Introduction to Multi-Agent Planning

Stephen Gass

12/9/20

1 Introduction

Automated planning has been one of many successful application of artificial intelligence to domains of practical reasoning. Modern applications of automated planning include manufacturing, Mars rover missions, game playing, military maneuvering, traffic signal control, drones flight, optimization of electrical power networks, automated trading, and more [20][15][4][14]. In general, AI planners comprise some representation of an agent and its environment to generate a series of actions that the agent can execute to fulfill a goal. These AI planners can take the form of a wide variety of architectures with the most general distinction being between *model-based* and *model-free* algorithms. Both approaches have been proven successful in practical applications, and both have their own strengths and weaknesses. Model-based systems use symbol representation of the planning problem to search the action-space in order to find an efficient partial or total ordered plan that transitions from an initial state to a the goal state. Model-free systems use neural nets, often trained with reinforcement learning techniques, to generate a policy that results in the maximization of its long-term value function. Despite the tremendous rate of progress that these model-free systems have seen in the past decade with the introduction of deep neural nets, model-based planning systems still have the advantage of sampling efficiency, transferability, and high generality in various problems[15].

Another categorization that can be made is the distinction between domain-specific, domain-configurable and domain-independent planning systems. For model-based systems, domain specific planning systems are often successful in practice but are much more labor and knowledge-intensive to design and implement. For this reason, much of the focus of research and development is on domain-configurable or domain-independent systems. However, the labor and time savings of these two approaches often comes at a cost of quality of the resulting plan in practice[14].

Each of these domains brings with it a plethora of challenges, but over the decades, the field has seen significant achievements over the first AI planners. Two of the most notable of these developments include the introductions of the simulation-based approach known as STRIPS-like representation, named after the early planning system that pioneered this approach, and the hierarchical decomposition of tasks which enable planning systems to break down planning problems into various levels of granularity. The diversity of planning systems that have been proposed over the course of the development of the field is indicative of the diversity of the challenges that automated planning systems must face in building and executing a successful plan. To name a few, successful planners much have the ability to reason about cause and effect, reason temporally, reason under uncertainty, plan in dynamic environments, generate the lowest cost plan, generate contingency and continuous plans for when plans go awry or new information presents itself during the execution of a plan, and so on. One particular challenge to the field comes from planning in multi-agent environments in which multiple agents are acting on (and, therefore, changing) the environment simultaneously and must cooperate or negotiate in order to achieve their respective goals. Solving the challenges posed by multi-agent environments is a fundamental task in scaling up artificially intelligent systems, and many approaches have been devised which begin to solve these challenges.

2 Motivations and Challenges

Before getting into the details of various planning systems, I would like to illustrate more of these challenges that automated planning systems must overcome in order to be successful in practice. The following illustration of a planning problem is from [2]:

Person y lives in a multi-storey building where she can operate the entrance door by a buzzer. x can only enter the house while the door is temporarily unlocked by the buzzer. Furthermore, x must ring the doorbell first to notify y that she is there.

In this example, it's clear that the goal state of "x inside the building" can't be reached by the actions of a single agent but that x and y must coordinate their actions in some way. In a certain sense, x and y are specialists in that the actions and knowledge available to x are not the same as those available to y. In multi-agent planning (MAP) problems, this is known as heterogeneous agents and stand in contrast to homogeneous agents in applications such as swarm robotics and drone coordination. Furthermore, x and y also need to be able to reason and plan temporally because the door will only remain unlocked temporarily and only after the doorbell has been rung. Notice, too, the two agents need to have an expectation for what actions the other agent will take in response to certain events. For example, x needs to be able to model the belief state of y and reason as to what y will do in response to observing that the doorbell has been rung. Finally, this example is a case of active knowledge gathering in which agents are sensing events that occur in their environment and updating their beliefs and available actions accordingly, as well as updating their models of the belief states of other agents. Developing the semantics and theory to overcome complexities like these has been one of the fundamental focuses of multi-agent planning research.

