Robust Execution of Plans for Human-Robot Teams

Erez Karpas and **Steven J. Levine** and **Peng Yu** and **Brian C. Williams**
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar St., Cambridge, MA 02139
{karpase, sjlevine, yupeng, williams}@mit.edu

Abstract

Humans and robots working together can efficiently complete tasks that are very difficult for either to accomplish alone. To collaborate fluidly, robots must recognize the humans' intentions and adapt to their actions appropriately. Pike is an online executive introduced previously in the literature that unifies intent recognition and plan adaptation for temporally flexible plans with choice. While successful at coordinating human-robot teams, Pike had limited robustness to temporal uncertainty about the durations of actions. This paper presents two extensions to Pike that make it much more robust to temporal uncertainty. First, we extend Pike to handle uncontrollable action durations by enforcing strong temporal controllability. We accomplish this by generalizing standard strong controllability algorithms for STNUs to plans with choice. Second, in case a realized duration exceeds even the specified bounds and makes the entire plan infeasible, we attempt to intelligently negotiate with a human to relax some of the temporal constraints and restore feasibility, rather than immediately failing and halting execution. This negotiation is guided by a state-of-the-art conflict directed relaxation algorithm, which has previously only been used offline.

Introduction

Robots excel at perfoming repeated, monotonous tasks, but cannot perform many tasks that humans can. On the other hand, humans need flexibility as demonstrated, e.g., by the Skylab 4 strike, due to overly rigid schedules (Douglas 1991). This makes controlling a team of robots working together with a human a useful, yet challenging, objective. Pike (Levine and Williams 2014) is an online executive which unifies intent recognition and plan adaptation for temporally flexible plans with choice. It is responsible for making choices for the robots in the plan, monitoring execution, and dispatching actions at the appropriate times. Pike has been successfully used to coordinate human-robot teams, as demonstrated on a team of three robots and a human in an airplane manufacturing scenario (Burke et al. 2014). However, the previous version of Pike did not provide any sort of guatanteed robustness to temporal uncertainty.

Several other model-based executives interleave planning and online execution to improve robustness (Haigh and Veloso 1998; Ambros-Ingerson and Steel 1988; Ayan et al. 2007; Finzi, Ingrand, and Muscettola 2004; Alili et al. 2009; Clodic et al. 2009). Others use temporally flexible representations to further improve robustness (Lemai and Ingrand 2004; Py, Rajan, and McGann 2010; Chien et al. 2000; Shah, Conrad, and Williams 2009). However, these often treat the human as a disturbance, rather than a teammate.

The SAM executive (Pecora et al. 2012) explicitly considers both human activity recognition and appropriate robot adaptations in a single framework with temporal flexibility. Pike differs from SAM in that instead of generating temporal plans for robot adaptations online, it uses an a-priori known temporal plan with choices, allowing a great deal of preprocessing to be performed offline at compile time. This trades some generality for fast, reactive online performance.

Pike takes as input a task specified as a temporal plan network with uncertainty (TPNU), an extension of temporal plan networks (Kim, Williams, and Abramson 2001). TPNUs describe which activities should be executed, with a set of temporal constraints between them. Each activity has a duration, which might be controllable by Pike, or uncontrollable, in which case we only know a lower and upper bound on its duration. TPNUs also contain *choices* as to how execution proceeds. As with durations, choices can either be controllable, in which case the system makes the choice, or uncontrollable, in which case an external agent makes the choice. In the context of teamwork, we assume that these uncontrollable choices are made by a friendly teammate, who will not make a choice which causes the task to fail. While Pike can handle uncontrollable choices made by a teammate, it assumes that all activity durations are controllable, and thus has limited robustness to temporal uncertainty.

