Skills, tactics and plays for decentralized multi-robot control in adversarial environments

Lotte de Koning, Juan Pablo Mendoza, Manuela Veloso and René van de Molengraft*
c.k.m.d.koning@student.tue.nl, jpmendoza@ri.cmu.edu, veloso@cmu.edu, m.j.g.v.d.molengraft@tue.nl
Robotics Institute and Computer Science Department, Carnegie Mellon Univeristy, Pittsburgh, USA Control Systems Technology, Eindhoven Univeristy of Technology, Eindhoven, Netherlands

ABSTRACT

This work presents a pioneering collaboration between two robot soccer teams from different RoboCup leagues, the Small Size League (SSL) and the Middle Size League (MSL). In the SSL, research is focused on fast-paced and advanced team play for a centrally-controlled multi-robot team. MSL, on the other hand, focuses on controlling a distributed multi-robot team. The goal of cooperation between these two leagues is to apply teamwork techniques from the SSL, which have been researched and improved for years, in the MSL. In particular, the Skills Tactics and Plays (STP) team coordination architecture, developed for centralized multi-robot team, is studied and integrated into the distributed team in order to improve the level of team play. The STP architecture enables more sophisticated team play in the MSL team by providing a framework for team strategy adaptation as a function of the state of the game. Voting-based approaches are proposed to overcome the challenge of adapting the STP architecture to a distributed system. Empirical evaluation of STP in the MSL team shows a significant improvement in offensive game play when distinguishing several offensive game states and applying appropriate offensive plays.

Keywords

Robot soccer, Multi-robot system, Distributed coordination, Team plan execution

1. INTRODUCTION

RoboCup is an international robot soccer organization, founded to promote research in robotics and artificial intelligence. The ultimate goal of RoboCup is to beat, by 2050, the winner of the most recent human soccer World Cup with a team of fully autonomous humanoid robot soccer players, complying with the official rules of FIFA. To accomplish this goal, teams compete annually in various soccer leagues, each of which focuses on the different challenges of robot soccer. This work is related to two of those soccer competitions: Small Size League[1] (SSL) and Middle Size League[2] (MSL) (Figure 1). SSL is specialized in fast-paced and advanced team play within a central coordinated multi-robot team. In MSL on the other hand, research focuses on coordination of a decentralized multi-robot team. Both teams operate in adversarial environments. In order to reach the RoboCup goal, research and accomplishments of the different leagues should be brought together. This work presents a co-operation between the SSL team CMDragons from the Carnegie Mellon University in Pittsburgh (United States) and the MSL team Tech United from Eindhoven University of Technology (the Netherlands).

Much research [9][14][3][8] has been done on team planning of centrally coordinated multi-robot teams. However, when the number of robots in the system increases, computations exponentially increase in complexity. Distributed multi-robot systems are a solution to this computation problem [13], as each individual robot computes its tasks. As high-level team planning is the same for both centrally coordinated and distributed multi-robot systems, the research on team planning in centrally coordinated systems can be beneficially used in distributed systems. This work presents an integration of a planning algorithm designed for a centrally coordinated multi-robot team into a distributed multi-robot team.

The case study used for this work is the RoboCup environment. Towards the RoboCup goal of beating a human soccer team, the number of robots in the team will increase to 11, therefore a distributed solution is desired [13]. The advanced planning strategies developed by the SSL over the past years [1][2][6], can be used in the distributed teams, such as the MSL. For this work, the Skills, Tactics and Plays (STP) planning algorithm [2], developed by the CMDragons team for small size soccer robots, is integrated into the Tech United middle size robots to increase the level of team play in Tech United attacks. Compared to planning algorithms designed for multi-robot teams [9][14][3][8], the STP architecture is specifically designed to control an autonomous multi-robot team in a dynamic environment with the presence of adversary's. The architecture contains predefined team plans, the plays, which are chosen during game play based on the world state. The tactics and skills associated with a play define the single robot behaviour during the execution of a play.

This paper presents the approach and challenges of integrating an architecture developed for a centralized multi-robot team into a decentralized multi-robot team. In particular, a distributed team of robots, with differences in their estimated state of the world, needs to agree on a team plan and role assignments in a fair way. To overcome this problem, the robots use a voting system to determine which play to use, as well as their optimal role assignment. The new

*Tech United Eindhoven (`www.techunited.nl`) and CMDragons (`http://www.cs.cmu.edu/~robosoccer/small/`)

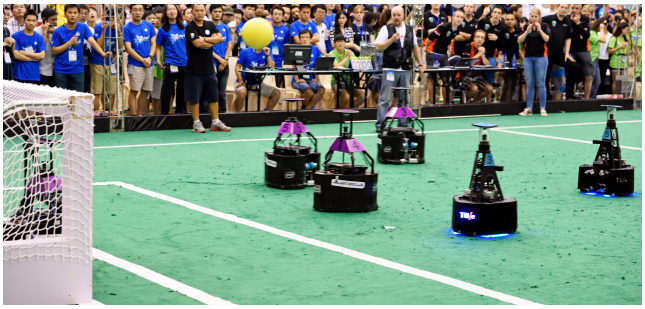
[1]`http://wiki.robocup.org/wiki/Small_Size_League`

[2]`http://wiki.robocup.org/wiki/Middle_Size_League`

algorithm in the Tech United MSL team is evaluated with simulations and show that using the SSL approach can significantly improve the level of team play.

