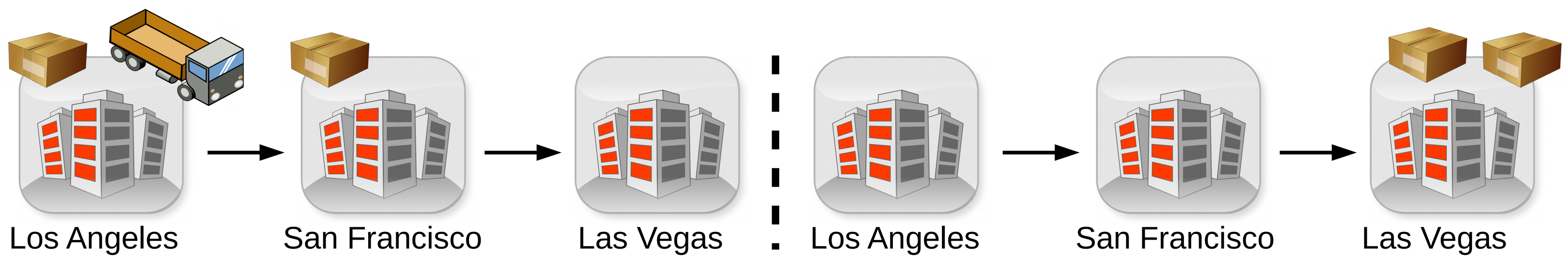
Stephan Gocht and **Tomáš Balyo**
({gocht.stephan, biotomas}@gmail.com)
Karlsruhe Institute of Technology
Germany

Accelerating SAT Based Planning with Incremental SAT Solving

1. What is Planning?

- World state: instantiation of multi valued state variables
- Actions:
  - require certain values of state variables to be used
  - change values of state variables by their effects
- **Objective:**
  - Given a set of actions
  - Given an initial state (start) and goal conditions
  - Find a plan (sequence of actions to get from start to goal)

2. Example: delivering 2 packages to Las Vegas



State Variables and their domains:
- Truck location T, dom(T) = {LA, SF, LV}
- Package locations P and Q
  dom(P) = dom(Q) = {LA, SF, LV, Tr}

Initial State: T=LA, P=LA, Q=SF
Goal Conditions: P=LV, Q=LV

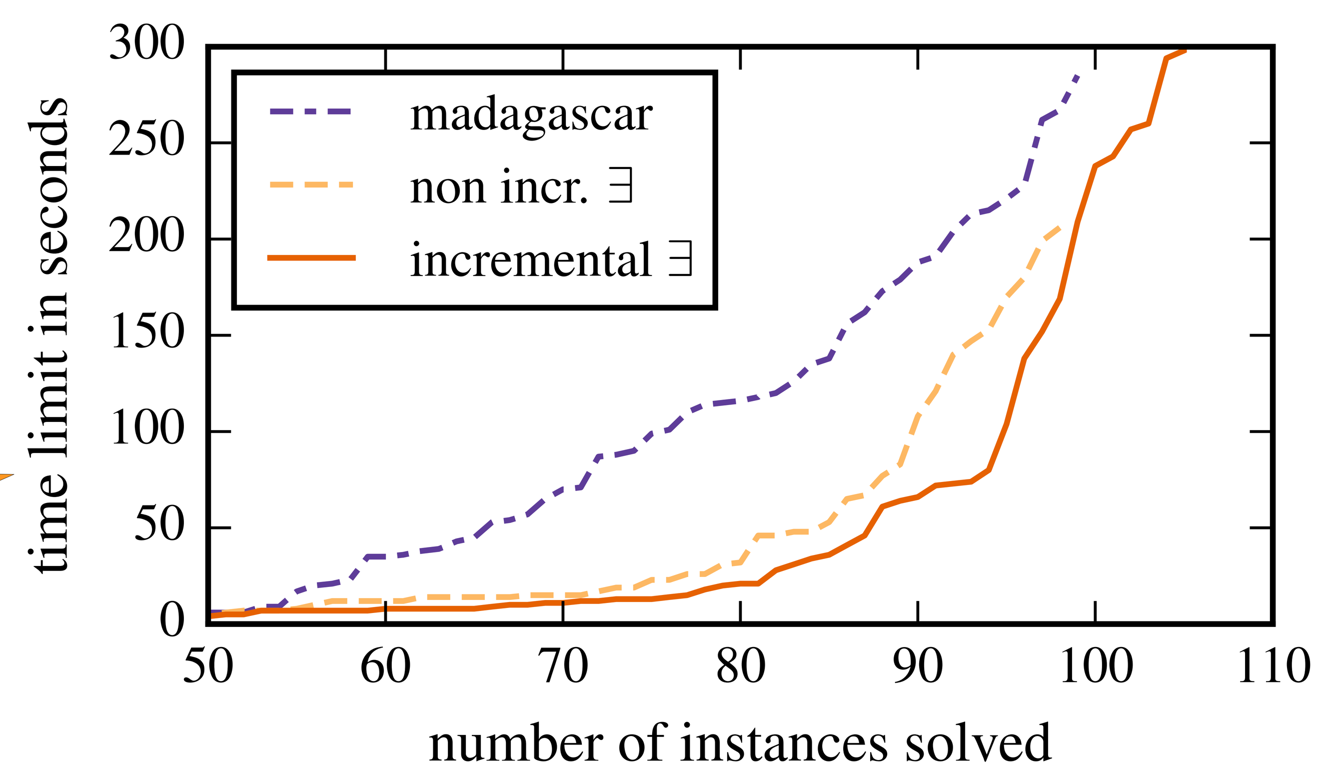
Actions:
- move(x,y)=[prec: {T=x}, eff: {T=y}]
- loadP(x)=[prec: {T=x, P=x}, eff: {P=Tr}]
- loadQ(x)=[prec: {T=x, Q=x}, eff: {Q=Tr}]
- dropP(x)=[prec: {T=x, P=Tr}, eff: {P=x}]
- dropQ(x)=[prec: {T=x, Q=Tr}, eff: {Q=x}]