In this paper, we describe two extensions to Pike, which increase its robustness in the face of temporal uncertainty. First, we describe how to guarantee that Pike's choices lead to a schedule which is consistent for all possible realized uncontrollable durations, thus ensuring that the plan will be feasible as long as these action durations are actually within their specified bounds. However, since in the real world things might go horribly wrong, actual durations might exceed even the specified bounds. Our second extension to Pike is to intelligently negotiate with a human to relax some of the temporal constraints of the TPNU in order to restore its feasibility, rather than immediately failing and halting execution. This negotiation uses the conflict directed relaxation under uncertainty (CDRU) algorithm (Yu, Fang, and Williams 2014), which has previously been used in an offline intelligent decision assistant.

To illustrate how these two extensions make Pike more robust to temporal uncertainty, we will use the following running example throughout the paper: consider a scenario where Pike controls a UAV, which is collaborating with a human-piloted helicopter. These are both stationed at the base, and are tasked with searching for a missing vehicle, which could be in 3 possible locations: A, B, or C. The plan calls for the UAV to search location A alone, the helicopter to search location C alone, and for both the UAV and the helicopter to search location B together. Travelling from the base to B takes some time between 2 and 4 minutes, travelling between any two other locations takes some time between 3 and 4 minutes — all durations are uncontrollable. Finally, the human pilot is not willing to wait for the UAV at location B for more than 2 minutes.

Background

Before we present our extensions to Pike, we review the definitions of temporal plan networks and how Pike works.

Temporal Plan Networks

We begin with a formal definition of a TPNU:

Definition 1. *A Temporal Plan Network with Uncertainty (TPNU) is a 4-tuple, $\langle P, V, E, E_u \rangle$, where:*

- $P = P_C \cup P_U$ *is a set of finite domain decision variables.* P_C *are controllable by the system, while* P_U *are uncontrollable. Each variable* $v \in P$ *is associated with a domain* $dom(v)$*. An assignment to a decision variable is called a* choice*.*
- V *is a set of events representing designated time points. Each event* v *is also associated with a* guard condition*,* $guard(v)$*, which is a (possibly empty) set of choices, representing their conjunction.*
- E *is a set of simple temporal constraints between pairs of events. A simple temporal constraint is of the form* $l \leq v_i - v_j \leq u$*. Some of these represent the execution of an action, while others correspond to user requirements. Each constraint* e *is also associated with a guard condition,* $guard(e)$*.*
- $E_u \subseteq E$ *is a set of uncontrollable durations (also known as contingent constraints).* $E \setminus E_u$ *is the set of all controllable durations (also known as requirement constraints).*

Note that TPNUs contain two sources of uncertainty: uncontrollable decision variables (choices), and uncontrollable durations. While in general these can both be made adversarially, in this paper we assume that the uncontrollable choices are made by a cooperative agent, that is, a human working in a team with our system. Thus, we will focus on controllability of the TPNU with regards to temporal uncertainty about the uncontrollable durations, while attempting to maintain flexibilty with regard to uncontrollable choices.

Given an assignment to all decision variables, the *active* events and constraints are those whose guard conditions are true. These active events and constraints contitute a Simple Temporal Network with Uncertainty, or STNU (Vidal and Fargier 1999) — an extension of Simple Temporal Network, or STN (Dechter, Meiri, and Pearl 1991) which incorporates set-bounded uncertainty about durations. Several different notions of controllability have been defined with regards to STNUs; we will focus on the notion of *strong controllability*. An STNU is strongly controllable if there exists a schedule for the controllable events, such that for every possible realization of the uncontrollable durations, all temporal constraints are satisfied.

Another important aspect is that actions have conditions, which must occur at certain points during an action's execution, and effects which are caused by the action. We say that a schedule for an STNU is *causally complete* if the conditions of each scheduled action are satisfied at the appropriate times. Finally, an assignment to all decision variables that results in a strongly controllable STNU with a causally complete schedule will be called *feasible*. A TPNU that has such a feasible assignent will also be called *feasible*. A partial assignment to decision variables, which can not be extended to a feasible assignment is a *conflict*.

