Dependency-Curated Large Neighbourhood Search

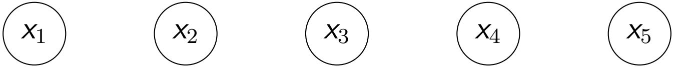
Pierre Flener,
Frej Knutar Lewander, Justin Pearson

Uppsala University
Sweden

21st August 2025

Large Neighbourhood Search (LNS)

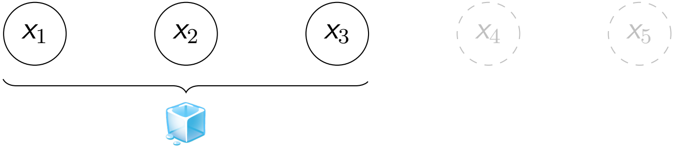
LNS is an iterative approach for solving optimisation problems.



Large Neighbourhood Search (LNS)

LNS is an iterative approach for solving optimisation problems.

In each LNS iteration, a *freeze set* is selected, where the value of each variable in the freeze set is kept from the incumbent solution.

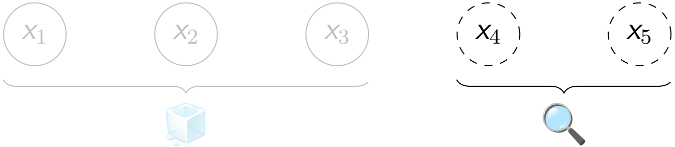


Large Neighbourhood Search (LNS)

LNS is an iterative approach for solving optimisation problems.

In each LNS iteration, a *freeze set* is selected, where the value of each variable in the freeze set is kept from the incumbent solution.

The value of each remaining variable is found and assigned via solving.



Selection Heuristics

- The selection heuristic selects a freeze set in each LNS iteration.

Selection Heuristics

- The selection heuristic selects a freeze set in each LNS iteration.

- A selection heuristic can be generic or specific to a problem.

Selection Heuristics

- The selection heuristic selects a freeze set in each LNS iteration.
- A selection heuristic can be generic or specific to a problem.
- Much research has been performed on finding LNS selection heuristics that select freeze sets that lead to high-quality solutions.

Relaxed Car Sequencing

Running example: relaxed car sequencing (RCS) description

- For each feature and any three subsequent cars, at most two of them can have that feature.
- Produce two cars of each car type.

Car Type	Red Paint	Windshield
🛻	✓	✓
🏎️	✓	✗
🚙	✗	✓

Relaxed Car Sequencing

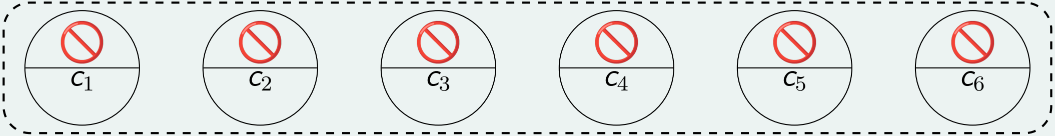
Running example: relaxed car sequencing (RCS) description

- For each feature and any three subsequent cars, at most two of them can have that feature.
- ~~Produce two cars of each car type.~~
- Produce *at most* two cars of each non-dummy car type.
- Minimise the number of produced dummy cars.

