Pages 332-347

Krzysztof R. Apt: Principles of Constraint Programming

Based on:

25th March 2004

Leena Salmela

Search Algorithms Continued
Search in the general search trees

Heuristics

Finite constraint optimization algorithms

Searching for all solutions

Constraint propagation in the backtracking algorithm

Outline

Search Algorithms Continued
Reminder: Backtracking Algorithm

```c
void back_track_prop(int j, domains D[], boolean* success)
{
    if not failure
    failure = prop(0, D)
    if not failure
    back_track_prop(j+1, D, success)
}

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

25th March 2004
Leena Salmela
Slide 3

Search Algorithms Continued
Backtrack search with forward checking as propagation algorithm does.

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.

Search in the FORWARD CHECKING search tree.

Constraints propagation: Forward checking

\{
(p,ands_{x})\mid [y]D \in p \} = [y]D

Revises the domain of each future variable. It contains the values of already instantiated variables.

is a consistent instantiation

Search Algorithms Continued
Constraint propagation: Partial lookahead

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 lookahead does search in the PARTIAL LOOK AHEAD search tree.

Search Algorithms Continued
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
We are given a CSP and a function $\mathcal{Q}$ for which $\mathcal{Q}(p)$ is maximal. Usually a heuristic function $h$ is used for splitting.

The algorithms here assume that the CSP is finite and thus labeling can be used for splitting.

1. Monotonicity: if $\mathcal{E} \subseteq \mathcal{F}$ then $\mathcal{H} \mathcal{E} \subseteq \mathcal{H} \mathcal{F}$
2. Bound: $\mathcal{Q}(\mathcal{F}) \geq (\mathcal{Q}(\mathcal{F}))_{\mathcal{E}}$

The following restrictions apply to $h$:

$$ \{\infty\} \cap \mathcal{R} \leftarrow (a^u \mathcal{D}) \times \cdots \times (a^g \mathcal{D}) \mathcal{D} : h = \text{maximal}. $$

We want to find the solution for which $\mathcal{Q}(p)$ is maximal.

**Constraint optimization problems**
We modify the backtrack-all algorithm.

Branch and bound

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.

We now keep track of the best value of the function of the solution found so far and the corresponding solution.
This might help the constraint propagation algorithm.

When a new better solution is found, the constraint
\[ \leq (u, x, \ldots) \]
\( \leq q \cdot y \)

CSP Language.

Suppose that the constraint is definable in the

Branch and Bound: Cost constraint

Search Algorithms Continued
Heuristics: Variable selection

- Variable with smallest difference between its domain bounds.
- Most constrained variable: The constraint propagation is likely to work better.
- Variable with smallest difference between its domain bounds for numerical constraints.
- Most constrained variable: The constraint propagation is likely to work better.
- Variable with smallest domain: The search tree is likely to have less nodes.

Possible heuristics:

- Variable should be the next to be instantiated.

Heuristics: Variable selection

Search Algorithms Continued
Search Algorithms

Heuristics: Value Selection

Which valueshould the variable be instantiated with?

In constrained optimization problems:

- The value which gives the highest value for the heuristic function
- Which value should the variable be instantiated with?

Numeric domains:

- Middle value
- Largest value
- Smallest value

- The value which gives the highest value for the heuristic function

25th March 2004
Leena Salmela
Slide 12
In the arbitrary search trees splitting might happen in some other way.

In previous algorithms we assumed that the splitting happens through labeling.

Search in general search trees.
Branch and Bound (General Search Trees)

```c
void branch_and_bound(searchtreechildren, CSP*solution, double*bound)
{
    if not failed(p)
    
    solution = NIL; bound = -inf; p = next(PInit)

    while children[p] not empty
    
    choose R from children[p] and remove R from children[p]
    
    if not failed(R) and h(R) < bound
    
    p = next(p)

    else

    bound = obj(R) * solution = p

    if not failed(p)
    
    branch_and_bound(children, solution, bound)
```

Branch and Bound in General Search Trees

Search Algorithms Continued
The backtrack algorithm can also be adapted to other search trees than labeling trees.

- The algorithms can also be adapted to other search trees than labeling trees.
- Heuristics can be useful in choosing the variable, which is next instantiated, and its value.
- Heuristics can be useful in choosing the variable, which is next instantiated, and its value.
- Heuristics can be useful in choosing the variable, which is next instantiated, and its value.
- Heuristics can be useful in choosing the variable, which is next instantiated, and its value.
- Heuristics can be useful in choosing the variable, which is next instantiated, and its value.
- Heuristics can be useful in choosing the variable, which is next instantiated, and its value.

**Conclusion**

The backtrack algorithm can also be adapted to other search trees than labeling trees.

- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.

- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.

- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.

- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.

- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.
- The backtrack algorithm can also be adapted to other search trees than labeling trees.