TU WIEN Informatics

BOSCH

kbs Knowledge-Based Systems Group

logics

Adaptive Large-Neighbourhood Search for Optimisation in Answer-Set Programming

Thomas Eiter, Tobias Geibinger, Nelson Higuera Ruiz, Nysret Musliu, Johannes Oetsch, Dave Pfliegler and Daria Stepanova

Motivation

- We study **Answer-Set Programming (ASP)**
- ASP is a popular **declarative problem-solving paradigm**
- **Efficient solves exist**, e.g., **clingo** from **Potassco Solutions**
- Its **flexible nature** makes it attractive for many domains

Parallel Machine Scheduling | Production Planning | Task Scheduling | Product Configuration | Model-Based Testing for Autonomous Driving

Finding high quality solutions for large industrial problems takes a long time

In practice, good solutions should be provided fast!

➡ We provide a framework for Adaptive Large-Neighbourhood Search for ASP which improves optimisation performance

The ALASPO System

JSON Configuration File

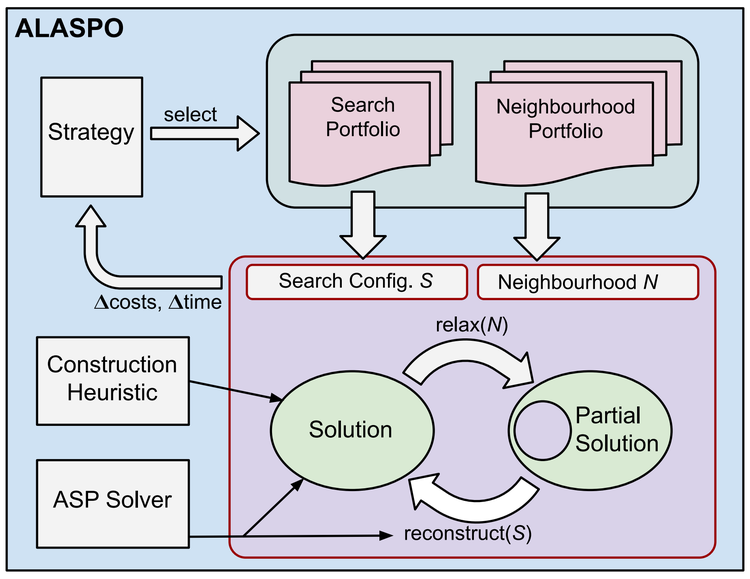
```
{
  "strategy": "uniform-roulette-wheel",
  "relaxOperators": [
    {
      "type": "declarative",
      "rates": [ 0.2, 0.4, 0.6, 0.8 ],
    } ],
  "searchOperators": [
    {
      "type": "default",
      "timeouts": [ 5, 15, 30 ],
      "configuration": {}
    } ]
}
```

ASP Encoding

```
player(1..g*p). group(1..g). week(1..w).

{ plays(P,W,G) : group(G) } = 1 :- player(P), week(W).
{ plays(P,W,G) : player(P) } = p :- week(W), group(G).
meets(P1,P2,W):- plays(P1,W,G),
                 plays(P2,W,G), P1<P2.
:~ #count { W : meets(P1,P2,W) } > 1, player(P1),
      player(P2), P1 < P2. [1,P1]
#show plays/3.

_lns_select(W) :- week(W).
_lns_fix(plays(P,W,G),W) :- _lns_select(W), plays(P,W,G).
```

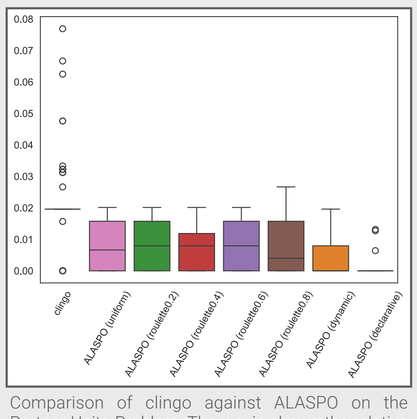


Adaptive Large-Neighbourhood Search: We change the operators during the run depending on the selected strategy

Experiments on Industrial Applications

Partner Units Problem

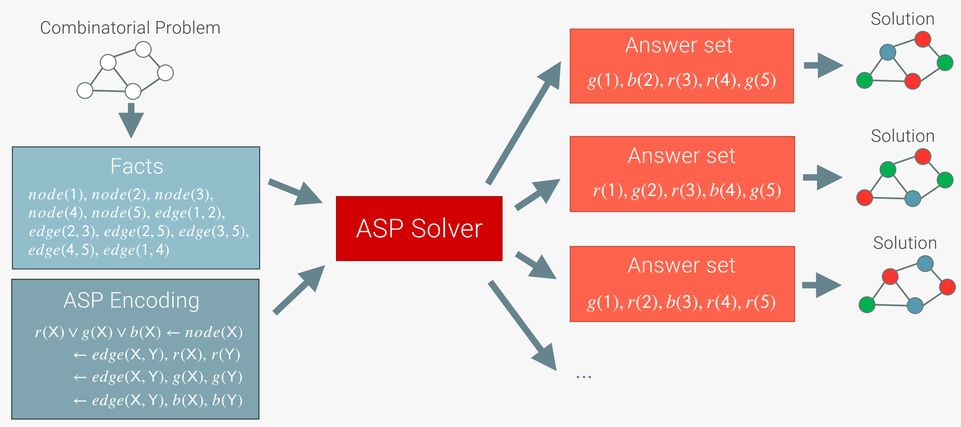
- Challenging **real-life configuration problem**
- It requires to **group sensors into zones** and **connecting them to control units**
- Control units need to be connected to a limited number of other units such that **all communication requirements are fulfilled**
- The objective is to **minimise the number of control units that are used**
- We ran clingo against ALASPO for **5 min** on a benchmark of **78 instances**
- Using a tailored neighbourhood, **ALASPO beats clingo**



