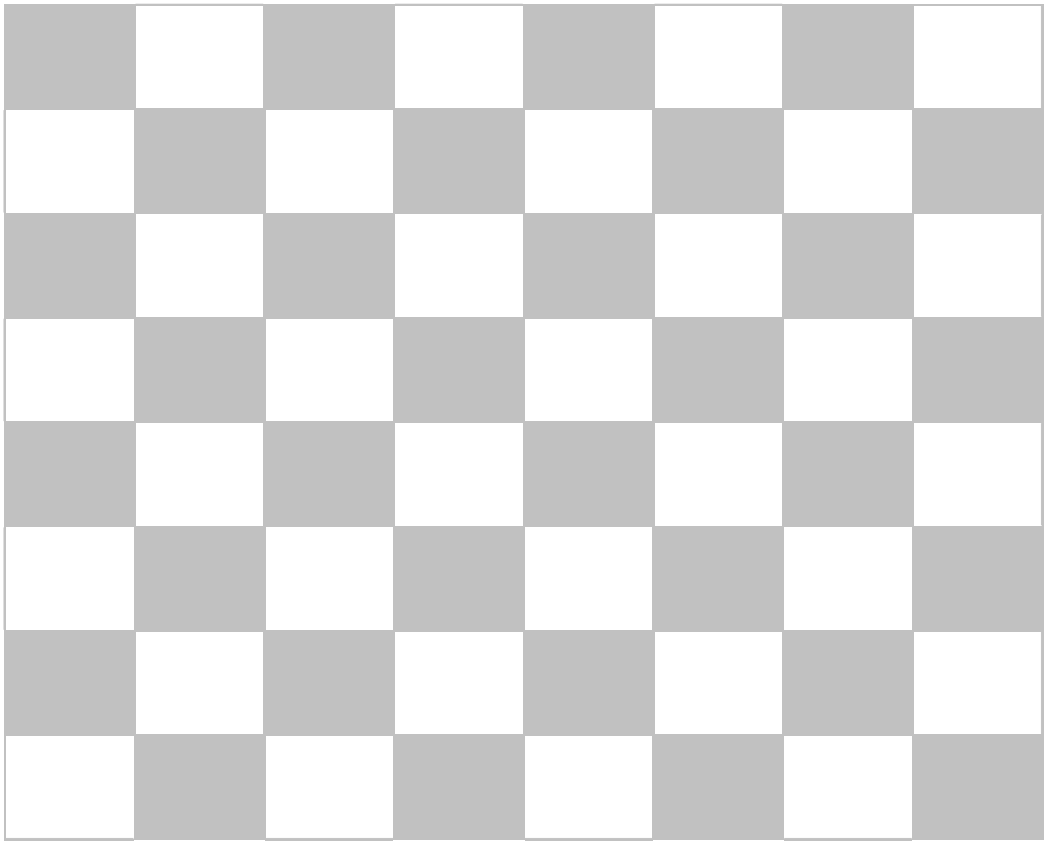
Stochastic greedy local search

Chapter 7

Example: 8-queen problem



Main elements

- Choose a full assignment and iteratively improve it towards a solution
- Requires a cost function: number of unsatisfied constraints or clasuses. Neural networks uses energy minimization
- Drawback: local minimas
- Remedy: introduce a random element
- Cannot decide inconsistency

Algorithm Stochastic Local search (SLS)

Procedure SLS
Input: A constraint network $\mathcal{R} = (X, D, C)$, number of tries MAX_TRIES. A cost function.
Output: A solution iff the problem is consistent, "false" otherwise.

