Gradient-Based Mixed Planning with Symbolic and Numeric Action Parameters (Extended Abstract)[*]

Kebing Jin[1] , **Hankz Hankui Zhuo**[1] , **Zhanhao Xiao**[2] and **Hai Wan**[1] and **Subbarao Kambhampati**[3]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]School of Computer Science, Guangdong Polytechnic Normal University, Guangzhou, China
[3]Department of Computer Science and Engineering, Arizona State University, US
jinkb@mail2.sysu.edu.cn, zhuohank@mail.sysu.edu.cn, xiaozhanhao@gpnu.edu.cn, wanhai@mail.sysu.edu.cn, rao@asu.edu

Abstract

Dealing with planning problems with both logical relations and numeric changes in real-world dynamic environments is challenging. Existing numeric planning systems for the problem often discretize numeric variables or impose convex constraints on numeric variables, which harms the performance when solving problems, especially when the problems contain obstacles and non-linear numeric effects. In this work, we propose a novel algorithm framework to solve numeric planning problems mixed with logical relations and numeric changes based on gradient descent. We cast the numeric planning with logical relations and numeric changes as an optimization problem. Specifically, we extend the syntax to allow parameters of action models to be either objects or real-valued numbers, which enhances the ability to model real-world numeric effects. Based on the extended modeling language, we propose a gradient-based framework to simultaneously optimize numeric parameters and compute appropriate actions to form candidate plans. The gradient-based framework is composed of an algorithmic heuristic module based on propositional operations to select actions and generate constraints for gradient descent, an algorithmic transition module to update states to the next ones, and a loss module to compute loss. We repeatedly minimize loss by updating numeric parameters and compute candidate plans until it converges into a valid plan for the planning problem.

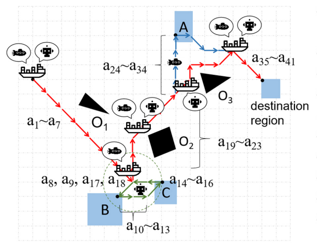
1 Introduction

Autonomous robots have become commonplace in commercial and industrial settings. For example, hospitals use autonomous mobile robots to move materials. Warehouses exploit mobile robotic systems to efficiently move materials from stocking shelves to order fulfillment zones. In scientific missions, Woods Hole Oceanographic Institution (WHOI) uses autonomous underwater vehicles (AUVs) to collect data of scientific interest. In those real-world applications, it is desirable to have autonomous robots be capable of autonomously planning with optimization of numeric objectives, such as minimizing resources, during planning towards desirable goals (e.g., in the form of propositions).

To handle numeric planning problems, there have been approaches [Eyerich *et al.*, 2009; Gerevini and Serina, 2002; Helmert, 2006; Hoffmann, 2001] proposed to discretize numeric space and then use heuristic searching. They, however, do not address planning missions over long-term reasoning for autonomous robots since the size of discretization needs to be fixed in advance manually. It is hard to determine a proper size of discretization beforehand for various planning problems with respect to different environments.*For example, in the ocean mission scenario as shown in Figure 1, a ship is equipped with an AUV (Autonomous Underwater Vehicle), i.e., the submarine in the figure, and an ROV (Remotely Operated Vehicle), i.e., the robot. The AUV aims to take images in Region A, and the ROV aims to take samples in Regions B and C. The planning mission is to make the three vehicles reach the destination region (the blue area denoted by "destination region"), with avoiding obstacles (the black areas). Each action has multiple parameters, where a movement is determined by three numeric ones, i.e., x-velocity v_x, y-velocity v_y, and duration d. In particular, if the ship deploys the ROV, which moves within a circle centered over the ship with a radius R, the ship needs to stay at the deployment location until the ROV comes back again. We use red arrows to indicate the trajectory of the ship, green arrows to indicate the trajectory of the ROV, and blue arrows to indicate the trajectory of the AUV. An example plan generated by Metric-FF can be found in Figure 1(a) (the corresponding action sequence is shown on the right). All the movements are fixed, owing to the requirements of fixed numeric effects. The transition of locations is done by dividing space beforehand and using propositions to indicate them.*As shown in Figure 1(a), each movement computed by Metric-FF is fixed, which indicates they cannot make up a flexible plan. The desired solution is to dynamically adapt step lengths according to the positions and shapes of obstacles when generating the plan, as the one shown in Figure 1(b).

