Kernelization in constraint satisfaction and reasoning

Serge Gaspers

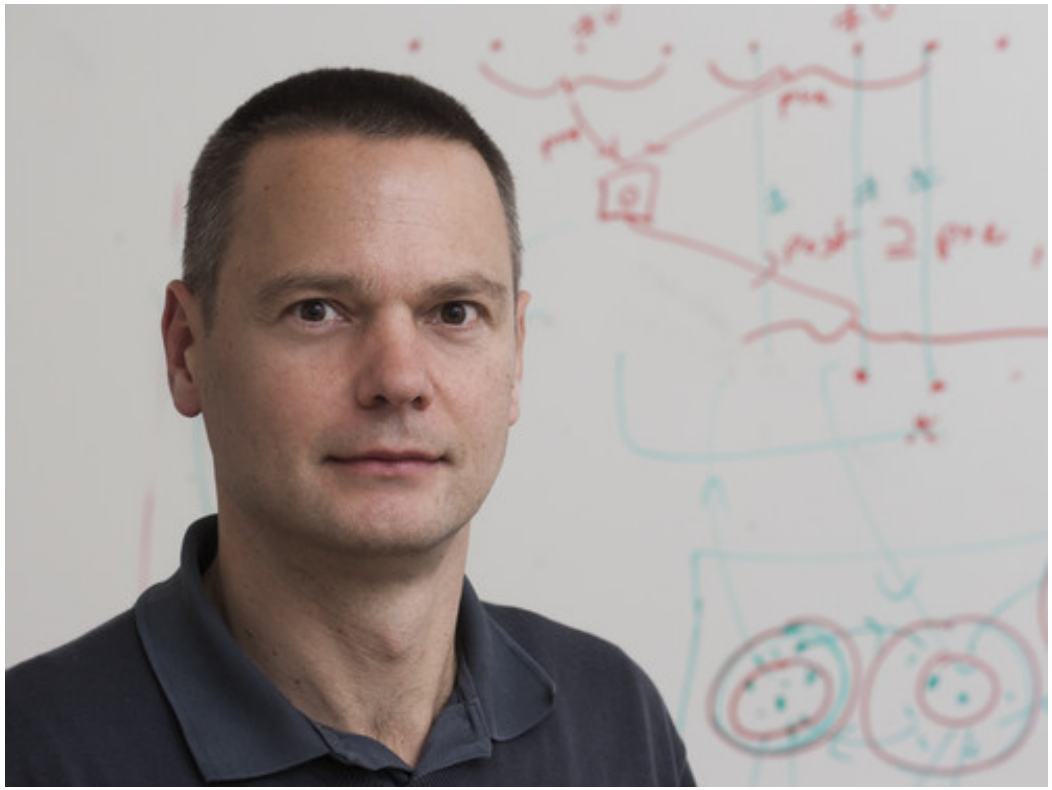
University of New South Wales, Sydney, Australia

NICTA, Sydney, Australia

WorKer 2015

Collaborator

Joint work with Stefan Szeider (Vienna University of Technology)



Serge Gaspers and Stefan Szeider. *Guarantees and Limits of Preprocessing in Constraint Satisfaction and Reasoning*. Artificial Intelligence 216: 1-19, 2014.

Outline

Outline

Preprocessing in AI and Reasoning

Simplifying the formula before giving it to the SAT solver is a crucial part of the solving chain. Several techniques, such as bounded variable elimination, blocked clause elimination, equivalent literal elimination, probing or automated re-encoding by using extended resolution, have been proposed as formula simplifications.

Kilian Gebhardt and Norbert Manthey. *Parallel Variable Elimination on CNF Formulas*. KI 2013.

Preprocessing in AI and Reasoning

The size of CNF formulas is often very large, particularly in the context of formal verification. In theory, a very large formula may be easy to solve and a small formula hard. However, in practice, it is often observed that the runtime of a SAT solver is very much related to the size of the input formula, at least when the formulas stem from the same set of problems.

Niklas Eén, Armin Biere. *Effective Preprocessing in SAT Through Variable and Clause Elimination*. SAT 2005.

Preprocessing in AI and Reasoning

Network preprocessing can also be quite effective in the presence of local structure, especially determinism, and is orthogonal to the algorithms used afterwards. For example, preprocessing has proven quite effective and critical for networks corresponding to genetic linkage analysis, allowing exact inference on networks with very high treewidth.

Adnan Darwiche. *Chapter 11: Bayesian Networks*. In Handbook of Knowledge Representation, Elsevier 2008.

Preprocessing in AI and Reasoning

The amount of preprocessing the AI does is insane. It was one of the main challenges to get the preprocessing even on a huge map done within a maximum time frame of 30 seconds. Most of this is done by highly optimized routines that fit all data into the processor cache to gain a factor of 5–10.

Quantomas. Heroes V AI, www.bonddisc.com/ref/h5/

Preprocessing in AI and Reasoning (2)

- Practical preprocessing thrives in SAT, CSP, Bayesian Reasoning, Nonmonotonic Reasoning
- Preprocessing is often crucial for implementations to handle reasonably-sized instances

Preprocessing in AI and Reasoning (2)

- Practical preprocessing thrives in SAT, CSP, Bayesian Reasoning, Nonmonotonic Reasoning
- Preprocessing is often crucial for implementations to handle reasonably-sized instances
- Let's look at the kernelization complexity of some of the most central parameterized problems in these domains

Outline

SAT

SAT

Input: CNF formula F

Question: Is F satisfiable?

Example:

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

Backdoors

Definition 1 (sub-solver [Williams, Gomez, Selman, '03])

A sub-solver is an algorithm A that takes as input a CNF formula and

- either rejects the input or determines $sat(F)$ correctly,
- runs in polynomial time,
- A determines $sat(F)$ if $F = \emptyset$ or $\emptyset \in cla(F)$,
- if A determines $sat(F)$, then A also determines $sat(F[x = 0])$ and $sat(F[x = 1])$.

Definition 2 (backdoor)

An A-backdoor of a CNF formula F is a set B of variables such that for each truth assignment τ to B, the satisfiability of $F[\tau]$ is determined by the sub-solver A.

Backdoor problems

Rec-Sat(A-backdoor)

Input:	CNF formula F, integer k
Parameter:	k
Question:	Does F have an A-backdoor of size at most k?

FPT when A solves HORN formulas (each clause has at most 1 positive literal) or 2CNF formulas (each clause has at most 2 literals)

Sat(A-backdoor)

Input:	CNF formula F, A-backdoor B
Parameter:	$k = \|B\|$
Question:	Is F satisfiable?

FPT

Recognize Backdoors

Proposition 1

Rec-Sat(HORN-backdoor) has a kernel with $O(k)$ variables.
Rec-Sat(2CNF-backdoor) has a kernel with $O(k^2)$ variables.

Recognize Backdoors

Proposition 1

Rec-Sat(HORN-backdoor) has a kernel with $O(k)$ variables.
Rec-Sat(2CNF-backdoor) has a kernel with $O(k^2)$ variables.

Proof.

Using obstructions

- HORN: 2 positive literals occurring in the same clause
- 2CNF: 3 literals occurring in the same clause

translate to Vertex Cover / 3-Hitting Set
use known kernels
translate back to formulas (each edge/hyperedge gives a clause with positive literals) □

Evaluate Backdoors

Theorem 3

SAT(A-backdoor) has no polynomial kernel for any sub-solver A unless NP $\subseteq$ coNP/poly.

Proof.

Polynomial parameter transformation from SAT(vars).

SAT(vars)

Input:	CNF formula F
Parameter:	n, the number of variables of F
Question:	Is F satisfiable?

(The set of all variables is an A-backdoor.) □

Evaluate Backdoors

Theorem 3

SAT(A-backdoor) has no polynomial kernel for any sub-solver A unless NP $\subseteq$ coNP/poly.

Proof.

Polynomial parameter transformation from SAT(vars).

SAT(vars)

Input:	CNF formula F
Parameter:	n, the number of variables of F
Question:	Is F satisfiable?

(The set of all variables is an A-backdoor.) □

What about 3SAT(HORN-backdoor) and 3SAT(2CNF-backdoor)?

Evaluate Backdoors for 3CNF

Theorem 4

3SAT(HORN-backdoor) and 3SAT(2CNF-backdoor) do not have polynomial kernels unless NP $\subseteq$ coNP/poly.

Proof.

OR-composition.

Let (F_i, B), $1 \leq i \leq t$ be t instances with $|B| = k$.

(1) $t > 2^k$.

Solve all instances in $O(t2^k n) = O(t^2 n)$ time, where n is the max formula size.

If some F_i is satisfiable, output (F_i, B_i); otherwise, output (F_1, B_1).

Evaluate Backdoors for 3CNF

Theorem 4

3SAT(HORN-backdoor) and 3SAT(2CNF-backdoor) do not have polynomial kernels unless NP $\subseteq$ coNP/poly.

Proof.

(2) $t \leq 2^k$.
Let $s = \lceil \log t \rceil$. Denote by V_i the variables in F_i and assume $V_i \cap V_j = B$ for $i \neq j$.
Assume $2^s = t$, otherwise add dummy unsatisfiable formulas.
For each variable $x \in V_i \setminus B$, take $s+1$ new variables $x_0, \ldots, x_s$.
Replace literal x by x_0 and literal $\neg x$ by $\neg x_s$.
Add implication clauses $(\neg x_{j-1} \vee x_j)$ for $1 \leq j \leq s$.

Evaluate Backdoors for 3CNF

Theorem 4

3SAT(HORN-backdoor) and 3SAT(2CNF-backdoor) do not have polynomial kernels unless NP $\subseteq$ coNP/poly.

Proof.

New variables $Y = \{y_1, \ldots, y_s\}$. Denote by $C_1, \ldots, C_{2^s}$ all possible clauses on Y, and write $C_i = (\ell_1^i \vee \cdots \vee \ell_s^i)$.
To each connection clause $(\neg x_{j-1} \vee x_j)$ of F_i, add the literal ℓ_j^i, giving the clause $(\neg x_{j-1} \vee x_j \vee \ell_j^i)$.
This allows to break the implication chains for all F_i, except one.
We obtain a formula that is equisatisfiable to the OR of all t original formulas, and $B \cup Y$ is a backdoor to 2CNF and HORN of size $2k$.