Comparison of clingo against ALASPO on the Partner Units Problem. The x-axis shows the relative difference to the best found solution for that instance.

Answer-Set Programming

Methodology



Syntax

ASP programs are finite sets of rules:

default negation

$$a_1 \vee \ldots \vee a_k \leftarrow b_{k+1}, \ldots, b_m, not\ b_{m+1}, \ldots, not\ b_n$$

$a_1, \ldots, a_k$, $b_{k+1}, \ldots, b_m$ and $b_{m+1}, \ldots, b_n$ are **atoms**

Atoms can be **ground** or have **variables**:

instantiated during **grounding**

p $\qquad$ $color(\mathtt{c})$

An **interpretation** I is a set of ground atoms, which **satisfies** a rule if:

whenever $b_{k+1}, \ldots, b_m \in I$ and $b_{m+1}, \ldots, b_n \notin I$, then $a_i \in I$ for some $(1 \leq i \leq k)$

Semantics

I is an **answer set** of program P if it is a **minimal model** of the Gelfond-Lifschitz reduct

$$P^I := \{\ a_1 \vee \ldots \vee a_k \leftarrow b_{k+1}, \ldots, b_m \mid a_1 \vee \ldots \vee a_k \leftarrow b_{k+1}, \ldots, b_m, not\ b_{m+1}, \ldots, not\ b_n \in P, b_{m+1}, \ldots, b_n \notin I\ \}$$

Intuition: Assuming everything not in I is **false**, the rest is **stable** w.r.t. P

Shift Design

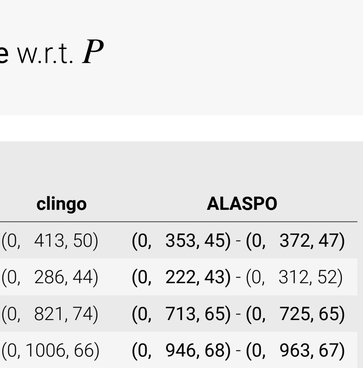
- The goal is to **align shifts** so that **over- and understaffing** of workers **is avoided**
- We used a run time of **60 minutes** on a benchmark of **8 instances**
- ALASPO finds **better bounds on all instances**

	clingo	ALASPO
3-04	(0, 413, 50)	**(0, 353, 45)** - **(0, 372, 47)**
3-06	(0, 286, 44)	**(0, 222, 43)** - (0, 312, 52)
3-11	(0, 821, 74)	**(0, 713, 65)** - **(0, 725, 65)**
3-20	(0, 1006, 66)	**(0, 946, 68)** - **(0, 963, 67)**
3-26	(0, 1061, 77)	**(0, 1037, 78)** - (0, 1078, 75)
3-27	(0, 393, 25)	**(0, 376, 24)** - **(0, 393, 24)**
3-29	(0, 509, 67)	**(0, 465, 59)** - **(0, 470, 63)**
4-02	(0, 466, 50)	**(0, 388, 39)** - **(0, 401, 54)**

Comparison of clingo against ALASPO on the Shift Design Problem. The numbers are the three components of the lexicographic objective (lower is better).

Parallel Machine Scheduling

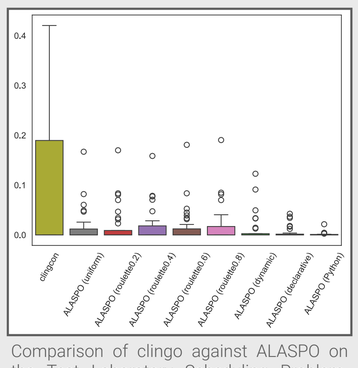
- We have to **assign jobs to machines** such that capability constraints are upheld
- The objective is to **minimise makespan** while respecting release dates and setup times
- We used a run time of **15 minutes** on a benchmark of **500 instances**
- ALASPO with a **greedy initial solution performs best**



Comparison of clingo against ALASPO on the Parallel Machine Scheduling Problem. The x-axis shows the relative difference to the best found solution for that instance.

Test Laboratory Scheduling

- Complex **real-world project scheduling problem** with novel constraints and objectives
- We used a run time of **30 minutes** on a benchmark of **33 instances**
- **Best results** are achieved by **ALASPO using a relax operator written in Python**



Comparison of clingo against ALASPO on the Test Laboratory Scheduling Problem. The x-axis shows the relative difference to the best found solution for that instance.