IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Video Game-Crowdsourcing Approach to Discover a Player's Strategy for Problem Solution to Housing Development

ARTURO SILVA-GÁLVEZ[1],RAÚL MONROY[2],JOSE E. RAMIREZ-MARQUEZ[3] AND CHI ZHANG.[4]

[1]School of Engineering and Science, Tecnologico de Monterrey, MTY 64849 Mexico (email: artsg130994@gmail.com)
[2]School of Engineering and Science, Tecnologico de Monterrey, MEX 52926 Mexico (email: raulm@tec.mx)
[3]Enterprise Science and Engineering Division, School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, NJ 07030 USA (e-mail: jmarquez@stevens.edu)
[4]School of Economics and Management, Beijing University of Technology, Beijing 100124 China (e-mail: czhang@bjut.edu.cn)

This work was supported in part by Consejo Nacional de Ciencia y Tecnología Políticas

ABSTRACT

The Video game-Crowdsourcing model to recollect data motivates people to participate by entertaining them. Research showed that the solutions players make in this model are competitive against experts in the area. Yet, the studies in the area focus on mimicking people's behavior, including their mistakes. Therefore, we use a Video game-Crowdsourcing to model a problem of interest to find strategies for it. To describe matches from the video game we created, we designed a representation that simplifies the discovery of strategies. Our experimentation compares high score matches against low score ones to find the best behaviors. We used 13 matches played by us to validate the methodology with a known strategy for the problem. Then, we applied it with matches from players. The results suggest that extracting subsequences is a process to find strategies and that we can use them to design algorithms to improve current algorithmic solutions for that problem.

INDEX TERMS Crowdsoursing, Strategy, Video game, Housing Development Problem (HDP)

I. INTRODUCTION

Nowadays, it is impossible to picture a life without technology; this fact gives feedback to Big Data. Every day, people generate a great amount of information through sensors in different locations connected to the internet [1]; all of it composes a Big Data collection. Such data can be analyzed with tools design for Big Data and enhance decision-making processes. From this idea surged a concept called Crowdsourcing. Instead of collecting data that generates daily, Crowdsourcing achieves solutions using the skills of a huge crowd of volunteers [2], [3]. Its objective is to broadcast a specific problem to individuals with certain knowledge [4]. It has been used in a wide range of applications, from identifying solutions for unsolved problems to building virtual prototypes or 3-D designs [4]. Their objective is to detect novel, high-quality solutions; therefore, the range of problems it can cover is wide. Some recent applications include smart cities [5], genetic variability [6] and Natural Language Processing research that take advantage of applications like Amazon Mechanical Turk [3]. Very rarely, though, crowdsourcing is used to uncover an understanding of the decision-making process and strategizing that users follow during problem-solving. Extracting this knowledge and, more importantly, discovering strategies is of paramount importance if one aims to improve the general solutions to problems, which are applicable in many contexts.

In a crowdsourcing task-solving environment, it is vital to have active participants; therefore, creating the need to motivate them. Studies suggest various reasons for people to participate in a platform [7]. Going from giving money, entertainment, networking, or skill training, a crowdsourcing strategy has several options to attract people [7]. Given the popularity of video games, a well-known source of entertainment, recent studies in crowdsourcing have used them

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

to attract more people to their platforms [8]–[10]. This approach, we call *Video game-Crowdsourcing*, seeks to give people an intrinsic satisfaction while participating in an experiment [7]. In addition to normal video games, video game-Crowdsourcing collects data from a match according to an objective; the primary goal of a video game-Crowdsourcing is not to entertain the player. These types of approaches have shown that a crowd can obtain solutions competitive against experts [10].

Video game-Crowdsourcing has been applied in areas such as health, exercise, education, commerce, environmental behavior, among others [7]. The type of video game depends on the application and the target crowd. In tasks where the crowd performs predictions or large quantities of homogeneous tasks, studies commonly use a scoring or leaderboard games [7]. Different applications, like problem-solving or content generation, use more complex mechanisms to entertain participants [7].

Existing research in the use of Video game-Crowdsourcing has centered around player modeling, which attempts to model the behavior of a user while gaming. There are many aspects of what counts as player behavior. One such aspect is how a player navigates in a game [11]. This is often carried out through visual models, such as the use of heat maps [12]; the definition of regions where most actions happen [13]; the use of one-hot encoding [14]; or the use of traces [15]. Player modeling has also been used in the creation of human-like agents, which may act either as an associate or as an opponent [16]–[21].

