2018 STOx's Extended Team Description Paper

Saith Rodríguez, Eyberth Rojas, Katherín Pérez, Carlos Quintero, Oswaldo Peña, and Andrés Reyes

Faculty of Electronics Engineering
Universidad Santo Tomás
Bogotá, Colombia
{saithrodriguez,eyberthrojas,kathyperez,carlosquintero,
oswaldopena,andresreyesf}@stoxs.org
http://www.stoxs.org

Abstract. In this paper we show STOx's latest methodology for multiagent/robot coordination. The general idea consists on a Game Analyzer module that dynamically creates tasks that need to be solved in every game situations together with an Assignment Module that automatically assigns tasks to agents optimally. This architecture has the advantage that establishes coordination among all agents automatically and optimally. We show a detailed description of our optimal assignment module by modeling it as a minimax assignment problem and also present our proposed solution, briefly discussing its computational complexity. Finally, we present a concrete example of plays that happened during games in RoboCup 2017 and discuss implementation details and future challenges.

1 Introduction

The Small Size League (SSL) is one of the first leagues in RoboCup and it deals with the high level coordination of multiple agent/robots in a highly dynamic and rapidly changing environment. The league's distinctive mark is precisely its fast pace, i.e., the high frequency at which events occur during games. For a team to be competitive and successful within the league, several factors need to be taken into account. On one hand, robust and reliable robots are required, in terms of their electrical and mechanical design. It is also important to have accurate low level control of the robots, capable of handling the most basic skills required to create more complex plays. Finally, it is necessary to create smart, fast and dynamic plays as well as techniques that allow the team to adapt to complex behaviors and learn from experience.

The STOx's team has grown in experience during its years of participation, from 2011 to 2017, achieving a 4^{th} place in 2015 and among the top 8 in 2013, 2014 and 2017. Although the team still has plenty of room to improve on all the elements mentioned above, our greatest interest has been on developing more sophisticated AI and the ability to make the system able to take better decisions. This is noteworthy in the latest STOx's team description papers as described below.

In 2014, the TDP [13] showed the details of mechanical and electrical aspects of our latest generation of robots. In 2015 [7], however, we presented a set of data processing techniques that aimed at improving the representation of the virtual world, i.e., the position and velocities of ball and robots within the field, even in the presence of noise from the vision system. These tools have been of paramount importance to the decision maker since they provide confidence and allow the system to make predictions. In 2016 [2], we developed a tool to automatically identify and reconstruct chip kicks during games and also a methodology to move from completely fixed plays (i.e., where the positions and actions of all agents are previously defined by the programmers and are the same every time the play is ran) to plays automatically generated by the AI system, that are created on the run depending on the opponent reaction. Finally, in 2017 [1] we showed our team's defensive strategy, which is based on robot-to-robot marking, threats identification, a hierarchical attackers selection (to be marked) and finally an *optimal marker assignment*.

With this set of developed tools and inspired in our latest optimal marker assignment for defensive strategies, we have come up with a general methodology that allows us to optimally distribute tasks between agents in any game situation. This new methodology has the following advantages:

- It allows automatic assignment of agents to tasks, avoiding situations where missing agents might be assigned with tasks.
- It creates optimal agents-tasks assignments, where the traveled distance of each agent is the lowest possible.
- It is general enough to be used in a variety of different applications other than robotic soccer.
- It facilitates the creation of defensive and attacking plays that are automatically created by the system on the run in contrast to fixed, preprogrammed plays.

In the following sections we will show the most important concepts of our proposed methodology as well as our proposed solution. We also briefly discussed about its computational load, compared to other possible implementations and finally, we show some initial results of our methodology applied to games from RoboCup 2017. Fig. 1 shows a picture of the STOx's team members in RoboCup 2017 at Nagoya.

2 Multiagent robot coordination methodology

The SSL game is characterized as one of the fastest among the RoboCup community and it is very important being capable of make intelligent decisions. Currently there is enough space in the field for fasts and plays difficult to predict that make the tasks of attacking and defending a difficult one. Specifically, the distribution of tasks among the agents is of paramount importance, since the election of one or other agent to perform certain task could radically change the game result.