3 Background

3.1 Classical Planning Problems

The early work on automated planning was primarily focused on what is known as Classical Planning problems which make a series of simplifying assumptions about the agent and its environment. These assumptions are stated in Table 1 [14]. These assumptions help to simplify the planning problem and its execution by limiting it to a fully observable, state-space problem with implicit time and a static environment, etc. However, much work has been done in the subsequent decades, and while generating optimal plans in an classical planning context is still an active area of research and competitions (such as the biennial International Planning Competition (IPC)), new planning languages and models have been developed to handle non-classical domains such as temporal reasoning, planning under uncertainty, constraint-based planning, dynamic/continual planning, etc.

Table 1: Classical Planning Assumptions

Number	Formal Assumption
Assumption A0	The system has a finite set of discrete states
Assumption A1	The system is fully observable
Assumption A2	The system is deterministic with respect to an agent's actions
Assumption A3	The system is static with no internal dynamics. It stays in the current state until some action is applied
Assumption A4	The only kind of goal is an attainment goal state. This excludes, for example, states to be avoided, constraint on state trajectories, and utility functions
Assumption A5	A solution plan is a linearly ordered finite sequence of actions
Assumption A6	Actions and events are instantaneous
Assumption A7	All planning is off-line planning, only considering the given initial and goal states

One example of the limited applicability of these assumptions in classical planning problems, alluded to by Assumption A4 in Table 1, is that agents are given no notion of intrinsically rewarding actions or situations that could be exploited along the way to the goal. This type of planning is known as *reward-based* as opposed to *goal-driven* models of classical planning and has been proven to be highly successful in practical applications, as in the application of model-free systems utilitizing Reinforcement Learning (RL)[15]. This reward-based approach is extremely useful for continuous planners and is not limited to model-free systems, e.g. [11]. In particular, model-based systems can use the concept of a *utility function* to generate some initial actions that create a trajectory for a plan before working out a complete and actionable plan that reaches the goals state. This is particularly important for reducing the cost of generating a plan, a critical consideration for continuous planners who repeat the planning process after each action is executed or task completed. That being said, model-based continual planners have not seen the rate of rapid success that RL systems have seen in recent years.

In general, most modern, model-based solvers handle complexity by attempting to transform or project the non-classical planning problem onto a classical planning space. In this way, planning researchers have gradually incorporated one complexity after another into their automated planning systems. But implicit in these classical planning assumptions is that the environment only contains a single-agent. The semantics required to solve some multi-agent planning problems will be discussed in the following section, but in practice, the primary obstacle for research in this area comes from balancing effective coordination with efficient communication. However, this is far from the only challenge that faces MAP solvers. The heterogeneity of agents, introduced in Section 2, creates a series of complications and burdens on the planning system. In practice, it is often more feasible to treat problems such as these as independent single-agent planning problems, thereby reducing the initial MAP problem to a series of classical planning problems. Additionally, practical application of MAP problems must deal with the challenges of privacy, negotiation over shared resources, human-machine interaction, and many more[20]. These are all areas of active research with many interesting advances.

3.2 Semantics of Planning

This section begins with the basic semantics of the MA-STRIPS planning system[1], an extension of the commonly used STRIPS system for MAP problems, and goes on to discuss various considerations, such as privacy, which the semantics of an ideal MAP solver would address. It should be said at the start of this discussion that there exists about as many semantics and languages as there are planning systems, but the MA-STRIPS framework is particularly common in model-based, multi-agent planning systems.

In MAP planners utilizing the MA-STRIPS framework, knowledge about the environment is represented as a set of first-order predicates which make up an agent's knowledge propositions, P^i. An agent, i, is an entity that can perform actions, a_i, on the environment. The entire set of actions that can be performed by i is denoted as A_i, where $a_i \in A_i$. Each action is a tuple of the form $a_i = \langle pre(a_i), eff(a_i), c(a_i) \rangle$, where $pre(a_i)$ is the set of preconditions that the agent must believe to be true in order for the action to be executed, $eff(a_i)$ is the set of effects represented as propositions to be added or deleted to the agent's knowledge base, and $c(a_i)$ is the cost of the action.

