# Computational system for strategy design and match simulation in team sports

Leonardo Lamas[1], Guilherme Otranto[2], and Junior Barrera[2]

[1] University of Brasilia, Faculty of Physical Education, Campus Darcy Ribeiro, Brasília, Brazil
[2] University of São Paulo, Institute of Mathematics and Statistics, Rua do Matão, n.1010, Cidade Universitária, São Paulo, Brazil

**Abstract.** The goal of the present work was to create a computational system that supports the design of strategies and the match simulation based on those strategies. A formal model of team strategy and match dynamics supported the specification of the computational system. In this model, team strategy was defined as a discrete dynamic system. The specification of individual action rules enables the team players to organize the collective action in every state of the system. A play is modelled by a sequence of compatible pairs of states. The system implementation encompasses a designing tool, whose resultant strategies are used as the input in a simulator capable of recognizing match states and applying the defined strategy to plan actions. Besides the inherent contribution to the investigation of team sports performance features, the presented framework may be helpful in other scientific areas, such as those that investigate cooperative actions in competitive environments and to the design of video-games with a greater realism, approximating them to real match simulators.

Key-words: dynamical system, action rule, planning, artificial intelligence

## 1 Introduction

In team sports, the set of specifications conceived by the coaching staff to support the collective action of the players define the concept of strategy [3]. Even though in many circumstances of the match the execution of the team strategy should be modified due to the spatio-temporal constraints mutually imposed by the adversaries, it greatly influences the tactical patterns observed in the confront [5].

The empirical observation of a match leads to the conjecture that, in most of the cases, the more collectively a team plays the greater should be its performance. Hence, the success of a team appears to be related both to the features of the designed strategy and to the team efficiency to execute them in the match. Computational systems that support the design of team strategies and simulate their execution may contribute to the scientific investigation of the several elements involved in this complex context. However, in the present moment, interactive computational environments for strategy design (i.e. environments that can correct and improve the human planning) and team sports simulators, analogous to those in existent for improving car races' strategies [6] and for training the pilots in the designed strategies, are not available. Interestingly, in computer science, several algorithms have been systematically improved to reproduce more precisely the competition features of team sports in video games [7, 2], indicating an eminent common research field for scientists from this area and sport sciences. Based on this inter-disciplinary approach, the goal of the present work was to create a computational system that supports the design of strategies and, based on the designed strategies, simulate the match in team sports.

## 2 Methods

### 2.1 Specification of the computational system

A model of the team strategy and the match dynamics [4] supported the development of both modules of the computational system, the team strategy designer (TSD) and the match simulator.

The team strategy model was defined through the following elements: i) control of a player actions by strategy specifications; ii) organization of cooperative relations between team players in each strategy state; iii) graph representation of sequences of states [4]. Additionally, strategies of the adversary teams were inputs to the match dynamics [4].

The most fundamental element of the model is the action rule, a conditional statement of the form if <condition>, then <action> (e.g. in basketball defense, if <an attacker overcomes my teammate>, then <I should help defending his attacker>). An action rule formalizes the logical control of the players dynamics through the action choice of the players in a given context. Players' action rules provide the specifications for creating small cooperative groups of team players, defined as strategic units (SUs), which should perform a coordinated action to achieve a certain strategic goal in a state.

A strategy state is composed by the following constitutive elements: i) players in the match field; ii) a region for each player, represented by an equivalence class for positioning that encompasses all points in the match field with similar meaning for a player in a given state; iii) players dynamics; iv) whether the team has ball possession (i.e. offense or defense); v) the ball dynamics. A state transition specifies the roles of the team players (i.e. trajectory in the match field and respective technical skills performed) and their dynamics. To design a consistent sequence of states, two subsequent states must be compatible. Thus, the output of the first state should be identical to the input of the second state. A team strategy is composed of a finite set of states and connections between pairs of compatible states. A play is modelled by a sequence of compatible pairs of states and two or more plays may be originated in a common state. Thus, every team strategy can be represented by a directed graph. The graph enables the identification of the general structure of the strategy and provides easy visualization of specific sequences of states (i.e. plays) in the context of the complete strategy.

In a match, during the interval between two subsequent interruptions, the confrontation is continuous and both teams try to make adjustments on their performance to approximate to the specifications of the strategy states. Hence, each modification that emerges from the confront, originated from the interaction between the two teams, generates information that is confronted through feedback, with the planned state from the team strategy. This input leads to the comparison between the match state and the closest state specified in the strategy of each of the teams. Based on the result of the comparison, the next goal of the strategy is defined, considering the alternatives presented by the team strategy (Figure 1).

