PlanCurves: An Interface for End-Users to Visualise Multi-Agent Temporal Plans

Pierre Le Bras[1], **Yaniel Carreno**[1,2], **Alan Lindsay**[1], **Ronald P. A. Petrick**[1,2], **Mike J. Chantler**[1,2]
[1]Department of Computer Science, Heriot-Watt University, Edinburgh, UK.
[2]Edinburgh Centre for Robotics, UK.
{p.le_bras, y.carreno, alan.lindsay, r.petrick, m.j.chantler}@hw.ac.uk

Abstract

In operational contexts, there is a growing need to make automatically generated plans available for assessment, verification, and accountability purposes, in order to evaluate the risks associated with such plans prior to their execution. However, this task can be challenging, especially in situations where multiple agents are scheduled to interact and carry out activities over long periods of time. In this paper, we present PlanCurves, a web application for visualising multi-agent plans generated by temporal planners, to enhance end-user understanding and interpretation of these plans. We implement a familiar chart visualisation for temporal plans, in combination with a novel layout designed to provide a rapid overview of the plan while highlighting the relative similarities between agents. We describe how the approach is both planner and domain agnostic, and show potential applications of the visualisation interface using example domains from the International Planning Competition and domains with industrial applications.

Introduction

While automated planning and scheduling systems progress towards producing more reliable and efficient plans, there are situations when a human operator needs to assess, evaluate, and approve of such plans prior to their execution. In industrial contexts, for example, the movement of mobile agents might need to be accounted for by a supervisor who has up-to-date local knowledge that the system might not have. In more unpredictable environments, this approval might be critical for guaranteeing the plan's safety and success. It is therefore essential to consider how plan information is communicated to end-users in such situations.

Multi-agent temporal plans pose a particular challenge for human end-users in terms of understanding both the plan status and the interactions between agents over long periods of time. To address this problem, we present *PlanCurves*, a visualisation system that enables end-users to view and interactively explore multi-agent temporal plans. In particular, PlanCurves implements two main visualisation methods for a given plan. First, it uses the intuitive and familiar Gantt chart representation to display the list of actions as bars positioned on a temporal axis. Second, it incorporates a novel application of Time Curves (Bach et al. 2015). This visualisation technique is used to show the timeline of an object, distorted to illustrate the similarities between different states. In the context of multi-agent plans, we use this method to show multiple simultaneous actor timelines and their relative similarities throughout the plan.

In addition to describing the available visualisations and their uses, we also provide details of the implementation of PlanCurves as an online tool for end-users and planning practitioners. While PlanCurves was initially created to address a specific issue–the analysis of planned movements for remote agents in hazardous environments–we have evolved its design to be both planner and domain agnostic. PlanCurves presently focuses on two core visualisations, but we believe it also has the potential for further applications, and we present details on how this system could be better integrated with existing planning and verification tools.

This paper is organised as follows. First, we survey some background on visualisation methods in the planning community. Next, we describe the particular visualisations we use in this work. We then discuss the implementation of the system and additional features. Finally, we highlight possible future work for this project.

Related Work

It has been recognised that systems should exploit operator expertise to facilitate optimisation tasks such as planning and scheduling (Scott, Lesh, and Klau 2002), and visualisations have been shown to be an effective aid (Kirkpatrick, Dilkina, and Havens 2005). While humans have a natural capacity to discern patterns from visual cues such as proximity (Ware 2012), there is however an apparent lack of visual tools, beyond domain-specific ones, designed to support end-users assessing automatically generated plans.

Previous work has been done to support user developing domains with visual representations such as node-link diagrams, notably within *GIPO* (McCluskey and Simpson 2006) but not exclusively (Kim and Blythe 2003; Vodrážka and Chrpa 2010; Vrakas and Vlahavas 2003). Porta (2000) also uses node-link representations to support plan author-

