Hierarchical Task Network Planning Using SAT Techniques

Dominik Schreiber

École nationale supérieure d'Informatique
et Mathématiques appliquées Grenoble

Karlsruhe Institut für Technologie

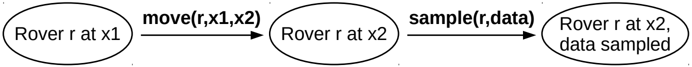
mail@dominikschreiber.de

July 24, 2018

Overview

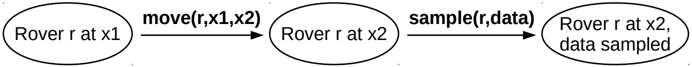
Motivation

- General context: Automated planning
- Generic model of world states and of actions transforming states
- Example: Rover vehicle collecting data of interest



Motivation

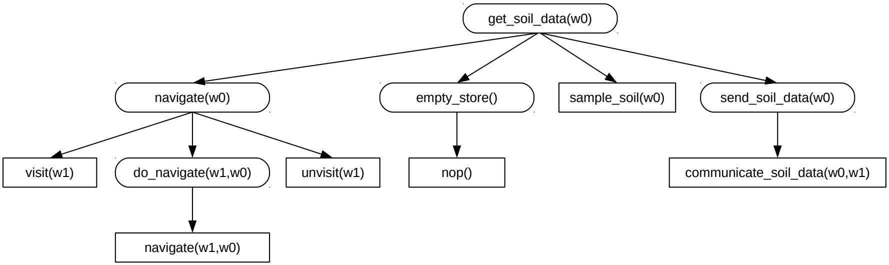
- General context: Automated planning
- Generic model of world states and of actions transforming states
- Example: Rover vehicle collecting data of interest



- Objective: Find a sequence of actions (a plan) leading to some goal state

Background: HTN Planning

- Popular extension of classical planning: Hierarchical Task Network (HTN) planning [GNT04]
  - Group certain sequences of actions into tasks
  - Recursively group tasks into bigger tasks ⇒ Graph (or network) of tasks
  - Provide some initial sequence of tasks to be performed
  - Planner explores valid reductions of these tasks until only actions remain
- More focused search of a plan possible ⇒ more efficient planning



Background: SAT Solving

- SAT problem: Given a propositional logic formula F, is F satisfiable?
  - F usually in Conjunctive Normal Form:

$$F = \bigwedge_{i=1}^{n} C_i = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} L_{ij}$$

  - If possible, find an assignment to each variable in F to `true` or `false` such that F evaluates to `true`
  - If not possible, report unsatisfiability of the formula

Background: SAT Solving

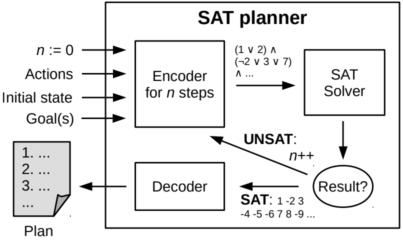
- SAT problem: Given a propositional logic formula F, is F satisfiable?
  - F usually in Conjunctive Normal Form:

$$F = \bigwedge_{i=1}^{n} C_i = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} L_{ij}$$

  - If possible, find an assignment to each variable in F to `true` or `false` such that F evaluates to `true`
  - If not possible, report unsatisfiability of the formula
- Practical usage of SAT solving:
  - Given some problem, find encoding of the problem in propositional logic
  - Solve resulting formula with a SAT solver
  - Decode satisfying assignment back into the problem domain

Background: SAT Planning

- Usual SAT planning procedure:
  Encode for *n* maximum steps,
  increase *n* until satisfiability
- Successfully applied to classical planning [KS+92],
  but few research towards HTN planning [MK98]



⇒ Task: Combine SAT planning approach with HTN planning model

GCT Encoding

Point of departure: LBF (Linear Bottom-Up Forward) encoding [MK98]

- No recursive task relationships (constant amount of max. actions per task)
- Complexity of variables / clauses cubic in amount of steps
- Used HTN model differs from model used by available preprocessing

⇒ Initial goal: Update LBF to modern HTN model

GCT Encoding

Point of departure: LBF (Linear Bottom-Up Forward) encoding [MK98]