□

Outline

Constraint Network

Definition 5 (Constraint Network)

A constraint network is a triple $I = (X, D, C)$ where

- X is a set of variables,
- D is a set of values, and
- $C = \{C_1, \ldots, C_m\}$ is set of constraints.

Each constraint C_i is a pair (S_i, R_i) where

- S_i is a list of variables called the constraint scope, and
- R_i is an $|S_i|$-ary relation over D, called the constraint relation.

The tuples of R_i indicate the allowed combinations of simultaneous values for the variables S_i.
A solution is a mapping $\tau : X \to D$ such that for each $1 \leq i \leq m$ and $S_i = (x_1, \ldots, x_{r_i})$, we have $(\tau(x_1), \ldots, \tau(x_{r_i})) \in R_i$.
A constraint network is satisfiable if it has a solution.

D-CSP(tw)

Definition 6 (constraint graph)

The constraint graph of a constraint network $I = (X, D, C)$ is the graph $G = (X, E)$ containing an edge uv iff variables u and v appear together in a constraint scope.

D-CSP(tw)

Input:	Constraint network $I = (X, D, C)$, tree decomposition T of I's constraint graph
Parameter:	w, the width of T
Question:	Is I satisfiable?

FPT for every fixed D [Dechter, Pearl, '89]
poly kernel?

D-CSP(tw)

Theorem 7

$\{0,1\}$-CSP(tw) has no polynomial kernel unless NP $\subseteq$ coNP/poly*.*

D-CSP(tw)

Theorem 7

$\{0,1\}$-CSP(tw) has no polynomial kernel unless NP $\subseteq$ coNP/poly*.*

Proof.

OR-composition. Given $(I_i = (X_i, \{0,1\}, C_i), T_i)$, $1 \leq i \leq t$, construct new constraint network $I = (X, \{0,1\}, C)$ with $X = \bigcup_{i=1}^{t} X_i \cup \{a_1, \ldots, a_t\} \cup \{b_0, \ldots, b_t\}$

D-CSP(tw)

Theorem 7

$\{0,1\}$-CSP(tw) has no polynomial kernel unless NP $\subseteq$ coNP/poly.

Proof.

OR-composition. Given $(I_i = (X_i, \{0,1\}, C_i), T_i)$, $1 \leq i \leq t$, construct new constraint network $I = (X, \{0,1\}, C)$ with $X = \bigcup_{i=1}^{t} X_i \cup \{a_1, \ldots, a_t\} \cup \{b_0, \ldots, b_t\}$ and constraints

- $c^0 = ((b_0), (0))$ and $c^1 = ((b_t), (1))$,
- $c_i^* = ((b_{i-1}, b_i, a_i), ((0,0,1), (0,1,0), (1,1,1)))$ for each $1 \leq i \leq t$,

D-CSP(tw)

Theorem 7

$\{0,1\}$-CSP(tw) has no polynomial kernel unless NP $\subseteq$ coNP/poly.

Proof.

OR-composition. Given $(I_i = (X_i, \{0,1\}, C_i), T_i)$, $1 \leq i \leq t$, construct new constraint network $I = (X, \{0,1\}, C)$ with $X = \bigcup_{i=1}^{t} X_i \cup \{a_1, \ldots, a_t\} \cup \{b_0, \ldots, b_t\}$ and constraints

- $c^0 = ((b_0), (0))$ and $c^1 = ((b_t), (1))$,
- $c_i^* = ((b_{i-1}, b_i, a_i), ((0,0,1), (0,1,0), (1,1,1)))$ for each $1 \leq i \leq t$,
- for each $1 \leq i \leq t$ and each $((x_1, \ldots, x_r), R) \in C_i$, add a new constraint $((x_1, \ldots, x_r, a_i), R')$ where $R' = \{(u_1, \ldots, u_r, 0) : (u_1, \ldots, u_r) \in R\} \cup \{(1, \ldots, 1)\}$.

D-CSP(tw)

Theorem 7

$\{0,1\}$-CSP(tw) has no polynomial kernel unless NP $\subseteq$ coNP/poly.

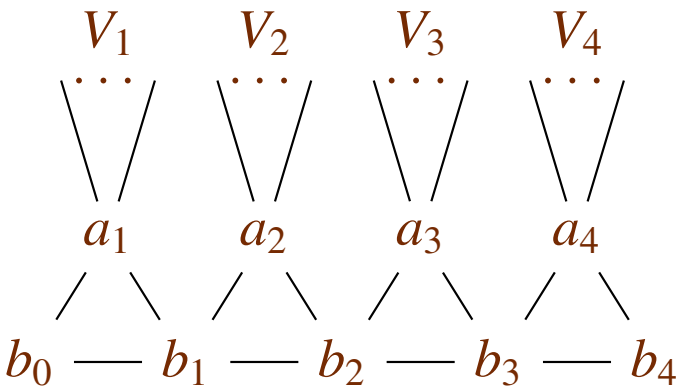
Proof.

OR-composition. Given $(I_i = (X_i, \{0,1\}, C_i), T_i)$, $1 \leq i \leq t$, construct new constraint network $I = (X, \{0,1\}, C)$ with $X = \bigcup_{i=1}^{t} X_i \cup \{a_1, \ldots, a_t\} \cup \{b_0, \ldots, b_t\}$ and constraints

- $c^0 = ((b_0), (0))$ and $c^1 = ((b_t), (1))$,
- $c_i^* = ((b_{i-1}, b_i, a_i), ((0,0,1), (0,1,0), (1,1,1)))$ for each $1 \leq i \leq t$,
- for each $1 \leq i \leq t$ and each $((x_1, \ldots, x_r), R) \in C_i$, add a new constraint $((x_1, \ldots, x_r, a_i), R')$ where $R' = \{(u_1, \ldots, u_r, 0) : (u_1, \ldots, u_r) \in R\} \cup \{(1, \ldots, 1)\}$.

Construct width-$(w+1)$ tree decomposition of I's constraint graph. □

D-CSP(width)



D-CSP(width)

Note:

TREEWIDTH is FPT [Bodlaender '96]

TREEWIDTH has no poly kernel [Bodlaender, Downey, Fellows, Hermelin, '09], [Drucker '12]

Outline

Outline

CSP

Definition 8 (Constraint Satisfaction Problem (CSP))

Given:

- a set of variables X,
- a set of values D,
- a domain $dom(x) \subseteq D$ for each variable $x \in X$, and
- a set of constraints, where a constraint C has a scope $scope(C) \subseteq X$ and specifies the allowed combinations of values for the variables in $scope(C)$.

Question: Is there an assignment $\alpha : X \to D$ such that

- $\alpha(x) \in dom(x)$ for each variable $x \in X$ and
- all constraints are satisfied.

Global Constraints

A constraint C is specified either

- extensionally, by listing all legal instantiations of $scope(C)$, or
- intensionally, by giving an expression involving the variables in $scope(C)$.

Global Constraints

A constraint C is specified either

- extensionally, by listing all legal instantiations of $scope(C)$, or
- intensionally, by giving an expression involving the variables in $scope(C)$.

Global Constraints

Intensionally specified constraints involving an arbitrary number of variables.

Examples of Global Constraints

Some global constraints

- AllDifferent(Y) requires that all variables of Y receive pairwise distinct values.
- NValue(Y, N) requires that the number of distinct values taken by the variables of Y is in $dom(N)$.
- Disjoint(Y, Z) requires that no variable from Y takes the same value as a variable in Z.
- Uses(Y, Z) requires that for each variable in Y, there is a variable in Z that takes the same value.
- EGC(Y, L), with $L : \bigcup_{y \in Y} dom(y) \rightarrow 2^{\mathbb{N}}$, requires that for each $d \in \bigcup_{y \in Y} dom(y)$, the number of variables of Y that take the value d is in $L(d)$.

... many more (currently 423) in the Global Constraint Catalog.

Consistency of Global Constraints

Definition 9 (Consistent)

A global constraint is consistent if there is a legal instantiation of its variables.

Definition 10 (Domain Consistent)

A global constraint C is domain consistent if for each $x \in scope(C)$ and each $v \in dom(x)$, there is a legal instantiation α such that $\alpha(x) = v$ (in that case, C supports v for x).

CONSISTENCY problem: check whether a constraint is consistent.

FILTERING problem: remove from the domain of each variable the values that are not supported.

Consitency of NVALUE

NVALUE

NVALUE(Y, N) requires that the number of distinct values taken by the variables of Y is in $dom(N)$.

- NP-hard
- FPT with parameter $k = |\bigcup_{y \in Y} dom(y)|$
  [Bessière, Hebrard, Hnich, Kiziltan, Quimper, Walsh '08]
- FPT with parameter $k = |Y|$
- W[2]-hard with parameter $k = \max(dom(N))$ [BHHKQW '08]
- polynomial if all domains are intervals [Beldiceanu '01]
- FPT with parameter $k = \#\text{holes}(Y)$ [BHHKQW '08]

Decomposition of NVALUE

[BHHKW '06] and [Beldiceanu '01] decompose NVALUE into:

ATLEAST-NVALUE

ATLEAST-NVALUE(Y, N) requires that the number of distinct values taken by the variables of Y is $\geq N$.

