Real-time team-mate AI in Games

A Definition, Survey, & Critique

Kevin McGee
National University of Singapore
Singapore
mckevin@nus.edu.sg

Aswin Thomas Abraham
National University of Singapore
Singapore
aswinthomas@nus.edu.sg

ABSTRACT

Many contemporary games are team-based and there is a growing interest in, and need for, advances in team-mate AI for games. However, although there have been surveys of agent AI in games, to date there has been no survey of work on team-mate AI. Furthermore, the concept of "team-mate AI" is not currently well delineated to distinguish between work on independently-acting agents that happen to be on the same side from work on agents the coordinate their behaviors and decision-making in terms of their team-mates behaviors, intentions, and the like. Also, it is important to distinguish between game AI that is used as an optimization technique from real-time game AI, so this paper proposes a definition for real-time team-mate AI (highlighted with examples by game genre), reviews work to date to implement real-time team-mate AI for games in terms of a number of AI research areas (e.g., coordinated action, prediction, learning), and concludes with a brief discussion of significant issues about the state of the art.

Categories and Subject Descriptors

A.1 [**General Literature**]: Introductory and Survey; I.2.6 [**Artificial Intelligence**]: Learning—*Induction*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*

1. INTRODUCTION

Many contemporary games are team-based and in some cases, those teams consist partly or entirely of non-player characters (NPCs). Such team-mate NPCs are driving a recent but growing interest in, and need for, advances in team-mate game AI. This is an area of work still in its infancy; indeed, researchers have commented on the relative lack of work in this area (compared to other aspects of game AI). For example, as recently as 2007, Jansen wrote, "In most current computer games however, artificial intelligences do not really cooperate with their companion. They co-exist besides their partner, doing their own tasks, without attacking the partner player" [10]. Statements like this capture the broad outlines of team-mate behavior in many team-based games – and also highlight the fact that much of what has been done does not seem to very *team*-like. It would be useful for researchers and developers to have a good model of *team-mate* AI (one that corresponds with our intuitions about good team-mates cooperating for the collective good), a more detailed picture of what has been done in this area, and the challenges that remain. However, although there have been surveys of agent AI in games [39, 19, 8], to date there has been no significant survey of work on team-mate AI (indeed, most surveys of game AI barely touch on the topic). This paper is intended to provide an initial definition of real-time team-mate AI and to survey the existing games and research that fall within this definition.

ICFDG 2010, June 19-21, 2010, Monterey, California, USA

2. REAL-TIME TEAM-MATE AI DEFINED

One of the issues that emerges upon examining the problem of team-mate AI is that the concept of a game "team-mate" (i.e., partner or "buddy") does not seem to be well-delineated in the game AI literature (although, see [30] for some relevant discussion from work on multi-agent systems). To be sure, there are many kinds of NPC behavior evident in team-mate-based games – and much work devoted to different aspects of such NPCs. However, such behaviors range from those that involve coordinating with (or considering) the player and other team members – to those that involve little or no attention to group members or needs. Compare, for example, two very different examples of behavior that one might find grouped under work on AI team-mates. On the one hand, there is the case of "covering fire"; an NPC supports the movement of the player in the context of attaining a shared, player-level game-goal. On the other hand, there is the case where members on a team of NPC soldiers always fire on opponents if they have a clear line of sight; in one sense, they are "on a team" – but although one could argue that such a rule implicitly takes team-mates into account, without further refinement or additional rules, such a rule does not distinguish between such things as line-of-sight being partially blocked by a tree or by a team-mate. More complex examples that highlight the differences include a player on a football team that must decide whether to move forward with the ball or to pass it to another player – compared to sports team "formations" that are implemented as a way of simplifying the technical problem of improving the ball-passing performance of NPCs.

This paper concentrates specifically on agent-centric work to develop NPCs that coordinate with their team-mates in order to attain collective goals. So, for the purposes of this paper, real-time team-mate AI involves team-mate NPCs that

- take into account the behavior, state, needs, goals, plans, or intentions of players on the same team
- as part of coordinated *behaviors* (e.g., passing the ball effectively to a moving team-mate) or *decision-making* (e.g., deciding between passing and shooting the ball)

- that are relevant to shared game-oriented goals (or sub-goals)
- when such coordinated behaviors or decision-making involve some amount of real-time, player-related uncertainty that requires real-time calculation, reasoning, inference, classification, constraint-satisfaction, or the like
- while prioritizing the player participation whenever possible

By this definition, then, contributions to real-time team-mate AI involve the development of real-time algorithms a) for behaviors or decision-making that take team-mates into account; b) for identifying (human) player preferences/styles, strengths, or weaknesses (to the extent that such things *impact* the team-mate oriented behaviors or decision-making); or c) real-time *learning* that helps the NPC to improve its human team-mate relevant behaviors or decision-making. This applies both to team-mates that are on the human player's team ("allies") to those on the team that opposes the human player ("opponents").[1]

There are some subtleties to this definition, particularly with regard to the emphasis on *real-time* AI. To be clear, this means that if a system was developed in which Bayesian reinforcement learning was used to offline to train a system to recognize different player types, the work is only summarized here if the taxonomy of player types is used as part of an actual real-time game algorithm (e.g., for real-time classification and reclassification of human players which is used as part of real-time team-mate gameplay). This means that the related work surveyed here may not be presented in terms of the primary contribution of the cited work (e.g., the main contribution of the hypothetical paper mentioned above may, in fact, be in terms of the use of Bayesian reinforcement to train the system, but for our current purposes, the work would be described in terms of how it impacts real-time team-mate AI in terms of real-time classification).