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

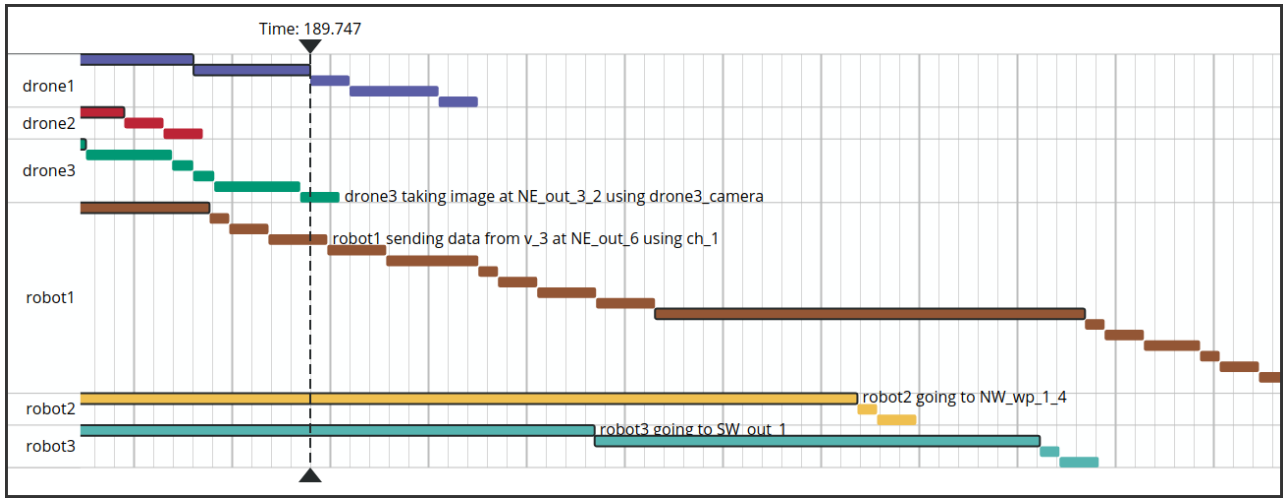

Figure 1: Activity Charts are similar to Gantt charts and depict the order, sequence, and timing of activities. Bars representing activities are grouped by actor, providing users with a perspective on their individual missions.

ing. In *itSIMPLE*, Vaquero et al. (2007) make use of Petri-nets to represent domains. The scope of our work, however, emphasises the visualisation of plans themselves (rather than domains) for assessment by non-planning experts prior to their implementation.

Gantt charts are probably the most common type of visualisation used to map a set of tasks (or actions) onto a time axis. As such, they are the go-to representation for systems visualising temporal plans, e.g., in *itSIMPLE* (Vaquero et al. 2007), *EUROPA* (Barreiro et al. 2012), or *ModPlan* (Edelkamp and Mehler 2005). These classical Gantt charts come with a constraint: the activities are simply listed and ordered chronologically. The relationship between tasks, based on actor or resources used, are therefore not represented. Recent implementations, however, propose to reorder activities in more meaningful ways (Pralet et al. 2018; Sampath et al. 2013), grouping them by actor, resources, or locations. These works are however integrated within domain-specific applications.

Gupta et al. (2016) formalise this revision of Gantt charts to depict similarity information between activities. In their work, they use two categorical axes to estimate the similarities between activities: actors and discrete location. Since our work focuses on depicting multi-agent plans and, in particular, making the different agent tracks understandable to end-users, we implement a similar activity chart, grouping activities by agent, hence providing a familiar display.

Another common visualisation method for plans is to use simulations, for example of a set environment (Do et al. 2011; Haas and Havens 2008), enabling the users' tacit knowledge of the environment (Fraternali et al. 2012). These techniques can be combined with the above-mentioned Gantt charts to provide more in-depth analyses of plans (Hoogendoorn et al. 2006). These types of visualisations, however, rely on the assumption that the domain also describes an environment, sometimes too specific to be generalisable. We therefore propose to add the optional use of tailored environment simulation for users.

Edelkamp and Mehler (2005) and Vaquero et al. (2007) use abstract simulations (e.g., UML diagrams). While these can be informative, they are still limited by their frame-by-frame display method, requiring more time and attention from operators analysing the plans.

A final, less common, type of representation for temporal plans are timelines. Chakraborti et al. (2017) use timelines to display alternative plans. Timelines have also been used to represent resource usage (Barreiro et al. 2012), constraints (Porteous et al. 2011), or property evolution (Vaquero et al. 2009). Batrinca et al. (2013) use this representation to enable collaborative planning across multiple teams. This approach could be used to represent the parallel timelines of multiple agents in a plan. Additionally, while timelines are often represented in a straight linear manner, visualisations such as Time Curves propose to encode similarity information and distort timelines to represent proximity between states (Bach et al. 2015). In our work, we augment the Time Curves visualisation method and display to users the simultaneous activities of multiple agents while showing their relative interactions based on custom similarity measures.

Visualisations

In this section, we present the two main visualisation methods we use to display plans to users: the Activity Chart and Time Curves. We then present how interactivity can support users in their exploration, and the potential uses of these visualisations.

Activity Chart

The Activity Chart (Figure 1) relies on the familiar principles of Gantt charts to display activities as horizontal bars, scaling their position and width to a time axis in order to convey two kinds of information: the sequence in which activities take place, and their timings.