Car Type	Red Paint	Windshield
🛻	✓	✓
🏎	✓	✗
🚙	✗	✓
🚫	✗	✗

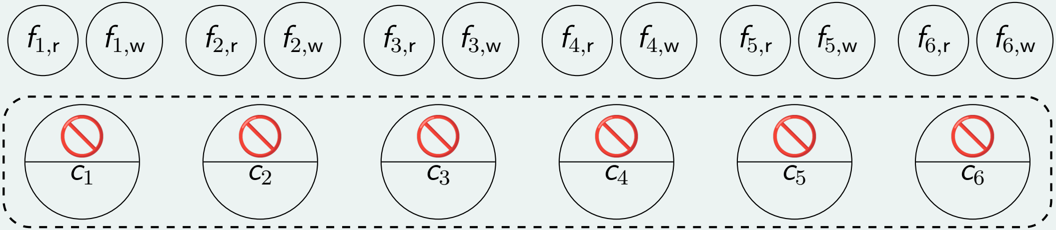
Relaxed Car Sequencing

Running example: RCS model & visualisation



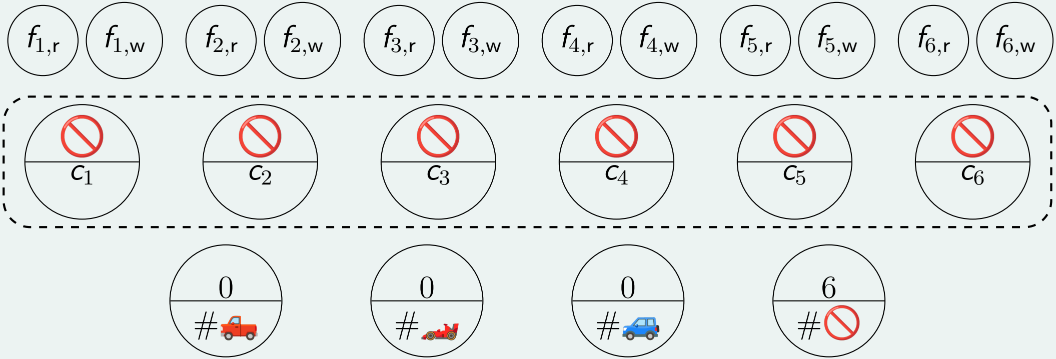
Relaxed Car Sequencing

Running example: RCS model & visualisation



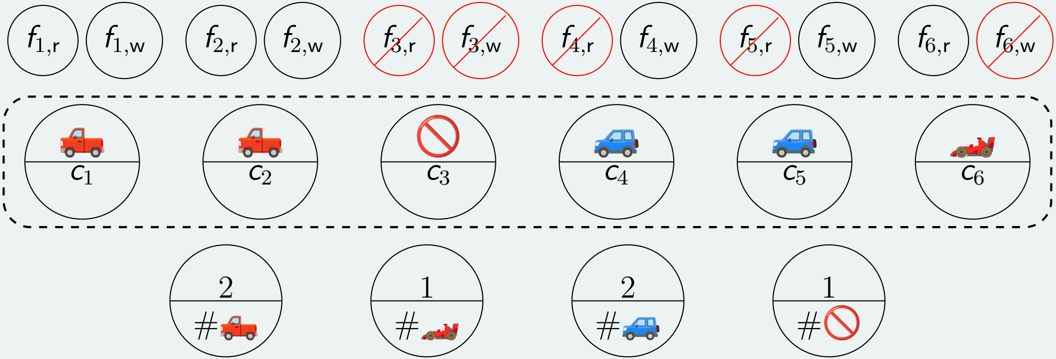
Relaxed Car Sequencing

Running example: RCS model & visualisation



Relaxed Car Sequencing

Running example: RCS model & visualisation



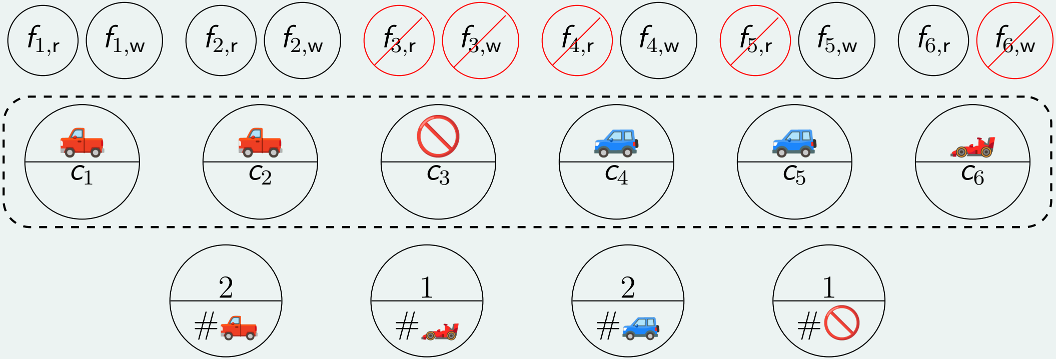
$f_{1,r}$ $f_{1,w}$ $f_{2,r}$ $f_{2,w}$ $\cancel{f_{3,r}}$ $\cancel{f_{3,w}}$ $\cancel{f_{4,r}}$ $f_{4,w}$ $\cancel{f_{5,r}}$ $f_{5,w}$ $f_{6,r}$ $\cancel{f_{6,w}}$

🛻 c_1 🛻 c_2 🚫 c_3 🚙 c_4 🚙 c_5 🏎️ c_6

2 #🛻 1 #🏎️ 2 #🚙 1 #🚫

Relaxed Car Sequencing

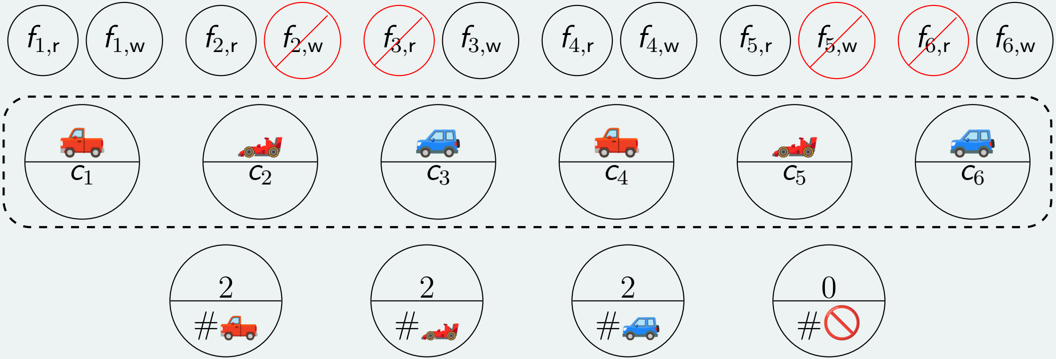
Running example: RCS model & visualisation