Perhaps the most relevant and significant game-play phenomena that cannot be determined entirely in advance are those that have to do with player performance, preferences, strengths and weaknesses, and intentions (e.g., having NPC team-mates learn/classify a player's style in real-time and incorporate this information into team-mate oriented behaviors and decisions). This means that the focus is on how the related work addresses uncertainties or unknowns about the human player(s) (e.g., where the player will move next) – or which are introduced because of human player actions (e.g., the location of NPC team-mates that are responding to the movement of the human player). One compromise that attempts to address the performance issues while taking unknowns about the player into account is to have the player train ally NPCs; for example, in *Black and White* the player can reward or punish creatures as a way of customizing their behavior to player-preferences.

This definition of real-time team-mate AI also highlights one of the classic debates in game AI: the tension between academic research on AI problems and the use of AI techniques to solve particular game development needs. To be clear, the focus in this paper is on the subset of academic research problems in real-time AI that need to be solved in order to solve actual real-time *game* problems – and which satisfy various real-time game requirements (for a discussion of some possible requirements, see [26]). This distinction can be illustrated by the way the topic of *real-time* (i.e., during gameplay) team-mate learning is covered in this paper. Contributions to real-time learning that *could* be done offline are not covered (e.g., [40]), since such work tends to treat games as viable testbeds for work on AI problems [11], rather than addressing actual real-time *game* problems. This does not mean that all academic research on team-mate AI is excluded, but such contributions are not included here unless there are actual real-time uncertainties or unknowns that need to be taken into account. The most obvious consequence of our distinction is what counts as relevant contributions to learning for real-time team-mate AI: what remains are contributions at the intersection of AI and game development. So, although there is much work on game-related AI learning that involves such things as the use of emergent self-organizing maps to reveal/create a taxonomy of player types [7], such work, though interesting and important, is excluded from this review unless it is the basis of a contribution to actual real-time, in-game team-mate AI (e.g., using the player types as the basis for real-time pattern recognition, player classification, or decision-making). This focus on "justified" real-time learning means that there is very little that counts (as recently as 2005, Bakkes et. al. noted that they could not find a single example of a commercial game that included online learning [2]). Of course, the absence of real-time learning in commercial games could be due to other factors – and we will note research games that included justified real-time learning – but overall, this is a challenging issue.

[1]In the interests of brevity, unless otherwise indicated, the terms "real-time team-mate AI" and "team-mate AI" are used interchangeably through-out the remainder of this paper.

2.1 Team-mate AI by Genre

Before turning to specific contributions to real-time team-mate AI, it is worth noting that many of the problems are genre-specific, so here we briefly highlight some relevant concepts and work by genre, noting some differences between teams that include one or more human players versus NPC-only teams.

Team Sport Games Team sport games include traditional sports such as soccer, American football, rugby, basketball, ice hockey, tennis, baseball, cricket, and curling. Typically, the player's role in a team-based sports game is to coordinate with team-mates to score against an opposing team. Often scoring is based on having or taking possession of a token (such as a ball) and positioning it somewhere while a team of defensive players attempts to prevent this. The main issue that needs to be addressed for ally team-mate AI in team sport games is to coordinate their movement in order to send/receive passes and to make use of knowledge about team-mates when deciding to pass or shoot/move. For example, in *ProEvolution Soccer*, *FIFA 10*, and *Madden NFL*, allies do such things as move forward to support attacking moves, avoid passes not intended for them, avoid blocking team-mate shots, and throw themselves in the way of attacker's shots. By contrast, the main issue that needs to be addressed for opponent team-mate AI in team sport games is to coordinate their movement in order to prevent the opponent team from scoring, and, for some games, to try and regain the initiative. For example, in *Madden NFL* and *ProEvolution Soccer*, the AI opponent team is able to adapt their play in various ways to the actual performance and preferences of the human player and his team.

Note that although robot-based football (e.g., *RoboCup*) is obviously a team-based sport with AI team-mates, the physical nature of the robot NPCs raises some different issues and in some cases require AI solutions to certain types of behavior, perception, and communication problems that do not arise (or do not require the same kinds of solutions) as for software-based team games. In the sections below, relevant research will be highlighted where appropriate.

