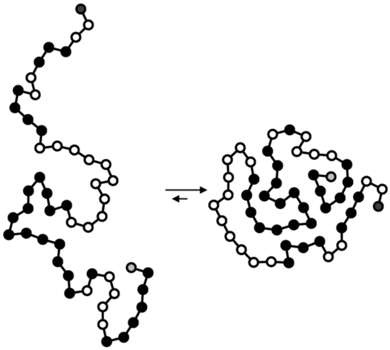
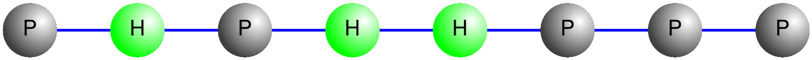
Can we predict protein structure?

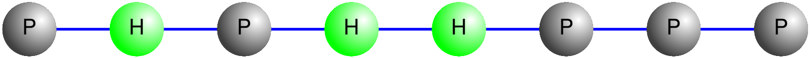


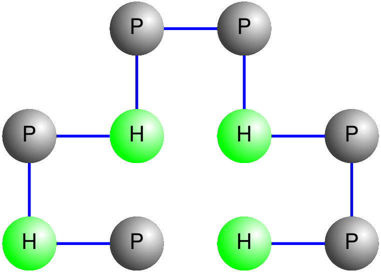
- Molecular Dynamics on Full Atom Models
- Simpler Protein Models:
  - Folding simulation
  - Stochastic optimization, e.g. Genetic Algorithms
  - Combinatorial optimization, e.g. Constraint Programming

Simple Proteins: HP-Model

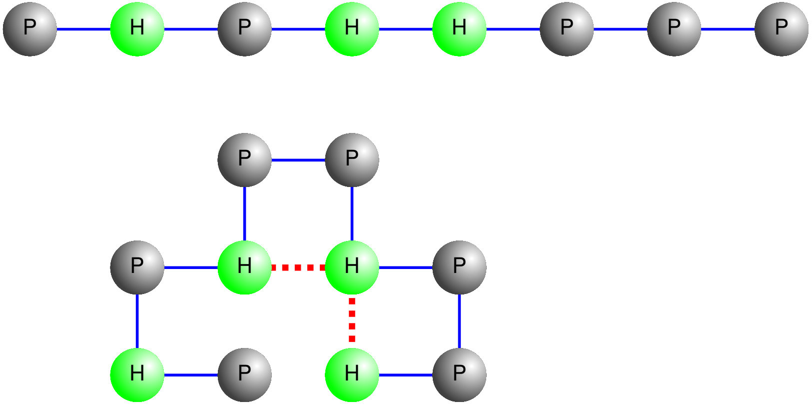


Simple Proteins: HP-Model

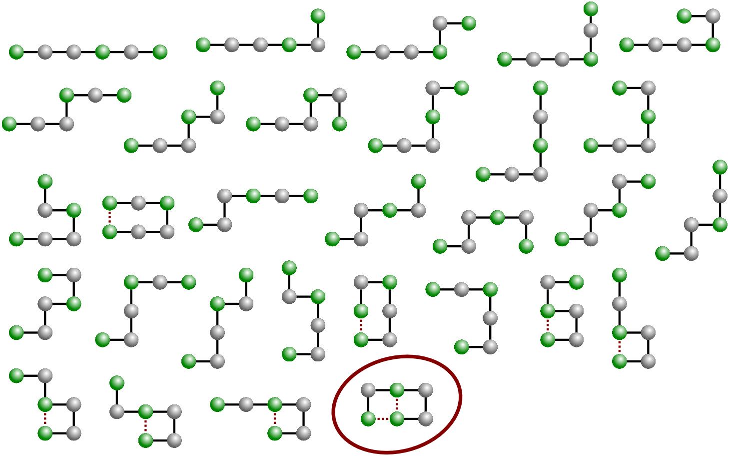




Simple Proteins: HP-Model



Sequence HPPHPH



Constraint Programming

Constraint programming . . .

- . . . is a programming technique
- . . . describes what rather than how
- . . . i.e. it is declarative
- . . . combines logic reasoning with search
- . . . performs “intelligent” enumeration
- . . . “slays NP-hard dragons”

Well, But What Are Constraints?

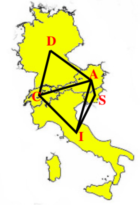
Example: Map Coloring

Constraints:
$A, C, D, I, S \in \{red, green, blue\}$,
$A \neq C$, $A \neq D$, $A \neq I$, $A \neq S$,
$C \neq D$, $C \neq I$, $I \neq S$

- We say only what a solution of the map coloring is
- We need not care how the problem is solved
- A solution is computed by guessing and reasoning
  E.g. guess $A =$ red implies $C, D, I, S \neq$ red;
  then guess $C =$ green ...

Well, But What Are Constraints?

