Team Description of Helli-Mersad 2005

Ahmad Boorghany Farahany, Sasan Haraji, Darioush Jalali, Mostafa Rokooey, Mohammad Salehe, Meisam Vosoughpour

Allameh Helli High School,
National Organization for Development of Exceptional Talents
http://www.allamehelli.ir

Abstract. This paper is aimed to present a brief overview of some techniques used in Helli-Mersad 2005 2D Soccer Simulation team. First, an architecture is presented for decision-making system of agents and then a system for generating and evaluating candidate passes together with a rule-based system to improve inter-agent coordination are introduced and characterized.

1 Introduction

Our team is consisted of 6 students of Allameh Helli High School. Several teams from our high school have participated previous RoboCup competitions and proved to be powerful teams in the competitions. Our previous experiences of working with these teams and also having an online soccer coach team led us to design and implementation of a team based on techniques of various fields in Multi-Agent systems and AI, which finished last year competitions ranking *3rd*.

In this paper, first a flexible deliberative architecture for decision-making system called *Hierrarchical Planning System* is introduced, and then a brief overview of methods used for designing a powerful passing system and implementing some aspects of coordination between agents is presented.

2 Hierrarchical Planning System

In order to acheive an efficient Multi-Layer decision system, a special *Hierrarchical Planning System*(HPS) has been designed and implemented. The basic component of such a system is a plan, which is defined as a policy/schedule in order to reach a predefined goal.

In the highest level of HPS, plans are in the form of *Strategies*, which are complete policies/schedules for managing the game towards the expected results. At each time instance, only one strategy can be active and in use by the agent. The strategy of an agent can only be changed with an advise received from the coach.

Plans that are not in the form of a strategy are *Sub-Plans* aiming to acheive some parts of the original plan. Each sub-plan, having a model of the environment at each time instance, gives a self-evaluation of how much successful will it

be in acheiving its goal. Also, a boolean flag indicates whether it has successfully reached its goal or not. Generally, plans, whether in the form of strategies or sub-plans, have common properties. Each plan contains an initially empty stack of child sub-plans. In addition, there are also two decision-making modules called *Urgent Module* and *Main Module*.

A plan, at execution time, first of all, executes its *Urgent Module* and then starts popping from the sub-plan stack until an incomplete sub-plan with a reasonably high success rate is found, and then executes that sub-plan. If no such sub-plan is found then the main block is executed which is responsible for choosing sub-plans and modifying contents of the stack, or in the case of plans located on the lowest level, decides on the atomic actions to perform. Having a sub-plan stack gives the agent the ability to suspend its currently executing sub-plan in the case of the need to execute an urgent low-duration sub-plan, by pushing the new plan into the stack and resume it again after finishing the urgent plan.

A Plan, in its simplest form, may be designed in the form of a *Scenario* where all child sub-plans are selected and executed statically, in a predefined manner. And in its most complex form, it may consist of a list several plans generated and evaluated in realtime in order to find the best sub-plan to execute. In this system, the evaluation made by the parent plan is different in nature from the self-evaluation made by the child plan. In other words, while the child plan evaluates the degree to which it is close to its defined goal, the parent plan evaluates the utility/fitness of the child plan with respect to its own goal.

This system proves to be successful at acheiving a high-level of modularity and deliberation while maintaining flexibility of the decision-making system.

3 Passing System

Passing is one of the fundamental skills which plays a significant role towards acheiving efficient and reliable teamwork. Major problems towards designing a high-performance passing system are *pass generation* and *pass evaluation*. The former deals with generation of a set of candidate passes to be evaluated, while the later deals with predicting how much effective a certain pass will be.

One conventional method used in pass generation systems is considering passes to all or a subset of teammates as the candidate passes to be evaluated. Unfortunately, this method, while straight-forward, is hard to implement efficiently, as an additional search is needed to find suitable pass directions to each teammate. In addition, this method reduces the modularity of a general passing system by adding an extra evaluation phase in choosing suitable pass directions to each player.

As a result, we used a different method for pass generation. In other words, instead of modelling passes to each player, a set of kicks is generated in various directions, and passed to the evaluation sub-system which has the responsibility to find the fastest player to intercept the kicked ball and evaluate the resulting

situation. The choice of direction and power of each kick depends on the type of pass to be modelled. Generally, four types of passing exist:

Offensive Pass: Passes given with the aim of transfering the ball to a player closer to the opponent's goal.
Offside Breaker: Passes given with the aim of breaking the offside line.
Secure Pass: Passes given with the aim of preventing the ball to be lost.
Cross Pass: Passes often given from the sides of the penalty area with the aim of putting a teammate into a direct shoot-to-goal situation.

Different direction generation policies are used for each type of pass, each generating multiple directions in small sectors while completely ignoring several sectors around the passer.

3.1 Virtual Passing System

The main disadvantage of the passing system described above is its being expensive with respect to computation time. Inspecting the system more slightly, it is noted that in this system, pass generation and evaluation is only done by the player that owns the ball, although certain players may have a better view of some parts of the scene and can therefore generate and evaluate a set of passes from the ball owner more precisely than the ball owner itself.

The system designed for taking advantage of such abilities of these players is called a *Virtual Passing System* as the player, generating and evaluating a pass, does not actually use it. In this system, a player who is not the ball owner decides to generate virtual passes based on the degree to which the information about the ball owner is up to date in its world-model. There, if it finds a reasonably good pass it audits a triple consisting of direction, power, and evaluated value of the pass.

The ball owner, receiving a pass information triple, may accept it if its value is more than the value of the best pass among its own generated passes.

4 Examples of Inter-Agent Coordination

One of the most important aspects of problems involving multi-agent systems having limited inter-agent connection, e.g. *RoboCup Soccer Simulation* is coordination between agents. Examples of problems regarding coordination is choice of the player to catch the ball in a specific situation, and also choice of the player to block an opponent player. This problems becomes harder if we take into account the limited and unprecise information each player has about the environment.

In order to solve this problem, each player uses a rule-based expert system to determine the player who should intercept the ball. Each player, in each cycle audits the ball position and number of the teammate who should intercept the ball due to the information in its own world-model.

When a player hears an announcement of the interceptor, he accepts it if the following conditions are satisfied:

1. if the announcement is from the interceptor with respect to its world-model it is accepted.
2. if the information of the current interceptor, and the information of the new player suggested for becoming the interceptor is also too old in its world-model the new information is accepted.

There are also some rules by which an agent decides to set its interceptor to null. These rules are as follows:

1. if the interceptor is itself but it is not the fastest player to catch the ball, the interceptor is set to null.
2. if another teammate is the interceptor but its information precisely determines that it is not the fastest player to reach the ball, the interceptor is set to null.
3. if the information about the ball is old, the interceptor is set to null.

And in the case the interceptor is set to null, there is a set of extra rules to determine another interceptor:

1. if the difference of the catch time of the first and second fastest player regarding to the world-model is high, the first fastest player is chosen as the interceptor.
2. if several players have approximately equal chance to reach the ball, the tie is broken based on team formation.

Similar approach is also used to determine the player who should block a specific oponent player. This approach has been used in *Mersad 2004*[1] and proved to be successful. This year, in order to improve the performance of the approach and decrease its crispness, the rule base is being transformed into a fuzzy one as most conditions are qualitative in nature.

References

1. Booghany et. al.: Mersad 2004 Team Description. RoboCup 2004 Soccer Simulation League. (2004)