Where x,y are LA, SF, and LV

Plan: loadP(LA), move(LA,SF), loadQ(SF), move(SF,LV), dropP(LV), dropQ(LV)

3. Finding Plans with Satisfiability Solvers

- Create F_k which is satisfiable only if plan of size k exists
- Many encoding schemes for F_k exist
  → use abstract view, such that all can be used:
  - initial clauses I:
    satisfied in the initial state
  - goal clauses G:
    satisfied in the goal state
  - transition clauses T:
    satisfied at each pair of consecutive states
- Solve F_1, F_2, ... until a satisfiable formula F_n is reached
- Use the solution of F_n to construct a plan



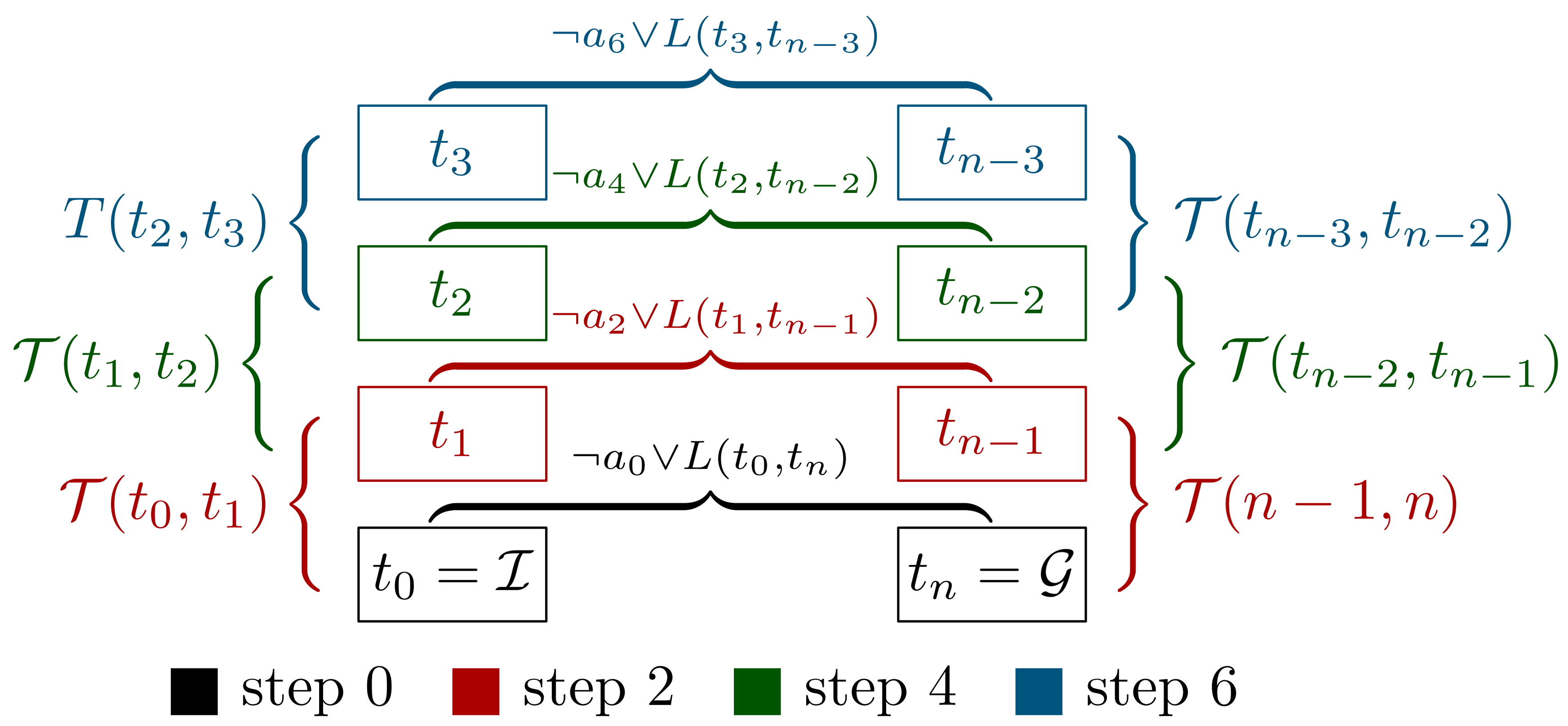
Cactusplot comparing madagascar with our approach.

