Automation of Laser Ranging Systems Session

Towards optimal pass scheduling for SLR

Jens Steinborn, André Kloth

20th International Workshop on Laser Ranging
Potsdam, October 9-14, 2016

Motivation

- With some exceptions SLR systems are mostly operated manually
- Higher level of system automation is of increasing interest
- Ideas and technologies from others fields should be evaluated
- At the current stage we try to get funding for a feasibility study

- Scheduling is a core task at every SLR station
- Fast kHz system can do heavy interleaving of passes
- Great potential for automation and optimisation

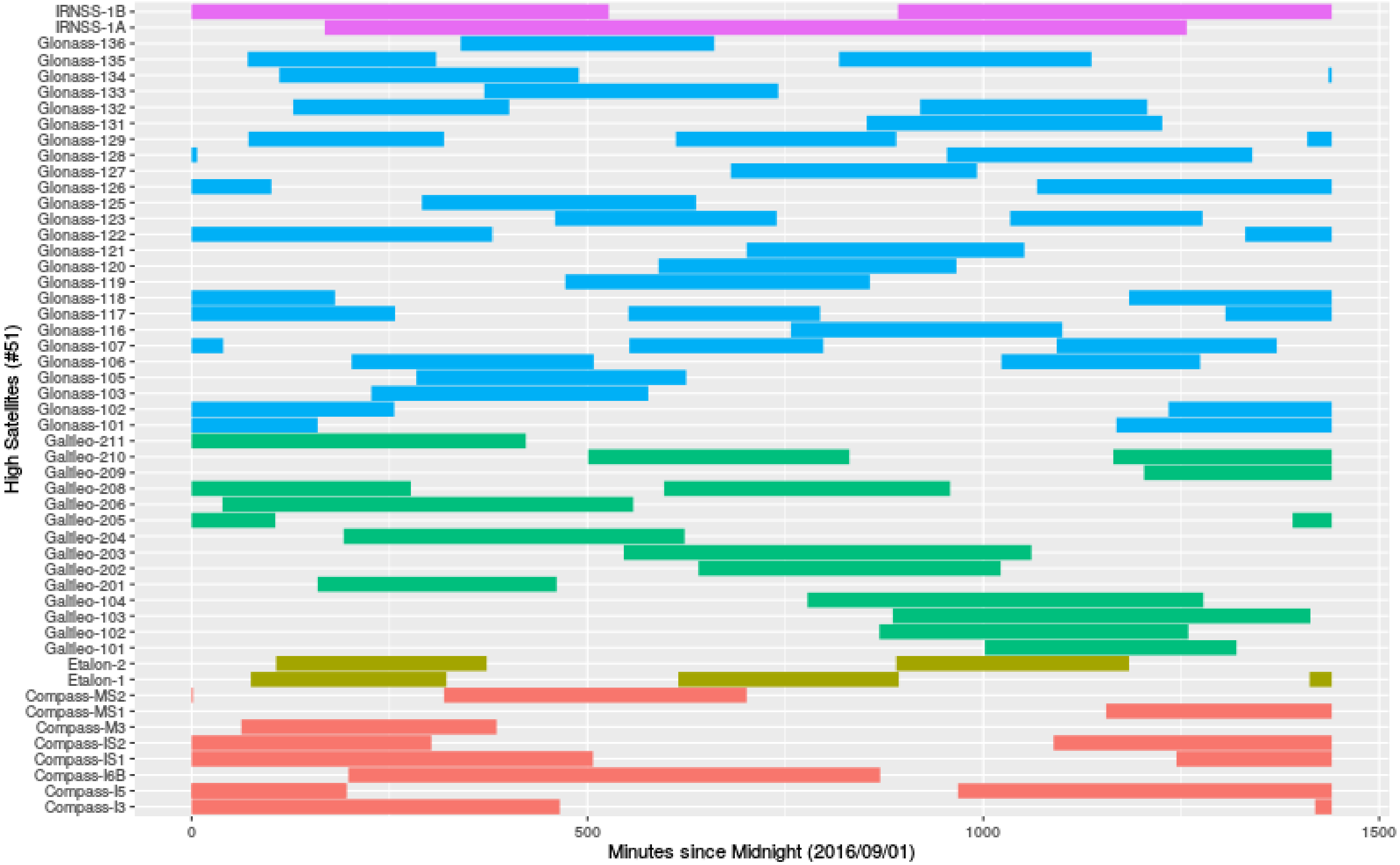
- Increasing number of targets increases complexity
- Scheduling strategies will change / evolve over time
- New scenarios and requirements will come up

- How hard is the problem?
- What kind of tracking strategies do we need to support?
- How to implement a solution which generates an optimal result?