Therefore, a planning problem is defined in MA-STRIPS by the tuple $\langle P, \{A_i\}_{i=1}^{k}, I, G \rangle$, where P is the finite set of all public propositions known by all the agents, k is the number of agents in the system, A_i is the set of public actions that can be performed by agent, i, I is the initial state represented by additional propositions added to P, and G is the set of goal conditions, also represented by first-order propositions. Formally, a solution to the planning problem is a sequence of actions $(a_1, ..., a_j)$ that transition the initial state, I, to a state satisfying the goal conditions, G [17].

Notice the implicit assumptions around the execution of a plan in the definitions just listed. First, what the agent believes to be true at a given state of the environment, P^i, is assumed to be ground truth, which can raise potential errors in execution if an agent's beliefs deviate from ground truth. Also notice the implicit determinism: if an action is executed, the effects are guaranteed to take hold in the knowledge base. Again, this raises concerns around the agent's execution of a plan, but ultimately, this serves to simplify the planning process. It is simply worth keeping in mind the potential vulnerabilities and trade-offs that are made in the architecture of any planning system.

These definitions also introduce the concepts of *public* and *private* knowledge and actions. There are many practical applications of multi-agent systems that require a level of privacy, e.g. a company outsourcing some of its supply chain might not want to share sensitive internal dynamics through the process of automated coordination with the other company. For this reason, privacy is a crucial focus in multi-agent semantics. In general, propositions, entities, agents, actions, preconditions and effects can either be private or public. A piece of information is defined as private if it is known to an agent but ignored by the rest of the agents in the system. Public actions may have public and private effects, while private actions only have private effects. The public effects of public actions are always known to all agents, i.e. when an action changes the value (True or False) of a proposition, it is "observed" by all agents in the system simultaneously. The truth value of private information, on the other hand, is not shared with any agent except the owner of that private information.

The private and public information known by any particular agent is known as that agent's *local view* of the planning problem. Formally, the local view of a planning problem from agent i's perspective can be represented as $\pi_i = \langle P^i, \{\pi_i(A_j)\}_{j1}^{k}, \pi_i(I), \pi_i(G)\rangle$, where $P^i = \mathit{public}(P) \cup \mathit{private}(P)$, $\pi_i(I) = I \cap P^i$, and $\pi_i(G) = G \cap P^i$. One challenge facing MAP solvers is how to coordinate action between agents without revealing *any* private information.

Building a plan where each agent is reluctant to disclose it's knowledge and capacities might be challenging enough, but it's not the only way in which the planning system's knowledge about the planning task and environment might be incomplete. In particular, continuous planners attempt to take advantage of newly observed knowledge about the environment that might not have been available to the planning system during the initial planning phase. The continuous planner developed in [11] illustrates this concept with the simple example in which a planning agent is initially in a room with a closed box which contains a key. The information that the key is inside the box is *occluded knowledge* that is separated from the local view of the agent. If, however, the box was opened at some point through the actions of an agent, and the agent was able to observe the key, then this knowledge could be incorporated into the knowledge base of the planning agent and a new plan derived. Additionally, some systems are capable of modeling and planning in dynamic environments by representing the environment as a special kind of agent which takes actions according to probabilistic preconditions encoded in a Bayesian Net. This can be particularly useful for contingency planning as well as continuous planning problems[2].

As stated earlier, a solution to a planning problem is the total or partial ordering of actions that transition from possible initial states to the satisfaction of goal conditions. Therefore, the core mechanism of planning can be accomplished by a heuristic search of the action space, for which there are many approaches. To name only a couple, FF [7] and LAMA [16], which employs a variant of the FF heuristic, are staples among the winners of the International Planning Competitions (IPC)[20]. Another widely used approach is the belief-desire-intention (BDI) architecture which emphasizes the advantages of committing to a plan and thereby reducing the total space that an agent needs to search in order to arrive at a solution[12]. Notice, also, the difficulty in generating an effective plan when the action space is restricted to only the public actions of each agent. This is a fundamental challenge in distributed planning and has been addressed by a variety approaches.

