Roles that Plan, Activity, and Intent Recognition with Planning Can Play in Games

Richard G. Freedman and **Shlomo Zilberstein**
College of Information and Computer Sciences
University of Massachusetts
Amherst, MA 01003, USA
{freedman,shlomo}@cs.umass.edu

Abstract

Planning is one of the oldest areas of research within artificial intelligence, studying the selection of actions for accomplishing goals. The more recently established areas of plan, activity, and intent recognition instead study an agent's behavior and task(s) given observations of its chosen actions. While these areas have been independently studied and applied to games in the past for both understanding player behavior and developing game characters, the potential for their integration presents even more opportunities via adaptive interaction with the player. In this manuscript, we discuss recent research on the integration of these areas and investigate potential uses for such integrated systems in games.

1 Introduction

The design, development, and even playing of games can be studied from interdisciplinary perspectives. The involvement of artificial intelligence (AI) in these studies spans many challenges that have been identified over the years in venues such as AAAI's AI for Interactive Digital Entertainment (AIIDE) and IEEE's Computational Intelligence and Games (CIG). Among the approaches proposed, the areas of planning and plan, activity, and intent recognition (PAIR, when referred to altogether) have been used almost independently of each other. However, there has been recent interest in integrating these methods to create more adaptive dynamic interactions between humans and computational agents. While some plan recognition methods build heavily on existing planning algorithms, that is not what we refer to as integration of planning and recognition. Instead, we focus on integrating the recognition process with the response planning process so that each process benefits from the information produced by the other in real time. There are several potential benefits for such integration of planning and PAIR in games with respect to adaptive content generation. We will introduce these in the remainder of this manuscript following a brief background on planning and PAIR along with some of their past applications to games and game-related research.

Planning

One of the earliest challenges posed to the AI community involves machines being capable of making autonomous decisions at or above the level of human experts. This led to the establishment of the planning and scheduling community that particularly studies representation of tasks, problem solving under various conditions ranging from uncertainty to resource constraints, and higher-level decision making processes such as metareasoning. Formally, a planning problem $\mathcal{P}$ is generally defined by a domain $\mathcal{D}$ that models the world and an objective that drives the agent's decision making. $\mathcal{D}$ describes a state space S and the set of actions A that transition between states, which can range from logic-based factored representations such as the STanford Research Institute's Problem Solver (STRIPS) (Fikes and Nilsson 1971) and Planning Domain Definition Language (PDDL) (McDermott et al. 1998; Fox and Long 2003) to grammar-inspired breakdowns of complex actions into simpler ones such as hierarchical task networks (HTNs) (Erol, Hendler, and Nau 1994) to direct enumerations with transition probability tables such as Markov decision processes (MDPs) (Bellman 2003). Objectives also vary greatly depending on the assumptions and formulation. For example, STRIPS and PDDL both specify an initial state $I \in S$ and goal conditions (logical statement G) that indicate which states satisfy the objective, HTNs further include some high-level task/action $T \in A$ that must appear at the root of the solution's tree, and MDPs measure the desirability of states with a reward function $R : S \to \mathbb{R}$ rather than specifically provide an initial and goal state. Solutions π are typically sequences of actions $a_1, a_2, \ldots \in A$ called plans or functions mapping states to actions $\pi : S \to A$ called policies.

Planning has been used in quite a few game-related applications besides the typical motion planning tasks for character navigation (Rabin and Sturtevant 2016). The game FEAR (Orkin 2006) developed its own representation for AI-controlled agents based on high-level planning representations such as STRIPS and PDDL, and Kelly, Botea, and Koenig (2008) used HTNs to create unique schedules for each non-playable character (NPC) to follow. A planning formulation was also used to create actions as game mechanics given an initial configuration and completed configuration of a level (Zook and Riedl 2014). This enabled content creators to receive suggestions based on what the player-

controlled characters need to do in simulation to complete the level.