Action Games The player's role in team-based action games is to control a character to shoot down enemies and complete missions. The main issue that needs to be addressed for ally team-

mate AI in action games is to coordinate supporting fire for human players (while avoiding friendly fire). For example, in games such as *Left4Dead*, *Star Trek: D-A-C*, and *Killzone 2*, allies do such things as provide covering fire, restore the health of the player at the expense of other allies in greater need, and drag wounded players back to cover. By contrast, the main issue that needs to be addressed for opponent team-mate AI in action games is to coordinate splitting the team and attacking the player from various location/directions in different cooperative ways (e.g., simultaneous attack, coordinated ambush, etc). For example, in *Half Life* opponents can coordinate to provide suppressing fire in attacks against the player's team.

Real-time Strategy Games The player's role in team-based, real-time strategy games (RTSs) is to make decisions about resource and town management, combat tactics, and army deployment; the player has to manage all these tasks and coordinate ally actions. The main issue that needs to be addressed for ally team-mate AI in RTS games is to coordinate activities at the unit level (such as when an entire group needs to coordinate taking cover). For example, in *Age of Mythology*, a titan in the team agrees to focus on fighting a titan opponent while the other smaller units face smaller position units. By contrast, the issue that needs to be addressed for opponent team-mate AI in RTS games is to coordinate the building and expansion of their civilization at both the unit level *and* at the overall level of the opponent's "side" (by analogy to the human player's level). For example, in *Warcraft 3*, there is a top-level AI that controls more specialized AIs and can decide to go on the offensive and coordinate unit-level activities to this end.

Role-Playing Games The player's role in team-based RPG games typically involves controlling one or more characters with different characteristics in order to solve quest-based challenges that involve encounters with enemies, allies, and mystical items. Ally team-mate AI as we define it here consists of partner characters – and supporting characters with which the player can interact to learn game-related information; thus, the main ally team-mate AI issues concern coordination of partner behavior and the development of team-oriented dialog/communication. By contrast, the main issue that needs to be addressed for opponent team-mate AI in RPG games is to generate coordinated combined attacks to challenge the player. For example, in *Baldur's Gate*, while "beholders" throw shooting spells from afar, demon knights come forward for close combat.

3. ADVANCES IN TEAM-MATE AI

Given the definition proposed, contributions are now summarized that highlight real-time team-oriented:

- classification: assigning observed elements to pre-existing categories (e.g., determining that a player is belongs to the pre-existing category of "aggressive" or "passive")
- prediction (e.g., anticipating where a player will move or where a player will kick the ball)
- action-selection (e.g., choosing whether to shoot the ball or pass it to a team-mate)
- adaptation and learning (e.g., real-time updates to assigned roles/behaviors, evolving progressively better tactical formations, etc.)

The next few sections we summarize the main contributions to team-mate AI in terms of some AI research areas such as coordinated action, problem-solving, planning, communication, learning. Of course, it is not always so clear whether certain work should be categorized as, for example, a contribution to classification or to action selection. This is not a criticism of the published work, but reflects the fact that the present paper is trying to highlight specific kinds of work. Thus, for example, it has sometimes been necessary to make a decision about whether the bigger part of the reported work involved real-time classification of some aspect of the game – or the real-time action-selection based on the real-time classification. The main point is that the way different work is organized below is simply meant to help orient the reader, not to make any strong claims about how the such work is organized.

3.1 Team-mate Classification

Work in this category includes such things as recognizing (or classifying) player behaviors or styles.

For example, a system has been developed that is able to classify player behaviors in *Quake II* [33]. Initially, player behaviors from recorded games are clustered using self organizing maps and each cluster is then assigned to a multilayer perceptron (one for view angle, the other for velocity adjustment); the classifier uses using back propagation to train a classifier that can identify player behaviors such as running and aiming. During gameplay, the classifier identifies the human player behaviors and the ally imitates them (e.g., the ally imitates the aiming behavior of the player when he attempts to shoot an enemy). Similarly, there is a proposal for how NPCs in shooter games can recognize player behavior and inform their team-mates [35]. The system is implemented such that logged test-player information is manually labeled to denote various behaviors; the logged data and corresponding behaviors serve as inputs and outputs for training of a naive Bayesian classifier in order to be able to identify player movement and combat behaviors (e.g., being idle, reloading, pulling back from a fight, engaging distant threats, moving forward and engaging targets, sneaking and preparing to ambush). The system also includes scripted support behaviors for the team-mate (e.g., moving near the player and covering him, keeping low and suppressing targets that engage the player). During gameplay, the player behaviors are identified by the classifier and the appropriate scripted support behaviors are executed (e.g., when the player moves forward, the allies provide cover from all sides and suppress opponents that engage the player).

