DM811

Heuristics for Combinatorial Optimization

Lecture 3

Construction Heuristics and Metaheuristics

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

Search Methods

- initial state: the empty assignment $\{\}$ in which all variables are unassigned
- successor function: a value can be assigned to any unassigned variable, provided that it does not conflict with previously assigned variables
- goal test: the current assignment is complete
- path cost: a constant cost

Types of problems:

- Assignment
- Sequencing
- Subset
- Routing
- ...

Complete Tree Search

Uninformed

- Breadth-first search
- Depth-first search
- Depth-limited search
- Iterative deepening depth-first search
- Bidirectional Search

Informed

- Informed search algorithm: exploit problem-specific knowledge
- Best-first search: node that "appears" to be the best selected for expansion based on an evaluation function $f(x)$
- Implemented through a priority queue of nodes in ascending order of f (See later discussion on A* search)

Complete Tree Search

Uninformed

Search Space

tree with branching factor at the top level nd
at the next level $(n-1)d$.
The tree has $n! \cdot d^n$ even if only d^n possible complete assignments.

Informed

- CSP is commutative in the order of application of any given set of action. (we reach same partial solution regardless of the order)
- Hence generate successors by considering possible assignments for only a single variable at each node in the search tree.
- look-ahead, best first, etc.

Dealing with Constraints

Backtracking search

depth-first search that chooses one variable at a time and backtracks when a variable has no legal values left to assign.