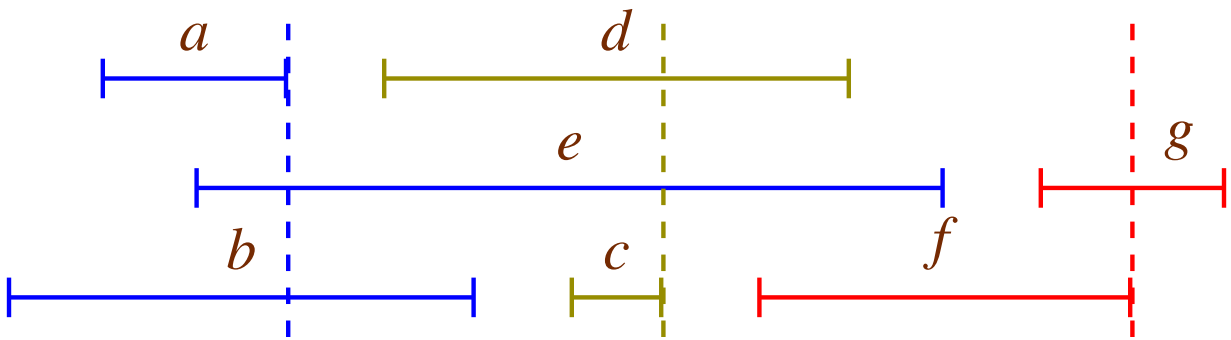
Consistency for ATLEAST-NVALUE is polynomial.

ATMOST-NVALUE

ATMOST-NVALUE(Y, N) requires that the number of distinct values taken by the variables of Y is $\leq N$.

Consistency for ATMOST-NVALUE is NP-hard.

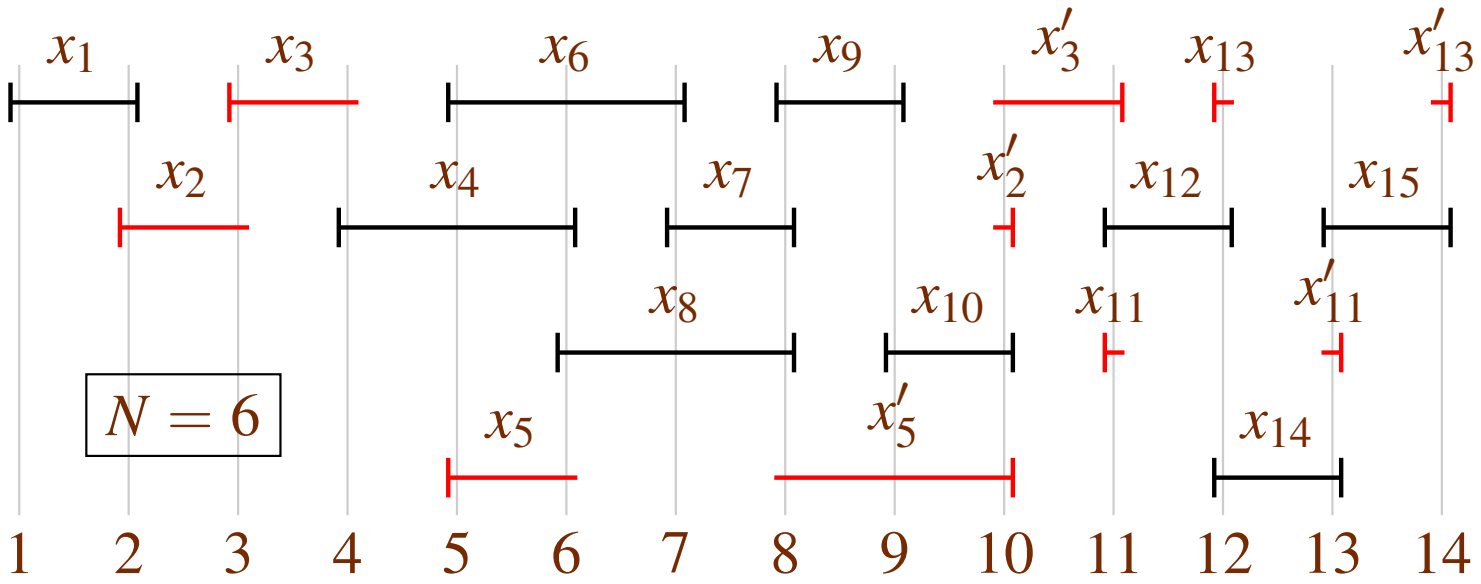
AtMost-NValue with interval domains



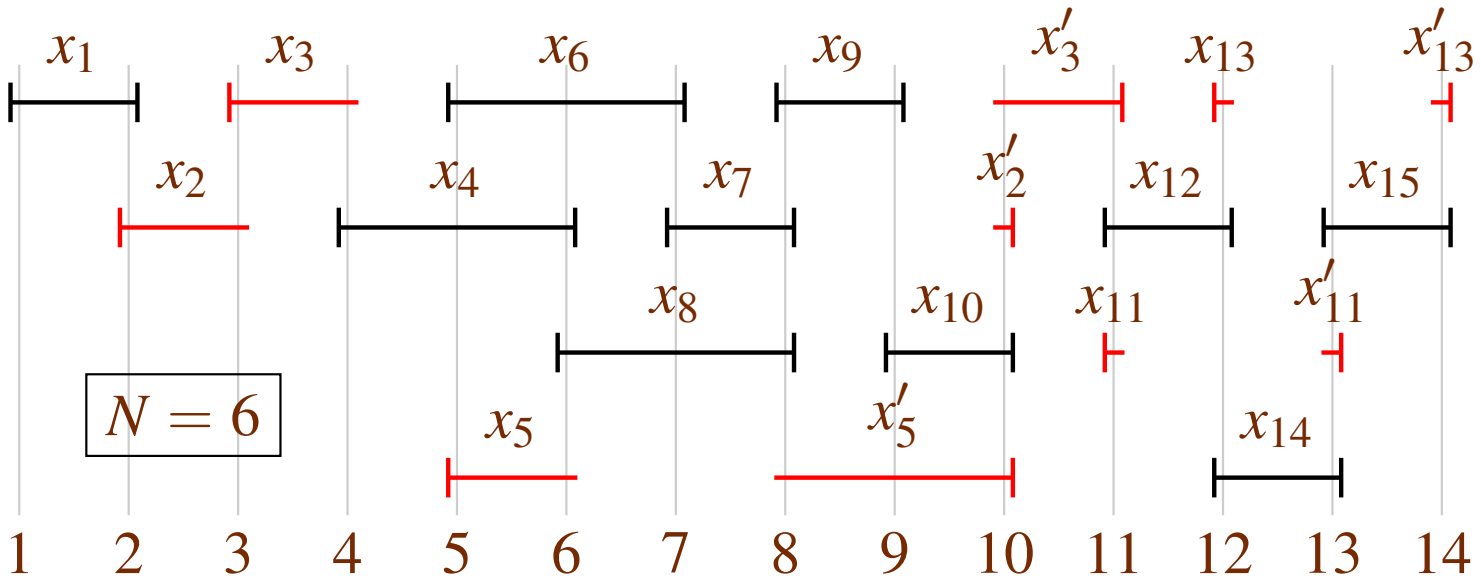
Clique Cover for interval graphs.

