Adapting the Markov Chain based Algorithm M-DBScan to Detect Opponents' Strategy Changes in the Dynamic Scenario of a StarCraft Player Agent

Eldane Vieira Júnior, Rita Maria Silva Julia and Elaine Ribeiro Faria
Computer Department, Federal University of Uberlândia, Uberlândia, Minas Gerais, Brazil

Keywords: Novelty Detection, Markov Chain, StarCraft, M-DBScan Algorithm.

Abstract: Digital games represent an appropriate test scenario to investigate the agents' ability to detect changes in the behavior of other agents trying to prevent them from fulfilling their objectives. Such ability allows the agents to adapt their decision-making process to adequately deal with those changes. The Markov Chain based algorithm M-DBScan is a successful tool conceived to detect novelties in data stream scenarios. Such algorithm has been validated in artificially controlled situations in games in which a single set of features and a single Markov Chain are sufficient to represent the data and to detect the occurrence of novelties, which usually is not enough to make the agents able to adequately perceive the environment changes in real game situations. The main contribution of the present work is then to investigate how to improve the use of M-DBScan as a tool for detecting behavior changes in the context of real and dynamic StarCraft games by using distinct sets of features and Markov Chains to represent the peculiarities of relevant game stages. Further, distinctly from the existing researches, here M-DBScan is validated in situations in which the timestamp, between successive novelties, is not constant.

1 INTRODUCTION

Intelligent agents have been increasingly used to solve a great variety of practical problems, such as disease diagnosis, medical image analysis, natural language processing, robot navigation, games, Internet of Things (IoT), etc (Gubbi et al., 2013; Norvig and Intelligence, 2002).

In order to cope with such problems, frequently the agents have to face opponents that try to prevent them from succeeding in their tasks. In these situations, it becomes crucial that the agents map their opponents' behavior, being aware of eventual occurrence of strategic changes. In general, these changes can be analyzed through the novelties that the opponents' actions provoke in the environment.

The attempt to identify novelties in data streams has to cope with difficulties such as the followings: the data obtained from the data streams is potentially infinite; the lack of control in the data arrival order; the data already processed must be discarded; the data can arrive in very high frequency; the data can go through changes (Krawczyk et al., 2017; Babcock et al., 2002).

The learning process of an agent in a data stream scenario must be incremental since it must reflect the most recent changes observed. Then, the set of features used to represent the data must be defined to allow the perception of the agent over any data alterations.

In Real-time Strategy (RTS) games, for example, the detection of the alterations in the opponent's behavior is essential to adapt the agent decision-making to such changes. In these cases, the alterations may be perceived through the analysis of novelties detected in the data stream(Yannakakis and Maragoudakis, 2005; Vallim et al., 2013).

The Markov Chain based proceeding Micro-Clustering DBScan (M-DBScan) has excelled in the context of researches regarding the detection of novelties in data stream scenarios (Vallim et al., 2013). In such researches, M-DBScan has been validated using artificially controlled situations related to some games, without compromising the accuracy of the algorithm. The datum is represented by a unique set of features and only one Markov Chain (MC) was used to detect the occurrence of novelties (Vallim et al., 2013; Vieira et al., 2019). Further, in the situations investigated by the existing researches with M-DBScan, the timestamp between every successive

Vieira Júnior, E., Julia, R. and Faria, E.
Adapting the Markov Chain based Algorithm M-DBScan to Detect Opponents' Strategy Changes in the Dynamic Scenario of a StarCraft Player Agent.
DOI: 10.5220/0008985802140223
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 2*, pages 214-223
ISBN: 978-989-758-395-7; ISSN: 2184-433X

novelty is constant. However, in many practical problems treated by Artificial Intelligence (AI), the agents deal with extremely dynamic environments, whose intense changes require distinct sets of features to represent them over time. It happens, for example, in many RTS games, in which there is a great variety of game scenarios in the course of a real match. In these cases, it is practically impossible that the agents can adequately perceive the novelties occurred in the environment through a unique set of features.

Considering the arguments aforementioned, the main contribution of the present work is to investigate how to use M-DBScan to detect novelties in data stream scenarios in which the environment undergo changes associated with the moment of the process, requiring distinct sets of features to represent the peculiarities of the data over time. The study here, using the StarCraft RTS game as a case study, also differs from other existing works for taking into consideration situations in which the timestamp between successive novelties is not constant. More specifically, the purpose here is to use M-DBScan to capture novelties in the successive game scenarios that appear throughout a real match and, based on such novelties, to investigate eventual changes in the opponents' behavior. The hypothesis of this approach is the following: as the opponent's decision-making directly impacts the game scenario, its profile can be investigated through continuous analysis of the dynamic environment over the successive game stages.