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment according to CONSTRAINTS[csp] then
      add {var = value} to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove {var = value} from assignment
  return failure
```


Backtrack Search

- No need to copy solutions all the times but rather extensions and undo extensions
- Since CSP is standard then the alg is also standard and can use general purpose algorithms for initial state, successor function and goal test.
- Backtracking is uninformed and complete. Other search algorithms may use information in form of heuristics.

Backtracking
General Concepts

Decisions in general purpose methods:

1) Which variable should we assign next, and in what order should its values be tried?

2) What are the implications of the current variable assignments for the other unassigned variables?

3) When a path fails – that is, a state is reached in which a variable has no legal values can the search avoid repeating this failure in subsequent paths?

Search (1) + Inference (2) + Backtracking (3) = Constraint Programming

In the general case, at point 1) we use heuristic rules.

If we do not backtrack (point 3) then we have a construction heuristic.

1) Which variable should we assign next, and in what order should its values be tried?

- Select-Initial-Unassigned-Variable
- Select-Unassigned-Variable
  - most constrained first = fail-first heuristic
    = Minimum remaining values (MRV) heuristic
    (tend to reduce the branching factor and to speed up pruning)
  - least constrained last

  Eg.: max degree, farthest, earliest due date, etc.
- Order-Domain-Values
  - greedy
  - least constraining value heuristic
    (leaves maximum flexibility for subsequent variable assignments)
  - maximal regret
    implements a kind of look ahead

2) What are the implications of the current variable assignments for the other unassigned variables?

Propagating information through constraints:

- Implicit in Select-Unassigned-Variable
- Forward checking (coupled with Minimum Remaining Values)
- Constraint propagation in CSP
  - arc consistency: force all (directed) arcs uv to be consistent:
    $\exists$ a value in $D(v)$: $\forall$ values in $D(u)$, otherwise detects inconsistency

    can be applied as preprocessing or as propagation step after each assignment (Maintaining Arc Consistency)

    Applied repeatedly

Propagation: An Example

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA*=*red*	(R)	G B	R G B	R G B	R G B	G B	R G B
After *Q*=*green*	(R)	B	(G)	R B	R G B	B	R G B
After *V*=*blue*	(R)	B	(G)	R	(B)		R G B

Figure 5.6 The progress of a map-coloring search with forward checking. $WA = red$ is assigned first; then forward checking deletes *red* from the domains of the neighboring variables NT and SA. After $Q = green$, *green* is deleted from the domains of NT, SA, and NSW. After $V = blue$, *blue* is deleted from the domains of NSW and SA, leaving SA with no legal values.

3) When a path fails – that is, a state is reached in which a variable has no legal values can the search avoid repeating this failure in subsequent paths?

Backtracking-Search

- chronological backtracking, the most recent decision point is revisited
- backjumping, backtracks to the most recent variable in the conflict set (set of previously assigned variables connected to X by constraints).

An Empirical Comparison

Problem	Backtracking	BT+MRV	Forward Checking	FC+MRV
USA	(> 1,000K)	(> 1,000K)	2K	60
n-Queens	(> 40,000K)	13,500K	(> 40,000K)	817K
Zebra	3,859K	1K	35K	0.5K
Random 1	415K	3K	26K	2K
Random 2	942K	27K	77K	15K

Median number of consistency checks

Dealing with Objectives

A* search

- The priority assigned to a node x is determined by the function

$$f(x) = g(x) + h(x)$$

$g(x)$: cost of the path so far
$h(x)$: heuristic estimate of the minimal cost to reach the goal from x.

- It is optimal if $h(x)$ is an
  - admissible heuristic: *never overestimates* the cost to reach the goal
  - consistent: $h(n) \leq c(n, a, n') + h(n')$

A* best-first search

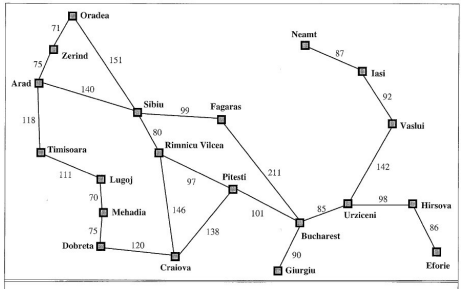


Figure 3.2 A simplified road map of part of Romania.

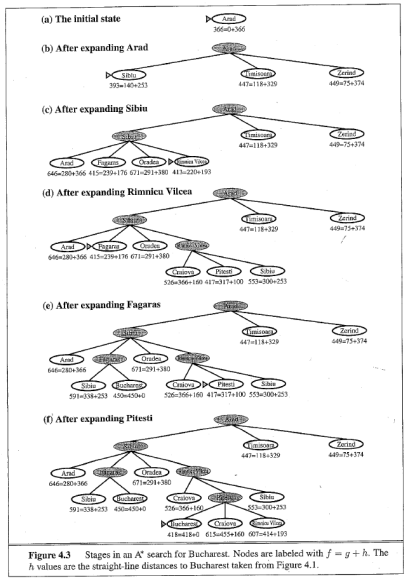


Figure 4.3 Stages in an A* search for Bucharest. Nodes are labeled with $f = g + h$. The h values are the straight-line distances to Bucharest taken from Figure 4.1.

A* search

Possible choices for admissible heuristic functions

- optimal solution to an easily solvable **relaxed problem**
- optimal solution to an easily solvable **subproblem**
- learning from experience by gathering statistics on state features
- preferred heuristics functions with higher values (provided they do not overestimate)
- if several heuristics available $h_1, h_2, \ldots, h_m$ and not clear which is the best then:

$$h(x) = \max\{h_1(x), \ldots, h_m(x)\}$$

A* search

Drawbacks

- Time complexity: In the worst case, the number of nodes expanded is exponential,
  (but it is polynomial when the heuristic function h meets the following condition:
  $$|h(x) - h^*(x)| \leq O(\log h^*(x))$$
  h^* is the optimal heuristic, the exact cost of getting from x to the goal.)
- Memory usage: In the worst case, it must remember an exponential number of nodes.
  Several variants: including iterative deepening A* (IDA*), memory-bounded A* (MA*) and simplified memory bounded A* (SMA*) and recursive best-first search (RBFS)

Outline

Incomplete Search

Complete search is often better suited when ...

- proofs of insolubility or optimality are required;
- time constraints are not critical;
- problem-specific knowledge can be exploited.

Incomplete search is the necessary choice when ...

- non linear constraints and non linear objective function;
- reasonably good solutions are required within a short time;
- problem-specific knowledge is rather limited.

Greedy algorithms

Greedy algorithms

- Strategy: always make the choice that is best at the moment
- They are not generally guaranteed to find globally optimal solutions (but sometimes they do: Minimum Spanning Tree, Single Source Shortest Path, etc.)

We will see problem sepcific examples

Outline

Metaheuristics

On backtracking framework (beyond best-first search)

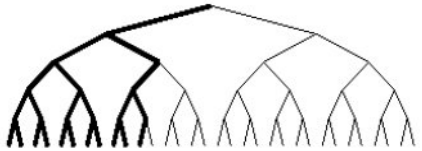
- Bounded backtrack
- Credit-based search
- Limited Discrepancy Search
- Barrier Search
- Randomization in Tree Search

Outside the exact framework (beyond greedy search)

- Rollout/Pilot Method
- Beam Search
- Iterated Greedy
- GRASP
- Adaptive Iterated Construction Search
- Multilevel Refinement

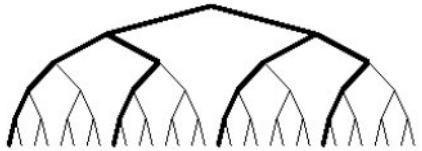
Bounded backtrack

Bounded-backtrack search:



bbs(10)

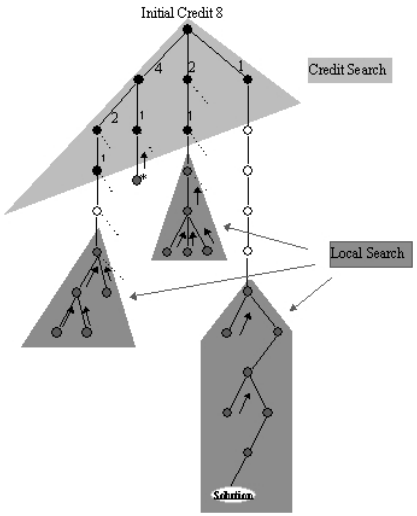
Depth-bounded, then bounded-backtrack search:



dbs(2, bbs(0))

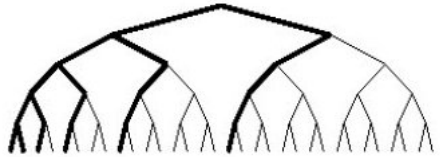
Credit-based search

- 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, the distribution of credit among the children, and the amount of local backtracking at the bottom.



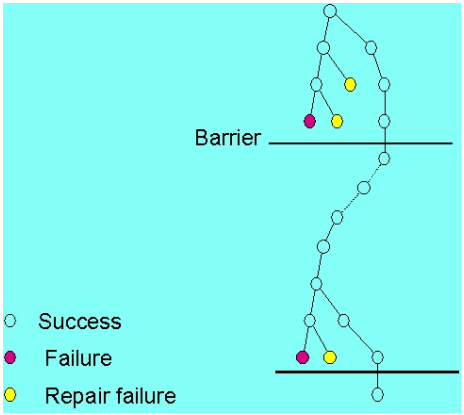
Limited Discrepancy Search

- 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



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



Randomization in Tree Search

The idea comes from complete search

- 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