Answer Set Programming for Scheduling at Bosch

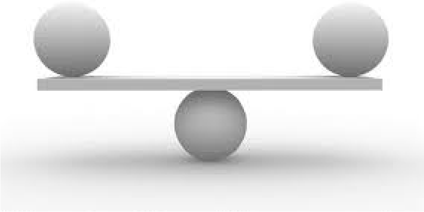
T. Eiter[1], T. Geibinger[1], N. Musliu[1], J. Oetsch[1], P. Skocovsky[1,2], D. Stepanova[2]

[1]Vienna University of Technology (TU Wien)
[2]Bosch center for AI, CR/PJ-AI-R26

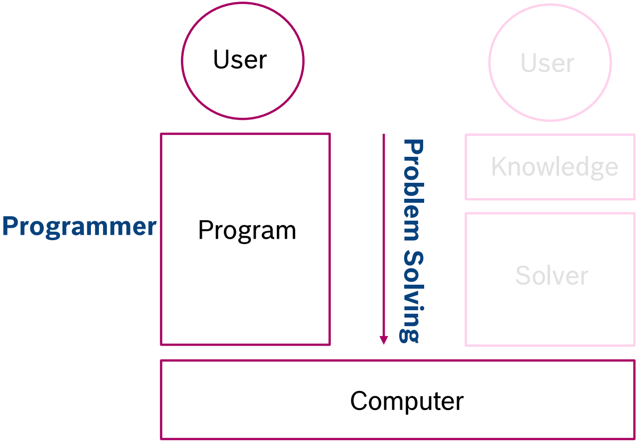
TU WIEN BOSCH

Pain Point 1: Solution Robustness

- Independence
  - Depending on commercial software is problematic
    - Additional costs
    - Issues if the company is sold, disappears, changes contracting conditions...
- Reliability
  - Software should be
    - Efficient
    - Maintainable
    - Ideally, optimality guarantees

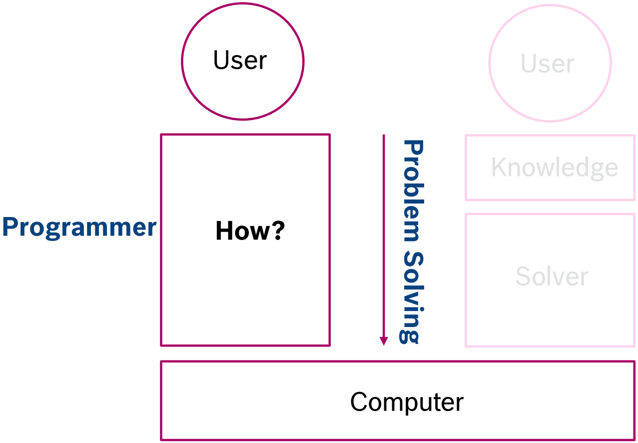


Traditional VS Knowledge-driven Problem Solving



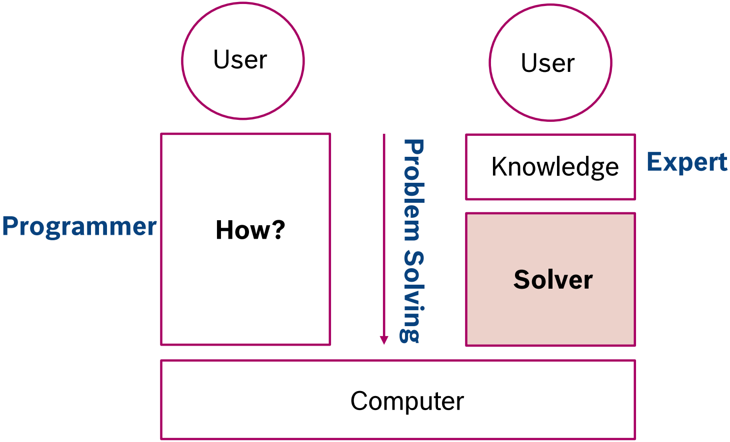
Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Traditional VS Knowledge-driven Problem Solving

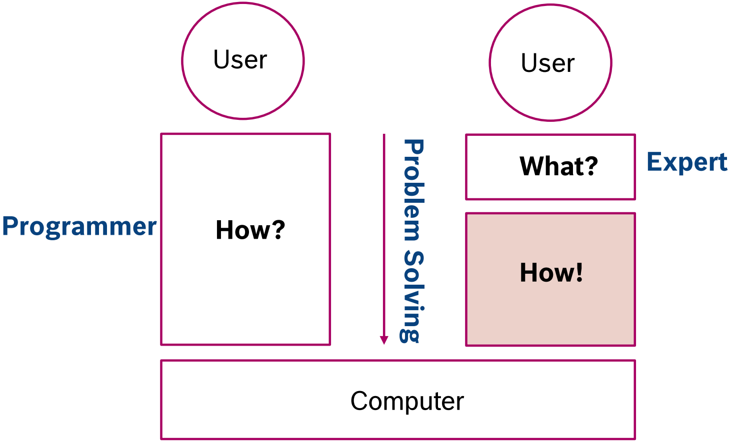


Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Traditional VS Knowledge-driven Problem Solving



