Google Research

ViolationLS
Constraint Based Local Search in CP-SAT

Toby Davies, Frédéric Didier, & Laurent Perron
Google Research, Paris

Constraint-Based Local Search

Constraint Based: inspired by Constraint Programming

Many problems have similar combinatorial structures, don't make users re-implement them!

Constraint-Based Local Search

Local Search: makes local changes to an assignment

Constraints define "how violated" they are in an assignment, solvers first minimize total violation, then the objective.

Constraints typically also define a neighborhood.*

Violation Functions

$$V\left(\sum cx \leq K\right) = \min\left(\sum cx - K, 0\right)$$

Feasibility Jump

Feasibility Jump = Single-var moves + Guided Local Search

Feasibility Jump

Feasibility Jump changes **one variable at a time**

The variable "jumps" to the value that minimizes the **weighted sum** of constraint violations for that variable.

If no moves reduce weighted violation, **increase the weight of violated constraints**

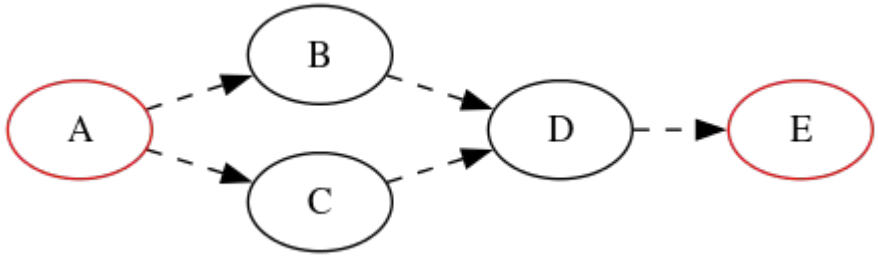
If feasible, **emit the new solution** and replace objective constraint:
objectiveVar < currentObjectiveValue

Feasibility Jump

Computing the "Jump Value" is cheap if we assume violation is convex **when changing a single variable**:

$$O\left(\ln(x_{max} - x_{min})\right)$$

Network Flow Example

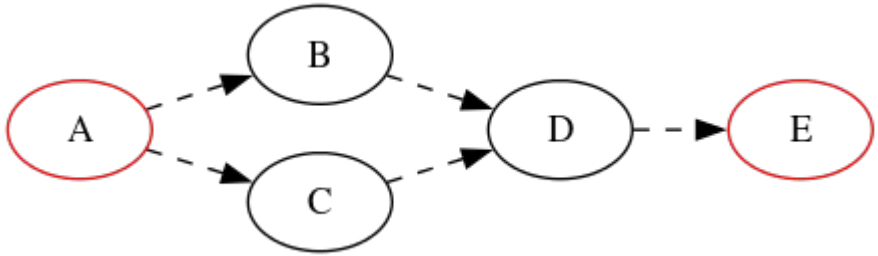


Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	1	1	1	1	1
Violation	1	0	0	0	1

	AB	AC	BD	CD	DE
Score	0	0	-2	-2	0
Value	0	0	0	0	0

Network Flow Example

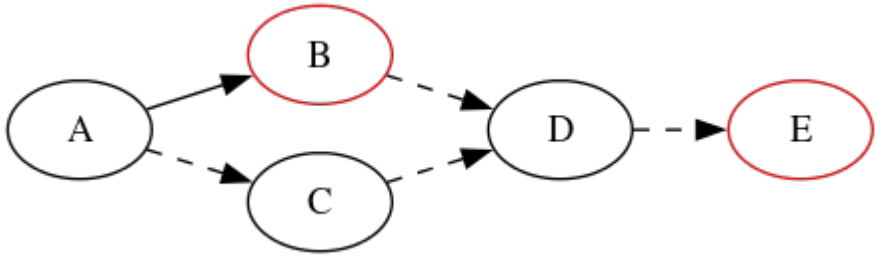


Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	**2**	1	1	1	**2**
Violation	1	0	0	0	1