1. **for** i=1 to MAX_TRIES
   - **initialization: let** $\bar{a} = (a_1, ..., a_n)$ be a random initial assignment to all variables.
   - **repeat**
     (a) **if** $\bar{a}$ is consistent, return $\bar{a}$ as a solution.
     (b) **else** let $Y = \{< x_i, a'_i >\}$ be the set of variable-value pairs that when x_i is assigned a'_i, give a maximum improvement in the cost of the assignment; pick a pair $< x_i, a'_i > \in Y$,
     $\bar{a} \leftarrow (a_1, ..., a_{i-1}, a'_i, a_{i+1}, ..., a_n)$ (just flip a_i to a'_i).
   - **until** the current assignment cannot be improved.
2. **endfor**
3. return **false**

Example: CNF

Example 7.1 Consider the formula $\varphi = \{(\neg C)(\neg A \vee \neg B \vee C)(\neg A \vee D \vee E)(\neg B \vee \neg C)\}$. Assume that in the initial assignment all variables are assigned the value "1". This assignment violates two clauses, the first and the last, so the cost is 2. Next we see that flipping A, E or D will not remove any inconsistency. Flipping C to "0" will satisfy the two violated clauses but will violate the clause $(\neg A \vee \neg B \vee C)$, yielding a cost of 1. Flipping B to $\neg B$ will remove one inconsistency and has a cost of 1 as well. If we flip C to $\neg C$, and subsequently flipping $B$to $\neg B$ yields a cost of 0 – and a solution. □

- Example:
- z divides y,x,t
- z = {2,3,5}, x,y = {2,3,4}, t={2,5,6}

Heuristics for improving local search

- **Plateau search:** at local minima continue search sideways.
- **Constraint weighting:** use weighted cost function
  - The cost C_i is 1 if no violation. At local minima increase the weights of violating constraints.

$$F(\bar{a}) = \sum w_i C_i(\bar{a})$$

- **Tabu search:**
  - prevent backwords moves by keeping list of asigned variable-values. Tie-breaking rule may be conditioned on historic information: select the value that was fliped least recently
- **Automating Max-flips:**
  - Based on experimenting with a class of problems
  - Given a progress in the cost function, allow the same number of flips used up to current progress.

Random walk strategies

- Combine random walk with greediness
  - At each step:
    - choose randomly an unsatisfied clause.
    - with probability p flip a random variable in the clause, with (1-p) do a greedy step minimizing the breakout value: the number of new constraints that are unsatisfied

Figure 7.2: Algorithm WalkSAT

Procedure WalkSAT
Input: A network $\mathcal{R} = (X, D, C)$, number of flips MAX_FLIPS, MAX_TRIES, probability p.
Output: True iff the problem is consistent, false otherwise.

1. For i= 1 to MAX_TRIES do
2. Compare best assignment with $\bar{a}$ and retain the best.
   (a) **start** with a random initial assignment $\bar{a}$.
   (b) **for** i=1 to MAX_FLIPS
      - **if** $\bar{a}$ is a solution, return **true** and $\bar{a}$.
      - **else,**
        i. **pick** a violated constraint C, randomly
        ii. **choose** with probability p a variable-value pair $< x, a' >$ for $x \in scope(C)$, or, with probability $1-p$, choose a variable-value pair $< x, a' >$ that minimizes the number of new constraints that break when the value of x is changed to a', (minus 1 if the current constraint is satisfied).
        iii. Change x's value to a'.
3. **endfor**
4. return **false** and the best current assignment.

Example of walkSAT:

start with assignment of true to all vars

Example 7.2 Following our earlier example 7.1.1, we will first select an unsatisfied clause, such as $(\neg B \vee \neg C)$, and then select a variable. If we try to minimize the number of additional constraints that would be broken, we will select B and flip its value. Subsequently, the only unsatisfied clause is $\neg C$ which is selected and flipped. □

Simulated Annealing (Kirkpatric, Gellat and Vecchi (1983))

- Pick a variable and a value and compute delta: the change in the cost function when the variable is flipped to the value.
- If change improves execute it,
- Otherwise it is executed with probablity e^(-delta/T), T is a temperature parameter.
- The algorithm will converge if T is reduced gradualy.

