Outline

- Introduction
- Exact algorithms
  - Search
  - Dynamic Programming
- Approximate Algorithms
  - Upper bounds: Incomplete Search (greedy, local)
  - Lower bounds: EPT, Relaxation
  - Unbounded approx: Iterative Message Passing

Outline

- Introduction
- Exact algorithms
  - Search
  - Dynamic Programming
- Approximate Algorithms
  - Upper bounds: Incomplete Search (greedy, local)
  - Lower bounds: EPT, Relaxation
  - Unbounded approx: Iterative Message Passing

Lower Bounds

- Equivalence Preserving Transformations (EPT):
  - Modifications of the problem while preserving the query.
  - We only consider moving costs among factors (the constraint graph is not changed)
- Relaxations:
  - Modifications of the problem w/o preserving the query, which makes the problem easier to solve.
  - We only consider removing factors (the constraint graph is changed reducing cyclicity)

Lower Bounds

- Equivalence Preserving Transformations (EPT):
  - Modifications of the problem while preserving the query.
  - We only consider moving costs among factors (the constraint graph is not changed)
- Relaxations:
  - Modifications of the problem w/o preserving the query, which makes the problem easier to solve.
  - We only consider removing factors (the constraint graph is changed reducing cyclicity)

Operations on factors: division ($\oslash$)

- Inverse of combination: $(a \oslash b) \otimes b = a$
- In our running example: $(5 - 2) + 5 = 5$
- Over factors:

It must remain in the valuation structure

x_1	x_2	$f(x_1, x_2)$
0	0	4
0	1	3
1	0	5
1	1	10

-

x_1	$g(x_1)$
0	3
1	4

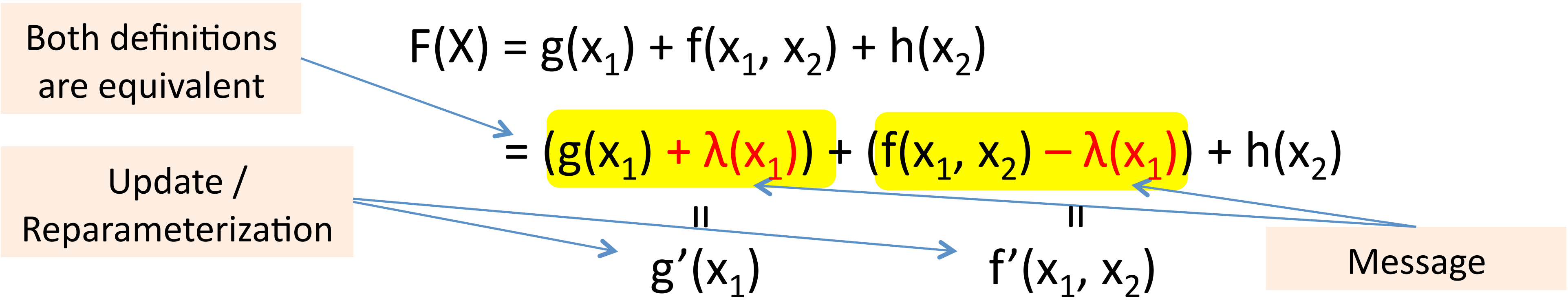
=

x_1	x_2	$h(x_1, x_2)$
0	0	4 - 3
0	1	3 - 3
1	0	5 - 4
1	1	10 - 4

$$F(X) = f(x_1, x_2) = (f(x_1, x_2) - g(x_1)) + g(x_1)$$

EPTs

- Project:
  - Moves costs from higher to smaller arity functions



$$F(X) = g(x_1) + f(x_1, x_2) + h(x_2)$$

$$= \underset{\overset{\shortparallel}{g'(x_1)}}{(g(x_1) + \lambda(x_1))} + \underset{\overset{\shortparallel}{f'(x_1, x_2)}}{(f(x_1, x_2) - \lambda(x_1))} + h(x_2)$$

Both definitions are equivalent

Update / Reparameterization

Message

- Extend:
  - Moves costs from smaller to higher arity functions

EPTs

- Project:
  - – Moves costs from higher to smaller arity functions

$$F(X) = g(x_1) + f(x_1, x_2) + h(x_2)$$

Lower bound

$$= (g(x_1) + \lambda(x_1)) + (f(x_1, x_2) - \lambda(x_1)) + h(x_2) \qquad f_{\varnothing}$$

$$\shortparallel$$

$$= (g(x_1) + \lambda(x_1)) + (f(x_1, x_2) - \lambda(x_1)) + (h(x_2) - A) + A$$

- Extend:
  - – Moves costs from smaller to higher arity functions

EPTs

- Project:
  - Moves costs from higher to smaller arity functions

$$F(X) = g(x_1) + f(x_1, x_2) + h(x_2)$$

$$= (g(x_1) + \lambda(x_1)) + (f(x_1, x_2) - \lambda(x_1)) + h(x_2)$$

$$= (g(x_1) + \lambda(x_1)) + (f(x_1, x_2) - \lambda(x_1)) + (h(x_2) - A) + A$$

- Extend:
  - Moves costs from smaller to higher arity functions

$$F(X) = g(x_1) + f(x_1, x_2) + h(x_2)$$

$$= (g(x_1) - \delta(x_1)) + (f(x_1, x_2) + \delta\ (x_1)) + h(x_2)$$

EPTs

- Project:
  - Moves costs from higher to smaller arity functions

$$F(X) = g(x_1) + f(x_1, x_2) + h(x_2)$$

$$= (g(x_1) + \lambda(x_1)) + (f(x_1, x_2) - \lambda(x_1)) + h(x_2)$$

$$= (g(x_1) + \lambda(x_1)) + (f(x_1, x_2) - \lambda(x_1)) + (h(x_2) - A) + A$$

- Extend: = Projecting the inverse
  - Moves costs from smaller to higher arity functions