Example: Map Coloring

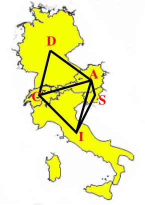


Constraints:
$A, C, D, I, S \in \{red, green, blue\}$,
$A \neq C$, $A \neq D$, $A \neq I$, $A \neq S$,
$C \neq D$, $C \neq I$, $I \neq S$

- We say only what a solution of the map coloring is
- We need not care how the problem is solved
- A solution is computed by guessing and reasoning
  E.g. guess $A =$ red implies $C, D, I, S \neq$ red;
  then guess $C =$ green ...

Well, But What Are Constraints?

Example: Map Coloring

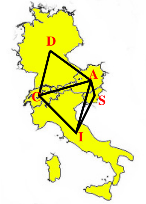


Constraints:
$A, C, D, I, S \in \{red, green, blue\}$,
$A \neq C$, $A \neq D$, $A \neq I$, $A \neq S$,
$C \neq D$, $C \neq I$, $I \neq S$

- We say only what a solution of the map coloring is
- We need not care how the problem is solved
- A solution is computed by guessing and reasoning
  E.g. guess $A =$ red implies $C, D, I, S \neq$ red;
  then guess $C =$ green ...

Well, But What Are Constraints?

Example: Map Coloring

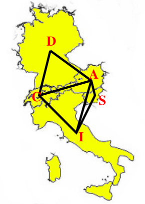


Constraints:
$A, C, D, I, S \in \{red, green, blue\}$,
$A \neq C$, $A \neq D$, $A \neq I$, $A \neq S$,
$C \neq D$, $C \neq I$, $I \neq S$

- We say only what a solution of the map coloring is
- We need not care how the problem is solved
- A solution is computed by guessing and reasoning
  E.g. guess $A =$ red implies $C, D, I, S \neq$ red;
  then guess $C =$ green ...

Well, But What Are Constraints?

Example: Map Coloring

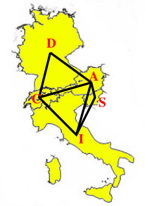


Constraints:
$A, C, D, I, S \in \{red, green, blue\}$,
$A \neq C$, $A \neq D$, $A \neq I$, $A \neq S$,
$C \neq D$, $C \neq I$, $I \neq S$

- We say only what a solution of the map coloring is
- We need not care how the problem is solved
- A solution is computed by guessing and reasoning
  E.g. guess $A =$ red implies $C, D, I, S \neq$ red;
  then guess $C =$ green …

Well, But What Are Constraints?

Example: Map Coloring



Constraints:
$A, C, D, I, S \in \{red, green, blue\}$,
$A \neq C$, $A \neq D$, $A \neq I$, $A \neq S$,
$C \neq D$, $C \neq I$, $I \neq S$

- We say only what a solution of the map coloring is
- We need not care how the problem is solved
- A solution is computed by guessing and reasoning
  E.g. guess $A =$ red implies $C, D, I, S \neq$ red;
  then guess $C =$ green ...

Another Constraints Example

Example

A mathematician forgot the last position of a number code.
She only remembers

- it's odd
- of course, its a digit, i.e. in [0..9]
- it's no prime number and not 1.

She can derive the digit (by constraint reasoning)!

Another Constraints Example

Example

A mathematician forgot the last position of a number code.
She only remembers

- it's odd
- of course, its a digit, i.e. in [0..9]
- it's no prime number and not 1.

She can derive the digit (by constraint reasoning)!

Commercial Impact of Constraints

Some examples

Michelin and Dassault, Renault	Production planning
Lufthansa, Swiss Air, ...	Staff planning
Nokia	Software configuration
Siemens	Circuit verification
French National Railway Company	Train schedule

Constraint Satisfaction Problem (CSP)

Definition

A Constraint Satisfaction Problem (CSP) consists of

- *variables* $\mathcal{X} = \{X_1, \dots, X_n\}$,
- the domain D that associates finite domains $D_1 = D(X_1), \dots, D_n = D(X_n)$ to $\mathcal{X}$.
- a set of constraints C.

A solution is an assignment of variables to values of their domains that satisfies the constraints.

Constraint Satisfaction Problem (CSP)

Definition

A Constraint Satisfaction Problem (CSP) consists of

- *variables* $\mathcal{X} = \{X_1, \ldots, X_n\}$,
- the domain D that associates finite domains $D_1 = D(X_1), \ldots, D_n = D(X_n)$ to $\mathcal{X}$.
- a set of constraints C.

A solution is an assignment of variables to values of their domains that satisfies the constraints.

Constraint Satisfaction Problem (CSP)

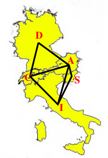
Definition

A Constraint Satisfaction Problem (CSP) consists of

- *variables* $\mathcal{X} = \{X_1, \ldots, X_n\}$,
- the domain D that associates finite domains $D_1 = D(X_1), \ldots, D_n = D(X_n)$ to $\mathcal{X}$.
- a set of constraints C.

A solution is an assignment of variables to values of their domains that satisfies the constraints.

We have already seen one example: map coloring.



