Modeling Solution Dominance over CSPs

Tias Guns, Peter Stuckey, Guido Tack

ModRef 2018

VUB VRIJE UNIVERSITEIT BRUSSEL
MONASH University

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!*

Solution dominance

A solution **dominance relation** specifies when one solution dominates another

$$\text{find } \{X \in \mathcal{S} | \nexists Y \in \mathcal{S}...\} \text{ where } \mathcal{S} \text{ i}\quad \text{set of all solutions of a CSP}$$

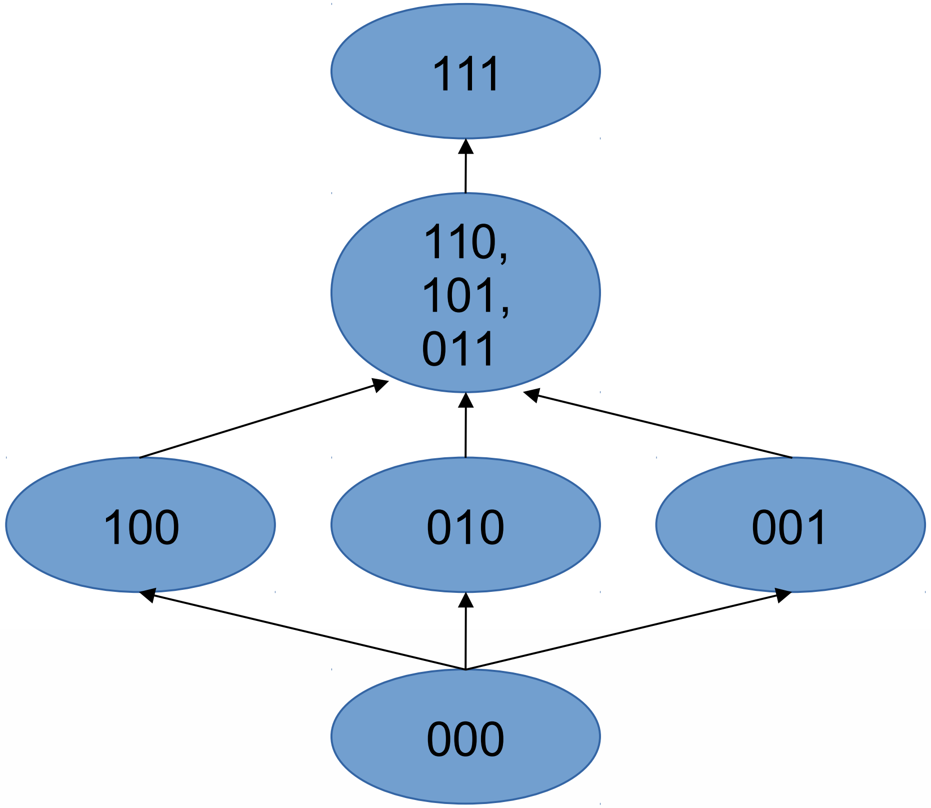
How to formalize that one solution dor

VRIJE UNIVERSITEIT BRUSSEL

Pre-order

A pre-order is *reflexive* and *transitive*

$\rightarrow$ think partial order with equivalence classes



Examples dominance relations:

- Optimisation (min): $X \preceq_f Y \Leftrightarrow f(X) \leq f(Y)$
- Multi-objective optimisation: $X \preceq_F Y \Leftrightarrow \forall_i f_i(X) \leq f_i(Y)$
- X-minimal models: $X \preceq_{\mathcal{X}} Y \Leftrightarrow \forall v \in \mathcal{X} : X(v) \leq Y(v)$ *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 dominanted 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)

VRIJE UNIVERSITEIT BRUSSEL

Detailed example: multi-objective

Multi-objective

$$
\begin{aligned}
& \{X \in S \mid \nexists Y \in S : Y \preceq_F X \wedge X \nsim_F Y\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_i f_i(Y) \leq f_i(X) \wedge \neg(\forall_j f_j(X) = f_j(Y))\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_i f_i(Y) \leq f_i(X) \wedge \exists_j f_j(X) \neq f_j(Y)\} \\
\leftrightarrow\ & \{X \in S \mid \nexists Y \in S : \forall_i f_i(Y) \leq f_i(X) \wedge \exists_j f_j(X) < f_j(Y)\}
\end{aligned}
$$

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

More examples...

