Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC

Arthur Bit-Monnot

Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC

Arthur Bit-Monnot

LAAS-CNRS, Université de Toulouse, INSA, CNRS, Toulouse, France
abitmonnot@laas.fr

Abstract

In this paper we give a high level overview of the ARIES planner at the time of its participation in the hierarchical track of the 2023 *International Planning Competition (IPC)*.

ARIES is an experimental solver whose aim is to jointly evolve a combinatorial solver for scheduling-like problems and an encoding of task planning that exploits it.

Overview

Focus The focus of ARIES is on optimization planning problems with rich temporal and concurrency requirements, driven by applications in robotics and logistics. As such, the problems targeted by the 2023 IPC on hierarchical planning are certainly out of its comfort zone. Nevertheless, with ARIES participation in the IPC we hope to highlight the relative weaknesses and hopefully strengths of its approach.

As HDDL is not its primary target, at the time of its submission, very little effort has been dedicated to improving the performance of ARIES on a corpus of HDDL problems. This participation is also the occasion of setting a milestone and reference point to be used as a baseline for future improvements.

Search Space In terms of search space, ARIES pertains to the family of lifted plan-space planners like IxTeT (Ghallab and Laruelle 1994) and FAPE (Bit-Monnot et al. 2020). In practice, it means that the solver does not construct snapshots of the entire state at various epochs but instead reasons on the interactions and dependencies between individual actions (through threats and causal links, Ghallab, Nau, and Traverso 2004). In addition, being lifted implies that the problem is never grounded: instead each action of the plan is parameterized with variables that must obey a set of constraints, notably reflecting the causal requirements of the action. The solver must eventually impose a value for these parameters that ensures that the plan is valid and achieves the intended objectives.

Search Strategy Unlike IxTeT and FAPE, ARIES delegates the responsibility of search to an independent solver. We adopt the approach of generating a CSP of bounded size that is submitted to a combinatorial solver, as common in SAT-based planners such as Lilotane (Schreiber 2021). ARIES uses in own specialized solver: a hybrid CP-SAT combinatorial solver for optional scheduling (Bit-Monnot 2023). At a very high level, the planner adopts the following procedure:

1. Parse HDDL problem files and translates them into chronicles, where each chronicle consists of a collection of timed conditions, effects and subtasks linked by shared variables and constraints.
2. From the objective tasks of the problem, build the decomposition tree with a maximum depth (initially 1).
3. Encode the decomposition tree into a Constraint Satisfaction Problem (CSP)
4. Solve the CSP with the internal combinatorial solver
5. If the CSP was proven unsatisfiable, repeat from step 2 with an increased maximum depth.

Encoding HDDL

The planning model used internally by ARIES is based on *chronicles* (Ghallab, Nau, and Traverso 2004), a formalism to compactly represent requirements and effects of a process in time. In planning, chronicles find their first usage in IxTeT (Ghallab and Laruelle 1994) and have been then extended for hierarchical planning in FAPE (Bit-Monnot et al. 2020).

Translating HDDL to chronicles is fairly straightforward as they are for the most part strictly more expressive. In practice, it mostly consists in assigning temporal semantics to a non-temporal language. An example of a translation of an HDDL method into a chronicle is given in Figure 1.

Given a problem representation as chronicles, we encode it as a CSP using the formulae of Godet and Bit-Monnot (2022). In spirit, this encoding corresponds to a lifted plan-space representation. For each action/mehtod node of the lifted decomposition tree, it introduces decision variables representing *(i)* the presence of the action/method in the solution plan, *(ii)* its parameters and *(iii)* its start and end times. These decision variables are linked by a set of constraints, notably:

- *Decomposition constraints* that force the refinement of each task by exactly one method or action, if present.
- *Support constraints* that require each condition to be either absent or supported by an effect.

```
(:method m8_send_soil_data
  :parameters (
    ?x - rover
    ?from - waypoint
    ?l - lander
    ?w1 - waypoint
    ?w2 - waypoint)
  :task (send_soil_data ?x ?from)
  :precondition (and (at_lander ?l ?w2)
                     (visible ?w1 ?w2))
  :ordered-subtasks (and
    (t1 (do_navigate1 ?x ?w1))
    (t2 (comm_soil_data1 ?x ?l ?from ?w1 ?w2))
      ))
```

(a) An HDDL method from the *Rovers* domain.

m8-send-soil-data

variables:	x,*from*, l, w_1, w_2 (parameters) $t_s, t_e, t_s^1, t_e^1, t_s^2, t_e^2$ (timepoints)
task:	$[t_s, t_e]$ *send-soil-data*$(x$,*from*$)$
conditions:	$[t_{start}]$ *at-lander*$(l, w_2) =$ true $[t_{start}]$ *visible*(w_1, w_2) = true
subtasks:	$[t_s^1, t_e^1]$ *do-navigate1*(x, w_1) $[t_s^2, t_e^2]$ *comm-soil-data1*$(x, l$,*from*$, w_1, w_2)$
constraints:	$t_s = min(t_s^1, t_s^2)$ $t_e = max(t_e^1, t_e^2)$ $t_e^1 < t_s^2$ (total order)

(b) Equivalent chronicle representation.