Traditional VS Knowledge-driven Problem Solving



Knowledge-driven Problem Solving

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

+ Generality
+ Efficiency
+ Optimality
+ Availability

Knowledge

Expert

Solver

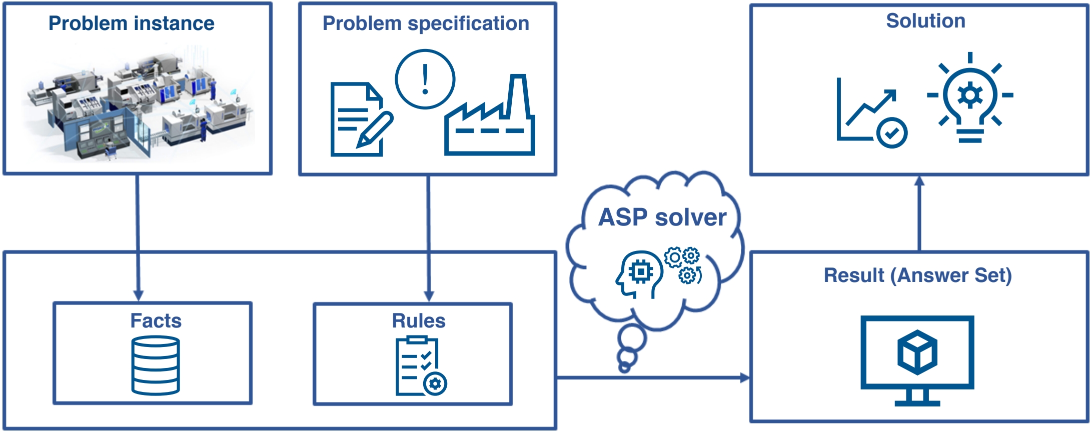
Knowledge-driven Problem Solving

+ Transparency
+ Flexibility
+ Maintainability
+ Reliability

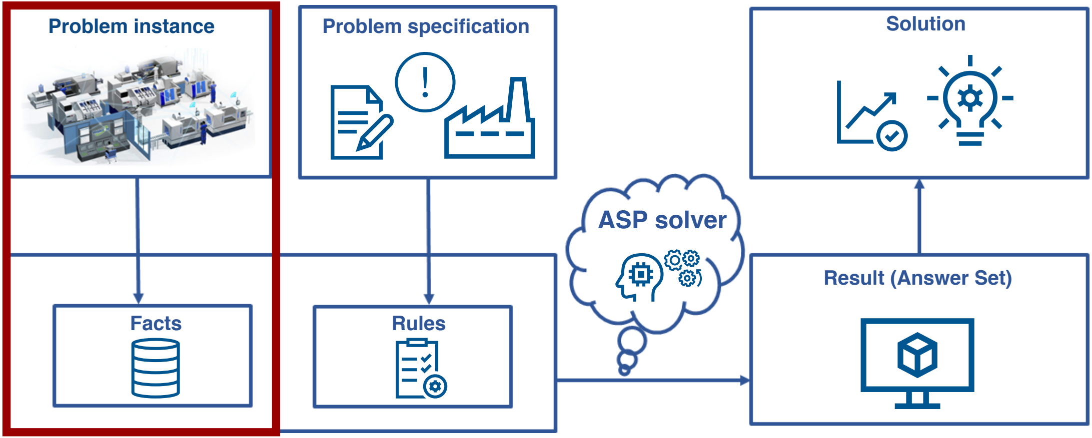
+ Generality
+ Efficiency
+ Optimality
+ Availability

Knowledge	**Expert**
Solver	

Answer Set Programming



Facts



Problem instance
Problem specification
Solution
ASP solver
Facts
Rules
Result (Answer Set)

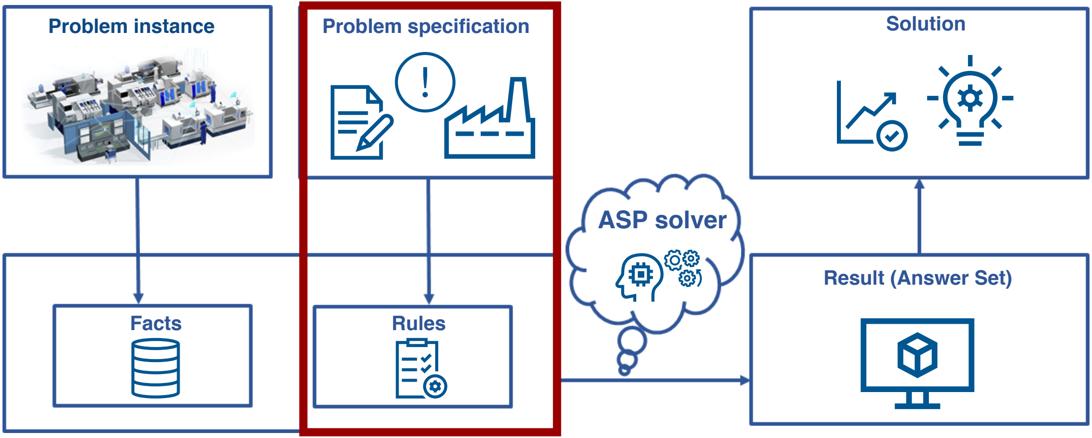
Facts

Example

Machine **m1** is capable of processing the job **j6** with impact of type **“r”**, whose duration is **50** minutes, due time is in **180** minutes and whose weight on machine **m1** is **67**.