How will be the satellite “weather” tomorrow?

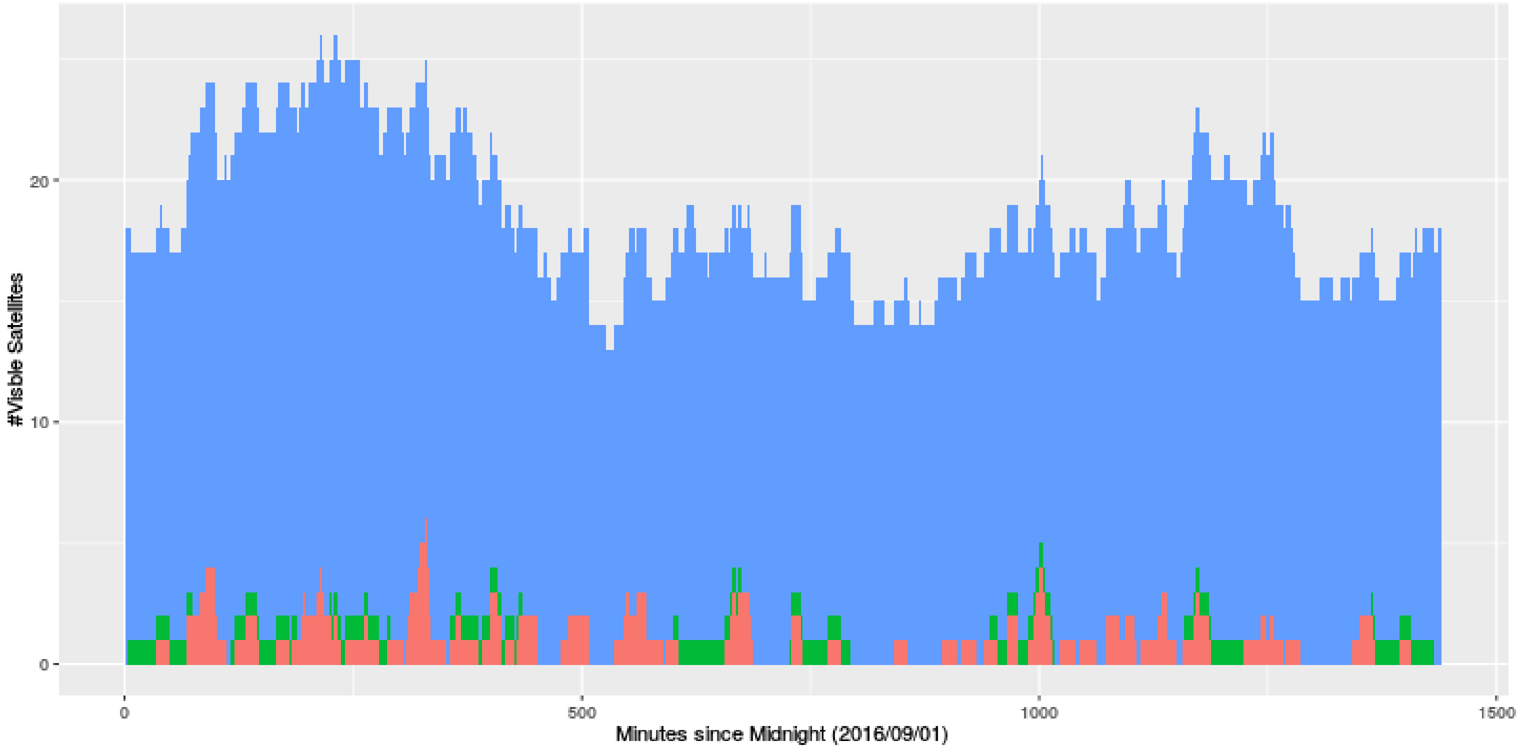
- 80% of the time there is at least one satellite visible, but nearly 50% of the time there is an overlap
- Many overlaps caused by satellites in close formation

How will be the satellite “weather” tomorrow?



- Not all targets are visible (e.g. IRNSS)
- Satellite are visible over several hours

How hard is the problem?