Modelling our running example as a TPNU is straightforward. The TPNU contains an uncontrollable choice — whether the human pilot goes to C or B first, and a controllable choice — whether the UAV goes to A or B first. The uncontrollable durations in the TPNU correspond to the possible actions in the plan (e.g., UAV flies from base to A, UAV flies from base to B, ...). The guard conditions for each of these are simply the choices that enable them. Finally, we have a temporal constraint which denotes that the human pilot is not willing to wait for the UAV for more than 2 minutes.

Pike: An Executive for Human-Robot Teams

We now provide a brief overview of Pike (Levine and Williams 2014), an online plan executive which can execute TPNUs. Pike's function is to assign values to the *controllable* decision variables and temporal durations, based on previous choices made by the human and robot, preconditions and effects of future actions in the plan, temporal constraints, and unexpected disturbances.

Pike takes as input a TPNU to be executed. During execution, Pike also takes input from a state estimator, which allows it to respond to unanticipated state disturbances, and from an activity recognition module, which recognizes the choices made by the human (i.e., via computer vision or any other sensing mechanism). Pike uses this information to infer possible intents of the human teammates (who are assumed to be rational and non-adverserial), and make decisions for the robots consistent with the humans' past choices. Using Pike to control robots results in a mixed-initiative execution in which humans and robots simultaneously work and adapt to each other to accomplish a task.

Pike must deal with two different types of constraints: causal constraints, such as ensuring that the conditions of the actions in the TPNU are satisfied, and the temporal constraints in the TPNU. To handle the causal constraints, Pike extracts labeled causal links from the TPNU, and encodes them in an Assumption-based Truth Maintenance System (De Kleer 1986) which allows Pike to efficiently check whether a choice will definitely violate causal consistency or not. For complete details we refer the reader to the original Pike publication (Levine and Williams 2014).

In order to dispatch activities at their proper times, which may also depend on the choices made, Pike leverages ideas originally presented in the Drake executive (Conrad 2010). A labeled all-pairs shortest path (APSP) is computed on the TPNU, to determine which events occur before which other events (Conrad, Shah, and Williams 2009). This labeled APSP is used as a dispatchable form for execution.

However, Pike assumes that the durations of all actions are controllable. We illustrate why this is problematic with a possible execution of our running example. First, assume the human helicopter pilot chooses to go to B first, and will get there sometime between 2 and 4 minutes. If Pike then chooses to send the UAV to A first and then to B, then the UAV will reach B sometime between 6 and 8 minutes. This is not a feasible combination of choices, as the human might reach B in 2 minutes, the UAV would reach B after 4 minutes (at least), and the human pilot is not willing to wait for more than two minutes. In the next section, we explain how to extend Pike to guarantee strong controllability subject to uncontrollable set-bounded durations.

Handling Uncontrollable Durations

In this section, we discuss extending Pike to make it robust to temporal uncertainty. This extension is a pre-processing step that ensures the *strong controllability* of the given temporal plan by generating additional temporal constraints.

We start with a review of existing methods for checking the strong controllability of STNUs. Checking whether an STNU is strongly controllable can be done by mapping the STNU into an equivalent STN with some additional temporal constraints (Vidal and Fargier 1999). The key for checking controllability is to identify what constraints may "squeeze" the uncontrollable durations, and apply additional constraints to shrink their ranges in order prevent it. This approach looks at each requirement constraint (that is, a temporal constraint which is not an uncontrollable duration), and checks which uncontrollable durations share an event with it. For each such uncontrollable duration, this approach adds a new temporal constraint (between the non-shared events) which ensure that all possible realizations of the uncontrollable duration are consistent with the requirement constraint and the added constraint. This new constraint can be obtained using simple arithmetic between the bounds of the requirement constraint and the contingent constraint, as described by Vidal and Fargier (1999).