- No recursive task relationships (constant amount of max. actions per task)
- Complexity of variables / clauses cubic in amount of steps
- Used HTN model differs from model used by available preprocessing

⇒ Initial goal: Update LBF to modern HTN model

Result: GCT (Grammar-Constrained Tasks) Encoding

- Emulates idea of LBF, applied to modern framework, with recursion
- Complexity: quadratic in amount of steps and tasks
  - Too complex for non-trivial planning problems
- Still allows for interleaving of subtasks in some special cases

⇒ Need significantly different encoding approach for efficient planning

Towards an incremental encoding

- Idea towards improvement: Make encoding incremental
  - Incremental SAT solving: Multiple solving attempts on a formula which may be successively changed in-between attempts
  - Solver can memorize learned conflicts from past attempts to speed-up solving procedure [NSII06]
  - Much more compact representation of encoding possible

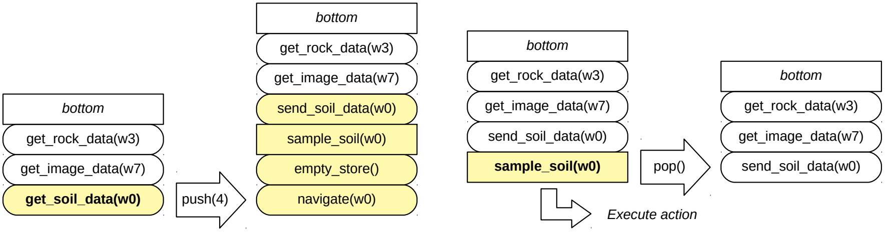
Towards an incremental encoding

- Idea towards improvement: Make encoding incremental
  - Incremental SAT solving: Multiple solving attempts on a formula which may be successively changed in-between attempts
  - Solver can memorize learned conflicts from past attempts to speed-up solving procedure [NSII06]
  - Much more compact representation of encoding possible
- To enable incremental expansion of encoding, each clause should only contain variables from adjacent steps
- **Consequence:** Entire "active hierarchy" needs to be memorized at each step, transferred to next step

SMS Encoding (1)

2nd proposed encoding: SMS (Stack-Machine Simulation)

- Encode a stack of tasks at each computational step
  - Initial step: Stack contains initial tasks, *bottom* symbol
  - Goal: Stack contains *bottom* as the first (and only) element
- Two central stack transformations between steps:
  push subtasks of composite task, and pop primitive action



SMS Encoding (2)

Clause examples:

- Necessary and sufficient conditions of an action execution

$$stackAt(0, a) \implies execute(a) \wedge pop()$$
$$execute(a) \implies stackAt(0, a)$$

- Stack movement when a push is done

$$push(k) \wedge stackAt(s, t) \implies stackAt'(s + k, t)$$

SMS Encoding (2)

Clause examples:

- Necessary and sufficient conditions of an action execution

$$stackAt(0, a) \implies execute(a) \land pop()$$
$$execute(a) \implies stackAt(0, a)$$

- Stack movement when a push is done

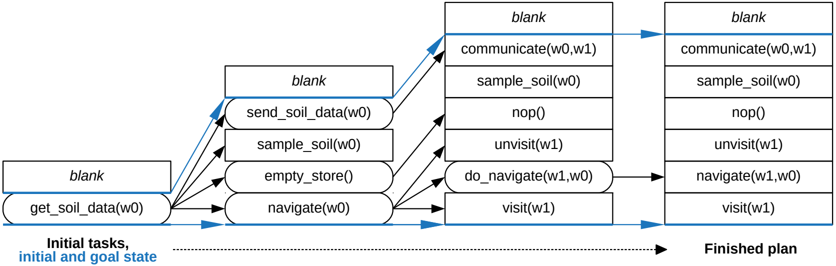
$$push(k) \land stackAt(s, t) \implies stackAt'(s + k, t)$$

Discussion

- Works reliably on all considered special cases
- Requires stack size as problem-dependent parameter
- Encoding is quadratic in number of steps if stack size is chosen cautiously
- High amount of incremental steps needed to find a plan
  - Long sequence of little changes to the current task hierarchy