The remainder of the paper is organized as follows: Section 2 provides a short introduction in the SSL and MSL; Section 3 presents the current Tech United strategy and the components which play a role in the STP integration; Section 4 explains the Skill, Tactics and Play (STP) architecture designed by the CMDragrons team; Section 5 describes the integration of STP in the Tech United software and finally; Section 6 gives the simulation results which show the improvement of the Tech United's offense; the paper is concluded in Section 7.

(a) Small Size League at RoboCup European Open 2016, Eindhoven, the Netherlands



(b) Middle Size League at RoboCup 2015, Hefei, China

Figure 1: Both competitions at RoboCup tournaments

2. SSL VERSUS MSL

In the Small Size League, robots with a maximum diameter of 180 mm and a maximum height of 150 mm, compete in teams of six robots on a green carpeted field of size 9 m × 4 m. Cameras overhead capture images of the field which are processed by a common vision module, SSL-Vision [15]. During the game, the robots are commanded by the team's centralized controller using radio. SSL is a high speed competition which focuses on fast-pace multi-robot coordination. Robots move at speeds up to 3.5 m/s and the ball is shot with a velocity up to 8 m/s.

CMDragons, from the Robotics Institute at Carnegie Mellon University Pittsburgh, is the top ranked SSL team in RoboCup and has competed since the beginning of RoboCup in 1997. Along those years, the CMDragons reached the final match nine times and won the competition five times [12].

In the MSL the main challenge is the control of a decentralized multi-robot team. The robots in MSL are at most 80 cm high and 50 cm wide and play on a green carpeted field of 12 m × 18 m. Each robot is equipped with sensors and a computer. For the Tech United team, one of these sensors is a catadioptric vision system with a range of approximately 7 m. Since robots are not able to perceive the whole field with their own vision system, world state information is shared among robots using WiFi in order to gain as much world state information as available within the team. Section 3 discusses the solution of this limited vision in MSL in more detail.

At the moment of writing, the current champion of the Middle Size League competition is Tech United from the Eindhoven University of Technology. Tech United participates since 2006, reached the final match every year since 2008 and won in 2012, 2014 and 2016 [11].

3. TECH UNITED MSL STRATEGY

For this study STP is used to integrate offensive plays into the Tech United team. The current Tech United offensive strategy assigns roles to the active robots, which include specific tasks. The Goalkeeper defends the goal, two Defenders position between the ball and their own goal, and two attackers execute an attack on adversary's half. Two attacker roles are distinguished: the AttackerMain is the role that manipulates the ball and the AttackerAssist positions on the adversary's half to potentially receive the ball. Roles execute tasks which involve decision making and selecting skills to complete the task.

The Tech United team currently applies a single offensive strategy used in all offensive game states. One of the drawbacks of having only one offensive strategy is that playing with only two attackers, while the opponent mostly defends with four robots, makes it hard to create goal attempts. For example, during the final match at RoboCup 2016, from the 13 goal attempts only two where created during regular gameplay. All other goal attempts were results of stopped-ball situations, such as a throw-in or free-kick. Another drawback of having only one offensive strategy, is that the team cannot adapt the strategy to the game state. The STP architecture is designed to overcome these drawbacks.

Before the detailed explanation of the integration of the STP architecture in the Tech United software, first the current communication and role assignment is explained in this section. Both components play a role in the STP integration.

Communication.

Communication is an essential part to control a distributed multi-robot team. In the Tech United team, robots broadcast data at a frequency of 40 Hz. This data can be roughly distinguished in two components: i) world state information and ii) data to make proper team play feasible.

Each individual robot contains sensors to perceive its near environment. These sensors are limited to a vision range of approximately 7 m. Since the field is of size 18 m × 12 m, a single robot is not able to perceive the whole field by itself. Therefore, each robot shares a local environment description with its peers, which contains obstacles and ball candidates. All of this shared information is merged in the world model of each robot. Therefore, robots are able to create a model of the world state with all available information. However, it may still be the case that the combined perception of the team does not cover the whole field.