$$F(X) = g(x_1) + f(x_1, x_2) + h(x_2)$$

$$= (g(x_1) + (-\delta(x_1))) + (f(x_1, x_2) - (-\delta(x_1))) + h(x_2)$$

“Generalized” EPT

- Given a function f:
  - Compute a function λ s.t. :
    - $\mathrm{var}(\lambda) \subseteq \mathrm{var}(f)$
    - $f \obslash \lambda$ remains in the valuation structure
  - Update the network by:
    - $(f \obslash \lambda) \otimes \lambda$

- The resulting network is equivalent.

EPT’s Algorithms

- Goal: maximize $f_{\emptyset}$
- Chaotic application of EPTs:
  1. It may not end.
  2. The ending point may be different.
- Planned application of EPTs:
  - Over the naturals: finding a sequence of EPTs that maximizes $f_{\emptyset}$ is NP-complete.
  - Over the rationals: finding a set of EPTs that maximize $f_{\emptyset}$ is polinomial time.

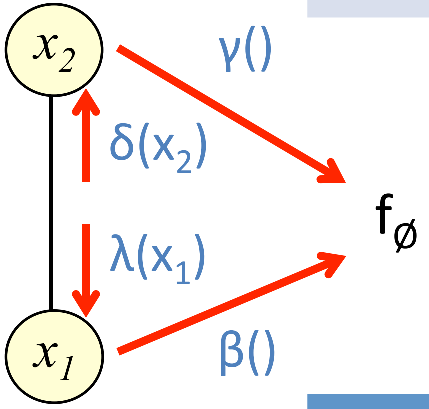
EPT: Optimal Soft Arc Consistency

x_2	$\delta(x_2)$
0	c
1	d

x_1	$\lambda(x_1)$
0	a
1	b

$\gamma()$
i

$\beta()$
e



$$F(X) = \left\{ \begin{array}{ll} g(x_1) + \lambda(x_1) - \beta() & g'(x_1) \\ + & \\ f(x_1, x_2) - \lambda(x_1) - \delta(x_2) & f'(x_1, x_2) \\ + & \\ h(x_2) + \delta(x_2) - \gamma() & h'(x_2) \\ + & \\ f_\emptyset + \beta() + \gamma() & f_\emptyset' \end{array} \right.$$

max $f_\emptyset + \beta() + \gamma()$

subject to:

$$\left\{ \begin{array}{l} g(x_1) + \lambda(x_1) - \beta() \geq 0 \\ h(x_2) + \delta(x_2) - \gamma() \geq 0 \\ f(x_1, x_2) - \lambda(x_1) - \delta(x_2) \geq 0 \end{array} \right.$$

$g(0) + a - e \geq 0$

$g(1) + b - e \geq 0$

…

EPT: Other Algorithms

- Idea:
  - To restrict the application of EPTs s.t. its termination is guaranteed.

- Over the naturals:
  - Local Consistencies: NC*, AC*, DAC*, FDAC*, EDAC*.

- Over the rationals:
  - Virtual Arc Consistency / Augmenting DAG.
  - Min-Sum Diffusion.

Branch and Bound + EPT Algorithms

```
function Solve($F$, $ub$)
    if Constant(F) then return min{$F$,$ub$};
    if (LB($F$)$\geq$ $ub$) return $ub$;
    $x$:= SelectVar($F$);
    ub:=Solve(EPT_Algorithm($F$($\mathbf{x}'$)),$ub$);
    return Solve(EPT_Algorithm($F$($\mathbf{x}$)),$ub$);
endfunction
```

$f_{\varnothing}$ is the LB

Trade-off between quality of $f_{\varnothing}$ and time complexity

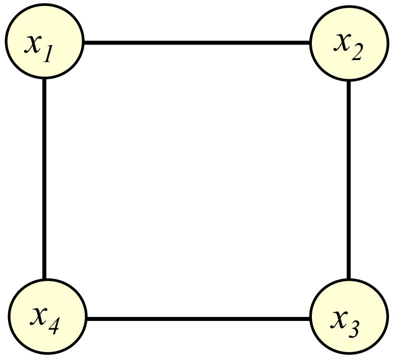
Initial call: Solve(F, $UB(F)$)

Cost $O(2^n)$

Lower Bounds

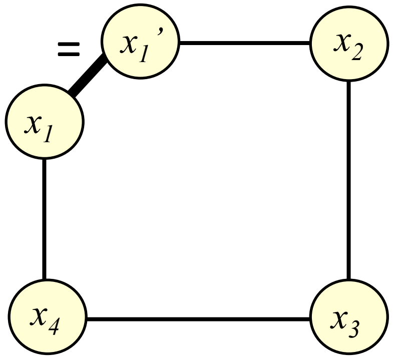
- Equivalence Preserving Transformations (EPT):
  - Modifications of the problem while preserving the query.
  - We only consider moving costs among factors (the constraint graph is not changed)
- Relaxations:
  - Modifications of the problem w/o preserving the query, which makes the problem easier to solve.
  - We only consider removing factors (the constraint graph is changed reducing cyclicity)