X-minimal models: $\rightarrow \{X \in S \mid \nexists Y \in S : pos_{\mathcal{X}}(Y) \subset pos_{\mathcal{X}}(X)\}$

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)

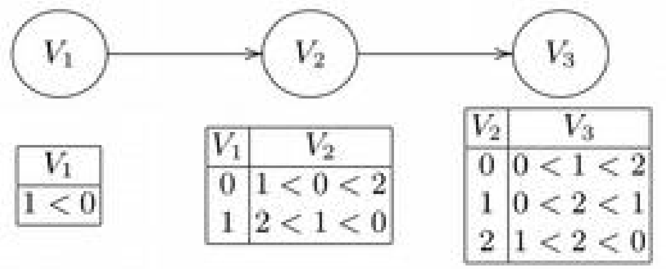


Fig. 1. CP-net example over 3 variables.

Domain specific examples...

Frequent itemset mining: find all solutions X where freq(X,D) >= Value

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

$\rightarrow$ X-maximal solutions!

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

$\rightarrow$ conditional X-maximal solutions!

→ **compatible with arbitrary constraints** (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]

Search

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($V, D, C, \preceq, \mathcal{O}$):

$A := \emptyset$
while $S := \mathcal{O}(V, D, C)$ **do**
  $A := A \cup \{S\}$
  $C := C \cup \{S \not\preceq V \vee S \sim V\}$
end while
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

[A. Rendl, T. Guns, P. Stuckey, G. Tack. MiniSearch:
A solver-independent meta-search language for minizinc, CP 2015]

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

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

Instance	gecode-api		gecode		ortools		chuffed		optcpx	
	free	ord	free	ord	free	ord	free	ord	free	ord
cabinet-5570	—	0.9	—	—	36	0.2	257	—	3.9	0.3
cabinet-5571	—	0.9	—	—	36	0.2	257	—	3.9	0.4
latinSq-dg-3_all	0.2	0.1	0.5	0.3	0.1	0.1	0.2	0.3	0.1	0.1
latinSq-dg-4_all	0.6	0.9	0.8	6.8	0.5	1.3	0.5	13	0.6	0.3
quasigrp4-4	46	—	—	—	4.5	—	3.8	18	1.4	7.7
quasigrp5-4	0.4	1651	1158	—	1.1	—	1.6	5.4	1.6	1.3
q13-1110973670	479	1.1	32	0.9	540	0.7	635	43	11	7.5
q13-1111219348	569	1.1	32	1.3	385	0.9	641	72	8.8	7.0

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

Example: Bi-objective TSP

Instance	gecode-api		gecode		ortools		chuffed		oscar	
	time	sols	time	sols	time	sols	time	sols	time	sols
ren10	0.5	108	7.5	108	6.8	108	38	105	2.3	110
ren15	368	949	—	545	–	565	—	343	61	891
ren20	—	998	—	382	–	392	—	381	—	—
ren10-mg	1.8	41	2.8	41	1.3	45	5	38	n.a.	n.a.
ren15-mg	14	135	247	135	541	145	—	128	n.a.	n.a.
ren20-mg	—	925	—	292	—	294	—	171	n.a.	n.a.

- 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

$\rightarrow$ creates breathing room for domain-specific dominance relations? (examples?)

Modeling Solution Dominance over CSPs

Tias Guns, Peter Stuckey, Guido Tack

ModRef 2018

VRIJE UNIVERSITEIT BRUSSEL
MONASH University