Modeling Solution Dominance over CSPs

Tias Guns, Peter Stuckey, Guido Tack

ModRef 2018
Constrained satisfaction and optimisation

Constraint modeling languages

Satisfaction
- Find a satisfying solution
  (or find all satisfying solutions)

Optimisation
- Minimize/maximize one objective
- Find a best solution
Beyond optimisation

- Lexicographic optimisation
- Multi-objective optimisation (*pareto-frontier solutions*)
- X-minimal models (*solutions with smallest subset of true Boolean variables in set X*)
- Weighted (partial) MaxCSP (*like MaxSAT*)
- Valued CSP (*each constraint has a value for being satisfied*)
- Maximally Satisfiable subsets (*MSS, MCS, MUS*)
- CP-nets (*expresses preferences through a DAG of conditional preference tables*)
- Domain specific dominance relations (*e.g. in itemset mining: closedness and maximality*)

→ *not available in constraint modeling languages!*
A solution **dominance relation** specifies when one solution dominates another.

\[ \text{find } \{ X \in S | \not\exists Y \in S \ldots \} \text{ where } S \text{ is the set of all solutions of a CSP} \]

How to formalize that one solution dominates another?
A pre-order is reflexive and transitive
→ think partial order with equivalence classes

Examples dominance relations:
• Optimisation (min): \( X \preceq_f Y \iff f(X) \leq f(Y) \)
• Multi-objective optimisation: \( X \preceq_F Y \iff \forall_i f_i(X) \leq f_i(Y) \)
• \( X \)-minimal models: \( X \preceq_X Y \iff \forall v \in X : X(v) \leq Y(v) \)
  \( X(v) \) is truth value \( \{0,1\} \) of \( v \) in \( X \)
From dominance relation to solution set

What is the **solution set** of a Constrained Dominance Problem (CDP)?

- Complete (every CSP solution is dominated or equivalent to one of the CDP solution)
- Domination-free (CDP solutions are not dominated by other CDP solutions, except equivalent ones)

→ this set is unique

→ in Multi-Objective optimisation, this is the *efficient set*

- Complete
- Domination-free
- Equivalence-free (no two CDP solutions are equivalent to each other)

→ this set is NOT unique

→ equivalent solutions are typically not of interest

(even so in standard optimisation)
Detailed example: multi-objective

Multi-objective

\[
\{ X \in S \mid \exists Y \in S : Y \preceq_F X \land X \simeq_F Y \}
\]

\[\leftrightarrow \{ X \in S \mid \exists Y \in S : \forall_i f_i(Y) \leq f_i(X) \land \neg(\forall_j f_j(X) = f_j(Y))\}\]

\[\leftrightarrow \{ X \in S \mid \exists Y \in S : \forall_i f_i(Y) \leq f_i(X) \land \exists_j f_j(X) \neq f_j(Y)\}\]

\[\leftrightarrow \{ X \in S \mid \exists Y \in S : \forall_i f_i(Y) \leq f_i(X) \land \exists_j f_j(X) < f_j(Y)\}\]

which is the classical definition of multi-objective optimization [9].
More examples...

X-minimal models: $\{X \in S|\exists Y \in S : pos_X(Y) \subset pos_X(X)\}$

CP-net:

- dominance in terms of preference ranking (the typical one): NP-hard
- can play with other dominance relations, e.g. local dominance (for equal parents only)

![CP-net example over 3 variables.](image)
Domain specific examples...

Frequent itemset mining: find all solutions \( X \) where \( \text{freq}(X,D) \geq \text{Value} \)

Maximal freq. itemsets: there does not exist a subset that is also frequent

\[ \rightarrow X\text{-maximal solutions!} \]

Closed freq. itemsets: there does not exist a subset that has the same frequency

\[ \rightarrow \text{conditional } X\text{-maximal solutions!} \]

\[ \rightarrow \text{compatible with arbitrary constraints} \text{ (a positive thing in constrained itemset mining)} \]

Specifically for itemset mining studied in:
[B. Negrevergne, A. Dries, T. Guns, S. Nijssen, Dominance programming for itemset mining, ICDM 2013]
Specific settings have specific, efficient, solving methods
e.g. multi-objective, MaxCSP, MUS, ...