To cope with such purpose, the authors defined the following game stages as being relevant to pursue their objectives (based on the game characteristics): the *Beginning Stage of the Battle* (BSB); the *Middle Stage of the Battle* (MSB); finally, the *Final Stage of the Battle* (FSB). Further, in order to represent the game scenarios, the authors used sub-sets of features extracted from a universe set of 16 features (related to the units that compose the StarCraft armies), which is able to appropriately represent the specificities of the three stages, being such selection based on their own experience with the game.

To evaluate the gains obtained from the approach proposed herein, in which M-DBScan takes into account and fits the specificities of every stage of the environment, this work calculates and compares the results produced by such algorithms through two distinct test scenarios: first, MDBScan is used in real game situations of StarCraft matches in the same way as it had been tested in the related works, that is, in the course of a complete StarCraft game, a unique set of features is used to represent the game scenarios and a unique MC is constructed to detect the changes in the opponents' behavior; second, MDBScan is used according to the approach proposed herein, that is, over real StarCraft contests, a distinct set of features and a distinct MC are used to represent every game stage. In this case, some set of features corresponds to a sub-set of the 16 features aforementioned. The results confirm the advances produced by the approach proposed in the present work.

The main motivations for choosing StarCraft as a case study are the followings: 1) Game problems frequently involve challenges that are very similar to those found in real-world problems, including situations in which opponents try to minimize the success of agents that try to cope with such problems (Neto and Julia, 2018); 2) The StarCraft game scenarios are very dynamic over time, requiring distinct sets of features to perceive their peculiarities in different game stages (for example, the feature *Dead enemy unit*, in spite of being essential to represent novelties in endgame scenarios, FSB stage, it is irrelevant at the beginning of the game, BSB stage); 3) In StarCraft games, any novelty must be identified in a short period of time, so as to allow that any action in response to a detected novelty is taken within the time period in which the current set of features is still adequate to represent the game scenario.

This paper is organized as following: in section 2, the theoretical foundations are presented; section 3 resumes the related works; section 4 describes how the M-DBScan based novelty detection process can be used to cope with the dynamic environment of StarCraft; section 5 presents the experiments and the results; finally, section 6 summarizes the conclusion and points out some possible future works.

2 THEORETICAL FOUNDATIONS

This section presents a short description of some concepts and techniques used in the development of the work presented in this paper.

2.1 Player Modeling

Player modeling is defined as the study of computational models of game players. It is focused on the way that a player interacts with a game, which may include the detection and prediction of its features, and also how to model and express these features (Yannakakis et al., 2013).

Player modeling is used in human-computer interaction, and it has also been used in the context where the player is an agent, and in this case, player modeling can be used to improve these agents as described in (Weber and Mateas, 2009).

The present paper aims at identifying changes in the playing style of the adversary in different moments of the game. A model is created using the algorithm M-DBScan and it will be adapted over time, in order to reflect recent information. The algorithm M-DBScan was improved in this study by the use of different MCs for different moments of the game.

2.2 Novelty Detection in Data Stream

Novelty detection is an efficient process to identify an instance of data that differs in a significant manner from some already known concepts. This research area has received a great deal of attention in machine learning and data mining (Faria et al., 2016).

Novelty detection has been used in researches related to the context of a data stream, which can be defined as a sequence of data instances that arrives continuously, with specific characteristics as the following (Gama and Gaber, 2007): the data arrives online; the order that data arrives can not be controlled; the size of a data stream is unlimited; a data already processed is usually discarded, once there are limited resources as memory.

2.3 M-DBScan

The algorithm M-DBScan is an approach to detect changes in data streams in which the data is not labeled (Vallim et al., 2013). This algorithm has an incremental clustering process, based on DenStream (Cao et al., 2006), which is a clustering technique applied in a data stream. The clustering process is followed by a mechanism to detect novelties, which can be used to identify behavior change (Vallim, 2013).

The clustering step proposed by M-DBSCAN is composed of two phases, online and offline. While the online phase keeps a statistical summary of the dataset, based on micro-clusters, which are used to maintain compact information as radius, center, and weight, considering the data that arrives from the stream. The micro-clusters can be classified as potential-micro-cluster (p-micro-cluster), outlier-micro-cluster (o-micro-cluster) and core-micro-cluster (c-micro-cluster), and each of them has its own requirements, as described in (Cao et al., 2006). The offline phase generates clusters based on the result produced by the online phase, and it uses the concepts of density and connectivity (Cao et al., 2006).

Once the offline phase has been concluded, the clusters generated in the offline phase are used in a novelty detection module, that aided by a sliding window, is capable to indicate the occurrence of changes. The novelty detection module uses MC, and in this process, each state in the MC represents a group produced in the offline phase, and the transitions between the states follow the dynamics observed in the arrivals of new data and how the assignment of this data to a micro-cluster occurred.

2.3.1 Novelty Detection Module

In order to detect novelties, the M-DBScan algorithm uses two entropy measures, named spatial entropy and temporal entropy. The dynamics of how to calculate the entropies, and how to update them follows the specifications presented in (Vieira et al., 2019; Vallim et al., 2013).