Plan, Activity, and Intent Recognition

The more recently established field of plan recognition (Goldman et al. 2011) instead studies the identification of an agent's task given observations of its chosen actions, while intent recognition aims to predict the agent's upcoming actions or specific goal state from these observations. On the other hand, activity recognition develops higher-level interpretations of lower-level events, ranging from descriptions of raw sensor data to more complex descriptions of sequences of simpler actions like button-presses. Building on top of the planning formulation above, a recognition problem receives a sequence of observations $o_1, o_2, \ldots \in \mathcal{O}$ as input and returns some form of context from a library $c \in \mathcal{L}$ depending on the type of recognition and representation. For plan recognition, it is often the case that $\mathcal{O} = A$ and $L = \{\pi_1, \pi_2, \ldots\}$ so that some subsequence of a plan is observed and the rest should be filled in as an 'explanation'. Intent recognition also frequently uses $\mathcal{O} = A$, but $L = \{G_1, G_2, \ldots\}$ to instead identify the driving motivation behind the actions performed or $L = A$ to predict upcoming actions implied by the previous ones. Activity recognition usually assigns $L = A$ as the higher-level interpretation of lower-level inputs that have some other set of symbols $\Sigma = \mathcal{O}$.

Plan recognition with probabilistic bounds has been used with StarCraft to identify player strategies represented with HTNs (Synnaeve and Bessière 2011). Poo Hernandez, Bulitko, and Spetch (2015) recognized themes from player choices in a choose-your-own adventure story to guide the events so that they better connected emotionally with the player; emotions were tracked using a MDP with player choices representing the actions. Likewise, predicting a player's intent of hiding locations in a first-person shooter game allows computer-controlled agents to better decide where to go (Tastan, Chang, and Sukthankar 2012). Using stacked denoising autoencoders rather than traditional clustering, Min et al. identified general tasks players were performing in an educational game that did not explicitly reveal the goals to the players; underlying user goals in educational simulators have also been recognized using HTN-like representations called exploratory grammars (Mirsky, Gal, and Shieber 2017). However, more than single-player strategies can be identified. Hajibagheri et al. (2015) studied logs from massively multiplayer on-line games in order to study the formation and evolution of ad-hoc groups.

2 Integration of Planning and Recognition

Integration of planning with at least one of PAIR introduces the potential for more dynamic interaction. An early framework to propose this concept was specially developed for use in a factory domain (Levine and Williams 2014) where the workers' possible sequences of actions are typically known in advance. Thus it was possible to monitor a worker's progress up to specific *branching points* where the worker could perform one of several actions for the overall task; then the planning system identified what would not be accomplished from the chosen branch and assigned tasks to partner robots that completed them in time (for example, providing a screw if the worker picked up a screwdriver when both were on the table). Similarly, Geib, Craensen, and Petrick (2016) parsed HTNs to identify a human's task and predict upcoming subtasks, which a robot could then confirm and ask for permission to do instead. If given permission, the robot ran a planner to solve the subtasks while avoiding further interference with the human. As an attempt to compromise the openness of the latter approach (not monitoring a predetermined sequence of actions) with the actual opportunity for agents to interact, a method for plan recognition derived from off-the-shelf classical planners (Ramírez and Geffner 2010) was modified to predict necessary goal conditions that the interactive agent can then solve as a two-agent problem using the same planner (Freedman and Zilberstein 2017). This allows adaptive interaction between the human and agent because the recognized behavior can change over time, which also influences the planner's chosen actions, and vice-versa.

CPU-Controlled Characters

While the works above only use applications with robotic interactive agents, they can easily be generalized to interactive agents in games. Often operated by expert systems such as finite state machines, the artificial intelligence of game agents has frequently been hard-coded by programmers. Such a design choice poses issues depending on the relationship between the character and player, but adaptive interactive agents present an opportunity to address them. We will consider the three interaction types proposed by Freedman and Zilberstein (2017) as analogues of these relationships.