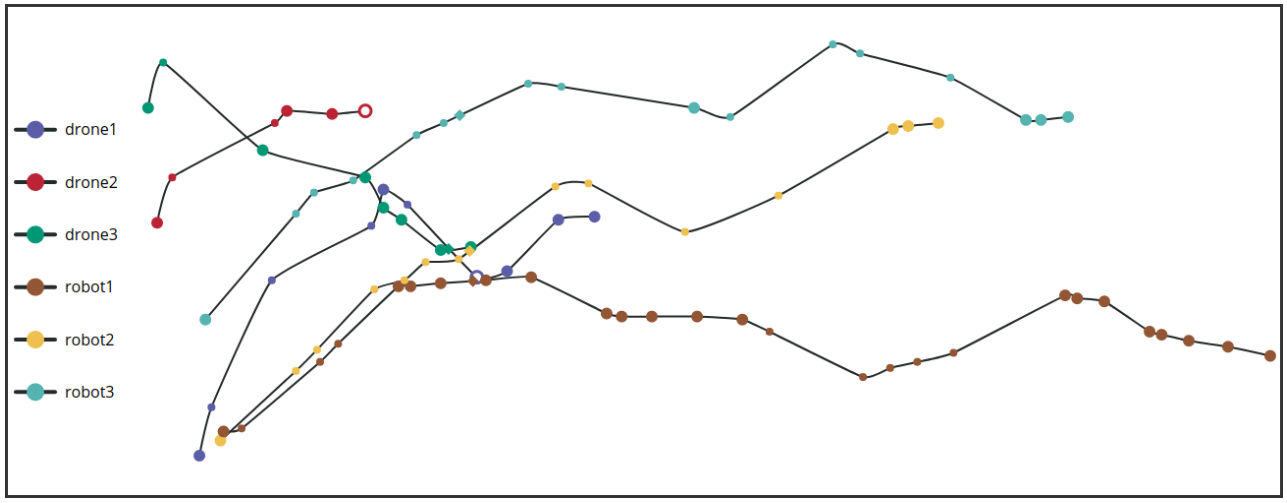

Figure 2: Time Curves depict the timeline of multiple agents, across the entire plan, simultaneously. They *distort* the timelines using similarity measures between states, allowing users to estimate the distances between agents throughout the entire plan.

Unlike classical Gantt charts, where bars (or activities) would only be ordered by timing, we first group them by actor. In multi-agent contexts, this representation provides a different perspective to the user, by focusing on the different agents' overall role and individual contribution to the plan.

While it offers a familiar and intuitive layout, the Activity Chart fails to show the interactions agents can have with each other across the span of the plan. We therefore use a second visualisation to represent multi-agent temporal plans: Time Curves.

Time Curves

Time Curves were originally designed to represent the similarity of a single entity across time (Bach et al. 2015). In PlanCurves, we extend this visualisation technique with a novel application: showing the timelines of multiple agents simultaneously, as generated by temporal plans (Figure 2).

Time Curves present the timeline of each actor in the plan, as defined by the user, with each timeline being represented using a sequence of connected dots: the actors' states throughout the plan. These timelines are however *distorted* to communicate the relative similarities between states. We create these distortions by calculating the similarity matrix between states from a set of state features defined by the user. This multidimensional similarity matrix is then embedded onto a 2D projection, the Time Curves' set of dots.

This type of visualisation for multi-agent temporal plans offers the advantage of showing, in one display, the relative task timing and the *interaction* between actors. These interactions are based on the similarity measure used, which is a function of time (to visualise the time dimension) and other features determined by the user, such as physical position in the environment or resources used.

Typically, an operator in charge of evaluating the planned movements of multiple agents for a mission can use Time Curves to display state similarity based on physical locations. An example of such visualisation is shown in Figure 2. In this example, they can immediately notice that: *drone1*, *robot1*, and *robot2* all start from nearby locations; then, *robot1* and *robot2* follow the same route closely for the first part of the plan; *drone3* will hover over *robot3* and eventually, *drone1* and *drone3* will meet closely; finally, the drones will finish their respective tasks early in the plan.

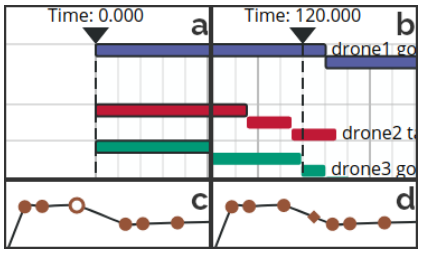

Figure 3: Time interactions implemented in the visualisation: panning on the Activity Chart and clicking on the Time Curves. This would update the position of the time cursor on the Activity Chart (a and b), and the focus on dots on the Time Curves (c and d).

Interactivity

Plan interpretability can not solely rely on visual displays. Interaction is required to further aid the user in understanding and assessing the plan presented to them.

To facilitate the temporal exploration of the plan, we have enabled a panning behaviour on the Activity Chart, allowing the user to see, for a selected time in the plan, the actions carried out by the actors (Figures 3a and 3b). Simultaneously, the Time Curves' dots change focus if the state they represent is reached at that time in the plan (Figure 3c). Should the selected time fall between states, a diamond-shaped marker is shown instead between the associated dots, positioned following a linear time scale (Figure 3d). Given the Time Curves' potential to show the overall plan to users, we have