- `machine(m1)`
- `job(j6)`
- `impact(j6, r)`
- ...
- `weight(j6, m1, 67)`

Rules


Problem instance
Problem specification
Solution
ASP solver
Facts
Rules
Result (Answer Set)

Rules

Example

Job must be assigned to a single machine that is capable of performing the job.

- `1 { assigned(J,M) : machine(M), capable(M,J) } 1 :- job(J).`

Rules

Example

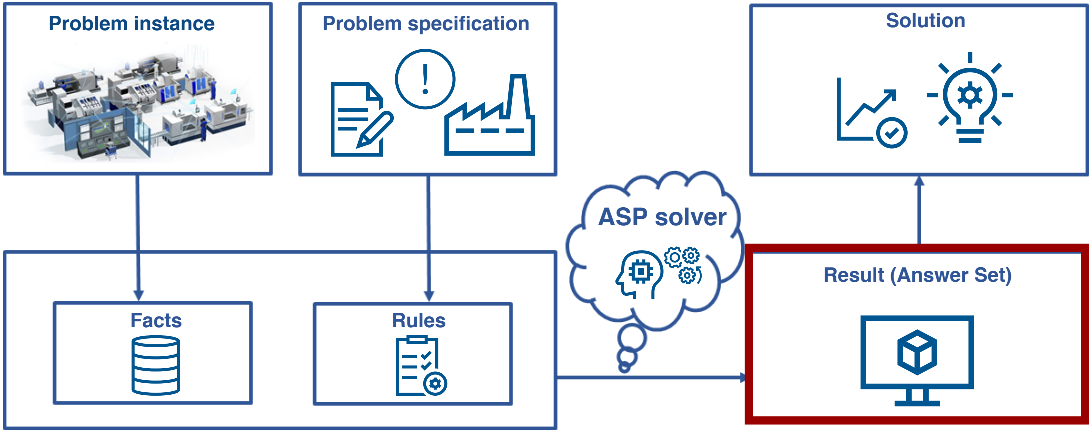
Job must be assigned to a single machine that is capable of performing the job.

- `1 { assigned(J,M) : machine(M), capable(M,J) } 1 :- job(J).`

A job has to start after its release time.

- `&diff{ 0 - start(J) } <= -R :- job(J), assigned(J,M), release(J,M,R).`

ASP Output (Answer Set)

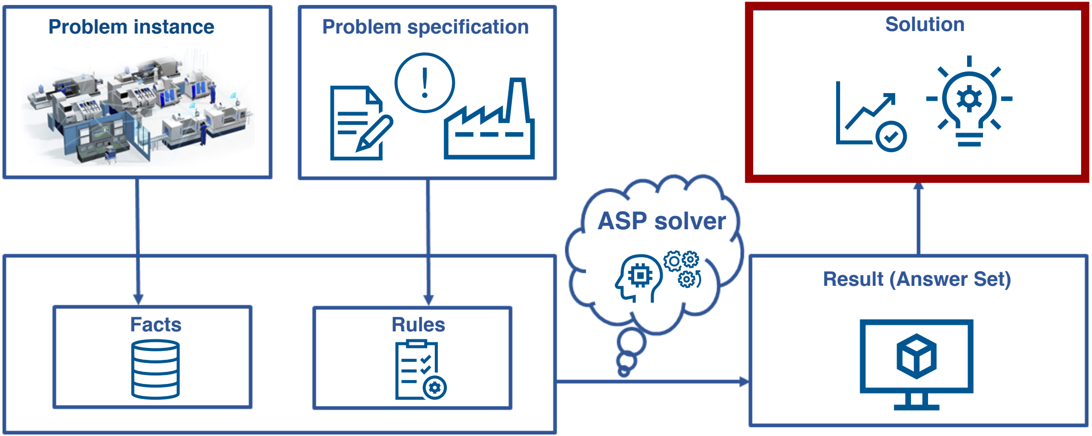


ASP Output (Answer Set)