In terms of classifying player styles, a system has been developed for football in which an opponent team's play style is classified [13]. Various defensive strategies of the opponent are labeled and an SVM classifier is trained based on this data so that the system can identify the opponent's defensive play strategy and then use a look-up table with different offensive plays (e.g. hook shot, pass, block) the team can perform. During gameplay, the classifier identifies the opponent's strategy and the team executes an appropriate offensive strategy. Similarly, a system has been developed that is able to classify the play style of the opponent in an RTS game called *Spring* [23]. A fuzzy classifier is trained to identify different opponents based on game features; as a result, the system can identify the play style of the opponent based on top level characteristics such as defensive or aggressive and lower-level features such as robots, tanks, bunkers, super weapons and the like. During gameplay, the classifier identifies the type of the opponent based on the type of units built (e.g., opponents are identified as aggressive if they build airplanes and tanks). A third example involves identifying the strategic style of a player in an RTS game [6]. The system is encoded with preference functions that contain various game features (e.g., military strength, scientific strength, population size) and a probabilistic preference model of different player types (e.g., Genghis Khan, Gandhi and Abraham Lincoln) is

created using different weighted combinations of the game features mentioned. During gameplay, the preference function is evaluated for the player and the closest matching profile is identified (this is subsequently used to predict player actions, e.g., if the player is identified as a Genghis Khan type, then the system predicts the player's diplomatic action will be hostile). Finally, there has been work to create NPCs that adapt their behaviors/responses in RTS games based on such player traits as "aggressive" or "defensive" [10]. Prior to gameplay, a neural network is trained to identify different characteristics of the player based on game features and different support actions that can be performed by the ally are scripted. As a result, the system has a classifier that can identify if the player is aggressive, neutral, or defensive. For example, during gameplay, if allies determine that the opponent team is being aggressive, defensive units are moved into place and counter-attacks are initiated.

3.2 Team-mate Prediction

There has been some success in developing real-time algorithms for predicting the movement of game elements.

A predictive system has been developed for hockey [12]. Using dead reckoning based on a model of the inertia and kinematics of game elements (e.g., the puck and players) are modeled in order to predict their future positions. So, for example, during gameplay an ally with possession of the puck can plan a pass to the player based on the predicted positions and movement of opponents and team-mates (and the puck).

Opponent NPCs have been developed for the game of *Pac-Man* that predict player moves [5]. The system has a look-ahead tree based on different kinds of pre-determined behavioral traits of the player (e.g., aggression, caution, planning, speed, control, skill, etc.); the tree represents future movement patterns for each of the possible behavioral features of the player. During gameplay, the actions of the player are input to the tree, the movement behavior that best matches the player is found, and the next move of the player is then predicted (e.g., when the ghosts are in the hunt mode, depending on the distance maintained by the player, the system might predict that the player will move away to eat pills which are farther from the ghosts).

Finally, a system has been developed to predict the movement patterns of the player for an FPS game [16]. The system works by having a set of pre-defined locations (e.g., room 1, room 2, etc.), a semantic representation of locations the player can go to, and a means of capturing and representing sequential patterns of behavior (e.g., the sequence in which a player enters a collection of rooms). During gameplay, the sequence in which the player visits rooms locations in a particular area is monitored and subsequent player movement is predicted for similar scenarios in the future (so that, for example, when ally team-mates predict the room the player will search first, they can split up to advance and search other rooms).

3.3 Team-mate Action-Selection

There have been some contributions in terms of developing real-time algorithms for selecting appropriate actions in sports games, tactical actions in FPS games, and army deployment strategies in RTS games.

Some work has been done on coordinated action selection for the game of baseball [36]. The offensive, defensive and shared plans (e.g., pass, shoot, drive to the basket) for each ally in the team is coded in terms of an FSM representing common game states (e.g., getting the ball in play, getting ball into the front court, rebounding the ball after it has been shot). During gameplay, brute force is used to pick a plan and execute it. For example, the decision to make a particular shot is based on assessment of player's skill, preconditions for the shot, the position of the allies and opponents on the court, and the like. More complex coordinated actions involving anticipation have been developed for baseball [15]. Initially, each fielder is given a fixed slot (depending on their role) and coordinated formations for all kinds of team play are scripted – and the situations in which the formations should be executed are also coded. During gameplay, after the possibility of a particular formation is detected, the slots of the team-mates are set in motion for the team-mates to move in a coordinated manner (e.g., when the possibility of a double play is detected, the fielders are brought into place to throw out both batters).

For action games, there is work on action-selection based on different game-states [20]. The system includes rules for team-mate support actions for various game states which are grouped under player awareness (e.g. ensuring that the allies get out of the player's way), threat awareness (e.g., decisions about whether to engage an enemy) and environmental awareness (e.g. looking for cover); the result is a layered architecture in which correlations between game states and support-actions are arranged based on priority. During gameplay, team-mates query this architecture to obtain rules for support action execution (e.g., when the team-mate is in front of the player and a threat is nearby, the priority of player awareness is higher than threat awareness, so the NPC moves out of the player's line of sight). Another example of this kind of work is on real-time, coordinated planning (and re-planning) for the FPS game *F.E.A.R.* in which the human player forms squads to advance under cover and to suppress and flank [17, 18]. Using a blackboard system and in-game sensors, the system gathers real-time information and generates plans for individual NPCs. However, an individual NPC is able to decide to satisfy other goals rather than follow the plan (e.g., save itself rather than continue to provide covering fire for an ally) – and such decisions are incorporated into the real-time re-planning. Another system has been developed in which NPC teams take appropriate tactical actions in the FPS game *Killzone* [28]. The system uses position evaluation functions that use a number of characteristics as inputs (e.g., outside friendly line-of-fire, cover from primary threat, distance from friendly positions) to identify tactical locations. Based on this, the system is able to do such things as choose appropriate paths for tactical troop movement. For example, in order for ally teams to suppress the fire from the opponents when the player is on the move, during gameplay the allies move to a position that can be reached by the opponent under cover, fire at the opponents, and quickly take cover. Finally, work has been done to create systems capable of complex, coordinated behaviors in a combat game [24]. Prior to gameplay, neural networks are used to represent the action of each opponent in the game and a neuro-evolution framework is used as an action selector. Then an evolutionary algorithm is used to modify existing weights and to add neurons and connections; the fitness function is based on such things as damage received, time alive, and the like. The resulting system has, for example, monsters that are able to stay close to a bot and avoid its weapon while waiting for it to turn its back so they can attack.