A Simple Example CSP

- Variables $\mathcal{X} = \{X, Y, Z\}$
- Domains $D(X) = D(Y) = D(Z) = \{1, 2, 3, 4\}$
- Constraints $C = \{X < Y, Y < Z, Z \leq 2\}$

Remarks

- The constraint set is interpreted as the conjunction

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2.$$

- The domains are interpreted as the constraint

$$X \in D(X) \text{ and } Y \in D(Y) \text{ and } Z \in D(Z).$$

A Simple Example CSP

- Variables $\mathcal{X} = \{X, Y, Z\}$
- Domains $D(X) = D(Y) = D(Z) = \{1, 2, 3, 4\}$
- Constraints $C = \{X < Y, Y < Z, Z \leq 2\}$

Remarks

- The constraint set is interpreted as the conjunction

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2.$$

- The domains are interpreted as the constraint

$$X \in D(X) \text{ and } Y \in D(Y) \text{ and } Z \in D(Z).$$

A Simple Example CSP

- Variables $\mathcal{X} = \{X, Y, Z\}$
- Domains $D(X) = D(Y) = D(Z) = \{1, 2, 3, 4\}$
- Constraints $C = \{X < Y, Y < Z, Z \leq 2\}$

Remarks

- The constraint set is interpreted as the conjunction

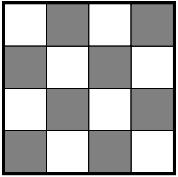
$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2.$$

- The domains are interpreted as the constraint

$$X \in D(X) \text{ and } Y \in D(Y) \text{ and } Z \in D(Z).$$

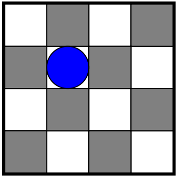
The N-Queens Problem

4-Queens: place 4 queens on 4×4 board without attacks

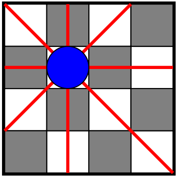


The N-Queens Problem

4-Queens: place 4 queens on 4×4 board without attacks

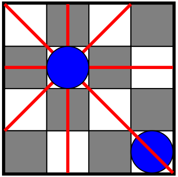


4-Queens: place 4 queens on 4×4 board without attacks

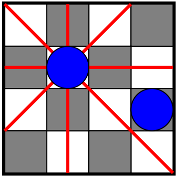


The N-Queens Problem

4-Queens: place 4 queens on 4×4 board without attacks



4-Queens: place 4 queens on 4×4 board without attacks



The N-Queens Problem

4-Queens: place 4 queens on 4×4 board without attacks

Model 4-Queens as CSP (Constraint Model)

- Variables
$$X_1, \ldots, X_4$$
$X_i = j$ means "queen in column i, row j"
- Domains $D(X_i) = \{1, \ldots, 4\}$ for $i = 1..4$
- Constraints (for different columns i and i')

no horizontal attack	$(X_i \neq X_{i'})$
no attack in first diagonal	$(i - X_i \neq i' - X_{i'})$
no attack in second diagonal	$(i + X_i \neq i' + X_{i'})$

The N-Queens Problem

4-Queens: place 4 queens on 4×4 board without attacks

Model 4-Queens as CSP (Constraint Model)

- Variables

$$X_1, \ldots, X_4$$

$X_i = j$ means "queen in column i, row j"

- Domains $D(X_i) = \{1, \ldots, 4\}$ for $i = 1..4$

- Constraints (for different columns i and i')

no horizontal attack	$(X_i \neq X_{i'})$
no attack in first diagonal	$(i - X_i \neq i' - X_{i'})$
no attack in second diagonal	$(i + X_i \neq i' + X_{i'})$

The N-Queens Problem

4-Queens: place 4 queens on 4×4 board without attacks

Model 4-Queens as CSP (Constraint Model)

- Variables
$$X_1, \ldots, X_4$$
$X_i = j$ means "queen in column i, row j"

- Domains $D(X_i) = \{1, \ldots, 4\}$ for $i = 1..4$

- Constraints (for different columns i and i')

  no horizontal attack $(X_i \neq X_{i'})$

  no attack in first diagonal $(i - X_i \neq i' - X_{i'})$

  no attack in second diagonal $(i + X_i \neq i' + X_{i'})$

The N-Queens Problem

4-Queens: place 4 queens on 4×4 board without attacks

Model 4-Queens as CSP (Constraint Model)

- Variables
$$X_1, \ldots, X_4$$
$$X_i = j \text{ means "queen in column i, row j"}$$

- Domains
$$D(X_i) = \{1, \ldots, 4\} \text{ for } i = 1..4$$

- Constraints (for different columns i and i')

no horizontal attack $(X_i \neq X_{i'})$

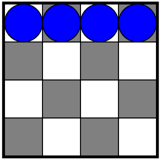
no attack in first diagonal $(i - X_i \neq i' - X_{i'})$

no attack in second diagonal $(i + X_i \neq i' + X_{i'})$

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 1$$

inconsistent!

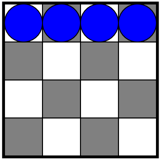
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 1$$

inconsistent!

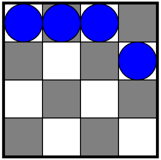
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 2$$

inconsistent!