Outline

Kernel for AtMost-NValue-Consistency

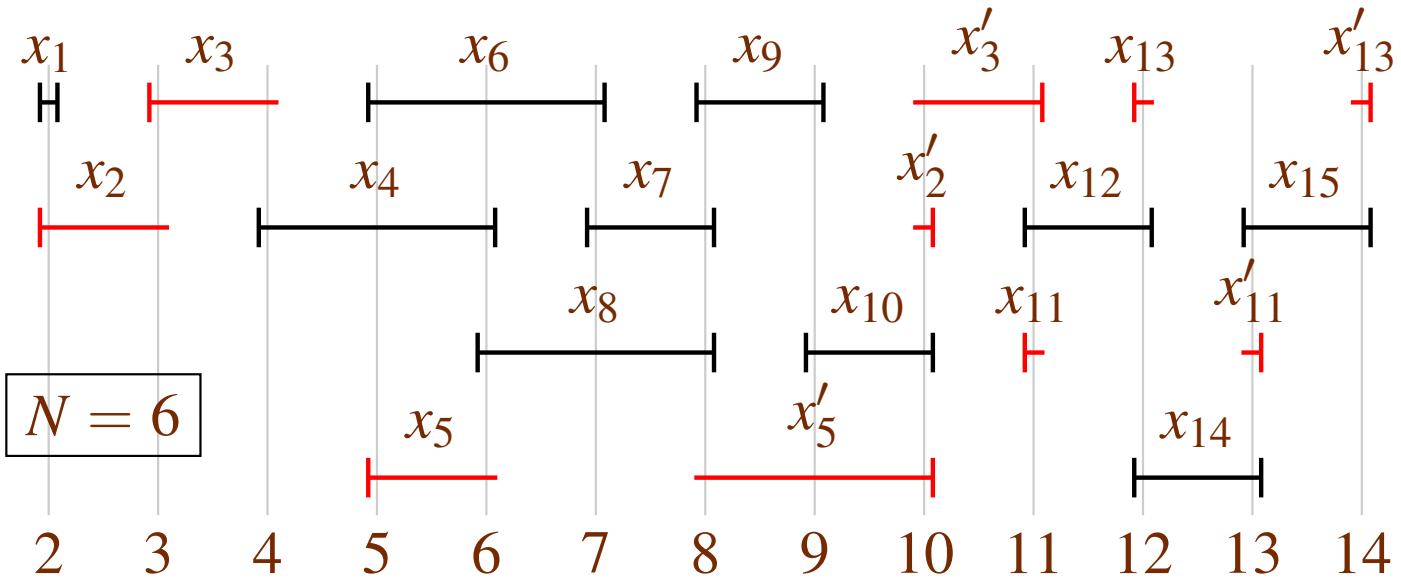


Kernel for AtMost-NValue-Consistency

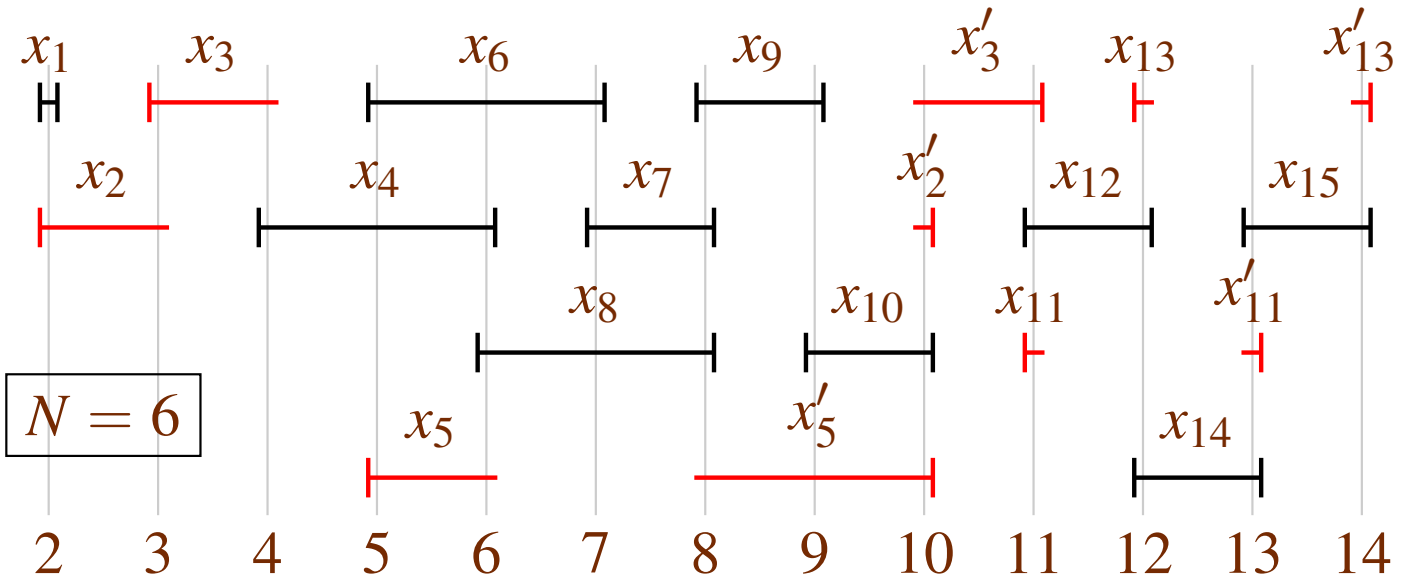


$ivl(1) \subseteq ivl(2)$

Kernel for AtMost-NValue-Consistency

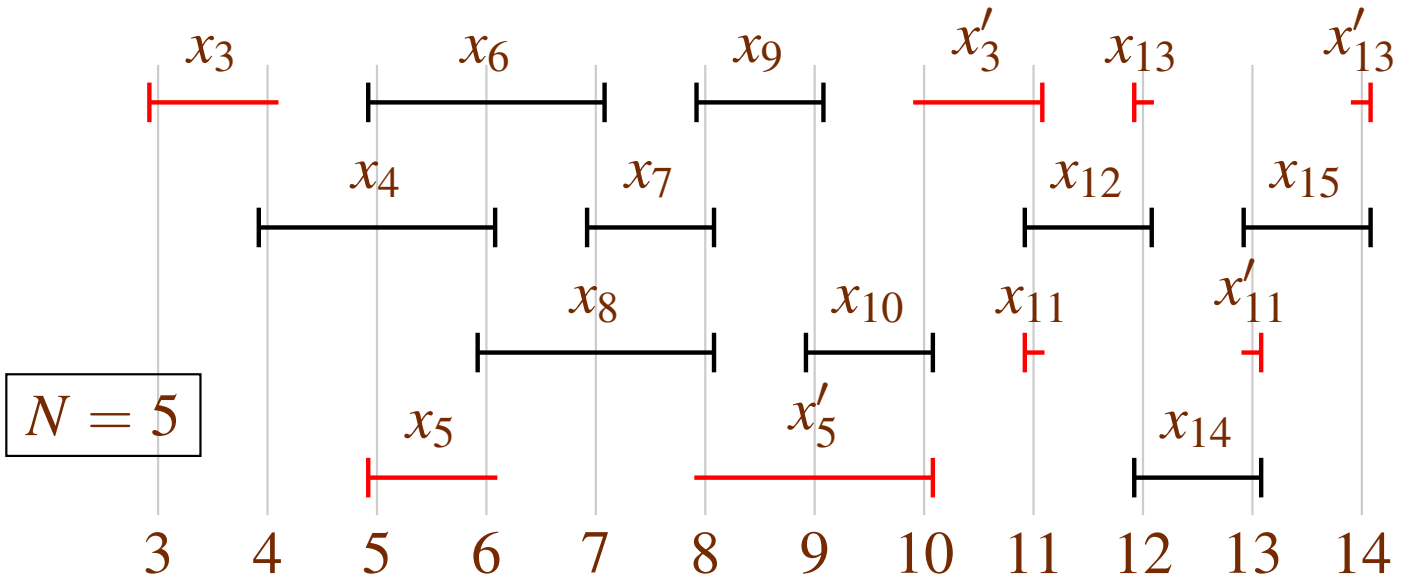


Kernel for AtMost-NValue-Consistency

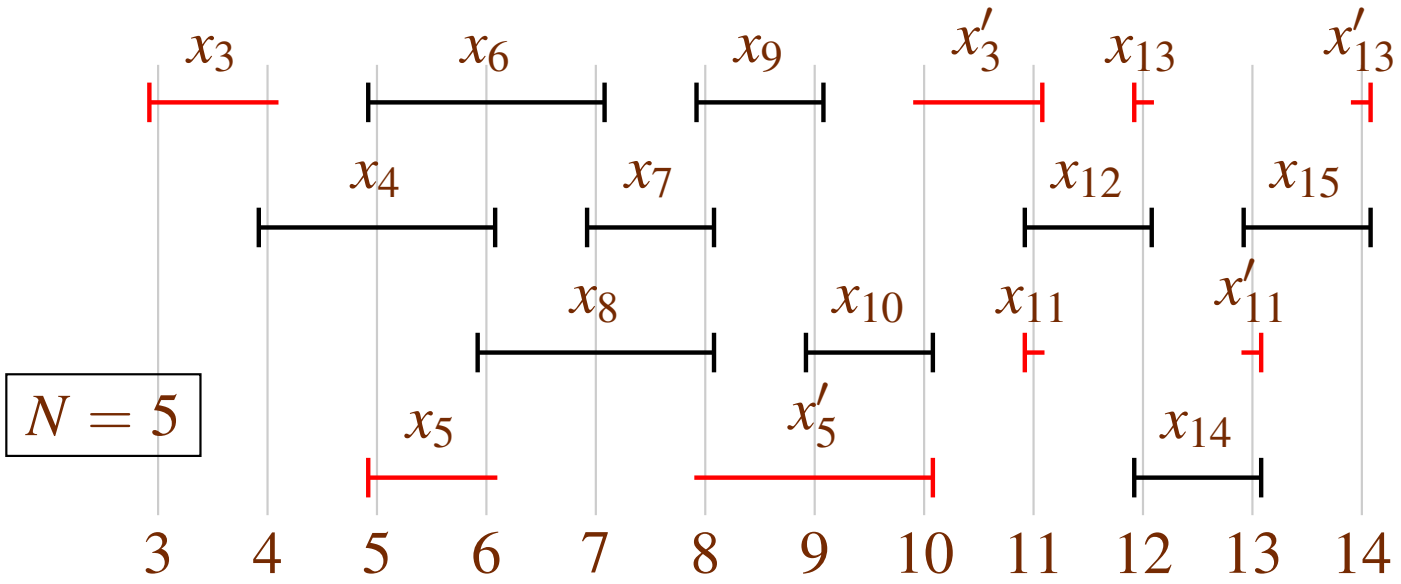


$|dom(x_1)| = 1$

Kernel for AtMost-NValue-Consistency



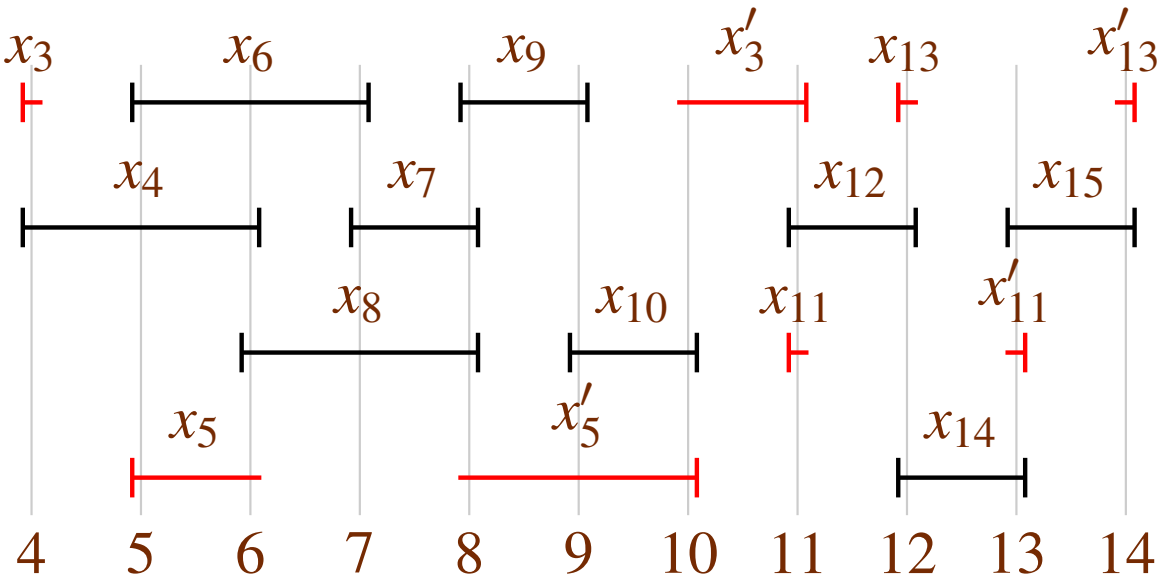
