DM826 – Spring 2012

Modeling and Solving Constrained Optimization Problems

Lecture 10
Search

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Search

- Complete
  - backtracking
  - dynamic programming
- Incomplete
  - local search

Outline

1. Complete Search

2. Incomplete Search

Backtracking: Terminology

- backtracking: depth first search of a search tree
- branching strategy: method to extend a node in the tree
- node visited if generated by the algorithm
- constraint propagation prunes subtrees
- deadend: if the node does not lead to a solution
- thrashing repeated exploration of failing subtree differing only in assignments to variables irrelevant to the failure of the subtree.

Simple Backtracking

- at level $j \longleftarrow$ instantiation $I = \{x_1 = a_1, \ldots, x_j = a_j\}$
- branches: different choices for an unassigned variable: $I \cup \{x = a\}$
- branching constraints $\mathcal{P} = \{b_1, \ldots, b_j\}$, $b_i, 1 \leq i \leq j$
- $\mathcal{P} \cup \{b^1_{j+1}\}, \ldots, \mathcal{P} \cup \{b^k_{j+1}\}$ extension of a node by mutually exclusive branching constraints

Branching strategies

Assume a variable order and a value order (e.g., lexicographic):

A. Generic branching with unary constraints:

1. Enumeration, d-way

$$x = 1 \quad | \quad x = 2 \quad | \ldots$$

2. Binary choice points, 2-way

$$x = 1 \quad | \quad x \neq 1$$

3. Domain splitting

$$x \leq 3 \quad | \quad x > 3$$

$\rightsquigarrow$ d-way can be simulated by 2-way with no loss of efficiency. The contrary does not old.

$\rightsquigarrow$ 2-way seems theoretically more powerful than d-way

Branching strategies

B. Problem specific:

- Disjunctive scheduling
- Zykov's branching rule for graph coloring

Constraint propagation

- constraint propagation performed at each node: mechanism to avoid thrashing
- typically best to enforce domain based but with some exceptions (e.g., forward checking is best in SAT)
- nogood constraints added after deadend is encountered:
  - set of assignments and branching constraints that is not consistent with a solution
  - backtracking has already ruled out the subtree but inserting nogood constraints the hope is they contribute to propagate
  - e.g., $I = \{x_1 = 2, x_2 = 5, x_3 = 3, x_5 = 4\}$ and $x = 6$ deadend post $\neg\{x_1 = 2, x_2 = 5, x_3 = 3, x_5 = 4\}$

Backjumping

- standard backtracking: chronological backtracking
- non-chronological backtracking: retracts the closest branching constraint that bears responsibility.
  backjumping or intelligent backtracking:
  $\mathcal{P} = \{b_1, \ldots, b_j\}$
  $J(\mathcal{P}) \subseteq \mathcal{P}$ jumpback nogood for $\mathcal{P}$
  largest i $1 \leq i \leq j : b_i \in J(\mathcal{P})$
  jumpback and retracts b_i and all those posted after b_i
  and delete nogoods recorded after b_i

Restoration Service

What do we have at the nodes of the search tree?
A computational space:

1. Partial assignments of values to variables
2. Unassigned variables
3. Suspended propagators

How to restore when backtracking?

- Trailing Changes to nodes are recorded such that they can be undone later
- Copying A copy of a node is created before the node is changed
- Recomputation If needed, a node is recomputed from scratch

Variable-Value ordering

Possible goals

- Minimize the underlying search space
- Minimize expected depth of any branch
- Minimize expected number of branches
- Minimize size of search space explored by backtracking algorithm
  (intractable to find "best" variable)

Variable ordering

dynamic vs static

- it is optimal if it visits the fewest number of nodes in the search tree
- finding optimal ordering is hard

dynamic heuristics:

- based on domain size $\mathrm{rem}(x|\mathcal{P})$ remaining after propagation
- $\mathrm{dom} + \mathrm{deg}$ (# constraints that involve a variable still unassigned)
- $\frac{\mathrm{dom}}{\mathrm{wdeg}}$ weight incremented when a constraint is responsible for a deadend
- min regret
- structure guided var ordering:
  instantiate first variables that decompose the constraint graph
  graph separators: subset of vertices or edges that when removed
  separates the graph into disjoint subcomponents

Value ordering

- estimate number of solutions:
  counting solutions to a problem with tree structure can be done in polytime
  reduce the graph to a tree by dropping constraints

- if optimization constraints: reduced cost to rank values

Variants to best search

- Limited Discrepancy search

  Discrepancy: when the search does not follow the value ordering heuristic and does not take the left most branch out of a node.

  explored tree by iteratively increasing number of discrepancies, preferring discrepancies near the root
  (thus easier to recover from early mistakes)

  Ex: ith iteration: visit all leaf nodes up to i discrepancies
  $i = 0, 1, \ldots, k$ (if $k \geq n$ depth then alg is complete)

- Interleaved depth first search
  each subtree rooted at a branch is searched for a given time-slice using depth-first.
  If no solution found, search suspended, next branch active.
  Upon suspending in the last the first again becomes active.
  Similar idea in credit based.