- Low and Lageos satellites
  - On average one satellite visible
  - Selected Space Debris objects will increase the number of low targets
- High satellites
  - On average 17 different satellites visible at the same time
  - Completion and extension of GNSS and RNSS will increase the number of high satellites

What kind of tracking strategies do we need to support?

- Station related aspects
  - Location (observation days, latitude)
  - Resources (staff members, shifts per day/week, system sharing)
  - Tracking capabilities (day/night tracking, range limits, elevation limits, kHz)

- Target related aspects
  - Orbit (station location, station tracking capacity)
  - Target properties (return signal, optical visibility)
  - Predictions (update frequency, position accuracy, time bias)

- Mission requirements / objectives
  - Priorities (mission priorities, campaign priorities)
  - Quantity requirements? (per pass, per pass segment, per station, per orbit, ...)
  - Quality requirements? (3x3x3 rule, NP every X min, NP at horizon vs. culmination, ...)

How do implement?

- Knowledge representation and reasoning (KR) is a field of artificial intelligence (AI)
- KR has developed a number of interesting technologies over the last decades

- Answer set programming (ASP) is a form of declarative programming
- ASP based on stable model semantic computed by an ASP solver
- Industrial strength solver are available as open source projects

- Potassco, the Potsdam Answer Set Solving Collection, bundles tools for Answer Set Programming developed at the University of Potsdam
- Used as solver backend in Debian based Linux distributions to resolve package dependency
- Used to ensure compliance with interference conditions during the reorganization of radio frequencies in USA 2016 (2.991 radio stations)

http://potassco.sourceforge.net/

Answer Set Programming

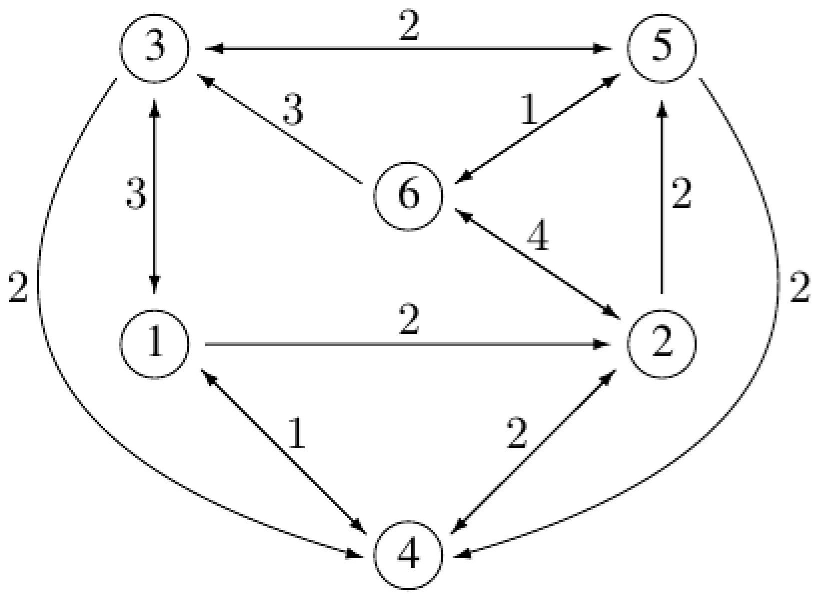
- ASP uses a simple text based format to represent a logical program
- Building block of each program are rules
- Each rule can be seen as implication

Format	Meaning	Example
<head> :- <body> .	Rule	b :- a,d.
<head>.	Fact	c.
:- <body>.	Constrained (head is false)	:- f,g.

- Other language elements

Format	Meaning	Example
{p, q, r}	Choice	{b,c,d} :- a.
1 {p, q, r} 2	Constrained choice	1 {b,c,d} 2 :- a.
p(X) : q(X)	Condition	edge(X:Y): Y = X + 1.
a..b	Interval	time(0..3) -> time(0) time(1) time(3)

Hello World (TSP)



Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

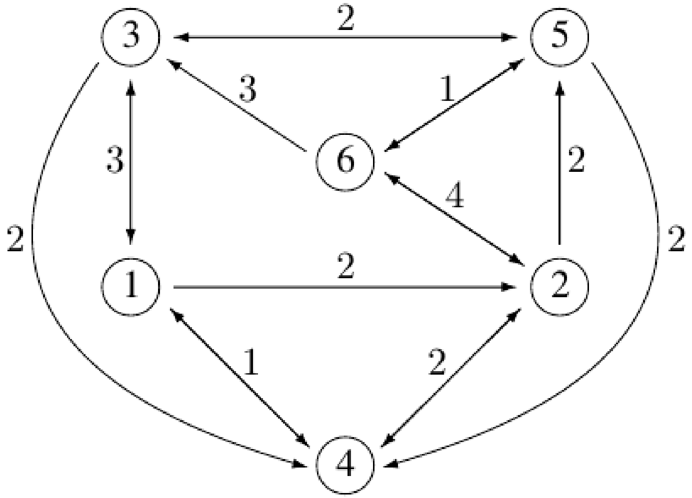
- Traveling Salesperson problem (TSP), Hamiltonian cycle with minimal costs
- For unconstrained TSP best known algorithm is trying all combination

Example from Potassco User Guide

Hello World (TSP) – Problem Instance

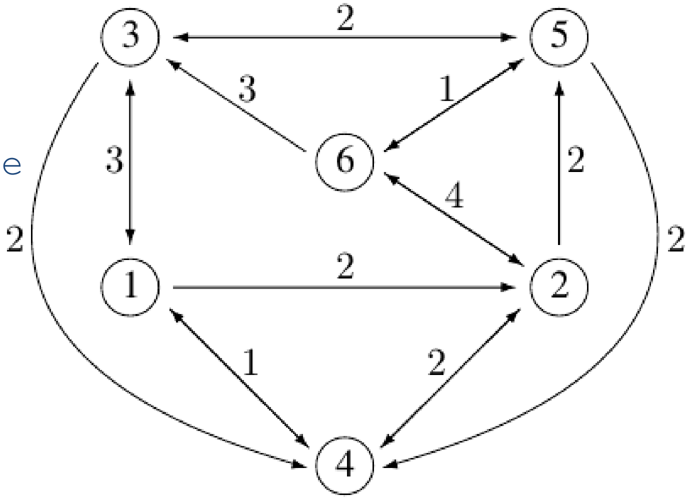
```
% Nodes
node(1..6).

% (Directed) Edges
edge(1,(2;3;4)). edge(2,(4;5;6)).
edge(3,(1;4;5)). edge(4,(1;2)).
edge(5,(3;4;6)). edge(6,(2;3;5)).
```

```
• % Edge Costs
• cost(1,2,2).  cost(1,3,3).  cost(1,4,1).
• cost(2,4,2).  cost(2,5,2).  cost(2,6,4).
• cost(3,1,3).  cost(3,4,2).  cost(3,5,2).
• cost(4,1,1).  cost(4,2,2).
• cost(5,3,2).  cost(5,4,2).  cost(5,6,1).
• cost(6,2,4).  cost(6,3,3).  cost(6,5,1).
```



Example from Potassco User Guide

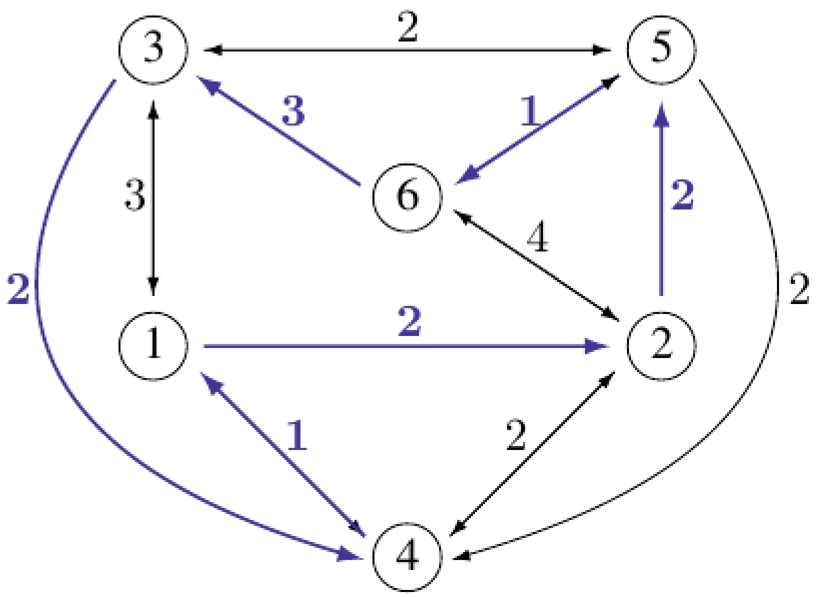
Hello World (TSP) - Encoding



```
% In a cycle each note must have one outgoing edge
1 { cycle(X,Y) : edge(X,Y) } 1 :- node(X).

% In a cycle each note must have one incoming edge
1 { cycle(X,Y) : edge(X,Y) } 1 :- node(Y).

% Each node must be part of the cycle
reached(Y) :- cycle(1,Y).
reached(Y) :- cycle(X,Y), reached(X).

% Remove solutions where a note exists which is not part of the cycle
:- node(Y), not reached(Y).

% Optimize sum of edge costs
#minimize { C,X,Y : cycle(X,Y), cost(X,Y,C) }.

% Display
#show cycle/2.
```