Properties of local search

- Guarantee to terminate at local minima
- Random walk on 2-sat is guaranteed to converge with robablity 1 after N^2 steps, when N is the number of variables.
- Proof:
  - there is ½ chance that a flip will reduce the distance to a satisfying assignment N/2 distance, by 1.
  - Random walk will cover this distance in N^2 steps
- Analysis breaks for 3-SAT
- Empirical evaluation shows good performance compared with complete algorithms

Hybrids of local search and Inference

- We can use exact hybrids of search+inference and replace search by SLS (Kask and Dechter 1996)
  - Good when cutset is small
- The effect of preprocessing by constraint propagation on SLS (Kask and Dechter 1995)
  - Great improvement on structured problems
  - Not so much on uniform problems

SLS and Local Consistency

- **Structured** (hierarchical 3SAT cluster structures) vs. **(uniform) random**.

Basic scheme :

- Apply preprocessing (resolution, path consistency)
- Run SLS
- Compare against SLS alone

SLS and Local Consistency

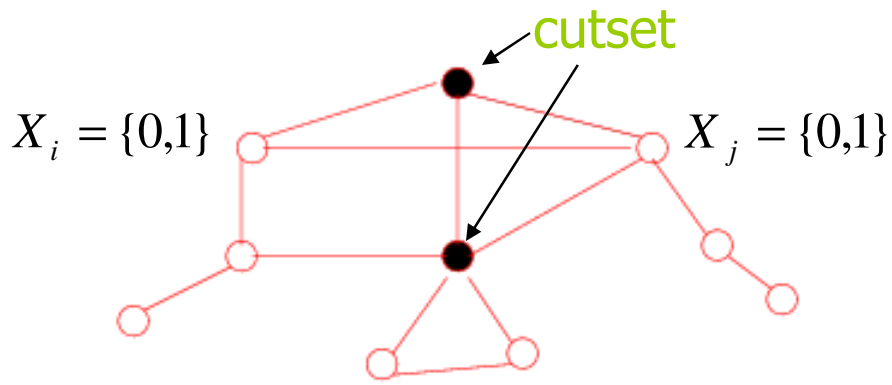
Summary:

- For structured problems, enforcing local consistency will improve SLS
- For uniform CSPs, enforcing local consistency is not cost effective: performance of SLS is improved, but not enough to compensate for the preprocessing cost.

SLS and Cutset Conditioning

Background:

- Cycle cutset technique improves backtracking by conditioning only on cutset variables.



SLS and Cutset Conditioning

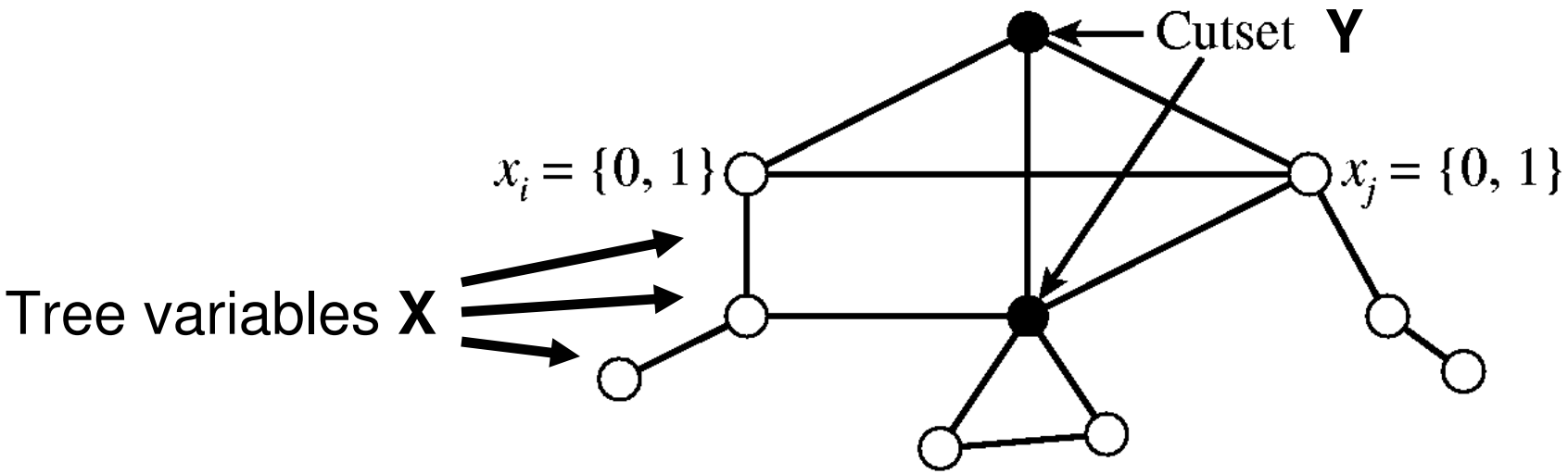
Background:

- Tree algorithm is tractable for trees.
- Networks with bounded width are tractable*.

Basic Scheme:

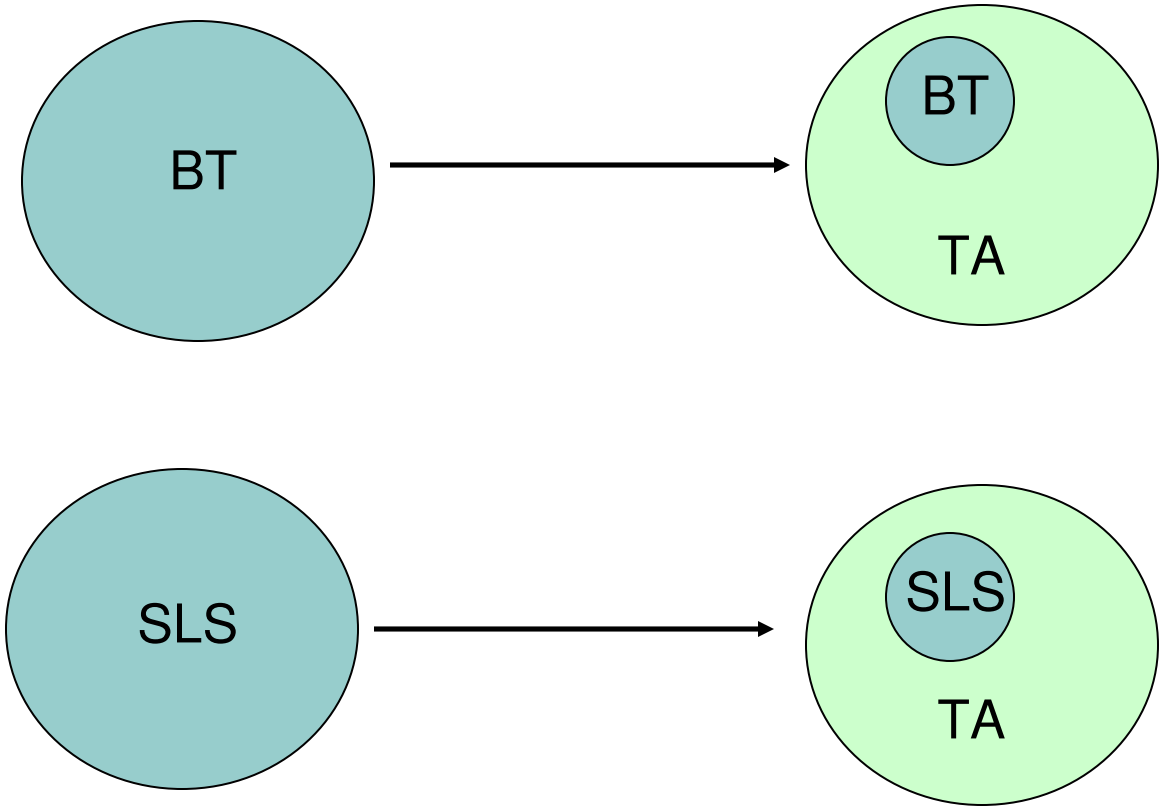
- Identify a cutset such that width is reduced to desired value.
- Use search with cutset conditioning.