Fig. 1: Picture with the current STOx's members in RoboCup 2017

We have developed a general framework that allows the system to automatically distribute tasks among agents in a variety of situations during the game.

In our approach we have defined a set T of tasks that need to be performed by one or more agents in specific situations. Each task is usually associated with a physical location within the field. For example, a task associated to perform robot-to-robot marking is related to the physical location of the attacking robot. The task of going after the ball is related to the ball's coordinates and so on.

The goal is to assign agents to tasks dynamically during gameplays in an optimal manner (in some sense). This approach has became a powerful general approach to distribute tasks among agents that works for a variable number of agents in the field in any state of the game and for any number of tasks.

We claim that our multi-agent robot coordination is general enough that it can be used in applications others than robotic soccer where a set of agents need to perform a set of tasks. In Fig. 2 we show the general scheme of our coordination framework.

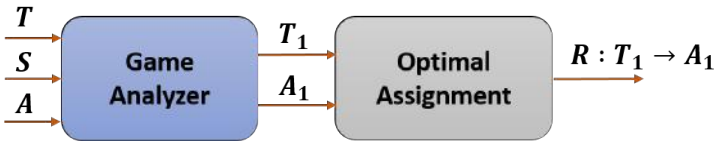


Fig. 2: STOx's Multi-agent robot coordination framework

The framework includes two main modules, namely **Game Analyzer Module** and **Optimal Assignment Module**. The former is in charge of deciding which tasks need to be done depending on the current game state and the available agents, together with their locations. The module receives the complete set of possible tasks, the current game state and analyzes which tasks should be performed and the subset of agents that will participate in the assignment. The **optimal assignment module** receives such information and outputs an assignment of agents to tasks such that some cost function is optimized.

As an example, consider a situation as the one shown in Fig. 3 where an indirect free kick has been awarded to the yellow team. Such a team will require one agent to kick the ball (kicker), two agents that will potentially receive the pass (attackers) and one agent that will create distraction to the defenders (distractor). The blue team, on the other hand, will require one agent to block the ball (wall) and at least two agents to mark the two yellow attackers (depending on the teams strategies, the defending agents could do something else). Under this scenario, the Game Analyzer module for the yellow team has decided that from all possible tasks, the team should perform the four tasks discussed above (i.e., kicker, attacker, attacker and distractor). Similarly, the Game Analyzer module for the blue team has decided that 3 tasks (wall, marker and marker) should be performed for this game state. This module has also related each task with a physical location in the field. For example, for the attacking team, the kicker task is related to the ball, the attackers can be related to locations where the chances of successfully receiving the pass are high and finally, the distractor can be related to a distraction routine or other strategy. Similarly, for the defending team, the wall task is closely related to the ball's position and the marker tasks are related to the location of the attackers.

For RoboCup 2017, we have defined the set of all possible tasks in the following way:

- **Defender:** A task where the agents are required to locate near the defense area edge as in a wall, blocking potential shots to the goal.
- **Ball Handler:** This task is assigned to an agent that will be performing activities related with the ball. For example, in a defending situation, it could be the agent that will act as wall when the opponent team will kick the ball. When attacking, in a free kick situation, the ball handler will be the agent that will kick the ball.
- **Marker:** This is a task related to defending situations. Some threatening opponents will be assigned markers to perform robot-to-robot mark.
- **Supporter:** In offensive situations, the supporters are agents that need to find locations within the field to potentially receive passes.
- **Attacker:** This task is assigned to an agent with the ball's possession. This agent can dribble the ball and will subsequently perform passes.
- **Distractor:** This task is assigned to agents that will perform distracting moves durinig the game.
- **Clearer:** This is usually assigned to agents in charge of clearing the ball near the defensive area.