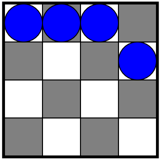
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 2$$

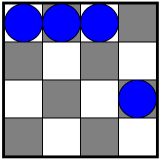
inconsistent!

What's wrong with GT?

- Redundancy
- Inconsistency local!

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 3$$

inconsistent!

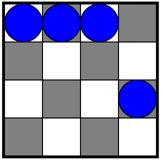
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 3$$

inconsistent!

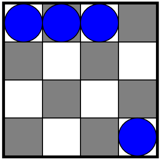
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 4$$

inconsistent!

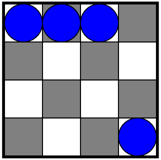
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 1, X_4 = 4$$

inconsistent!

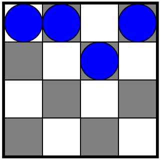
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 2, X_4 = 1$$

inconsistent!

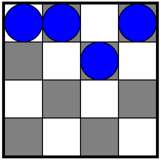
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 2, X_4 = 1$$

inconsistent! ...it's getting boring.

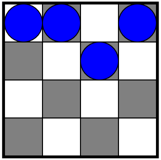
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 2, X_4 = 1$$

inconsistent!

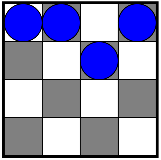
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



$$X_1 = 1, X_2 = 1, X_3 = 2, X_4 = 1$$

inconsistent!

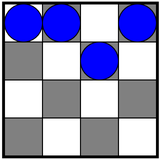
What's wrong with GT?

- Redundancy
- Inconsistency local!

Solving the CSP

Generate and Test

generate assignments and test each



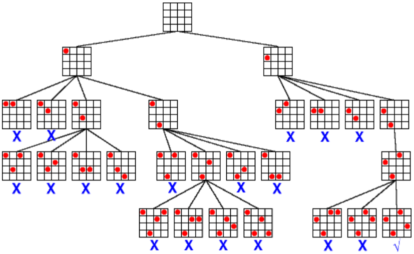
$$X_1 = 1, X_2 = 1, X_3 = 2, X_4 = 1$$

inconsistent!

What's wrong with GT?

- Redundancy
- Inconsistency local!

Backtracking

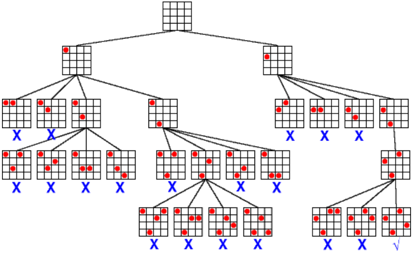


Problems

- Thrashing
- Redundancy
- Late Detection of Inconsistency

Overcoming GT's weakness

Backtracking

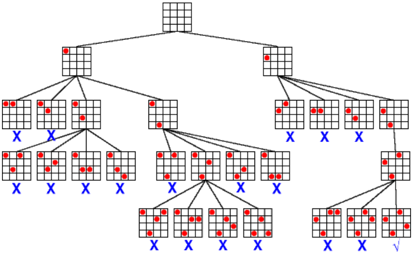


Problems

- Thrashing
- Redundancy
- Late Detection of Inconsistency

Overcoming GT's weakness

Backtracking

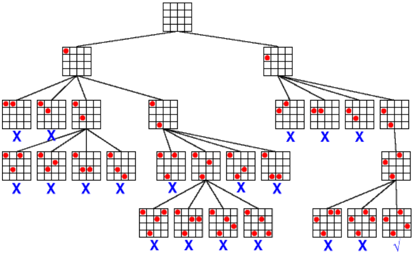


Problems

- Thrashing
- Redundancy
- Late Detection of Inconsistency

Overcoming GT’s weakness

Backtracking

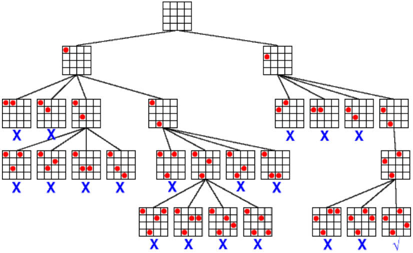


Problems

- Thrashing
- Redundancy
- Late Detection of Inconsistency

Overcoming GT's weakness

Backtracking



Problems

- Thrashing
- Redundancy
- Late Detection of Inconsistency

CP's Answer

Consistency Techniques

- detect inconsistency much earlier
- avoid redundancy and thrashing of BT

Definition

A consistency method transforms a CSP into an equivalent, consistent CSP.

How we will use it

Alternate consistency transformation and enumeration

CP's Answer

Consistency Techniques

- detect inconsistency much earlier
- avoid redundancy and thrashing of BT

Definition

A consistency method transforms a CSP into an equivalent, consistent CSP.

How we will use it

Alternate consistency transformation and enumeration

CP's Answer

Consistency Techniques

- detect inconsistency much earlier
- avoid redundancy and thrashing of BT

Definition

A consistency method transforms a CSP into an equivalent, consistent CSP.