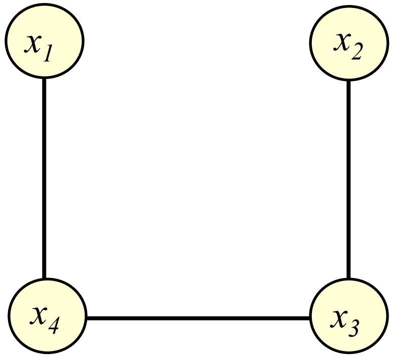
Relaxations



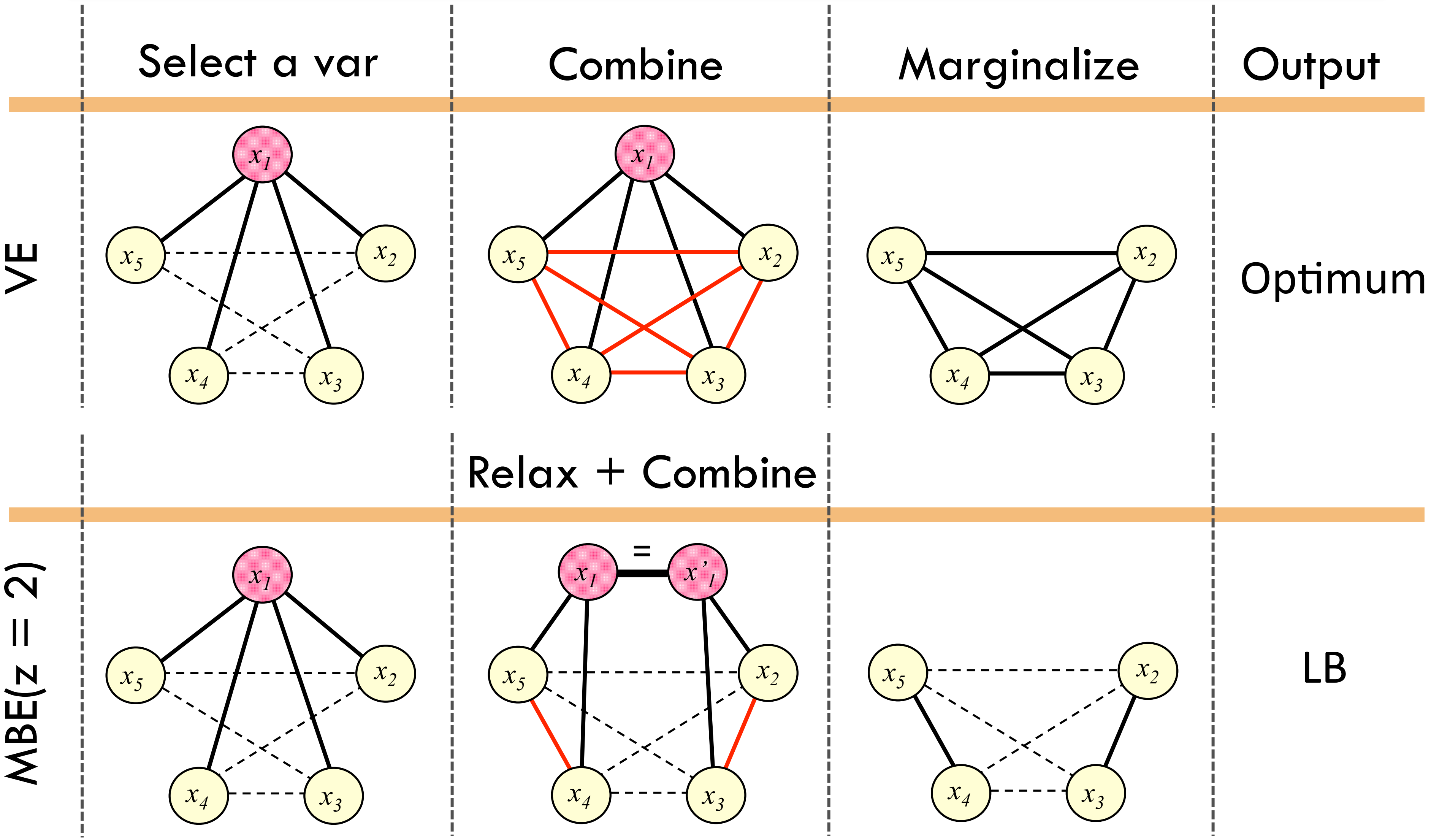
1. Duplicating a variable + removing equality constraint

2. Removing a function





Mini-Bucket Elimination



Select a var

Combine

Marginalize

Output

VE

Optimum

Relax + Combine

MBE(z = 2)