Example

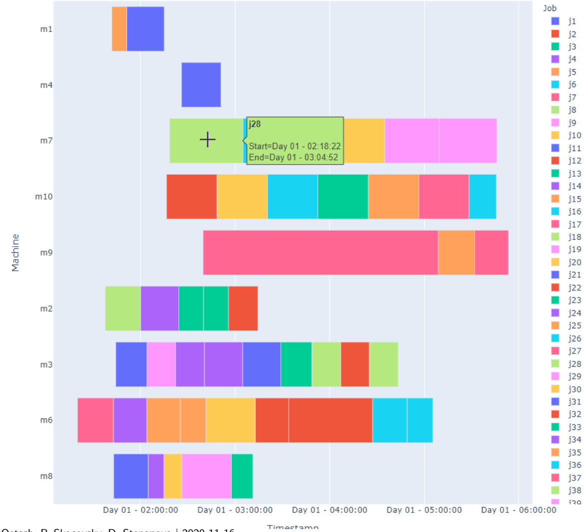
```
$ clingo-dl -heuristic=domain -minimize-variable=makespan asp_encoding.lp
Optimization: 115835
⋮
Optimization: 93707
Answer: 11
assigned(j1, m6), ..., assigned(j98,m5), assigned(j99,m14)

OPTIMUM FOUND
Models : 11
Time : 6.68s (Solving: 3.03s 1st Model: 3.33s Unsat: 0.02s)
CPU Time : 5.718s
```

Solution



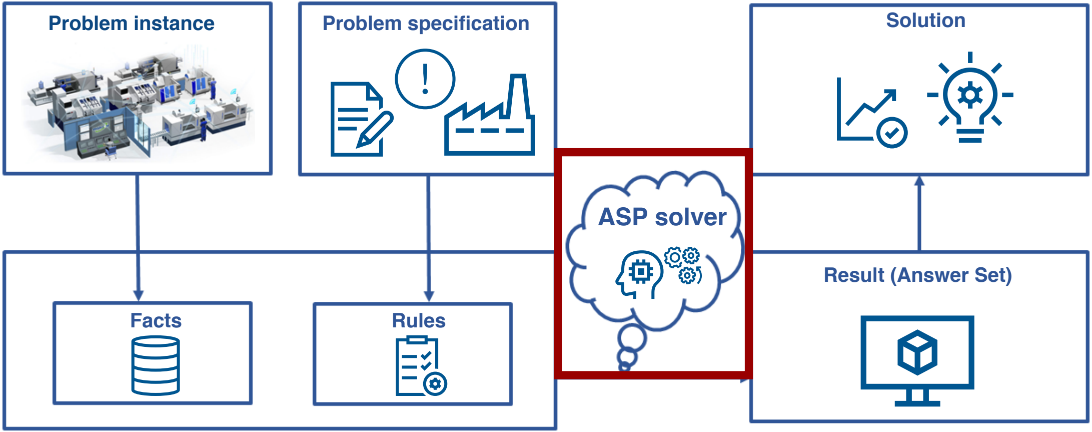
Problem instance
Problem specification
Solution
ASP solver
Facts
Rules
Result (Answer Set)

Solution



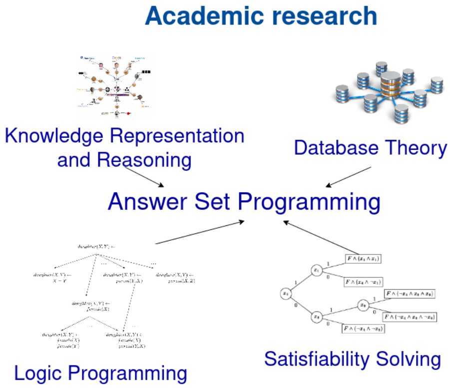
Job
j1
j2
j3
j4
j5
j6
j7
j8
j9
j10
j11
j12
j13
j14
j15
j16
j17
j18
j19
j20
j21
j22
j23
j24
j25
j26
j27
j28
j29
j30
j31
j32
j33
j34
j35
j36
j37
j38
m1
m4
m7
m10
m9
m2
m3
m6
m8
Machine
j28
Start=Day 01 - 02:18:22
End=Day 01 - 03:04:52
Day 01 - 02:00:00
Day 01 - 03:00:00
Day 01 - 04:00:00
Day 01 - 05:00:00
Day 01 - 06:00:00