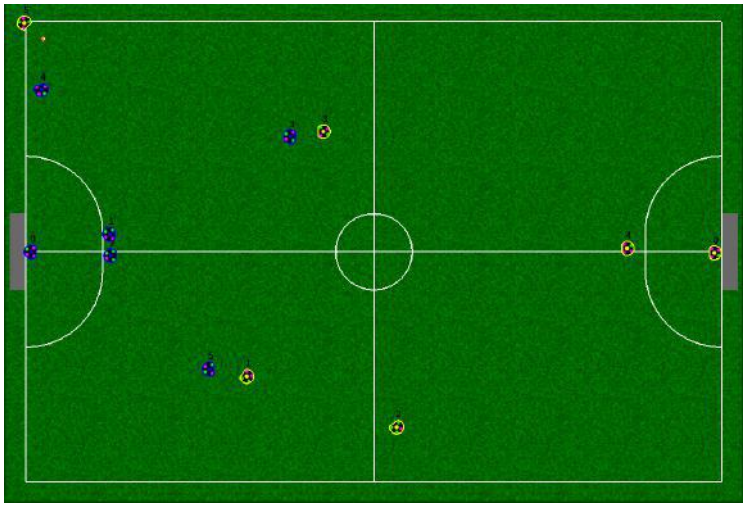


Fig. 3: Example of how the Game Analyzer and Optimal Assignment modules might work for two teams in a free kick situation. The yellow team is attacking while the blue team is defending.

Notice that this set of tasks is not fixed and may updated/changed according to specific needs and requirements. This module is also easily scalable in terms of the number of available agents. For example, if the team is missing one or two agents for whatever reason (yellow cards or mechanical/electrical problems) during the game, the module will just not include them in the agents-tasks assignment. This strategy avoids situations where the system commands a specific robot to go to kick the ball in a free kick and the play is not executed because the robot is temporarily out of the field. Also, if the number of agents is increased from the current number, the Game Analyzer will just include the new agents into the assignment and there will be no need to radically change the program.

Notice that this module is the one that contains the expertise of the problem at hand (soccer in this case), since it receives the set of all possible tasks, the set of all available agents and what the current state of the game is and it has to decide what needs to be done next. The team's strategies are expressed within this module (i.e., the attacking and defending specifics). Also, notice that for other applications, one would need to include a specific Game Analyzer module for the task at hand to create tasks according to the specific challenge. However, the general multi-agent robot architecture may remain the same.

In the following section we describe the details of the optimal assignment module as well as its current implementation.

2.1 Optimal Assignment Module

Formally, the optimal assignment module receives a set T_1 of tasks, together with their corresponding physical location within the field and a subset A_1 of agents

also with their corresponding locations. Its output is a bijection $R : A_1 \rightarrow T_1$ such that the distance traveled by the agents is the smallest possible.

The idea is that the agents/tasks distribution is achieved such that each agent is assigned a task that is as closest to it as possible, minimizing the total distance traveled by all agents. To this end, we propose to find an assignment that minimizes the largest distance traveled by all agents. This approach has shown to find assignments where all agents are required to travel short distances in contrast to other proposals where the sum of all distances is minimized. The latter usually finds minimal total traveled distance, but with individual distances that may be large.

In combinatorial optimization, this can be expressed as the minimax assignment problem, which is in turn, a modification of the global assignment problem, where the objective function is nonlinear. The decision variables x_{ij} are binary and $x_{ij} = 1$ represents agent $i \in A_1$ is assigned to task $j \in T_1$, while $x_{ij} = 0$ otherwise. Finally, one agent must be assigned at most one task and the number of agents must exceed or at least be equal to the number of tasks i.e., $|A_1| = m \geq |T_1| = m$. The problem can be formulated as follows:

$$\begin{gathered}
\min z = \max_{i,j} \{x_{ij} C_{ij}\} \\
\text{subject to} \\
\sum_{i=1}^{m} x_{ij} = 1, \forall j \in T_1 \\
\sum_{j=1}^{n} x_{ij} = 1, \forall i \in A_1 \\
x_{ij} \in \{0, 1\}, \forall j \in T_1, i \in A_1
\end{gathered}$$

The General Assignment Problem has shown to be NP-complete [14], meaning that there is no algorithm that can solve it in polynomial time. The exhaustion method has shown to be useful to solve the minimax assignment problem on a computer when $n = m < 10$. Under our scenario, we consider cases where $m > n$, but in all cases $m, n < 10$.