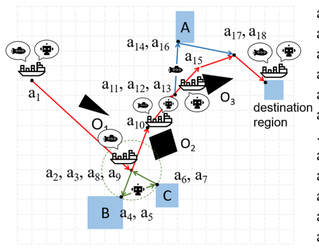
There are also approaches that introduce numeric vari-

[*]This paper is an extended abstract of the paper published in Artificial Intelligence, Volume 313, 2022.



a_1-a_7: ship-AUV-Navigate-down-right (ship ROV AUV)
a_8: Deploy-ROV (ship ROV)
a_9: ROV-Navigate-down-left (ship ROV)
a_{10}: ROV-Take-sample (ROV B)
...
a_{35}: AUV-Navigate-right (AUV)
a_{36}: AUV-Navigate-down-right (AUV)
a_{37}-a_{38}: AUV-Navigate-right (AUV)
a_{39}: Recover-AUV (ship AUV)
a_{40}-a_{41}: ship-AUV-Navigate-down-right (ship ROV AUV)

(a) Metric-FF



a_1: ship-AUV-Navigate (ship ROV AUV 3 3.5 2)
a_2: Deploy-ROV (ship ROV)
a_3: ROV-Navigate (ship ROV -1 -2 1)
a_4: ROV-take-sample (ROV B)
a_5: ROV-Navigate (ship ROV 1 2 1)
a_6: ROV-take-sample (ROV C)
...
a_{14}: AUV-take-images (AUV A)
a_{15}: ship-Navigate (ship ROV 3 1 0.8)
a_{16}: AUV-Navigate (AUV 4 -1 1)
a_{17}: Recover-AUV (ship AUV)
a_{18}: ship-AUV-Navigate (ship ROV AUV 1 -1 2)

(b) `mxPlanner`

Figure 1: Valid plans based on Metric-FF and `mxPlanner` in ocean mission scenario.

ables [Coles *et al.*, 2008; Keyder *et al.*, 2014; Ivankovic *et al.*, 2014] and control parameters [Li and Williams, 2008; Pantke *et al.*, 2014] into action models. POPCORN [Savas *et al.*, 2016] was built, using real number control parameters to allow action models with infinite domain parameters. It can, however, only be used in discrete numeric effects with constant rates of change, and only supports linear constraints. To relax the limitations of linear constraints, ScottyActivity [Fernández-González *et al.*, 2018] was developed to effectively generate plans for autonomous robots over long-term reasoning. ScottyActivity utilizes convex optimization to choose continuous states, control parameters, and times. Despite the success of ScottyActivity, it still requires the continuous search space to be convex, which in consequence does not allow "obstacles" in a navigation domain, making the search space non-convex. In many real-world applications, however, the continuous search space is often non-convex. It is undoubtedly challenging to solve planning missions that require both discrete action planning and non-convex continuous state searching.

In this paper, we extend the scope of the previous numeric planning problem by simultaneously allowing three properties: non-convex continuous numeric space, propositional and numeric action preconditions and effects, and more complicated action effects. We propose a novel approach, `mxPlanner` [Jin *et al.*, 2022], which stands for gradient-based **mi**x**ed** **Planner**, to deal with problems with logical relations and numeric changes in non-convex numeric space, determined by multiple symbolic and numeric action parameters. We build a gradient-based framework, composed of an algorithmic heuristic module, an algorithmic transition module, and loss functions. The effectiveness of using gradient-based planning has been proved in [Cui and Khardon, 2016; Bajpai and Garg, 2018; Garg *et al.*, 2019; Hafner *et al.*, 2020]. Specifically, the heuristic module estimates appropriate actions to form a plan σ to transit initial state s_0 to the goal. The transition module updates states according to plan σ. After that, we explore gradient descent to optimize numeric parameters to minimize accumulated loss computed by loss functions. We repeat the above-mentioned procedures of computing plans and optimizing numeric parameters until our approach converges to valid plans. Note that we do not only update the continuous numeric parameters of actions, but also modify the actions in σ simultaneously. Our `mxPlanner` is capable of handling flexible numeric changes determined by multiple numeric parameters of actions, instead of fixing numeric effects as done by Metric-FF. *For example, in the AUV domain shown in Figure 1, another plan calculated by our* `mxPlanner` *can be found from Figure 1(b). The navigation distance computed by* `mxPlanner` *is much shorter than the other one, since* `mxPlanner` *is more flexible than Metric-FF with fixed step length.*