Figure 1: HDDL and chronicle representation of an HTN method of the *rovers* domains

- *Coherence constraints* that ensure that each state variable is never given more than one value at a time.

Compared to SAT-based encodings for HTN, the size of this encoding is mostly independent of the length of the plan and the number of predicates in the state. On the other hand, it grows essentially quadratically with the size of the lifted decomposition tree, because of the support and coherence constraints.

Matching PDDL semantics Despite the overall compatibility of chronicles with {HP}DDL, several peculiarities of PDDL must be explicitly handled.

PDDL introduces the concept of *mutex actions* to forbid the concurrent execution of two interfering actions. This is required in PDDL due to the instantaneous nature of actions but introduces an awkwardness for a temporal model as the handling of a condition statement depends on the place where it was asserted. To handle such cases, we extend the constraints of Godet and Bit-Monnot (2022) with *mutex constraints* that forbid a condition's interval to meet the start of an effect from another action chronicle. Note that mutexes induced by pairs of *effects* are already enforced by *coherence constraints*.

For historical reasons, the *delete-before-add* semantics of PDDL stipulates that if a single action assigns both *true* and *false* to a common boolean state-variable, then only the positive assignment should be considered. This violates the principle that a state variable should not be assigned two variables at the same time enforced by the *coherence constraints*. ARIES does not support this particular corner case of the PDDL semantics as handling it properly in a fully lifted representation would notably complexify the encoding and may induce non-negligible runtime penalties. As a result, ARIES will consider inapplicable any action that relies on the *delete-before-add* semantics, making it incomplete in the presence of such problems. This is for instance the case of the *woodworking* domain where ARIES is unable to find a solution.

Finally, support for HDDL syntax remains partial. For instance, the multiple inheritance of types exploited in the *UMTranslog* domain of IPC 2020 is unsupported and ARIES would refuse to parse the problem.

Combinatorial Solver

The encoding of Godet and Bit-Monnot (2022) is generic and could in theory exploit any solver capable of representing disjunctions and reified difference constraints, which is the case of most CP, SMT or MILP solvers. Our experience with existing solvers for this encoding is however lukewarm.

Previous experiences

First let us state that the highly combinatorial structure of the encoding does not appear to play well with the linear relaxations of MILP solvers that were at difficulty even for the simplest instances.

Experiments with the Z3 SMT solver (De Moura and Bjørner 2008), showed interesting results with reasonable performance on non-hierarchical domains (Bit-Monnot 2018). Most impressive in particular was the ability of the activity-based search of SAT/SMT solver to robustly converge to a solution or prove its absence without any planning-specific knowledge. On the other hand, SMT-based solvers remain hard to tune with limited support for optimization. While Z3 excelled at proving unsatisfiability, the runtime to a first solution remained very high even on simple problems.

Our own experiments with CPOptimizer (Laborie et al. 2018), a leading CP solver for scheduling, showed that it had complementary strength: fine-tuning and strong propagation allowed it to quickly converge to a first solution and incrementally improve it. A key feature for strong propagation was a native support of optional activities, that are ubiquitous in task planning. CPOptimizer may however struggle to prove unsatisfiability or escaping dead-ends in its search space in situations that would be easily handled by the conflict-directed clause learning of SAT/SMT solvers.

While our use of existing solvers clearly showed limitations, FAPE provided an alternate model. FAPE uses best-search in the space of lifted partial plans, with each partial plan encoded as a CSP that is iteratively refined and constrained as search progresses. This CSP was handled with

a custom propagation engine developed within the solver's code base. Our experience is that developing side-by-side a propagation engine and the planner that uses it, allowed us to identify and address fundamental limitations of the propagation engines. For FAPE, this resulted in very strong propagation engine and compact search space (Bit-Monnot et al. 2020).