Since the states in the MC are the representation of groups produced in the offline clustering step of M-DBScan, the probabilities in the MC transitions are updated every time that a data arrives from the stream. A weighting factor η_t is used to control the intensity of each probability update. The sum of probabilities linked to transitions with the same origin state has 1 as its maximum value when a transition is no longer activated, it will suffer a value deterioration.

In order to identify a novelty, at least one of the entropies must surpass its pre-defined threshold. The algorithm M-DBScan requires that a minimum amount of novelty must be identified in an interval of time represented by a sliding window with size k, to characterize a behavior or strategy change. Every novelty occurrence is registered in the sliding window, where it is necessary to verify if the amount of registered novelties has reached the minimum amount necessary to declare, decisively, a change. A novelty that occurred in a certain moment can disappear from the window if it has slided k times, considering as the initiation, the moment that this novelty was registered. However, if a behavior change was detected, from this point on every novelty will be ignored for a period of time equal to k. This process is used to discard novelties linked to the same change that has already been identified (Vallim et al., 2013).

3 RELATED WORKS

Among the related works, some applied machine learning techniques to RTS games, to optimize strategic tasks, these are those tasks that carry an influence throughout the game, such as the order of constructions and what to construct. A study presented in (Justesen and Risi, 2017), for example, describes a way of dealing with the task that decides which structures must be constructed in the game StarCraft, using the algorithm *Continual Online Evolutionary Planning* (COEP). There are related works that try to improve battle skills of RTS automatic players, as in the study presented in (Shao et al., 2017), which describes the development of a decision module for the game StarCraft, based on a neural network with reinforcement learning, that can point out an action for each living unit. In (Stanescu and Čertický, 2016) another study is presented that tries to improve an automatic player specialized in battles, this study uses a prediction method to indicate the most likely combination of units produced by the enemy in an RTS game like StarCraft. In the work described in (Niel et al., 2018) has developed a simple custom RTS game, in which the main goal of the players is to defend their bases and destroy the enemy's base. In this work was used reinforcement learning (RL) combined with a multi-layer perceptron (MLP) to determine how the agent will perform the tasks in the game. The Q-learning was tested against Monte Carlo learning as reinforcement learning algorithms, and two different methods for assigning rewards to agents were tested, the individual and the sharing reward. The results showed that the combination of Q-learning and individual rewards presented the best win-rate. The works presented in (Justesen and Risi, 2017; Shao et al., 2017; Stanescu and Čertický, 2016; Niel et al., 2018) do not deal with novelty identification and there is no attempt to represent different moments of the game using appropriate features, to improve the strategy detection, which are great differences to the present paper.

The work in (Álvarez Caballero et al., 2017) used supervised learning techniques as MLP, Random Forest and Logistic Regression, over a big amount of data obtained from StarCraft replays. The objective of this work is to predict the winner of a match in the game StarCraft. This work proved that the use of appropriate StarCraft features will increase the accuracy of this prediction, which can occur at a very satisfying level after 10 minutes of gameplay. In (Synnaeve et al., 2012), a method is presented for discovering tactics and strategies in the game StarCraft. A created dataset was analyzed by the technique *Gaussian Mixture Model*, to identify every single cluster (army formation) that will indicate the details of each army composition, and these details go on to provide a way of finding the best army formation. In (Weber and Mateas, 2009), a study is put forward that tries to create a model that represents a general enemy based on several others, in order to be used in the prediction of strategic action. In (Weber and Mateas, 2009) was used techniques for classification as *K-Nearest Neighbor*, *Non-Nested generalized exemplars* and *Additive Logistic Regression*. The works described in (Álvarez Caballero et al., 2017; Synnaeve et al., 2012; Weber and Mateas, 2009) show significant differences to the present paper, since they do not select attributes for a better representation of the game phases, and they apply a supervised learning process, which is not the case in the present paper.

In (Vallim, 2013), the authors use M-DBScan to detect player behavior changes in RTS game, in *first person shooter* game and in artificial datasets, showing the diversity and success in the usage of this technique. However, such work differs from the approach presented herein for the following reasons: it deals with artificially controlled situations in which the timestamp between every pair of successive novelties is constant; and it only copes with game scenarios for which the dynamics can be appropriately represented by a unique set of features, using just one MC to investigate eventual changes. In the study presented in (Vieira et al., 2019) the algorithm M-DBScan was used with data from the game StarCraft to show that the technique works in controlled test scenarios that, like (Vallim, 2013), just used one MC.