4. Incremental SAT Solving

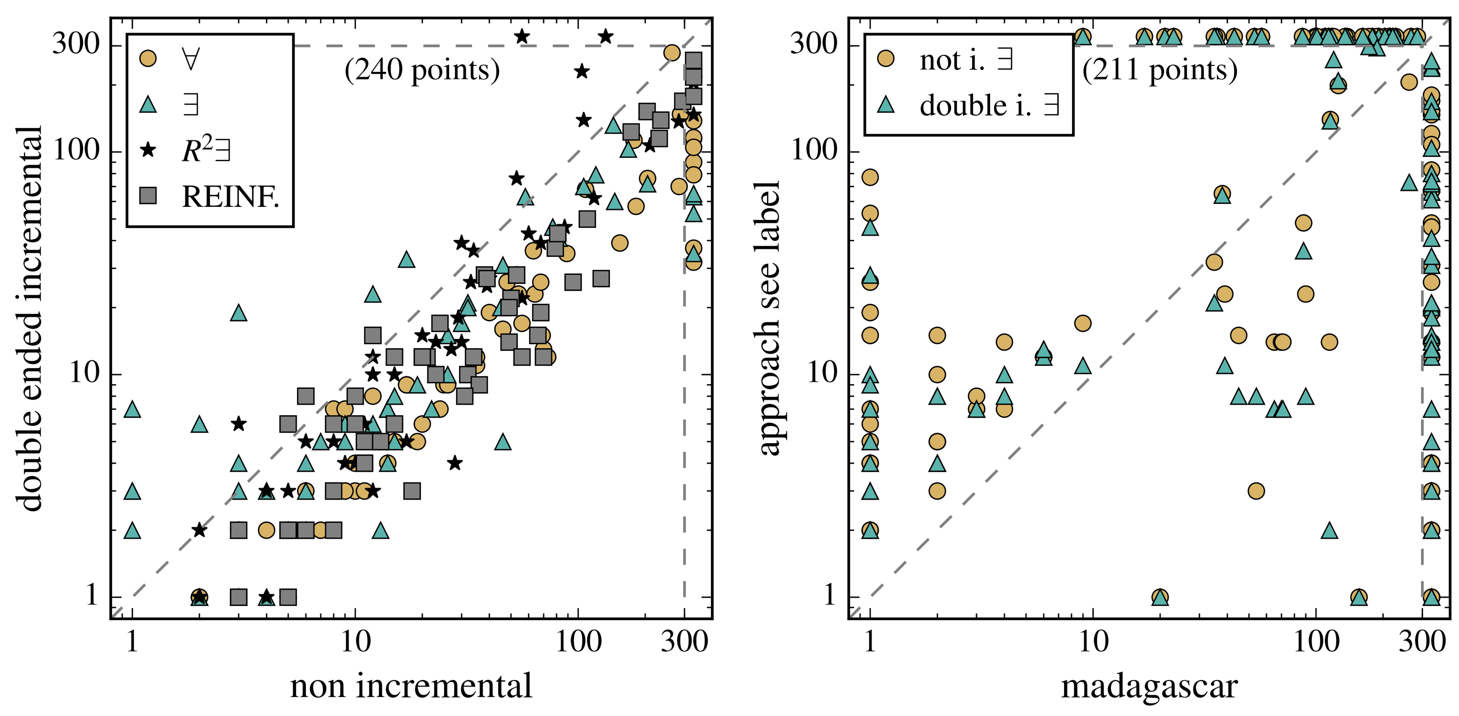
- Standardized interface IPASIR
  → profit from future SAT solver development
- **Idea:** Solve multiple slightly different formulas
- **Advantage:** Reuse information from previous solve:
  - Internally learned clauses
  - Importance of Variables and Clauses
  - Good assignments (phase saving)
  - ...
- **Realization:**
  - Keep all added Clauses
  - Solve with assumptions (temporary unit clauses)
    - Allows de-/ activation of clauses via activation literals

5. Extending F_n to F_{n+1}

- Add new timepoints with transitions alternating to goal and initial state.
- At step n the formula F_n is solved with assumption a_n and contains all clauses of previous steps.
- $L(t_i,t_k)$ is a link clauses such that all variables in t_i have the same value as in t_k and vice versa. As it has an activation literal, only the latest link clause is active.



Double Ended Incremental Encoding up to step 6 and with step size 2.



The Scatter Plots show a data point for each problem. If a problem is only solved by one approach but not the other it is plotted behind the 300 second.

6. Experiments

- Recent SAT solver from 2016 SAT Competition:
  - COMiniSatPS 2Sun nopre
- Benchmarks:
  - Agile Track of the 2014 International Planning Competition (IPC)
- Limits:
  - Time limit: 300s, 1 CPU core @ 2.10GHz, 8 GB RAM

7. Conclusion

- Incremental SAT solving is very beneficial
- It is important to keep track of SAT solver development

The 27th International Conference on Automated Planning and Scheduling - ICAPS 2017