ARIES Scheduler

Following a similar route, ARIES relies on its own combinatorial solver that could be captioned as a *hybrid CP-SAT* solver for *optional scheduling*. Its most distinctive features are the following:

- The core of a SAT solver, (clause learning, unit propagation and activity-based search) integrated with finite-domain constraint solver.
- Dedicated reasoner for difference logic (aka. Disjunctive Temporal Networks), that notably enables conflict detection and explanations for temporal constraints.
- First-hand support of optional activities, enabling eager propagation of the potential timing and parameters of activities that are not yet part of the plan

While the solver is still in its early days, preliminary evaluation shows that it has state-of-the-art performance on disjunctive scheduling problems such as the jobshop and openshop scheduling problems (Bit-Monnot 2023).

Technical Remarks

ARIES is implemented from the ground up in Rust and is freely available under the MIT license at https://github.com/plaans/aries. Most of the code base is dedicated to the implementation of our hybrid CP-SAT solver. Comparatively, planning specific code only occupies a small fraction of the code base, mostly dedicated to straightforward parsing and encoding tasks.

Beside its partial support for HDDL, ARIES also provides an integration as a backend for the *unified-planning* library with richer modeling features. Direct encoding of planning problems as chronicles or as a CSP is another way to exploit ARIES in a more flexible way, e.g., for scheduling problems.

Competition Results

The 2023 Hierarchical Planning Competition had tracks for Totally-Ordered (TO) and Partially-Ordered (PO) problems. In TO problems all task networks are totally ordered, thus specifying a total order among all tasks to be carried out. PO problems have not such restrictions.

Both TO and PO problems had three tracks each:

- **Satisficing** track: that favors valid plans returned within 30 minutes, with a typically small bonus for higher-quality plan.
- **Agile** track: that favors plans returned quickly regardless of their quality.
- **Optimal** track: that requires that only provably optimal plans are returned.

ARIES participated in all six tracks with a single search configuration. The only difference in between the tracks was on the decision of when to abandon the search and write a solution plan to the result file.

We analyse below the results of ARIES in the different tracks. While each planner could have several variants, when presenting results, we only show the performance of the best configuration of a given planner. The full results remain available from the IPC website.[1]

Total Order Tracks

TO HTN planning imposes a total order on all activities in the plan. Even objective tasks must be achieved in a predefined order, eliminating the (qualititative) scheduling subproblem of typical planning problems. As ARIES is primarily a scheduler, it unsurprisingly performs badly in the total order tracks.

The results of the satisficing track (Table 1), reveal that solvers specifically developped for TO HTN perform substantially better than generic solvers also capable of handling PO problems, with the worse specialized TO planner solving twice as many problems as the best generic one.[2]

A short analysis reveals that, TO instances tend to be very large and feature recursive decomposition. In a competition context, this is necessary to maintain the hardness of the instances while compensating for the lack of a scheduling subproblem. For ARIES, it means that most problems lie in the area where it is the most uncomfortable: where the decomposition trees are very large and with an unbounded depth due to the recursive tasks. On such problems, ARIES performs badly as its constraint encoding grow quadratically with the size of the decomposition tree, and it needs to prove the absence of solution for a given depth of the decomposition tree before proceeding to the next (only for recursive problems).

In the optimal track, ARIES essentially did not compete as it only supported the *Barman BDI* problem. Except for the woodworking domain (that requires delete-before-add semantics) all other 20 domains where recursive (for which ARIES would never assert the optimality of a plan it found).

Partial Order Tracks

Result of the Partial Order (PO) tracks are more interesting to us. In particular, the domains are more balanced, with only half of the domains being recursive. Yet domains exploit little specificities available with PO HTN, as witnessed by the excellent performance of the *Linear* family of planners that work by treating PO HTN problems as totally-ordered problems. Most regrettable is the use of the *sequential plan length* as the quality metric where, e.g., a *parallel plan length* would have been more appropriate for a partial order track. An additional concern is that *sequential plan length* correlates very well with the search effort of progression-

[1] https://ipc2023-htn.github.io/#results

[2] Note that PANDAPro has non-specialized variants that can be run on PO problems but differ in the search algorithm employed (Höller and Behnke 2021).

	Specialized TO planners				Generic planners		
Planner	PandaDealer	PANDAPro	TOAD	LiftedTreePath	SIADEX	ARIES	OptiPlan
Total Score	15.29	15.15	10.93	7.13	3.45	3.19	0.72

Table 1: Total Order – Satisficing

Planner	Total Score	Barman BDI	Monroe Fully	Monroe Partially	PCP	Rover	Satellite	Transport	Ultralight Cockpit	Woodworking	Colouring
Linear	**7.60**	**0.66**	**0.94**	0.70	**0.82**	0.86	0.98	0.26	**1.00**	**0.70**	**0.69**
PANDAPro	6.47	0.15	0.91	**0.75**	**0.82**	0.43	0.99	**0.33**	**1.00**	0.42	0.66
ARIES	4.73	0.15	0.44	0.52	0.59	0.79	**1.00**	0.32	0.66	–	0.27
SIADEX	2.18	0.62	0.26	0.04	0.00	0.31	0.83	0.03	0.00	0.09	0.00