How we will use it

Alternate consistency transformation and enumeration

Node and Arc Consistency

- Idea: Find equivalent, consistent CSP by removing values from the domains
- Examine one (basic) constraint at a time
- Node consistency: unary constraints $c(X)$
  remove values from $D(X)$ that falsify c
- Arc consistency: binary constraints $c(X, Y)$
  remove from $D(X)$ values that have no support in $D(Y)$ such that c is satisfied and vice versa

Node and Arc Consistency

- Idea: Find equivalent, consistent CSP by removing values from the domains
- Examine one (basic) constraint at a time
- Node consistency: unary constraints $c(X)$
  remove values from $D(X)$ that falsify c
- Arc consistency: binary constraints $c(X, Y)$
  remove from $D(X)$ values that have no support in $D(Y)$ such that c is satisfied and vice versa

Node and Arc Consistency

- Idea: Find equivalent, consistent CSP by removing values from the domains
- Examine one (basic) constraint at a time
- Node consistency: unary constraints $c(X)$
  remove values from $D(X)$ that falsify c
- Arc consistency: binary constraints $c(X, Y)$
  remove from $D(X)$ values that have no support in $D(Y)$ such that c is satisfied and vice versa

Node and Arc Consistency

- Idea: Find equivalent, consistent CSP by removing values from the domains
- Examine one (basic) constraint at a time
- Node consistency: unary constraints $c(X)$
  remove values from $D(X)$ that falsify c
- Arc consistency: binary constraints $c(X, Y)$
  remove from $D(X)$ values that have no support in $D(Y)$ such that c is satisfied and vice versa

Node Consistency

Definition

A unary constraint $c(X)$ is node consistent with domain D if $X = d$ satisfies $c(X)$ for each $d \in D(X)$.

Definition

A CSP $(\mathcal{X}, D, C)$ is node consistent, iff each of the unary constraints in C is node consistent with D.

Node Consistency Example

Our example CSP is not node consistent (see Z)

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = D(Z) = \{1, 2, 3, 4\}$$

Node consistent, equivalent CSP

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = \{1, 2, 3, 4\}, D(Z) = \{1, 2\}$$

Remark

- The 4-Queens CSP was node consistent, why?
- Computing node consistency is easy. Just look once at each unary constraint and remove inconsistent domain values.

Node Consistency Example

Our example CSP is not node consistent (see Z)

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = D(Z) = \{1, 2, 3, 4\}$$

Node consistent, equivalent CSP

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = \{1, 2, 3, 4\}, D(Z) = \{1, 2\}$$

Remark

- The 4-Queens CSP was node consistent, why?
- Computing node consistency is easy. Just look once at each unary constraint and remove inconsistent domain values.

Arc Consistency

Definition

A binary constraint $c(X, Y)$ is arc consistent with domain D if

- for each $d_X \in D(X)$ there is a $d_Y \in D(Y)$ s.t. $c(d_X, d_Y)$
- vice versa (for each $d_Y \in D(Y)$ there is a $d_X \in D(X)$ s.t. $c(d_X, d_Y)$)

Definition

A CSP $(\mathcal{X}, D, C)$ is arc consistent, iff each of the binary constraints in C is arc consistent with D.

Arc Consistency Example

The following CSP is node consistent but not arc consistent

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = \{1,2,3,4\}, D(Z) = \{1,2\}$$

For example $4 \in D(Y)$ and $Y < Z$
Arc consistent, equivalent CSP

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = D(Z) = \{\}$$

Remark
Our 4-Queens CSP is arc consistent.

Arc Consistency Example

The following CSP is node consistent but not arc consistent

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = \{1, 2, 3, 4\}, D(Z) = \{1, 2\}$$

For example $4 \in D(Y)$ and $Y < Z$
Arc consistent, equivalent CSP

$$X < Y \text{ and } Y < Z \text{ and } Z \leq 2$$
$$D(X) = D(Y) = D(Z) = \{\}$$

Remark

Our 4-Queens CSP is arc consistent.

Computing Arc Consistency

```
procedure REVISE(c, X, Y, D)
  D(X) := { d_X ∈ D(X) such that there exists d_Y ∈ D(Y)
                                  where c(d_X, d_Y) is satisfied }
endproc

do
  D' := D
  foreach binary constraint c ∈ C do
    let X, Y denote the variables of c
    REVISE(c, X, Y, D)
    REVISE(c, Y, X, D)
  done
until D = D'
```

Remark

This algorithm is called AC-1, usually one uses improved variants of this algorithm (e.g. AC-3).

Computing Arc Consistency

```
procedure REVISE(c, X, Y, D)
  D(X) := { d_X ∈ D(X) such that there exists d_Y ∈ D(Y)
                           where c(d_X, d_Y) is satisfied }
endproc

do
  D' := D
  foreach binary constraint c ∈ C do
    let X, Y denote the variables of c
    REVISE(c, X, Y, D)
    REVISE(c, Y, X, D)
  done
until D = D'
```

Remark

This algorithm is called AC-1, usually one uses improved variants of this algorithm (e.g. AC-3).