Improving on SMS

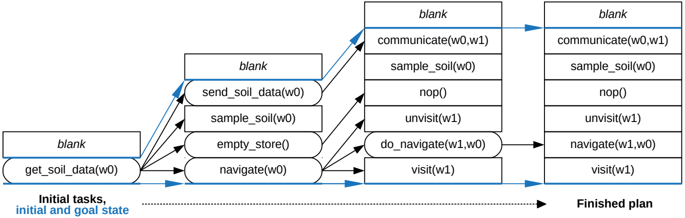
- To decrease number of incremental steps needed: Reduce all present non-primitive tasks at once, simultaneously
  - Each step represents an abstract plan, becoming more and more concrete until only actions remain ⇒ Plan!
- Result: Tree-like exploration of task hierarchy; encoding grows along maximum depth considered (∼ Breadth-first search)



T-REX Encoding

Final encoding: T-REX (Tree-like Reduction Exploration)

- Uses abstract clause notation tailored to encoding approach
- Custom interpreter application instantiates clauses just as needed
- Only encode actions, reductions at positions where they can occur
- Variable / clause complexity: no quadratic term any more, instead bounded by sum of all array sizes



T-REX Plan optimization

- T-REX may find longer plans than previous encodings
- Optional plan length optimization after finding initial solution:
  - Additional Clauses "count" effective plan length
  - Search for shorter plan by assuming a literal of the type "*The plan length is shorter than k*" and calling SAT solver again
    - Satisfiable: New, shorter plan found
    - Unsatisfiable: Lower bound on possible plan length found

T-REX Plan optimization

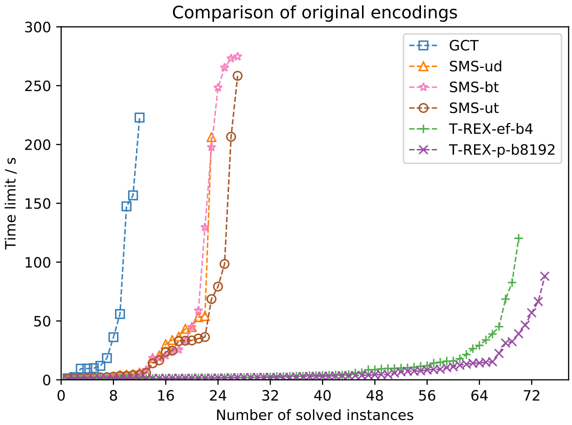
- T-REX may find longer plans than previous encodings
- Optional plan length optimization after finding initial solution:
  - Additional Clauses "count" effective plan length
  - Search for shorter plan by assuming a literal of the type "*The plan length is shorter than k*" and calling SAT solver again
    - Satisfiable: New, shorter plan found
    - Unsatisfiable: Lower bound on possible plan length found
  - Can be combined with any search strategy (e.g. bisection, linear) to successively tighten bounds on plan length
- Interruptible Anytime algorithm

Evaluation: T-REX Parameter tuning

- Many potential encoding variants possible within T-REX
- Use of ParamILS tuning framework [HHLBS09], popular within context of SAT [HBHH07] and planning [AB12]
  - Provide set of tuneable parameters (cmd arguments), training instances
  - ParamILS does very intelligent search over parameter space
- Best found configuration has been used for further evaluation steps

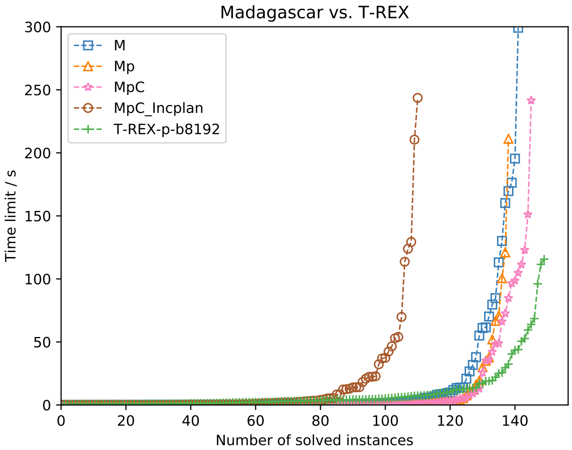
Evaluation: GCT, SMS, T-REX

- Domains from IPC benchmarks
- Performance scores: GCT < SMS < T-REX (by significant margins)
- **Plan lengths:** GCT finds shortest plans, SMS very comparable, T-REX sometimes finds longer plans (< 150%)