Solving the general minimax assignment problem As the number of instances for our optimal assignment problem remains low (i.e., the number of agents and tasks are lower than 10), we propose to use the exhaustion method to attain an optimal assignment. To this end, we first propose to solve instances where $m = n$ and then, use such solutions to find the solution of the general minimax assignment problem. In the following section we will go through the details of solving the first problem.

Balanced Minimax Assignment Problem The formulation of the balanced minimax assignment problem is identical to the one shown in Eq. (1), but constrained that $m = n$. In this method, we build all feasible assignments of agents to tasks and compute the cost for each assignment, which corresponds to the maximum distance between agents and tasks in such assignment. Afterwards, we keep the cost of all assignments and find the minimum.

Each assignment corresponds to one permutation $\{j_1, j_2, ..., j_n\}$ of tasks of $\{1, 2, ..., n\}$ with cost $z = \max\{C_{1j_1}, C_{2j_2}, ..., C_{nj_n}\}$. The goal is to compute all $n!$ permutations and then find the one with minimum value. Fig. 4 shows one example of such permutations where $m = n = 3$.

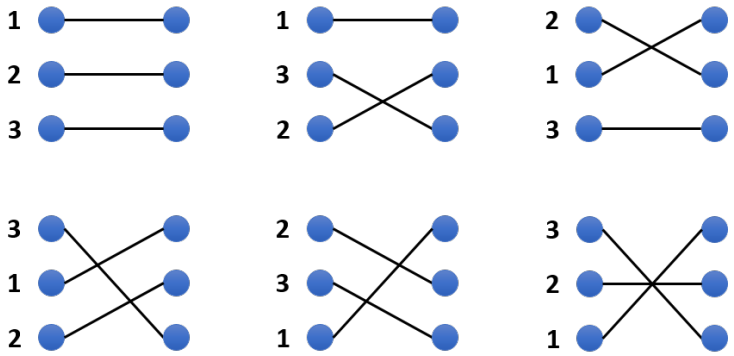


Fig. 4: All possible permutations of the 3 available agents. Notice that each agents permutation corresponds to one unique assignment to tasks where its cost is the maximum distance between agents and tasks in such assignment.

General Minimax Assignment Problem For the general case, we allow more agents than tasks (i.e., $m < n$). This makes sense since under some considerations, it may be possible that only a few specific tasks need to be done. In these cases, al available agents are included in the decision process and those selected by the optimization problem are assigned to accomplish the set of tasks. The remaining agents can be assigned "routinary" tasks such as general defense.

In order to solve the general minimax assignment problem we propose to decompose the problem into several balanced minimax assignment problems and select the one with best value of the cost function. The decomposition consists on selecting a subset of n agents out of the m and solving the balanced minimax assignment problem for those n agents. Afterwards, we select a different subset of n agents and solve the problem again. This process is repeated for all possible combinations of n agents from the complete set of m and finally select the one with lowest value of the cost function.

Fig. 5 shows one example of such methodology for $m = 4$ and $n = 2$. For each shown combination, one balanced minimax assignment problem of size 2 needs to be solved.

Complexity Analysis As stated in Sec. 2.1, it has been shown that the general assignment problem is NP-complete, which restricts the solution in computers to

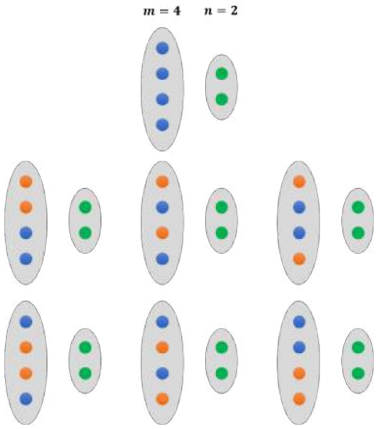


Fig. 5: All possible subset of 2 agents out of 4 that is possible to create. In each situation, the blue dots correspond to *not selected* agents, the orange dots are the agents selected to solve the balanced minimax assignment problem and the green dots correspond to the tasks.

only instances with low values of m and n. Additionally, the exhaustion method used here iterates over all possible combinations of agents and permutations of tasks to find the optimal assignment.