There is quite a bit of work to do action selection based on more complex kinds of classifications. In the context of RTS games, for example, work has been done to decide army deployment strategies. In one system [22], before gameplay, the results of encounters between the player and opponent units are simulated for all combination of strategies in order to create a payoff matrix for interaction between different strategies (e.g., mass attack, spread attack, half base defense, hunting). During gameplay, the payoff matrix is used to select the appropriate ally deployment strategy and coordinate the necessary team activity (e.g., for a specific encounter, the sys-

tem may choose an ally "hunter" deployment at which point certain ally groups would follow a multi-step process of forming a single large group which then attacks the nearest enemy group). Another approach [3] involves a system for which game data is collected, indexed in terms of the desirability of various game states, and clustered in terms of similar game observations. The result is a game in which the best action is stored for each category (based on such things as phase of the game, material strength, commander safety, positions captured, economical strength, and unit count). During gameplay, the game indexes are used to observe current gameplay and select the relevant game strategy. Finally, there is some work on action selection for opponents based on classifying dynamic player formations for the *Age of Empires III* RTS game [34]. The system characterizes dynamic formations (in terms of, e.g., line distance, horizontal distance, vertical distance) and a Bayesian classification algorithm is trained to identify different kind of opponents so that the resulting system correlates different formations to different types of opponents. During gameplay, the classifier is used to identify the opponents and the appropriate formation is used.

3.4 Team-mate Adaptation & Learning

The focus is on automatic (as opposed to player-taught) real-time learning of the relevant "components" of the "performance element(s)" [21] and/or of the appropriate responses to those components. Specifically, this section includes work on real-time adaptation (e.g., system that do, for example, real-time monitoring and assigning/re-assigning of player/NPCs roles) and real-time learning of components that help improve the NPC team-mates' abilities to coordinate team-mate oriented behaviors or decision-making.

Relevant examples of work on team-mate adaptation and learning in games includes adapting in real-time to changes in player behaviors or styles, generating new tactics or scripts, and learning new kinds of coordinated movement to make games more interesting and variable.

In terms of adaptation, a system has been developed for a zombie FPS in which the ally behaviors adapt to real-time assessments about player state [32]. In the game, the player and 5 ally NPCs are supposed to attack and kill zombies; the ally NPCs learn when to help the player in terms of engaging the enemy, shooting them from afar, or healing the player based on the context. The system encodes all possible ally actions (e.g., healing, staying idle, crouching, shooting) as scripts – and a feed-forward neural network is used for the ally's action selector (the network is trained by the back-propagation algorithm with some hand-crafted data). During gameplay, allies choose from particular support actions, and then, at the end of an execution cycle, the weights of the neural network are updated in order to adapt the ally's action in the next encounter (e.g., if the player often dies in an encounter, the agents focus more on healing the player rather than shooting). Similar techniques have been used for action selection in the context of doubles tennis [31]. The system uses POMDP player abstractions (e.g baseliner, volleyer) and MDP action abstractions (e.g. move right, left, up, down), with weights assigned to each action. During real-time gameplay, actions are selected and the weights are updated based on a fitness function. Thus, the system is able to classify the player, assign the team-mate to the complementary role, and, significantly, change these classifications (and perform the appropriate actions) based on ongoing player-performance. Finally, work has been done to adapt opponent behaviors/responses based on player traits in an FPS game [9]. The player is modelled in terms of concrete traits (e.g., avoids cameras, avoids guards, moves silently, can use knife, can use blackjack) or more complex, abstract traits (e.g., stealthy, prefers close combat). These traits can then be updated, based on observing and measuring player performance.

Work has also been done to evolve ally tactics in real-time for the naval RTS game *Capture the Flag* [1]. For this system, all entities in the game world are represented in terms of an influence map with influence values representing the flag, enemy and friendly ships. During gameplay, an evolutionary algorithm monitors ally actions to evolve the influence values on the map, with fitness being measured in terms of winning score, enemy count, friendly count and time. For example, allies may initially not do well in capturing the flag, but over time will succeed in generating more complex and successful formations. Similarly, work has been done to develop winning policies for NPC teams in *Unreal Tournament* [25]. The system uses fixed, pre-defined bot behaviors and team strategies so that there is a set of movement patterns the allies can perform. During gameplay, the team uses reinforcement learning to adapt action selection of the allies in the team and, based on the success of the state-action pair, the probability of the action occurrence is updated.