ASP Solvers



 Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

ASP: Research, Solvers and Applications



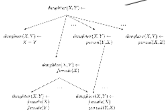
ASP: Research, Solvers and Applications

Academic research

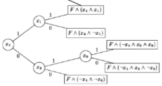
Knowledge Representation and Reasoning

Database Theory

Answer Set Programming



Logic Programming



Satisfiability Solving

ASP solvers

Clingo

-Open source (MIT license)
-Well-supported

Potassco

dlvsystem
SPIN-OFF
OF UNIVERSITY OF CALABRIA

 Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

TU WIEN BOSCH

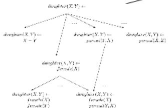
ASP: Research, Solvers and Applications

Academic research

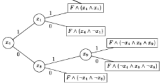
Knowledge Representation and Reasoning

Database Theory

Answer Set Programming



Logic Programming



Satisfiability Solving

ASP solvers

Clingo

-Open source (MIT license)
-Well-supported

Potassco

dlvsystem
SPIN-OFF OF UNIVERSITY OF CALABRIA

Industrial applications

SIEMENS
Ingenuity for life

Railroad interlocking configurators

Mercedes-Benz

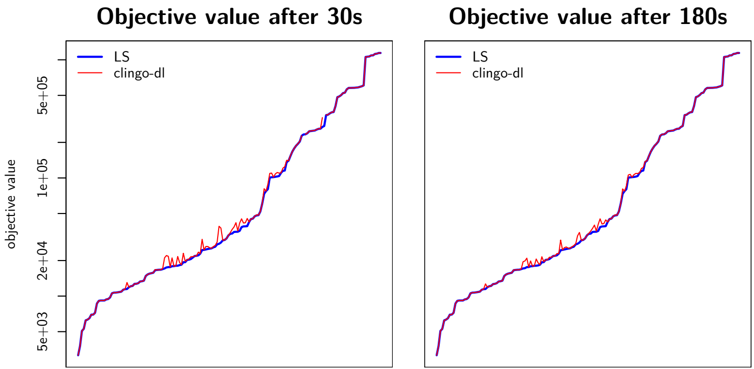
Routing driverless transport vehicles in car assembly

SBB CFF FFS

Train scheduling for Swiss railroad

Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Comparison to Local Solver on Implantation Rtp Problem