For multi-agent systems, the planning search is only half the battle, so to speak, with the coordination of actions between the multiple agents often being considerably more difficult than generating the plan itself. In what are referred to as *unthreaded* planning systems, the planning and coordination phases are black-box separated from each other, and coordination between agents essentially becomes an independent satisfyability problem in which local tasks of agents are merged to satisfy the global objective. Unfortunately, the inefficiency and quality of this step is the primary obstacle to this approach of cooperative planning in multi-agent systems[20].

Finally, I'd like to add a few more words on the issue of privacy which, in practice, is one of the primary motivations for adopting a multi-agent system. However, despite this fact, it has been a relatively neglected topic of research for much of the field's history[9]. In general, privacy can be *imposed* by the planning system or *induced* by the method of planning. The authors of [18] attempt to formally quantify privacy leakage for MA-STRIPS, which, as discussed above, imposes privacy measures in managing the basic information sharing between agents by preventing the communication of predefined private information. These systems often employ MA-PDDL[8], the multi-agent planning language extended from the popular PDDL language common to international planning competitions, to mark propositions in an agent's knowledge base as *public* or *private*. Additionally, agents can be given different initialization files which contain the local view of the planning problem unique to that agent[20]. This is only the bare basics of what is required to ensure privacy in a MAP system, and the practical methods of ensuring privacy are an ongoing area of research. The primary point to be gathered here is that the formal treatment of privacy is a growing area of interest for the field of multi-agent planning, and any modern semantics which seek to incorporate MAP problems must give sufficient weight to the handling of private information.

Table 2: Conceptual schemes and example MAP solvers according to the number of planning and execution agents

		Planning Agents	
		1	n
Execution Agents	1	Single-Agent Planning FD [6]	Factored planning ADP [3]
	n	Planning For Multiple Agents TFPOP [10]	Planning By Multiple Agents FMAP [19]

3.3 Hierarchical Task Networks

One highly successful approach to planning in general, which applies to both model-based and model-free systems, is the use of hierarchical task network (HTN) planning for task decomposition in which a planning problem is recursively broken down into intermediate operations called tasks. The recursive nature of this approach allows tasks to be further decomposed into sub-tasks and sub-sub-tasks, etc., until the original problem comprises only "primitive tasks" which are essentially actions taken by agents in classical planning problems. Modified approaches to classical planning can then be applied to identify the appropriate action for each primitive task. The resulting solution obtained by an HTN planner is then a total or partial ordering of these actions, called a task network, in which all the preconditions for each action is met at its location in the network and the completion of the network results in the global goal state[5]. As one might intuit, this hierarchical approach is also very effective for multi-agent problems[20]. It also maintains a basic level of privacy between the agents because the task delegation process doesn't necessarily share the tasks of the other agents[20]. This is an example of *induced* privacy and is a rather simplistic, informal kind of privacy. Nevertheless, HTN has proven to be an extremely successful approach to planning in general, and has an intuitive aspect in its similarity to how humans generate long term plans for themselves.