Kernel for AtMost-NValue-Consistency



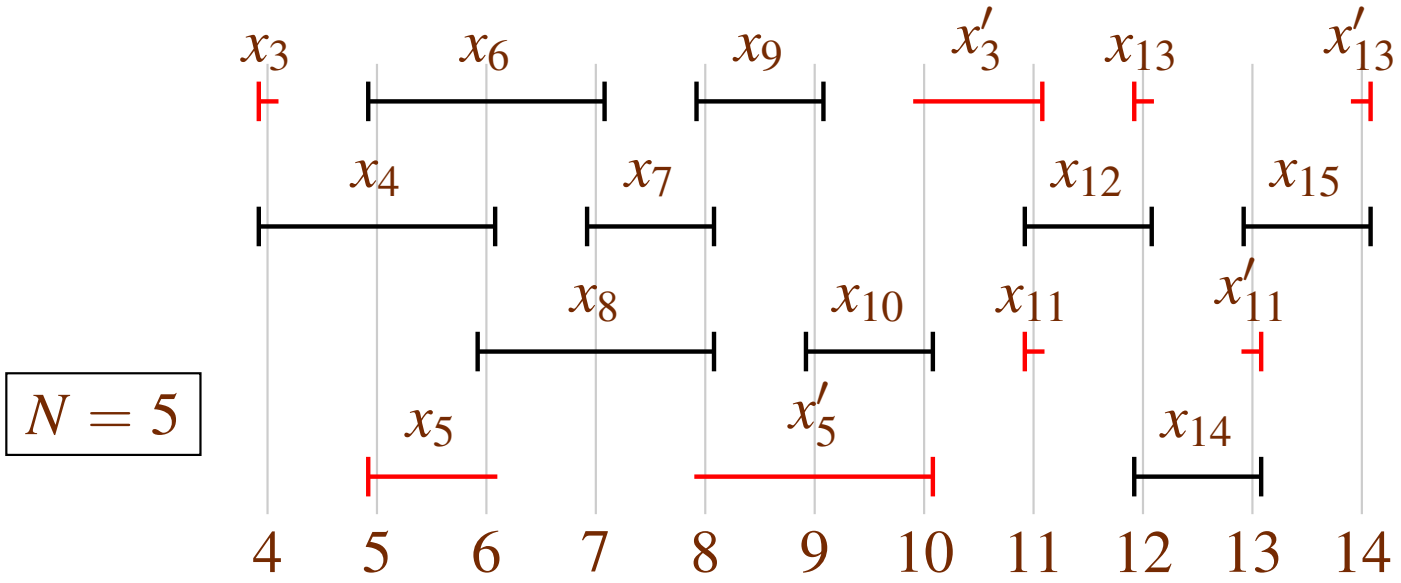
$ivl(3) \subseteq ivl(4)$

Kernel for AtMost-NValue-Consistency

$N = 5$



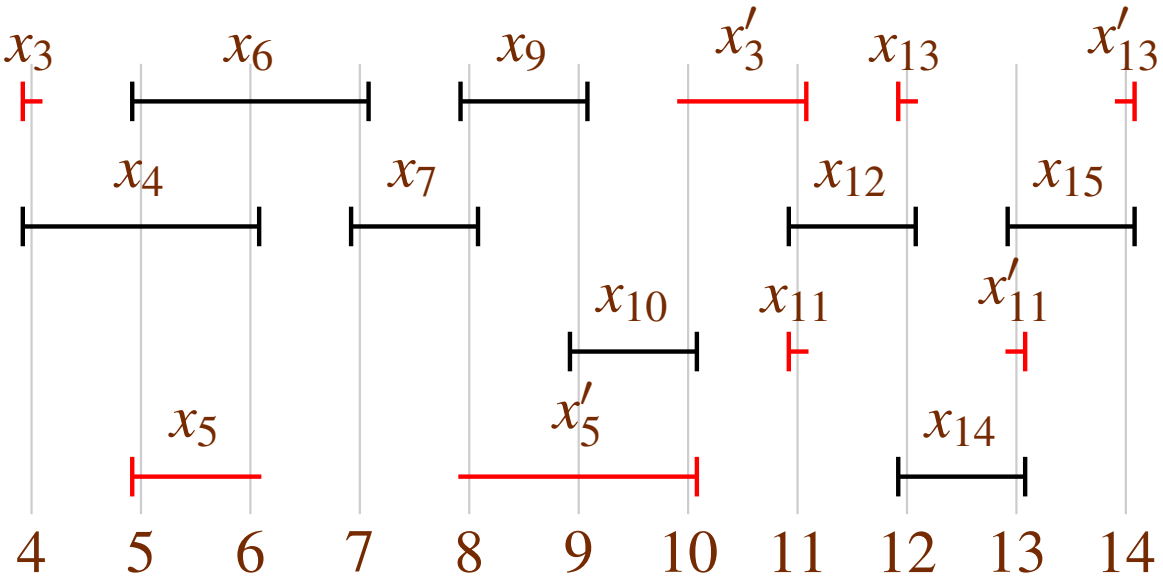
Kernel for AtMost-NValue-Consistency



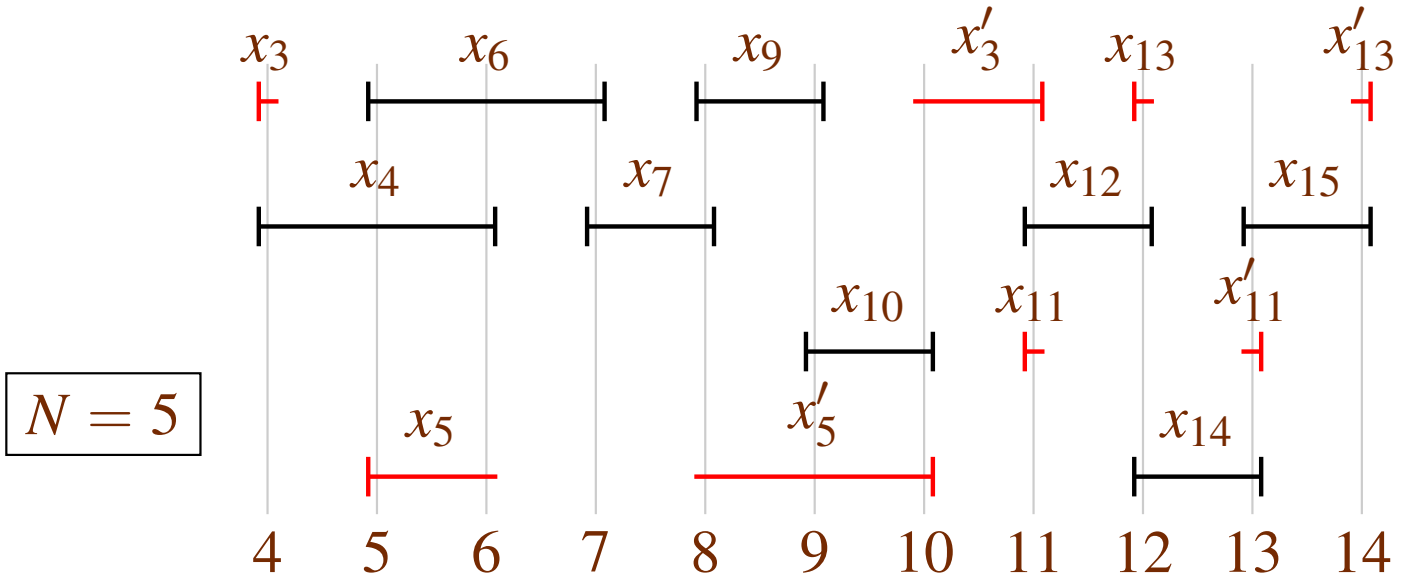
$dom(x_7) \subseteq dom(x_8)$

Kernel for AtMost-NValue-Consistency

$N = 5$

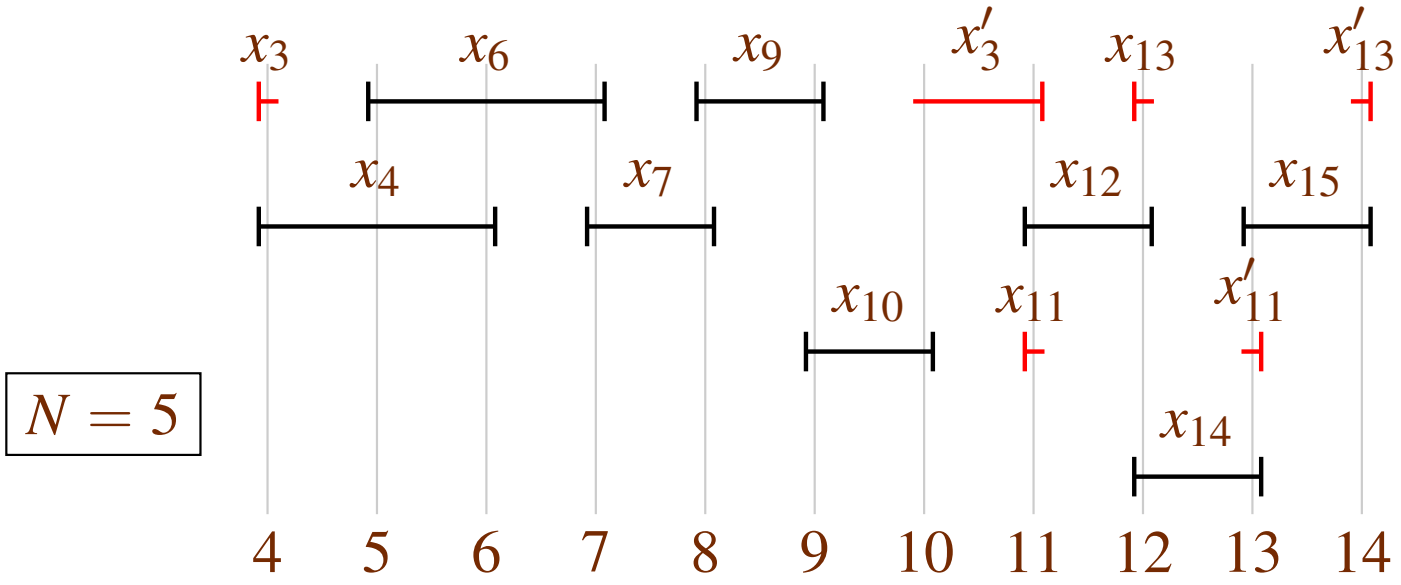


Kernel for AtMost-NValue-Consistency

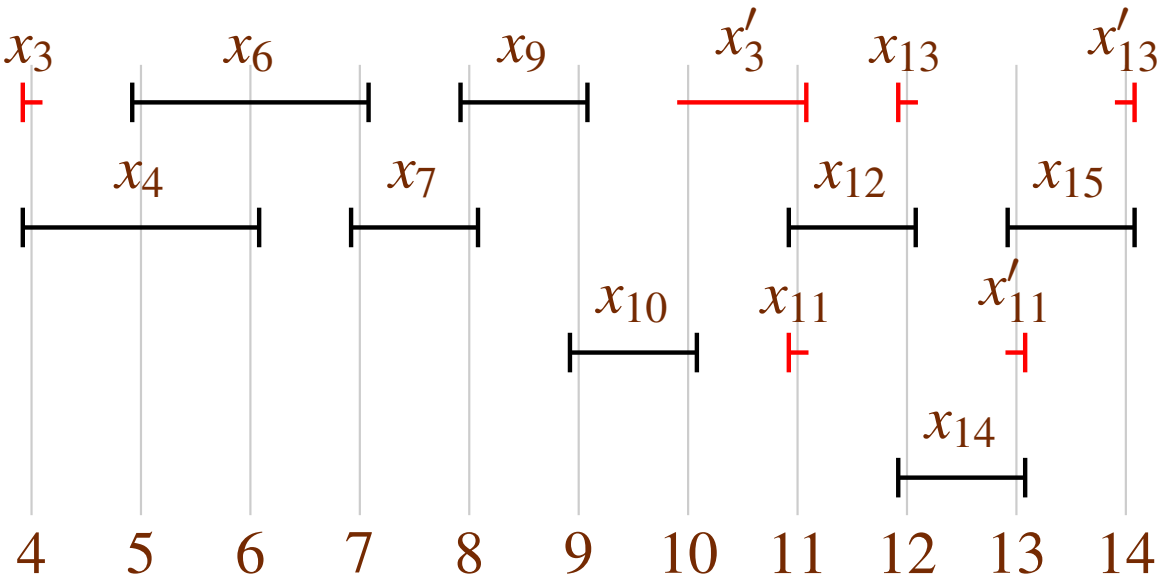


