Towards Robust Constraint Satisfaction in Hybrid Hierarchical Planning[*]

Tobias Schwartz, Michael Sioutis, Diedrich Wolter
University of Bamberg
An der Weberei 5
Bamberg
{tobias.schwartz, michail.sioutis, diedrich.wolter}@uni-bamberg.de

Abstract

Hybrid planning is essential for real world applications, as it allows for reasoning with different forms of abstract knowledge, such as time, space or resources. This unavoidably leads to a combinatorial explosion of the search space that has previously been tackled using a hierarchical task network (HTN) planning approach. Existing HTN planners mostly focus on finding a solution as fast as possible, with only recent work considering length-optimal solutions. In real world scenarios, it can easily happen that the environment changes before a plan is fully executed. We are motivated to conduct planning in such a way that the solution has the best chance of withstanding such changes in the environment. We call this ability the robustness of a solution. Defining robustness, however, is an inherently difficult challenge, as many different forms and notions exist. In this paper, we start the discussion by outlining a possible notion of robustness recently introduced in the light of Qualitative Spatial Reasoning (QSR) within the scope of hybrid hierarchical planning.

Introduction

We are interested in the question of how to do robust planning in the face of a dynamic execution environment. Existing planners are able to quickly derive a plan given a valid domain and problem file. In the real world, however, we deal with a high degree of uncertainty. Some events may take longer than anticipated or other (dependent) objects in the scene may have moved and thus are not available on the same conditions as during the initial planning phase.

Traditionally, classical planning is concerned with finding a sequence of actions that lead to a desired goal state (Ghallab, Nau, and Traverso 2016). Actions for this matter are defined in a specific domain file, stating at least their preconditions and effects. Intuitively, an action a is only applicable in a state s if all its preconditions are satisfied. However, in practical applications it is usually not enough to only state static preconditions and effects and indirectly impose an ordering constraint on the possible actions. Other aspects, such as a sense of time, the spatial characteristic of the scenario, physical restrictions in movements or resource limitations often have to be considered.

In order to reason about all those different classes of knowledge, the notion of *meta-CSP* has been introduced (Mansouri and Pecora 2016). It formulates an abstract high-level Constraint Satisfaction Problem (CSP) which is solved using a meta-reasoner combining dedicated reasoners for each of the different types of knowledge. All different viewpoints (e.g., spatial, temporal, or resource) can then be modeled within one reasoning framework by defining a constraint in the meta-CSP. One challenge in such a meta-CSP is the combinatorial explosion of the search space. To still use this approach in planning, a hierarchical planning approach has been considered to systematically reduce the scope and computational complexity (Stock et al. 2015).

Hierarchical Task Network (HTN) planning differs from classical planning in that it distinguishes between primitive and compound (or abstract) tasks. Instead of defining a specific goal state and let the planner find applicable actions leading to a plan, in HTN planning a (series of) compound task(s) is given and then decomposed into executable primitive actions. The planner here tries to find an applicable decomposition. Many extensions and varying realizations within the hierarchical planning framework have been considered (cf. Bercher, Alford, and Höller (2019)).

Like Stock et al. (2015), we are motivated to apply abstract reasoning in the context of HTN planning. Stock et al. (2015) transform the HTN planning problem into a CSP by encoding causal links of the HTN problem as so-called meta-constraints into the aforementioned meta-CSP. Note that this is similar to the combination of HTN planning with Partial Order Causal Link (POCL) planning (Schattenberg 2009; Bercher et al. 2016), which builds into many current hierarchical planners (Bercher 2021).

The meta-CSP formulation allows for abstract reasoning not only within the task network, but for example also about space, time and resource constraints. Reasoning about abstract spatial and temporal information is usually done using Qualitative Constraint Networks (QCNs) (Ligozat 2013; Dylla et al. 2017). In this context, Sioutis, Long, and Janhunen (2020) recently studied a notion of robustness, which concerns the perturbation tolerance of QCN solutions, i.e., their likelihood to resist a change in the environment.

[*]We would like to thank the anonymous reviewers for their insightful feedback. This research is partially supported by BMBF AI lab dependable intelligent systems.