3.4 MAP as an extension of single-agent planning

The desire to model systems with multiple agents naturally extended from single-agent systems with the practical applications of automated planning to real world problems. Table 2, excerpted from [20], helps to categorize the various architectures that these systems can take as well as provides the names of well-known systems in each architecture. The first distinction to be made is between *planning agents* and *execution agents*. Depending on the application, an execution agent can be anything from an simple actuator to a complex robot and is defined as an entity that can carry out some action on the environment. By contrast, a planning agent is responsible for generating the actions which are then communicated to the execution agents to fulfill. In the case of a single-agent system, these agents are one in the same, as shown in the cell in Table 2 at the intersection of 1:1. A simple extension of this would be planning for multiple agents in which a single planning agent communicates the necessary tasks to the various execution agents in the system. By analogy, this is like a single brain controlling multiple limbs in a biological organism to fulfill a single goal. This can be contrasted with factored planning in which multiple planning agents generate a plan for a single execution agent. This is perhaps analogous to a board meeting in which each executive offers their own perspective in deciding the direction of the company, however, there are many approaches to factored planning, and not all of them map onto this analogy very well. Nevertheless, finally there is planning by multiple agents, located at the intersection of n:n in Table 2 in which multiple agents generate their respective plans in a multiple agent environment. These systems can be *cooperative*, in which all agents take actions towards a single global goal (as is the case in the example in Section 2), or *negotiative*, in which agents negotiate over shared resources to optimally coordinate their actions in pursuit of their respective goals. Examples of negotiative systems include energy sharing buildings, smart grid technology, and trading agents. As one might expect, each planning scheme has its own strengths, limitations and relevant applications.

Another terminology commonly used to refer to this distinction between planning that is done by a single agent versus planning done by multiple agents is *centralized* versus *distributed* planning. In a technique known as interleaved planning, distributed planning agents each contribute to the construction of the composite action space and jointly navigate the space to find a solution plan[20]. Considering that the hardware for individual agents in a distributed system has the potential to be separated by many miles, it's no surprise that centralized planners are often more convenient and efficient MAP solvers than distributed planners. However, there are many applications in which centralized planning systems simply aren't applicable as in the case where privacy is a key concern. Therefore, there's a great deal of active research in the effort to reduce communications cost between agents during the planning and coordination phases of MAP problems. This paper will discuss how one system in particular attempts to address this challenge.

4 System Profiles

In this section, two planning algorithms will be outlined as examples of successful planners. The first is a model-based, multi-agent planning system first developed in 2015 called the Dependency-Preserving Projection (DPP) system. This model was able solve more benchmark problems than any other state-of-the-art privacy preserving planners at the time of its release, outperforming many of the top performers in the centralized track of the 2015 CoDMAP competition[17]. The second model that will be discussed is the single-agent, deep learning algorithm, also developed in 2015, called deep Q-network (DQN) which utilizes reinforcement learning to train a model-free planning system in an effort to maximize long-term utility[13]. Many systems have since been developed based on this framework of DQN, including multi-agent systems. In the interest of brevity, the intention here is simply to contrast the approach to planing between successful model-based and model-free systems.

4.1 DPP

The DPP algorithm was design specifically with privacy in mind and uses an extension of the MA-STRIP system called *privacy-preserving* MA-STRIPS. This system uses formal definition of private versus public knowledge and applies constraints to limit the amount of privacy leakage. The core emphasis of the design of DPP is on generating plans for actions that have *private dependencies*, i.e. a public action that has at least one private precondition. This system utilizes the concept of a regression tree by which it is able to share certain information about private dependencies without violating privacy constraints.

First, a high-level "projection" of this MAP problem is formed from only the public knowledge of the problem and each agent's public actions. By ignoring all private dependencies in this initial step and treating all agents as if they were controlled by a single planner, the DPP algorithm can solve this projection of the planning problem like any ordinary Classical Planning problem. Therefore, the DPP algorithm simply uses an off-the-shelf Classical Planning solver to generate a high-level blueprint from which further detailed plans are generated. To do this, each agent decomposes this high-level plan and applies their private knowledge to generate a series of tasks that will guarantee that each public action in the high-level plan has all of its preconditions met, public *and* private. At this point in the process, each agent is essentially acting as a single-agent planner and can ignore the other agents in the system. In this way, the original high-level blueprint, comprising only public actions, is extended with the private knowledge of each agent without agents ever having to share their private knowledge with the other agents. Notice that this algorithm is able to generate multi-agent solutions to MAP problems by projecting the original problem onto classical and single-agent problem spaces. This general concept of solving complex planning problems by utilizing Classical Planning algorithms is common throughout many MAP solvers. However, as the authors of [17] point out, this particular method of extending the high-level plan is not complete. The authors qualify this major limitation by restating the great experimental success the algorithm has seen with respect to the 2015 CoDMAP benchmarks. Clearly there is much progress to be expected from such a promising position.