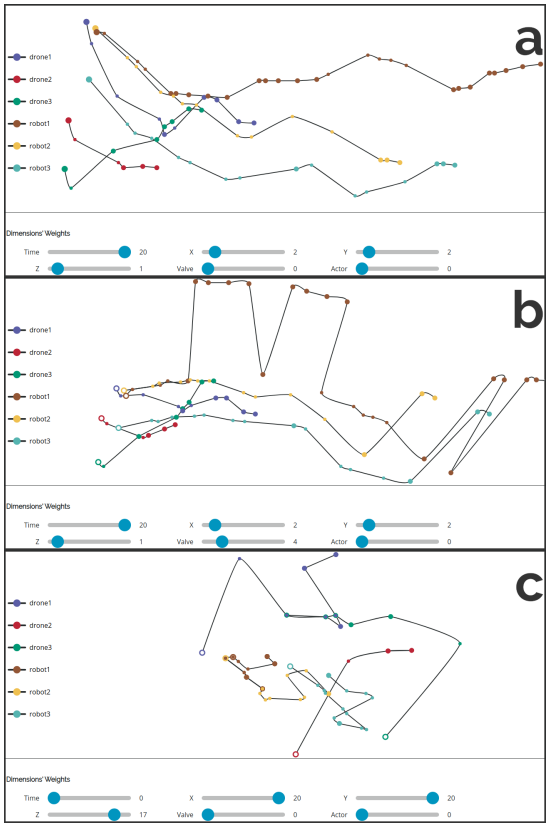

Figure 4: Adjusting the state feature weights (with sliders) allows users better control over the Time Curves. In this example, (a) shows the Time Curves with a stretched time axis, while keeping position weights (*X*, *Y*, and *Z*) to show relative distances. Increasing the weights of the feature *Valve* highlights the states where robots are operating on valves, i.e. using a resource (b). Having only position features allows the user to get a side view of the robots' paths in their environment (c): robots on the floor and drones up in the air.

also enabled a click behaviour on the dots to allow the users to reach and assess certain parts of the plan faster.

Contextual interactions were also added to help the user interpret the visualisations. Mousing over a bar or dot reveals a description of the associated action or state. These descriptions are initially partially hidden to keep the visualisation uncluttered and improve clarity. The actor's tracks (activity sequence, or timeline) are also highlighted to help the user analysing the two visualisations simultaneously.

Lastly, to support the readability of Time Curves, we have added zoom, pan, and rotation interactions, offering the user many viewpoints to analyse them. To support their interpretability, we have added weight sliders on the state features used to create the 2D embedding of these states. Figure 4 shows an example of this functionality, on a plan with six agents and six user-defined state features: *Time*; *X*, *Y*, and *Z* coordinates; the *Valve* operated on (if any); and the *Actor*. This level of control and adjustability over the feature weights enables users to get different perspectives on

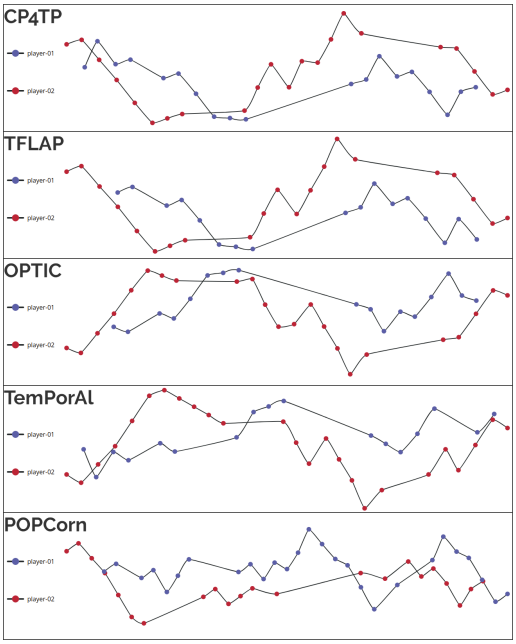

Figure 5: Time Curves for the Sokoban problem 2 of IPC 2018. Looking at the generated plans, we can quickly understand that CP4TP, TFLAP, OPTIC, and TemPorAl propose almost identical solutions (the latter two Time Curves were generated with an inverted vertical axis). Although it starts similarly, POPCorn's solution finishes differently.

the time curves, in order to understand the importance of particular features and gain further insights into the plan.

Example Uses

PlanCurves is domain and planner agnostic. For example, we have been able to visualise the plans generated from five different planners for the *Sokoban*, *Road traffic*, *Airport*, and *Trucks* domains submitted to the temporal tracks of IPC 2018 (Coles, Coles, and Martinez 2018).

Using the Time Curves visualisations, we can quickly read the actors' interactions across the span of the plan. This also enables users to quickly compare plans. Figure 5 for example, shows the Time Curves of the five plans for the second Sokoban problem of IPC 2018. Looking at these Time Curves it becomes clear that CP4TP, TFLAP, and OPTIC proposed almost identical solutions, although the Time Curves from the OPTIC plan have been inverted as an artefact of the projection. TemPorAl's solution differs slightly but retains a similar shape. Finally, we can see that POPCorn's solution starts similarly, but soon differs.

This rapid analysis improves the assessment of plans by operators prior to their implementation. In the case of complex environments, it can be difficult for operators to parse and validate the safety of plans when multiple agents are involved. PlanCurves can assist this review by highlighting zones of close interactions.