Some work has been done to automatically generate more interesting games of *Pac-Man* using online learning techniques [38]. Prior to gameplay, a model of the player is trained with a Bayesian network using features (e.g., score, time played, grid-cell visits entropy of the player) of a model player. The result is a team of ghosts that evolve differently when playing against different players – and produce games that meet the criteria for "interesting" games defined by the researchers. Other work has been done to make the opponents in *Pac-Man* less predictable by having team performance evolve [37] During gameplay, an evolutionary algorithm is used to modify the weights of a neural network based on a fitness function that evaluates the performance of the team as a whole. For example, the ghosts learn to perform complex team movements to chase the *Pac-Man*. Another example involves *dynamic scripting* to generate appropriate spells against the player's team *Baldur's Gate* that use [27, 26, 29]. The system involves a rule base of weighted rules about actions the opponent. During gameplay, a fitness function (based on, e.g., average remaining health of all party members, average damage done to opposition, remaining health of a character) is applied after each encounter to update weights of rules and then new opponent scripts are dynamically generated. For example, in an encounter, the opponent adapts to the point that it is able to throw a high-level damaging area-effect spell in the middle of the enemy team during the first round of fight.

4. ANALYSIS & CRITIQUE

In the previous sections we have seen the kinds of work that have been done to date to develop team-mate AI.

It would be nice if it was possible to say something about the relationship between different real-time team-mate AI problems, techniques, and genre-specific issues – and even do some comparative analysis about, say, the use of different techniques for different problems. Unfortunately, the work may not yet be mature enough to do such analysis. In the case of real-time team-mate learning, for example, there does not seem to be enough work done to really compare different learning techniques (e.g., supervised, reinforcement, competitive) applied to similar problems.

Having said that, most of the relevant work seems to have been devoted to real-time action-selection and classification, with some devoted to prediction, and, in recent years, some attempts to use real-time learning in ways that meet real-time game requirements. This survey suggests that there are also some aspects of real-time team-mate AI where there seems to be little or no work: concept learning and clustering (e.g., using unsupervised learning techniques to generate and identify categories of players, situations,

and the like), inferencing, and communication. Given the real-time constraints, the lack of work on the first two is understandable, but they may be worth exploring. Real-time generation and understanding of different kinds of communication (e.g., natural language, facial expressions, gesture) will presumably be more and more important as NPCs develop in complexity, as input devices and recognition techniques become more sophisticated, and as more games start to include physical/robotic team-mates.

In terms of game genres, most of the work on team-mate AI to date seems to have been for combat NPCs, whether it is for action, RTS, or RPG games – although, of that work, most of it has been on action games. In terms of how things have changed, the earliest work on team AI was in the context of team-sport games. Most of the research focus today is on player modelling and prediction – especially in action games. The least amount of work on team-mate AI seems to have been for RPGs. Although some of the differences may be attributed to such things as the relative popularity of the genres (and resultant availability of game engines), some of the differences can be attributed to the relative need for team-mate AI (relative to other needs) in various genres – and to the difficulty of the different problems. Compare, for example, developing action-selection algorithms for FPS games with developing algorithms to classify player strategies in RTS games.

We now highlight several issues that emerge as a result of this survey: prioritizing the player, ally versus opponent teams, improving allies as well as opponents, centralized coordination, learning challenges, and perception and communication.

Prioritizing the Player In addition to supporting the player in game-play terms, one of the key functions of team-mate NPCs is to "prioritize" the human player in terms of engagement and participation [20, 28]. This paper has mostly highlighted work where the real-time team-mate AI is player-oriented. Nonetheless, in the process of researching this paper it was actually quite surprising to learn how often the player is simply ignored in games with teams of NPCs. For example, it is often the case that when attempting to coordinate a group, player preferences are neglected – and when planning for a mission, the fact that a human player is also involved may not be taken into consideration. When the player experience is actually prioritized, a couple of challenging issues arise. First, of course, is how such prioritization factors into action selection. Second, and less obvious, is that although teams with human players and teams with only NPCs can both use largely the same techniques, prioritization of the player does introduce one kind of asymmetry: teams with human player(s) have fewer/different options at their disposal in terms of task assignment, team-mate sacrifice, and the like – and NPC-only teams need to treat their human opponents in ways that differ from the ways that teams with humans can treat their opponents.

Ally teams versus opponent teams This paper has briefly indicated when the contributions relate to ally NPC teams or opponent NPC teams. In one sense it may be obvious that the design of ally NPC teams has to take the player into account, but actually, due to player prioritization, so does the design of opponent NPC teams. It will be an interesting area of future work to understand in more detail the similar and differing requirements for the two different kinds of NPC teams – and how these correlate with the different kinds of activities in different game genres.