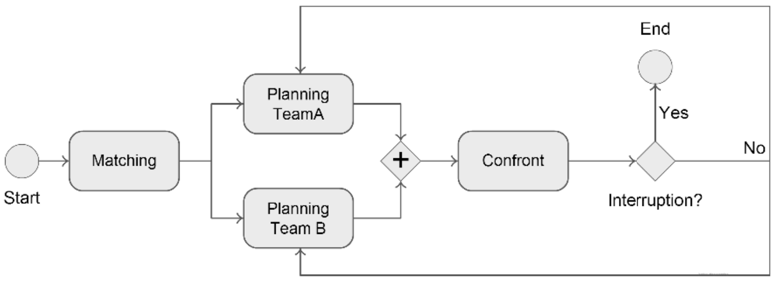

**Fig. 1.** Match dynamics, where: circles represent the start and the end of the flux; rectangles represent processes; diamonds with a cross and empty diamonds represent, respectively, a flux integration and a Boolean test; arrows indicate the direction of the control signal.

## 2.2 System design

The framework of the computational system was divided into two separate modules, the team strategy designer (TSD) and the match simulator. The TSD module allows an user to specify a team's strategy, according to the formalism of the strategy model (see Section 2.1), and creates a file to represent it. The designed strategies for two teams can also be loaded into the match simulator to create a virtual match. The simulation cycle is composed of 3 main processes, the collective planning, the individual interference and a confront. A confront is simulated until the calculated plans cease to be valid, for instance, if a crucial action fails or ends, triggering another round of planning.

The high level collective planner uses the team strategy and the current match state as input. The objective of the planner is to find a path in the team strategy that is both applicable in the current situation and beneficial to the team if successfully navigated. The plan is then modified on an individual scale to introduce automatic individual behaviours (e.g., players can adjust their speed of displacement based on level of fatigue). The planned actions are then simulated until the next round of planning becomes necessary. This cycle is illustrated in Figure 2.

The framework of the computational system can be customized for different team sports. In the present work, the applications were made in the context of basketball.

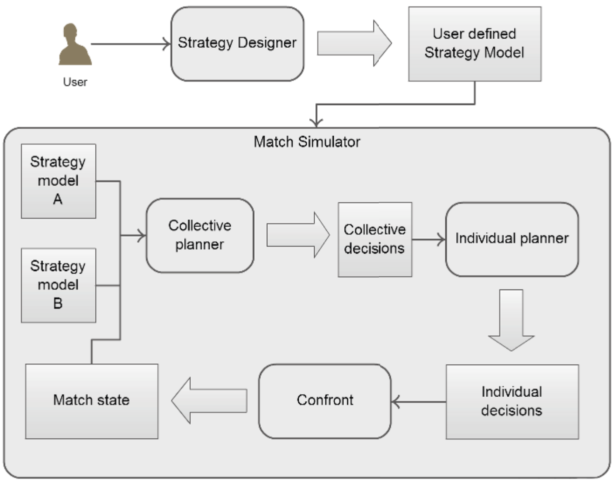

**Fig. 2.** System overview and match simulator main cycle

## 3 Results

### 3.1 Simulation input

The TSD generates a XML file output with the complete description of the specified strategy. Also in the TSD environment, the XML file is represented through a graph (Figure 3A), where the nodes are strategy states (Figures 3B and 3D) and the edges are states transitions (Figure 3C). Then, it provides immediate visualization of the general structure of the designed strategy. In a state, the TSD supports the specification of the players label, players' areas (i.e., equivalence classes for positioning), and ball possession (a small orange circle represents the ball). In the case of the defensive players, their body rotations are also considered due to its relevance for defensive displacements (Figures 3B and 3D). For the purpose of the simulation, two previously specified strategies should be available.