Any assignment of the variables $\{c_1, \ldots, c_6\}$ fixes the values of the remaining variables.

Relaxed Car Sequencing

Running example: RCS model & visualisation


$f_{1,r}$ $f_{1,w}$ $f_{2,r}$ $\cancel{f_{2,w}}$ $\cancel{f_{3,r}}$ $f_{3,w}$ $f_{4,r}$ $f_{4,w}$ $f_{5,r}$ $\cancel{f_{5,w}}$ $\cancel{f_{6,r}}$ $f_{6,w}$
c_1
c_2
c_3
c_4
c_5
c_6
2 #
2 #
2 #
0 #

Any assignment of the variables $\{c_1, \dots, c_6\}$ fixes the values of the remaining variables.

Dependency Constraints

Definition: dependency constraint

A *dependency constraint* c:

Dependency Constraints

Definition: dependency constraint

A *dependency constraint c*:

- has variables $\mathcal{X} \cup \mathcal{Y}$, where

Dependency Constraints

Definition: dependency constraint

A *dependency constraint* c:

- has variables $\mathcal{X} \cup \mathcal{Y}$, where
  - $\mathcal{X}$ are the input variables and
  - $\mathcal{Y}$ are the output variables.

Dependency Constraints

Definition: dependency constraint

A *dependency constraint* c:

- has variables $\mathcal{X} \cup \mathcal{Y}$, where
  - $\mathcal{X}$ are the input variables and
  - $\mathcal{Y}$ are the output variables.
- determines the values of the output variables $\mathcal{Y}$ given the values of the input variables $\mathcal{X}$.

Dependency Constraints

Definition: dependency constraint

A *dependency constraint* c:

- has variables $\mathcal{X} \cup \mathcal{Y}$, where
  - $\mathcal{X}$ are the input variables and
  - $\mathcal{Y}$ are the output variables.
- determines the values of the output variables $\mathcal{Y}$ given the values of the input variables $\mathcal{X}$.

We say that c functionally defines output variables $\mathcal{Y}$ given input variables $\mathcal{X}$.

Dependency Constraints

Definition: dependency constraint

A *dependency constraint* c:

- has variables $\mathcal{X} \cup \mathcal{Y}$, where
  - $\mathcal{X}$ are the input variables and
  - $\mathcal{Y}$ are the output variables.
- determines the values of the output variables $\mathcal{Y}$ given the values of the input variables $\mathcal{X}$.

We say that c functionally defines output variables $\mathcal{Y}$ given input variables $\mathcal{X}$.

Example: dependency constraint

The number of variables in $\{c_1, \ldots, c_6\}$ that take value 🛻.

General Idea

For a generic selection heuristic, *any* variable can typically be included in the freeze set.

General Idea

For a generic selection heuristic, *any* variable can typically be included in the freeze set.

We want to exclude variables from the freeze set that are functionally defined by other variables.

General Idea

For a generic selection heuristic, *any* variable can typically be included in the freeze set.

We want to exclude variables from the freeze set that are functionally defined by other variables.

Our hope is that doing this will:

- reduce the overall memory footprint and

General Idea

For a generic selection heuristic, *any* variable can typically be included in the freeze set.

We want to exclude variables from the freeze set that are functionally defined by other variables.

Our hope is that doing this will:

- reduce the overall memory footprint and
- lead to high-quality solutions.

Set of Search Variables

Definition: set of search variables

A set of *search variables* transitively functionally defines all other variables.

Set of Search Variables

Definition: set of search variables

A set of *search variables* transitively functionally defines all other variables.

Example: set of search variables

- The set of car type variables $\{c_1, \ldots c_6\}$ is a set of search variables for RCS.

Set of Search Variables

Definition: set of search variables

A set of *search variables* transitively functionally defines all other variables.

Example: set of search variables

- The set of car type variables $\{c_1, \ldots c_6\}$ is a set of search variables for RCS.
- The set of all variables is a set of search variables.

