Topic 5: Search[1]
(Version of 25th September 2017)

Pierre Flener

ASTRA Research Group
on Combinatorial Optimisation
Uppsala University
Sweden

Course 1DL441:
Combinatorial Optimisation using
Constraint Programming

[1] Based also on some material by Christian Schulte and Yves Deville

UPPSALA UNIVERSITET

Search = Branching + Exploration

Branching
Exploration

- Branching describes how to define the search tree.
- Exploration describes how to explore the search tree:
  - first solution
  - all solutions
  - best solution: via branch-and-bound
  - depth-first
  - breadth-first
  - multi-start
  - ...

UPPSALA UNIVERSITET

Outline

Branching

Exploration

1 Branching

2 Exploration

UPPSALA UNIVERSITET

Outline

Branching
Exploration

1 **Branching**

2 Exploration

Definition (Brancher)

A brancher b satisfies the following conditions, when $b(R, s) = \langle R_1, ..., R_n \rangle \wedge \forall i.\text{Propagate}(R \cup R_i, R_i, s) = \langle \_, s_i \rangle$:

- **Contraction:** $\forall i : s_i \precneqq s$. (Hence a finite search tree.)
- **No solutions lost or duplicated:** $\forall \sigma \in s : \exists! i : \sigma \in s_i$.

where propagator set R_i is called the i^{th} decision or guess.

Definition (Branch & propagate search tree)

Let $\langle V, U, P, b \rangle$ be a model extended with a brancher b. The search tree is as follows, for $s_0 = \{v \mapsto U \mid v \in V\}$:

- The root is Propagate(P, P, s_0).
- Node $\langle R, s \rangle$ has the n nodes Propagate($R \cup R_i, R_i, s$) as children, where $b(R, s) = \langle R_1, \ldots, R_n \rangle$ with $n \neq 1$; it is a leaf if $s = \emptyset$ (failed node) or $n = 0$ (solved node).

Definition (Variable selection strategies)

A brancher $b(R, s)$ selects a variable, based on either the current store s, or the current set R of propagators, or both (dynamic selection); or neither (static selection); or also the previously visited nodes (adaptive selection):

- Next: Select the next variable by order in the model.
- Random: Randomly select a variable unfixed by s.
- SizeMin: Select an unfixed var with smallest dom in s.
- DegreeMax: Select a variable v unfixed by s with the largest degree $(= |\{p \in R \mid v \in \text{var}(p)\}|)$ in R.
- AFCmin: Select a variable unfixed by s with the smallest accumulated failure count.
- ...

Ties are broken under any combination of these strategies.

See Topic 9 for selection strategies for set variables.

Definition (Value selection strategies)

Further, $b(R, s)$ selects values for the chosen variable v:

- Select the minimum: $\underline{m} = \min(s(v))$.
- Select the middle: $\dot{m} = \lfloor \frac{\min(s(v))+\max(s(v))}{2} \rfloor$.
- Select all the values of $s(v) = \{d_1, \ldots, d_n\}$.
- ...

We assume domains are ordered.

Definition (Decisions, or guesses)

Finally, $b(R, s)$ builds decisions, which are propagator sets:

- ValMin: Branch left on $\{v = \underline{m}\}$ and right on $\{v \neq \underline{m}\}$.
- SplitMin: Branch left on $\{v \leq \dot{m}\}$ and right on $\{v > \dot{m}\}$.
- ValuesMin: Branch left-right on $\{v = d_1\}, ..., \{v = d_n\}$.
- ...

See Topic 9 for selection strategies for set variables.

First-Fail Brancher

Example (SizeMin $\times$ Random + ValMin)

function $b(R, s)$
if $\exists v : |s(v)| > 1$ **then**
  pick random v
    such that $|s(v)| > 1$ **and** $|s(v)|$ is smallest
  return $\langle \{p_{v=\min(s(v))}\}, \{p_{v\neq\min(s(v))}\} \rangle$
else
  return $\langle\rangle$

UPPSALA UNIVERSITET

Outline

1 Branching

2 **Exploration**

Example (Depth-first first-sol'n search, bin. branching)

For $\langle V, U, P, b\rangle$ call DFE(P, P, s_0, b), which is defined as follows:

function DFE(R, Q, s, b)
$\langle R', s'\rangle :=$ Propagate(R, Q, s)
if $s' = \emptyset$ **then** // failed node
  return s'
else // s' is not necessarily a solution store
  $B := b(R', s')$
  if $B = \langle\rangle$ **then**
    return s' // solved node: s' is a solution store!
  else
    let $B = \langle R_1, R_2\rangle$
    $s'' :=$ DFE$(R' \cup R_1, R_1, s', b)$
    if $s'' = \emptyset$ **then** // failed node
      return DFE$(R' \cup R_2, R_2, s', b)$ // backtrack
    else
      return s'' // solved node: s'' is a solution store!

State Restoration Upon Backtracking

Approaches:

- Trailing: Remember changes and undo them.
  ☞ Most common approach, but difficult to implement, and difficult to combine with concurrency.
- Batch recomputation: Recompute state from the root.
  ☞ Problem-independent memory usage, but slow.
- Copying (or cloning): Store an additional copy.
  ☞ Easy to implement, and easy to combine with concurrency or parallelism, but too costly in memory.

Gecode uses a hybrid of copying and batch recomputation, called adaptive recomputation, which remembers a copy on the path from the root.

Diversification

Example (Multistart Exploration)

Perform several searches, sequentially or in parallel, especially in order to benefit from randomisation in branching strategies or from adaptive branching strategies:

- Stop each search (especially in sequential multistart) at some cutoff, say on the execution time, the number of visited nodes, or the number of failed nodes. Under the chosen cutoffs, the search may be incomplete.
- Specified as a sequence of $\langle b, e, c \rangle$ triples, each with a brancher b, exploration e, and cutoff c. Example:

$$\left[\begin{array}{c} \langle \text{SizeMin} \times \text{DegreeMax} + \text{ValMin}, \mathit{DFE}, 1000 \text{ fails} \rangle, \\ \langle \text{AFCmin} \times \text{Random} + \text{Random}, \mathit{DFE}, +\infty \text{ hours} \rangle \end{array}\right]$$

One can also solve a COP as a sequence of CSPs.