$dom(x_{10}) \subseteq dom(x_5)$

Kernel for AtMost-NValue-Consistency



Kernel for AtMost-NValue-Consistency

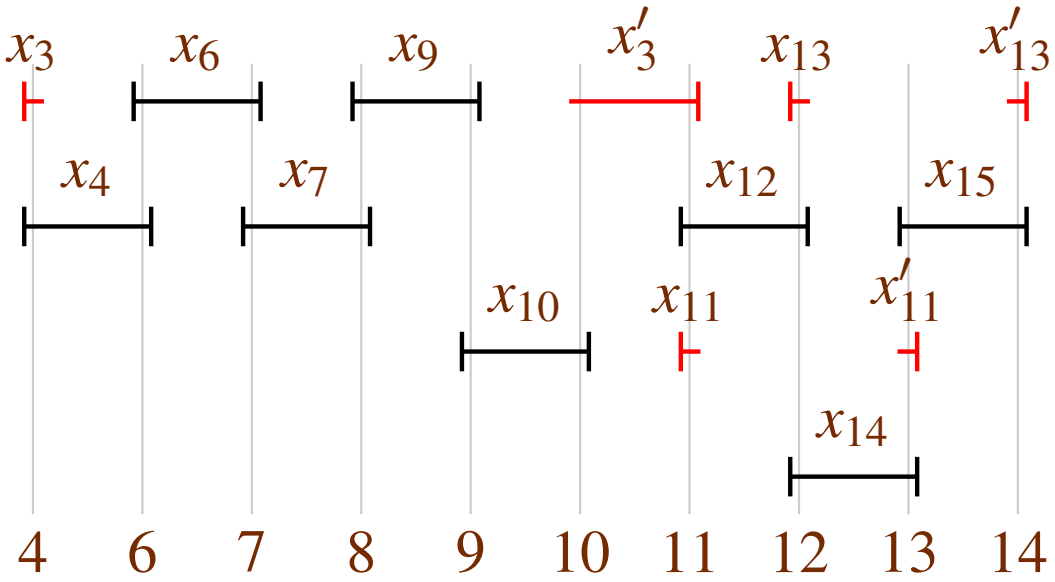


$N = 5$

$ivl(5) \subseteq ivl(6)$

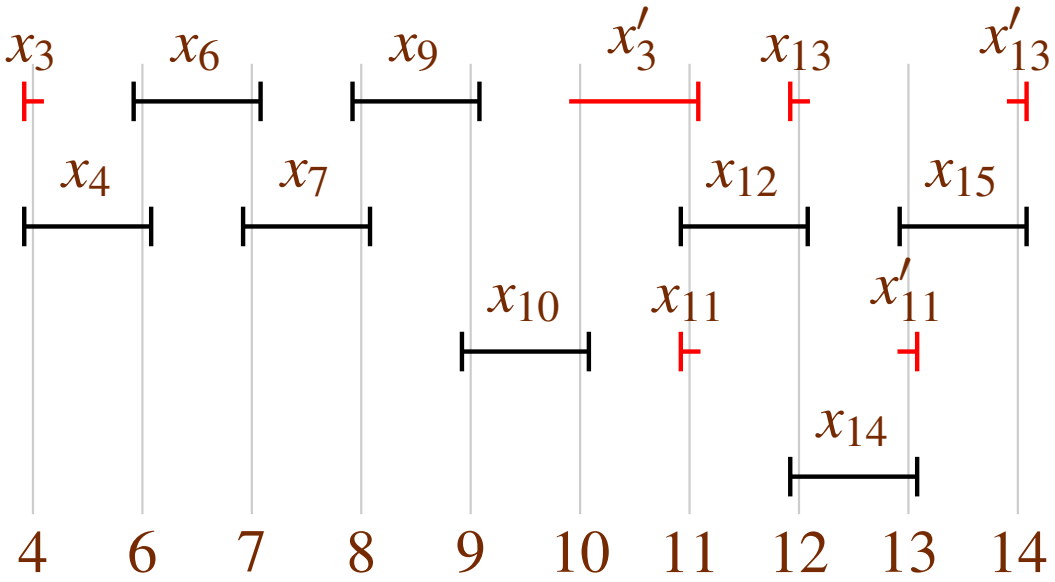
Kernel for AtMost-NValue-Consistency

$N = 5$



Kernel for AtMost-NValue-Consistency

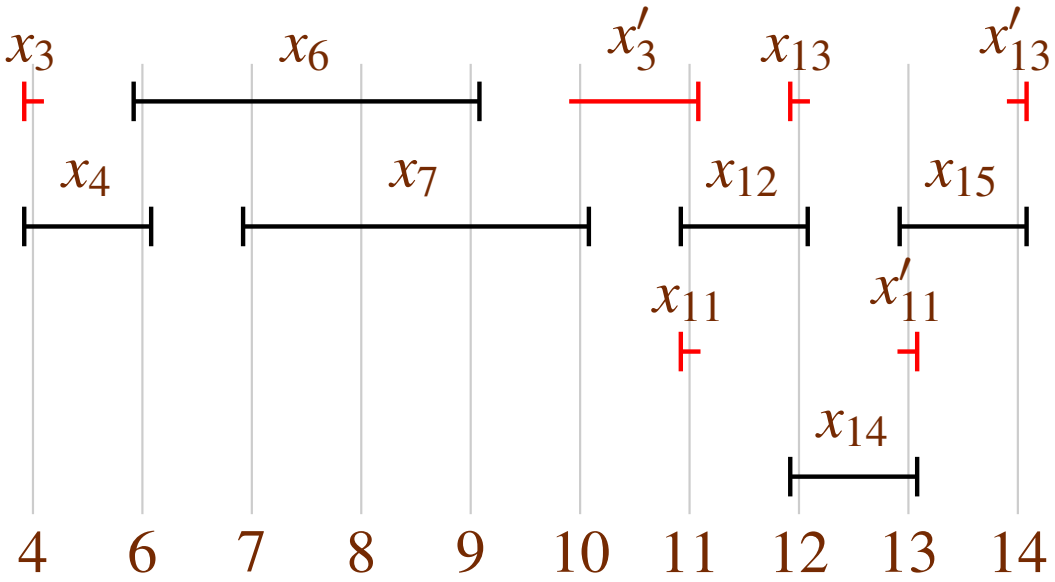
$N = 5$



select $4 \rightarrow$ select 7 and 9
discard $4 \rightarrow$ select 6, 8, and 10