Example from Potassco User Guide

Hello World (TSP) - Solution



```
$clingo instance.lp encoding.lp 0
Answer: 1
cycle(1,3) cycle(2,4)  cycle(3,5)  cycle(4,1) cycle(5,6)  cycle(6,2)
Optimization: 13
Answer: 2
cycle(1,2) cycle(2,5) cycle(3,4) cycle(4,1) cycle(5,6) cycle(6,3)
Optimization: 11
```

Example from Potassco User Guide

SLR knowledge

- Target knowledge

Template	Examples	Description
orbit_type(O).	orbit_type(lageos).	Orbit type definition
target(S).	target(tla1).	Target definition
target_prio(S, I).	target_prio(tla1, 2).	ILRS priority assignment
target_type(S, O).	target_type(tla1, lageos).	Type assignment
target_min_time(S,M).	target_min_time(tla1, 2).	Min. tracking time

- Pass knowledge

Fact	Example	Description
pass(P).	pass(p20160901_0038_lageos1).	Pass definition
pass_target(P, S).	pass_target(p20160901_0038_lageos1, tla1).	Target assignment
pass_arc(P, T1, T2).	pass_arc(p20160901_0038_lageos1, 4, 73).	Pass interval

Generated Facts

```
%Orbits
orbit_type(leo). orbit_type(lageos). orbit_type(gnss).


%Targets
target(tlar). target_type(tlar,leo). target_prio(tlar,18).
target_min_time(tlar,1).
target(tla2). target_type(tla2, lageos). target_prio(tla2, 22).
target_min_time(tla2, 2).
target(t134g). target_type(t134g, gnss). target_prio(t134g, 37).
target_min_time(t134g, 5).
...


% Passes
pass(p20160901_0038_lageos1). pass_target(p20160901_0038_lageos1, tla1).
pass_arc(p20160901_0038_lageos1, 4, 73).
pass(p20160901_0042_larets). pass_target(p20160901_0042_larets, tlar).
pass_arc(p20160901_0042_larets, 35, 48).
pass(p20160901_0514_glonass134). pass_target(p20160901_0514_glonass134, t134g).
pass_arc(p20160901_0514_glonass134, 111, 489).
...
```

