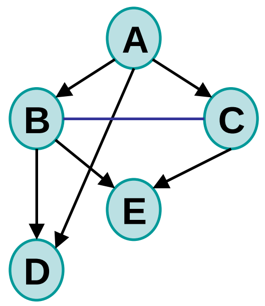
$$a_i = (N-1)^{|X_i|}$$

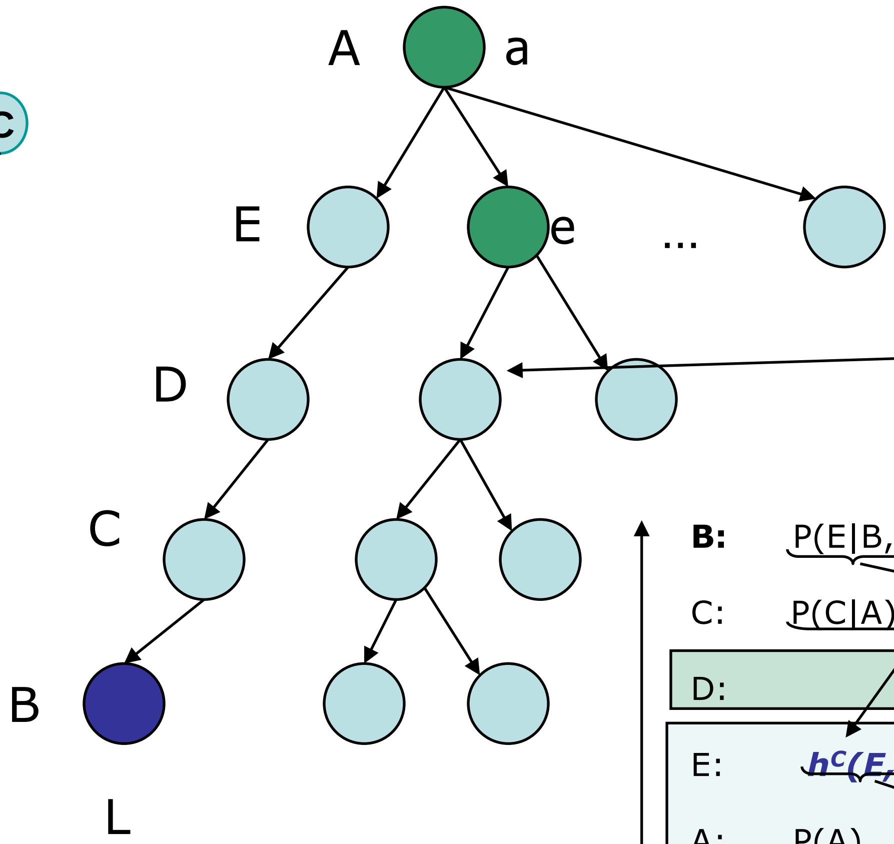
$$m_i = 2^{|X_i|}$$

$$y_{i_k}, (k=1,\ldots,m) = B(X_{i_k}), (X_{i_k} \subseteq X_i)$$

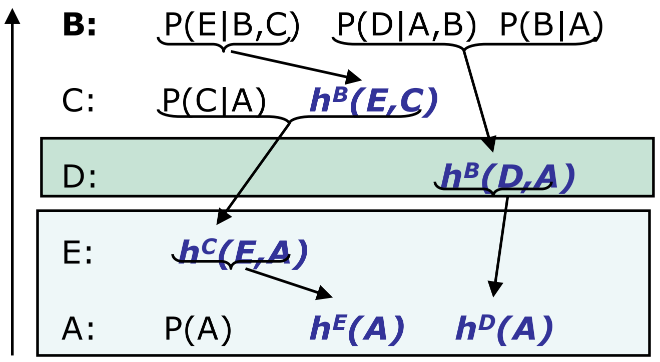
$$F = \sum_{1}^{m} y_i k_i(X_i) \, .$$

Mini-bucket Heuristics for BB search (Kask and dechterAIJ, 2001, Kask, Dechter and Marinsecue UAI 2003)





h(x) computed by MB(i) before or during search



$$f(a,e,D) = P(a) \cdot h^B(D,a) \cdot h^C(e,a)$$

So, where are we today?

- Lots of progress on tractability for constraints and for general graphical models (Graph-based and constraint-based).
- Local computation by mini-bucket for heuristic generation yield powerful search solvers for optimization
- Belief propagation algorithm are central for probabilistic reasoning
- Compilation scheme: the minimal network and multi-valued decision diagrams
- Powerful solvers for SAT/CSP for combinatorial optimization and counting.

In Summary

Thanks UGO!!!