### 3.2 Planning, Matching and Simulation

In the TSD, the user can describe complex strategies in a small state space due to its use of equivalence classes. This allows the description of many similar

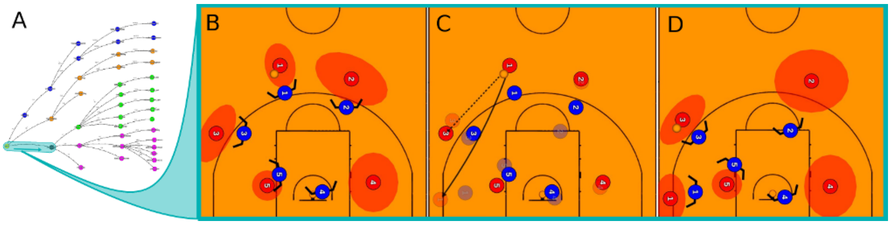

**Fig. 3.** Top-down visualization of a team strategy: 3A - graph of a complete strategy; 3B - previous state; 3C - states transition; 3D - subsequent state. Outer orange ellipses indicate equivalence classes for positioning.

situations in a single strategy state and greatly reduces the strategy graph explored when a plan is calculated. Figure 4 presents an integrative perspective of planning, matching and simulation.

The actual planning problem can be solved using traditional planning algorithms such as the Real-time dynamic programming (RTDP)[1] with one major distinction: the state representation. The planning problem is defined as follows:

1. A state space $S$: the nodes of the strategy graph;
2. An initial state $S_0 \in S$: the node returned by a matching algorithm;
3. A subset of goal states $G \subseteq S$: the terminal states on the graph. A terminal state is any state where a player is in the position to finish the play (i.e., score);
4. A set of actions $A$, and a transition function $f(s, a) \rightarrow S$, with $s \in S$ and $a \in A(s)$: the edges of the graph;
5. A function to assign cost to the actions $c(a, s) \in R^+$. This function represents the risk of a transition and is currently assigned by the user when generating the strategy graph.

A new matching layer must be added to the classical planning algorithms (RTDP) to account for the gap between a match state and the states available in the team strategy. The new layer assigns a match state to the strategy state that best represents it. This assignment can be a perfect fit or an approximation.

A metric to compare a match state to a strategy state was devised to allow the matching layer to function regardless of the match state. Thus, the layer will always return a matched strategy state, even if there is no perfect fit (Figure 5). The metric currently in use is the sum of the quadratic distances between all players in the match state and their counterparts in the strategy state. Since many allocations of players can be used, the one that minimizes the sum is chosen.

The metric considers the distance between a match state player and his counterpart in the strategy state to be zero when the match player is inside the equivalence class for positioning of his counterpart (e.g., players A, C and E in Figure 5). Otherwise the distance considered is the squared Euclidean distance between his position and the closest point to it in the equivalence class for positioning (e.g., players D and B in Figure 5).

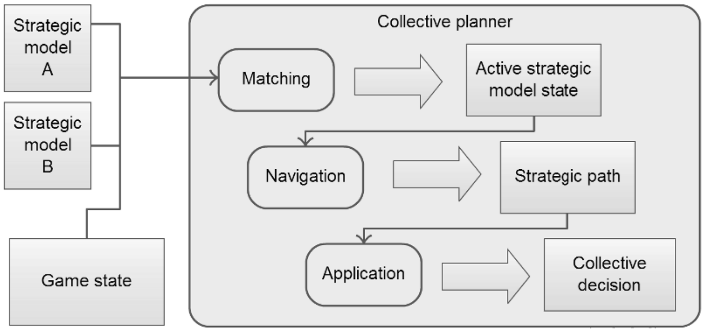

**Fig. 4.** The collective planner execution: Matching provides an initial state; Navigation implements a classical planner; Application adapts the plan to the current match state.

The correct strategy state is then calculated by locating the state which minimizes the sum of the metrics for all players. This calculation also yields the allocation of each player (i.e., his strategy state counterpart).

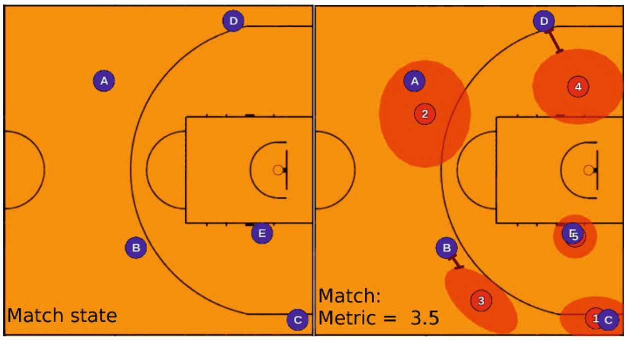

**Fig. 5.** A match state (left), with players assigned in blue, and its counterpart in the strategy (right), with players assigned in orange. The metric 3.5 is the sum of the square distances between players A to E positioning in the match state and their counterparts' equivalence classes for positioning in the strategy state.