The second part of communication relates to strategy. In-

game decisions are based on the world model as well as the state of peer robots. Therefore, parts of the robots' internal states are also communicated; this is a substantial difference between MSL and SSL, in which such limited communication is not necessary. To illustrate this difference, consider this example where robot X has decided to pass the ball to robot Y. In a centrally controlled team, the coordinator contains the state information of each robot at the same timestep. Therefore, when the decision for a pass is made by the coordinator, each robot transits to the next state at the same timestep. In a distributed robot team, such decisions are made based on handshakes between robots. The pass receiving robot, robot Y, only transits to the pass receiving state at the moment robot X is in pass giving state. Robot X transits to the pass giving state as soon as it is in ball possession. At the moment robot X is ready to give a pass, the robot only passes the ball when the receiver is in pass receiving state. The states are being communicated at an update rate of 40 Hz; however, due to packet losses, the effective rate may be lower.

Role assignment.

In the Tech United software two main game states are distinguished: refbox tasks and regular game play. Stopped-ball situations such as a free kick, throw in or goal kick are examples of refbox tasks. During refbox tasks or game play a role is assigned to the robot, where each role is assigned only once. These roles define the task for the robot to perform. During either refbox tasks or game play, each robot is assigned to a role and no role occurs twice. Game roles are set as default for each robot at the beginning of a game. However, role switches may happen when a specific situation occurs, –e.g., when the AttackerAssist is closer to the ball than the AttackerMain, the AttackerAssist requests a role switch with the AttackerMain such that he can intercept the ball. No further role assignment is needed for game roles.

For the refbox roles on the other hand, role assignment is done each time a new refbox situation occurs. When integrating STP into the Tech United software, the same role assignment is used to divide play roles optimally. The role assignment is based on the distance of the current robot position $x(\rho_i)$ to the fixed role position of the selected refbox task $x(r_j)$. Depending on the number of active robots N, N roles are assigned based on priority $R = \{r_1, r_2, ..., r_N\}$. Each robot computes a cost matrix C with the distances from each robot ρ_i to role r_j: $C : \mathcal{P} \times R \rightarrow \mathbb{R}$. Then, computations are done to determine the optimal role assignment where the total to be travelled distance by the robots from current position to role position is minimal: $\arg\min_C(\sum \rho_i \in \mathcal{P}[C^i])$.

4. SKILL TACTICS PLAY ALGORITHM

The Skills, Tactics and Plays (STP) architecture was designed by CMDragons to coordinate a centralized team of autonomous robots to achieve long-term goals [2]. This architecture enables simple specification of team plans as plays, which are then executed individually by each robot through tactics and skills. STP enables the team to adapt their team strategy as a function of the state of the game, resulting in highly versatile teamwork. This section describes each component of STP and how the CMDragons team uses STP for team planning.

4.1 STP Components

Skills.

At the most basic level, each robot is capable of performing a certain set of low-level skills. Skills are base behaviors that can be parameterized to achieve various goals. In the domain of robot soccer, these skills include navigating safely and quickly to a specific target location, shooting the ball with a specified velocity, and intercepting a moving ball, among others. Skills alone do not encode goal-oriented behavior, but are used in a set of skills to achieve goals.

Tactics.

Tactics are goal-oriented behaviors that each individual robot performs to carry out a team plan. These behaviors may be composed of skills, organized to work cohesively towards achieving a goal. In the CMDragons team, tactics are implemented as finite state machines that control the flow of the different skills that make up a tactic.

Each role has a specific set of tactics to perform. Some of the most prominent tactics in the CMDragons team include the Goalkeeper to block shots from the adversary, Defenders to prevent the adversaries from passing and shooting, the AttackerMain to manipulate the ball to score goals on the adversary, and AttackerAssists to position themselves to receive a pass from the AttackerMain. Tactics, like skills, may be parametrized. For example, the AttackerAssist may restrict their search for a good pass location to a specific region of the field.

Plays.

In STP, team strategy is encoded into a playbook, made up of a set of plays. A play is a team plan specified as a list of roles that the team of robots must fulfill. Each role, within the context of STP, is defined as a sequence of tactics to be completed sequentially. Additionally, plays specify applicability conditions: *preconditions* specify the set of game states under which a play should be considered for selection, while *invariants* specify the set of game states under which the team does not need to abort this play and select a new one.

Table 1 shows a slightly simplified example of a play in the CMDragons playbook. This play is applicable when all preconditions match the world state: the ball is in the adversary's half of the field (their_side), and there are no opponents on our half of the field (!opp_our_side), and our team has possession of the ball (offense). Furthermore, if this play is selected for execution, it will remain in execution until either the invariants become false, –i.e., the ball moves to our side of the field (their_side becomes false) or the opponent team gains possession of the ball (!defense becomes false), or all the robots have finished performing their role.

4.2 STP Procedure

During each step of execution, the team selects a joint plan, optimally assigns roles to each robot, and then each robot individually executes its role. This section describes this procedure, with an emphasis on the team components.

Play Selection.