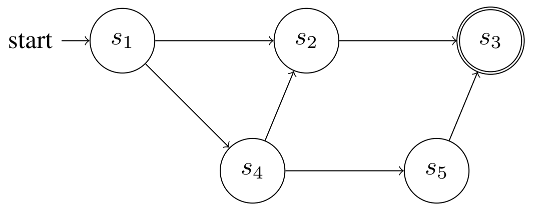

Figure 1: A train network with five stations and multiple paths leading from the start s_1 to the destination s_3

Robustness itself certainly is not a new concept, and can probably be traced back even to the first algorithms for problem solving, as with different methods to obtain a solution to a given problem, there was also the need to compare those solutions on a (usually robustness-related) basis (see for example Ginsberg, Parkes, and Roy (1998); Verfaillie and Jussien (2005)). Still, the work by Sioutis, Long, and Janhunen (2020) was the first time robustness as a measure was considered in QCNs. Likewise, to the best of our knowledge, the same holds for constrained reasoning in the context of hybrid hierarchical planning, where no measures of robustness have been established so far.

This paper contributes by pointing out the challenge of defining robustness in the scope of hierarchical planning, starting a discussion towards more robust solutions in HTN planning. We demonstrate our motivation and its usefulness in the context of a train routing problem. Following the work by Mansouri and Pecora (2016) and Stock et al. (2015) on modelling HTN planning in a meta-CSP, we establish similarities to QCNs and indicate how the notion of robustness from Sioutis, Long, and Janhunen (2020) may be applied. This may be seen as a first starting point for further work on robustness in hierarchical planning.

Robustness

Let us consider the simple train network depicted in Fig. 1, which, for the sake of our example, encodes the following planning task: Given n trains $t_1, \dots, t_n$ with the goal to drive from station s_1 (start) to s_3 (destination). Driving a train from one station to another is a complex task on its own, where multiple signals have to be adhered, speed has to be adjusted accordingly, the tracks have to be monitored, etc. (see Cardellini et al. (2021) for a recent formulation of a similar problem in PDDL+). For the sake of the example, we are for now abstracting from the details of said driving process and only focus on the high-level decisions of which train has to drive to which station at what time. We are essentially reducing the problem such that given n trains in a specified order, we try to find a viable route for each train trough the network from the start to the destination. As additional constraint, trains may only traverse in the direction indicated by the arrow, i.e. reversing is not permitted.

Assuming a constant driving time between stations and enough distance between incoming trains, the fastest and thus likely preferred choice would be to use the direct path via only station s_2 for all trains. The problem with this route is that once started at s_1, there is no turning point and no alternative route to switch to. If there occurs any form of delay on a preceding train on this route, this unavoidably affects all other trains. Likewise, if a part of the track between s_2 and s_3 has to close temporarily, e.g., because of an animal accident, no alternative path exists.

Uncertainty in the plan execution is the reason why current work in railway planning increasingly focuses on robustness in the quality of solutions to a given planning problem, such as the one outlined above. Here, a plan may be considered robust if it is able to withstand such unexpected changes in the environment. Accordingly, robustness can be defined as the ability of a system to resist change (Verfaillie and Jussien 2005; Lusby, Larsen, and Bull 2018). Note, however, that based on the context also other notions of robustness exist. One can further differentiate between flexible and robust solutions to a problem (Verfaillie and Jussien 2005; Muise 2014). A flexible solution is anything that can quickly generate a new solution in case of change, whereas a robust solution has every chance to resist all possible changes given by a model. Hereby, it is not only important how possible changes are modeled (qualitative or quantitative, probabilistic or not), but also when those changes are available to a planner (before or during plan execution). Depending on those characteristics, different notions of robustness may be applicable.

For example, reacting dynamically to a sudden change in the environment, such as a technical disturbance that prevents a train from reaching its next station, requires the use of some form of execution monitoring (Fritz 2009; Muise 2014). Thereby, it is actively observed whether the executed plan remains *valid*, given what is known in the current state. Muise (2014) defines a plan in classical planning valid, iff the plan is executable in the initial state I and results in the goal state G. In contrast to such a sequential plan, the least-commitment approach followed in partial-order planning promises more flexibility during execution, as some ordering choices can be delayed until run-time. A partial-order plan (POP) is valid iff every linearization achieves the goal from the initial state, clearly a strong requirement if we are only interested in finding a possible way to achieve the goal. More practically, a POP is called *viable* iff there exists a linearization that achieves the goal, which may be efficiently found by exploiting state *relevance*. Robustness may then be achieved by actively monitoring the execution of the plan and always selecting the most relevant partial plan fragment for achieving the goal.

