Multi-agent Learning Seminar
An Architecture for Action Selection in Robotic Soccer

Daniel Silva, Nguyen Cu and Joseph Chege

June 23, 2012

1 Introduction

In [Stone and McAllester, 2001], the authors describe key elements in **ATT-CMUnited-2000** strategy. Namely:

1. Option evaluation architecture: all actions (passing, dribbling, kicking, etc) can be scored and compared among each other;
2. Leading passes (LP): pass the ball at an oblique angle of the intended receiver, not passing only directly;
3. Efficient numerical algorithm: leading passes incurs much more calculations, which need to be done as fast as possible;
4. Force fields (FF): to control the position of players without the ball.

The goal of this work is to incorporate *Leading passes* and *Force fields* in WrightEagle's RoboCup client [Bai et al., 2010] and measure the influence on its performance.

2 Implementation

Like in [Stone and McAllester, 2001], WrightEagle uses an *Option evaluation architecture*, different actions in nature are planned, scored and compared against each other, the highest score determines what the player will do. Options are called *Behaviors*. There are four main behaviors:

- *BehaviorPenaltyPlanner*
- *BehaviorSetplayPlanner*
- *BehaviorAttackPlanner*
- *BehaviorDefensePlanner*

Each behavior may be composed of other behaviors. *BehaviorAttackPlanner* is compromised of *BehaviorInterceptPlanner*, *BehaviorShootPlanner*, *BehaviorPassPlanner*, *BehaviorDribblePlanner*, *BehaviorPositionPlanner* and *BehaviorHoldPlanner*. *BehaviorDefensePlanner* on the other hand, is made of *BehaviorFormationPlanner*, *BehaviorBlockPlanner* and *BehaviorMarkPlanner*.

2.1 Leading passes

WrightEagle only consider direct passes, that is, passing directly to your teammate. We altered *BehaviorPass.cpp* to take into account nearby opponents, and slightly change the angle of the pass, by adding the following:

```
int position_predicted_scale=2;
Vector root_target = pass.mTarget;
Vector teamate_predict_pos =
  mWorldState.GetTeammate(tm2ball[i]).GetPredictedPos(position_predicted_scale);
Vector rel_opp_pos = mWorldState.GetOpponent(opp2tm[index_closed_opponent]).GetPos();
Vector V1= rel_opp_pos-root_target;
Vector V2 = teamate_predict_pos -root_target;
double reduce_rate = V2.Mod() /V1.Mod();
reduce_rate = pow(reduce_rate,2);
pass.mTarget= root_target+ (teamate_predict_pos - rel_opp_pos)*reduce_rate;
```

Figure 1 illustrates what is happening.

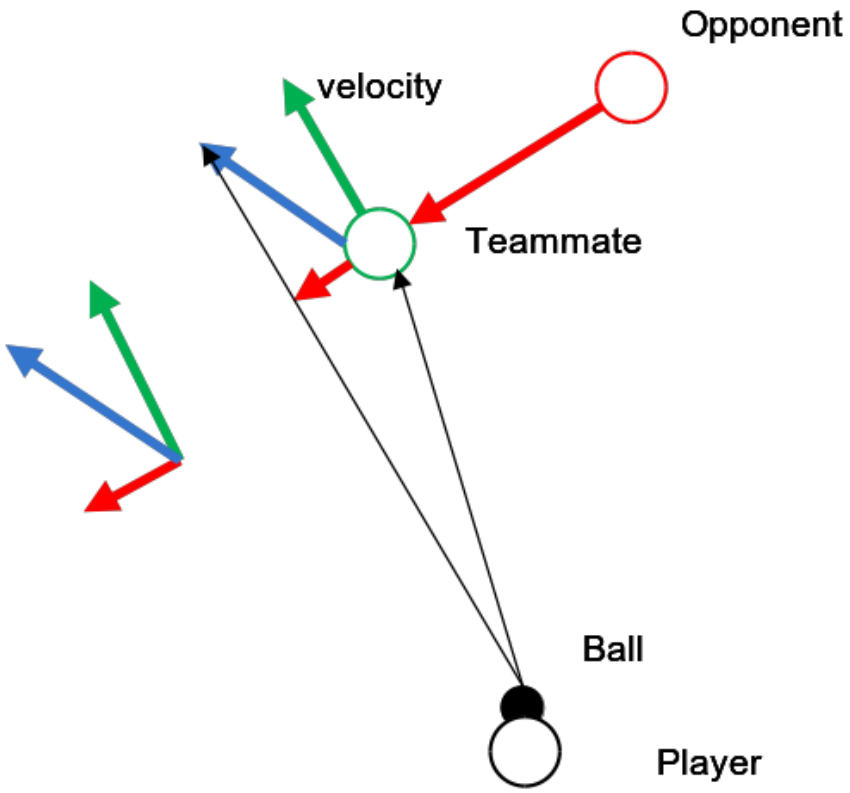

Figure 1: Passing with an opponent nearby.

The closest opponent influences pass direction, but only slightly, teammate positioning is still the largest factor and its velocity is important also. Keep in mind that, teammate has a reason to move with current velocity, because of formation tactic and positioning strategy, therefore passing should be done to the expected position of the teammate. Besides, influence by closest opponent is reversed by distance to teammate, that means if the opponent is far enough passing position is effected only by the position and velocity of the teammate.