Our approach extends the above with additional procedures to operate on TPNUs, which contain decision variables and guards in addition to events and temporal constraints. The guarded constraints make it difficult to apply the above rules, since arbitrary pairs of constraints might have incompatible guard conditions. In order to apply these rules to guarded constraints, whenever we add a new temporal constraint according to the above rules, we set its guard condition to be the conjunction of the two guards for which

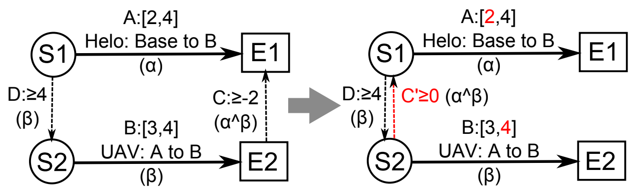

Figure 1: Original TPNU (left) and Modified TPNU (right)

the rule was applied on. This is similar to the conflict learning component of CDRU (Yu, Fang, and Williams 2014), except that they record which original constraints support each of the derived constraints, in order to be able to resolve conflicts. We do not need to keep track of this information explicitly, as the guard condition is enough for our purpose.

The final step is to check the consistency of the modified TPNU, except that we can now treat each uncontollable duration as if it were controllable — as the additional temporal constraints ensure that if the modified TPNU is temporally consistent, then the original TPNU is strongly temporally controllable. If the TPNU is consistent, Pike will proceed to reformulate it for execution. If any conflict is detected in the network, Pike will collect and record the set of guard conditions of constraints in a conflict. During execution, Pike constantly reviews these conflicting sets of choices and avoids making decisions that will result in any of them.

We demonstrate this procedure on a temporal network of four constraints extracted from our running example (Figure 1). Constraints A and B, which represent the traversal of Helo and UAV to location B, are contingent constraints with uncontrollable durations. Constraint C and D are both requirement constraints. C indicates the maximum waiting time imposed by the pilot, while D represents the minimum time of travel for the UAV to location A. We simplified the original network by converting the uncontrollable traversal time between Base and A to a requirement constraint with its upper bound. The guard conditions of constraint A, B, C and D are α, β, $\alpha \wedge \beta$ and β, respectively. Guard condition α encodes the choices of 'Helo goes to Location B first',while β is 'UAV goes to Location A first'.

Constraint C starts and ends at contingent time points (denoted by squares in the graph), therefore we need to eliminate it through reduction. We start by reducing it through the upper bound of Constraint B, a contingent constraint that shares the same start node with C; then through the lower bound of Constraint A, which shares the same end node with the reduced constraint after the first step. The result is a new constraint C' from S2 to S1 with a lower bound of 0. Its guard condition is the conjunction of the guard conditions from constraint A, B and C: $\alpha \wedge \beta$. This is inconsistent due to a negative cycle formed by constraints D (-4) and C' (0), and thus Pike discovers that $\alpha \wedge \beta$ is a conflict, and will avoid making this guard true. Thus, if Pike detects that the human pilot is heading to B first (α), it will dispatch the UAV to B first as well ($\neg\beta$). If the human pilot heads to C first ($\neg\alpha$), then both choices for the UAV are consistent, as the human will reach B sometime between 6 and 8 minutes, and both choices for the UAV ensure it reaches B on time.

Handling Unexpected Durations

In the previous section, we explained how Pike can *guarantee* that the schedule it finds will be feasible assuming that all action duration realizations are indeed within their uncertainty bounds. Unfortunately, unexpected things often happen in the real world, which might cause the durations of some action to fall outside of its uncertainty bounds. In such a case, Pike may no longer be able to continue executing the same TPNU, as the TPNU may no longer be feasible. Instead of simply halting execution with an error message, we would like to degrade gracefully, and find a way to continue.