4.2 DQN

Model-free systems have seen a tremendous amount of progress in recent years with the recent advances in deep learning. The deep Q-network (DQN) algorithm was the first successful algorithm to combine deep learning with reinforcement learning (RL) to create a stable agent capable of taking actions towards the long-term maximization of utility[13]. While DQN is not a planning algorithm by the definitions given above, it is the foundation of many RL systems that are capable of planning and coordination, including several multi-agent systems[15]. The objective of this section is to contrast the basic architecture of model-based with model-free systems specifically with respect to their approaches to planning sequences of actions.

DQN was first developed as a game playing algorithm, trained and evaluated on 49 of the classic Atari 2600 games. The algorithm utilized a convolutional neural network structure to map raw visual data (210x160 color video at 60 Hz) to the appropriate actions in each game state. The mapping of specific states to specific actions is known as the agent's *policy.* One of the fundamental concepts in RL is that the neural network structure approximates an action-value function, Q (hence the name, Q-network). Therefore, the general task of RL is to find the optimal policy that maximizes the long-term reward as calculated by the action-value function by iteratively training the neural network on large amounts of randomly sampled game-play data. The training process of the neural network can essentially be described as a kind of greedy search that seeks to optimize the Q value function. In practice, this takes a tremendous amount of data as well as dozens or hundreds of iterations (called training *epochs*) in order to achieve a successful algorithm. However, perhaps the most astonishing benefit to this approach is that no description of the game or model of the rules is needed in order for the algorithm to learn how to play the game successfully. This is what enables the algorithm to successfully learn how to play multiple games with surprisingly diverse objectives, constraints, and graphical representations.

4.3 Comparison

DQN and its variants' ability to generate basic concepts (e.g. obstacles in a game to be avoided) from raw sensory data is perhaps it's greatest advantage over model-based systems. While the requirement of large amounts of training data is no doubt a heavy burden, deep learning systems such as DQN have been extremely successful in practical applications in recent years in part because of their ability to process sensory data. This is in contrast to the model-based solvers in which the planning problem, including the model of the environment, the available actions of all the agents, as well as clearly defined initial and goal states, need to be carefully defined in near completeness in advance.

But while this naive mapping of states to actions might be successful at playing games, it's no substitute for the ability to *reason.* Put bluntly, the concept of an optimal policy is a cheap substitute for a *plan.* The knowledge represented in model-based systems is more transferable than the training data that build model-free systems. The policies of DQN and other model-free systems are limited to the states that are represented in their training data set, while model-based systems are able to explore much larger action spaces. To borrow a phrase from Steven Pinker, the current state of model-free neural nets are nothing but "idiot savants" when the ultimate goal is to build a complete and efficient planning algorithm that can solve any planning problem for which there is a solution.

The short comings of generating *policy* in comparison to *planning* can be further illustrated by noticing which particular Atari 2600 games DQN was most successful at playing and which games it failed at. In total, DQN performed better than human expert players in 29 of the 49 games in which it was tested and variants of DQN have since surpassed its record. This record is no doubt an important milestone, but it's far from the capabilities we should want from a general automated planner. To put it roughly, to be impressed with DQN's level of game play is a bit like being impressed that a hand-held calculator is better at arithmetic than a human. The games in which DQN is most successful are those which require quick reflexes and little planning (e.g. Pinball and Boxing). Games which require even small amounts of reasoning capacity (e.g. Ms. Pac Man and Montezuma's Revenge) are simply too difficult for the algorithm. This is, of course, because DQN has the capacity to make quick decision according to its optimal policy but lacks the ability to reason or make inferences about future game states. However, it must be pointed out that DQN optimizes for *long-term* action-value, and thus, appears to be able to form some strategic policies for particular games (e.g. Breakout).