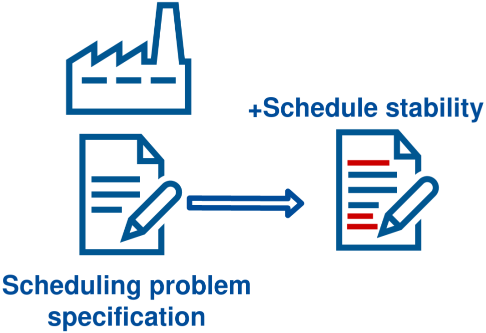


Objective values of local solver and clingo-dl achieved on 162 real snapshots

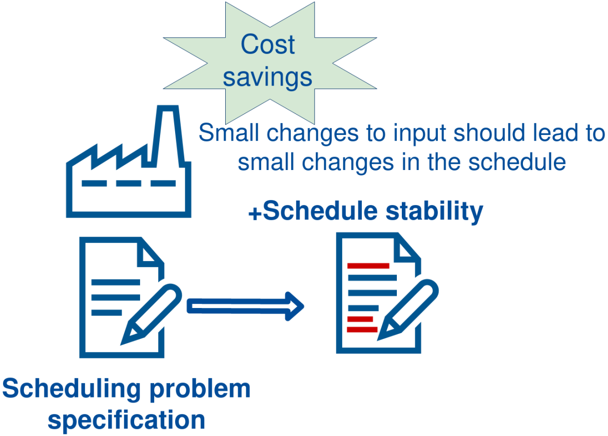
Pain Point 2: Solution Adaptability

Scheduling problem specification

Pain Point 2: Solution Adaptability

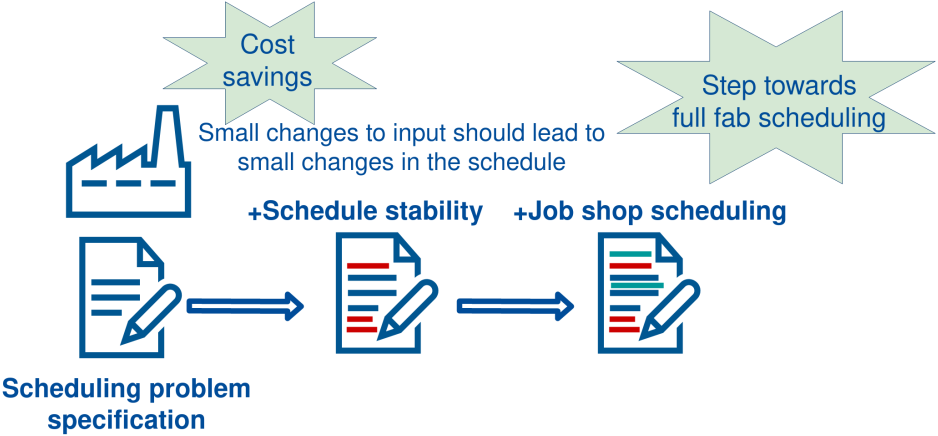


Pain Point 2: Solution Adaptability

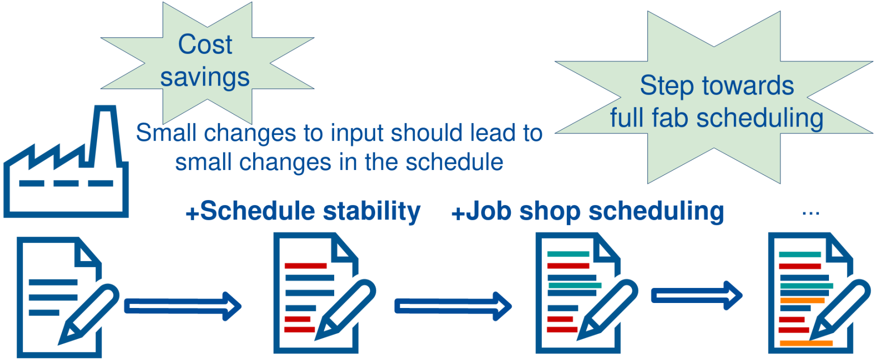


…

Pain Point 2: Solution Adaptability



Pain Point 2: Solution Adaptability



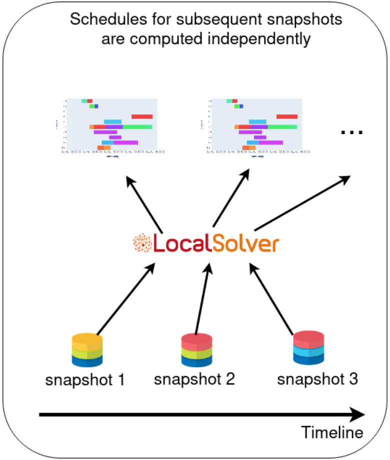
Scheduling problem specification

Problem: New constraints require significant changes in the solution leading to extra costs

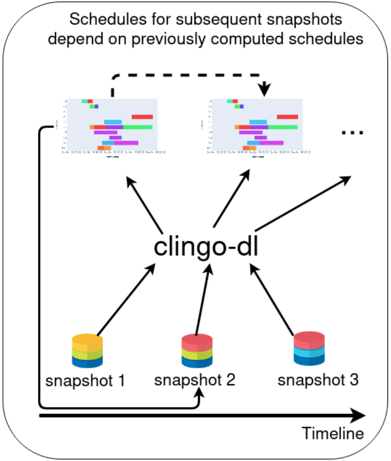
Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Schedule Stability

Current solution



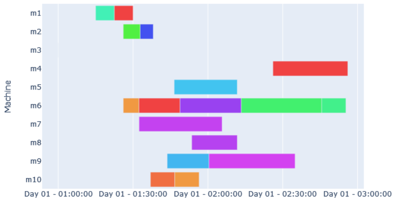
ASP-based solution

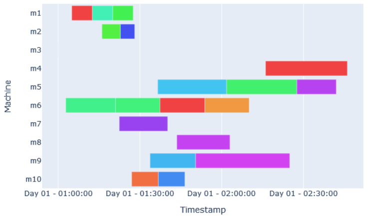


 Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

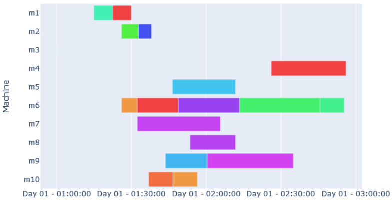
Schedule Stability

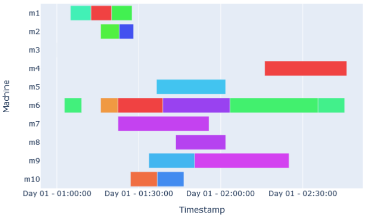
Current solution (unstable)





ASP-based solution (stable)





Job Shop Scheduling

Example

ASP scheduling encoding

```
before(J1,J2,M) | before(J2,J1,M) :- assigned(J1,M), assigned(J2,M), J1 < J2.
:- before(J1,J2,M), before(J2,J3,M), not before(J1,J3,M).
first(J1,M) :- assigned(J1,M), #count { J2 : before(J2,J1,M) } = 0.
1 { next(J1,J2,M) : before(J1,J2,M) } 1 :- assigned(J2,M), not first(J2,M).
:- next(J1,J2,M), before(J1,J3,M), before(J3,J2,M).
1 { assigned(J,M) : machine(M), capable(M,J) } 1 :- job(J).
earliest(J,M,T) :- job(J), machine(M),
                  T = #max { 0 ; R : release(J,M,R) ;
                  F : lastJobFinished(M,F) ;
                  A : availableFrom(M,A) }.
&diff{ 0 - compl(J) } <= -(T+D) :- assigned(J,M),
                                   duration(J,M,D),
                                   earliest(J,M,T).
&diff{ 0 - compl(J1) } <= -(T+D+S) :- first(J1,M), duration(J1,M,D),
                                      histJob(M,J2), earliest(J1,M,T),
                                      setupTime(M,J2,J1,S).
&diff{ compl(J2) - compl(J1) } <= -(D+S) :- before(J2,J1,M),
                                            next(J3,J1,M),
                                            setupTime(M,J3,J1,S),
                                            duration(J1,M,D).
&diff{ 0 - compl(J1) } <= -(T+D+S) :- next(J2,J1,M),
                                      setupTime(M,J2,J1,S),
                                      duration(J1,M,D),
                                      earliest(J1,M,T).
&diff{ compl(J) - 0 } <=  T :- assigned(J,M), availableTo(M,T).
:- next(J2,J1,M), forbidden(J2,J1).
:- first(J1,M), histJob(M,J2), forbidden(J2,J1).
&diff{ compl(J) - makespan } <= 0 :- job(J).
&diff{ 0 - makespan } <= -F :- machine(M), lastJobFinished(M,F).
```