Similarly, in the context of hierarchical planning, Patra et al. (2020) apply a UCT-like Monte-Carlo tree search procedure called UPOM, an online planner for the Refinement Acting Engine (RAE) (Ghallab, Nau, and Traverso 2016), to guide the selection of a method instance in case multiple ones for a task exist. The RAE can not only accomplish tasks but, like execution monitoring, also react to external events.

While the outlined planning approaches can consider changing environments and to an end possess the capability to include robustness, they may only be able to really include one notion, e.g., probabilities or qualitative information might not be considered. Also, they do not directly incorporate robustness as a measure before plan execution.

Case Study: Meta-CSPs

In what follows, we elaborate on a notion of robustness in the context of hybrid hierarchical planning using meta-CSPs, drawing on prior work by Sioutis, Long, and Janhunen (2020) in the context of QCNs. We start by briefly presenting some frameworks that are relevant to our discussion and that we will be referring to throughout the paper.

Constraint Satisfaction Problems

We adopt the standard notation of a CSP from Russell and Norvig (2020). A CSP is a tuple (X, D, C) where

- X is a set of variables $\{x_1, \ldots, x_n\}$, where each variable $x_i \in X$ corresponds to exactly one domain $D_i \in D$,
- and C is a set of constraints that restrict possible value assignments to variables.

An assignment that does not violate any constraints is called a consistent assignment. Solving a CSP now is the task of finding a consistent, complete assignment for all variables. A CSP is often visualized using a constraint graph $G = (V, E)$, where the vertices V represent variables and the edges E define constraints between any two variables.

Qualitative Constraint Networks

Reasoning on infinite domains, such as space and time, is typically done using Qualitative Constraint Networks (QCNs) (Ligozat 2013; Dylla et al. 2017). Similar to a constraint graph for a CSP, a QCN is a network where the vertices represent spatial or temporal entities and the edges are labeled with qualitative spatial or temporal relations, based on a finite set of *jointly exhaustive and pairwise disjoint* relations, called the set of base relations B.

Hybrid Planning

Autonomous systems, such as robots, typically operate in dynamic environments. Planning in such an environment is particularly difficult as many different forms of knowledge, such as temporal, causal or spatial information and constraints, have to be considered. Motivated by those needs of real world applications, hybrid planning methods have been studied. Hybrid planning in this context describes a classical planner that instead of focusing on one particular type of constraint can reason with multiple classes of knowledge. Note that the notion of hybrid planning has also been used in the context of combining hierarchical with state based Partial Order Causal Link (POCL) planning (Schattenberg 2009; Bercher et al. 2016).

For hybrid reasoning, Mansouri and Pecora (2016) have proposed the use of a so-called *meta-CSP*. Here, fluents are used to represent causal, temporal, spatial or resource semantics. A fluent may be used to represent an action (e.g. "drive") or display the current situation ("a train is at the station"). Given a set of fluents $\mathcal{F}$ and a set of constraints $\mathcal{C}$ among the fluents in $\mathcal{F}$, we can define a constraint network as the pair $(\mathcal{F}, C)$. Note that this is similar to the notation used for a CSP, with fluents being the variables of a heterogeneous set of domains. Now, we can cast the problem of finding a feasible plan as a *meta-CSP*, i.e. a high-level CSP that captures the heterogenous information of the overall problem, and is defined as a collection of meta-constraints. A meta-constraint is a triple (M, Ξ, Δ), where $M = (\mathcal{F}, C)$ is a constraint network, Ξ is a set of meta-variables $\{\xi_1, \ldots, \xi_n\}$, each of which is a subnetwork of M, i.e. $\xi_i = (\mathcal{F}_i \subseteq \mathcal{F}, C_i \subseteq C)$, and $\Delta = \{\delta(\xi_1), \ldots, \delta(\xi_n)\}$ is a set of domains, one for each meta-variable.

Encoding a classical planning problem as a CSP has already been studied in the literature (Barták, Salido, and Rossi 2010). The intuition is that we can restrict the length of a possible plan and then subsequently increase this limit until a plan is found[1]. The planning problem can then be modeled as a series of boolean satisfiability (SAT) problems, where each SAT instance is the problem of finding a plan of a given length. Those instances can be encoded as CSP.

