Disjunctive Temporal Refinement Planning with Variable Action Duration and Execution Makespan Bounds

Mica Gardone
Kahlert School of Computing, University of Utah
Salt Lake City, United States
m.gardone@utah.edu

Rogelio E. Cardona-Rivera
Division of Games, University of Utah
Salt Lake City, United States
r.cardona.rivera@utah.edu

ABSTRACT

Advances in temporal planning have largely focused on improving the tractability of automated planning in temporally complex domains. However, in certain applications, it is necessary to reason about acceptable lower and upper bounds to a solution's temporal execution makespan. Planning with these novel execution constraints is similar to disjunctive temporal planning problems, which are EXPSPACE complete. We test our technique with two refinement strategies and compare their performance on modified IPC 2002 temporal benchmarks.

KEYWORDS

Disjunctive temporal planning, temporally-uncertain execution

ACM Reference Format:
Mica Gardone and Rogelio E. Cardona-Rivera. 2026. Disjunctive Temporal Refinement Planning with Variable Action Duration and Execution Makespan Bounds. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/ZRWF2944

1 MOTIVATION

Most temporal planning frameworks prioritize calculating the shortest duration (i.e., *makespan*) plan possible. These planners typically assume the best case scenario: the agent(s) can execute everything in the minimum amount of time as specified by the solution plan. However, there exist many different real world examples where "soonest possible" is not always feasible nor desirable, either by design or due to exogenous events.

For example, consider shipping [5, 7, 15]: there are extreme time constraints most notably on the shipping of perishable goods (e.g., food). A poorly executed plan that causes shipments to go over time leads to a loss of product and revenue due to perishing. However, the shortest duration plan might still cause a problem: delivering too soon will cause stores to reject shipments due to overstock, harming revenue and the environment by introducing a bottleneck [12, 14]. Further, these two extremes must be managed while considering that shipping activity times may be subject to high variance. Thus, how might we find temporal temporal plans with upper and lower makespan bounds, when we cannot anticipate how long actions will last?

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/ZRWF2944

The myriad of ways to attempt this under simple temporal planning [1–3, 8, 9, 13, 18] are not sufficient due to semantics of a solution being *only* the minimal makespan time. Simple Temporal Networks with Uncertainty (STNU) [11, 16, 17] are a logical next step; however, these focus on the execution of a plan, whereas we want the planner to generate a viable solution. This does not preclude that these dispatch techniques cannot be used in conjunction with our solution. Thus, we propose a new temporal planning framework for solving this kind of problem: **temporally-uncertain execution**.

2 THE PROBLEM SPACE

To situate the problem we want to solve, consider a Simple Temporal Problem (STP) [4]. STPs consider *only* the minimal temporal network, i.e., the minimum makespan of the plan to satisfy all goals. For a plan to be a solution to a STP requires that, in the minimal network, all conditions (on goals and actions) are satisfied and that no action threatens to undo established conditions. Execution-level constraints are not typically considered in simple temporal planners; thus, they only produce one solution that is viable according to the minimum temporal network.

The problem we are addressing is one where the executing agent (e.g., a robot, spaceship, autonomous truck) makes no guarantees about when it will actually finish an action, just that it will obey time constraints on said action. As such, the minimum makespan solution represents *one possible* execution; the executing agent might not adhere to it though. A solution to a STP might contain a threat that undoes a condition if an action or some actions take longer than expected; requiring the planner to re-plan when it could have been avoided in this scenario.

In order for a plan to be a solution in this problem, there must be *no potential* violation at any point in the plan. This requires reasoning about both the minimum *and* maximum temporal networks. Additionally, time-windows [13] now require more consideration: action durations might need to be *decreased* rather than simply increased as in STPs. We note that this is potentially an EXPSPACE-complete operation and that there are ways to maintain efficiency [10]; however we forgo this discussion for space.

3 EXTENSIONS TO PDDL

In order to support this, we made a few extensions to the PDDL 2 [6, 8] that can be ported to other versions of PDDL. We introduce a new flag, `:tuep`, to the parser that indicates that the following problem(s) need to consider the whole potential timeline of a plan. The type of problem can exist in both PDDL 2 and 3, and HDDL 2.1's temporal semantics.