The remainder of this paper is a high-level summary of [Jin *et al.*, 2022]. We summarize the contributions as follows:

- We extend numeric planning problems to simultaneously allow continuous numeric space to be non-convex, preconditions and effects to be both propositional and numerical, and numerical effects to be non-linear.
- To handle *mixed* planning problems, we propose a novel approach, `mxPlanner`, which borrows the framework of recurrent neural networks with the integration of algorithmic heuristic searching in the framework.
- We empirically show that our `mxPlanner` outperforms existing planners in solving *mixed* planning problems with non-convex numeric space (i.e., including obstacles) and complex preconditions and effects (involving propositions and numeric expressions determined by multiple action parameters). Besides, the experimental results show `mxPlanner` is also competitive when handling *mixed* planning problems with convex continuous numeric space.

2 Problem Definition

In this paper, we aim to solve sequential planning problems with both discrete logical relations and continuous numeric changes. Specifically, we aim to consider the planning problem that has the following features:

- The *continuous numeric space* (which is composed of a set of numeric variables in each state) is allowed to be non-convex; note that the objective based on the non-convex space is also non-convex.
- The effects of actions are allowed to be discrete (propositional operations), continuous (numeric changes) or both.
- The numeric effects are allowed to be non-linear, linear, or both.
- The numeric variables are allowed in preconditions of actions.

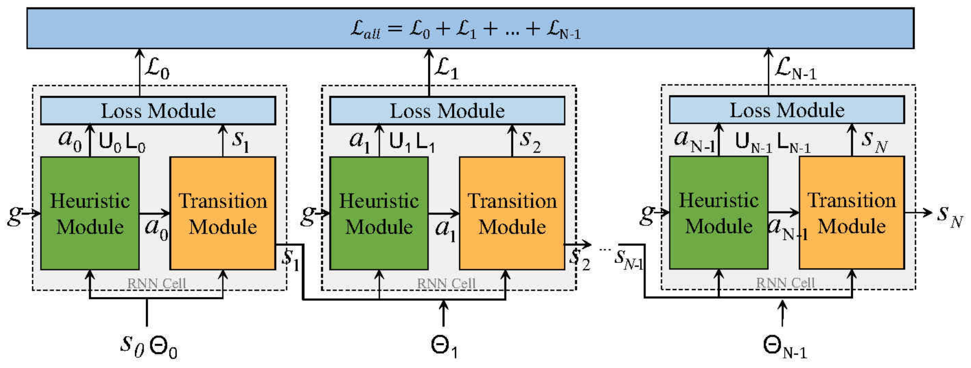


Figure 2: The Gradient-based framework of `mxPlanner`, which follows the RNN framework to perform forward simulation.

We call the above-mentioned planning problem as a *mixed* planning problem. Formally, we represent our mixed planning problem as a tuple $\mathcal{M} = \langle \mathcal{S}, s_0, g, \mathcal{A}, \mathcal{B} \rangle$, where $\mathcal{S}$ is a set of states, each of which is composed of a set of propositions and assignments of numeric variables in the prefix form. We define a *continuous numeric space* by the set of numeric variables. $\mathcal{A}$ is a set of action models. An action is a grounding of an action model, i.e., every parameter in the action model is an object or a real number. Specifically, we use a vector $\Theta = \langle \Theta_0, \Theta_1, \dots, \Theta_{N-1} \rangle$ to denote all numeric parameters occurring in a plan σ with N steps, where N is the maximal length of the potential solution plan. $\Theta_i = \langle \theta_i^1, \theta_i^2, \dots, \theta_i^T \rangle$ is all of the numeric parameters of actions in step i. T is the number of different numeric parameters of all actions in $\mathcal{A}$. Note that the parameters of actions are different from the set of variables in states. An action model or action is called *numeric* if it has numeric updating effects; otherwise, it is called *logical*, indicating that the action only includes logical operations. An action is applicable in a state if its precondition is satisfied by the state. $\mathcal{B}$ is an interval to constrain numeric parameters, which is defined by $\mathcal{B} = [\underline{\mathcal{B}}, \overline{\mathcal{B}}]$. Besides, we use $\psi(a)$ to denote the cost of action a. In this paper, we define the cost of action as the distance yielded by it. We define the cost of a plan σ as the sum of the cost of all actions in σ, i.e., $C(\sigma) = \sum_{a_i \in \sigma} \psi(a_i)$.