In this paper, we are interested in the strategy that a player follows while finding a successful or competitive solution to a problem; this is a new paradigm that prioritizes solution and strategy development rather than the solution itself. The motivation behind our work is to help improve existing algorithmic solutions to a problem. To take a few steps towards that goal, we have selected a key part of the Housing Development Problem (HDP) [22], [23]. In our version of HDP, a player is given both a piece of land and a set of different house types; then, the player is asked to place as many houses of different types as possible, under the restriction that the houses ought to be connected by a road. This version of the HDP shares characteristics with the well-known two-dimensional Bin Packing Problem (2DBPP). We have developed a video game that crowdsources strategies of the HDP, giving the players entertainment as reward. The application is a web page that stores every decision a player makes in real-time such that we can reconstruct the whole match with that information.

To discover player strategies, we have abstracted out a typical match, where the player may place a house or a road in space $S \subset \mathbf{R}^2$ to one limited to a finite number of places, using a mesh. Furthermore, we represent a game as a sequence of plays, where each play denotes the placement of a house (complying with the restriction of connectivity) plus a displacement to a position where the player will either place the next house or finish the game. Since a match is now a sequence of plays, or a sequence of symbols, where each symbol denotes a single play, we have used Sequitur [24] to extract sub-sequences of plays of frequent occurrence. This gives us a grammar structure through which we may express a player strategy.

Using this approach, we have discovered strategies used by players who achieved high scores. Some such strategies resemble ones that have already been proposed in the literature [25]. Others seem to be variants thereof, but that has not been reported so far. Interestingly, matches from players who did not do good at our HDP have little or no grammar structure when represented through our abstraction. As a result, these matches contain no pattern that can be allegedly said to convey a strategy. While there is a novelty in the strategies we have found, evaluating their significance, for example, at solving the general 2DBPP, is beyond the scope of this paper. Here we focus on the process of finding strategies from players.

The ultimate aim of this research is the identification of novel and powerful heuristics for the solution of interesting, but often complex, real problems. This paper takes some important steps towards this end. In summary, our research makes three main contributions:

1) An abstract representation of a video match able to express a player's match as a sequence of plays.
2) A means to identify the strategy, if any, followed by a player in a match. Good players follow a clear strategy through which they actually achieve a high score. We have identified a strategy not reported on in the literature.
3) A video game representation for the HDP.

The rest of our paper is organized as follows: In Section II, we show the related work. In Section III, we describe the problem of interest we chose to work with, approaches to solve it, and other similar problems. Section IV presents the Video game-Crowdsourcing model we designed. Section V explains the reasons we decided to represent a match with a specific method and shows the steps to transform one match of our video game into it. In Section VI, we detail the methodology we followed to find the strategies players use in the video game, and we discuss the results of extracting strategies. Section VII discusses our results. In Section VIII, we describe the conclusions and guidance for future research derived from this work.

II. RELATED WORK

A. VIDEO GAME-CROWDSOURCING

Video games are well known for being ludic. Each year, people spend millions of hours playing video games [26], for reasons that include procrastination and stress relief, among others. When used for crowdsourcing, video games make people contribute to achieving a solution to a problem without even being aware of it.

Video game-Crowdsourcing has applications in different areas, involving cybersecurity, computer vision, etc. For example, Peekaboom [27] is a web video game that helps

the construction of computer vision algorithms gathering massive amounts of data. This video game has been played by many people. Further, players have been reported to spend around 12 hours daily on Peekaboom. Not only has video game used to collect a paramount amount of data, but it has also been used to collect solutions with a quality comparable to that of an expert; an example of this is image tagging, as documented in [10], [28]. In particular, Bio-Games [10] has been used to ask the crowd to label images of red blood cells; here, participants got an error of 2%, compared to the result of experts. Bio-Games has also been used to facilitate the process of training medical students for disease classification.