Evaluation: T-REX vs. Madagascar

- Comparing T-REX against classical SAT planner Madagascar [Rin14]
- T-REX overall competitive
- Performances per domain depend on
  - potential to parallelize actions for Madagascar
  - HTN model design for T-REX

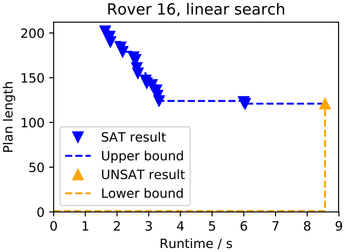


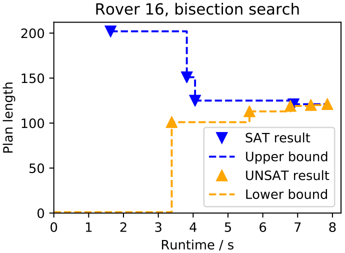
Evaluation: T-REX vs. GTOHP

- Comparing T-REX against HTN planner GTOHP [RPFP17]
  - Uses same preprocessing as T-REX
  - No SAT techniques involved
- Run times: T-REX takes longer
  - GTOHP does very focused search,
    SAT-based approach requires enumerating all possible reductions
  - Large computational overhead of encoding, instantiation etc.,
    Preprocessing specifically developed for GTOHP
- Advantages of T-REX over GTOHP:
  - Plan length optimization
  - Ability to prove properties of a problem
  - Operates much more robustly if preconditions are missing

Evaluation: T-REX Plan optimization

- Two search strategies:
  - Linearly descending search
  - Bisection search
- Linear search overall more convincing
  - Quick payoff (many slight optimizations)
  - Faster termination on average





Conclusion

- Efficient SAT planning on HTN domains is possible and viable, if engineered carefully
- Difficult to achieve run times of conventional planners, but SAT-based approach may have other merits
- Incremental SAT solving allows for compact problem representations and efficient optimization strategies

Conclusion

- Efficient SAT planning on HTN domains is possible and viable, if engineered carefully
- Difficult to achieve run times of conventional planners, but SAT-based approach may have other merits
- Incremental SAT solving allows for compact problem representations and efficient optimization strategies

Future work: Further optimization of approach

- Extend employed HTN model by additional constraints
- Optimize preprocessing for SAT encoding purpose
- Investigate different schedulings of extending the formula and parallel SAT solving techniques within T-REX

References I

Maher Alhossaini and J.Christopher Beck, *Macro learning in planning as parameter configuration*, Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 7310, Springer Berlin Heidelberg, 2012, pp. 13–24.

Malik Ghallab, Dana Nau, and Paolo Traverso, *Automated planning: theory and practice*, Elsevier, 2004.

Frank Hutter, Domagoj Babic, Holger H. Hoos, and Alan J. Hu, *Boosting verification by automatic tuning of decision procedures*, Formal Methods in Computer Aided Design, 2007. FMCAD '07, 2007, pp. 27 –34.

Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle, *ParamILS: An automatic algorithm configuration framework*, Journal of Artificial Intelligence Research **36** (2009), 267–306.

Henry A Kautz, Bart Selman, et al., *Planning as satisfiability.*, ECAI, vol. 92, Citeseer, 1992, pp. 359–363.

Amol Dattatraya Mali and Subbarao Kambhampati, *Encoding HTN planning in propositional logic.*, AIPS, 1998, pp. 190–198.

Hidetomo Nabeshima, Takehide Soh, Katsumi Inoue, and Koji Iwanuma, *Lemma reusing for SAT based planning and scheduling.*, ICAPS, 2006, pp. 103–113.

References II

Jussi Rintanen, *Madagascar: Scalable planning with SAT*, Proceedings of the 8th International Planning Competition (IPC-2014) **21** (2014).

Abdeldjalil Ramoul, Damien Pellier, Humbert Fiorino, and Sylvie Pesty, *Grounding of HTN planning domain*, International Journal on Artificial Intelligence Tools **26** (2017), no. 05, 1760021.

The End