We accomplish this by using the Conflict Directed Relaxation with Uncertainty (CDRU) algorithm (Yu and Williams 2013; Yu, Fang, and Williams 2014). CDRU was previously used in an offline decision assistant, which negotiates with a human about how to relax the constraints in an over-subscribed TPNU to restore its feasibility. CDRU uses preferences the human has about what can be relaxed. Given a TPNU $N = \langle P, V, E, E_u \rangle$, these preferences take the following form:

- $RE \subseteq E \setminus E_u$ is a set of relaxable temporal constraints whose bounds can be relaxed;
- $RE_u \subseteq E_u$ is a set of relaxable contingent constraints whose bounds can be tightened;
- $f_p : \times_{p \in P} dom(p) \rightarrow \mathbb{R}^+$ is a function that maps each assignment to every decision variable to a positive **reward**
- $f_e : (e_i, e'_i) \rightarrow r \in \mathbb{R}^+$ is a function that maps the following to a **non-negative cost**.
  - the **relaxation** to a requirement constraint, $e_i \rightarrow e'_i, e_i \in RE$;
  - or the **tightening** to a contingent constraint, $e_i \rightarrow e'_i, e_i \in RE_u$;

If Pike encounters an execution failure due to temporal inconsistency, it will capture the current status of execution as a TPNU: the choices that were already made (both controllable and uncontrollable) and the realized durations of actions which have already finished are encoded in this TPNU. It then runs CDRU on this over-subscribed TPNU to attempt to restore consistency by relaxing some constraints through a collaborative dialog with the human.

CDRU enumerates relaxations in best-first order of utility. It uses an approach that is based on conflict-directed A* (Williams and Ragno 2002), extended to continuous variables and constraints (Yu and Williams 2013). There are three major steps in the algorithm:

1. Conflict learning: given the infeasible combination of choices detected by Pike and CDRU, extract the conflicting sets of temporal constraints that cause the failure.
2. Conflict resolution: compute the optimal continuous relaxations for constraints in the conflicts to resolve them.
3. Negotiation with the users: communicate the relaxations to the users and ask for approval. If any additional requirements are given, record them as new conflicts and return to the second step.

CDRU loops through these three steps and generates new candidate relaxations until one candidate makes the TPNU feasible. Within each loop, CDRU looks at the best candidate so far, and checks if it resolves all known conflicts. If not, an unresolved conflict will be learned and used to generate other candidate relaxations. These candidates are obtained by either setting a decision variable to some value, or relaxing a continuous constraint which repairs a conflict, according to RE and RE_u. If CDRU does not find a controllable relaxation before the candidate queue exhausts, it will return null indicating that no relaxation exists for the over-constrained TPNU.

In addition, it is important to explain how the CDRU algorithm interacts with the human. Every time a feasible relaxation is identified, the system will present it to the user and ask for approval. If the user is not satisfied, they can supply additional requirements and ask for an alternative. Each new requirement will be recorded as an additional conflict and all future relaxations generated by CDRU will respect it. The negotiation will continue until an agreement with the user is reached. At this point, we can apply the relaxation to generate a new TPNU, which Pike will then execute.

Currently the system supports two modes of communications: a graphical user interface and a natural language interface. While finding the right modality for interacting with the user depends on the application, the environment, and many other factors, it is beyond the scope of this paper. We simply assume that some form of interaction is possible.

To illustrate this with our running example, consider the case that the helicopter pilot heads to C first, gets there in 3 minutes, and then heads to B. The helicopter pilot will reach B between 6 and 7 minutes from the start. Since the human started off going to C, Pike sent the UAV to A first, but due to a completely unexpected and unmodeled storm, it took the UAV 6 minutes to reach A — a duration which exceeds the specified bounds. The UAV will reach B sometime between 9 and 10 minutes, thereby violating the pilot's temporal constraint of not waiting for more than 2 minutes. Instead of halting execution with an error message, our new version of Pike will negotiate with the human to find a relaxation which would enable execution to proceed. As the only temporal requirement is the pilot's, Pike would ask her whether she is willing to relax her temporal constraint, and wait at most 4 minutes for the UAV. Assuming the pilot agrees, execution can then proceed.