Other studies combined the knowledge of people to improve the performance of machine learning models. EyeWire is a video game that maps a three-dimensional reconstruction of neurons from the retina into images of two-dimensional slices [9]. The gameplay's objective is to follow the trace of a neuron. To do so, the player can color pixels near a neuron or in a new place. Since the video game does not know the correct answer, it needs to compare different solutions to return a score. If a player colors spaces that many others did, the match will obtain a high score. The purpose of this video game is to aid a convolutional neural network train to detect neural boundaries. This network encourages the players to focus their attention on places where it is not sure of the correct answer.

Video game-Crowdsourcing has also been used to complement algorithmic solutions to a problem. This approach has been used, for example, in [8], in the context of the robust Facility Location Problem. FLP consists of a set of demand centers that represent potential locations for opening facilities. The objective is to find the places where facilities should be opened while minimizing both the opening cost and the cost associated with a customer traveling to that facility. The robust version requires a good response when the failure of a facility occurs.

A problem that crowdsourcing applications face is the issue of unmotivated participants. A video game, Cell Evolution, introduced a cooperative crowdsourcing model using blockchain technologies to study crowd intelligence [29]. There, players need to develop a single cell. However, the world of this video game depends on millions of cells. For example, the world is affected negatively if a player decides to upload a toxic cell. This paper studies the interaction among people but concludes that motivational effects are hard to measure.

One method to overcome this obstacle in crowdsourcing approaches is by separating excellent ideas from mediocre ones. To do that, methods like Majority voting [30], Bag of Lemons [30] and the Diverse Bag of Lemons (DBLemons) strategies [30] help ranking solutions to avoid the ones of unmotivated participants.

B. MATCH REPRESENTATION

A classic game model [31] definition divides a video game into six components: (1) The rules that define the game, (2) a quantifiable outcome, (3) scores assigned to the outcome, (4) player actions to get an outcome, (5) attachment from the player to the outcome, and (6) negotiable consequences.

We are interested in identifying the strategy a (successful) player uses to obtain a good score. However, extracting information from a game often modeled as a sequence of actions is not an easy task due to the size of the play space a player has in most video games. Research focuses on providing representation for a match to reduce this space [11]–[13].

One method to represent a video game consists of dividing the game's map into regions. Cavendeti *et al.* [13] proposed one version of it for a single map video game called The Ancients 2 (DOTA2). The representation locates spaces where most of the player's actions happen and selects them as regions. This methodology can mine frequent patterns of winning matches and explain the difference between a player and a reference behavior model from the best players. A recent study employs an algorithm called TRACE [32] to represent a match for a similar video game [15].

A representation of a match abstracts details of the video game that may hide a player's strategy. It gives a structure to the steps the person followed; we can consider a match as a sequence of plays. Here, the strategy should be observable through the complete sequence for that match. The structure of a strategy for a video game can be decomposed in many different ways; studies about player modeling think of it as a set of rules. One method to explain it consists of using descriptors, called patterns that can be expressed as conditionals to evaluate if they are present or not in a match [33]. As an example, a pattern can describe a match where the user played action 1 $(A1)$ after action 4 $(A4)$; it will be $A4 \rightarrow A1 = True$.

C. PLAYER MODELING

Player modeling is concerned with improving the abilities of a computer by generating algorithms capable of following players' behavior under a video game environment [34]. There are three approaches in which an algorithm can mimic a player to achieve the final goal [34].

1) Model actions: It links the state of a game together with a set of applicable plays; then, it selects one via a likelihood function. This approach is similar to a greedy algorithm, in which the algorithm does not evaluate anything but the current state [16], [35].
2) Model based on tactics: Those models have several plans that complete a local goal, making the algorithm follow a series of actions until that goal is reached, then the model will evaluate which tactic to use next. The algorithm keeps using tactics until it fulfills the final goal [36].
3) Model based on strategies: uses tactic models and features to organize the distinct tactics and order the series of actions that will reach the final goal [18].

Modeling strategies is one of the possible approaches researchers in the area use. Although the objective is the same,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

existing applications that use a model-based on strategies are different. They can predict the initial strategy [37], generate the strategy [20], [36], predict the opponent's strategy [19], or finish a match achieving specific goals [38].