	AB	AC	BD	CD	DE
Score	**1**	**1**	-2	-2	**1**
Value	0	0	0	0	0

Network Flow Example



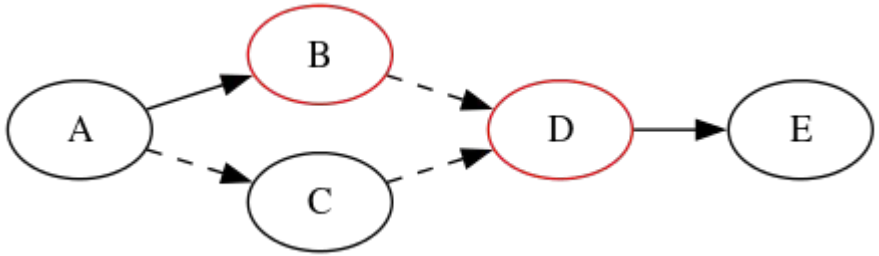
Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	2	1	1	1	2
Violation	**0**	**1**	0	0	1

	AB	AC	BD	CD	DE
Score	**0**	**-3**	**0**	**0**	1
Value	**1**	0	0	0	0

Google Research

Network Flow Example

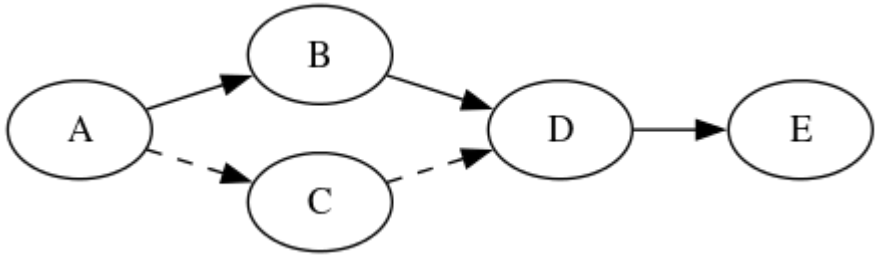


Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	2	1	1	1	2
Violation	**0**	**1**	0	**1**	**0**

	AB	AC	BD	CD	DE
Score	**0**	**-3**	**2**	**0**	**0**
Value	**1**	0	0	0	**1**

Network Flow Example



Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	2	1	1	1	2
Violation	0	**0**	0	**0**	0

	AB	AC	BD	CD	DE
Score	0	-1	**0**	**-1**	0
Value	1	0	**1**	0	1

Multi-Variable Moves

Usually local search changes more than 1 variable at a time!

We'd like a way to find chains of moves automatically,
while using the same operations as Feasibility Jump

But I'm lazy: I don't want to write specialized neighbourhoods

Local Search as Planning

Idea: Treat the search for a move as a domain-independent planning problem.

Note: this is just an inspiration, we don't really solve a planning problem!

Novelty

"Novelty" based search: only expands "novel" states

More structured than random exploration

Polynomial work per search

Novelty

What makes a state novel?

Some atomic fact is true in the state that has never been true in any prior one

Novelty

What makes an assignment novel?

A constraint is satisfied that has not been satisfied before?

Novelty

Wh assignment

nstraint is s that ha been satisfied before?

Novelty

What makes an assignment novel?

*A constraint is **violated** that has not been **violated** before*

Or (non-discounted) weighted violation is less than any prior assignment

Novelty Jump

How to make "jump if weighted violation decreases" find novel states:

- Discount the weight of satisfied constraints (multiply by epsilon)
- Reset the weight when a constraint becomes violated
- Keep a stack of var changes & backtrack when we have no good moves*
- Clear the stack when sum of (non discounted) scores are positive

Novelty Jump: Why backtrack?