Approach

1. Before search starts, find a set $\mathcal{S}$ of search variables.

Approach

1. Before search starts, find a set $\mathcal{S}$ of search variables.
   (We are guaranteed to find one as the set of all variables is a set of search variables.)

Approach

1. Before search starts, find a set $\mathcal{S}$ of search variables.
   (We are guaranteed to find one as the set of all variables is a set of search variables.)

2. During each LNS iteration, enforce the freeze set to be a subset of $\mathcal{S}$.

The Dependency Graph

The variables and the functional definitions via dependency constraints induce a directed graph, called the *Dependency Graph*.

The Dependency Graph

The variables and the functional definitions via dependency constraints induce a directed graph, called the *Dependency Graph*.

- Each variable is a vertex in the graph.

The Dependency Graph

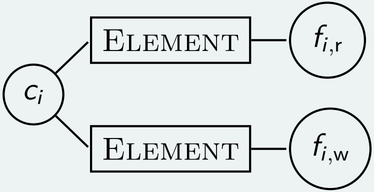
The variables and the functional definitions via dependency constraints induce a directed graph, called the *Dependency Graph*.

- Each variable is a vertex in the graph.

- For each dependency constraint with input variables $\mathcal{X}$ and output variables $\mathcal{Y}$, there is a vertex d in the graph and
  - an arc from each $x \in \mathcal{X}$ to d and
  - an arc from d to each variable $y \in \mathcal{Y}$.

Example: the Dependency Graph

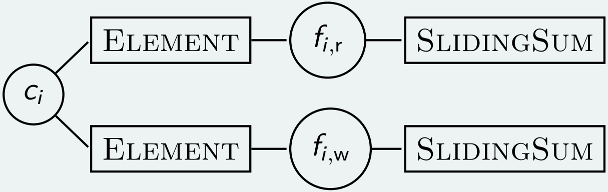
Running example: RCS dependency subgraph



Note that not all variables, arcs, and edges are depicted in the subgraph.

Example: the Dependency Graph

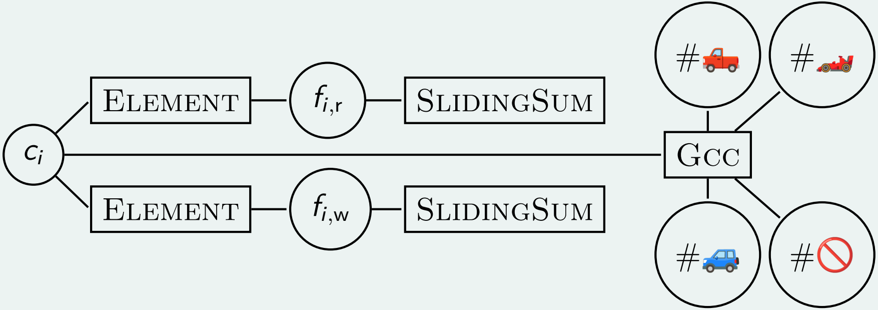
Running example: RCS dependency subgraph



Note that not all variables, arcs, and edges are depicted in the subgraph.

Example: the Dependency Graph

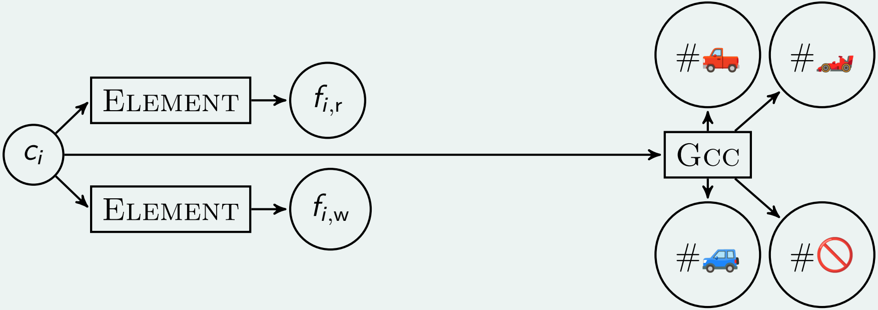
Running example: RCS dependency subgraph



Note that not all variables, arcs, and edges are depicted in the subgraph.

Example: the Dependency Graph

Running example: RCS dependency subgraph



Note that not all variables, arcs, and edges are depicted in the subgraph.

The Dependency Curation Scheme (DCS)

We have developed a *dependency curation scheme* (DCS), that finds a low-cardinality set of search variables.

- If the dependency graph is acyclic, then the set $\mathcal{S}$ of search variables is the set of source variables.