Mansouri and Pecora (2016) represent actions as operators, defined as the pair $(f, (\mathcal{F}, C))$, where $f = (A, \cdot, \cdot, u, \cdot)$ is a fluent indicating that action A is being executed, $\mathcal{F}$ describes the set of fluents including the set of precondition fluents $\mathcal{F}_p$ and both negative effect fluents $\mathcal{F}_- \in \mathcal{F}_e$ and positive effect fluents $\mathcal{F}_+ \in \mathcal{F}_e$; and, finally, C is a set of causal (CC), temporal (TC), spatial (SC), and symbolic (BC) constraints on $\mathcal{F} \cup \{f\}$. Consider following example based on the train problem outlined in Fig. 1, using the notations introduced by Stock (2016):

$$\begin{aligned}
f &= (!\mathit{driveTo}(?t_1, ?s_2), [0, \infty], [0, 30], u(\mathit{track}1) = 1)\\
F_p &= \{f_1 = (\mathit{At}(?t_2, ?s_1), \cdot, \cdot)\}\\
F_- &= \{f_1\}\\
F_+ &= \{f_2 = (\mathit{At}(?t_3, ?s_3), \cdot, \cdot)\}\\
CC &= \{f_1 \text{ pre } f, f \text{ closes } f_1, f \text{ opens } f_2, f \text{ planned } f\}\\
BC &= \{S_1^{(f)} = S_1^{(f_1)}, S_1^{(f)} = S_1^{(f_2)}, S_2^{(f)} = S_2^{(f_2)}\}\\
TC &= \{I^{(f)} \mathit{ oi } I^{(f_1)}, I^{(f)} \mathit{ fi } I^{(f_2)}\}
\end{aligned}$$

A certain task `driveTo` requests a train t_1 to drive from its current location (s_1) to the station s_2 and finish this task no later than at time 30, i.e. arrive within 30 minutes. Additionally, driving here requires a resource of track1. Also, we model a set of causal and temporal constraints between the fluents and define symbolic constraints stating for example that the symbolic variables $?t_1$, $?t_2$, and $?t_3$ represent the same train (i.e. t_1). All constraints are added to and considered within the general constraint network of the meta-CSP and, thus, allow for joined reasoning over all available information. A plan is then only feasible iff the constraint network is consistent regarding all sources of input, e.g. it is temporally, symbolically, causally and resource consistent. Sophisticated reasoners for each domain may be used.

While this framework allows for potentially straightforward addition of all kinds of knowledge, this comes at a high computational cost. An intuitive solution is thus to employ some heuristics. For example, Stock et al. (2015) propose a number of variable ordering heuristics to potentially boost efficiency in the CSP search.

[1]As Barták, Salido, and Rossi (2010) we here assume that a plan always exists.

Hierarchical Hybrid Planning with Meta-CSPs

Hierarchical planning extends classical planning by introducing a task hierarchy. Instead of only using the notion of applicable actions, it essentially differentiates between primitive and compound tasks. Primitive tasks are hereby comparable to the normal actions in classical planning. Compound tasks describe a more abstract notion of a set of actions. This grouping can impose additional restrictions that might not be easily achievable using only preconditions and effects of actions. For example, an imposed ordering constraint can be easily encoded in a compound task and drastically improve efficiency of the planner. In fact, ordering tasks according to a partial order can be seen as the motivation behind HTN planning, the most influential subarea of hierarchical planning (Bercher, Alford, and Höller 2019).

In what follows, we briefly recall the definitions for HTN planning as defined by Bercher, Alford, and Höller (2019). The basis for HTN planning is the so called *task network*, which essentially imposes a strict partial order on a finite set of *tasks* T. The HTN *planning domain* D is defined as a tuple (F, N_P, N_C, δ, M), where

- F is a finite set of facts,
- N_P and N_C are names of primitive and compound tasks, respectively,
- $\delta : N_P \rightarrow 2^F \times 2^F \times 2^F$ maps actions to primitive task names,
- and M is a finite set of decomposition methods.

Finally, a HTN problem P is a tuple (D, s_I, tn_I), where D is the planning domain, $s_I \in 2^F$ is the initial state, and tn_I is an initial task network. A *solution* to this problem is then a final task network tn_S which is reachable from s_I by only applying methods and compound tasks. In the process, all compound tasks need to be decomposed into primitive actions, such that tn_S does not contain compound tasks anymore. The enforced task hierarchy directly restricts the set of possible solutions to only those that can be obtained by task decomposition (Bercher, Alford, and Höller 2019).

Following the formulation by Bercher et al. (2016), compound tasks may have their own set of preconditions and effects. These may be modeled using *causal links* as seen in POCL planning. Informally, a causal link is used to impose a direct ordering between two tasks by linking the effects of the former to the preconditions of the latter task, such that no other task may be allowed to be ordered between them. In a hierarchical setting, we may pass down causal links to all appropriate subtasks.