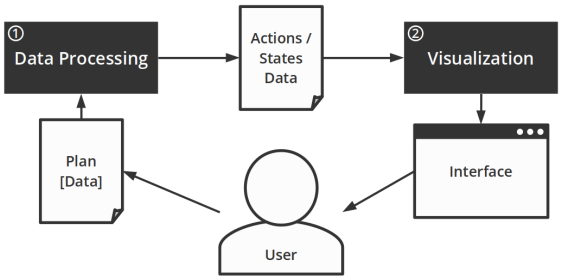

Figure 6: PlanCurves works in two phases: a) processing the plan data provided by the user, and b) visualising the generated action and state data to present back to the user.

System

While the PlanCurves system was initially developed within a larger project, we have designed it to be domain and planner agnostic by allowing the data processing rules to be customised. As a result, we have developed PlanCurves as an online application to be available for practitioners to use in conjunction with any temporal planner and adapted to any domain specification that includes actors. In this section, we describe how the system works and the options available to users. The application only requires the user to load their plan as a text file. The plan is then processed to generate action and state data for the visualisation, which are then displayed on the interface (Figure 6).

We have observed that, although selected to compromise on competing factors (e.g., conciseness, usability, and performance), a planning domain model does not always provide the most appropriate information for creating plan visualisations. Similar limitations have been observed in the contexts of solvability (e.g., macro actions (Newton et al. 2007)), specifying control knowledge (e.g., concept learning (Martin and Geffner 2004)), and explainability (Lindsay 2019). This has led us to establish a loose coupling between our system and the specific planning domain, instead organising the system around the plan outputs directly. The system's operation relies on inference rules that connect the plan actions to states (similar to the states captured by the planning model) and interpretation rules, which organise the content (e.g., natural language annotations) for presentation purposes.

Generating Action Data

The action data is directly generated from the plan file. The plan is firstly parsed to produce *raw* action data (Figure 7 *Parsing*): a list of action tokens and temporal information. The raw action is then interpreted to produce the final action data in a JSON format (Figure 7 *Interpreting*). Two elements are mandatory for the interpretation: a) the first item in the list of action tokens has to be the action name, and b) the action must have an actor. The interpretation then assumes the following default rules:

- The second item in the action token list is the actor;
- The other items constitute the actions' parameters;

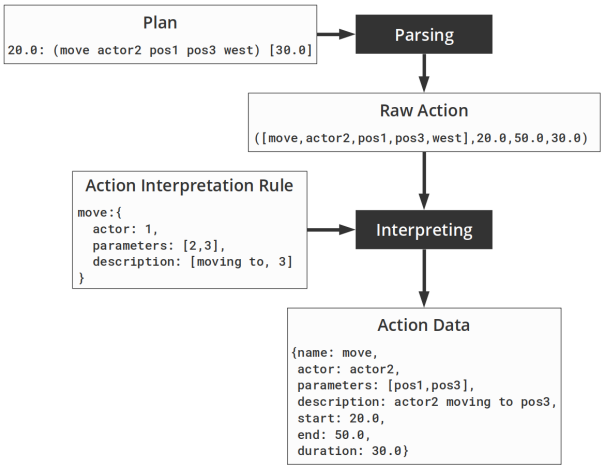

Figure 7: Interpretation rules allow the user to define the action data and adapt it to their domain. In this example, after parsing an action from the plan, the system interprets it using the rule associated with the action name "*move*".

- The description of the action is simply the concatenated list of the action's name, actor, and parameters.

The user is, however, able to customise these rules, indicating, by index, which item is the action's actor, which items are parameters, and how the description should be constructed (Figure 7 *Interpretation Rule*).

This action data is directly used to render the Activity Chart. It is also the base for the state data.

Generating State Data

State data is derived from the previously generated action data. For each action, the system infers a resulting state (Figure 8 *Inferring*). Initial states are also inferred from the first actions of each actor. By default, a state inherits the action name, actor, parameters and finish time (start time for initial states). However, users are able to provide rules for the state name and parameters (Figure 8 *Inference Rule*).

The state data is then further interpreted, adding a description and a list of features to the state (Figure 8 *Interpreting*). The list of features will be used to create the state similarity matrix before projecting the states for the Time Curves. The state time is automatically added to these features. Using interpretation rules, the user is allowed to provide their custom set of features, using a key-value map: the key being the feature's name, and the value a parameter index (Figure 8 *Interpretation Rule*). If no rules are provided, the state actor is added to the list of features by default.

Every type of state must have the same set of features. The user is able to propose numerical, categorical or ordinal features. If ordinal features are used, *Ordinal Feature Rules* must be provided, mapping the parameter value to its corresponding value on a numerical scale.

Two feature names are reserved and parsed differently by the system. By using "*actor*", a categorical feature taking the value of the state's actor will be created. Using "*position*" as a feature, the corresponding point coordinates will be used.