Given a mixed planning problem $\mathcal{M}$, we aim at computing a plan $\sigma = \langle a_1, a_2, \dots, a_n \rangle$ to achieve g from s_0 with the minimal cost. Different from classical planning problems, in this paper, we require to not only find appropriate actions but also determine their continuous numeric parameters. We give a formal form of expression of *mixed* planning problems in [Jin *et al.*, 2022] and compare it with PDDL 2.1 [Fox and Long, 2003] and PDDL+ [Fox and Long, 2006].

3 An Overview of `mxPlanner`

A framework of `mxPlanner` is shown in Figure 2 in the form of unfolded RNN Cells. Each RNN Cell represents a step in a plan, which is composed of a heuristic module, a transition module, and a loss module. The heuristic module is an algorithm to output an action, the transition module is used to update a state according to action models, and the loss module is composed of functions to calculate losses. It is notable that we just "borrow" the framework of RNN and take advantage of its features about sharing the same RNN Cell across all steps. The heuristic module and the transition module are implemented as algorithmic procedures or functions instead of neural networks. Using the RNN framework offers an efficient way to build sequential plans via gradient descent. Note that, different from RNNs, we optimize the input numeric parameters with gradient descent, rather than weights of neural networks in RNN Cells, as done by [Wu *et al.*, 2017], which have been replaced by three modules in this paper. We assume the number of RNN cells, N, is sufficiently large to compute a plan. The overall procedure of computing a plan is as shown below:

1. We first randomly initialize numeric parameters Θ of actions which will be optimized by the subsequent steps.
2. The heuristic module, which is based on the relaxed planning graph [Blum and Furst, 1997] and the interval-based relaxation [Scala *et al.*, 2016], takes a state s_i, numeric parameters Θ_i, and the goal g as inputs, and outputs an action a_i, upper bound vector U_i and lower bound vector L_i of numeric variable values. The upper bound vector and lower bound vector are two real number vectors with the size of the number of numeric variables. These two bound vectors designate the value ranges of all numeric variables in the next state.
3. The transition module, which essentially is a transition function γ, takes a state s_i, action a_i, and numeric parameters of actions Θ_i in a_i as inputs, and outputs the next state s_{i+1}.
4. The loss module, which is composed of loss functions, takes an action a_i, upper bound vector U_i, lower bound L_i and state s_{i+1} as inputs, and outputs the loss of the step $\mathcal{L}_i$. $\mathcal{L}_i$ includes three parts: $\mathcal{L}_{b_i}$ for guiding states to fall within the required bounds, $\mathcal{L}_{o_i}$ for guiding agents avoiding obstacles, and $\psi(a_i)$ for minimizing costs. We calculate the total loss $\mathcal{L}_{all}$ by accumulating losses from all of the RNN Cells.
5. We inversely optimise numeric parameters in each step $\Theta_0, \Theta_1, \dots, \Theta_{N-1}$ by minimizing $\mathcal{L}_{all}$. It is noted that,

after updating numeric parameters, the transferred states at each step are also updated in the next iteration, along with the numeric parameters updated. Hence, the actions computed by the heuristic module may be different in the following iterations. With the values of numeric parameters updated, the heuristic module repeatedly estimates actions based on current updated states until a plan with minimal cost is achieved.

6. We repeat the second to the fifth steps until we find a valid plan σ. Note that actions in σ are not fixed and they can be changed during iterations. That is, we do not only update numeric parameters but also update actions at the same time.

4 Results

We briefly introduce the experiments in [Jin *et al.*, 2022], which are based on three domains, i.e., AUV, Taxi, and Rover. We compare with `mxPlanner` with three state-of-the-art planners with modifications, Metric-FF [Hoffmann, 2003], POPCORN [Savas *et al.*, 2016], and ScottyActivity [Fernández-González *et al.*, 2018] with different settings. We first evaluate `mxPlanner` with respect to four aspects in convex and non-convex problems with different parameter bounds settings. Then we test the performance, i.e., costs and iterations, of `mxPlanner` with different hyperparameters. At last, we evaluate the running time of `mxPlanner`, Metric-FF, ScottyActivity, and the adapted POPCORN with respect to different parameter bounds, and analyze the merits and demerits of `mxPlanner` with comparison to other approaches based on four cases. Here, we describe two experimental results where the remaining details are in [Jin *et al.*, 2022].