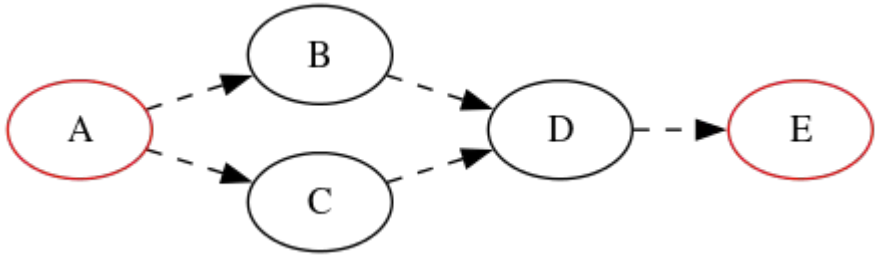
Discounted score can be positive, but real score negative (*which is the point*)!

If we commit such a move we may not find our way back.

This would be an interesting metaheuristic, but not a compound move.

This interferes with GLS rather than complements it

Novelty Example

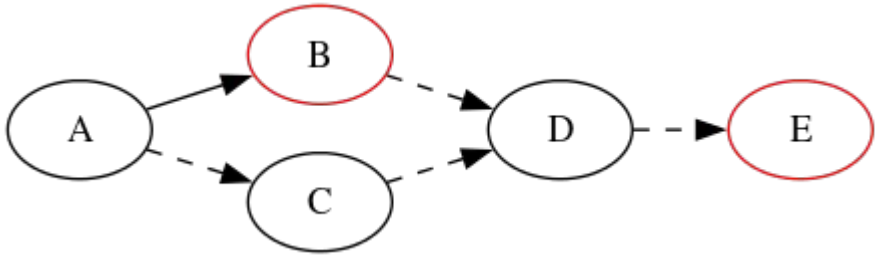


Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	1	ε	ε	ε	1
Violation	1	0	0	0	1

	AB	AC	BD	CD	DE
Score	1-ε	1-ε	-2ε	-2ε	1-ε
Value	0	0	0	0	0

Novelty Example

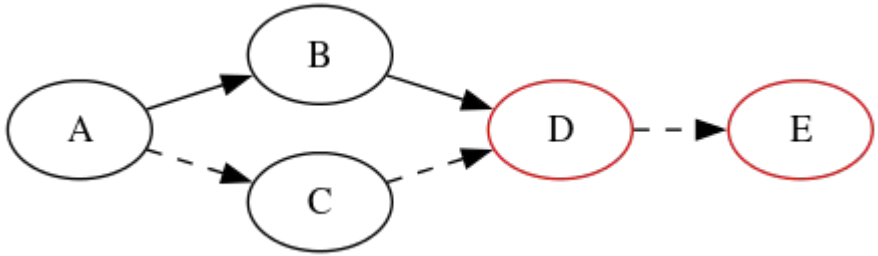


Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	1	**1**	ε	1	1
Violation	**0**	**1**	0	0	1

	AB	AC	BD	CD	DE
Score	**0**	**-1-ε**	1-ε	-2ε	1-ε
Value	**1**	0	0	0	0

Novelty Example

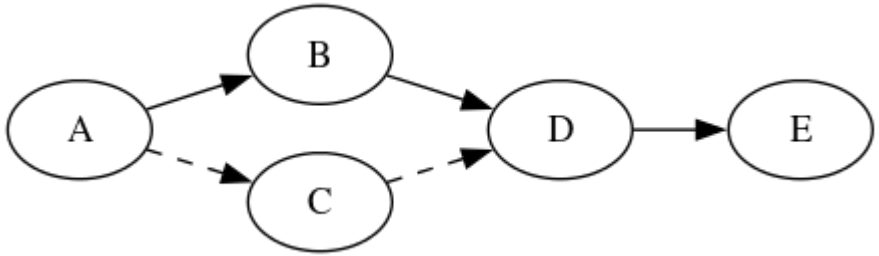


Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	1	1	ε	**1**	1
Violation	**0**	**0**	0	**1**	1

	AB	AC	BD	CD	DE
Score	**0**	**-1-ε**	**0**	**1-ε**	**2**
Value	**1**	0	**1**	0	0

Novelty Example



Arc vars have domain {0,1};
1 inflow at A; 1 outflow at E

	A	B	C	D	E
Weight	1	1	ε	1	1
Violation	0	**0**	0	**0**	0

	AB	AC	BD	CD	DE
Score	0	-1-ε	-2	-1-ε	0
Value	1	0	1	0	1

Is Novelty Jump always better?