LB

Mini-Bucket Elimination

```
function LB(F, z)
    if Constant(F) then return F
    x := SelectVar(F);
    return Solve(MiniElim(F, x, z));
endfunction

function MiniElim(F, x, z)
    B := factors with x in their scope;
    {$Q_1$,…,$Q_p$} := Partition(B, z);
    F' := replace B by $\{\min_x \sum_{f \in Q_j} f\}_{j=1}^{p}$ in F;
    return F';
endfunction
```

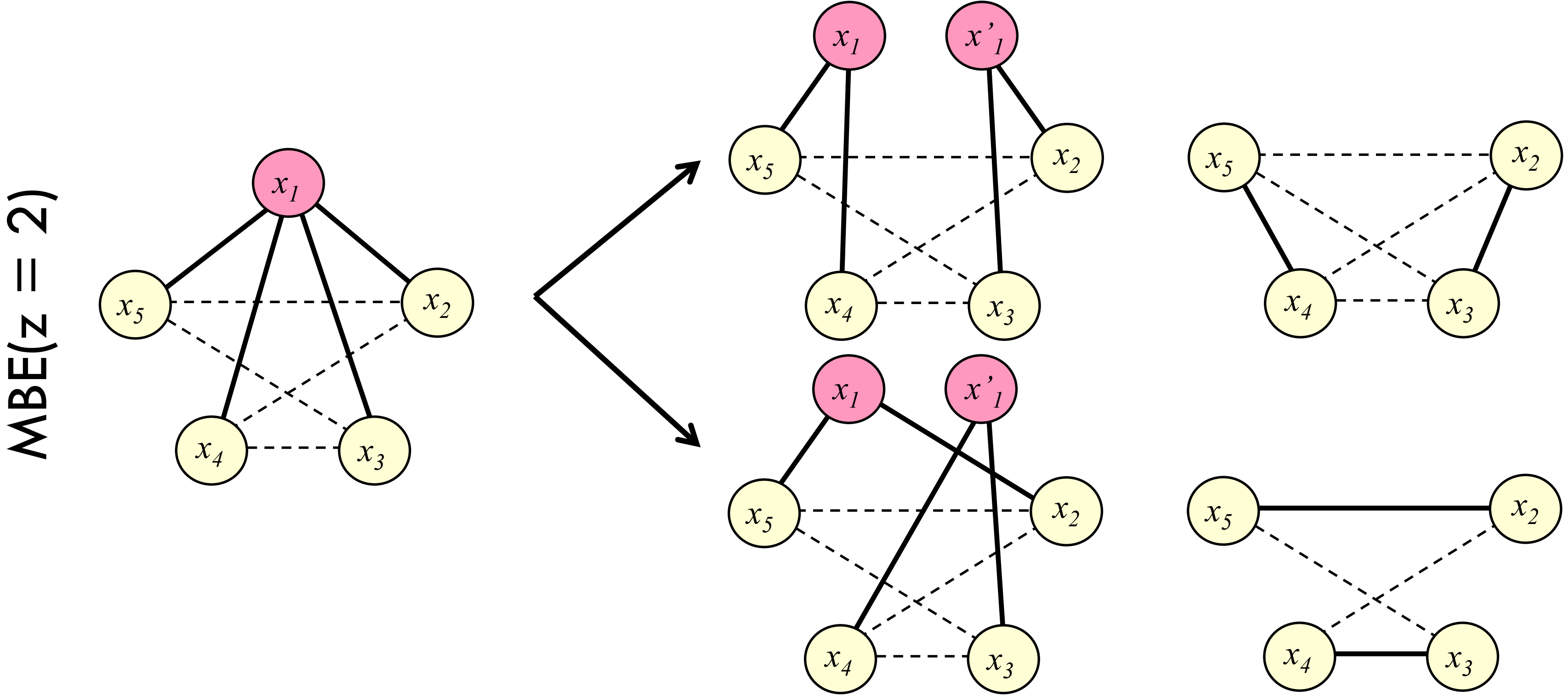
- Is that partition unique?
- How to partition a bucket?

Cost MiniElim *x*: O(2^{z+1})

Cost: O(2^{z+1})

Mini-Bucket Elimination

- The partition is not unique:



- The choice is made heuristically:
  - scope-based or content-based heuristics.