Figure 3 shows the results of average costs with different parameter bounds in convex problems, i.e. including no obstacles. As shown in the Figure, `mxPlanner` is competitive. The performance of `mxPlanner` is similar to ScottyActivity, which utilizes convex optimization and has the ability to compute plans of high quality for some simple instances. The average plan costs of Metric-FF$^+$ are the largest in most cases of the three domains. Because the discretization of Metric-FF$^+$ lets agents move more compared with the planners using an optimization algorithm.

Another experiment is shown in Figure 4, which shows two example plans in the AUV domain with obstacles computed by Metric-FF and `mxPlanner`. `mxPlanner` dynamically adjusts the step lengths and searching angles in each step to generate flexible plans, instead of fixing step lengths determined by prior knowledge, as done by Metric-FF.

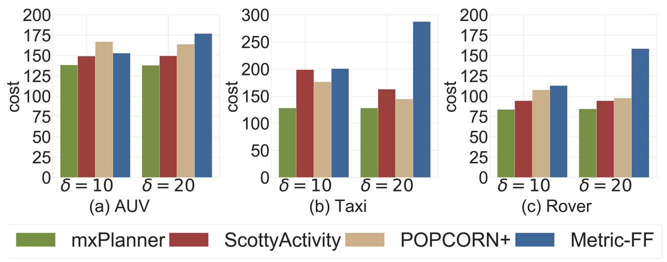


Figure 3: Average cost of obstacle-free instances in the three domains with different parameter bounds.

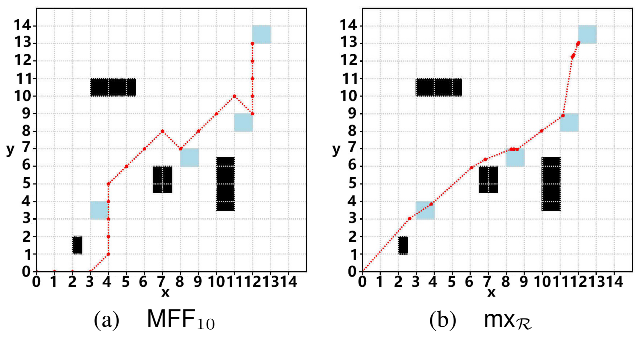


Figure 4: Two example plans in the AUV domain computed by Metric-FF and `mxPlanner`, respectively. The bold red dot is the stop position of each movement. Blue areas are target regions and black areas are obstacles.

5 Conclusion

In this paper, we introduce *mixed* planning problems, which are extended from numeric planning problems with control parameters. We present `mxPlanner`, a gradient-based approach that borrows the framework of RNNs by replacing the neural cells with heuristic search and transition modules. We evaluate `mxPlanner` in three domains and the experimental results show the superiority of `mxPlanner` on plan quality, especially in obstacle avoidance problems compared against state-of-the-art approaches. Also, we evaluate the influence of the hyperparameters on `mxPlanner`. The combination of heuristic searching and gradient-based framework gives `mxPlanner` the ability to handle *mixed* planning problems without discretization. Especially, `mxPlanner` performs well when handling *mixed* planning problems with non-convex continuous numeric space, i.e., containing obstacles. In this paper we assume the logical relations in action models are given beforehand. In the future it would be interesting to investigate embedding action model learning approaches [Zhuo and Kambhampati, 2017; Zhuo and Yang, 2014; Zhuo *et al.*, 2014; Zhuo *et al.*, 2010]

Acknowledgements

This research was funded by the National Natural Science Foundation of China (Grant No. 62076263, 61906216), Guangdong Natural Science Funds for Distinguished Young Scholar (Grant No. 2017A030306028), Guangdong Special Branch Plans Young Talent with Scientific and Technological Innovation (Grant No. 2017TQ04X866), Guangdong Basic and Applied Basic Research Foundation (2020A1515010642). Kambhampati's research is supported in part by the ONR grants N00014-16-1-2892, N00014-18-1-2442, N00014-18-1-2840, the AFOSR grant FA9550-18-1-0067, and the NASA grant NNX17AD06G.

References

[Bajpai and Garg, 2018] Aniket Nick Bajpai and Sankalp Garg. Transfer of deep reactive policies for mdp planning. *Advances in Neural Information Processing Systems*, 31, 2018.

[Blum and Furst, 1997] Avrim Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artif. Intell.*, 90(1-2):281–300, 1997.