First, the team's central intelligence decides whether the play that was last used is still applicable. That is, it checks

Table 1: An example of a play

```
PRECOND their_side && !opp_our_side && offense
INVARIANT their_side && !defense
NUMROBOTS 1 6

ROLE 0
    Goalkeeper
ROLE 1
    AttackerMain
ROLE 2
    Defender
ROLE 3
    AttackerAssist front_center
ROLE 4
    AttackerAssist front_left
ROLE 5
    AttackerAssist front_right
```

if the last play's invariants are still true, and if at least one of the robots has not yet finished its role. If these are both true, no new play is selected; if it is false, a new play is selected from the set of plays whose preconditions hold. If there are multiple plays whose preconditions hold, a play among the set is chosen randomly, with potentially different probabilities for each play.

Optimal Role Assignment.

Once a play has been selected, each of the roles in the play is assigned to a robot in the team. Given N active robots $\mathcal{P} = \{\rho_0, \rho_1, \ldots, \rho_N\}$, STP optimally assigns the first N roles $R = \{r_0, r_1, \ldots, r_N\}$ to them optimally, based on a cost function $C : \mathcal{P} \times R \to \mathbb{R}$. In the CMDragons, this cost function $C(\rho_i, r_j)$ is an estimate of the time that it would take robot ρ_i to complete role r_j. To do this, a cost matrix $\mathcal{C}$ of size $N \times N$ is created, such that the entry at row i and column j corresponds to the computed cost $C(\rho_i, r_j)$. From this cost matrix, algorithms such as the Hungarian algorithm [10] can be used to find the assignment from roles to robots that minimizes the total cost. It is important to note that the costs in matrix $\mathcal{C}$ are computed from the centrally-estimated state of the world, common to all the robots in the team.

The optimal role assignment computation used by the CMDragons is similar to the approach of Tech United, the only difference are the costs. Where the SSL team uses time, the MSL team uses distances. Assuming constant speed during repositioning we can consider time and distance to be equivalent.

Individual Role Execution.

After STP distributes the roles of the selected play to each of the robots, they proceed to individually execute their assigned tactic. In SSL, this step also occurs in a centralized controller, with only the final motion command being sent to each individual physical robot. However, the computation of each tactic happens in parallel, with only very limited communication between robots at this stage.

4.3 STP in distributed team

The bulk of the problem of adapting the STP architecture to the MSL lies in the previous two steps: play selection and optimal role assignment. In SSL both steps are performed by a central controller, in MSL however, each robot computes strategy individually. Therefore, the play selection and role assignment is done in five separate computations. To execute a team plan towards a common goal, it is important for the team to agree on the selected play and role assignment. Section 5 discusses in detail the integration of this joint play selection and role assignment in MSL.

5. INTEGRATION OF STP IN MSL

The integration of STP involves four main components from the Tech United software: i) communication ii) the game state evaluation, iii) the role assignment and iv) the tactics execution. Figure 2 is an overview of the strategy components needed for the STP integration. Highlighed are the new designed components, the playbooks with plays, the play selection algorithm and the tactics, and the reused and extended components, communication, game state evaluation and role assignment. This section explains each component in more detail.

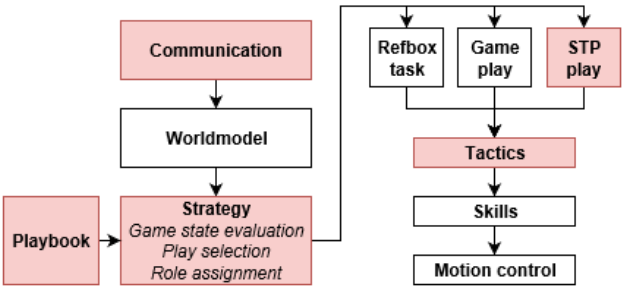


Figure 2: Three main components for STP: playbook, strategy, play execution.

5.1 Overview STP algorithm in MSL

Algorithm 1 shows the complete STP integration algorithm. First the world state is evaluated on the preconditions and based on those preconditions a play may be selected (lines 1-2). If a play is selected, the play roles are assigned to the active robots using the role assignment (lines 4-5). In the execution part, first the robot checks whether its team mates are still executing the same play, to ensure all robots execute the same team plan. In addition, the invariants are checked whether those are still true (lines 7-10). If one of those is false, the robot will abort the play and updates its play status to NOT_IN_PROGRESS. Finally, the robot uses the *play_selected* variable from the play selection, and the play status to determine whether a play or the default game play will be executed (lines 12-19). Default game play refers to the strategy as explained in Section 3.