4 DYNAMIC NOVELTY DETECTION IN StarCraft

The main contribution of the approach proposed in this study is to adapt the use of the algorithm M-DBScan to improve its capacity in detecting novelties in the game StarCraft. Such novelties concern significant alterations on the game scenarios that eventually may point out an opponent's strategic change. As presented in this section, the basis of this approach consists on using distinct and appropriate sets of features to represent the game scenarios according to the peculiarities of the following game stages: *Beginning Stage of the Battle* (BSB), in which the adversaries begin the conflict; MSB *Middle Stage of the Battle* (MSB), in which deaths begin to occur in one of the armies involved in the battle; and, finally, FSB *Final Stage of the Battle* (FSB), which can be characterized by the significant reduction in the number of battle units that compose the armies involved in the contest. In order to represent the game scenarios, the authors use sub-sets of features extracted from a universe set of 16 features, being such selection based on their own experience with the game. In this way, during the process of detecting eventual changes in the opponent's strategy, the present approach has to build 3 distinct MCs, one for each game stage. This approach also differs from other existing works for taking into consideration situations in which the timestamp between successive novelties is not constant.

The Figure 1 illustrates the process of assigning the data to the correct MC.

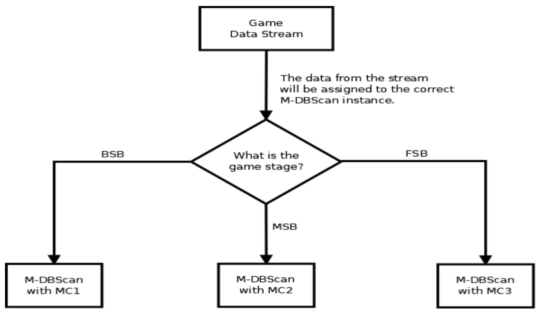

Figure 1: Selecting the correct instance of MC.

4.1 StarCraft: General Description

StarCraft is an RTS game released by *Blizzard Entertainment* in 1998 (Entertainment, 1998), being used specifically the expansion StarCraft: BroodWar in this work. This engine is based on rules that define the dynamics in the game to assign tasks for units as combat, locomotion and how to gather resources from the environment. In the game, players can control one of three races: *Terran*, *Protons* and *Zerg*. Each race has its peculiarities related to the aspects of weaknesses and strengths. There are differences among the units, some are appropriate on building structures and collecting resources from the environment, others are more appropriate for combat. However, the army must be analyzed considering all the units that compose it, which makes it possible to have an overview concerning how good that army is, considering the effective level of damage that it can cause or its level of defense.

In the game StarCraft, some challenges emphasize it as a case study with characteristics of a data stream scenario. For example, the data is obtained in a very high frequency, and the uncertainty about how long the game will last makes it impossible to guess how much data can be obtained from the stream. The area in the map that has not been explored is kept hidden, making it impossible to know what exists in this unknown area of the map, which makes applying strategies to the game difficult.

4.2 Feature Extraction

In this work, sub-sets of a set composed of 16 features are used to describe the game stages. In particular, the features used in the process of novelty detection can make the process of identification easier. As such, in this work, features were empirically selected for different moments of the game, to better represent the specificities of distinct game stages that are strategically relevant.

Each instance of the data set, created by the authors, corresponds to an accumulated perception (represented by the aforementioned features) of the last 15 seconds of the game. These features can be summarized as three groups: i) hit point (HP), which represents the amount of life of each unit. This group of features is used to represent the intensiveness and effectiveness of the agent and opponent attack. ii) army power, which represents the damage/defense that the enemy army can cause; iii) amount of damage, represented by the number of dead units of the enemy and score of damage to the structures. Concerning the types of the units, we can consider that: i) Unit 1 has low attack power; ii) Unit 2 has medium attack power; and iii) Unit 3 has high attack power. The 16 features are described as follows:

- Feature 1 (Dead enemy unit): provides the number of deaths in the enemy army.
- Feature 2 (Low HP Agent Unit 1): provides the number of units type 1 in the agent army with low hit points.
- Feature 3 (Low HP Agent Unit 2): provides the number of units type 2 in the agent army with low hit points.
- Feature 4 (Low HP Agent Unit 3): provides the number of units type 3 in the agent army with low hit points.
- Feature 5 (High HP Agent Unit 1): provides the number of units type 1 in the agent army with high hit points.
- Feature 6 (High HP Agent Unit 2): provides the number of units type 2 in the agent army with high hit points.
- Feature 7 (High HP Agent Unit 3): provides the number of units type 3 in the agent army with high hit points.
- Feature 8 (Attack power): represents the damage that the enemy army can cause.
- Feature 9 (Defense power): represents the defense power of the enemy army.
- Feature 10 (Damage in construction): provides a score for the damage that the enemy caused to structures.
- Feature 11 (Low HP Enemy Unit 1): provides the number of units type 1 in the enemy army with low hit points.

- Feature 12 (Low HP Enemy Unit 2): provides the number of units type 2 in the enemy army with low hit points.
- Feature 13 (Low HP Enemy Unit 3): provides the number of units type 3 in the enemy army with low hit points.
- Feature 14 (High HP Enemy Unit 1): provides the number of units type 1 in the enemy army with high hit points.
- Feature 15 (High HP Enemy Unit 2): provides the number of units type 2 in the enemy army with high hit points.
- Feature 16 (High HP Enemy Unit 3): provides the number of units type 3 in the enemy army with high hit points.