In the literature, we can find algorithms that model the strategy by predicting actions given a state in the game. Some of them use machine learning to represent a strategy [39]–[42]. Those models consist of a set of rules: $R = \{(s_0, a_0, e_0), (s_1, a_1, e_1), ..., (s_n, a_n, e_n)\}$ where s_i is a state of the video game, a_i is the action to make in that state, and e_i is the effect that occurs to the current state by performing a_i; it is important to notice that e_i might not be explicit in these models, but every action a_i they take at state s_i produces the new state e_i. At this set of rules, two triplets can be the same, and a rule may not produce a new state existing at another rule from the set. The first statement indicates that in the set of rules it can exist two triplets such that $(s_i, a_i, e_i) = (s_j, a_j, e_j)$, for $i \neq j$. The second statement says that a rule r in the set R can affect e_i such that no rule element of R has a state $s = e_i$; in that case, the algorithm uses a method to find the closest state of a rule to the unknown one and make its action.

One research that uses this method works in a robot soccer game [20]; they define a rule with specific descriptions of the objects on the map. The state involves three components: the coordinates of the player's robots, of the opponent's robots, and the ball; the action tells the coordinates of where the player's robots should move. At every step, the algorithm compares the current game field situation with the strategy's rules to select the closest one. Since it focuses on predicting actions, a rule in the strategy may not connect with others; there might also be duplicate ones. Using a dataset of matches, they extract sequences of game situations, and with a clustering algorithm, they visualize the relations between them. Knowing the different game situations, they believe it is possible to improve strategies for robot soccer games.

Another method to represent a strategy focuses on the whole plan; they do not have the two statements of the models that predict actions. The set of rules for this type of method consists of actions and states $R = \{(s_0, a_0), (s_1, a_1), ..., (s_n, a_n)\}$. The difference is that the actions do not produce an unknown state, and the set does not have repeated rules. Therefore, every action a of a rule in R produces a state s of one rule r in R; also there is no pair $(s_i, a_i) = (s_j, a_j)$, for $i \neq j$ and i, j taking values from 1 to n. Nevertheless, this second approach might not be viable for every video game; for multi-agent video games, the player has no full control. For example, in Super Mario Bros [39], the objects and the map are not always the same, and the player cannot modify those variables, then it is necessary to decide the action in real-time. In different video games, like puzzle-type, the user is the only one controlling the features, allowing the use of whole plan models.

One thing these models have in common is that they define a strategy with a set of rules. Each rule consists of a condition that evaluates one state of the game and the action to proceed if the condition holds. To obtain that set, they follow one of two approaches. In the first one, they start with an initial set of rules selected based on experience and use a database with matches to train their model [18], [36]. The second one uses a database of experts to train their algorithm [19], [20], [37], [38].

A group of video games that attracted a lot of research is Real-time strategy games. In them, players control a set of units distributed on a map, and actions occur in real-time, meaning that they cannot store information during turns [43]. Existing studies apply Bayesian approaches to predict initial strategies [37], evolutionary algorithms to define the behavior according to a set of rules [18], and Case-Injected Genetic algorithm to generate strategies [36]. These studies establish the rules based on experience or a structure that guides the video game. Another research by Ontañon *et al.* [38] builds a framework for Wargus, a video game. To train their system, they use traces from experts with the actions made to complete specific goals. At execution time, it creates a plan tree that consists of goals it wants to fulfill and the actions needed to do it.

A more recent approach uses a video game of a 2DBPP to get human players [16]. The study got ten players and build decision trees using their matches to create strategies. They solve the problem by creating a plan that packs one item at a time. However, they do not make a distinction between the solution of the players. The problem is that they can create a heuristic using a solution that performed poorly against the others.

III. THE HOUSING DEVELOPMENT PROBLEM

The main goal of this document is to provide insight into strategies people apply in a problem of interest. To be of interest means that there are currently people working on it and that a single method does not guarantee the best solution for all instances. The one we found suitable for this study is the Housing Development Problem (HDP). It consists of supplying people with affordable houses. Variations of this problem might include different features an architect needs to consider for this objective, like environmental, aesthetics, or communication.

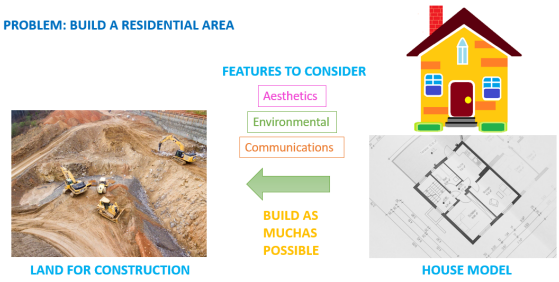

FIGURE 1. HDP aims to comply with the supply of affordable houses. The main goal is to have as many houses as possible on the available land for building a residential area. The real scenario considers features like aesthetics, environment, and communications.