Another possible solution that is often used when $m < n$ is to add to the problem "dummy" tasks with zero cost in order to make it a balanced assignment problem. However, in this case we did not consider it since it requires more operations than the method proposed here. Below, we show a simple complexity analysis of the number of operations that need to be done in both cases and show that our proposal requires less computational load.

In our method, we first calculate the distance matrix C of all distances between agents and tasks. Afterwards, we iterate over all possible combinations of n agents, which is $\binom{n}{m} = \frac{m!}{n!\,(m-n)!}$. For each combination we solve the balanced assignment problem by generating all possible permutation of tasks to agents for the n-size problem which is $n!$. In each permutation we need to compute the maximum distance of the assignment problem (maximum value among n values), meaning that the proposed method needs to perform $\frac{m!}{(m-n)!}$ operations to find the optimal value.

For the "dummy task" approach, the number of total iterations required to find the optimal assignment is always $m!$, i.e, the number of iterations required to perform all possible permutations of m agents/tasks. This analysis shows that our method requires less operations than the one adding the "dummy task".

3 Experiments and results

Our multi-robot agent coordination methodology has been implemented in STOx's software since RoboCup 2017 and tested in all games. As mentioned above, the pre-programmed plays are not possible when using the methodology since the system is automatically assigning tasks to agents optimally on the run, which brings us closer to our ultimate target of achieving an automatic and optimal AI system.

In Fig. 6 we show a sequence of images that correspond to one specific game situation in one game from RoboCup 2017. The images are representations of logs visualized in our software and show sequences of what happened during a short lapse during the game together with some information that helps to better understand the situation. In the sequence, we show one play where several assignments are performed. The reader will be able to judge the optimality of the assignment in each situation depending on the generated tasks. During the play, we also show that our team starts in a defensive position and changes to an attacking position after recovering the ball.

Other results already available, but not shown here will be presented in an ongoing paper. Some of them are qualitative results of the assignment optimality, quantitative measures of the computational time taken to solve the minimax assignment problem and comparisons with other assignment methods.