The subset of features selected to represent the BSB game stage, named as V_1, is composed of the following features: 5, 6, 7, 8, 9, 14, 15 and 16; the MSB moment is represented by the universe set of 16 features, named as V; and, finally, the FSB game stage is represented by the subset whose elements are the features 2, 3, 4, 5, 6, 7, 11, 12, 13, 14, 15 and 16, named as V_2. All the features were empirically selected considering what is reasonable at each moment, for example, units with low HP will not exist at the beginning of the battles.

4.3 M-DBScan

The M-DBScan algorithm has been used to detect changes in the opponent's strategy in two distinct manners: the game scenarios were represented by the universe set V; and subsets of V was used represent different game stage.

The algorithm M-DBScan is illustrated in the flowchart presented in Figure 2 (Vieira et al., 2019). Initially, the algorithm M-DBScan checks if there is a data sample in the stream (arrow #1): if so, the algorithm retrieves the datum (arrow #2) that will be presented as an input to the online part of the algorithm M-DBScan (arrow #3), which will recalculate all the concerning parameters related to the assigned micro-cluster for this datum. In sequence, a decay factor is applied to all remaining micro-clusters (arrow #4). Step (arrow #5) checks if it is time to perform a rating verification: if so, this verification is executed (arrow #6), transforming into o-micro-clusters all p-micro-clusters that do not satisfy anymore the p-micro-cluster constraints (Cao et al., 2006) (due to the use of the decay factor). In the same way, in this rating verification, all o-micro-clusters that do not present anymore the requirement for an o-micro-cluster will be deleted (arrow #7). Next, the offline part of M-DBScan is run (arrow #8). Otherwise, if in step (arrow #5) it is not time to perform the rating verification, the offline part of M-DBScan will be directly run (arrow #9). Then, the entropy value (temporal entropy or spatial entropy) and its corresponding threshold are calculated (arrow #10 and #11). If the entropy exceeds its threshold, then a novelty is detected and registered in the sliding window (arrow #13). Next, a module for detecting change is used (arrow #14), which verifies if the minimum amount of novelties has been reached. The module for detecting change is also responsible for controlling the sliding window. Next, upon detecting a novelty or not, the algorithm tries to retrieve a new data sample from the stream (arrow #15 and #16). The execution of M-DBScan ends when there is no more data from the stream (arrow #17).

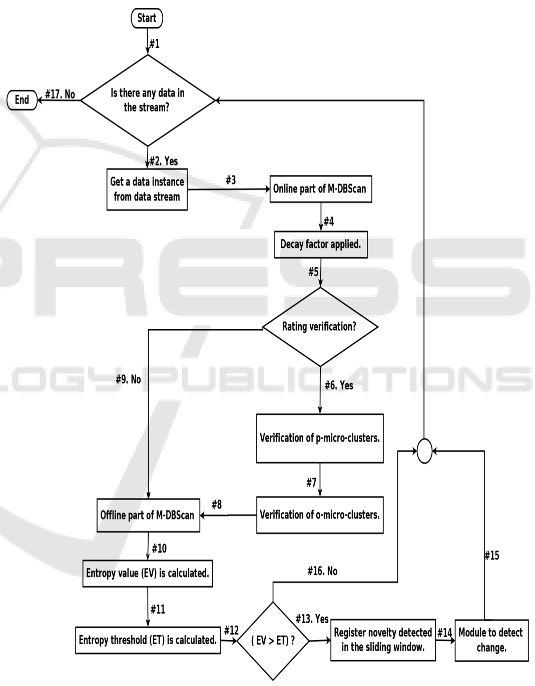
Start
#1
Is there any data in the stream?
#17. No
End
#2. Yes
Get a data instance from data stream
#3
Online part of M-DBScan
#4
Decay factor applied.
#5
Rating verification?
#9. No
#6. Yes
Verification of p-micro-clusters.
#7
Verification of o-micro-clusters.
#8
Offline part of M-DBScan
#10
Entropy value (EV) is calculated.
#11
Entropy threshold (ET) is calculated.
#12
(EV > ET) ?
#16. No
#13. Yes
Register novelty detected in the sliding window.
#14
Module to detect change.
#15

Figure 2: M-DBScan flowchart.

5 EXPERIMENTS AND RESULTS

The conducted experiments aim to show the impact of the use of a distinct set of features and distinct MCs to represent every game stage in the M-DBSCAN algorithm, in contrast to the use of a unique set of features and unique MC.

Different datasets were created to represent different game scenarios, and among them, there are those, in which the interval of time between novelties are not constant.

Table 1: Description of Datasets.