The Dependency Curation Scheme (DCS)

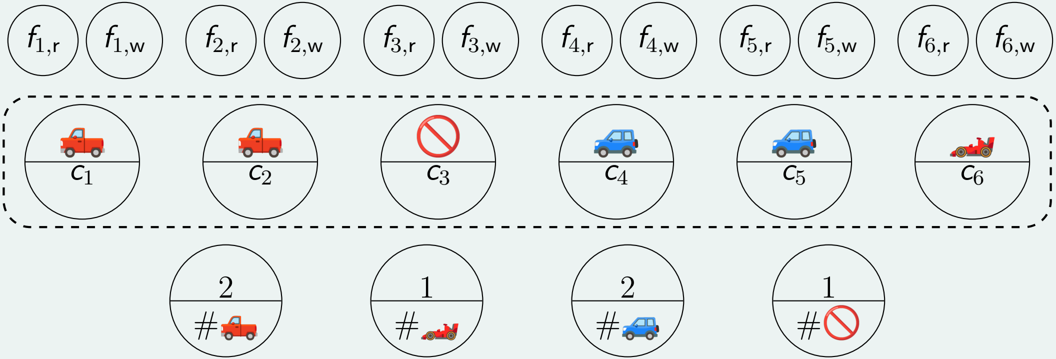
We have developed a *dependency curation scheme* (DCS), that finds a low-cardinality set of search variables.

- If the dependency graph is acyclic, then the set $\mathcal{S}$ of search variables is the set of source variables.
- Otherwise, we find a set of search variables using an algorithm that:

  1. finds the strongly connected components (SCCs) and orders them, before it
  2. constructs $\mathcal{S}$ while it explores the ordered SCCs in a manner similar to depth-first search.

The Dependency Curation Scheme (DCS): Example

Running example: RCS set of search variables



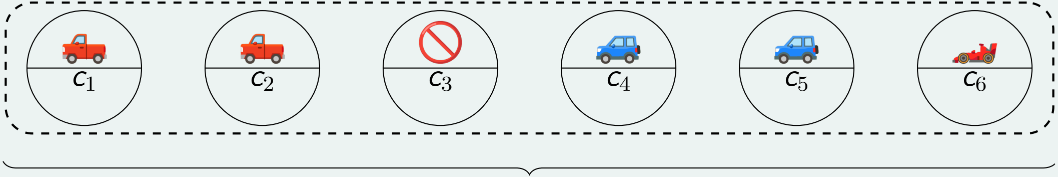
$f_{1,\mathrm{r}}$ $f_{1,\mathrm{w}}$ $f_{2,\mathrm{r}}$ $f_{2,\mathrm{w}}$ $f_{3,\mathrm{r}}$ $f_{3,\mathrm{w}}$ $f_{4,\mathrm{r}}$ $f_{4,\mathrm{w}}$ $f_{5,\mathrm{r}}$ $f_{5,\mathrm{w}}$ $f_{6,\mathrm{r}}$ $f_{6,\mathrm{w}}$

🛻 c_1 🛻 c_2 🚫 c_3 🚙 c_4 🚙 c_5 🏎️ c_6

2 #🛻 1 #🏎️ 2 #🚙 1 #🚫

The Dependency Curation Scheme (DCS): Example

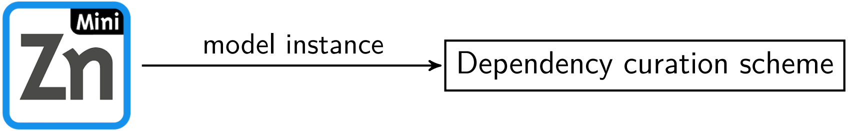
Running example: RCS set of search variables