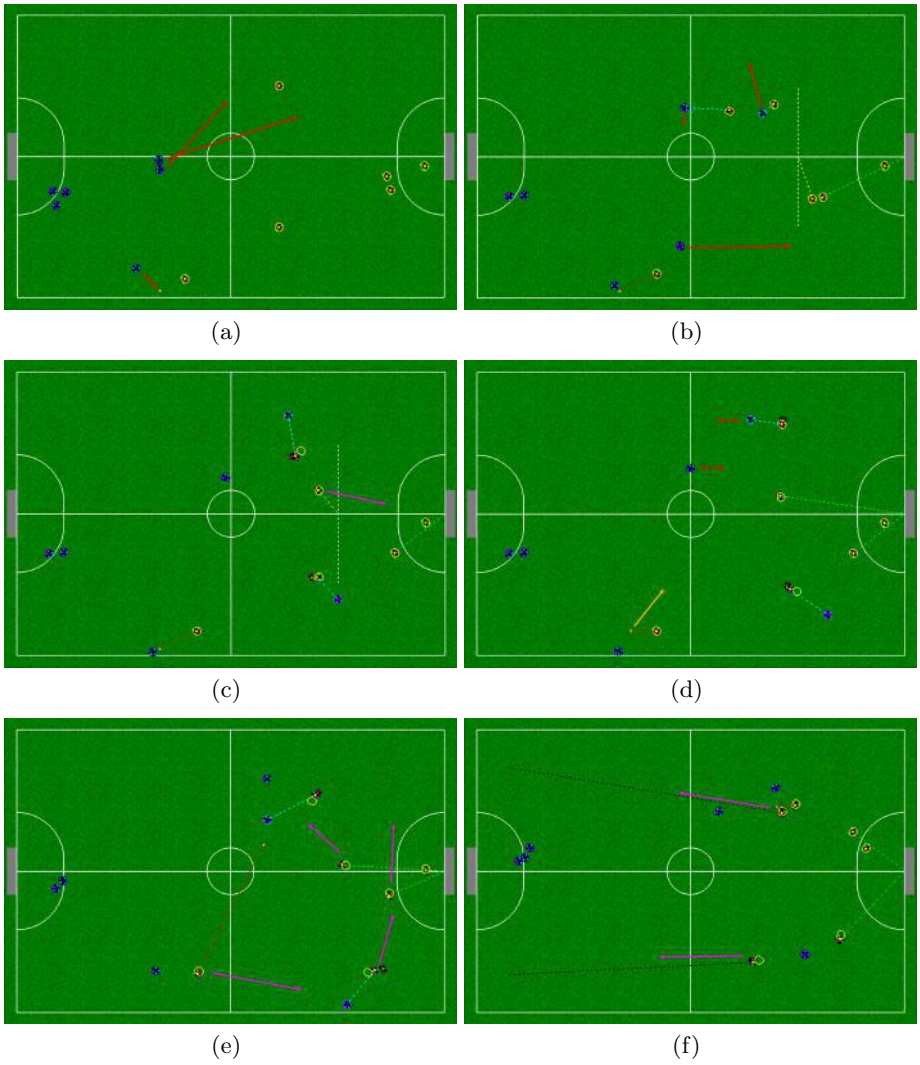
In the situation shown in Fig. 6, STOx's is the yellow team, the game is in stop mode and the referee have just issued a free kick in favor of the blue team. Fig. 6(a) shows the initial positions of all agents as well as the ball position. The red arrows show the change in position from several agents of the blue team when the game restarts. In Fig. 6(b), the blue agents have moved, one of them closer to the ball, another near STOx's defense area and two of them near the middle of the field, probably to create distraction. Since this is clearly a defense situation, our Game Analyzer has created a set of tasks to defend from the attacking play as follows: 1 **Ball Handler**, 1 **Clearer**, 2 textbfMarkers, 1 **Defender**. The optimal assignment is performed and the results are shown as colored dash lines in the figure. The ball handler is the agent that was previously closer to the ball (shown in dashed red line), the two most threatening attackers are marked by the two closest STOx's markers shown with blue line and the clearer and defender are also selected with minimum distance shown in white and green line respectively. Red arrows show the movements of three opponent agents. This is an important step in the sequence since notice in Fig. 6(c) that one opponent agent that was not previously marked moved closer to STOx's defense area becoming a serious threat in the play and requiring a new task/agent assignment. The STOx's agent that was previously assigned as **Clearer**, it is now assigned as **Marker** of the new threatening opponent. The two other previously marked opponents made small movements and out algorithm changes one of them as **Clearer** and the other remains as **Marker**, but changes the agent being marked. Notice that all chenges has performed by our assignment algorithm and are optimal in the traveled distance by the agents. In Fig. 6(d) the kicker has just kicked the ball and the Game Analyzer decides that a **Clearer** is no longer needed. When the

new assignment is performed, all agents remain with the same tasks, but the one that was previously assigned as **Clearer** is now a **Defender**. The orange arrow shows the ball movement direction, while red arrows show the opponents movement for the next image. In Fig. 6(e) the ball has traveled more distance and the opponent agent that was previously unmarked has moved to receive the ball and hence has become a threat. The assignment is performed and this agent is marked now by one **Marker**, who will move near the attacker to stop it from receiving the pass. Magenta arrows show how STOx's agents will move in the next and final step. In Fig. 6(f) we can appreciate the next assignment. The pass made by the blue team was not successful since the ball was recovered by one of STOx's markers. For this reason, the Game Analyzer changes from defensive mode to attacking mode and marks are no longer required. In this scenario the tasks are the following: 1 **Ball Handler**, 2 **Support Attackers** and 2 **Defenders**. In the figure we can see that the agent that was previously a marker is now assigned as **Ball Handler** (red dashed line), one of the defenders is now assigned as **Support Attackers** and the other previously marker is the other **Support Attacker**. The latter are shown in black dash lines. The **Ball Handler** will now possibly dribble the ball to attempt a pass to one of the support attackers while they will move close to the opponent defense area to receive the pass. In this last assignment, notice that the robots that are closer to the opponent defense area are the one chosen to be **Support Attackers**. Also notice that the assignment makes sense since they do not have to change sides (i.e., the one who was in the left side of the field will remain in that side as well as the one who was in the right side).