Improving allies as well as opponents Unlike the case where a player plays against a single opponent (even in the case where the "opponent" is not represented as an agent), the moment a player has even a single ally agent, it becomes *team* play. The reason for highlighting this obvious point it that it may explain one reason that there is less work on team-mate AI: it is much easier to create a well-designed and interesting game with a single opponent than to do the same with a single ally. But although it may be tempting to make games more interesting and complex by simply increasing the number and kind of opponents, as players have more and more experience with multi-player games it seems likely that they will come to expect interesting and competent ally NPCs. In other words, it will be important to develop more sophisticated and varied algorithms for generating adaptive opponents that challenge players *as well as* for generating adaptive partners that support players.

Learning challenges for team-mate AI In general, there are very few improvements to game-agent behavior that cannot be generated prior to actual game-play. Because of this, online learning is rarely used in games (other than for exploring AI learning problems). Indeed, there are those who argue that because of the specific nature of game-play needs, and because of the performance penalty of real-time learning, there is actually no need to use online learning techniques for games at all. The issue of learning is a complicated problem for developing team-mate AI; here we simply highlight a few of the challenges. For one thing, NPC allies who learn from players face some particular challenges. For example, imitation-based learning is not always an effective option. For many kinds of coordinated activities, the best behavior of the NPC does not involve *duplicating* what the player is doing, but *complementing* it. (A related dilemma is that ally NPCs should not learn the wrong lessons from the players who are performing poorly.) Another issue that arises for games that make use of player modeling or profiling is the fact that user intentions and preferences change over time, so such models can become inaccurate if they are not updated; this is the problem of concept drift [4] and dealing effectively with it is a largely unsolved problem in team-mate AI. Finally, most existing learning methods take too much time for the kinds of fast adaptation and improvement players require in real-time games. Given all these (and other) issues, it is not surprising to note that current work leaves many of these problems largely untouched.

Perception and Communication With the exception of recent games that are starting to make use of sensor-based input (e.g., for gesture-based control) or special case of robotics-based games, for the most part, work in team-mate AI does not need to solve typical AI perception problems as such. And the bulk of AI-relevant advances in team-mate communication is actually in terms of the analysis and decision-making about *what* to communicate. Although the current state of the art in communication (e.g., synthesis or understanding of various kinds of communication) real-time team-mate games is not very complex, we can expect this to change as games are developed that, for example, support more complex written or spoken language in the communication between human and NPCs. There are a number of obvious opportunities for more sophisticated communication with/between characters in virtually every genre.

5. CONCLUSION

This survey reveals a number of areas in which team-mate AI has made progress (e.g., real-time prediction) – as well as areas where there seems to be little or no work (e.g., real-time inferencing).

There are at least two limitations of the current survey that are worth noting: the current definition of team-mate AI proposed here – and the existence of work in other game genres.

The definition of real-time team-mate AI proposed here is intended partly to begin the discussion about an appropriate way to delineate the topic. For example, this paper has taken the rather strong position of only considering player-related phenomena as those aspects of the game world which are unknown or unfore-

seen. However, there are other kinds of problems which could be reasonably included as part of real-time team-mate AI: handling unknowns which exist because, although the information is available in principle, it is not calculated or maintained in game implementations for performance reasons – or calculations that require real-time heuristics due to the computational complexity of the calculations.

There is quite a bit of relevant work on robotic soccer (e.g., *RoboCup*), but the issues are so specific the current status of hardware team-mates that including it is beyond the scope of this brief survey Also, there is work in other genres (e.g., team/partner-based simulation and ALife games, interactive stories [14]) that falls within the broad outlines of team-mate AI proposed here. However, the nature of such games is often quite different from the kinds of goal-oriented games for which team-mates are typically implemented, so extending the survey to cover such research is future work. Finally, it is not clear whether, for example, issues confronted in developing turn-based team-mate AI should be considered within the scope of "real time" efforts.

Having said that, the work sampled for this survey is probably a reasonable approximation of the total amount of work that has been done on the topic. It does seem as if there has been some interesting and significant progress in the development of real-time team-mate AI for games – but there is much left to do.

6. ACKNOWLEDGMENTS

This research is supported by the Singapore-MIT GAMBIT Game Lab research grant "Designing Adaptive Team-mates for Games"

7. REFERENCES

[1] P. Avery, S. Louis, and B. Avery. Evolving coordinated spatial tactics for autonomous entities using influence maps. In *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, pages 341–348, 2009.

[2] S. Bakkes, P. Spronck, and E. O. Postma. Best-response learning of team behaviour in Quake III. In *Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 13–18, 2005.

[3] S. Bakkes, P. Spronck, and J. van den Herik. Rapid and reliable adaptation of video game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, 1:93–104, 2009.

[4] M. Black and R. J. Hickey. Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis*, 3(6):453–474, 1999.

[5] B. Cowley, D. Charles, M. Black, and R. Hickey. Analyzing player behavior in Pacman using feature-driven decision theoretic predictive modeling. In *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, pages 170–177, 2009.

[6] J. Donkers and P. Spronck. Preference-based player modeling. In *AI Game Programing Wisdom 3*. Charles River Media, 2006.