Randomization in Search Tree

- Dynamical selection of solution components
  in construction or choice points in backtracking.

- Randomization of construction method or
  selection of choice points in backtracking
  while still maintaining the method complete
  $\rightsquigarrow$ *randomized systematic search*.

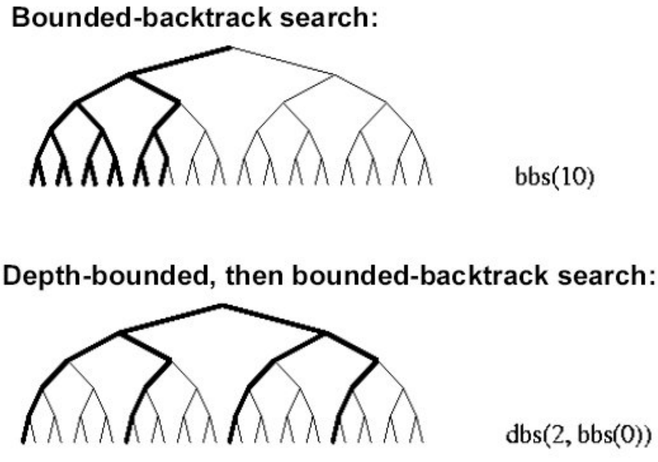
- Randomization can also be used in incomplete search

Outline

1. Complete Search

2. Incomplete Search

Incomplete Search

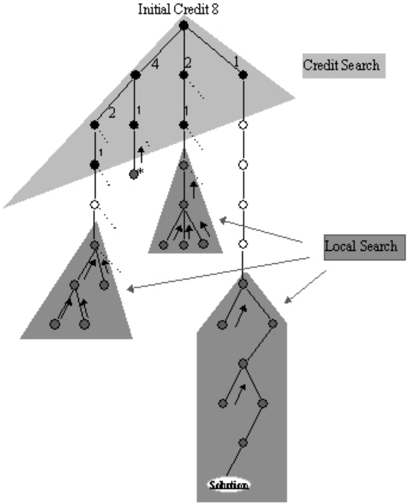


http://4c.ucc.ie/~hsimonis/visualization/techniques/partial_search/main.htm

Incomplete Search

Credit-based search

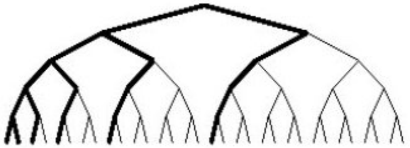
- Key idea: important decisions are at the top of the tree
- Credit = backtracking steps
- Credit distribution: one half at the best child the other divided among the other children.
- When credits run out follow deterministic best-search
- In addition: allow limited backtracking steps (eg, 5) at the bottom
- Control parameters: initial credit, distribution of credit among the children, amount of local backtracking at bottom.



Incomplete Search

Limited Discrepancy Search (LDS)

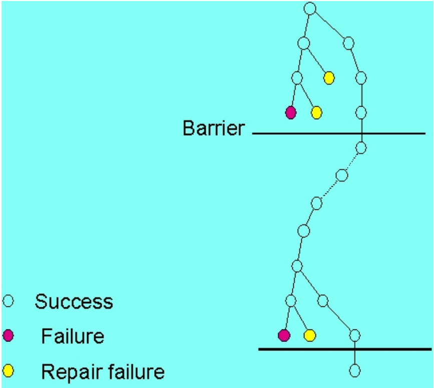
- Key observation that often the heuristic used in the search is nearly always correct with just a few exceptions.
- Explore the tree in increasing number of discrepancies, modifications from the heuristic choice.
- Eg: count one discrepancy if second best is chosen
  count two discrepancies either if third best is chosen or twice the second best is chosen
- Control parameter: the number of discrepancies



Incomplete Search

Barrier Search

- Extension of LDS
- Key idea: we may encounter several, independent problems in our heuristic choice. Each of these problems can be overcome locally with a limited amount of backtracking.
- At each barrier start LDS-based backtracking



Local Search for CSP [Hoos and Tsang, 2006]

- Uses a complete-state formulation
- Initial state: a value assigned to each variable (randomly)
- Changes the value of one variable at a time
- Evaluation of a state:
  number of constraints violated or variables to change (see soft constraints)
- Min-conflict heuristic [Minton et al., 1992]:
  - pick one variable involved in a constraint violation at random
  - assign to it the best value
- Run-time independent from problem size

References

Hoos H.H. and Tsang E. (2006). **Local Search Methods**, chap. 5. Elsevier.

Minton S., Johnston M., Philips A., and Laird P. (1992). **Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems**. *Artificial Intelligence*, 58(1-3), pp. 161–205.

Rossi F., van Beek P., and Walsh T. (eds.) (2006). **Handbook of Constraint Programming**. Elsevier.

Schulte C. and Carlsson M. (2006). **Finite domain constraint programming systems**. In Rossi et al. [2006].