Topic 15: Search[1]
(Version of 14th November 2018)

Pierre Flener

Optimisation Group
Department of Information Technology
Uppsala University
Sweden

Course 1DL441:
Combinatorial Optimisation and Constraint Programming,
whose part 1 is Course 1DL451:
Modelling for Combinatorial Optimisation

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

Outline

1. Branching

2. Exploration

3. Dynamic Symmetry Breaking

Search = 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

1. Branching

2. Exploration

3. Dynamic Symmetry Breaking

Outline

1. Branching

2. Exploration

3. Dynamic Symmetry Breaking

Definition (Brancher)

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

- **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 node is $\text{Propagate}(P, P, s_0)$.
- Node $\langle R, s \rangle$ has the c nodes $\text{Propagate}(R \cup R_i, R_i, s)$ as children, where $b(R, s) = \langle R_1, \ldots, R_c \rangle$ with $c \neq 1$; it is a leaf if $s = \varnothing$ (failed node) or $c = 0$ (solved node).

Definition (Variable selection strategy)

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 in R $(= |\{p \in R \mid v \in \text{var}(p)\}|)$
- AFCmin: Select a variable unfixed by s with the smallest accumulated failure count
- ...

Ties are broken under any combination of these strategies.

UPPSALA UNIVERSITET

Branching

Exploration

Dynamic Symmetry Breaking

Definition (Value selection strategy)

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

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

We assume domains are ordered.

Definition (Decision, or guess)

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 $\{v = d_1\}, \ldots, \{v = d_n\}$
- …

Set Variables (Reminder)

Definition

A set (decision) variable takes an integer set as value, and has a set of integer sets as domain. For its domain to be finite, a set variable must be a subset of a finite set Σ.

CP solvers over-approximate the domain of a set variable S by a pair $\langle \ell, u \rangle$ of finite sets, denoting the set of all sets σ such that $\ell \subseteq \sigma \subseteq u \subseteq \Sigma$:

- ℓ is the current set of mandatory elements of S;
- $u \setminus \ell$ is the current set of optional elements of S.

Example

The domain of a set var represented as $\langle \{1\}, \{1, 2, 3, 4\} \rangle$ has the sets $\{1\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, and $\{1, 2, 3, 4\}$. Removing $\{1, 2, 3\}$ is impossible!

Strategies for the selection of a set variable $S \doteq \langle \ell, u \rangle$:

- SizeMin: Select a set variable with smallest $|u \setminus \ell|$
- MinMax: Select a set variable with largest $\min(u \setminus \ell)$
- ...

Strategies for the selection of an optional element of S:

- Select the minimum: $\underline{m} = \min(u \setminus \ell)$
- Select the median $\dot{m}$ of $u \setminus \ell$
- Select a random element r of $u \setminus \ell$
- ...

Branching decisions on inclusion and exclusion:

- MinInc: Branch left on $\{\underline{m} \in S\}$ and right on $\{\underline{m} \notin S\}$
- RndExc: Branch left on $\{r \notin S\}$ and right on $\{r \in S\}$
- ...

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 $\left\langle \left\{ p_{v=\min(s(v))} \right\}, \left\{ p_{v \neq \min(s(v))} \right\} \right\rangle$
else
  return $\langle\rangle$

Outline

UPPSALA UNIVERSITET

Branching

Exploration

Dynamic Symmetry Breaking

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 $s' = \varnothing$ **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 $s'' = \varnothing$ **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.

UPPSALA UNIVERSITET

Outline

1. Branching

2. Exploration

3. **Dynamic Symmetry Breaking**

Dynamic Symmetry Breaking (DSB)

Definition

DSB = the elimination of symmetric solutions by search.

Classification:

- Via the addition of constraints by the search procedure.
- Via a problem-specific search procedure.

Benefit:
No interference with dynamic variable and value selection strategies, especially problem-specific ones!

State of the Art

Two dual approaches, with large bodies of research:

- Symmetry breaking during search (SBDS, ...): after reaching a leaf (failed or solved node) in the search tree, add constraints preventing its symmetric nodes from being visited in the future.
- Symmetry breaking by dominance detection (SBDD, GCF, ...): before expanding a node, check whether a symmetric node thereof has been visited in the past.

The SBD$*$ schemes are general and may take exponential time or space if there are exponentially many symmetries (and they are beyond the scope of this topic). Hence:

- Dynamic structural symmetry breaking: exploit the combinatorial structure of a problem for designing a symmetry-free search procedure (in SBDD style).

Full Value Symmetry

Example (Map colouring: symmetry-free search)

Given a partial colouring using u colours, only $u + 1$ colours need to be considered for the next country c:

- Colour c with one of the u already used colours.
- Colour c with an arbitrary unused colour, if any left.