🧊 freeze set must be a subset of this set

Helicopter View

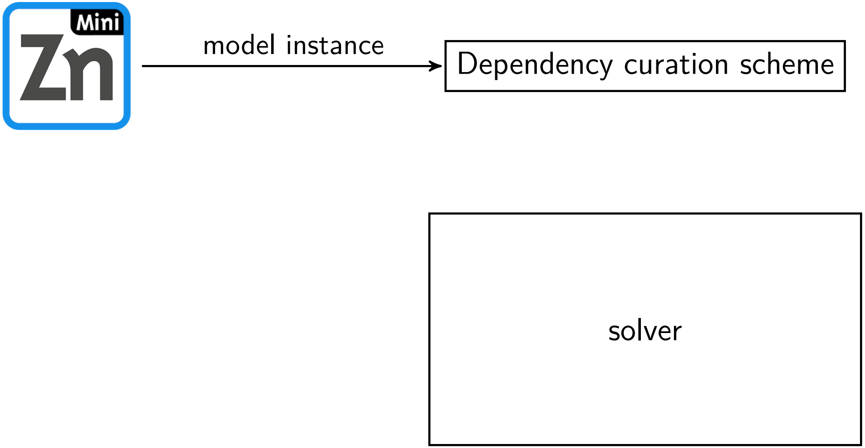
Helicopter View



model instance

Dependency curation scheme

Helicopter View

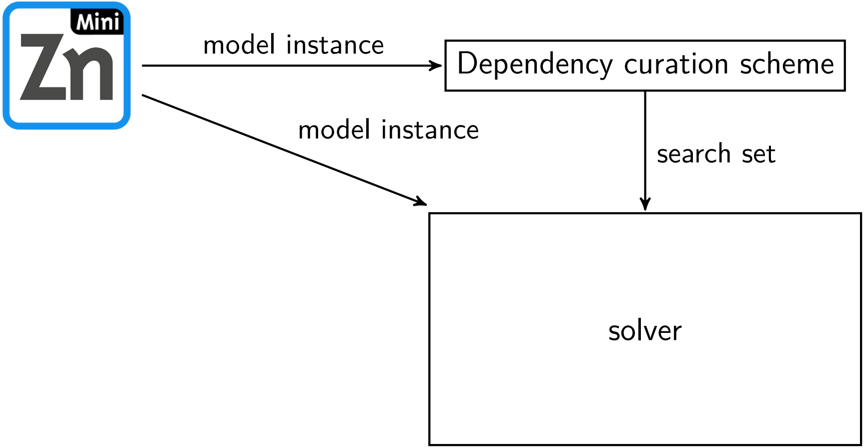


model instance

Dependency curation scheme

solver

Helicopter View



model instance

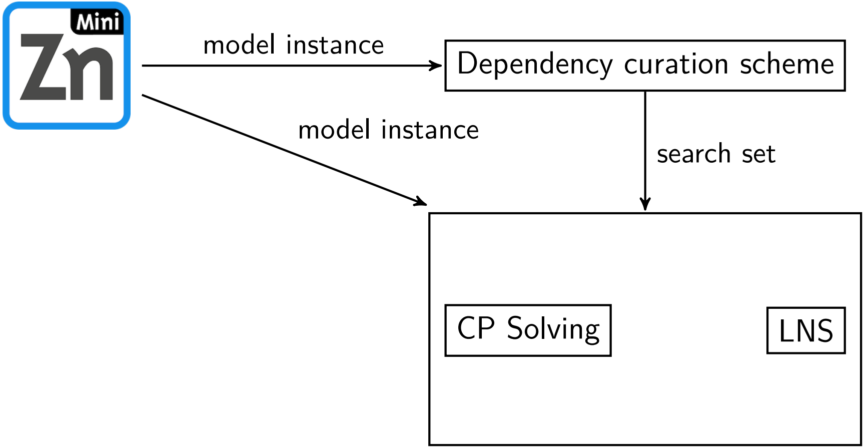
Dependency curation scheme

model instance

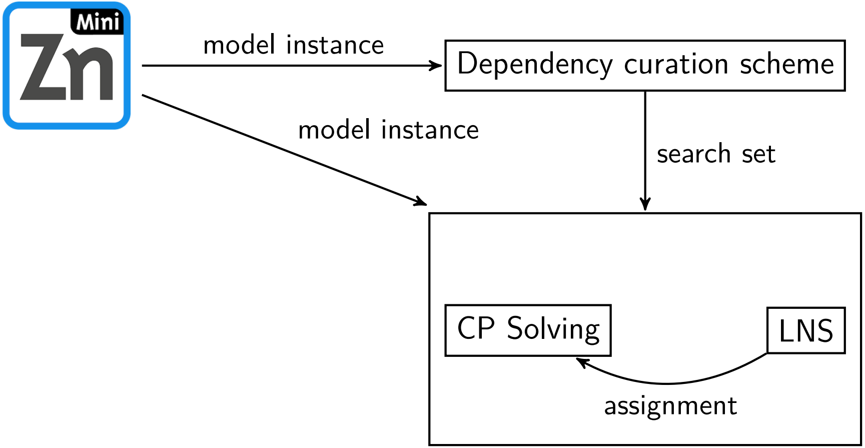
search set

solver

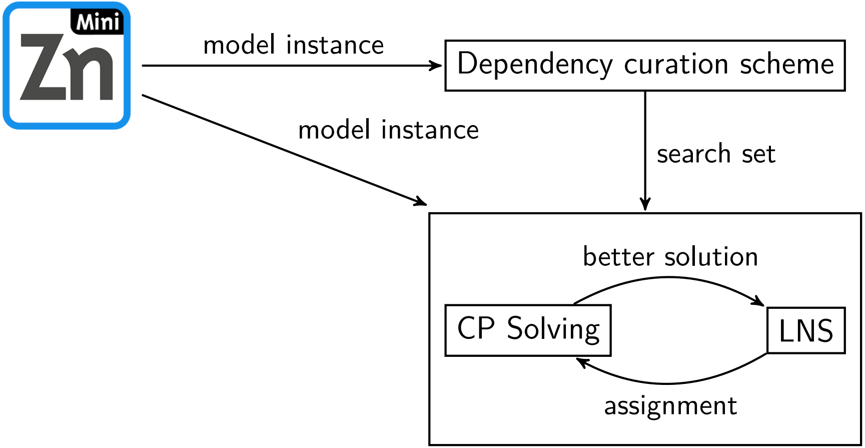
Helicopter View



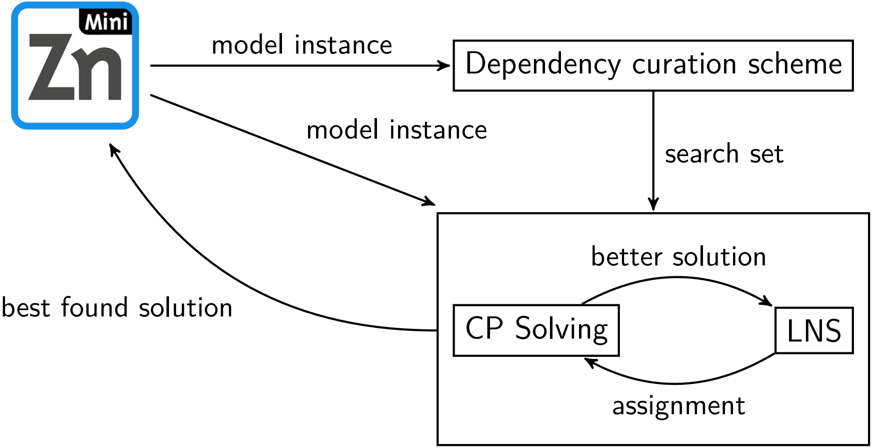
Helicopter View



Helicopter View



Helicopter View



LNS selection heuristics

We selected five generic LNS selection heuristics:

- Randomised LNS;
- Propagation guided LNS (PG-LNS);
- Reverse propagation guided LNS (RPG-LNS);
- Cost impact guided LNS (CIG-LNS); and
- Variable-relationship guided LNS (VRG-LNS).

Randomised LNS

We have a threshold $\phi \in [0.05, 0.95]$ that is updated during search.