Model-free systems certainly do have particular advantages over model-based systems, especially in their ability to process high-dimensional inputs and generate policies without tedious modeling of the environment or the problem to be solved. There is still much progress to be made, but considering the rapid developments that have taken place in recent years, this might be expected in relatively short order. The field of multi-agent deep reinforcement learning, known as MADRL, is a diverse and dynamic field, but the techniques and approaches to this domain of problems is outside of the scope of this paper. Suffice it to say that these systems will have to address the challenges that many model-based systems have been grappling with for decades, namely, coordination, cooperation, negotiation, communication, and more.

5 Conclusion

Given the rapid progress in both model-based and model-free systems, there is a lot to look forward to in terms of the direction of future research. One exciting possibility is to create a model-based systems that can deal with high dimensional data as successfully as model-free systems currently can. One possible approach would be to utilize a deep learning architecture within a model-based systems. Recent studies have investigated this possibility for the single-agent domains, but none have been extended to multi-agent systems[15]. It seems clear that the strengths and weaknesses of model-based and model-free systems complement each other well, and a successful integration of the two could potentially yield exciting results.

Additionally, scaling to large, multi-agent systems holds many challenges for both model-based and model-free systems. Model-free systems have a face obstacles with large training data requirements, the time cost of training multiple agents, and training heterogeneous agents, while model-based systems face challenges with high communication costs during planning and coordination as well as privacy. The problem of scaling is extremely relevant to practical applications and further advances would be immensely beneficial.

An extremely exciting research frontier is human-machine interaction and cooperation. Planning with external objects and goals is ultimately only a fraction of what we would want an intelligent agent to be able to accomplish. Extending planning systems to incorporate ”humans in the loop” requires a variety of additional advances including the ability to make logical inferences and model its own and other's mental states and intentions to a high degree. Ideally, human in the loop architectures could bring the scalability and precision of autonomous systems to the creativity capacities that humans naturally wield. Today, the best examples of this concept might be the control systems in a chemical or manufacturing plants, but in the long term, we would wish for intelligent agents to be able to reason about human intentions and be able to model the consequences of plans and provide humans with feedback on potential trade offs between proposed plans and potential alternatives. Yet another primitive example of this concept that is present in the world today is freestyle chess, where human chess players are allowed unrestricted access to a computer to run simulations of the match and choose the best strategy according to the feedback from the computer simulation. These so-called centaur players can routinely beat human-only and compute-only players. Not to mention, freestyle chess is itself an informal kind of multi-agent automated planning, where the other agent in the system is the simulated opponent. This is far from the traditional approaches to multi-agent planning that have been introduced in this paper, but it holds many interesting possibilities for future research.

The example of free style chess is instructive because it demonstrates that automated planning systems don't need to be *fully* automated to be successful. This emphasis on full autonomy is a source of bias that research competitions appear to bring to the field. With due respect, planning competitions do provide a range of benefits to the field, from increased publicity to the standardization of semantics and languages, as well as generating benchmark comparisons to evaluate the pantheon of conceptual schemes and algorithms. But an over-reliance on competitions to keep the field moving forward might also introduce bias in the direction of research, particularly when it comes to an emphasis on domain-independent planning and fully autonomous planning systems. There are many applications in which neither of those characteristics are strictly desirable, let alone necessary. This is simply to say that researchers should generally be sensitive to potential blind spots of the research in their field.

This potential bias in the automated planning community has an analogous counterpart in the AI research field generally. The emphasis, admittedly from my lowly perspective, is on building human-level or super-human agents in every capacity. As one anecdote of this, notice the sheer excitement with which the authors of [13] explain how DQN was able to beat a human at a few dozen arcade games. Admittedly, the paper emphasizes this anthropocentric perspective in part because so many components of the algorithm, from convolution filters to the mechanisms of the neural networks, to reinforced learning itself, were inspired by biological realities. But the success of freestyle chess should serve as a lesson that research in automated intelligent systems should not seek to simply replace human capacities, but to complement them.