Table 2: Partial Order – Satisficing

Planner	Total Score	Barman BDI	Monroe Fully	Monroe Partially	PCP	Rover	Satellite	Transport	Ultralight Cockpit	Woodworking	Colouring
Linear	**7.04**	0.81	**0.57**	**0.44**	**0.82**	**1.00**	**1.00**	**0.52**	**0.93**	**0.63**	**0.30**
PANDAPro	5.02	0.08	0.50	0.36	**0.82**	0.36	0.99	0.34	0.92	0.36	0.29
ARIES	3.22	0.05	0.21	0.26	0.38	0.44	**1.00**	0.20	0.58	–	0.12
SIADEX	3.03	**0.92**	0.24	0.05	0.00	0.70	**1.00**	0.03	0.00	0.10	0.00

Table 3: Partial Order – Agile

Planner	Total Score	Barman BDI	Rover	Satellite	Ultralight Cockpit
PANDAPro	**2.36**	**0.10**	0.30	0.96	**1.00**
ARIES	2.11	0.05	**0.40**	**1.00**	0.66

Table 4: Partial Order – Optimal

based planners and contributes to their good performance in this setting.

Nevertheless, ARIES remains competitive with the top planners in the satisficing track (Table 2), ranking third out of four competing planners (a fifth planner, *OptiPlan*, was disqualified for returning invalid plans). Interestingly, ARIES substantially outperforms SIADEX that was the winner of the track in the previous competition and the only other temporal planner. ARIES score decreases in the *agile* track, indicating that it is comparatively slower than SIADEX in finding plans (but with twice the coverage, Table 3).

Results of the optimal track are presented in Table 4, restricted to domains where ARIES supports optimal planning (i.e. non-recursive domains). In these four domain, PANDAPro and ARIES are mostly tied, with each planner outperforming the other in two domains. The final score gives a small advantage to PANDAPro.

Conclusion

The 2023 IPC was the occasion for us to enroll a temporal scheduler in the Hierarchical Planning Competition. The focus of the planner lies in optimization problems at the border between temporal planning and scheduling, and very little tuning has been done to improve its performance on HDDL problems.

Results in the Total-Order track confirmed that such planner do not belong there. However, ARIES obtained a runner-up position in the Partial-Order optimal track and performed more than honorably in the partial-order satisficing track.

To us these results demonstrate the potential of the approach of ARIES, especially in the case where plan quality matters, which is the case in most real-world problems (with more realistic metrics). While we intend to keep the focus of ARIES on temporal and numeric planning, it would be interesting to analyze the shortcoming of ARIES in the more restricted benchmarks of the IPC in order to gain insights on possible optimizations targeting causal reasoning.

Acknowledgments

This work has been partially supported by AIPlan4EU, a project funded by EU Horizon 2020 research and innovation program under GA n.101016442.

References

Bit-Monnot, A. 2018. A Constraint-Based Encoding for Domain-Independent Temporal Planning. In *International Conference on Principles and Practice of Constraint Programming (CP)*.

Bit-Monnot, A. 2023. Enhancing Hybrid CP-SAT Search for Disjunctive Scheduling. In *European Conference on Artificial Intelligence (ECAI)*.

Bit-Monnot, A.; Ghallab, M.; Ingrand, F.; and Smith, D. E. 2020. FAPE: a Constraint-based Planner for Generative and Hierarchical Temporal Planning.

De Moura, L.; and Bjørner, N. 2008. Z3: An Efficient SMT

Solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

Ghallab, M.; and Laruelle, H. 1994. Representation and Control in IxTeT, a Temporal Planner. In *International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*.

Godet, R.; and Bit-Monnot, A. 2022. Chronicles for Representing Hierarchical Planning Problems with Time. In *ICAPS Workshop on Hierarchical Planning (HPlan)*.

Höller, D.; and Behnke, G. 2021. Loop Detection in the PANDA Planning System. *Proceedings of the International Conference on Automated Planning and Scheduling*.

Laborie, P.; Rogerie, J.; Shaw, P.; and Vilím, P. 2018. IBM ILOG CP Optimizer for Scheduling. *Constraints*.

Schreiber, D. 2021. Lilotane: A Lifted SAT-based Approach to Hierarchical Planning. *Journal of Artificial Intelligence Research (JAIR)*.