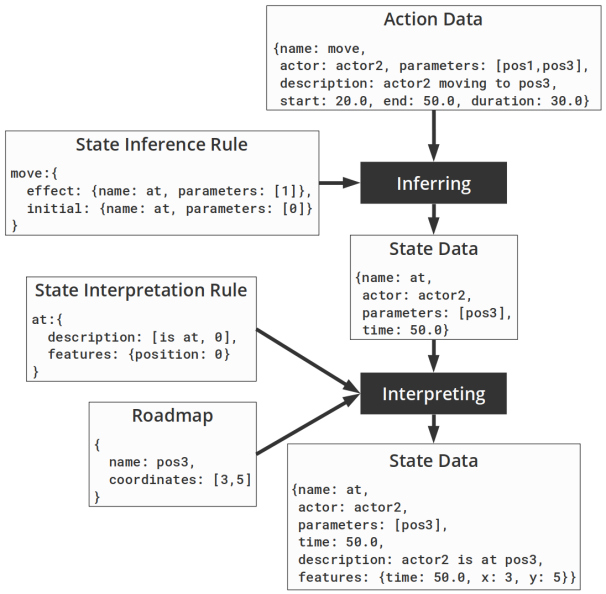

Figure 8: The user can describe how states should be generated. Following from Figure 7, the system will create and interpret an "*at*" state from the "*move*" action, using rules and roadmap information provided by the user.

These coordinates must be included in a Roadmap data file (Figure 8 *Roadmap*).

From the state data, the system will compute a distance (or dissimilarity) matrix. To support the combination of categorical and numerical features, we use the *general (dis)similarity coefficient* (Gower 1971). The distance between state x and y is computed as:

$$d(x,y) = \frac{\sum_{f \in F} \omega_f \delta(x_f, y_f)}{\sum_{f \in F} \omega_f}$$

Where: F is the set of state features, ω_f is the weight of feature f, and $\delta(x_f, y_f)$ is the distance between the features f of states x and y. If f is numerical, $\delta(x_f, y_f)$ is the normalised difference between x_f and y_f; if f is categorical and $x_f = y_f$, $\delta(x_f, y_f) = 0$ (1 otherwise). Finally, these state distances are passed to a dimensionality reduction algorithm (Multi-Dimensional Scaling - MDS), to generate the projection data for Time Curves (Brandes and Pich 2007).

Additional Features

Generating Activity Charts and Time Curves visualisations forms the basis of the PlanCurves application. We have however implemented additional features to enhance the user's interaction with multi-agent temporal plans, in particular to support the analysis of plans evolving in an environment.

Scene Mapping

While Time Curves can give an overview of the agents' relative positions throughout the plan (if using their physical position as the basis for similarity), it fails to depict their accurate location within their environment. To support this type

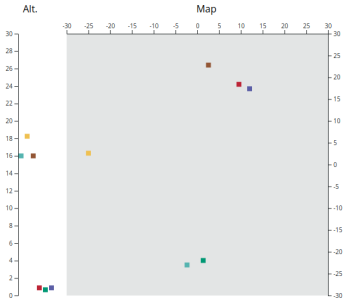

Figure 9: The Scene Map simulation provides better estimations of the actors' physical location in their environment.

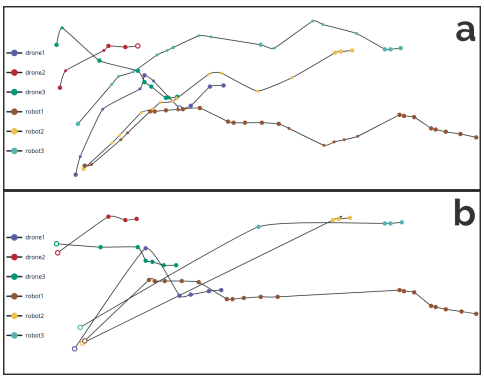

Figure 10: Path planning provides users with insights regarding the actors' planned movement and the potential zones of conflicts. Here we show, for the same plan, the Time Curves (a) with path planning, and (b) without path planning.

of analysis, we have implemented a Scene Map simulation (Figure 9), which shows, at any selected time of the plan, where the agents would be located using a top-down view. If the physical domain spans three dimensions (described in the Roadmap data file) the visualisation automatically includes an altitude axis too. Should the selected time correspond to an in-between-states stage for an actor, its position is estimated on a linear scale between the two states.

To use this visualisation, the user has to include the "*position*" feature for states, defining which parameter describes a physical point. Additionally, a Roadmap data file must be provided, including a list of points, with name and coordinates, and details about the environment (minimum and maximum of each spatial axis used).

Path Planning

Planners are often limited by their representation of space, and with complex environments, they can only implement a limited number of waypoints. To complement temporal planners and overcome this limitation, PlanCurves offers the possibility to have paths between points computed using a path planner based on A* search with a Euclidean distance heuristic (Hart, Nilsson, and Raphael 1968).