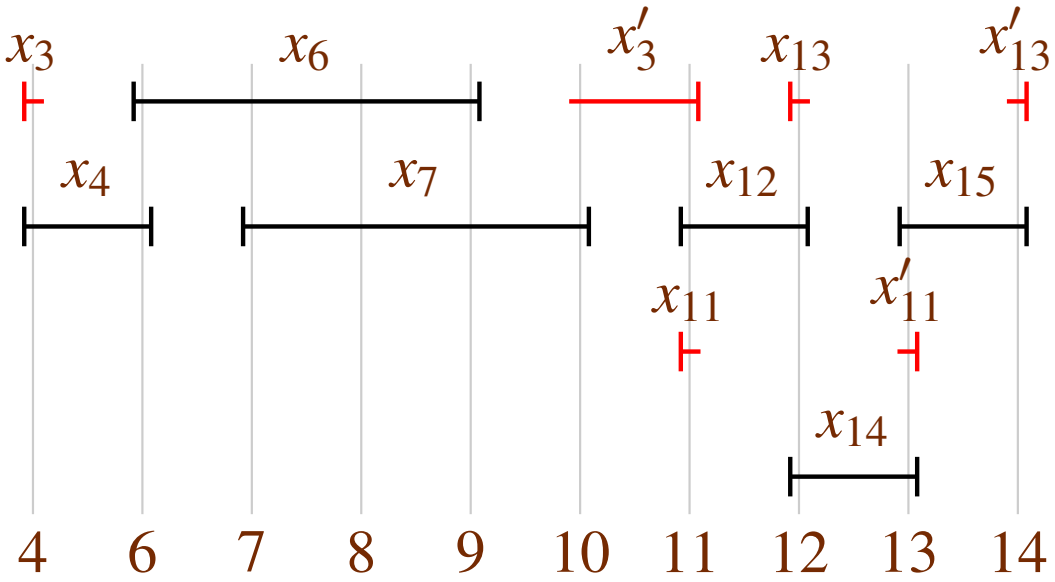
Kernel for AtMost-NValue-Consistency

$N = 4$



Kernel for AtMost-NValue-Consistency

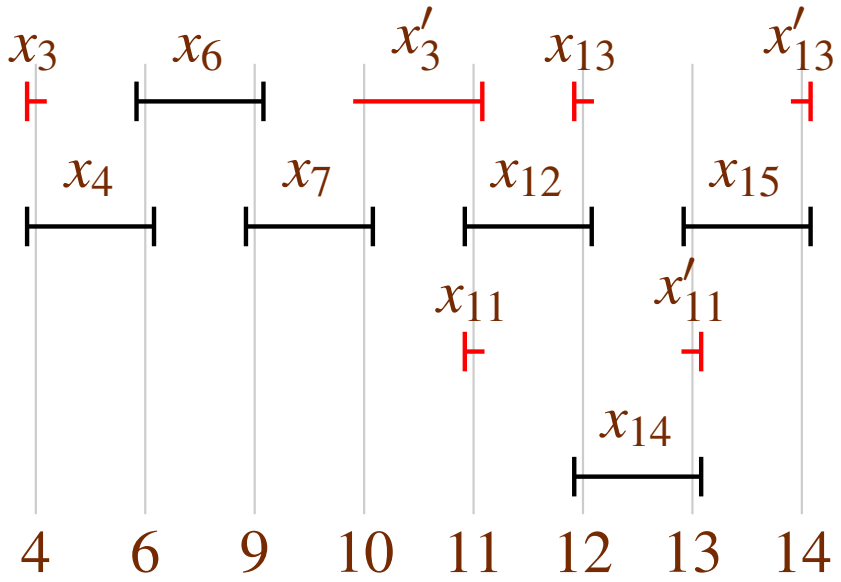
$N = 4$



$ivl(7) \subseteq ivl(8) \subseteq ivl(9)$

Kernel for AtMost-NValue-Consistency

$N = 4$



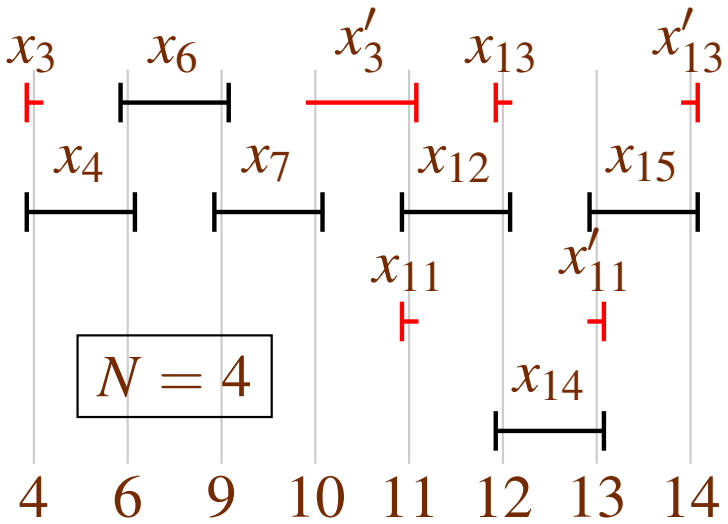
Kernel for AtMost-NValue

Theorem 11

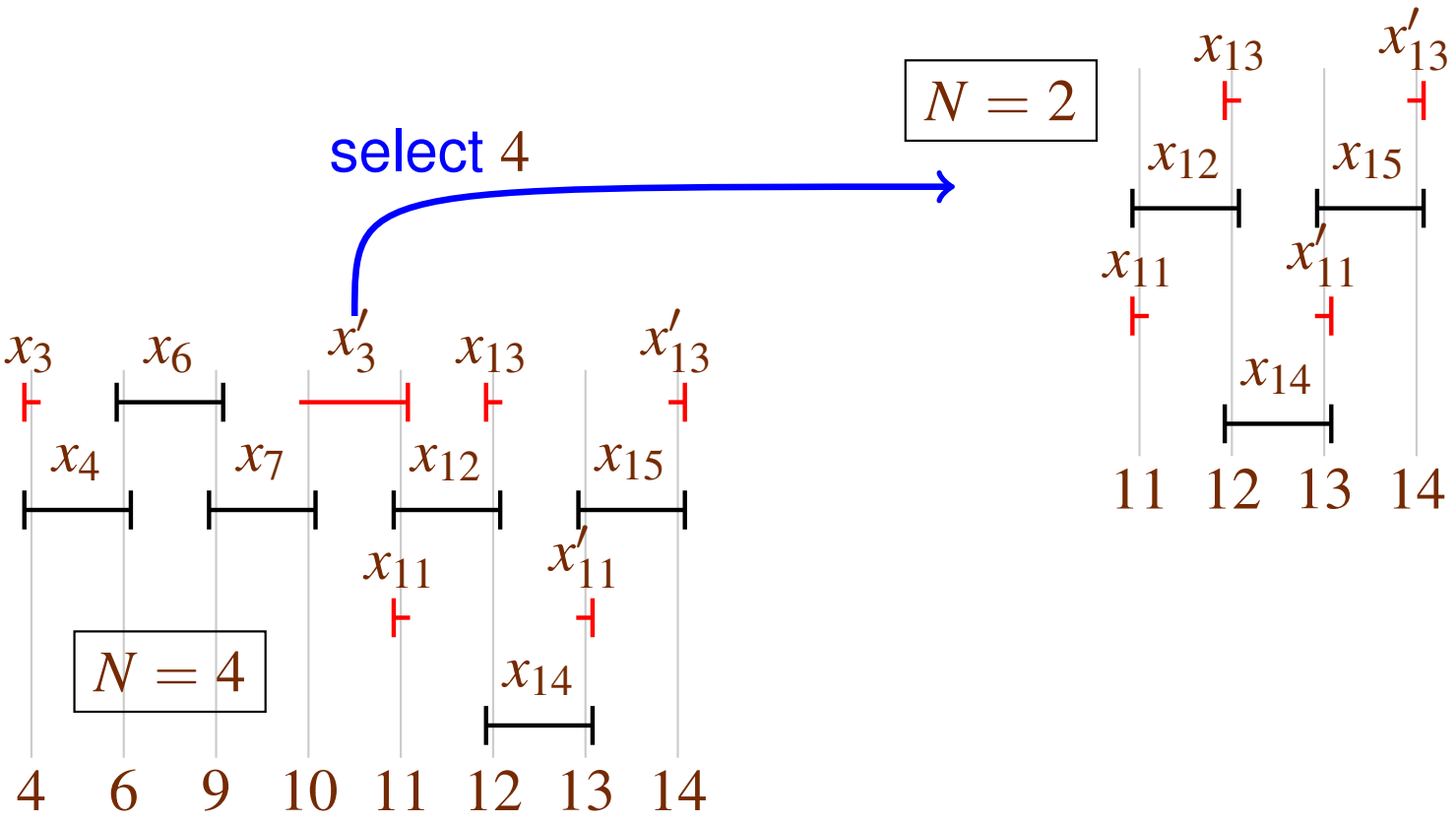
The consistency problem for AtMost-NValue *constraints, parameterized by the number* k *of holes, has a linear time computable kernel with* $O(k^2)$ *variables and* $O(k^2)$ *domain values.*