Inside every LNS iteration and for each variable x in the set of search variables, we select a uniform random number $p_x \in [0, 1]$.

If $p_x < \phi$, then x is included in the freeze set, else it is excluded.

Remaining Selection Heuristics

Each of the other generic selection heuristics stores data about the search variables.

The data is collected during LNS iterations and is exploited to help guide search.

Method

We extended a Gecode-based portfolio solver, implementing or reimplementing its generic selection heuristics to follow their descriptions from the literature more closely.

Method

We extended a Gecode-based portfolio solver, implementing or reimplementing its generic selection heuristics to follow their descriptions from the literature more closely.

We created or updated existing MiniZinc models for the:

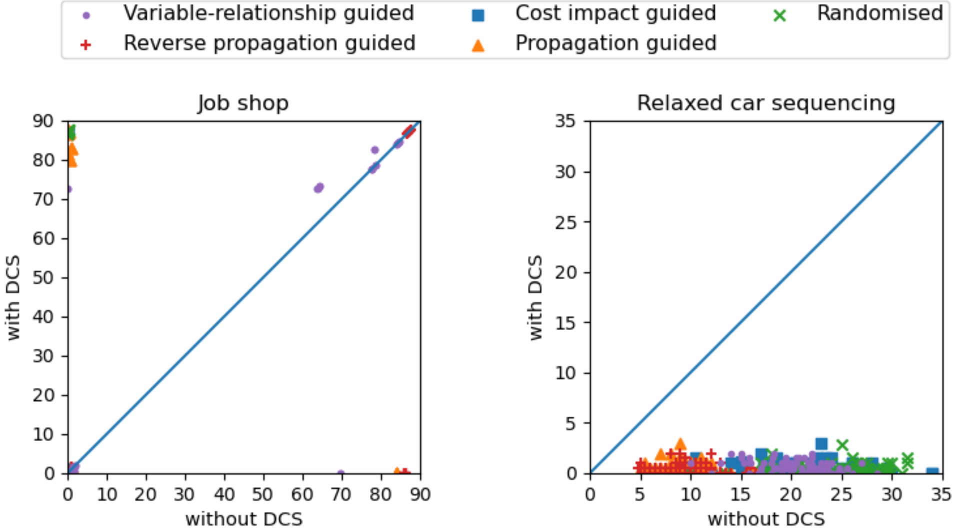
- job shop problem (JSP),
- steel mill slab design (SMSD),
- relaxed car sequencing problem (RCS), and
- travelling salesperson with time windows (TSPTW).