[Coles *et al.*, 2008] Andrew Coles, Maria Fox, Derek Long, and Amanda Smith. A hybrid relaxed planning graph'lp heuristic for numeric planning domains. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, pages 52–59, 2008.

[Cui and Khardon, 2016] Hao Cui and Roni Khardon. Online symbolic gradient-based optimization for factored action mdps. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3075–3081. IJCAI/AAAI Press, 2016.

[Eyerich *et al.*, 2009] Patrick Eyerich, Robert Mattmüller, and Gabriele Röger. Using the context-enhanced additive heuristic for temporal and numeric planning. In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling*. AAAI, 2009.

[Fernández-González *et al.*, 2018] Enrique Fernández-González, Brian C. Williams, and Erez Karpas. Scottyactivity: Mixed discrete-continuous planning with convex optimization. *J. Artif. Intell. Res.*, 62:579–664, 2018.

[Fox and Long, 2003] Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[Fox and Long, 2006] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.*, 27:235–297, 2006.

[Garg *et al.*, 2019] Sankalp Garg, Aniket Bajpai, et al. Size independent neural transfer for rddl planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 631–636, 2019.

[Gerevini and Serina, 2002] Alfonso Gerevini and Ivan Serina. LPG: A planner based on local search for planning graphs with action costs. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*, pages 13–22, 2002.

[Hafner *et al.*, 2020] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res.*, 26:191–246, 2006.

[Hoffmann, 2001] Jörg Hoffmann. FF: the fast-forward planning system. *AI Magazine*, 22(3):57–62, 2001.

[Hoffmann, 2003] Jörg Hoffmann. The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *J. Artif. Intell. Res.*, 20:291–341, 2003.

[Ivankovic *et al.*, 2014] Franc Ivankovic, Patrik Haslum, Sylvie Thiébaux, Vikas Shivashankar, and Dana S. Nau. Optimal planning with global numerical state constraints. In Steve A. Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*. AAAI, 2014.

[Jin *et al.*, 2022] Kebing Jin, Hankz Hankui Zhuo, Zhanhao Xiao, Hai Wan, and Subbarao Kambhampati. Gradient-based mixed planning with symbolic and numeric action parameters. *Artif. Intell.*, 313:103789, 2022.

[Keyder *et al.*, 2014] Emil Ragip Keyder, Jörg Hoffmann, and Patrik Haslum. Improving delete relaxation heuristics through explicitly represented conjunctions. *J. Artif. Intell. Res.*, 50:487–533, 2014.

[Li and Williams, 2008] Hui X. Li and Brian C. Williams. Generative planning for hybrid systems based on flow tubes. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, pages 206–213, 2008.

[Pantke *et al.*, 2014] Florian Pantke, Stefan Edelkamp, and Otthein Herzog. Planning with numeric key performance indicators over dynamic organizations of intelligent agents. In Jörg P. Müller, Michael Weyrich, and Ana L. C. Bazzan, editors, *Multiagent System Technologies - 12th German Conference*, pages 138–155. Springer, 2014.

[Savas *et al.*, 2016] Emre Savas, Maria Fox, Derek Long, and Daniele Magazzeni. Planning using actions with control parameters. In *ECAI*, pages 1185–1193, 2016.

[Scala *et al.*, 2016] Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramírez. Interval-based relaxation for general numeric planning. In *ECAI*, pages 655–663, 2016.

[Wu *et al.*, 2017] Ga Wu, Buser Say, and Scott Sanner. Scalable planning with tensorflow for hybrid nonlinear domains. In *NIPS*, pages 6273–6283, 2017.

[Zhuo and Kambhampati, 2017] Hankz Hankui Zhuo and Subbarao Kambhampati. Model-lite planning: Case-based vs. model-based approaches. *Artif. Intell.*, 246:1–21, 2017.

[Zhuo and Yang, 2014] Hankz Hankui Zhuo and Qiang Yang. Action-model acquisition for planning via transfer learning. *Artif. Intell.*, 212:80–103, 2014.

[Zhuo *et al.*, 2010] Hankz Hankui Zhuo, Qiang Yang, Derek Hao Hu, and Lei Li. Learning complex action models with quantifiers and logical implications. *Artif. Intell.*, 174(18):1540–1569, 2010.

[Zhuo *et al.*, 2014] Hankz Hankui Zhuo, Héctor Muñoz-Avila, and Qiang Yang. Learning hierarchical task network domains from partially observed plan traces. *Artif. Intell.*, 212:134–157, 2014.