Local search on Cycle-cutset



$$C_{min} = \min_{Y=y} C(y) = \min_{Y=y} \min_{X=x} \{C(X \mid Y = y)\}$$

Local search on Cycle-cutset



Tree Algorithm

Tree Algorithm : minimizing the cost of a tree-like subnetwork:
where $R_{z_i,z_j}(a_i, a_j)$ is the constraint between z_i and z_j and is either 0 if $(a_i, a_j) \in R_{z_i,z_j}$ or 1, otherwise.
Input: An arc consistent network $\mathcal{R} = (X, D, C)$. Variables X partitioned into cycle cutset Y and tree variables Z, $X = Z \cup Y$. An assignment $Y = \bar{y}$.
Output: An assignment $Z = \bar{z}$ that minimizes the number of violated constraints of the entire network when $Y = \bar{y}$.
Initialization: For any value $\bar{y}[i]$ of any cutset variable y_i, the cost $C_{y_i}(\bar{y}[i], \bar{y})$ is 0.

1. Going from leaves to root on the tree,
   (a) **for** every variable, z_i and any value $a_i \in D_{z_i}$, compute,

$$C_{z_i}(a_i, \bar{y}) = \sum_{\{z_j | z_j \ child \ of \ z_i\}} min_{a_j \in D_{z_j}} \left(C_{z_j}(a_j, \bar{y}) + R_{z_i,z_j}(a_i, a_j)\right)$$

   (b) **endfor**

Tree Algorithm (contd)

2. Compute, going from root to leaves, new assignment for every tree variable z_i:

   (a) **for** a tree variable z_i, let D_{z_i} be its consistent values with v_{p_i} the value assigned to its parent p_i, compute

   $$a_i \leftarrow arg\, min_{a_i \in D_{z_i}}(C_{z_i}(a_i, \bar{y}) + R_{z_i,p_i}(a_i,\ v_{p_i}))$$

   (b) **endfor**

3. **return** $(< z_1, a_1 >, ..., < z_k, a_k >)$.

GSAT with Cycle-Cutset (Kask and Dechter, 1996)

Input: a CSP, a partition of the variables into **cycle-cutset** and **tree variables**
Output: an assignment to all the variables

Within each try:
Generate a random initial asignment,
and then alternate between the two steps:

1. Run **Tree algorithm** (arc-consistency+assignment) on the problem with fixed values of cutset variables.
2. Run GSAT on the problem with fixed values of tree variables.

Theorem 7.1

Theorem 7.1 *The Tree Algorithm in Figure 7.4 is guaranteed to find an assignment that minimizes the number of violated constraints in every tree-like subnetwork, conditioned on the cutset values.*

Results: GSAT with Cycle-Cutset

(Kask and Dechter, 1996)

Binary CSP, 100 instances per line, 100 variables, 8 values, tightness 44/64						
number of constraints	average cutset size	Time Bound	GSAT solved	GSAT time per solvable	GSAT+CC solved	GSAT+CC time per solvable
125	11 %	29 sec	46	10 sec	90	2 sec
130	12 %	46 sec	29	16 sec	77	6 sec
135	14 %	65 sec	13	23 sec	52	10 sec

Binary CSP, 100 instances per line, 100 variables, 8 values, tightness 40/64						
number of constraints	average cutset size	Time Bound	GSAT solved	GSAT time per solvable	GSAT+CC solved	GSAT+CC time per solvable
160	20 %	52 sec	33	20 sec	90	7 sec
165	21 %	60 sec	13	30 sec	80	17 sec
170	22 %	70 sec	4	40 sec	54	22 sec