Dataset	Novelty ID	Timestamp	Game Moment	Strategy
Dataset 1	1	500	BSB	D1S1
	2	1000	MSB	D1S2
	3	1500	FSB	D1S3
Dataset 2	1	500	BSB	D2S1
	2	1000	MSB	D2S2
	3	1500	MSB	D2S3
	4	2000	MSB	D2S4
	5	2500	MSB	D2S5
	6	3000	MSB	D2S6
	7	3500	FSB	D2S7
Dataset 3	1	149	BSB	D3S1
	2	468	MSB	D3S2
	3	528	MSB	D3S3
	4	626	MSB	D3S4
	5	1019	MSB	D3S5
	6	1196	MSB	D3S6
	7	1901	MSB	D3S7
	8	3413	FSB	D3S8
Dataset 4	1	206	BSB	D4S1
	2	676	MSB	D4S2
	3	1546	MSB	D4S3
	4	1774	MSB	D4S4
	5	2734	MSB	D4S5
	6	5605	MSB	D4S6
	7	6067	MSB	D4S7
	8	7648	MSB	D4S8
	9	7772	FSB	D4S9

5.1 Datasets

The tools used to generate the dataset are the engine UAlbertaBot (Churchill and Buro, 2011; Churchill et al., 2012), a playing bot that won the 2013 AIIDE StarCraft AI Competition, and the *Brood War Application Programming Interface* (BWAPI), a way for programmers to interact with and control the full game StarCraft.

For the experiments, four datasets were generated[1]. These datasets were collected from real matches in the game StarCraft. Each data instance represents 15 seconds of the game playing and contains 16 features. A novelty is characterized when a change of opponent's strategy occurs.

The datasets are described in Table 1. In this table, the columns *Novelty ID* is a sequential number to identify each novelty of the dataset; *Timestamp* indicates how many data samples arrived from the beginning of the stream until the novelty occurrence; *Game Moment* indicates in which game moment the novelty occurred (BSB, MSB, or FSB); *Strategy* indicates the code of a strategy used in the novelty. Each strategy used is based on one of the following: the increase of attack power; the increase of defense power; army improvement with stronger units; defensive endurance for the endgame.

Although the datasets are composed of 16 features, a subset containing 8 and 12 features, here named V_1 and V_2, are used in the BSB and FSB moments of the game, respectively. In the MSB moment, the 16 features (V) are used.

5.2 Settings

The parameters used in the experiments were selected based on empirical tests. The parameters for the clustering process are: μ, which is a minimum amount of points in a micro-cluster, ε is a radius limit for micro-cluster, β is an outlier threshold and λ is a decay factor. The parameter used in the entropy process are: the weighting factor η_t, which is used to control the intensity of each probability update; the weighting factors γ and δ used in the process of threshold cal-

[1]https://drive.google.com/open?id=1nqndc7E3ru1sEb_r3Q7NLmySCiepbOgu

Table 2: Results of the experiments using 1 MC and 3 MCs.

Amount of MCs	**Dataset**	**Temporal Entropy**				**Spatial Entropy**			
		TP	**FP**	**FN**	**F1**	**TP**	**FP**	**FN**	**F1**
1 MC	**Dataset 1 (3 changes)**	2	1	1	0.666	3	1	0	0.857
	Dataset 2 (7 changes)	3	3	4	0.461	4	1	3	0.666
	Dataset 3 (8 changes)	3	4	5	0.4	3	2	5	0.461
	Dataset 4 (9 changes)	4	4	5	0.47	5	2	4	0.625
3 MCs	**Dataset 1 (3 changes)**	3	1	0	0.857	3	0	0	1
	Dataset 2 (7 changes)	4	3	3	0.571	5	1	2	0.769
	Dataset 3 (8 changes)	4	2	4	0.571	5	2	3	0.666
	Dataset 4 (9 changes)	6	4	3	0.631	6	1	3	0.75

culation; and the constant parameter θ, which is the number of standard deviations used to define a normal distribution for entropy values.

The experiments were run using the following values of parameters:

- The parameters used in the clustering process: μ is 9; β is 0.1; ε is 20; λ is 0.03.
- The parameters used in the experiments for temporal entropy: minimum amount of novelties is 2; size of sliding window is 20; η_t is 0.05; γ is 0.05; δ is 0.03; θ is 3.
- The parameters used in the experiments for spatial entropy: minimum amount of novelties is 2; size of sliding window is 20; η_s is 0.05; γ is 0.09;δ is 0.002; θ is 3.

The measures false positives (FP), false negatives (FN), true positives (TP), and F1 are used to evaluate the experiments. These measures are presented for each entropy strategy (temporal and spatial).

Table 2 show the results of the proposed approach in contrast to the original M-DBSCAN, considering the four datasets and the temporal and spatial entropies. Table 2 presents the results of the M-DBScan using only 1 MC and the results of the M-DBScan using 3 different MCs, each one built from a different set of attributes for a different moment of the game (V for MSB, V_1 for BSB and V_2 for FSB). Based on the results, it is possible to notice that the best results for all measures were achieved using the M-DBSCAN with 3 different MCs. Also, the spatial entropy achieved the best results in both scenarios.