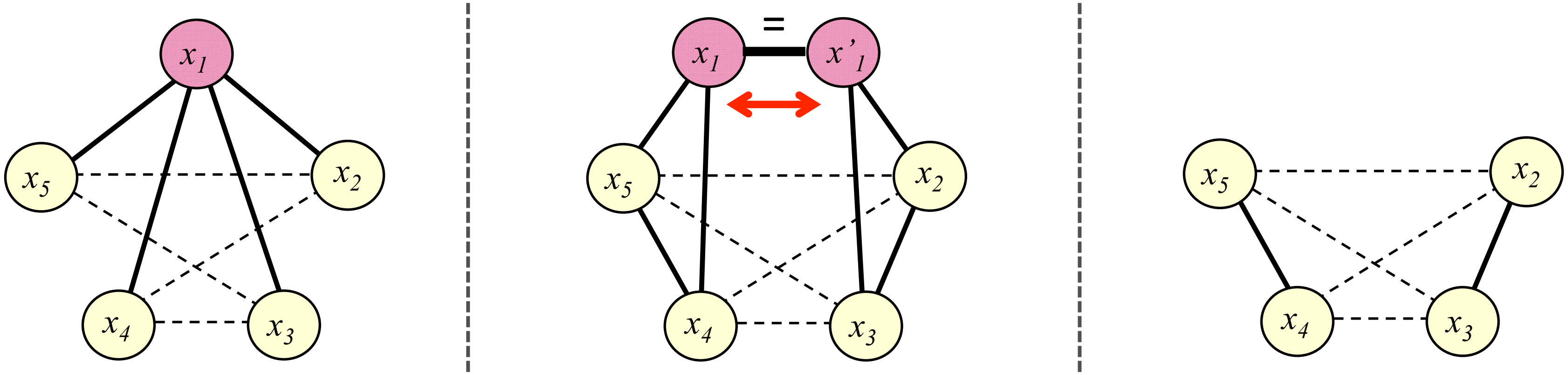
Mini-Bucket Elimination + EPT

Select a var

Relax + Combine

Marginalize

MBE(z = 2)



Move costs

(generalized EPT)

There are different strategies:

- based on a preestablished order
- based on min-sum diffusion

Branch and Bound + MBE

```
function Solve(F, ub, z)
    if Constant(F) then return min{F,ub};
    if (MBE(F, z) ≥ ub) return ub;
    x := SelectVar(F);
    ub := Solve(F(x’), ub);
    return Solve(F(x), ub);
endfunction
```

z controls the strenght of LB

MBE does not maintain equivalence of the original problem

Initial call: Solve(*F*, *UB*(*F*), *z*)

Cost $O(2^n)$

Branch and Bound + MBE

```
function Solve(F, ub, z)
    if Constant(F) then return min{F,ub};
    if (MBE(F, z) $\geq$ ub) return ub;
    x := SelectVar(F);
    ub := Solve(EPT_Algorithm(F(x')), ub);
    return Solve(EPT_Algorithm(F(x)), ub);
endfunction
```

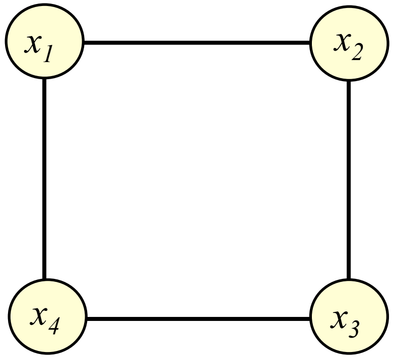
z controls the strenght of LB

MBE does not maintain equivalence of the original problem

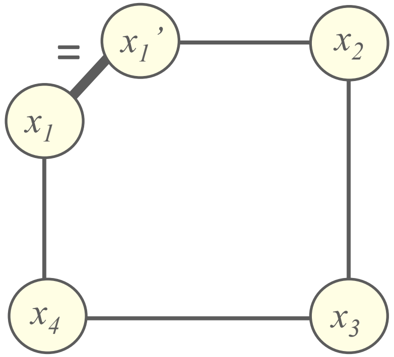
Initial call: Solve(*F*, *UB*(*F*), *z*)

Cost $O(2^n)$

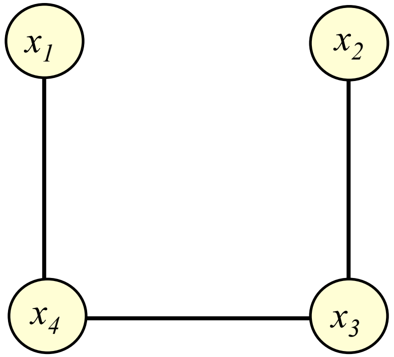
Relaxations



1. Duplicating a variable + removing equality constraint



2. Removing a function



EPT + removing a function

x_1	x_2	$f(x_1, x_2)$
0	0	4
0	1	3
1	0	5
1	1	10

-

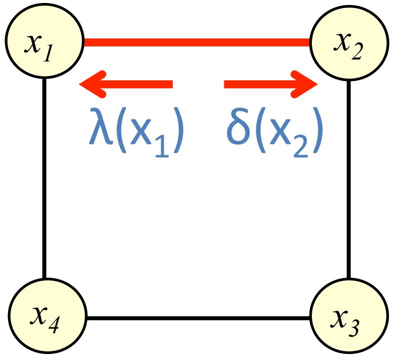
x_1	$\lambda(x_1)$
0	a
1	b

-

x_2	$\delta(x_2)$
0	c
1	d

=

x_1	x_2	$h(x_1, x_2)$
0	0	e
0	1	j
1	0	k
1	1	m



Move costs

minimize e + j + k + m

subject to:

4 – a – c = e
3 – a – d = j
5 – b – c = k
10 – b – d = m

EPT + removing a function

x_1	x_2	$f(x_1, x_2)$
0	0	4
0	1	3
1	0	5
1	1	10

-

x_1	$\lambda(x_1)$
0	a
1	b

-

x_2	$\delta(x_2)$
0	c
1	d

=

x_1	x_2	$h(x_1, x_2)$
0	0	e
0	1	j
1	0	k
1	1	m

x_1	x_2	$f(x_1, x_2)$
0	0	4
0	1	3
1	0	5
1	1	10

=

x_1	$\lambda(x_1)$
0	a
1	b

+

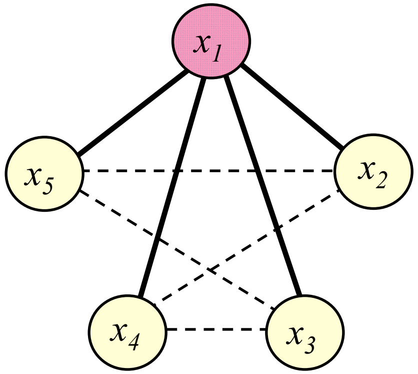
x_2	$\delta(x_2)$
0	c
1	d

+

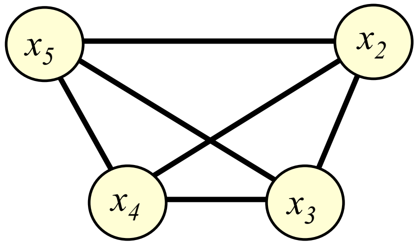
Error

x_1	x_2	$h(x_1, x_2)$
0	0	e **= 0**
0	1	j **= 0**
1	0	k **= 0**
1	1	m **= 0**

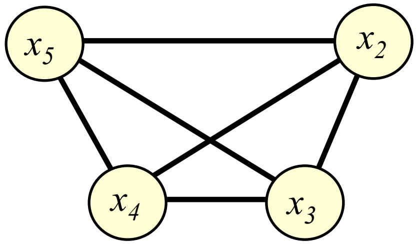
VE + (EPT + removing a function)



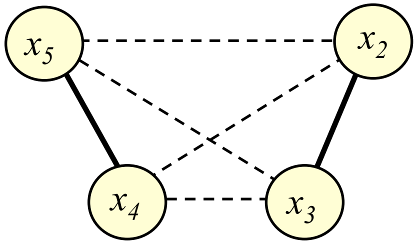
Fully eliminate x1 …



… but if the value of the control parameter is exceeded then:



EPT + remove a function

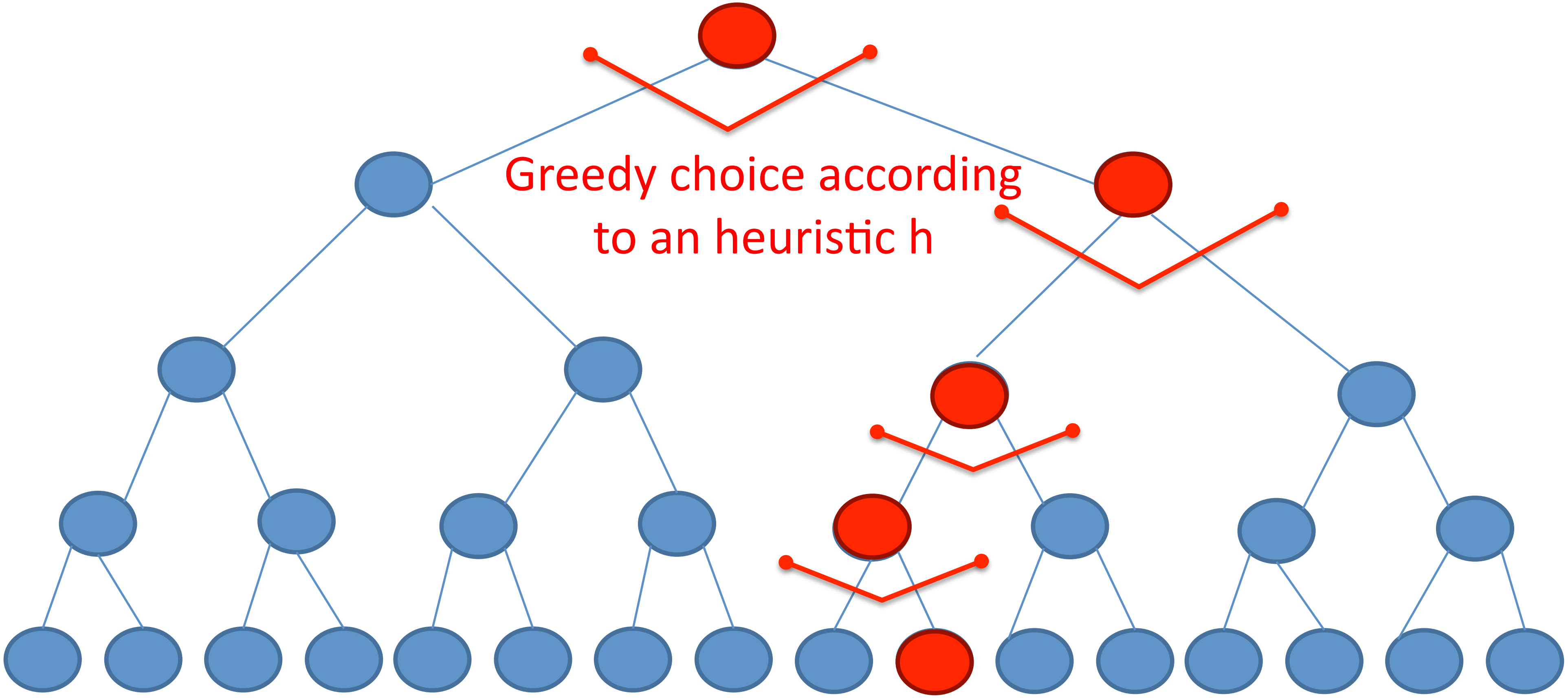


Outline

- Introduction
- Exact algorithms
  - Search
  - Dynamic Programming
- Approximate Algorithms
  - Upper bounds: Incomplete Search (greedy, local)
  - Lower bounds: EPT, Relaxation
  - Unbounded approx: Iterative Message Passing

Incomplete Search: Greedy

If we consider the whole search space:



Greedy choice according to an heuristic h

Greedy Search only traverses ONE path.