ASP job shop scheduling encoding

```
operation(J,1..N) :- job(J,N).
before(J1,J2,M) | before(J2,J1,M) :- assigned(J1,M), assigned(J2,M), J1 < J2.
:- before(J1,J2,M), before(J2,J3,M), not before(J1,J3,M).
first(J1,M) :- assigned(J1,M), #count { J2 : before(J2,J1,M) } = 0.
1 { next(J1,J2,M) : before(J1,J2,M) } 1 :- assigned(J2,M), not first(J2,M).
:- next(J1,J2,M), before(J1,J3,M), before(J3,J2,M).
1 { assigned(~~J~~(J,O),M) : machine(M), capable(M,J) } 1 :- ~~job(J)~~operation(J,O).
earliest(J,M,T) :- ~~job(J)~~job(J,_), machine(M),
                  T = #max { 0 ; R : release(J,M,R) ;
                  F : lastJobFinished(M,F) ;
                  A : availableFrom(M,A) }.
&diff{ 0 - compl(J,O) } <= -(T+D) :- assigned(~~J~~(J,O),M),
                                     duration(~~J~~(J,O),M,D),
                                     earliest(J,M,T).
&diff{ 0 - compl(J1,O1) } <= -(T+D+S) :- first(J1,M), duration(~~J1~~(J1,O1),M,D),
                                         histJob(M,J2), earliest(J1,M,T),
                                         setupTime(M,J2,J1,S).
&diff{ compl(J2,O2) - compl(J1,O1) } <= -(D+S) :- before(~~J2~~(J2,O2),~~J1~~(J1,O1),M),
                                                  next(~~J3~~(J3,O3),~~J1~~(J1,O1),M),
                                                  setupTime(M,J3,J1,S),
                                                  duration(~~J1~~(J1,O1),M,D).
&diff{ 0 - compl(J1,O1) } <= -(T+D+S) :- next(~~J2~~(J2,O2),~~J1~~(J1,O1),M),
                                         setupTime(M,J2,J1,S),
                                         duration(~~J1~~(J1,O1),M,D),
                                         earliest(J1,M,T).
&diff{ compl(J,O) - 0 } <=  T :- assigned(~~J~~(J,O),M), availableTo(M,T).
:- next(~~J2~~(J2,_),~~J1~~(J1,_),M), forbidden(J2,J1).
:- first(~~J1~~(J1,_),M), histJob(M,J2), forbidden(J2,J1).
&diff{ compl(J,O) - makespan } <= 0 :- ~~job(J)~~operation(J,O).
&diff{ 0 - makespan } <= -F :- machine(M), lastJobFinished(M,F).
&diff{ (J,O1) - (J,O2) } <= -D :- assigned((J,O1),M), duration((J,O1),M,D),
                                  operation(J,O2), O2 = O1+1.
```

Heuristics

Example

Scenario:

- Weighted shortest processing time (wpts) heuristic:
  schedule job `j` before `k` if `d_j/w_j < d_k/w_k`, where
  - `d_j` is duration of `j`
  - `w_j` is weight if `j`

ASP encoding of wpts heuristic to guide search:

```
wspt(J,R) :- duration(J,D), weight(J,W), R = D*1000 / W.
#heuristic next(J,K) : wspt(J,RJ), wspt(K,RK), RJ > RK. [-1,sign]
```

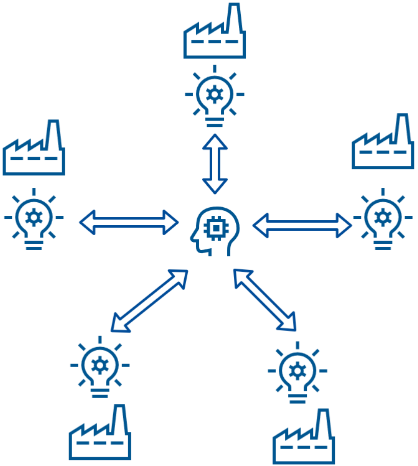

Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Summary

- Pain point: solution robustness
  - Clingo-dl: years of research, open source
  - ASP encoding for implantation work center
  - Performance on par with local solver
- Pain point: solution adaptability
  - Computation of stable schedules
  - Extension to job shop