Experiment Setup

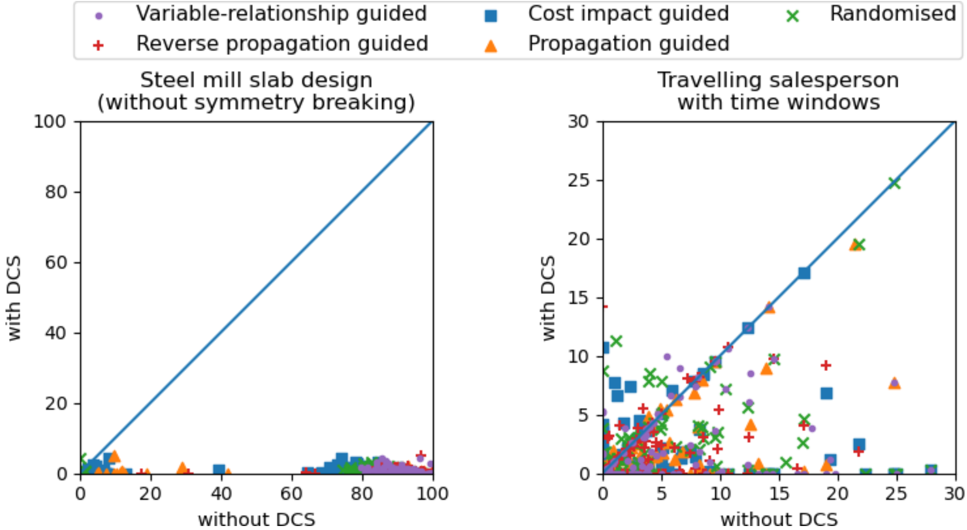
For each selection heuristic, both with DCS and without DCS, and each problem instance, we ran 10 independent runs, each under a timeout of 3 minutes from the same initial incumbent solution.

We used a scoring function where scores range over the interval $[0, 100]$, where a low score is better than a high one.

Results – Graphs



Results – Graphs



Results – Table

	CIG-LNS		PG-LNS		Randomised LNS		RPG-LNS		VRG-LNS	
DCS	no	yes	no	yes	no	yes	no	yes	no	yes
JSP	**0.13**	7.45	7.30	28.53	0.18	18.37	32.68	18.35	32.20	33.19
RCS	22.11	0.47	9.94	0.48	24.72	**0.34**	9.93	0.43	20.35	0.63
SMSD	1.62	0.87	4.06	**0.67**	1.14	0.85	5.64	0.73	6.45	0.81
TSPTW	4.65	1.82	3.67	**1.36**	4.87	2.18	3.71	1.80	5.41	2.20

Conclusion

We have presented our dependency curation scheme (DCS), which can be used with any LNS selection heuristic.

Conclusion

We have presented our dependency curation scheme (DCS), which can be used with any LNS selection heuristic.

We have compared the performance of a generic randomised selection heuristic and state-of-the-art generic selection heuristics, both with and without DCS.

Conclusion

We have presented our dependency curation scheme (DCS), which can be used with any LNS selection heuristic.

We have compared the performance of a generic randomised selection heuristic and state-of-the-art generic selection heuristics, both with and without DCS.

Our experiments show that the performance of using DCS with the generic randomised selection heuristic is (amazingly) competitive.

Thank You

Thank you for your attention.

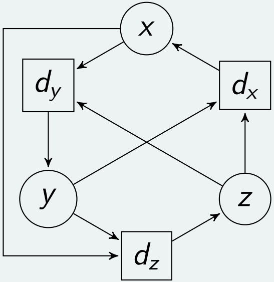
Multi-way Dependency Constraints

Some constraints have multiple input sets and multiple output sets. Each such constraint is represented as multiple vertices in the dependency graph.

Example: multi-way dependency constraint

The constraints $x + y = z$ can be rewritten:

- $z - y = x$ and
- $z - x = y$.



Conditional Dependency Constraints

Some constraints are dependency constraints under specific circumstances.

Example: the table constraint

For each car i, We can replace the element constraints with a table constraint:

$$\texttt{table([}c_i\texttt{, }f_{i,\mathrm{r}}\texttt{, }f_{i,\mathrm{w}}\texttt{],}\quad \begin{array}{l}\texttt{[[🛻, 1, 1],}\\ \texttt{ [🏎️, 1, 0],}\\ \texttt{ [🚙, 0, 1],}\\ \texttt{ [🚫, 0, 0]]}\end{array}\quad\texttt{)}$$

For the parameter matrix, the values in the first column are distinct. Therefore, this table constraint is a dependency constraint via which c_i functionally defines $f_{i,\mathrm{r}}$ and $f_{i,\mathrm{w}}$.