5.2 Plays and playbooks

A play P is a predefined team plan which executes a sequence of tactics T_i per robot. The playbook contains a selection of N plays: $\{P_0, P_1, ..., P_N\}$, this selection can specifically be chosen before a game, dependent on the opponent. During a game, one playbook is used. The active play is selected based on the world state. One of the two designed plays for this work, is given by Table 2. The play set-up

Algorithm 1 Algorithm executing a play

```
world.Precond ← evaluateWorldState()
play_selected ← playSelection(world.Precond)

if (play_selected) then
    assignedRoles ← RoleAssignment(play_selected)

if (PLAY_IN_PROGRESS) then
    peerInProgress ← checkPlayStatePeers()
    if (!peerInProgress OR !play.invariants) then
        PLAY_NOT_IN_PROGRESS

if (PLAY_NOT_IN_PROGRESS) then
    if (play_selected) then
        execute_play(assignedRoles)
        PLAY_IN_PROGRESS
    else
        execute_default_play()
else if (PLAY_IN_PROGRESS) then
    execute_play(assignedRoles)
```

is similar to the set-up designed by the CMDragons (Table 1). Preconditions and invariants are defined as world state conditions such as ball possession (ourBall), ball location (ballZone) and number of active opponents. Furthermore, the set of roles R is given with several parameters. First the type of role, which includes the sequence of tactics $\{T_1, T_2, ..., T_N\}$. Secondly, a target position, which is used for role assignment. The sequence of the N roles given by the play $R = \{r_1, r_2, ..., r_N\}$ indicates the priority of each role when assigning roles to the set of robots $\mathcal{P}$. In this case, the AttackerMain is the most important role and the Defender the least important. Section 5.4 discusses this role assignment in detail. Finally, in this play role 1 and 2, the two AttackerAssist roles, are assigned to a specific zone in the field. While positioning during the play execution, the role is bounded to this zone. Figure 3 shows the specified zones and target positions for the roles.

5.3 Play selection

For selection of a play from the playbook $\{p_1, p_2, ..., p_N\}$, first the game state is evaluated. Before, only offensive and defensive game play was distinguished in order to determine the game state. For play selection, the game state is evaluated for all preconditions of the plays, such as the number of opponents and the ball location in field. Each robot evaluates these preconditions individually based on the shared worldmodel as explained in Section 3.

Where in SSL play selection is done by one coordinator, the play selection in a distributed system is done by each robot individually. The robot compares the game state conditions with the preconditions in the plays. In this work, the playbook consists of two plays in addition to the default play. If no preconditions match the world state conditions, default game play is executed. When the world state conditions do match the preconditions, the matching play is selected. To successfully execute a team plan, it is relevant that the team agrees on executing the same play. Therefore, each robot communicates the selected play with peers such that robots can individually determine whether its chosen play is correct. Algorithm 2 describes the process of this check. First the selected plays of all robots are gathered in one list and sorted. From the sorted list, the most common play is found. This play is feasible if the play is chosen by more than half of the number of active robots. If the mode is not feasible, no play will be selected and default game play is executed. If the mode is feasible, each robot will select the mode as the selected play.

Table 2: An offensive play designed for MSL with three offensive robots and one defensive robot

```
AttackWithThree

PRECOND ourBall && ballZone1 && 4
INVARIANTS ourBall
role[0]
    Goalkeeper
    GOAL
role[1]
    AttackerMain
    BALL_POS
role[2]
    AttackerAssist
    PENALTYAREA_CORNER_LEFT
    ZONE_A
role[3]
    AttackerAssist
    PENALTYAREA_CORNER_RIGHT
    ZONE_B
role[4]
    Defender
    {0, -CIRCLERADIUS}
```

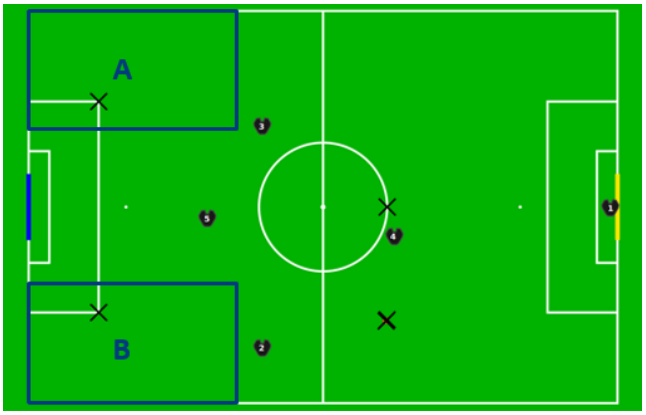


Figure 3: Target positions for roles are indicated with a cross. The AttackerAssist roles are bounded for positioning to zone A and B.