Summary

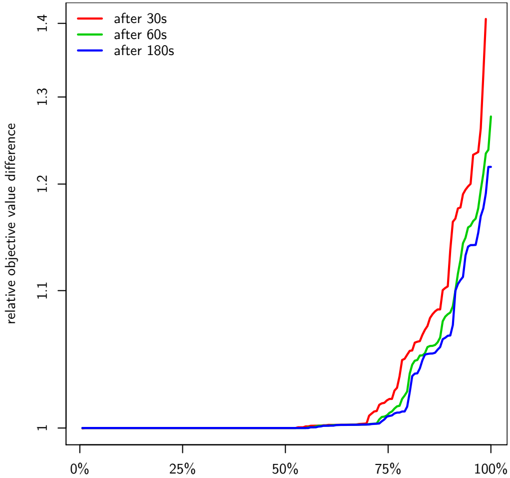
- Pain point: solution robustness
  - Clingo-dl: years of research, open source
  - ASP encoding for implantation work center
  - Performance on par with local solver
- Pain point: solution adaptability
  - Computation of stable schedules
  - Extension to job shop
- Further steps:
  - Realization of various optimization functions
  - Full fab scheduling
  - Learn heuristics using ML (e.g., re-opt, R34), encode as ASP rules



Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Distribution of Relative Objective Value Difference

- Open source ASP solver clingo-dl performs on par with Local Solver



Job Shop Scheduling

Example (ASP scheduling job shop encoding)

```
% Every job comprises of N operations: 1..N
operation(J,1..N) :- job(J,N).
```

Job Shop Scheduling

Example (ASP scheduling job shop encoding)

```
% Every job comprises of N operations: 1..N
operation(J,1..N) :- job(J,N).
```

```
% Every job must be assigned to exactly one machine capable of performing
the job
1{assigned(J,M):machine(M), capable(M,J)}1 :- job(J).
```


Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Job Shop Scheduling

Example (ASP scheduling job shop encoding)

% Every job comprises of N operations: 1..N

operation(J,1..N) :- job(J,N).

% Every operation of every job must be assigned to exactly one machine capable of performing the job

1{assigned(~~J~~(J,O),M):machine(M), capable(M,J)}1 :- ~~job(J)~~operation(J,O).

Job Shop Scheduling

Example (ASP scheduling job shop encoding)

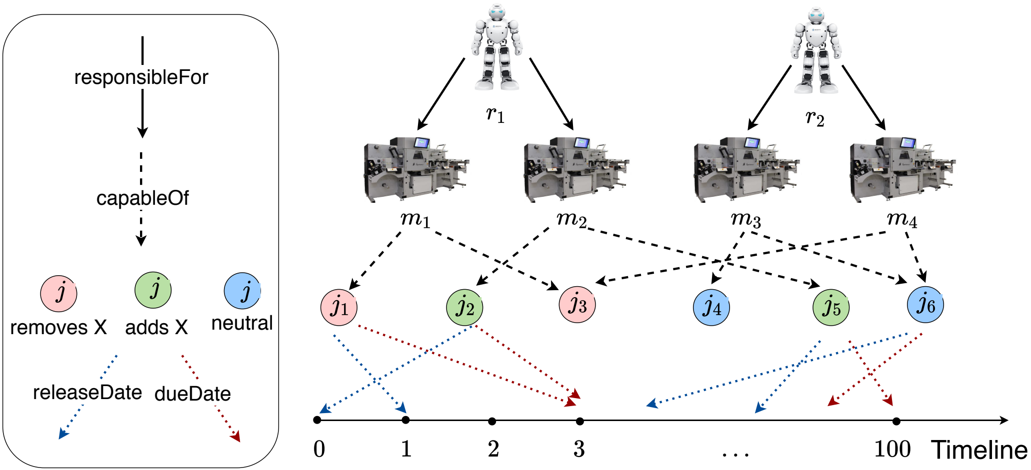
```
% Every job comprises of N operations: 1..N
operation(J,1..N) :- job(J,N).
```

```
% Every operation of every job must be assigned to exactly one machine
capable of performing the job
1{assigned(J(J,O),M):machine(M), capable(M,J)}1 :- job(J)operation(J,O).
```

```
% Operations in a job must not overlap
&diff{ (J,O1) - (J,O2) } <= -D :- assigned((J,O1),M), duration((J,O1),M,D),
                                   operation(J,O2), O2 = O1+1.
```


Internal | T. Eiter, T. Geibinger, N. Musliu, J. Oetsch, P. Skocovsky, D. Stepanova | 2020-11-16

Seminconductor Production at Reutlingen



Scheduling Constraints

- Each job has:
  - machine-specific duration
  - machine-specific weight
- Each machine has:
  - availability time period
  - limit on available time capacity per job type
  - setup times: time needed to switch from job j_i to j_k
  - forbidden sequences of jobs
- Each robot has
  - limit on the number of jobs that can be assigned to machines served by the robot
- Objectives:
  - minimize the maximum of the completion time of the last job performed on a machine
  - mininimize total weighted completion time

Comparison to CPLEX

- ASP is strictly more expressive than CPLEX
- ASP is more transparent, convenient to use, programs are easy to extend/modify
- Some implementations of ASP solvers even use CPLEX at the backend
- The implementation that we are working on is more efficient[1].

---

[1] Tomi Janhunen, Roland Kaminski, Max Ostrowski, Sebastian Schellhorn, Philipp Wanko, Torsten Schaub: Clingo goes linear constraints over reals and integers. Theory and Practice of Logic Programming, 17(5-6): 872-888 (2017)