Every false positive detected in the experiments occurred at the moment MSB. The MSB phase is a moment of the game when the player can perform actions with some effect over the dynamics of the MC, but not necessarily sufficient to characterize a novelty in the strategy.

Considering that the spatial entropy presented better results than the temporal entropy, the Figure 3 presents a comparison between the real occurrence of changes, and its corresponding detections by the algorithms using the spatial entropy. In this figure, it is pointed out the exact moment that a behavior novelty occurred, and the moment that each approach of M-DBScan identified the novelty or indicated a false positive. In the graphs in this figure, there are highlighted lines indicating the separation of the BSB, MSB and FSB phases, which always occur in this order. It is possible to observe in Figure 3 that the false positives occur when the detection of a novelty had a considerable delay, and when a novelty is pointed out by the algorithm in a moment that none novelty is expected.

Analyzing the Figure 3, all three novelties in dataset 1 were detected by our approach. However, the third novelty, which occurs in the FSB phase, was not detected by the M-DBSCAN with 1 MC. For dataset 2, among seven novelties, four were detected by our approach, and three were detected by M-DBSCAn with 1 MC. Again, a novelty which occurred in the FSB game moment was detected by M-DBSCAN with 3 MCs, but not by M-DBSCAN with 1 MC. For dataset 3, among eight novelties, four of them were identified by M-DBSCAN with 3 MCs and three by M-DBSCAN with 1 MC. Over again, the difference between the approaches is in the detection of novelties in the FSB game moment. Considering the dataset 4 in which occurs nine novelties, six of them were detected by our approach, and only four were detected by M-DBSCAN with 1 MC. For this dataset, the differences between the approaches occurred in the FSB and MSB stages.

The Wilcoxon test was applied over the four datasets, comparing the performance of the M-DBscan using temporal and spatial entropies in both scenarios, first using only one MC to detect the changes, and then, three MCs are used, each for a different moment of the game (BSB, MSB, and FSB). The tool KEEL (Alcalá-Fdez et al., 2011) was used to perform the Wilcoxon test.

The results obtained by the Wilcoxon test for M-

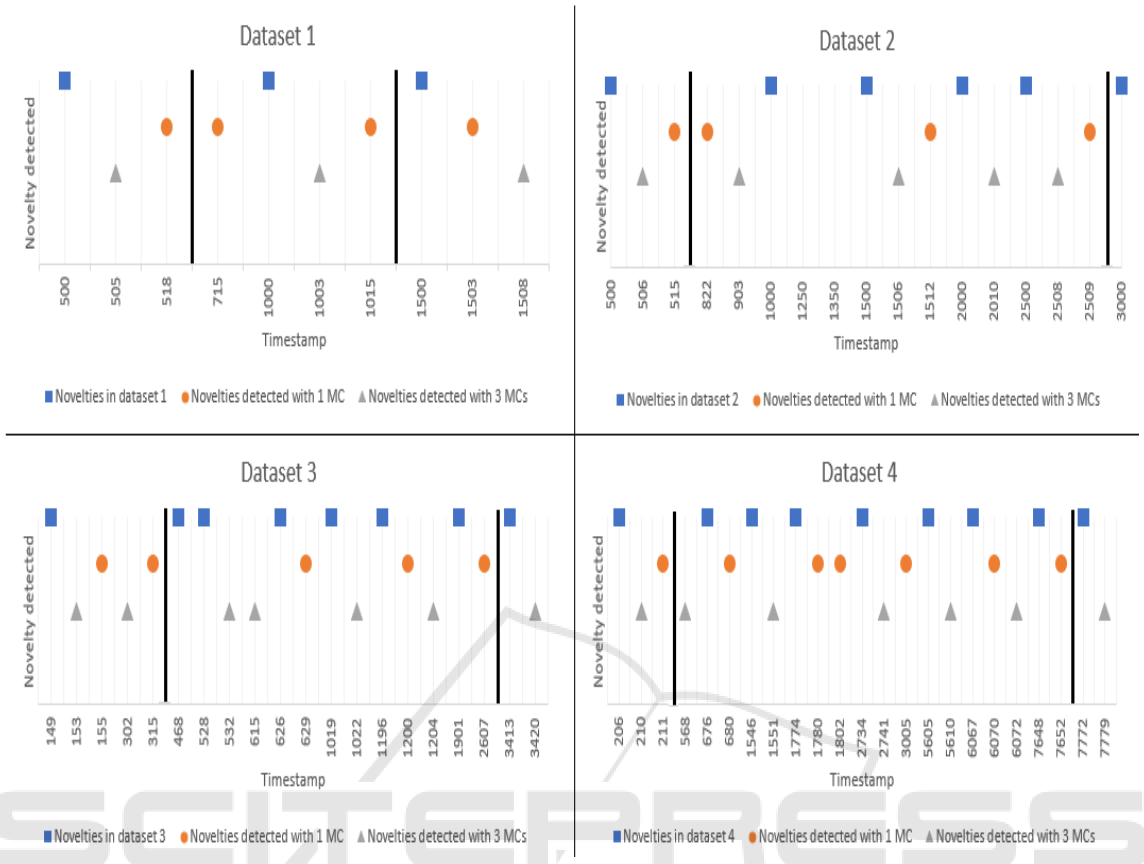

Figure 3: Graphics indicating the moment of a behavior novelty detection using Spatial Entropy.