5.4 Distributed role assignment

Once the optimal play has been selected given the state of the world, the roles for that play must be filled by the different robots. As described in Section 3 and 5.2, given the first N listed roles in a play $R = \{r_0, r_1, \ldots, r_N\}$ and the set of robots $\mathcal{P} = \{\rho_0, \rho_1, \ldots, \rho_N\}$, STP optimally assigns each robot to a role, based on a cost function $C : \mathcal{P} \times R \to \mathbb{R}$. The main challenge of transferring this optimal role assignment to the distributed team is that different robots may

Algorithm 2 Sort the list and compute the mode.

```
1: procedure FindMostCommonPlay(list[ROBOTS])
2:     commonPlay[0] ← list[0]
3:     counter ← 1
4:     maxOccurence ← 0
5:     sortedList ← sortList(list)
6:     for all robots i do
7:         if (sortedList[i] = sortedList[i + 1]) then
8:             counter ← counter + 1
9:             if (counter > maxOccurrence) then
10:                maxOccurrence ← counter
11:                commonPlay ← sortedList[i]
12:            else
13:                counter = 1
14:
15:    if (mode not feasible) then
16:        re-evaluate play selection
17:    else
18:        play_selected = commonPlay
```

have different estimates of the state of the game, and thus their cost matrix may differ from each other. It is crucial for robots to agree on the role assignment, to avoid leaving important roles unassigned –e.g., a soccer team with no goalie– or redundancy in roles that must only have one robot –e.g., multiple robots in the team trying to kick the ball at the same time. To ensure this global agreement on role assignment, we assume reliable communication among the robots. Depending on the bandwidth limitations on communication, solutions to this problem may differ. This work considers two solutions. The first is a solution that requires $O(N)$ communication bandwidth while the second solution requires $O(N^2)$. The first solution is a method Tech United currently uses for the role assignment of refbox roles as explained in Section 3. This approach involves each robot broadcasting its locally computed optimal role assignment in a $O(N)$ map from robots to roles. To ensure all robots use the same role assignment, the optimal assignment of one robot is used by the whole team. Robots select the role assignment of the first active robot, usually robot 1. The drawback to this method, is the uncertainty whether the role assignment of robot 1 is the optimal role assignment based on a correct world state. Therefore a second method is considered where each robot ρ_i communicates its cost matrix ρ_i to the team. Then, each robot independently aggregates the cost matrices computed by each teammate, into an aggregate cost matrix $\mathcal{C}$:

$$\mathcal{C} = \frac{\sum_{\rho_i \in \mathcal{P}} [C^i]}{N}. \quad (1)$$

Given $\mathcal{C}$, robots compute the optimal role assignment by minimizing the total costs. This work uses this approach to optimally assign roles to the robot once a play is selected. However, the solution may not be optimal, since the required communication of $O(N^2)$ increases the chance on packet losses.

A desirable solution would be an approach which only requires communication of $O(N)$ information and takes computations of all active robots into account. For example, each robot may choose to broadcast only its own cost for each role, as in previous work [4]. This approach implicitly assumes that each robot can compute its own cost functions best, which is not necessarily the case –e.g., if a robot has a bad estimate of the ball's location, it may compute very inaccurate cost functions for itself. Other approaches to distributed role assignment have also been studied in the literature for domains with different constraints (e.g., [5, 7]).

5.5 Tactics execution

To execute the selected play, a sequence of tactics is performed by each role. Algorithm 3 shows an example of the tactics to be performed by the AttackerMain role. A tactic may involve decision making, –e.g. choosing a pass receiver as shown in the example. To complete the tactic, a set of skills is selected using the function doAction. Tactic transitions take place either after finishing a tactic, –e.g., when the AttackerMain possesses the ball the role transits into the give_pass state, or when a peer robot completes a certain tactic, –e.g., when the AttackerMain possesses the ball, the AttackerAssist roles transits to the receive_pass tactic. Therefore, communication is also required while executing tactics.

Algorithm 3 Sequence of tactics to be executed by the AttackMain.

```
1: switch AttackerMain.tactic do
2:     case intercept_ball
3:         doAction(intercept_ball)
4:     case give_pass
5:         pass_receiver ← choosePassReceiver()
6:         doAction(shoot_at_target, pass_receiver)
7:     case wait_to_end_play
8:         target ← findOptimalPosition()
9:         doAction(go_to_target, target)
```

6. SIMULATIONS AND RESULTS

The integration of STP in the Tech United software is empirically evaluated with simulations. Two offensive plays were integrated, each for different world state, where the ball location precondition differs. Figure 4 show these ball locations. Play 1 will be executed when the ball is located in zone 1. Play 2 when the ball is located in zone 2. For all other ball locations, regular offensive game play, as explained in Section 3, is executed. The simulations are attacks from one of these zones with 4 robots: one goalkeeper and the first three roles defined by the play. These attackers are the black robots in Figure 4. The attacks are performed against two defenders, the red robots in the Figure. The results of 60 attacks using the STP integration, are compared to the results of 60 attacks playing without STP.