Incomplete Search: Greedy

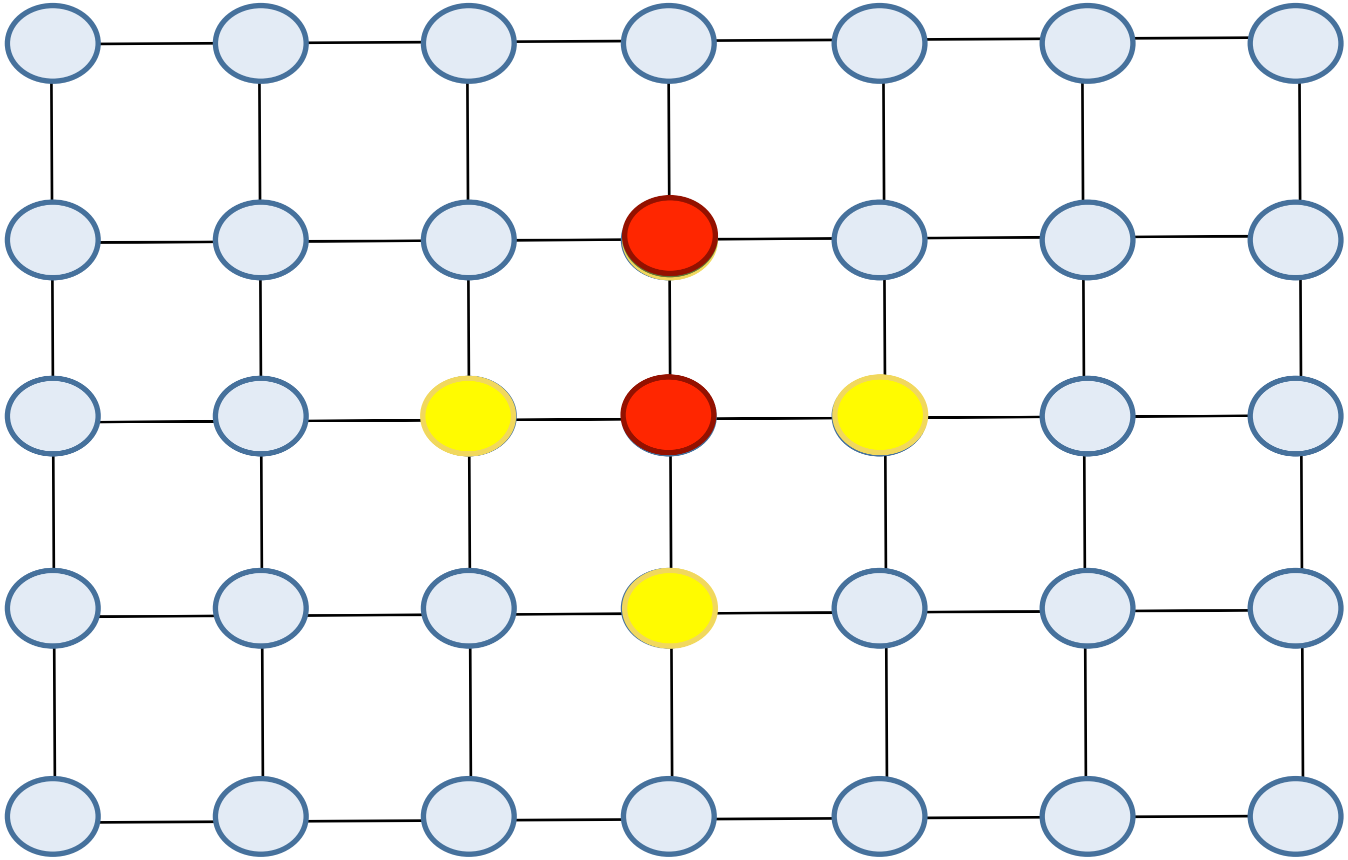
```
function UB($F$, $h$)
    if Constant(F) then return $F$;
    $x$:= SelectVar($F$);
    if $h(F(\mathbf{x'})) \geq h(F(\mathbf{x}))$ then return $UB(F(\mathbf{x'}),h)$;
    return $UB(F(\mathbf{x}),h)$;
endfunction
```

It’s the best local choice

There is no backtracking

Cost O(n)

Incomplete Search: Local



Choose one neighbour

Decide if it is replaced or not

Neighbourhood relation

Incomplete Search: Local

```
function UB(F)
    X := random assignment;
    while (Finalise?(X)) do
        L := Neighbour(X);
        X' := Choose(L);
        if Move?(X, X') then X := X';
    endwhile
endfunction
```

- or greedy local search

- X is good enough
- A computation budget T is exhausted

- Mutate X in some way

- Best according to F
- At random

- Only if best according to F
- If best and if worse with some probability

Cost O(T)