Adversarial The stereotypical enemy characters commonly found in games are designed to prevent the player from making progress. The use of expert systems for their artificial intelligence guarantees that the designers can create specific challenges for each type of enemy that the player will learn to overcome. However, this patterned behavior does not acknowledge the player's style significantly and can be countered systematically once the player has learned the pattern and identified a weakness. A more adaptive adversarial agent will instead react against players based on predictions of their goals and upcoming action decisions, which may provide a deeper sense of consequence to a player's choices. Furthermore, the use of planning for a response can avoid the monotony of a finite state machine's decision for a given game state, especially if there are multiple decisions to make or slightly less optimal actions are selected based on a satisficing method. As satisficing planners are more interested in finding any solution rather than the best one, these may be more ideal for games with real-time performance constraints.

Assistive Buddy characters that cooperate with the player to help level progression (or accompany the player as a component of a challenge) have become more complex over the

years. Rather than follow the player's character around the screen and act under a simpler finite state machine than most enemies, efforts are now taken to enhance the buddy's decisions to reflect unique personalities and develop a sense of companionship with the player throughout the play experience (Dyckhoff 2015). Most these enhancements are still created manually using human experts and precomputed formulas so that the buddy does not get in the player's way, which may interrupt and/or detract from the player's enjoyment and experience — this should be prioritized because players typically chose to participate in the game for their own entertainment.

Hence, if a buddy character pays attention to the players' decisions through PAIR algorithms, then it has a context of what the player is likely doing within the scenario. This information is very useful for creating a task that the buddy character can plan to accomplish under constraints that further avoid player interference. For example, if the player is focusing on a particular task within a challenge such as solving a puzzle component or targeting a specific subset of enemy characters, then it may be best for the buddy to work on the remainder of the challenge so that the player is not burdened with as many tasks and feels that the buddy is contributing to the group's success. Likewise, if the player can be recognized as retreating from a situation, then the buddy character may decide to stay slightly behind and defend when the likelihood of successfully retreating is lower and the buddy can sustain (self-sacrifice is not as ideal in many buddy character situations unless used as a plot device), or the buddy character may retreat alongside the player and aid the escape process when it cannot sustain. These situations can be assessed using planning to determine which choice is best and then acting out the decision. Such cases can portray a buddy's personality and develop companionship via acting responsively with respect to the player.

Independent NPCs usually compose the majority of characters in a game because they represent everyone who is not controllable, often excluding enemies and buddies. While some genres such as fighting games have few NPCs, other genres such as role playing games with large scale story-driven gameplay contain many NPCs. Due to their presence throughout the game at so many moments, it can be overwhelming to create unique and diverse personalities for all of them. Memory limitations once meant they were also given little code so that their interactions with the player were simple and static. However, artificial intelligence techniques are creating more dynamic NPCs as they can help the most with preserving the players' suspension of disbelief (Champandard, Champandard-Pail, and Wisniewski 2003 2017). Besides personalizing NPC dialogue (Kerr and Szafron 2009) and schedules for who appears where at various times (Kelly, Botea, and Koenig 2008), we can use planning on its own to allow NPCs to act within the environment. However, because these tasks may interfere with the player's progress, we may consider the integration of PAIR to adapt the NPC's action selection. Then the NPCs can still plan to achieve their own goals while also ensuring that the player is not stopped from doing what she is predicted to do. This has the potential to be a greater challenge when there are many NPCs near the player because all the NPCs need to accommodate each other as well.

Adaptive Gameplay

In addition to characters having agency, we may also regard the game itself as an agent. This perspective allows us to further explore adaptive interaction with respect to the player beyond just character choices, but also gameplay choices. Though the designers and developers have a specific experience that they wish to share with or express to players, it is not always possible to convey the message the same way to everyone. Hence adapting the gameplay to adjust how the experience is delivered to each player has the potential to better connect designers with their playerbase as well as personalize the player's experience.