6.1 Results

The difference in the attack using STP and not using STP is shown in Figure 5. The black robots are the attacking robots, the two red robots the defenders and the ball is orange and located in zone 1. The two defenders defend in between the ball and home goal. They follow the ball and try to block and intercept passes from the attackers. Figure 5a shows the positioning before the attackers are in ball possession. The AttackerMain (AM) intercepts the ball while the AttackerAssist (AA) roles are positioning within their

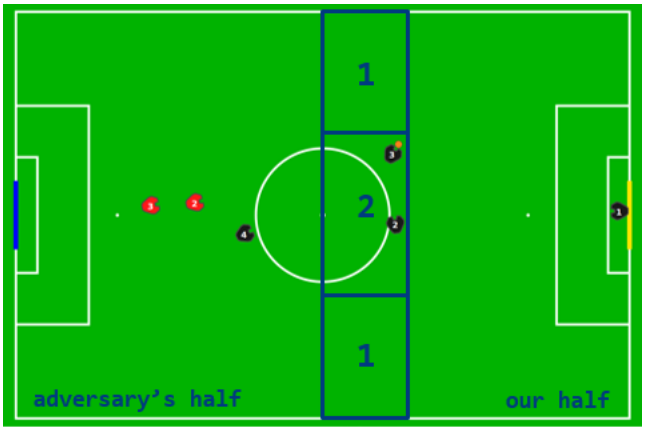
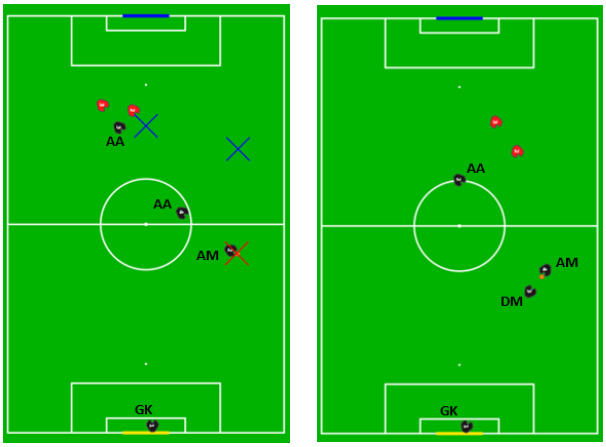


Figure 4: For ball in zone 1, play 1 is executed, for ball in zone 2, play 2 is executed. For all other situations, regular game play is executed. Black robots: attackers, red robots: defenders.

assigned zone. The target positions for the AA roles are indicated with the blue cross, the red cross indicates the target position of the AM. The difference with the attack without using STP, as shown in Figure 5b, is that two robots are positioning on the adversary's half. In Figure 5b can be seen that the AM intercepts the ball, while only one AA is positioning on adversary's half and the DefenderMain (DM) on their own half.



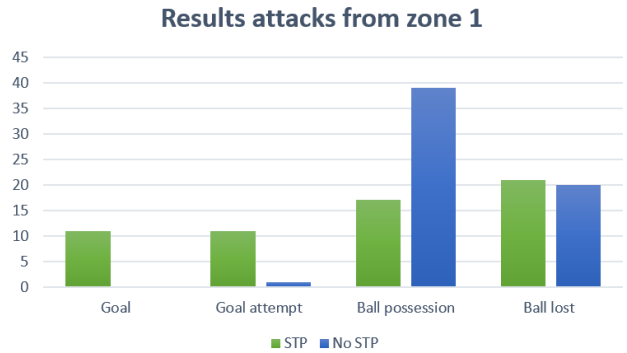
(a) Attack with STP: AM intercepts the ball (target red cross), AA's position in given zone (target blue cross).

(b) Attack without STP: AM intercepts the ball, AA positioning on opponent half, DM positioning on own half.

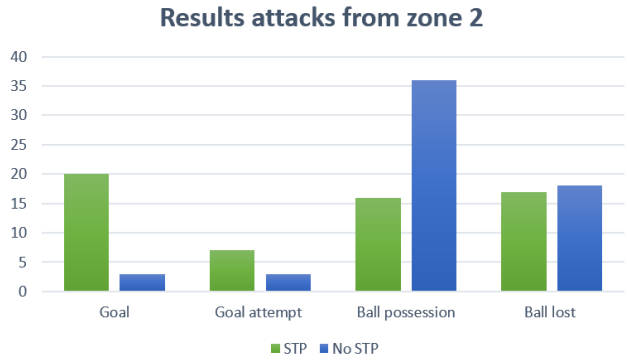
Figure 5: Simulations of attacks using STP and not using STP. Black robots are attackers, red robots defenders.

The results of the attacks from the two zones are given in Figure 6. Attacks were performed from each of the given zones (Figure 4). When the attacking team made four passes, the attack was rated as ball possession for the team. If the attack finished before these four passes, the attack results either in a goal, a goal attempt or the ball was lost. For both plays the similar conclusions can be drawn.

The total number of lost balls, either a failed pass or an interception by the opponent, is equal for both the STP integration and no STP integration. Therefore, it can be concluded that the STP integration does not have an influence on this parameter. Furthermore, attacking without the plays, resulted mostly in ball possession for the team, the attackers did not find the chance to shoot at goal. Using the STP integration on the other hand results in significantly more goals and goal attempts.