No! But you don't have to pick!

ViolationLS:

- If there is a new better solution from some other worker:
  - Set incumbent to the new solution
  - Randomly pick Novelty Jump or Feasibility Jump
- Do some iterations
- Repeat

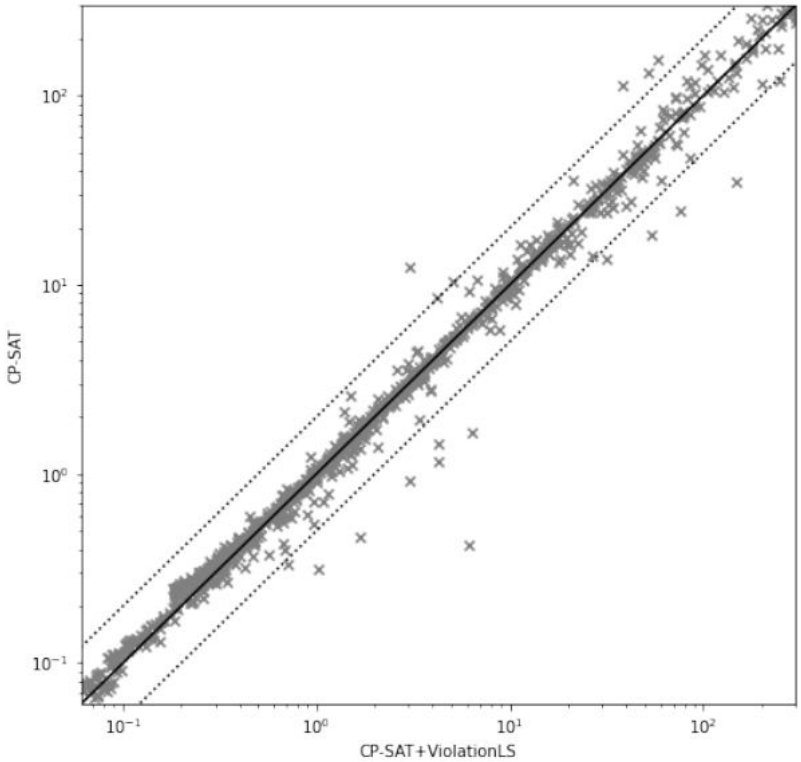
Results

Solver	ViolationLS	Feasibility Jump	Yuck	fzn-oscar-cbls
ViolationLS	-	**650.6**	**652.28**	**734.64**
Feasibility Jump	182.00	-	523.84	617.24
Yuck	**209.52**	268.96	-	435.72
fzn-oscar-cbls	101.96	135.36	147.6	-

#instances (of 1216) where ViolationLS beats Yuck

#instances where row solver beats column (1216 mzn challenge instances)
Using 8 cores, except fzn-oscar-cbls which is single-threaded
Averaged over 5 seeds per solver

Results



Solve time of CP-SAT with and without ViolationLS
60.5% faster; 39.5% slower

Results

CP-SAT+	ViolationLS	Feasibility Jump	Baseline
ViolationLS	-	**609.20**	**659.04**
Feasibility Jump	**588.36**	-	651.80
Baseline	538.32	545.48	-

#instances where column solver beats row (1216 mzn challenge instances)
Using 8 cores

Conclusions

- CBLS makes a powerful primal heuristic for a state-of-the-art CP solver
- Efficiently implemented, generic LS moves can beat constraint-specific ones
- Novelty is a good exploration criterion that integrates well with GLS