But domain-specific ones don't. General search mechanism?
→ incrementally add non-backtrackable nogoods

Algorithm 1 search\((V, D, C, \preceq, \emptyset)\):
1: \(A := \emptyset\)
2: \textbf{while} \(S := O(V, D, C)\) \textbf{do}
3: \(A := A \cup \{S\}\)
4: \(C := C \cup \{S \nrightarrow V \vee S \sim V\}\)
5: \textbf{end while}
6: \textbf{return} \(A\)
Modeling in a language

We propose to model **dominance nogoods**, rather than dominance relations:

1) can be used to specify both equivalence-free and with equivalences

2) we found it more intuitive to specify an *invariant* for the search
   (e.g. in case of minimisation, if S is a solution then $f(V) < f(S)$ for any future solution V)

   ```
   dominance_nogood f(V) < f(sol(V));
   ```
Modeling and search in MiniZinc

Modeling: a primitive for specifying a dominance nogood

```
dominance_nogood exists(i in index_set(B))(B[i] < sol(B[i]));
```

Search: post a (non-backtrackable) constraint each time a solution is found

```
solve search dominance_search;
function ann: dominance_search() =
repeat( if next() then
    commit() /
    print() /
    post_dng()
else break endif);
```

*solve search = MiniSearch extension

Example experiments

**Constraint dominance problems** in a declarative solver-independent language

Solvers:

- gecode-api with minisearch incremental API
- gecode/ortools/chuffed with minisearch black box restarts

Search strategy: **free** or such that preferred assignments are enumerated first (**ordered**)
Example: MaxCSP

Providing a guiding search strategy often helps, but not always!
Different solvers behave quite differently, can compare thanks to solver-independence

Table 1. MaxCSP runtimes in seconds, — timed out after 30 min.

<table>
<thead>
<tr>
<th>Instance</th>
<th>gecode-api</th>
<th>gecode</th>
<th>ortools</th>
<th>chuffed</th>
<th>optcpx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>free</td>
<td>ord</td>
<td>free</td>
<td>ord</td>
<td>free</td>
</tr>
<tr>
<td>cabinet-5570</td>
<td>—</td>
<td>0.9</td>
<td>—</td>
<td>—</td>
<td>36</td>
</tr>
<tr>
<td>cabinet-5571</td>
<td>—</td>
<td>0.9</td>
<td>—</td>
<td>—</td>
<td>36</td>
</tr>
<tr>
<td>latinSq-dg-3_all</td>
<td>0.2</td>
<td>0.1</td>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>latinSq-dg-4_all</td>
<td>0.6</td>
<td>0.9</td>
<td>0.8</td>
<td>6.8</td>
<td>0.5</td>
</tr>
<tr>
<td>quasigrp4-4</td>
<td>46</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>4.5</td>
</tr>
<tr>
<td>quasigrp5-4</td>
<td>0.4</td>
<td>1651</td>
<td>1158</td>
<td>—</td>
<td>1.1</td>
</tr>
<tr>
<td>q13-1110973670</td>
<td>479</td>
<td>1.1</td>
<td>32</td>
<td>0.9</td>
<td>540</td>
</tr>
<tr>
<td>q13-1111219348</td>
<td>569</td>
<td>1.1</td>
<td>32</td>
<td>1.3</td>
<td>385</td>
</tr>
</tbody>
</table>
Example: Bi-objective TSP

- Shows number of *intermediate* solutions (not final frontier size)
- Top-rows: free search, bottom-rows: max regret search → search strategy helps
- Oscar has efficient global bi-objective constraint (only relevant in free search)
Conclusion

Beyond satisfaction/optimisation:

**Constraint dominance problems**

in a declarative solver-independent language

- from dominance relation to dominance nogoods
- can be added to modeling languages

→ creates breathing room for domain-specific dominance relations? (examples?)
Modeling Solution Dominance over CSPs

Tias Guns, Peter Stuckey, Guido Tack
ModRef 2018