The planner extracts a high-level course of action (i.e., a path on the strategy model) using the matched state as an initial node. This is an approximation of the ideal solution, which means it is compatible but not an exact fit for the current match state.

An application process is used to adapt the plan to a specific match state by setting parameters to the actions to be executed. These actions are simple behaviours used by the simulation to execute a plan. An action describes a behaviour (e.g., run, pass, throw, steal) that can be parametrized to be performed exactly as anticipated by the plan.

Once a plan is calculated and adapted, the simulator creates, assigns and executes actions for each player. Some actions have a probabilistic result that can be randomized during the execution (e.g., a score attempt, pass). If one of these action fails, the plan is discarded and another round of planning is triggered. Figure 6 illustrates the simulation environment.

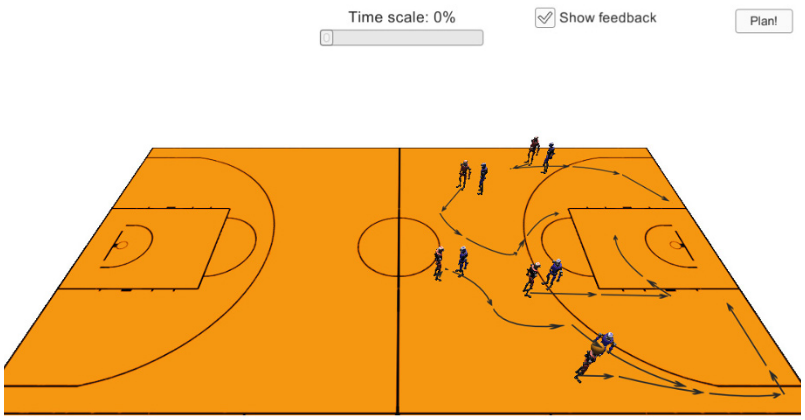

**Fig. 6.** Simulation environment: offensive team (black), defensive team (blue), arrows on the court indicate players' trajectories.

## 4 Conclusion

The main contribution of this study was to develop a computational system that enables the design of team strategies and uses these strategies as input for simulating the match in team sports. This system may support the enhancement of the knowledge about team sports through testing possible features of efficient strategies design and respective consequences for real matches through simulation procedures. This framework may be helpful for coaches and players to improve their capacity to plan the collective performance, interpret the opposition constraints and act more efficiently. Both strategies design and simulation procedures have been conducted in the context of basketball. Nonetheless, the coverage of the model that provides theoretical support for the system enables its customization to a large range of team sports.

These results may have positive impact in other scientific areas as well. For instance, it is possible that the planning and matching structure based on a model for representing strategies and the match dynamics contribute to the design of video-games with a greater realism than the observed in the present games, approximating them to real match simulators. Additionally, it may contribute to solve problems in some branches of artificial intelligence, such as modelling of the collective action in competitive environments, for instance in robot soccer tournaments.

**Acknowledgments**: LL was supported by Fundacão de Amparo à Pesquisa do Distrito Federal. JB is supported by Conselho Nacional de Pesquisa, grant: 306442/2011-6.

## References

1. Bonet, B., Geffner, H.: Labeled RTDP: Improving the convergence of real-time dynamic programming. In: Proceedings of the Thirtheenth International Conference on Automated Planning and Scheduling (2003)
2. Chan, B., Denzinger, J., Gates, D., Loose, K.: Evolutionary behavior testing of commercial computer games. In: Proceedings of the Congress on Evolutionary Computation (2004)
3. Grehaigne, J., Godbout, P., Bouthier, D.: The foundations of tactics and strategy in team sports. Journal of Teaching in Physical Education 18, 159–174 (1999)
4. Lamas, L., Barrera, J., Otranto, G., Ugrinowitsch, C.: Invasionteamsports: strategy and match modeling. International Journal of Performance Analysis in Sports 14, 307–329 (2014)
5. Lamas, L., Santana, F., Otranto, G., Barrera, J.: Inference of team sports strategies based on a library of states: application to basketball. In: Proceedings of the 2014 KDD Workshop on Large-Scale Sports Analytics (2014)
6. Wloch, K., Bentley, P.: Optimising the performance of a formula one car using a genetic algorithm. In: Proceedings of the 8th International Conference of Parallel Problem Solving from Nature (2004)
7. Xiao, G., Southey, F., Holte, R.: Software testing by active learning for commercial games. In: Proceedings of the Congress of the American Association for Artificial Intelligence (2005)