4 Conclusions and discussion

In this paper we have presented the latest contributions of the STOx's team in our participations in the RoboCup competitions. We have also showed our general coordination methodology which is based on a Game Analyzer that dynamically generates tasks that need to be done depending on the current game situation. These tasks and the set of available agents are inputs to our Optimal Assigner, which assigns agents to tasks in an optimal manner. We have modeled this situation as a minimax assignment problem, where the objective is to minimize the maximum distance between agents and tasks in all assignments. For the nature of the problem at hand, we have proposed to use the exhaustion method to solve it, by evaluating all possible combinations and showed that this method works better than assigning dummy tasks to the problem when the number of agents is not equal to the number of tasks.

Our methodology has been successfully tested in all STOx's games from RoboCup 2017 and only a simple play example is shown here where initially the team is in a defending state and goes to an attacking state by recovering the ball. All task generations and assignments are performed automatically and are optimal in the sense of agent's traveled distances.



(a) (b)

(c) (d)

(e) (f)

Fig. 6: Results of our multiagent robot coordination in one play of RoboCup 2017

From the implementation point of view, we have shown that our solution is feasible and that it is fast enough to run in real time during regular games. The generation of tasks and more importantly, the tasks assignment, is performed frame by frame during the games. It is noteworthy that due to the nature of the objective function in Eq. (1), it is possible to find more than one optimal assignment. In this situations, we have decided to keep the solution (among all the optimal solutions) with minimum sum of distances between agents and tasks. Another aspect that worth mentioning is that it is important to be aware that agents are usually moving and because the assignment is performed frame by frame, it only takes into account the current agents positions. Under some conditions this might be problematic since it changes assignments from frame to frame leading to an indecisive system. Our solution to this problem has been to perform the assignment in all frames, but only updating to the new assignment if it is better than the current one (e.g., if the new calculated assignment is lower by some threshold value than the current one). Another solution that we are currently exploring is to include velocities in the optimization problem that takes into account the agents motion.

Finally, new challenges might be added to our methodology under recent changes in the league rules. Probably, the most important one is the increase in the number of agents, which for RoboCup 2018 will be 8 and for 2019 will be 11. This might be a problem within our methodology since the computational time required to solve the minimax assignment problem grow exponentially with the number of agents. Further experiments will be required and new methods to solve the problem might also be needed.

References

1. S. Rodríguez, E. Rojas, K. Pérez, C. Quintero, O. Peña, A. Reyes, J.M. Calderón. STOx's 2017 Team Description Paper. Nagoya, Japan 2017. Available for download in http://wiki.robocup.org/images/d/d7/Robocupssl2017-final16.pdf
2. S. Rodríguez, E. Rojas, K. Pérez, C. Quintero, H. Báez, O. Peña, J.M. Calderón. STOx's 2016 Extended Team Description Paper. Leipzig, Germany 2016. Available for download in http://wiki.robocup.org/images/0/0d/Small_Size_League_-_RoboCup_2016_-_TDP_STOx%27s.pdf
3. C. Lobmeier, P. Blank, J. Buehlmeyer, D. Burk, M. Eischer, A. Hauck, M. Hoffmann, S. Kronberger, M. Lieret and M.B. Eskofier. ER-Force Extended Team Description Paper RoboCup 2016. Available for download in http://wiki.robocup.org/images/0/07/Small_Size_League_-_RoboCup_2016_-_TDP_ER-Force.pdf
4. R. De Iaco, S. Johnson, F. Kalla, L.k. Lu, M. MacDougall, K. Muthukuda, J. Petrie, M. Ragoonath, W.Y. Su, V. Tang, T. Tsu, B. Wang, G. Whyte, C. Xi, K. Yu, E. Zhang and K. Zhang. 2016 Team Description Paper: UBC Thunderbots. Available for download in http://wiki.robocup.org/images/f/f8/Small_Size_League_-_RoboCup_2016_-_TDP_UBC_Thunderbots_-_Review.pdf
5. S. Aoki, T. Degawa, K. Fujihara, Y. Notsu, T. Beppu. MCT Susano Logics 2016 Team Description. Available for download in http://wiki.robocup.org/images/2/27/Small_Size_League_-_RoboCup_2016_-_TDP_MCT_Susano_Logics.pdf