Progression Refinement Each unit of game content (level, event, etc.) is often refined and ordered after multiple iterations of design, develop, and playtesting in order to present the entire game in a manner that feels fair and engaging. Depending on a player's experience and skills, though, the chosen ordering may not be ideal. Thus the simplest example of a possible application for personalization is altering difficulty level. Depending on the player's style, PAIR may be able to extract trends such as preferred strategies and commonly used action patterns. Then the game may rearrange content based on how it relates to the player's recognized style. Content that can be successfully played using methods more familiar to the player's common strategies should technically be easier in difficulty while content that requires methods less familiar to these common strategies will likely be harder in difficulty. Thus, rather than force progression through the content as the designers preassigned it, the game may reorder it for a difficulty/learning curve closer to the one intended by the designers' initial ordering.

However, reordering requires some level of caution, especially when there are prerequisites for some content such as completing a certain task from another unit of content in the game. This is especially important for story-driven games because some elements of the story may rely on other events first happening or a spoiling revelation may occur that will reduce the enjoyment of other story content if it was not yet experienced. Riedl and Young's (2010) and Cheong et al.'s (2016) work on computer-generated stories using planning techniques can help to maintain ordering constraints between content while rearranging it based on the recognized player strategies. Open world and sandbox games can even use this integration of PAIR and planning to enforce a sequence of events without restricting linear progression through the world. Given the recognized intents of where a player is going, placement of triggers for events may be planned so that linearized content still unfolds with the player's choices. Though an illusion of choice, this readjustment to the content (revising characters, locations, etc.) allows the player to follow a nonlinear path through the game based on personal preference of exploration through the world.

Procedural Content The introduction of procedural content to a game can lead to greater replay value by generating original content during gameplay. However, many games with procedural content currently perform this generation process randomly where the individual pieces can always be combined without constraints. There has been research on developing procedural content with a greater focus on solution strategies (Smith et al. 2011), but these strategies are currently developed by hand or randomly. Given a player's recognized strategies and play styles, though, we should be able to personalize the set of strategies needed to complete the challenges posed by the newly generated content. As described above, this can determine the personal difficulty level of the procedural content. Additionally, to determine the extent to which the content is solvable, we can use planning to simulate a player and evaluate its performance in a manner similar to the process used for validating whether a generated set of mechanics can solve a given level (Zook and Riedl 2014).

In addition to adapting procedural gameplay content, there is also procedural media content ranging from dynamic audio (Collins 2008) to automatically generated dialogue (Kerr and Szafron 2009). While dialogue can be approached similarly to character behavior, procedural non-diegetic audio (background music and sound effects not caused by the game's entities) may need a different perspective. In particular, recognizing a player's plans, activities, or intents does not present a direct correlation to sound in context. Likewise, planning for music composition or sound effect allocation will lack the semantic associations humans have culturally assigned to various sound patterns. Hence, though the potential for personalized audio that adapts with the player's in-game actions exists, it is not directly clear how this may be developed via the integration of PAIR and planning. We leave this as a question for the community to consider.

3 Discussion

There has been tremendous progress in both planning and PAIR, but existing recognition algorithms typically process the observed activity and output recognized information independent of planning the response. The response to the user is determined, if at all, using a separate planning process. In that sense, each research area in PAIR as well as planning mostly studies its own specific problem independently of the other. Recent research on the integration of planning and recognition shows the potential for adaptive interaction with human users. Depending on what we classify as the interactive agent, we propose applications of adaptive interaction with players in games using either characters that respond to the player's actions or the game itself adjusting content based on the player's style and strategies. This presents opportunities for enriching the player experience with personalized gameplay and interactions that respond dynamically based on the player's choices.

Acknowledgements

We thank the anonymous reviewers for their feedback. This work was supported in part by the National Science Foundation under grant IIS-1405550.