[7] A. Drachen, A. Canossa, and G. N. Yannakakis. Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, pages 1–8, 2009.

[8] J. A. Glasser and L.-K. Soh. AI in computer games: From the player's goal to AI's role. Technical report, University of Nebraska, Nebraska Lincoln, USA, 2004.

[9] R. Houlette. Player modeling for adaptive games. In *AI Game Programing Wisdom 2*. Charles River Media, 2003.

[10] T. J. A. Jansen. *Player Adaptive Cooperative Artificial Intelligence for RTS Games*. Bsc. thesis, Universiteit Maastricht, 2007.

[11] J. E. Laird. Research in human-level ai using computer games. *Commun. ACM*, 45(1):32–35, 2002.

[12] F. D. Laramee. Dead reckoning in sports and strategy games. In *AI Game Programming Wisdom 2*. Charles River Media, 2004.

[13] K. Laviers, G. Sukthankar, M. Molineaux, and D. W. Aha. Improving offensive performance through opponent modeling. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, pages 58–63, 2009.

[14] M. Mateas and A. Stern. Facade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference*, 2003.

[15] I. Millington. *Artificial intelligence for games*. Morgan Kaufmann, 2006.

[16] F. Mommersteeg. Pattern recognition with sequential prediction. In *AI Game Programming Wisdom*. Charles River Media, 2002.

[17] J. Orkin. Applying goal-oriented action planning to games. In *AI Game Programing Wisdom 2*. Charles River Media, 2004.

[18] J. Orkin. Agent architecture considerations for real-time planning in games. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, pages 105–110, 2005.

[19] A. Ram, S. Ontañón, and M. Mehta. Artificial intelligence for adaptive computer games. In *Proceedings of the Florida Artificial Intelligence Research Society Conference*, pages 22–29, 2007.

[20] J. Reynolds. Team member AI in an FPS. In *AI Game Programing Wisdom 2*. Charles River Media, 2004.

[21] S. Russell and P. Norvig. *Artificial intelligence: A Modern Approach*. Prentice Hall, 2002.

[22] F. Sailer, M. Buro, and M. Lanctot. Adversarial planning through strategy simulation. In *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, pages 80–87, 2007.

[23] F. Schadd, S. Bakkes, and P. Spronck. Opponent modeling in real-time strategy games. In *Proceedings of the International Conference on Intelligent Games and Simulation*, pages 61–68, 2007.

[24] J. Schrum and R. Miikkulainen. Constructing complex NPC behavior via multi-objective neuroevolution. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, pages 108–113, 2008.

[25] M. Smith, S. Lee-Urban, and H. Munoz-Avila. RETALIATE: learning winning policies in first-person shooter games. In *Proceedings of the Innovative applications of artificial intelligence*, pages 1801–1806, 2007.

[26] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma. Adaptive game ai with dynamic scripting. *Machine Learning*, 63(3):217–248, 2006.

[27] P. Spronck, I. G. Sprinkhuizen-Kuyper, and E. O. Postma. On-line adaptation of game opponent AI with dynamic scripting. *Int. J. Intell. Games & Simulation*, 3(1):45–53, 2004.

[28] R. Straatman and A. Beij. Killzone's AI: dynamic procedural combat tactics. In *Game Developers Conference*, 2005.

[29] I. Szita, M. Ponsen, and P. Spronck. Effective and diverse adaptive game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, 1:16–27, 2009.

[30] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7(1):83–124, 1997.

[31] C. Tan, , and H. L. Cheng. IMPLANT: an integrated MDP and POMDP learning AgeNT for adaptive games. In *Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 80–87, 1996.

[32] C. T. Tan and H.-L. Cheng. Personality-based adaptation for teamwork in game agents. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, pages 37–42, 2007.

[33] C. Thurau, C. Bauckhage, and G. Sagerer. Combining self organizing maps and multilayer perceptrons to learn bot-behaviour for a commercial game. In *GAME-ON*, pages 119–123, 2003.

[34] M. van der Heijden, S. Bakkes, and P. Spronck. Dynamic formations in real-time strategy games. In *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, pages 47–54, 2008.

[35] W. van der Sterren. Being a better buddy: Interpreting the player's behavior. In *AI Game Programing Wisdom 3*. Charles River Media, 2006.

[36] T. Wellmann. Building a Sports AI Architecture. In *AI Game Programming Wisdom 2*. Charles River Media, 2004.

[37] M. Wittkamp, L. Barone, and P. Hingston. Using NEAT for continuous adaptation and teamwork formation in Pacman. In *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, pages 234–242, 2008.

[38] G. N. Yannakakis and J. Hallam. Evolving opponents for interesting interactive computer games. In *Proceedings of the 8th international conference on simulation of adaptive behaviour (SAB04)*, pages 499–508, 2004.

[39] S. Yildirim and S. B. Stene. A survey on the need and use of AI in game agents. In *Proceedings of the Spring simulation multiconference*, pages 124–131, 2008.

[40] R. Zhao and D. Szafron. Learning character behaviors using agent modeling in games. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, pages 179–185, 2009.