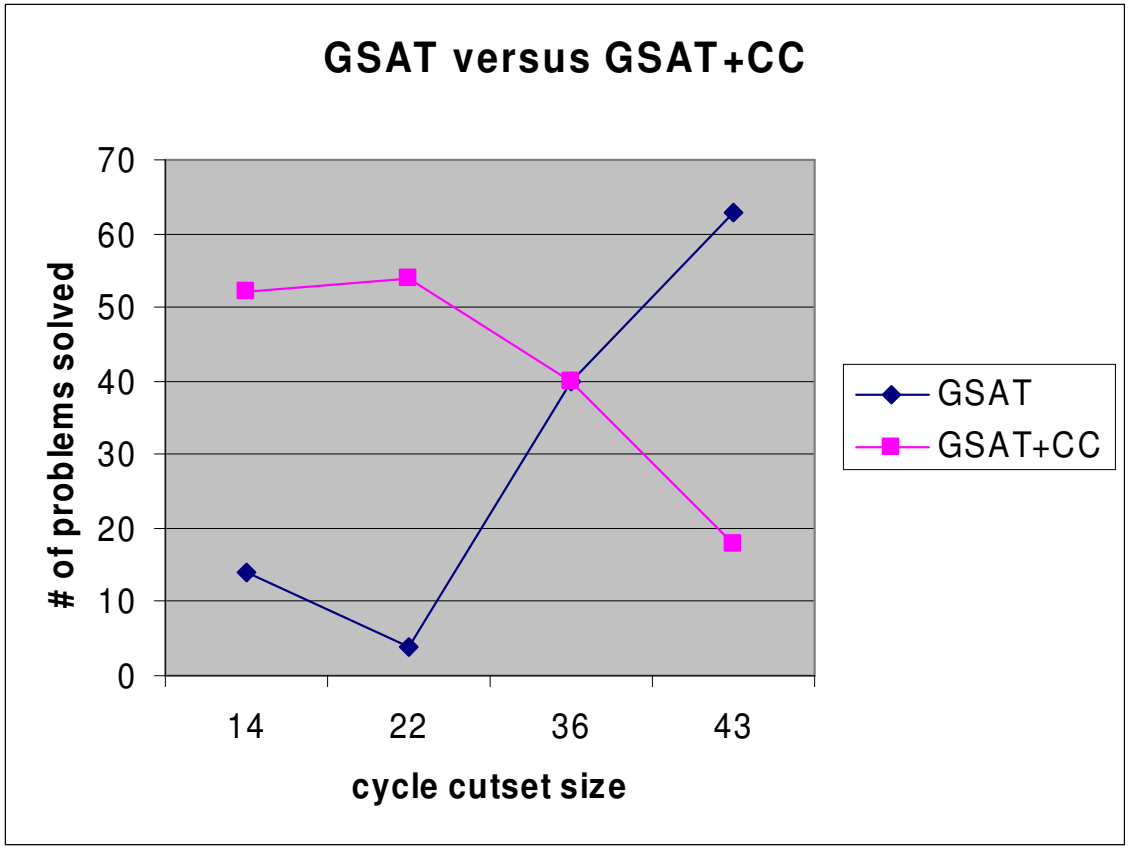
Binary CSP, 100 instances per line, 100 variables, 8 values, tightness 32/64						
number of constraints	average cutset size	Time Bound	GSAT solved	GSAT time per solvable	GSAT+CC solved	GSAT+CC time per solvable
235	34 %	52 sec	69	14 sec	66	18 sec
240	35 %	76 sec	57	22 sec	57	29 sec
245	36 %	113 sec	40	43 sec	40	43 sec

Binary CSP, 100 instances per line, 100 variables, 8 values, tightness 28/64						
number of constraints	average cutset size	Time Bound	GSAT solved	GSAT time per solvable	GSAT+CC solved	GSAT+CC time per solvable
290	41 %	55 sec	74	13 sec	30	25 sec
294	42 %	85 sec	80	25 sec	23	41 sec
300	43 %	162 sec	63	45 sec	19	82 sec

Table 1: GSAT vs. GSAT + CC

Results GSAT with Cycle-Cutset

(Kask and Dechter, 1996)



SLS and Cutset Conditioning

Summary:

- A new combined algorithm of SLS and inference based on cutset conditioning
- Empirical evaluation on random CSPs
- SLS combined with the tree algorithm is superior to pure SLS when the cutset is small