Titan-OTC

	Depots			Driver Log			Rovers			Satellite		
Solution Delta	$\|S_N\|$	$\|S\|$	**Runtime**	$\|S_N\|$	$\|S\|$	**Runtime**	$\|S_N\|$	$\|S\|$	**Runtime**	$\|S_N\|$	$\|S\|$	**Runtime**
unbound	0	5	2.16	0	10	1.43	0	14	33.84	0	6	4.22
16	0	3	2.79	0	9	2.97	0	13	8.06	0	3	8.97
8	0	4	17.79	0	7	5.81	0	13	12.81	0	6	4.35
4	0	4	11.80	0	8	10.24	0	10	9.81	0	6	4.57
2	0	4	165.33	0	7	45.18	0	6	1.67	0	6	0.45
1	0	7	12.96	0	15	16.57	0	21	2.73	0	9	0.39
-1	0	12	8.17	0	28	12.07	6	35	11.54	3	11	5.07
-2	0	7	123.64	0	13	27.09	0	16	4.37	0	8	3.53
-4	0	7	92.83	0	14	9.38	2	20	15.34	0	9	6.15
-8	0	7	17.25	0	14	22.67	4	23	19.19	3	8	5.26
-16	0	4	8.42	0	14	11.38	6	16	12.40	3	6	8.97

Titan-TL

	Depots			Driver Log			Rovers			Satellite		
Solution Delta	$\|S_N\|$	$\|S\|$	**Runtime**	$\|S_N\|$	$\|S\|$	**Runtime**	$\|S_N\|$	$\|S\|$	**Runtime**	$\|S_N\|$	$\|S\|$	**Runtime**
unbound	0	5	2.17	0	10	1.45	0	14	26.05	0	6	4.60
16	0	4	7.18	0	6	7.50	0	13	36.20	0	3	8.41
8	0	3	2.85	0	6	5.23	0	10	24.67	0	6	4.24
4	0	3	2.44	0	6	16.94	0	7	96.94	0	6	4.26
2	0	3	2.44	0	3	2.89	0	6	1.97	0	6	11.51
1	0	7	4.95	0	13	144.71	0	19	2.02	0	9	11.37
-1	0	8	6.46	0	16	49.95	0	24	23.19	0	12	9.90
-2	0	4	4.00	0	5	4.52	0	12	31.89	0	9	10.50
-4	0	4	6.74	0	8	14.53	0	12	88.23	0	9	5.65
-8	0	3	2.85	0	8	4.79	0	12	24.28	0	8	7.31
-16	0	4	7.18	0	8	7.50	0	12	36.20	0	3	8.41

Table 1: `Titan` two different strategies: Overtime Critical (OTC) and Time-Last (TL). S_N is the number reported with no solution, not simply out of time. S is the number of solved problems. OTC solved overall more problems, but was slightly less performant than TL specifically when the "optimal" makespan was known. When the optimal makespan is unknown, OTC is more performant and capable of solving more problems than TL. OTC had several instances where it explored the available search space but found no solution. OTC is using a satisficing resolver so it is not be representative of the *full* search space.

```
(:makespan-constraint
    [
        ([>|>=|<|<=|=] <real>) |
        ((and ([>|>=] <real>) ([<|<=] <real>))
    ]
)
```

We introduce one new component to the problem section of PDDL: the `makespan-constraint`. This lets an engineer specify an overall constraint on the makespan of a solution. One can use `:tuep` and either `:timed-initial-literals` [6] (TILs, execution windows) or a PDDL 3 preference [9] (goal specific preferences) depending on the use-case. Using either a global TIL or the same time preference on all goals can simulate `makespan-constraint`, respectively, at the cost of introducing dummy predicates.

4 EXPERIMENT & DISCUSSION

We explored the impact of time-frames in four modified 2002 IPC domains and problems: `Satellite`, `Rovers`, `DriverLog`, and `Depots`. `Titan` was run in two refinement configurations: time-last (TL) and overtime critical (OTC). `Titan` was configured with a temporal relaxed planning graph heuristic. Makespan constraints were derived from finding a solution to the original problem's minimum makespan using both refinement configurations. We tested both around the minimum makespan ($\pm[1, 2, 4, 8, 16]$) and below the minimum makespan ($-[1, 2, 4, 8, 16]$). Both variations of the planner ran on a Windows 10 PC with an i7-9700K CPU and 32 GB of RAM under Windows Subsystem for Linux 2.

In Table 1, we can see that OTC solved more problems over all than TL. OTC performed as well as TL but better in most cases. The most common reason for TL to fail was related to running out of time. OTC was able to report when it had exhausted its search space. However, because `Titan` is configured with a satisficing solver, we cannot truly know there truly was no solution or not. For known optimal makespans, TL was generally faster than OTC. There are some instances where OTC is able to, generally, find a solution faster than but it is not by orders of magnitude like TL is. TL is generally faster in *known* optimal time-frames because it deals with temporal refinement last, which is a costly operation.