Computing Arc Consistency

```
procedure REVISE(c, X, Y, D)
  D(X) := { d_X ∈ D(X) such that there exists d_Y ∈ D(Y)
                                  where c(d_X, d_Y) is satisfied }
endproc

do
  D′ := D
  foreach binary constraint c ∈ C do
    let X, Y denote the variables of c
    REVISE(c, X, Y, D)
    REVISE(c, Y, X, D)
  done
until D = D′
```

Remark

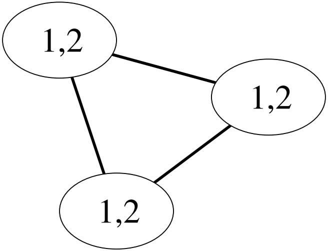
This algorithm is called AC-1, usually one uses improved variants of this algorithm (e.g. AC-3).

Avoiding Redundant Work: AC-3

```
Q :=empty queue
foreach binary constraint c ∈ C do
  push Q, (c, X, Y)
  push Q, (c, Y, X)
done

while Q ≠empty queue do
  (c,X,Y) := pop Q
  D':=D(X)
  REVISE(c, X, Y, D)
  if D(X) ≠ D' then
    for c' ∈ C and Z ∈ 𝒳 where c'(X, Z) or c'(Z, X) do
      push Q, (c', Z, X)
    done
  endif
done
```

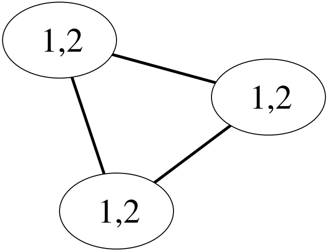
Node/Arc vs. Global Consistency



$$\mathcal{X} = \{X, Y, Z\}$$
$$D(X) = D(Y) = D(Z) = \{1, 2\}$$
$$C = \{X \neq Y, Y \neq Z, Z \neq X\}$$

- The CSP is node and arc consistent
- The CSP is globally inconsistent

Node/Arc vs. Global Consistency



$$\mathcal{X} = \{X, Y, Z\}$$
$$D(X) = D(Y) = D(Z) = \{1, 2\}$$
$$C = \{X \neq Y, Y \neq Z, Z \neq X\}$$

- The CSP is node and arc consistent
- The CSP is globally inconsistent

Consistency Methods: Summary

- Computing local consistency = constraint propagation
  - Node consistency
  - Arc consistency
  - (Hyper-arc consistency)
  - (Bounds consistency)
- Propagation is incomplete
- Solving a CSP requires search
  Combine backtracking and propagation

Complexity

- Local consistency: efficient
- CSP solving/global consistency: NP-complete

Consistency Methods: Summary

- Computing local consistency = constraint propagation
  - Node consistency
  - Arc consistency
  - (Hyper-arc consistency)
  - (Bounds consistency)
- Propagation is incomplete
- Solving a CSP requires search
  Combine backtracking and propagation

Complexity

- Local consistency: efficient
- CSP solving/global consistency: NP-complete

Consistency Methods: Summary

- Computing local consistency = constraint propagation
  - Node consistency
  - Arc consistency
  - (Hyper-arc consistency)
  - (Bounds consistency)
- Propagation is incomplete
- Solving a CSP requires search
  Combine backtracking and propagation

Complexity

- Local consistency: efficient
- CSP solving/global consistency: NP-complete

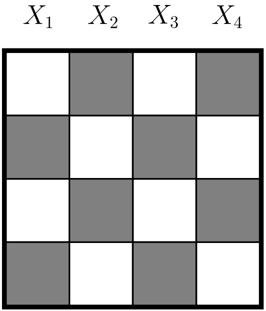
Consistency Methods: Summary

- Computing local consistency = constraint propagation
  - Node consistency
  - Arc consistency
  - (Hyper-arc consistency)
  - (Bounds consistency)
- Propagation is incomplete
- Solving a CSP requires search
  Combine backtracking and propagation

Complexity

- Local consistency: efficient
- CSP solving/global consistency: NP-complete

Solving 4-Queens (with Constraint Propagation)

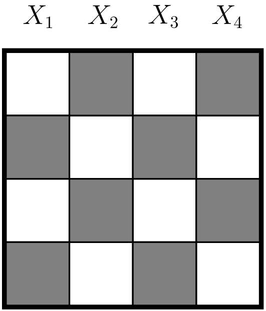


$$X_1, \ldots, X_4$$

$$D(X_i) = \{1, \ldots, 4\} \text{ for } i = 1..4$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 1$

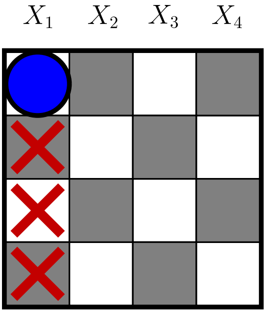


$$X_1, \ldots, X_4$$

$$D(X_i) = \{1, \ldots, 4\} \text{ for } i = 1..4$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 1$