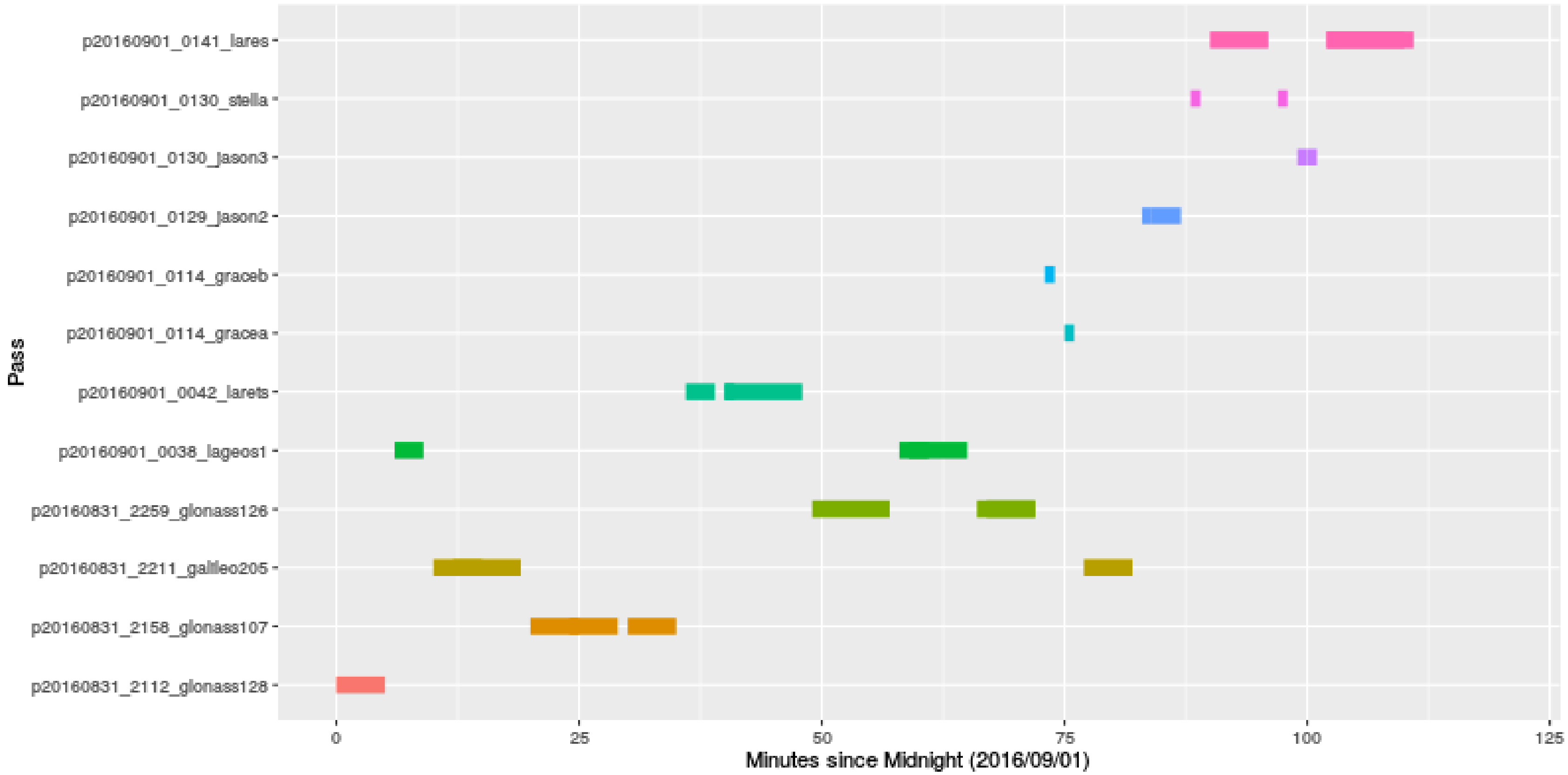
SLR scheduler – Encoding

```
% Expand requested time interval
time(minTime..maxTime).

% Expand time interval of each pass
pass_slice(P,T1..T2) :- pass_arc(P,T1,T2).

% Guess the position of up to three pass segments
0 { pass_segment(P,T1,T2) : pass_slice(P,T1), pass_slice(P,T2),
    target_min_time(S,M),
   T1 < T2, T1 + M <= T2 } 3 :- pass(P), pass_target(P,S).

% Expand tracking time interval
segment_slice(P,T1..T2) :- pass_segment(P,T1,T2).

% Drop conflicting solutions
:- segment_slice (P1,T), segment_slice(P2,T), P1 != P2.

% Optimization
#maximize {1@1,T : segment_slice(P,T)}.
#maximize {1@3,P : segment_slice(P,T)}.
#maximize {1@2,S : segment_slice(P,T), pass_target(P,S)}.
```

SLR scheduler – Solution

```
$ clingo -const maxTime=120 targets.lp passes.lp ecoding2.lp
clingo version 4.5.4
Reading from targets.lp ...
Solving...
Answer: 1
target_count(0) pass_count(0)
Optimization: 12 12 112
...
Answer: 68
segment_slice(p20160831_2112_glonass128,0)...
  segment_slice(p20160901_0141_lares,111) target_count(12) pass_count(12)
Optimization: 0 0 0
OPTIMUM FOUND

Models         : 68
  Optimum      : yes
Optimization : 0 0 0
Calls          : 1
Time           : 1.170s (Solving: 0.08s 1st Model: 0.00s Unsat: 0.00s)
CPU Time       : 1.160s
```

SLR scheduler – Solution 1



- No calibration, no switch time, no constraints, …
- But good starting pointing for writing a real encoding

Summary

- Pass scheduler is a core component of an automated system
- ASP is an promising technology
- Computational complexity has to be checked in feasibility study

- ASP provides readable implementation
- Different tracking strategies can be implemented
- With a good encoding all possible constraints can be expressed
- Optimization is part of the concept

- Simulation of different strategies on station level or network level possible