Given a meta-CSP as described above, it is now possible to encode the ordering relations imposed by the task network as causal constraints into meta-constraints. Meta-variables can then include all unplanned tasks whose predecessor tasks, indicated by the ordering constraint, have already been planned (Stock et al. 2015). Finding applicable tasks, i.e., the planning process, is then modelled as a CSP search, such as backtracking search.

Robustness in Meta-CSPs

Following the aforementioned definition of robustness, in the context of HTN planning, we are interested in a configuration that given one ore more tasks, has the ability to retain its feasibility more than any other in the case where some of the facts in the world have changed. In other words, we are interested in performing all tasks using a plan that has higher chance than any other to remain viable after changes in the environment occur. We call such a plan a *robust plan*, a plan with the maximum ability of resisting and avoiding infeasibility. Therefore, a robust plan can be seen as a *proactive measure* that limits as much as possible the need for successive repairs and replanning, and hence can play an important role in environments that are prone to perturbation and unexpected change, such as real-life configurations.

To further detail how robustness and dynamic reasoning can play a role in planning, let us reconsider the simple train network from Fig. 1. For convenience let us define all three possible routes from s_1 to s_3 as $r_1 = [s_1, s_2, s_3]$, $r_2 = [s_1, s_4, s_2, s_3]$, and $r_3 = [s_1, s_4, s_5, s_3]$. We can accomplish the task of driving a train to a specific destination with hierarchical planning using an abstract task like `driveTo(?train, ?destination)`. In case there is no direct connection, the task may be decomposed recursively to build a path from start to finish.

As mentioned before, when converting the whole planning task into a meta-CSP, the ordering of the tasks is encoded via causal constraints, where only the configurations representing the possible routes in the network are satisfiable scenarios. Following Sioutis, Long, and Janhunen (2020), we can calculate the *similarity* between any one particular solution of the meta-CSP and all other satisfiable ones. A robust scenario is then one with maximum average similarity to all other scenarios. Intuitively, a robust scenario on average shares the largest set of constraints with each other satisfiable scenario. In contrast to execution monitoring (Fritz 2009; Muise 2014), such a notion of robustness can guide the planning process actively and thus acts as a *proactive* measure, whereas the former mostly describes a *reactive* method. In this regard, the outlined proactive robustness formulation may best be compared with the measure of *relevance* guiding the search of Muise (2014).

So far, the focus in HTN planning has been mostly on finding a solution as quickly as possible, and only recent work considered length-optimal plans (Behnke, Höller, and Biundo 2019). One promising direction in this regard is the UPOM planner (Patra et al. 2020), where different utility functions can be optimized. While currently only efficiency is considered, it may also be possible to directly integrate a measure of robustness.

While it is difficult to say with certainty which path any current hierarchical planner would follow in the outlined train routing example, it seems more likely that a majority might choose the shortest option (i.e. route r_1), not because it is the length-optimal plan, but rather because its decomposition depth is also smaller than any of the alternative routes. Unless specified otherwise with some constraints, subsequent trains all follow the same route. In terms of minimizing the travel time, this might be a preferred configuration. From a collision avoidance perspective, it might be favorable to split traffic onto all available routes. And solving the problem conservatively, we might select only routes via station s_4, i.e. r_2 or r_3, since following a least commitment strategy it might be possible to only commit to one available task decomposition once the train actually reaches s_4. In case such online changes are not directly permitted, a potential plan repair by restarting the planning process from this configuration should yield similar results. It is difficult to judge which option should be implemented in a real scenario. The currently most likely behavior, however, in our opinion is the least suitable in terms of robustness.

Conclusion and Future Directions

We motivated the need for robust solutions in hierarchical planning using an example of a train routing problem. Robustness has not been studied in the context of hierarchical planning before and finding a suitable definition yet alone applying it in an actual planner poses a difficult challenge. To this end, robustness might be best understood as a metric that can be favored more or less, depending on the safety restrictions imposed by the planning environment and the likelihood that certain events may disrupt normal operation. Here it can also be useful to consider predicted knowledge from experts or machine learning systems. In our example of a train network, we may get the information that one track will be subject to buckling due to a heatwave (Nguyen, Wang, and Wang 2012), an information we clearly want to take into account when routing the trains.