Local Search + Branch and Bound

```
function Solve($F$, $ub$)
    if Constant(F) then return min{$F$,$ub$};
    if (LB($F$)$\geq$ $ub$) return $ub$;
    $x$:= SelectVar($F$);
    ub:=Solve($F(\mathbf{x}')$,$ub$);
    return Solve($F(\mathbf{x})$,$ub$);
endfunction
```

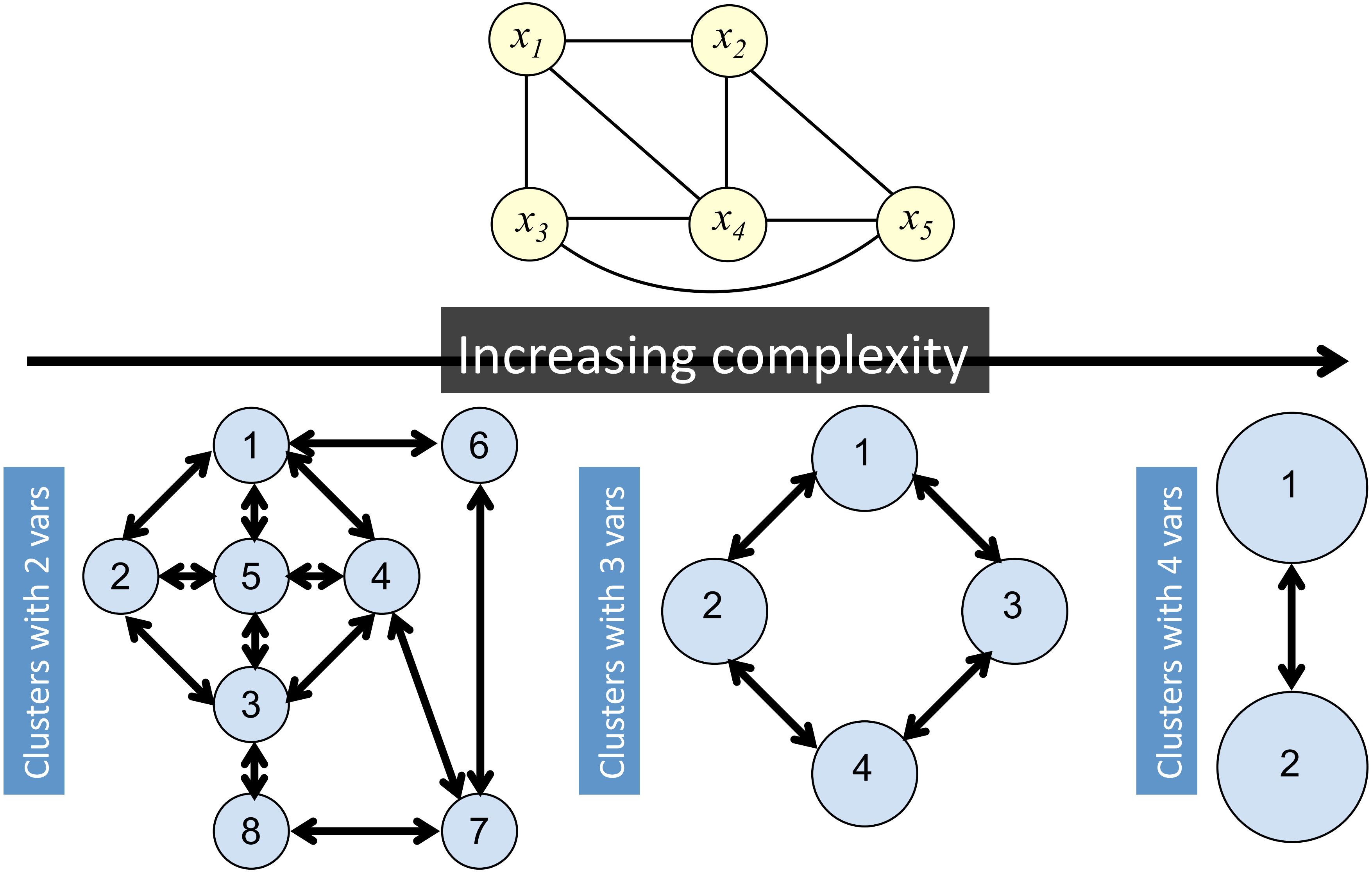
Initial call: Solve(*F*, *UB*(F))

Cost $O(2^n)$

Outline

- Introduction
- Exact algorithms
  - Search
  - Dynamic Programming
- Approximate Algorithms
  - Upper bounds: Incomplete Search (greedy, local)
  - Lower bounds: EPT, Relaxation
  - Unbounded approx: Iterative Message Passing

Iterative Cluster Message Passing



Increasing complexity

Clusters with 2 vars

Clusters with 3 vars

Clusters with 4 vars

Iterative Cluster Message Passing

- Termination condition:
  - msg remain unchanged from previous iterations (converge).
  - a maximum number of iterations has been reached.

- Convergence over arbitrarily graphs:
  - Not guaranteed on standard message-passing algorithms.

- Upon convergence, what is its accuracy?

- It’s an active line of research in the Machine Learning community.