References

[1] Ronen Brafman and Carmel Domshlak. “On the complexity of planning for agent teams and its implications for single agent planning”. In: *Artificial Intelligence* 198 (May 2013), pp. 52–71. DOI: 10.1016/j.artint.2012.08.005.

[2] Michael Brenner. *Planning for Multiagent Environments – From Individual Perceptions to Coordinated Execution.* 2005.

[3] Matthew Crosby, Michael Rovatsos, and Ronald P. A. Petrick. *Automated Agent Decomposition for Classical Planning.* 2013.

[4] Jürgen Dix et al. “Planning in a Multi-Agent Environment: Theory and Practice”. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems.* ACM Press, 2002, pp. 944–945.

[5] Kutluhan Erol, James Hendler, and Dana S. Nau. “UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning”. In: 1994, pp. 249–254.

[6] Malte Helmert. “The Fast Downward Planning System”. In: *J. Artif. Int. Res.* 26.1 (July 2006), pp. 191–246. ISSN: 1076-9757.

[7] J. Hoffmann and B. Nebel. “The FF Planning System: Fast Plan Generation Through Heuristic Search”. In: *Journal of Artificial Intelligence Research* 14 (May 2001), pp. 253–302. ISSN: 1076-9757. DOI: 10.1613/jair.855. URL: http://dx.doi.org/10.1613/jair.855.

[8] Daniel L. Kovacs. “A multi-agent extension of PDDL3”. In: *The International Planning Competition.* 2012, pp. 19–27.

[9] Van Der Krogt, Cork Constraint, and Computation Centre. *Privacy Loss in Classical Multiagent Planning Roman.* 2009.

[10] Jonas Kvarnstrom. “Planning for loosely coupled agents using partial order forward-chaining”. In: *In Proc. SAIS.* 2011.

[11] Daphne H. Liu and Lenhart Schubert. “AN INFRASTRUCTURE FOR SELF-MOTIVATED, CONTINUALLY PLANNING AGENTS IN VIRTUAL WORLDS”. PhD thesis. University of Rochester, 2012.

[12] Felipe Meneguzzi and Lavindra De Silva. “Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning”. In: *The Knowledge Engineering Review* 30.1 (2015), pp. 1–44. DOI: 10.1017/S0269888913000337.

[13] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: Feb. 2015. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236.

[14] Dana S. Nau. “Current Trends in Automated Planning”. In: *AI Magazine* 28 (2007), pp. 43–58.

[15] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. “Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications”. In: *IEEE Transactions on Cybernetics* 50.9 (Sept. 2020), pp. 3826–3839. ISSN: 2168-2275. DOI: 10.1109/tcyb.2020.2977374. URL: http://dx.doi.org/10.1109/TCYB.2020.2977374.

[16] Silvia Richter and Matthias Westphal. *The LAMA planner: Guiding cost-based anytime planning with landmarks.* 2010.

[17] Guy Shani, Shlomi Maliah, and Roni Stern. “Stronger Privacy Preserving Projections for Multi-Agent Planning”. In: URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13095.

[18] Michal Stolba, Jan Tozicka, and Anton Komenda. “Quantifying Privacy Leakage in Multi-Agent Planning”. In: *ACM Trans. Internet Technol.* 18.3 (Feb. 2018). ISSN: 1533-5399. DOI: 10.1145/3133326. URL: https://doi.org/10.1145/3133326.

[19] Alejandro Torreno, Oscar Sapena, and Eva Onaindia. “Global Heuristics for Distributed Cooperative Multi-Agent Planning”. In: vol. 2015. June 2015.

[20] Alejandro Torreno et al. "Cooperative Multi-Agent Planning". In: *ACM Computing Surveys* 50.6 (Jan. 2018), pp. 1–32. ISSN: 1557-7341. DOI: `10.1145/3128584`. URL: `http://dx.doi.org/10.1145/3128584`.