DBScan with spatial entropy, comparing the approach using three MCs to that using one MC, are R^+ equal to 10 and R^- equal to 0, with an asymptotic p-value equal to 0.04461. For M-DBScan with temporal entropy, also comparing the approach using three MCs to that using one MC, the results obtained by the Wilcoxon test are R^+ equal to 10 and R^- equal to 0, with an asymptotic p-value close to 0.04461. In both situations the null hypothesis can be rejected.

6 CONCLUSION AND FUTURE WORKS

This paper presents an approach to improve the ability of the MC based algorithm M-DBScan to detect behavior changes in the dynamic StarCraft game environment. To cope with such objective, here distinct sets of features and MCs were used to capture the game scenario peculiarities over time. Further, this study also takes into consideration situations in which the timestamp between successive novelties is not constant.

The results confirm the advances produced by the approach proposed herein.

Noteworthy here is that both, spatial and temporal entropy, produced satisfactory results, but the former presented a superior performance.

In future works, the authors intend to use Genetic Algorithms to automate the feature selection process, and to incorporate the module for detecting changes in the opponent's strategy into the decision-making module of StarCraft player agent, allowing this agent to adapt their decision making to the opponent profile.

REFERENCES

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM*

SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 1–16. ACM.

Cao, F., Estert, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM.

Churchill, D. and Buro, M. (2011). Build order optimization in starcraft. In *AIIDE*, pages 14–19.

Churchill, D., Saffidine, A., and Buro, M. (2012). Fast heuristic search for rts game combat scenarios. In *AI-IDE*, pages 112–117.

Entertainment, B. (1998). Starcraft: Brood war. *PC. Level/area: Episode IV, mission*, 2.

Faria, E. R., Gonçalves, I. J., de Carvalho, A. C., and Gama, J. (2016). Novelty detection in data streams. *Artificial Intelligence Review*, 45(2):235–269.

Gama, J. and Gaber, M. M. (2007). *Learning from data streams: processing techniques in sensor networks*. Springer.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660.

Justesen, N. and Risi, S. (2017). Continual online evolutionary planning for in-game build order adaptation in starcraft. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 187–194. ACM.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., and Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156.

Álvarez Caballero, A., Merelo, J. J., Sánchez, P. G., and Fernández-Ares, A. (2017). Early prediction of the winner in starcraft matches. In *Proceedings of the 9th International Joint Conference on Computational Intelligence - IJCCI,*, pages 401–406. INSTICC, SciTePress.

Neto, H. C. and Julia, R. M. S. (2018). Ace-rl-checkers: decision-making adaptability through integration of automatic case elicitation, reinforcement learning, and sequential pattern mining. *Knowledge and Information Systems*, 57(3):603–634.

Niel, R., Krebbers, J., Drugan, M. M., and Wiering, M. A. (2018). Hierarchical reinforcement learning for real-time strategy games. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pages 470–477. INSTICC, SciTePress.

Norvig, P. R. and Intelligence, S. A. (2002). *A modern approach*. Prentice Hall.

Shao, K., Zhu, Y., and Zhao, D. (2017). Cooperative reinforcement learning for multiple units combat in starcraft. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–6. IEEE.

Stanescu, M. and Čertickỳ, M. (2016). Predicting opponent's production in real-time strategy games with answer set programming. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(1):89–94.

Synnaeve, G., Bessiere, P., et al. (2012). A dataset for starcraft ai & an example of armies clustering. In *AIIDE Workshop on AI in Adversarial Real-time games*, volume 2012.

Vallim, R. M., Andrade Filho, J. A., De Mello, R. F., and De Carvalho, A. C. (2013). Online behavior change detection in computer games. *Expert Systems with Applications*, 40(16):6258–6265.

Vallim, R. M. M. (2013). *Mineração de fluxos contínuos de dados para jogos de computador*. PhD thesis, Universidade de São Paulo.

Vieira, E., Julia, R. M. S., and de Faria, E. R. (2019). Mining data stream to detect behavior change in a real-time strategy game. In *Proceedings of Machine Learning and Data Mining in Pattern Recognition*. ibai publishing.

Weber, B. G. and Mateas, M. (2009). A data mining approach to strategy prediction. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 140–147. IEEE.

Yannakakis, G. N. and Maragoudakis, M. (2005). Player modeling impact on player's entertainment in computer games. In *International Conference on User Modeling*, pages 74–78. Springer.

Yannakakis, G. N., Spronck, P., Loiacono, D., and André, E. (2013). Player modeling. In *Dagstuhl Follow-Ups*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.