The HDP surged because in high-demand areas at some places in the United States [44] or cities like Hong Kong [45], there is not enough affordable multifamily housing to meet the demand. This problem is of interest to the architecture community. Their job is to ensure that the environment in which people live satisfies their needs, using the land as best as possible. When residents are satisfied with the place they live, their quality of life improves [46]. To create a living environment, the architect must analyze distinct features like the number of houses, access to the different zones of the complex, and other things that can come into their mind.

Recent studies showed interest in distinct aspects of this problem, like satisfaction or environmental issues [47], [48]. Other studies, more related to ours, evaluated different ways of housing affordability [22]. There, the results suggest that increasing the development of moderately priced houses will be affordable over time. What drives these studies is the goal of supplying people with inexpensive housing in walkable neighborhoods [22].

A generalization of the HDP is the Spatial Configuration; it is a problem all physical designs must face. It consists of finding the best locations and dimensions of a set of interrelated objects that give the best performance to the design [23]. Examples of this optimization objective include component packing, route path planning, process and facilities layout design, and architectural layout. The goal of the architecture layout is to find the best location for places in a building restricted to certain conditions [49]; the HDP is similar, but its objective involves the objects in a residential complex.

After analyzing the HDP, we saw that it shares characteristics with another problem that is of interest to the scientific and industrial communities. The problem is the 2DBPP. There exist different types of Bin Packing Problems; each of them has its characteristics. The classical 2DBPPs consist of a set of rectangular items that need to be packed into 2D bins [50]. One variant of the problem works with irregular shape items [51]. A recent study tackles this problem with an approach using video games to train decision trees and obtain Human-Derived-Heuristics [16]

IV. VIDEO GAME DESIGN

Before making the Video game-Crowdsourcing for the HDP, we need the components that describe the HDP. Building residential areas involve several aspects of urban planning and design, production cost, construction law, lack of infrastructure, municipal allocation, taxation, financing, among others [52]. All the characteristics can turn this problem into one that is not easily understandable by all people due to the number of decisions they can make. The benefit of this crowdsourcing model is entertainment. Therefore, participants need to understand and enjoy the video game. Hence we will focus on one of the many aspects a HDP needs to optimize, maximizing the total number of houses, given the land for construction, subject to restrictions of connectivity. The variables we need to consider to simulate this problem are the available land for construction, the models of houses, and the roads that connect the houses.

A limitation on architectural problems like Housing Development is the available land for construction [53]; the initial shape of it depends on the project. Another factor that affects the area available for houses involves the distinct objects that the architect includes in the residential complex; they can be trees, lakes, parks, among others. Changing the shape of the land in the video game can abstract other objects. This paper, nevertheless, considers just one figure for the land. Evaluating the strategies on different land shapes is beyond the scope of the study.

A residential complex needs connectivity between its buildings; it must have one entrance linked to all the houses. The entry allows people to get to their homes either by car or by walking. Assuming that there is just one access, all the houses are connected. The video game uses this assumption to make the player link all the buildings between them. A player can use an item that represents a road to meet the restrictions of connectivity. The video game, however, does not model the entrance.

A. INITIAL STATE

The initial state of the video game presents the player with two objects. One is a fixed polygon; the other object can change between four options. The fixed polygon represents the available land to place houses; the other object represents the items the player can put. The possibilities the player has for the items are three different polygons. They simulate houses and a rectangular figure representing a road. The player can choose to place any of these four items at any time during a match.