This can be achieved if the user provides a Roadmap data file detailing the neighbours of each waypoint. In addition,

the user must describe which actions involve a movement by using a "*waypoints*" rule detailing which action parameters are the starting and finishing positions.

After getting a path, PlanCurves would then generate sub-actions to represent it, and later sub-states. These sub-states are directly shown in the Time Curves visualisation (Figure 10). To keep the Activity Chart uncluttered, sub-actions are only displayed when the user decides to expand (by clicking) the initial movement action.

Limitations and Future Work

The system described above constitutes the core of PlanCurves: a customisable visualisation interface for multi-agent temporal plans. It, however, still presents constraints and opportunities for future development and integration with other systems.

We must first acknowledge that PlanCurves was constructed to highlight the interaction between multiple agents. Its design is therefore actor-driven, and it assumes that the plans it represents have clear and identifiable actors.

In its current implementation, PlanCurves can only parse and visualise "*simple*" actions, that is, actions representing a single task using a single actor. Future instances of PlanCurves will allow users to detail, in action interpretation rules, whether an action is composite and should therefore be interpreted as multiple linked sub-actions. Such links will also be represented in the visualisations.

Concerning the visualisations, one major limitation for the user is the lack of visual encoding to easily distinguish between actions or states with different types. For example, whether an actor is planned to simply navigate or to perform a manoeuvre, both associated bars or dots will look identical to the user. Icons, or glyphs, are a type of categorical graphical encoding that would allow users to identify actions and states faster (Borgo et al. 2013).

Ultimately, we wish to allow PlanCurves to connect to other planning systems and give end-users more control and features over their domains, problems, and plans.

Integration with online planners could be beneficial to end-users and provide more functionalities. For example, `planning.domains` (Muise 2016) which already includes a set of simple visualisation tools as addons. Beyond the representation of plans generated by such systems, one could imagine the possibility for non-technical users to design and better control queries to the planner (e.g. updating problems) through interactive visualisations which improve their comprehension and context awareness. In particular, this could support explainability features, such as "*what-if*" queries to the planner (Fox, Long, and Magazzeni 2017).

PlanCurves infers state information from the plan provided by the user. While it opens customisability to better suit the user's needs, these state properties are limited in comparison to those used within planners. Connecting to systems such as VAL (Howey, Long, and Fox 2004) would enable the system to infer more state properties with better accuracy. Other information such as action dependency would allow further explainability for the end-user, such as "*why*" or "*why-not*" queries.

Finally, we note that implementing A* poses a limitation to the wide variety of path planning algorithms available. Additional work towards that feature should include giving more integrated options to the user (algorithms and heuristics) or enable externally-implemented algorithms to be used, supported by a semantic attachments interface.

Conclusion

In this paper, we address the issue of plan visualisation for end-users by presenting PlanCurves, an online application designed to visualise multi-agent temporal plans. We have focused this application around two main visualisations:

- an Activity chart, showing the listing and timing of proposed actions, grouped by actors to provide a rapid understanding of the different agents' missions; and
- Time Curves, projecting the timelines of each agent onto a 2D display using their relative similarities, as defined by the user, to produce an at-a-glance overview of the plan and the interactions between the agents.

We have designed PlanCurves to be planner and domain agnostic, making it usable for all practitioners. In particular, we have described how users can use it to customise the interpretation of plans for their needs and proposed several uses for it. While it is presently being used within a larger project with real-world industrial applications, we believe it can be used by others for a variety of purposes.

We have laid out possible extensions of this work, and hope it enables future collaboration within the community to facilitate the communication of plans to users.

Acknowledgement

This work was funded and supported by the ORCA Hub (`orcahub.org`), under EPSRC grant EP/R026173/1. PlanCurves is available at: `strategicfutures.org/PlanCurves`.

References

Bach, B.; Shi, C.; Heulot, N.; Madhyastha, T.; Grabowski, T.; and Dragicevic, P. 2015. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE transactions on visualization and computer graphics* 22(1):559–568.

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; et al. 2012. Europa: A platform for ai planning, scheduling, constraint programming, and optimization. *4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*.

Batrinca, L.; Khan, M. T.; Billman, D.; Aydemir, B.; and Convertino, G. 2013. A timeline visualization for multi-team collaborative planning. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 157–162. ACM.

Borgo, R.; Kehrer, J.; Chung, D. H.; Maguire, E.; Laramee, R. S.; Hauser, H.; Ward, M.; and Chen, M. 2013. Glyph-based visualization: Foundations, design guidelines, techniques and applications. In *Eurographics (STARs)*, 39–63.

Brandes, U., and Pich, C. 2007. Eigensolver methods for progressive multidimensional scaling of large data. In Kaufmann, M., and Wagner, D., eds., *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 42–53.

Chakraborti, T.; Fadnis, K. P.; Talamadupula, K.; Dholakia, M.; Srivastava, B.; Kephart, J. O.; and Bellamy, R. K. 2017. Visualizations for an explainable planning agent. *arXiv preprint arXiv:1709.04517*.