Conclusion

We have described two extensions to Pike. First, we added constraints that enforce strong temporal controllability, which guarantees that as long as the temporal uncertainty remains "within bounds", Pike will be able to successfully execute the plan. Second, we extended Pike to handle the case where temporal uncertainty goes "out of bounds", by negotiating with a human to relax some of the temporal requirements in the plan. These extensions make Pike much more robust to temporal uncertainty, as demonstrated on a simple search-and-rescue task.

Acknowledgements

The work was partially supported by the DARPA MRC Program, under grant number FA8650-11-C-7192, and the Boeing Corporation, under grant number MIT-BA-GTA-1.

References

Alili, S.; Warnier, M.; Ali, M.; and Alami, R. 2009. Planning and plan-execution for human-robot cooperative task achievement. In *19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Ambros-Ingerson, J. A., and Steel, S. 1988. Integrating planning, execution and monitoring. In *AAAI*, volume 88, 21–26.

Ayan, N. F.; Kuter, U.; Yaman, F.; and Goldman, R. P. 2007. Hotride: Hierarchical ordered task replanning in dynamic environments. In *Proceedings of the 3rd Workshop on Planning and Plan Execution for Real-World Systems (held in conjunction with ICAPS 2007)*, volume 2.

Burke, S.; Fernandez, E.; Figueredo, L.; Hofmann, A.; Hofmann, C.; Karpas, E.; Levine, S.; Santana, P.; Yu, P.; and Williams, B. 2014. Intent recognition and temporal relaxation in human robot assembly. In *ICAPS System Demos*.

Chien, S. A.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 2000. Using iterative repair to improve the responsiveness of planning and scheduling. In *AIPS*, 300–307.

Clodic, A.; Cao, H.; Alili, S.; Montreuil, V.; Alami, R.; and Chatila, R. 2009. Shary: a supervision system adapted to human-robot interaction. In *Experimental Robotics*, 229–238. Springer.

Conrad, P. R.; Shah, J. A.; and Williams, B. C. 2009. Flexible execution of plans with choice. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Conrad, P. R. 2010. Flexible execution of plans with choice and uncertainty. Master's thesis, Massachusetts Institute of Technology.

De Kleer, J. 1986. An assumption-based tms. *Artificial intelligence* 28(2):127–162.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Douglas, W. K. 1991. Psychological and sociological aspects of manned spaceflight. In *From Antarctica to Outer Space*. Springer. 81–87.

Finzi, A.; Ingrand, F.; and Muscettola, N. 2004. Model-based executive control through reactive planning for autonomous rovers. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, 879–884. IEEE.

Haigh, K. Z., and Veloso, M. M. 1998. Interleaving planning and robot execution for asynchronous user requests. In *Autonomous agents*, 79–95. Springer.

Kim, P.; Williams, B. C.; and Abramson, M. 2001. Executing reactive, model-based programs through graph-based temporal planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01)*, 487–493.

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution in robotics domains. In *AAAI*, volume 4, 617–622.

Levine, S., and Williams, B. 2014. Concurrent plan recognition and execution for human-robot teams. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*.

Pecora, F.; Cirillo, M.; Dell'Osa, F.; Ullberg, J.; and Saffiotti, A. 2012. A constraint-based approach for proactive, context-aware human support. *Journal of Ambient Intelligence and Smart Environments* 4(4):347–367.

Py, F.; Rajan, K.; and McGann, C. 2010. A systematic agent framework for situated autonomous systems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 2*, 583–590. International Foundation for Autonomous Agents and Multiagent Systems.

Shah, J. A.; Conrad, P. R.; and Williams, B. C. 2009. Fast distributed multi-agent plan execution with dynamic task assignment and scheduling. In *Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence* 11:23–45.

Williams, B. C., and Ragno, R. J. 2002. Conflict-directed A* and its role in model-based embedded systems. *Journal of Discrete Applied Mathematics* 155(12):1562–1595.

Yu, P., and Williams, B. 2013. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI-13)*, 2429–2436.

Yu, P.; Fang, C.; and Williams, B. 2014. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*.