(a) Attacks from zone 1.



(b) Attacks from zone 2.

Figure 6: Comparison of 60 attacks between a team using STP and a team not using STP. Graph shows results after max. 4 passes starting in the zone: goal, goal attempt, ball possession or the ball was lost.

7. CONCLUSIONS AND FUTURE WORK

This paper presented a novel effort in adapting team-level strategy from the centrally-controlled Small Size League of RoboCup to the distributed Middle Size League. In particular, the Skills, Tactics and Plays (STP) team-planning architecture was successfully integrated in the Tech United team. The main challenges for this integration were the agreement on a common team plan to execute and optimally assign the team plan roles to the active robots. Voting-based approaches were used to overcome these challenges. Each robot individually selects a play and computes the costs for optimal role assignment. Both are communicated among all robots. The most common selected play is chosen by each robot and role assignment is computed based on an averaged

cost matrix. Both methods are proven to be robust by the performed simulations.

Simulations with and without the STP integration were performed. Three robots performed attacks against to adversary's for game state situations for which two plays were designed. The results showw a significant improvement while applying appropriately-offensive play for specific game situations chosen for this work: ball possession in zone 1 and 2.

This work shows a successful cooperation between two different RoboCup leagues. While the SSL has been developing strategy algorithms for fast-paced team play over the years, in MSL teams have been focusing on controlling a distributed team. Integrating a well-developed strategy algorithm from the SSL, such as STP, into the MSL, helps the league many steps forward. Such collaborations, where knowledge and algorithms are shared among leagues, are desirable in order to accomplish the ultimate RoboCup goal: beating the human World Cup winner of 2050 with a fully autonomous humanoid robot soccer team.

REFERENCES

[1] J. Biswas, J. P. Mendoza, D. Zhu, B. Choi, S. D. Klee, and M. M. Veloso. Opponent-driven planning and execution for pass, attack, and defense in a multi-robot soccer team. In *Proceedings of AAMAS'14, the Thirteenth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2014.

[2] B. Browning, B. Bruce, M. Bowling, and M. Veloso. Stp: Skills, tactics and plays for multi-robot control in adversarial environments. In *Proceedings of the Institue of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, pages 219(1):33–52, 2004.

[3] B. Brumitt and A. Stentz. Dynamic mission planning for multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume No. 3, pp. 2396-2401, 1996.

[4] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Coordination among heterogeneous robotic soccer players. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 2, pages 1385–1390. IEEE, 2000.

[5] A. Farinelli, P. Scerri, and M. Tambe. Building large-scale robot systems: Distributed role assignment in dynamic, uncertain domains. In *Proceedings of Workshop on Representations and Approaches for Time-Critical Decentralized Resource, Role and Task Allocation*, volume 102, 2003.

[6] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti. Robot task planning using semantic maps. In *Robotics and Autonomous Systems*, volume 56 Issue 11, 2008.

[7] L. Hunsberger and B. J. Grosz. A combinatorial auction for collaborative planning. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 151–158. IEEE, 2000.

[8] R. Janssen. Centralized learning and planning: for cognitive robots operating in human domains. 2014.

[9] R. Jensen and M. Veloso. Obdd-based universal planning: Specifying and solving planning problems for synchronized agents in non-deterministic domains. In *Lecture Notes in Computer Science*, volume No 1600, pp. 213-248, 1999.

[10] W. Kuhn, H. The hungarian method for the assignment problem. In *Naval Research Logistics*, pages 83–97. IEEE, 2006.

[11] C. Lopez Martinez, F. Schoenmakers, G. Naus, K. Meessen, Y. Douven, H. van de Loo, D. Bruijnen, W. Aangenent, J. Groenen, B. van Ninhuijs, M. Briegel, R. Hoogendijk, P. van Brakel, R. van den Berg, O. Hendriks, R. Arts, F. Botden, W. Houtman, L. de Koning, R. Soetens, and R. van de Molengraft. Tech united eindhoven, winner robocup 2014 msl. In *RoboCup 2014: Robot World Cup XVIII*, pages 60—-69.

[12] J. P. Mendoza, J. Biswas, D. Zhu, R. Wang, P. Cooksey, S. Klee, and M. Veloso. Cmdragons 2015: Coordinated offense and defense of the ssl champions. In *RoboCup 2015: Robot World Cup XIX*.

[13] L. E. Parker. Designing control laws for cooperative agent teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume pp. 582-587, 1993.

[14] P. Svestka and M. Overmars. Coordinated path planning for multiple robots. In *Robotics and Autonomous Systems*, volume Vol 7, pp. 83-124, 1997.

[15] Zickler, Laue, Birbach, Wongphati, and Veloso. Ssl-vision: The shared vision system for the robocup small size league. In *RoboCup 2009 Symposium*, pages 425—436.