2.2 Force fields

As described in [Stone and McAllester, 2001] force fields are divided into two categories:

- Strategic: applies to players far from the ball;
  - B: bounds-repellent, strong when players are within 5 meters of the edge of the field;
  - O: offside-repellent, causes player to retreat if the opponent's offside line is within 5 meters;

- S: inter-player force, attractive when players are slightly more than 20 meters apart and repulsive if they are closer.

- Tactical: applies to players near the ball;
  - T: repulsive force between offensive players, strong when the ball is within 8 meters;
  - C: repulsive force between players and opponents, also called get clear, its strength is proportional to the likehood that the pass will be intercepted by a defender.

We altered *BehaviorPosition.cpp* and *BehaviorFormation.cpp* by adding functions: *softIf*, *softEq*, *dist*, *forceFieldB*, *forceFieldO*, *forceFieldS*, *forceFieldT* and *forceFieldC*.

3 Results

We compare results using the following attributes: score, ball possession and successful passes. Score is simply number of goals from each side in the match, ball possession and successful passes are much harder to compute. In this case a team has possession of the ball if a member of the team is the closest player to the ball, and the closest opponent to that player is at least 1 meter away. A successful pass, consists in executing a pass and having possession of the ball at least 10 time units afterwards (otherwise you are dribbling) and at most 15 time units (otherwise you recovered a ball that was intercepted).

To gather the required information we altered *BehaviorPenaltyPlanner*, since it is the first behavior every player plans, by adding the following:

```
Vector self=mSelfState.GetPos();
Vector opponent=mWorldState.GetOpponent(mPositionInfo.GetClosestOpponentToBall()).GetPos();
Vector ball=mBallState.GetPos();
double diBallSelf=sqrt(pow(ball.X()-self.X(), 2.0)+pow(ball.Y()-self.Y(), 2.0));
double diBallOpponent=sqrt(pow(ball.X()-opponent.X(), 2.0)+pow(ball.Y()-opponent.Y(), 2.0));
double diSelfOpponent=sqrt(pow(self.X()-opponent.X(), 2.0)+pow(self.Y()-opponent.Y(), 2.0));
std::cout<<"dgk possession "<<mWorldState.CurrentTime()<<" "<<mSelfState.GetUnum()
  <<" "<<diBallSelf<<" "<<diBallOpponent<<" "<<diSelfOpponent<<std::endl;
```

A total of 30 games were played. 10 control games where two WrightEagle clients played against each other without any of our modifications, 10 with LP only and 10 with LP and FF control. Ball possession and successful passes are calculated for the team with the result on the left, which is also the team that contains our modifications in Tables 2 and 3. Results are displayed below:

Game	Score	Ball possession	Successful passes
1	4 - 5	41%	71%
2	3 - 1	47%	77%
3	1 - 0	41%	73%
4	7 - 5	46%	78%
5	5 - 6	41%	77%
6	3 - 5	43%	67%
7	4 - 2	42%	79%
8	5 - 6	41%	70%
9	5 - 4	42%	79%
10	1 - 4	38%	71%

Table 1: Unmodified clients results.

Game	Score	Ball possession	Successful passes
1	1 - 3	40%	70%
2	3 - 2	38%	70%
3	3 - 7	37%	69%
4	6 - 3	41%	78%
5	7 - 6	42%	72%
6	8 - 5	36%	70%
7	0 - 1	36%	66%
8	3 - 5	39%	75%
9	7 - 3	39%	75%
10	5 - 8	36%	79%

Table 2: Leading passes only results.

Game	Score	Ball possession	Successful passes
1	1-5	44%	73%
2	0-2	41%	74%
3	2-4	40%	70%
4	0-2	40%	73%
5	4-4	43%	71%
6	1-1	42%	77%
7	4-3	41%	71%
8	1-3	39%	70%
9	1-4	42%	65%
10	2-2	47%	74%

Table 3: Leading passes and force fields control results.

Applying the paired *t-test* yields the following *p-values*:

Feature	Control vs. LP	Control vs. LP and FF
Score	1.00	0.08
Ball possession	0.00	0.79
Successful passes	0.36	0.18

Table 4: *p-values*.

Only using LP didn't change the number of wins or the number of successful passes, but it changed ball possession, unfortunately for the worse. When using FF the null hypothesis can be rejected regarding score, but performance is consistently worse, almost all games were lost.

4 Conclusion

The modifications suggested in the article are too shallow for WrightEagle. Passing and positioning are optimized, given its complex architecture, that allows for comparison of different actions in nature, and several possible behaviors, for instance *BehaviorSetplayPlanner* which is probably responsible for the majority of goals. During the testing phase, we noticed that the majority of goals happen when a pass is made from the corner of the opponent's goal, to a teammate in the center, that just kicks the ball forward, FF doesn't help at all, since field B and O are probably pushing the receiver back to its team field.

References

A. Bai, J. Wang, G. Lu, Y. Wang, H. Zhang, Y. Zhu, K. Shi, and X. Chen. Wrighteagle 2d soccer simulation team description 2010. In *RoboCup International Symposium*, 2010.

P. Stone and D. McAllester. An architecture for action selection in robotic soccer. In *Proceedings of the fifth international conference on Autonomous agents*, pages 316–323. ACM, 2001.