Coles, A.; Coles, A.; and Martinez, M. 2018. International planning competition 2018 temporal tracks. https://ipc2018-temporal.bitbucket.io/. Retrieved November 5, 2019.

Do, M.; Okajima, K.; Uckun, S.; Hasegawa, F.; Kawano, Y.; Tanaka, K.; Crawford, L.; Zhang, Y.; and Ohashi, A. 2011. Online planning for a material control system for liquid crystal display manufacturing. In *Twenty-First International Conference on Automated Planning and Scheduling*.

Edelkamp, S., and Mehler, T. 2005. Knowledge acquisition and knowledge engineering in the modplan workbench. *Proceedings of the First International Competition on Knowledge Engineering for AI Planning* 26–33.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. *arXiv preprint arXiv:1709.10256*.

Fraternali, P.; Castelletti, A.; Soncini-Sessa, R.; Ruiz, C. V.; and Rizzoli, A. E. 2012. Putting humans in the loop: Social computing for water resources management. *Environmental Modelling & Software* 37:68–77.

Gower, J. C. 1971. A general coefficient of similarity and some of its properties. *Biometrics* 857–871.

Gupta, S.; Dumas, M.; McGuffin, M. J.; and Kapler, T. 2016. Movementslicer: Better gantt charts for visualizing behaviors and meetings in movement data. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, 168–175. IEEE.

Haas, W., and Havens, W. S. 2008. Generating random dynamic resource scheduling problems. In *ICAPS 2008 Workshop on Knowledge Engineering for Planning and Scheduling*. Citeseer.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

Hoogendoorn, M.; Jonker, C. M.; Schut, M. C.; and Treur, J. 2006. Simulation, visualization, and validation of adaptive multi-agent organizations in naval applications. *SIMULATION SERIES* 38(1):377.

Howey, R.; Long, D.; and Fox, M. 2004. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301. IEEE.

Kim, J., and Blythe, J. 2003. Supporting plan authoring and analysis. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 109–116. ACM.

Kirkpatrick, A. E.; Dilkina, B.; and Havens, W. S. 2005. A framework for designing and evaluating mixed-initiative optimization systems. In *ICAPS Workshop on Mixed-Initiative Planning and Scheduling, held in conjunction with the Fifteenth International Conference on Automated Planning and Scheduling, Monterey, California, June*.

Lindsay, A. 2019. Towards exploiting generic problem structures in explanations for automated planning. In *Proceedings of the International Conference on Knowledge Capture*.

Martin, M., and Geffner, H. 2004. Learning generalized policies from planning examples using concept languages. *Applied Intelligence* 20(1):9–19.

McCluskey, T., and Simpson, R. 2006. Tool support for planning and plan analysis within domains embodying continuous change.

Muise, C. 2016. Planning.domains. *ICAPS system demonstration*.

Newton, M. A. H.; Levine, J.; Fox, M.; and Long, D. 2007. Learning macro-actions for arbitrary planners and domains. In *ICAPS*, volume 2007, 256–263.

Porta, M. 2000. Visps, a visual system for plan specification. In *Proceedings of the working conference on Advanced visual interfaces*, 307–310. ACM.

Porteous, J.; Teutenberg, J.; Pizzi, D.; and Cavazza, M. 2011. Visual programming of plan dynamics using constraints and landmarks. In *Twenty-First International Conference on Automated Planning and Scheduling*.

Pralet, C.; Roussel, S.; Polacsek, T.; Bouissière, F.; Cuiller, C.; Dereux, P.-E.; Kersuzan, S.; and Lelay, M. 2018. A scheduling tool for bridging the gap between aircraft design and aircraft manufacturing. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

Sampath, K.; Tezabwala, A.; Chabrier, A.; Payne, J.; and Tiozzo, F. 2013. Integrated operations (re-) scheduling from mine to ship. In *Twenty-Third International Conference on Automated Planning and Scheduling*.

Scott, S. D.; Lesh, N.; and Klau, G. W. 2002. Investigating human-computer optimization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 155–162.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itsimple 2.0: An integrated tool for designing planning domains. In *ICAPS*, 336–343.

Vaquero, T. S.; Silva, J. R.; Ferreira, M.; Tonidandel, F.; and Beck, J. C. 2009. From requirements and analysis to pddl in itsimple3. 0. *Proceedings of the Third International Competition on Knowledge Engineering for Planning and Scheduling, ICAPS 2009* 54–61.

Vodrážka, J., and Chrpa, L. 2010. Visual design of planning domains. In *Proceedings of ICAPS 2010 workshop on Scheduling and Knowledge Engineering for Planning and Scheduling (KEPS)*, 68–69.

Vrakas, D., and Vlahavas, I. 2003. A graphical interface for adaptive planning. In *Proceedings of the Doctoral Consortium of the 13th International Conference on Automated Planning and Scheduling*, 137–141.

Ware, C. 2012. *Information visualization: perception for design*. Elsevier.