Outline

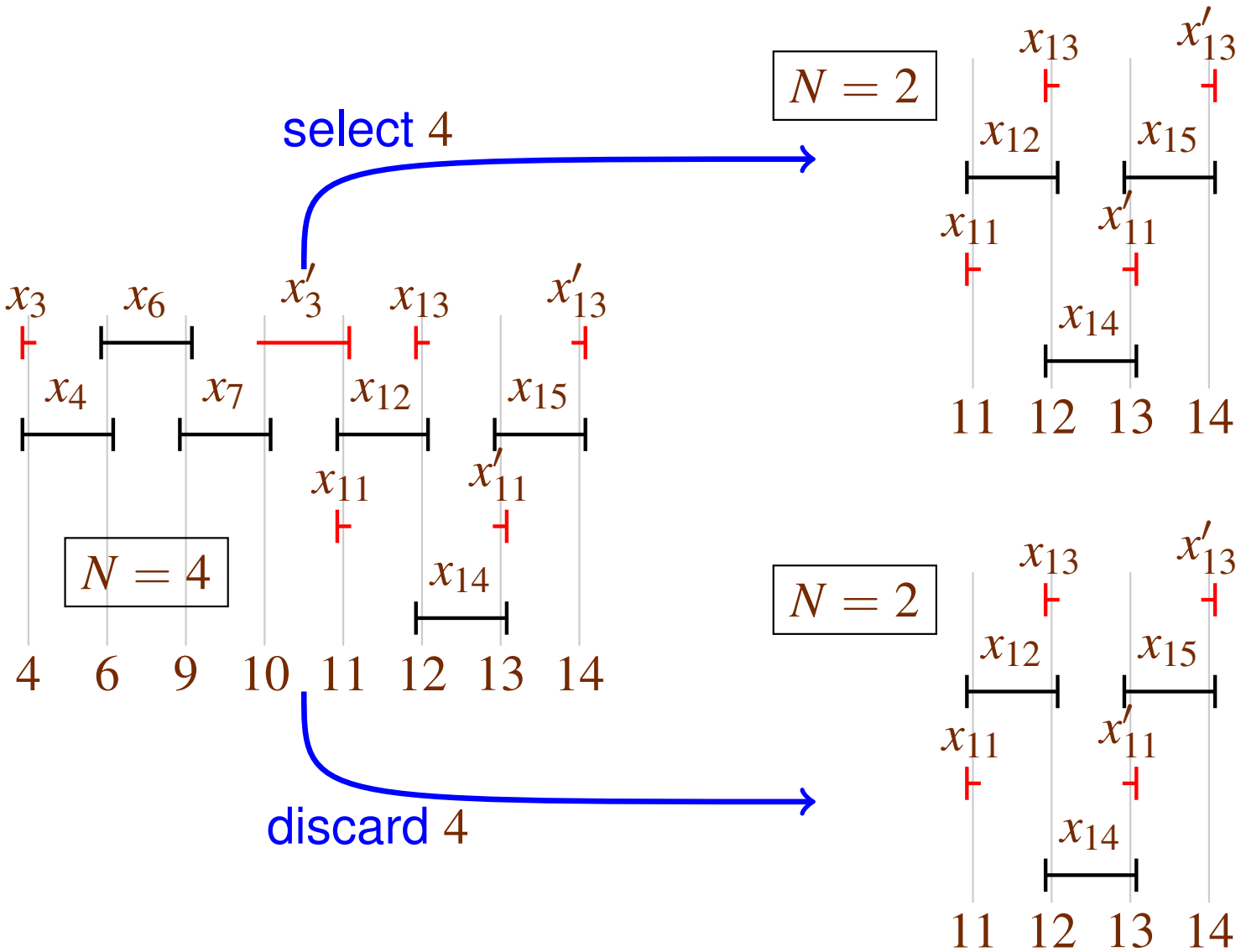
FPT algorithm for AtMost-NValue



FPT algorithm for AtMost-NValue



FPT algorithm for AtMost-NValue



FPT algorithms for AtMost-NValue

For input instances $\mathcal{I} = (X, D, dom, N)$ where k is the number of holes in the domains, and $\rho = \frac{1+\sqrt{5}}{2} < 1.6181$:

Theorem 12

The Consistency problem for AtMost-NValue *constraints can be solved in time* $O(\rho^k k^2 + |\mathcal{I}|)$.

Corollary 13

The Filtering problem for AtMost-NValue *constraints can be solved in time* $O(\rho^k \cdot k^2 \cdot |D| + |\mathcal{I}| \cdot |D|)$.

Outline

EGC

Extended Global Cardinality (EGC)

EGC(Y, L), with $L : \bigcup_{y \in Y} dom(y) \rightarrow 2^{\mathbb{N}}$, requires that for each $d \in \bigcup_{y \in Y} dom(y)$, the number of variables of Y that take the value d is in $L(d)$.

- FPT with parameter $k = \#\text{holes}(L)$ (holes in cardinality lists $L(\cdot)$) [BHHKQW '08]

Kernel lower bound for EGC

Theorem 14

EGC-*consistency(holes)*, NVALUE-*consistency*($|dom(X)|$), DISJOINT-*consistency*($|dom(X) \cap dom(Y)|$), *and* USES-*consistency*($|dom(Y)|$) *have no polynomial kernel unless* NP $\subseteq$ coNP/poly.

- Polynomial parameter transformations from SAT(vars)
- All FPT [BHHKQW '08]

Outline

Bayesian Networks

Definition 15 (Boolean Bayesian Network)

A Boolean Bayesian Network (BBN) is a DAG with a table T_v for each node v, specifying the probability of v being true given the values of v's in-neighbors.

The probability of an instantiation U of the variables is the product of the probabilities of all variables.

Positive-BBN-Inference(fvs)

Input:	DAG $G = (V, E)$, a set T of $\|V\|$ tables, a vertex $v \in V$, and a feedback vertex set S of $\hat{G}$ (underlying undirected graph)
Parameter:	$k = \|S\|$
Question:	Is $Pr(v = \text{true}) > 0$?

Bayesian Networks

Note: FVS is FPT and has a polynomial kernel [Thomassé '10]

Positive-BBN-Inference(fvs) is FPT by going through all instantiations of S and using a polynomial time algorithm for the acyclic case [Pearl '88].

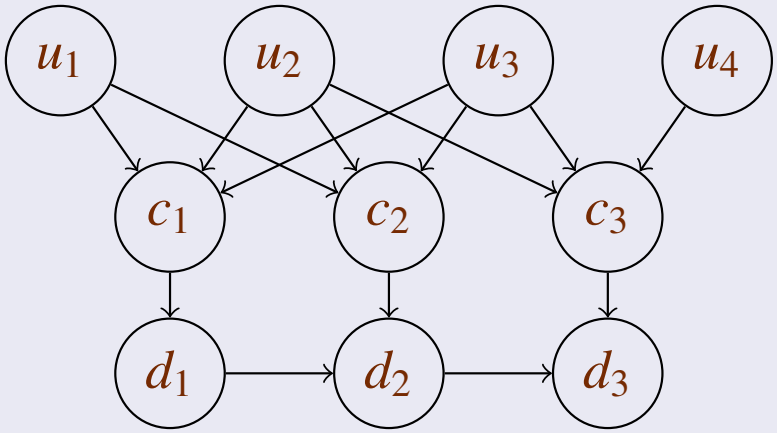
Bayesian Networks

Theorem 16

Positive-BBN-Inference(fvs) has no polynomial kernel unless NP $\subseteq$ coNP/poly.

Proof.

Polynomial parameter transformation from SAT(vars).
Inspired by [Cooper '90]'s reduction from 3SAT.



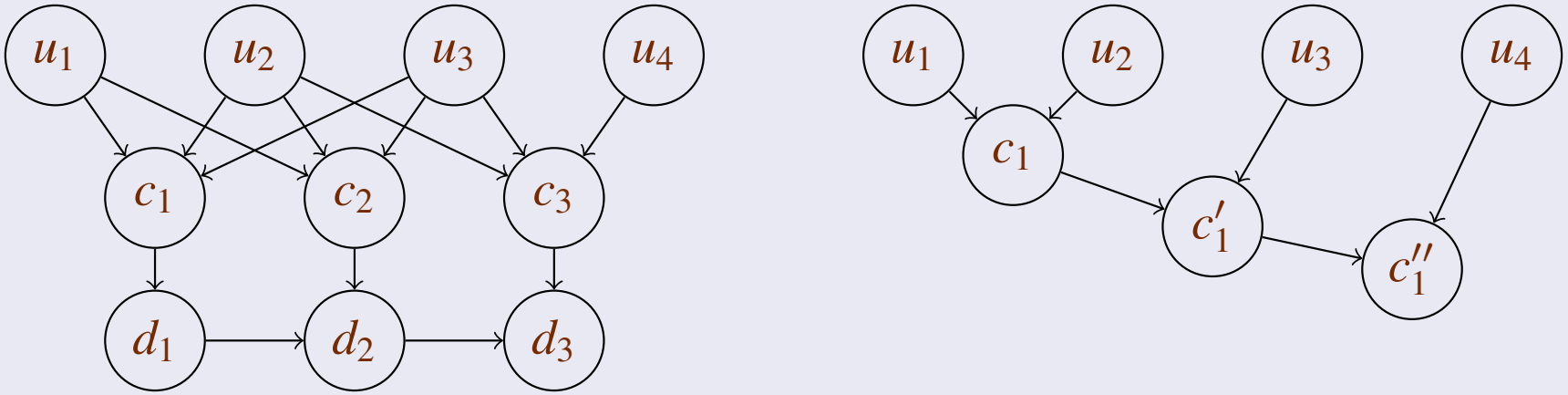
Bayesian Networks

Theorem 16

Positive-BBN-Inference(fvs) has no polynomial kernel unless NP $\subseteq$ coNP/poly.

Proof.

Polynomial parameter transformation from SAT(vars).
Inspired by [Cooper '90]'s reduction from 3SAT.



Outline

Logic programs

Definition 17

A logic program P is a set of rules r over atoms V of the form

$$h \longleftarrow a_1 \wedge \cdots \wedge a_m \wedge \neg b_1 \wedge \cdots \wedge \neg b_n,$$

where we write $H(r) = h$, $B^+(r) = \{a_1, \ldots, a_m\}$, and $B^-(r) = \{b_1, \ldots, b_n\}$.
The reduct P^I of P under $I \subseteq V$ is the program obtained from P by removing all rules r with $B^-(r) \cap I \neq \emptyset$, and removing from the body of each remaining rule all literals $\neg b$. I is a stable model of P if I is a minimal model of P^I, i.e., if (i) for each rule $r \in P^I$ with $B^+(r) \subseteq I$ we have $H(r) \in I$, and (ii) there is no proper subset of I with this property.

The dependency graph has vertices V and an edge xy if there exists a rule r with $H(r) = x$ and $y \in B^+(r) \cup B^-(r)$.

Logic programs

StableModel(fvs)

Input:	logic program P, feedback vertex set S of P's dependency graph
Parameter:	$k = \|S\|$
Question:	Does P have a stable model?

FPT [Gottlob, Scarcello, Sideri '02]

Logic programs

Theorem 18

StableModel(fvs) has no polynomial kernel unless NP $\subseteq$ coNP/poly.

Proof.

Polynomial parameter transformation from SAT(vars) by [Niemelä '99].
Given: CNF formula F on n variables.
For each variable x, add atoms $x, \hat{x}$ and rules $(\hat{x} \leftarrow \neg x)$ and $(x \leftarrow \neg \hat{x})$.
Add two atoms s and f and the rule $(f \leftarrow s \wedge \neg f)$.
For each clause C, add atom c, rule $(s \leftarrow \neg c)$, and for each positive literal a of C the rule $(c \leftarrow a)$, and for each negative literal $\neg a$ of C the rule $(c \leftarrow \hat{a})$.
The set $\{x, \hat{x} : x \in \mathsf{vars}(F)\}$ is a feedback vertex set of size $2n$. □

Outline

Conclusion

- Other kernelization work in AI and reasoning includes
  - Abductive reasoning [Fellows, Pfandler, Rosamond, Rümmele, '12]
  - Planning [Bäckström, Jonsson, Ordyniak, Szeider, '13]
- Typical AI problems do not seem to have polynomial kernels
- But preprocessing is incredibly successful
- $\Rightarrow$ explore Turing kernels, stronger parameterizations, etc.

Thank you!

Questions?

Comments?