References

Bellman, R. E. 2003. *Dynamic Programming*. Dover Books on Computer Science Series. Mineola, New York, USA: Dover Publications.

Champandard, A. J.; Champandard-Pail, P.; and Wisniewski, F. 2003–2017. AIGameDev.com. `http://aigamedev.com/`. [Online; last accessed 21-November-2017].

Cheong, Y.-G.; Riedl, M. O.; Bae, B.-C.; and Nelson, M. J. 2016. Planning with applications to quests and story. In Shaker, N.; Togelius, J.; and Nelson, M. J., eds., *Procedural Content Generation in Games*. Cham, Switzerland: Springer International. chapter 7, 123–141.

Collins, K. 2008. *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*. The MIT Press.

Dyckhoff, M. 2015. Ellie: Buddy AI in The Last of Us. In Rabin, S., ed., *Game AI Pro 2*. Boca Raton, FL, USA: CRC Press. chapter 35, 431–442.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1123–1128.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proceedings of the Second International Joint Conference on Artificial Intelligence*, 608–620.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20(1):61–124.

Freedman, R. G., and Zilberstein, S. 2017. Integration of planning with recognition for responsive interaction using classical planners. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 4581–4588.

Geib, C.; Craensen, B.; and Petrick, R. P. A. 2016. Combining plan recognition, goal reasoning, and planning for cooperative task behavior. In *Fourth IJCAI Workshop on Goal Reasoning*, 1–8.

Goldman, R. P.; Geib, C. W.; Kautz, H.; and Asfour, T. 2011. Plan Recognition – Dagstuhl Seminar 11141. *Dagstuhl Reports* 1(4):1–22.

Hajibagheri, A.; Lakkaraju, K.; Sukthankar, G.; Wigand, R. T.; and Agarwal, N. 2015. Conflict and communication in massively-multiplayer online games. In *Proceedings of the Eighth International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, 65–74.

Kelly, J. P.; Botea, A.; and Koenig, S. 2008. Offline planning with hierarchical task networks in video games. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Kerr, C. M., and Szafron, D. 2009. Supporting dialogue generation for story-based games. In *Proceedings of the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference*.

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*, 490–498.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL – The planning domain definition language – Version 1.2. Technical Report CVC TR-98-003, Yale Center for Computational Vision and Control, New Haven, CT, USA.

Min, W.; Ha, E. Y.; Rowe, J.; Mott, B.; and Lester, J. 2014. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'14, 37–43. Raleigh, NC, USA: AAAI Press.

Mirsky, R.; Gal, Y. K.; and Shieber, S. M. 2017. CRADLE: an online plan recognition algorithm for exploratory domains. *ACM Transactions on Intelligent Systems and Technology* 8(3):45:1–45:22.

Orkin, J. 2006. Three states and a plan: The A.I. of F.E.A.R. In *Proceedings of the Game Developers Conference*, 1–18.

Poo Hernandez, S.; Bulitko, V.; and Spetch, M. 2015. Keeping the player on an emotional trajectory in interactive storytelling. In *Proceedings of the Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

Rabin, S., and Sturtevant, N. R. 2016. Combining bounding boxes and jps to prune grid pathfinding. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 746–752. Phoenix, AZ, USA: AAAI Press.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 1121–1126.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39(1):217–268.

Smith, G.; Whitehead, J.; Mateas, M.; Treanor, M.; March, J.; and Cha, M. 2011. Launchpad: A rhythm-based level generator for 2-d platformers. *IEEE Transactions on Computing Intelligence and AI in Games* 3(1):1–16.

Synnaeve, G., and Bessière, P. 2011. A bayesian model for plan recognition in rts games applied to starcraft. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'11.

Tastan, B.; Chang, Y.; and Sukthankar, G. 2012. Learning to intercept opponents in first person shooter games. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*, 100–107.

Zook, A., and Riedl, M. O. 2014. Automatic game design via mechanic generation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 530–536. Québec City, Québec, Canada: AAAI Press.