6. A.G. Poudeh, M. Sobhani, A. HosseniKia, A. Karimpour, S. Mosayeb, H. Mahmudi, S. Esmaeelpourfard, M.K. Naeini and A. Adhami-Mirhosseini. MRL Extended Team Description Paper 2016. Available for download in http://wiki.robocup.org/images/8/8d/Small_Size_League_-_RoboCup_2016_-_TDP_MRL.pdf
7. S. Rodríguez, E. Rojas, K. Pérez, J. López, C. Quintero, O. Peña, J. Giraldo and J.M. Calderón. STOx's 2015 Extended Team Description Paper. Hefei, China 2015. Available for download in http://wiki.robocup.org/wiki/File:Small_Size_League_-_RoboCup_2015_-_ETDP_STOx%E2%80%99s.pdf
8. L. Jin, L. Chen, X. Chen, Y. Li, W. Hu and R. Xiong. ZJUNlict Extended Team Description Paper for RoboCup 2016. Leipzig 2016. Available for download in http://wiki.robocup.org/images/6/60/Small_Size_League_-_RoboCup_2016_-_TDP_ZJUNlict.pdf
9. T. Sakaguchi, K. Ohno, T. Mimura, N. Tanaka, K. Satoh, Y. Yamauchi, Y. Nagasaka, M. Watanabe and T. Sugiura. Leipzig 2016. Available for download in http://wiki.robocup.org/images/2/20/Small_Size_League_-_RoboCup_2016_-_TDP_KIKS.pdf
10. Y. Adachi, H. Kusakabe, N. Tsuzuki, H. Hayama, D. Yamaguchi, M. Ito and T. Naruse. RoboDragons 2016 Extended Team Description Paper. Available for download in http://wiki.robocup.org/images/5/51/Small_Size_League_-_RoboCup_2016_-_TDP_RoboDragons.pdf
11. 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 Proceedings of the International RoboCup Symposium, to appear 2016.
12. J.P. Mendoza, J. Biswas, D. Zhu, R. Wang, P. Cooksey, S. Klee, and M. Veloso. CMDragons 2016 Extended Team Description Paper. Available for download in http://wiki.robocup.org/images/d/dc/Small_Size_League_-_RoboCup_2016_-_TDP_CMDragons.pdf
13. S. Rodríguez, E. Rojas, K. Pérez, J. López, C. Quintero, and J.M. Calderón. STOx's 2014 Extended Team Description Paper. Joao Pessoa, Brazil 2014. Available for download in http://robocupssl.cpe.ku.ac.th/_media/robocup2014:etdp:stoxs_2014_etdp.pdf
14. L. Yang, M. Nie, Z. Wu and Y. Nie. Modeling and Solution for Assignment Problem. International Journal of Mathematical Models and Methods in Applied Sciences. Issue 2, Volume 2, 2008.
15. R. Rojas, M. Simon and O. Tenchio. Parabolic Flight Reconstruction from Multiple Images from a Single Camera in General Position. In Chapter RoboCup 2006: Robot World Cup X. Vol 4434 of the series Lecture Notes in Computer Science pp 183–193, 2007.