ACKNOWLEDGMENTS

This material is based on work supported by the United States National Science Foundation (Grant #2046294). We also thank our anonymous reviewers for their feedback during peer review. We would also like to thank Pablo Sauma-Chacón for his optimization insights.

REFERENCES

[1] S Chien, G Rabideau, R Knight, R Sherwood, B Engelhardt, D Mutz, T Estlin, B Smith, F Fisher, T Barrett, G Stebbins, and D Tran. 2000. ASPEN – Automated Planning and Scheduling for Space Mission Operations. In *Space Ops*, Vol. 82. Space Ops, 1–10.

[2] A. J. Coles, A. I. Coles, M. Fox, and D. Long. 2010. Forward-Chaining Partial-Order Planning. In *Twentieth International Conference on Automated Planning and Scheduling (ICAPS 10)*. AAAI Press, 42–49.

[3] Martin C. Cooper, Frederic Maris, and Pierre Régnier. 2010. Compilation of a High-level Temporal Planning Language into PDDL 2.1. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, Vol. 2. IEEE International Conference on Tools with Artificial Intelligence, 181–188. https://doi.org/10.1109/ICTAI.2010.99

[4] Rina Dechter, Itay Meiri, and Judea Pearl. 1991. Temporal constraint networks. *Artificial intelligence* 49, 1-3 (1991), 61–95.

[5] Rajesh Kumar Dhanaraj, Nilayam Kumar Kamila, Subhendu Kumar Pani, Balamurugan Balusamy, and Vani Rajasekar (Eds.). 2024. *Artificial intelligence for future intelligent transportation: smarter and greener infrastructure design* (first edition ed.). Apple Academic Press, Inc. ; CRC Press.

[6] Stefan Edelkamp and Jörg Hoffmann. 2004. *PDDL2.2: The language for the classical part of the 4th international planning competition.* Technical Report. Technical Report 195, University of Freiburg.

[7] Klaus Fischer, Jörg P. Müller, and Markus Pischel. 1996. Cooperative transportation scheduling: An application domain for dai. *Applied Artificial Intelligence* 10, 1 (Feb. 1996), 1–34.

[8] M. Fox and D. Long. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20 (Dec. 2003), 61–124.

[9] Alfonso Gerevini and Derek Long. 2005. *Plan constraints and preferences in PDDL3.* Technical Report. Technical Report 2005-08-07, Department of Electronics for Automation ….

[10] Nicola Gigante, Andrea Micheli, Angelo Montanari, and Enrico Scala. 2022. Decidability and complexity of action-based temporal planning over dense time. *Artificial Intelligence* 307 (2022), 9859–9866.

[11] Sriram Gopalakrishnan and Daniel Borrajo. 2022. Assignment and Prioritization of Tasks with Uncertain Durations for Satisfying Makespans in Decentralized Execution. *Proceedings of the International Conference on Automated Planning and Scheduling* 32, 1 (Jun. 2022), 119–123. https://doi.org/10.1609/icaps.v32i1.19792

[12] Dana Gunders and Jonathan Bloom. 2017. *Wasted: How America is losing up to 40 percent of its food from farm to fork to landfill.* Vol. 18. Natural Resources Defense Council New York.

[13] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, Christian Muise, Ronald Brachman, Francesca Rossi, and Peter Stone. 2019. *An introduction to the planning domain definition language.* Vol. 13. Springer.

[14] David L. Hummels and Georg Schaur. 2013. Time as a Trade Barrier. *American Economic Review* 103, 7 (2013), 2935–2959.

[15] Lakshmi Shankar Iyer. 2021. AI enabled applications towards intelligent transportation. *Transportation Engineering* 5 (Sept. 2021), 100083.

[16] Andrew Murray, Ashwin Arulselvan, Michael Cashmore, Marc Roper, and Jeremy Frank. 2023. A Column Generation Approach to Correlated Simple Temporal Networks. *Proceedings of the International Conference on Automated Planning and Scheduling* 33, 1 (Jul. 2023), 295–303. https://doi.org/10.1609/icaps.v33i1.27207

[17] Kevin Osanlou, Jeremy Frank, Andrei Bursuc, Tristan Cazenave, Eric Jacopin, Christophe Guettier, and J. Benton. 2022. Solving Disjunctive Temporal Networks with Uncertainty under Restricted Time-Based Controllability Using Tree Search and Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 36 (Jun. 2022), 9877–9885.

[18] H. L.S. Younes and R. G. Simmons. 2003. VHPOP: Versatile Heuristic Partial Order Planner. *Journal of Artificial Intelligence Research* 20 (Dec. 2003), 405–430.