In practice: The already used colours are the first u colours, say $0, \ldots, u - 1$, so that the new colour to be considered is u.
This breaks all the $n!$ value symmetries in constant time and constant space overhead at every node explored!
We say that it takes constant amortised time & space.

Applications (Van Hentenryck [& Michel]):

- Scene allocation (*INFORMS J. of Computing*, 2002)
- Steel mill slab design (*CPAIOR 2008*)

UPPSALA UNIVERSITET

Partial Value Symmetry (*IJCAI 2003*)

Branching

Exploration

Dynamic Symmetry Breaking

Example (Partial value symmetry; often in instances)

Weekdays vs weekend days; same-size boats.

Clustering:

Let $D = D_1 \cup D_2 \cup \cdots \cup D_m$ be the domain of the variables, where the values in each set D_i are fully interchangeable (full value sym for $m = 1$): cluster the variables for each D_i.

Search procedure at constant amortised time & space:

In each set D_i, only the values already used and one so far unused value need to be tried.

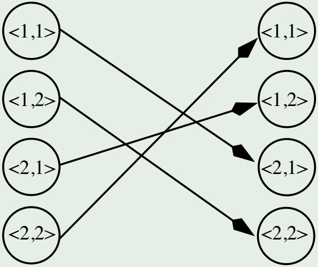
Application (Michel, . . . , Van Hentenryck, *CPAIOR'08*):

- Eventually-serialisable data service deployment

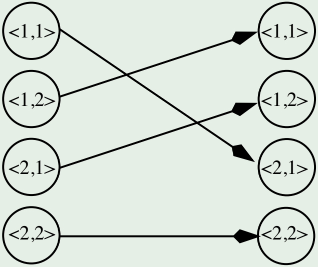
Wreath Value Symmetry (*IJCAI 2003*)

Example (Wreath value symmetry)

Schedule meetings in (day, room) pairs,
where the days are interchangeable,
and the rooms are interchangeable within each day:



Wreath permutation



Not a wreath permutation!

UPPSALA UNIVERSITET

Branching

Exploration

Dynamic Symmetry Breaking

Clustering:

Let $D = D_1 \times D_2$ be the domain of the pairs of variables, where the values in each set D_i are fully interchangeable (full value symmetry for $|D_2| = 1$): one cluster for D_1, and m clusters for D_2 when m values of D_1 are used, with variable clustering as for full value symmetry.

Search procedure at constant amortised time & space:

1. For the first value component, in set D_1, only the values already used and one so far unused value need to be tried. Let $d_1 \in D_1$ be the chosen value.
2. For the second value component, in set D_2, only the values already used with d_1 and one so far unused value need to be tried.

UPPSALA UNIVERSITET

Branching

Exploration

Dynamic Symmetry Breaking

Selected Other Results

Consider a combinatorial problem with n decision variables over a domain of k values:

- **Generalisation to any value symmetry**:
  group equivalence (GE) trees
  (Roney-Dougal *et al.*, *ECAI 2004*)
  ☞ $O(n^4)$ time overhead at every node explored.

- **Partial variable symmetry + partial value symmetry**
  (Sellmann & Van Hentenryck, *IJCAI 2005*)
  ☞ $O(k^{2.5} + n \cdot k)$ time at every node explored.
  ☞ Coinage of the term structural symmetry breaking.
  ☞ Can be specialised for full variable symmetry only.

Tractability: State of the Art

value symmetry \ variable symmetry	none	full	partial	wreath	
none		P **P**	P **P**	P **P**	scalar problem **set problem**
full	P **P**	P **NP**	P **NP**	NP **NP**	scalar problem **set problem**
partial	P **P**	P **NP**	P **NP**	NP **NP**	scalar problem **set problem**
wreath	P **P**	P **NP**	P **NP**	NP **NP**	scalar problem **set problem**
any	P				scalar problem **set problem**

P: All symmetric sub-trees can be eliminated with a polynomial time & space overhead at every node explored.

NP: Dominance-detection schemes (in SBDD style) are NP-hard.

UPPSALA UNIVERSITET

Branching

Exploration

Dynamic Symmetry Breaking

Bibliography

P. Flener, J. Pearson, M. Sellmann, P. Van Hentenryck, M. Ågren.
Dynamic SSB for constraint satisfaction problems.
Constraints, 14(4):506–538, December 2009.
(Supersedes our IJCAI 2003 and IJCAI 2005 papers.)

P. Flener, J. Pearson, and M. Sellmann.
Static and dynamic structural symmetry breaking.
Annals of Mathematics and Artif. Intell., 57(1):37–57, Sept. 2009.
(Supersedes our CP 2006 paper with P. Van Hentenryck.)

P. Flener and J. Pearson.
Solving necklace constraint problems.
Journal of Algorithms, 64(2–3):61–73, April–July 2009.
(Supersedes our ECAI 2008 paper.)

F. Ruskey.
Combinatorial Generation.
Unpublished book draft, 2003.