As a starting point, we chose a previously proposed approach to hierarchical planning by Stock et al. (2015) based on an abstract CSP representation, called meta-CSP (Mansouri and Pecora 2016). Not only the task network is modeled in this meta-CSP, but it also allows to incorporate many other types of information, such as resource, temporal, and spatial constraints. Reasoning in temporal and spatial domains is usually done using QCNs (Ligozat 2013; Dylla et al. 2017). In this context, Sioutis, Long, and Janhunen (2020) recently studied a notion of robustness, which concerns the perturbation tolerance of QCN solutions, i.e. their likelihood to resist a change in the environment. Based on similarities between the way knowledge is represented in meta-CSPs and QCNs, we discussed the potential applicability of a similar notion of robustness in the scope of hierarchical planning. This work poses as a first step towards more robust solutions in the hybrid hierarchical planning framework, and as such hopefully sparks a lively discussion on how to best define, measure, and incorporate robustness in existing applications.

References

Barták, R.; Salido, M. A.; and Rossi, F. 2010. Constraint satisfaction techniques in planning and scheduling. *J. Intell. Manuf.* 21: 5–15.

Behnke, G.; Höller, D.; and Biundo, S. 2019. Finding Optimal Solutions in HTN Planning - A SAT-based Approach. In *IJCAI*.

Bercher, P. 2021. A Closer Look at Causal Links: Complexity Results for Delete-Relaxation in Partial Order Causal Link (POCL) Planning. In *ICAPS*.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning - One Abstract Idea, Many Concrete Realizations. In *IJCAI*.

Bercher, P.; Höller, D.; Behnke, G.; and Biundo, S. 2016. More than a Name? On Implications of Preconditions and Effects of Compound HTN Planning Tasks. In *ECAI*.

Cardellini, M.; Maratea, M.; Vallati, M.; Boleto, G.; and Oneto, L. 2021. In-Station Train Dispatching: A PDDL+ Planning Approach. In *ICAPS*, 450–458. AAAI Press.

Dylla, F.; Lee, J. H.; Mossakowski, T.; Schneider, T.; van Delden, A.; van de Ven, J.; and Wolter, D. 2017. A Survey of Qualitative Spatial and Temporal Calculi: Algebraic and Computational Properties. *ACM Comput. Surv.* 50: 7:1–7:39.

Fritz, C. 2009. *Monitoring the Generation and Execution of Optimal Plans*. Ph.D. thesis, University of Toronto.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Ginsberg, M. L.; Parkes, A. J.; and Roy, A. 1998. Supermodels and Robustness. In *AAAI/IAAI*.

Ligozat, G. 2013. *Qualitative Spatial and Temporal Reasoning*. ISTE Ltd and John Wiley & Sons, Inc.

Lusby, R. M.; Larsen, J.; and Bull, S. 2018. A survey on robustness in railway planning. *Eur. J. Oper. Res.* 266: 1–15.

Mansouri, M.; and Pecora, F. 2016. A robot sets a table: a case for hybrid reasoning with different types of knowledge. *J. Exp. Theor. Artif. Intell.* 28: 801–821.

Muise, C. 2014. *Exploiting Relevance to Improve Robustness and Flexibility in Plan Generation and Execution*. Ph.D. thesis, University of Toronto.

Nguyen, M. N.; Wang, X.; and Wang, C.-H. 2012. A reliability assessment of railway track buckling during an extreme heatwave. *Journal of Rail and Rapid Transit* 226: 513–517.

Patra, S.; Mason, J.; Kumar, A.; Ghallab, M.; Traverso, P.; and Nau, D. S. 2020. Integrating Acting, Planning, and Learning in Hierarchical Operational Models. In *ICAPS*.

Russell, S. J.; and Norvig, P. 2020. *Artificial Intelligence - A Modern Approach, Fourth Edition*. Pearson Education.

Schattenberg, B. 2009. *Hybrid planning & scheduling*. Ph.D. thesis, University of Ulm, Germany.

Sioutis, M.; Long, Z.; and Janhunen, T. 2020. On Robustness in Qualitative Constraint Networks. In *IJCAI*.

Stock, S. 2016. *Hierarchische hybride Planung für mobile Roboter*. Ph.D. thesis, University of Osnabrück, Germany.

Stock, S.; Mansouri, M.; Pecora, F.; and Hertzberg, J. 2015. Online task merging with a hierarchical hybrid task planner for mobile service robots. In *IROS*.

Verfaillie, G.; and Jussien, N. 2005. Constraint Solving in Uncertain and Dynamic Environments: A Survey. *Constraints* 10: 253–281.