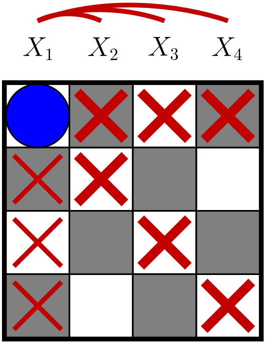


$$X_1, \ldots, X_4$$

$$D(X_1) = \{1\}, D(X_i) = \{1, \ldots, 4\} \text{ for } i = 2..4$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 1$

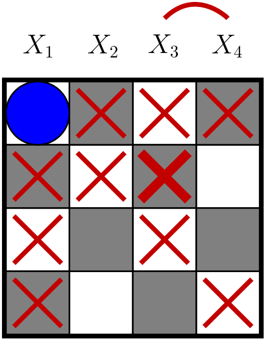


$$X_1, \ldots, X_4$$

$$D(X_1) = \{1\}, D(X_2) = \{3, 4\}, D(X_3) = \{2, 4\}, D(X_4) = \{2, 3\}$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 1$

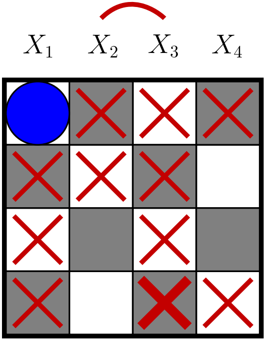


$$X_1, \ldots, X_4$$

$$D(X_1) = \{1\}, D(X_2) = \{3, 4\}, D(X_3) = \{4\}, D(X_4) = \{2, 3\}$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 1$

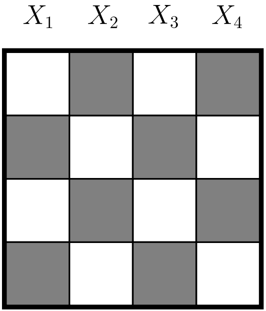


$$X_1, \ldots, X_4$$

$$D(X_1) = \{1\}, D(X_2) = \{3, 4\}, D(X_3) = \{\}, D(X_4) = \{2, 3\}$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 2$

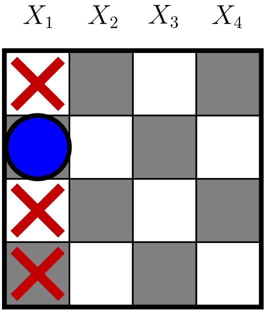


$$X_1, \ldots, X_4$$

$$D(X_i) = \{1, \ldots, 4\} \text{ for } i = 1..4$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 2$

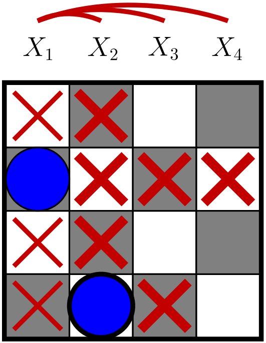


$$X_1, \ldots, X_4$$

$$D(X_1) = \{2\}, D(X_i) = \{1, \ldots, 4\} \text{ for } i = 2..4$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 2$

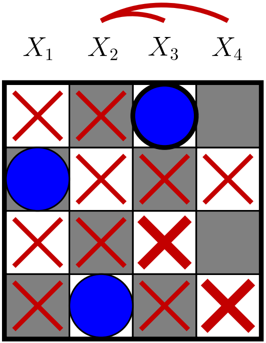


$$X_1, \ldots, X_4$$

$$D(X_1) = \{2\}, D(X_2) = \{4\}, D(X_3) = \{1, 3\}, D(X_4) = \{1, 3, 4\}$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 2$

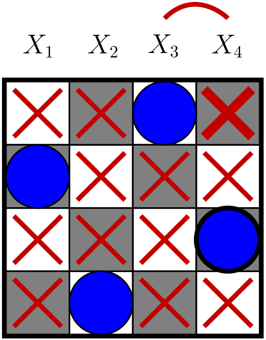


$$X_1, \ldots, X_4$$

$$D(X_1) = \{2\}, D(X_2) = \{4\}, D(X_3) = \{1\}, D(X_4) = \{3, 4\}$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Solving 4-Queens, $X_1 = 2$



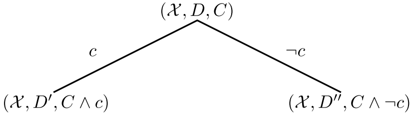
$$X_1, \ldots, X_4$$

$$D(X_1) = \{2\}, D(X_2) = \{4\}, D(X_3) = \{1\}, D(X_4) = \{3\}$$

$$X_i \neq X_{i'}, i - X_i \neq i' - X_{i'}, i + X_i \neq i' + X_{i'}$$

Constraint Search

- Combine Enumeration (backtracking) with propagation
- In general: enumeration by binary splits



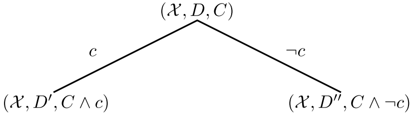
- Usually, we insert constraints of the form

$$X \diamond V, \qquad \diamond \in \{=, \leq, \geq, \ldots\}$$

- Variable and value selection important!
  - for size of search tree
  - not for completeness/correctness

