Search Algorithms Continued

Leena Salmela

25th March 2004

Based on:

Krzysztof R. Apt: Principles of Constraint Programming

Pages 332-347

Outline

- Constraint propagation in the backtracking algorithm
- Searching for all solutions
- Finite constraint optimization algorithms
- Heuristics
- Search in the general search trees

Reminder: Backtracking algorithm

```
void back_track_prop(int j, domains D, boolean *success)
    while D[j] not empty and not *success
        choose d from D[j]
        D[j] = D[j] - d
        if consistent(inst, j, d)
            inst[j] = d
            *success = (j == n)
            if not *success
                failure = prop(j, D)
                if not failure
                    backtrack_prop(j+1, D, success)

boolean success = false
failure = prop(0, D)
if not failure
    backtrack_prop(1, D, &success)
```

Constraint propagation: Forward checking

- $inst[1...j]$ contains the values of already instantiated variables
- Revise the domain $D[k]$ of each future variable x_k:

  $$D[k] = \{d \in D[k] | \{(x_1, inst[1]), \ldots, (x_j, inst[j]), (x_k, d)\} \text{ is a consistent instantiation}\}$$

- The propagation procedure goes through all the future variables and revises their domains as presented above. If the domain of a variable becomes empty the procedure returns failure.
- Backtrack search with forward checking as propagation algorithm does search in the FORWARD CHECKING search tree.

Constraint propagation: Partial look ahead

- Propagation consists of forward checking and maintaining directional arc consistency
- The propagation procedure does two things:
  1. Run the propagation procedure for forward checking
  2. If the forward checking does not fail run the directional arc consistency algorithm for the future variables.
- Backtrack search with partial look ahead propagation does search in the PARTIAL LOOK AHEAD search tree.

Constraint propagation: Maintaining arc consistency

- Now propagation consists of forward checking and maintaining arc consistency
- Again we first run the forward checking procedure and then the arc consistency algorithm for the future variables
- Backtrack search with MAC propagation does search in the MAC search tree

Finding all solutions

- The second case of CSP problems: Instead of finding only one solution we want to find all solutions.
- The backtrack search is easily adopted:
  - When a solution is found it is printed.
  - The search is not terminated when a solution is found. Instead all values in the domain of a variable are tried.
- This is called backtrack-all algorithm

Constraint optimization problems

- We are given a CSP and a function $obj : Sol \mapsto \mathcal{R}$, where Sol is the set of solutions
- We want to find the solution d for which $obj(d)$ is maximal.
- Usually a heuristic function $h : \mathcal{P}(D_s) \times \cdots \times \mathcal{P}(D_n) \mapsto \mathcal{R} \cup \{\infty\}$ is also given. The following restrictions apply to h:
  1. Monotonicity: if $E_1 \subseteq E_2$ then $h(E_1) \leq h(E_2)$
  2. Bound: $obj(d_1, \ldots, d_n) \leq h(\{d_1\}, \ldots, \{d_n\})$
- The algorithms here assume that the CSP is finite and thus labeling can be used for splitting.

Branch and bound

- We modify the backtrack-all algorithm
- We now keep track of the best value of the function obj found so far and the corresponding solution
- The idea is that after instantiating a new variable we check using the heuristic function if better solutions can be found by completing this partial instantiation.
- If constraint propagation is used the value of the heuristic function is checked after the propagation.

Branch and bound: Cost constraint

- Suppose that the constraint $obj(x_1, \ldots, x_n) > bound$ is definable in the CSP language.
- When a new better solution is found the constraint $obj(x_1, \ldots, x_n) > bound$ can now be added to the set of constraints.
- This might help the constraint propagation algorithm.

Heuristics: Variable selection

- Which variable should be the next to be instantiated?
- Possible heuristics:
  - Variable with smallest domain: The search tree is likely to have less nodes.
  - Most constrained variable: The constraint propagation is likely to work better.
- For numerical constraints:
  - Variable with smallest difference between its domain bounds.

Heuristics: Value selection

- Which value should the variable be instantiated with?
- In constrained optimization problems:
  - The value which gives the highest value for the heuristic funtion
- Numeric domains:
  - Smallest value
  - Largest value
  - Middle value

Search in general search trees

- In previous algorithms we assumed that the splitting happens through labeling
- In the arbitrary search trees splitting might happen in some other way.

Branch and bound in general search trees

```
void branch_and_bound(searchtree children, CSP *solution,
                      double *bound)
    while children[P] not empty
        choose R from children[P] and remove R from children[P]
            if not failed(R)
                P = R
                if solved(P)
                    if obj(P) > bound
                        *bound = obj(P); *solution = P
                else
                    P = next(P)
                    if not failed(P) and h(P) > bound
                        branch_and_bound(children, solution, bound)

solution = NIL; bound = -inf; P = next(PInit)
if not failed(P)
    branch_and_bound(children, &solution, &bound)
```

Conclusion

- The backtrack algorithm can be instantiated with several different constraint propagation algorithms.
- The backtrack algorithm can also be used for finding all solutions.
- Using the branch and bound principle algorithms for constraint optimization problems can be developed.
- Heuristics can be useful in choosing the variable, which is next instantiated, and its value.
- The algorithms can also be adopted to other search trees than labeling trees.