Constraint Search

- Combine Enumeration (backtracking) with propagation
- In general: enumeration by binary splits



- Usually, we insert constraints of the form

$$X \diamond V, \qquad \diamond \in \{=, \leq, \geq, \dots\}$$

- Variable and value selection important!
  - for size of search tree
  - not for completeness/correctness

Constraint Search

- Combine Enumeration (backtracking) with propagation
- In general: enumeration by binary splits

$$\begin{tikzcd} & (\mathcal{X}, D, C) \arrow[dl, no head, "c"'] \arrow[dr, no head, "\neg c"] & \\ (\mathcal{X}, D', C \wedge c) & & (\mathcal{X}, D'', C \wedge \neg c) \end{tikzcd}$$

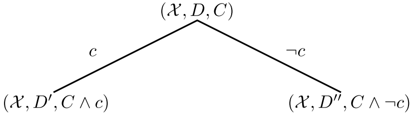
- Usually, we insert constraints of the form

$$X \diamond V, \qquad \diamond \in \{=, \leq, \geq, \dots\}$$

- Variable and value selection important!
  - for size of search tree
  - not for completeness/correctness

Constraint Search

- Combine Enumeration (backtracking) with propagation
- In general: enumeration by binary splits

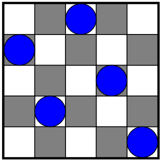


- Usually, we insert constraints of the form

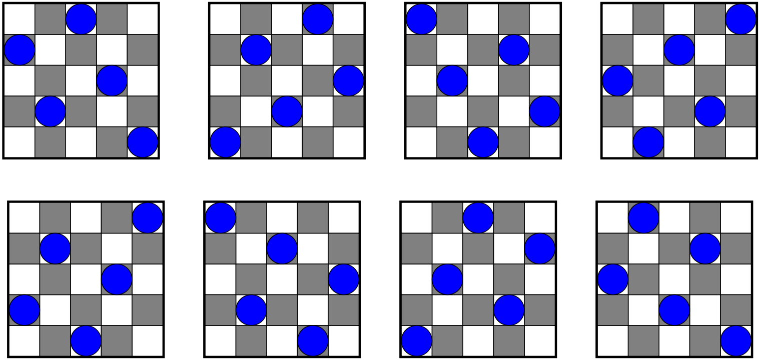
$$X \diamond V, \qquad \diamond \in \{=, \leq, \geq, \dots\}$$

- Variable and value selection important!
  - for size of search tree
  - not for completeness/correctness

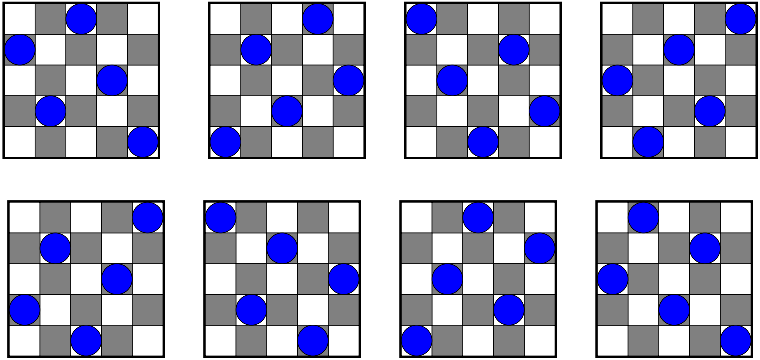
Symmetry



Symmetry

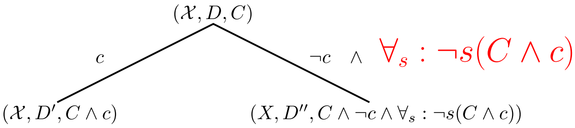


Symmetry



A symmetry is a (bijective) function on solutions.
This implies a symmetry function on constraints.

Symmetry Breaking Search



- Each right branch: forbid symmetries of the left branch
- By inserting a symmetric constraints for each symmetry

Constraint Optimization

Definition

A Constraint Optimization Problem (COP) is a CSP together with an objective function f on solutions.
A solution of the COP is a solution of the CSP that maximizes/minimizes f.

Solving by Branch & Bound Search
Idea of B&B:

- Backtrack & Propagate as for solving the CSP
- Whenever a solution s is found, add constraint "next solutions must be better than $f(s)$".

Constraint Optimization Example: Photo Problem

Alice,Bob,Carol, and Dave want to align for a photo

For example: $\boxed{\text{Alice, Carol, Dave, Bob}}$

However, they have preferences:

- Alice wants to stand next to Dave
- Bob wants to stand next to Dave and Carol
- Carol wants to stand next to Alice

Satisfy as many preferences as possible by constraint optimization.

Constraint Optimization Example: Photo Problem

Alice,Bob,Carol, and Dave want to align for a photo

For example: Alice, Carol, Dave, Bob

However, they have preferences:

- Alice wants to stand next to Dave
- Bob wants to stand next to Dave and Carol
- Carol wants to stand next to Alice

Satisfy as many preferences as possible by constraint optimization.