B. GAME PLAY

The objective of the video game is to maximize the number of non-overlapping houses (x) in the land. Equation (1) represents the goal of the HDP in the video game without the restriction of connectivity. Here y is the number of roads and a_l, a_{h_i} and a_r the area of the land, a house, and a road. The video game uses a function that ensures a house object is touching a road's polygon and that all the road objects in the land create a bigger one to verify the restriction of connectivity.

$$\begin{aligned} &\max_{x} \\ &\text{s.t.} : a_l - x a_{h_i} - y a_r \geq 0 \end{aligned} \tag{1}$$

C. EVALUATION FUNCTION

The evaluation function needs to be oriented to the goal of the Video game-Crowdsourcing model; it is to maximize the total number of houses. That will encourage players to focus their solutions on the objective we have. Therefore, the score in the video game depends just on the number of houses in the solution. Such a solution consists of the final set with the roads and houses in the land, knowing they all meet the restriction of connectivity.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

D. INTERFACE

The implementation of the Video game-Crowdsourcing considered two points to select the tools. The first one is the ease of distribution among people, which we resolved using a web page, a media capable of getting into a large group of people. The project is available in a Github repository (https://github.com/ArturoSG3/Bin-Packing-Videogame). To create it, we used Phaser2, a framework of Javascript that has functions for video game design. The result of the program is the interface presented in figure 2.

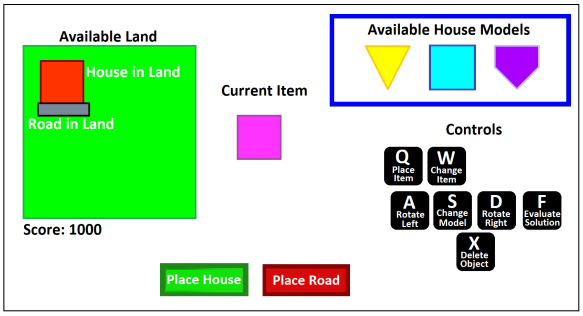

FIGURE 2. Ongoing match of the video game, the red figure represents a house placed in the land; the gray one is a road. The player needs to move the pink figure to the green square to add another red figure. The player must place as many red figures as possible.

The second point is the data capture in real-time; for this, we need a platform capable of storing information. Javascript can use Cloud Firestore, a NoSQL database from Firestore that works in real-time. It receives the information on every play a player performs. The tools we selected meet the requirements of the two points we need for the Video game-Crowdsourcing model. For a more detailed description of the video game, follow this link to download a short video (https://github.com/ArturoSG3/Bin-Packing-Videogame/blob/gh-pages/assets/Houssle.mp4).

V. MATCH REPRESENTATION

We designed our video game to store all the information required to replay every match held by a player. One match from the video game consists of all the plays the person made. Furthermore, each play contains four components: the action, the options are: orient (O), place (P) or eliminate (E); the object, that can be a house (H) or a connection (C); the position represented as a coordinate in the land; and the orientation of the object.

Obtaining a strategy from a player's match is not straightforward because the strategy is reified throughout all the plays of that match. The action space of our video game is huge; there are too many options for a play. That makes it hard to discover a strategy using our video game representation. In an attempt to reduce the play's space, we have abstracted out much of the detail from a video game's match.

A. SEQUENCE OF HOUSES

In the first step of abstraction, we aim to section the land to build a new representation. We split it using a 10×10 grid of bricks of the same size. The reason behind this amount is the number of houses that can be inside one brick. Each one has enough space to fit one house and a piece of road that serves as connectivity to the neighborhood.

Using the grid division, we can transform the coordinate space of the video game. The new one changes a coordinate (x, y) into (i, j), where x and y are the coordinates in the land, and i and j are integer numbers in discrete space. The grid defines the space for i and j; i is the number of column, and j is the row. Then, we transform the position parameter of all elements in the match into the grid coordinates.

TABLE 1. Sample of two plays using the original representation of the video game, the grid coordinates, and the sequence of houses.

Order	First	Second
video game	P, H, (182, 297), 90^o	O, R, (182, 297), 40^o
Grid coord.	P, H, (1, 2), 90^o	O, R, (1, 2), 40^o
Seq. of houses	(1, 2)	Removed

Since we think that the strategy is in the order a player placed the houses, regardless of the orientation and figure of one house, we can abstract more details from the sequence. The sequence includes actions for eliminating or orienting objects and connectivity plays. Following our hypothesis, we can remove those plays from the sequence. The ones remaining are one option for the action, place, and one for the type, house; additionally, we can remove the orientation. An element in the transformed sequence is $[Place, House, (i, j)]$; we will represent them just with the position for simplicity. Table 1 shows an example of the abstractions at this step, applied to a sequence of two plays.

B. REPRESENTING A MATCH USING A SEQUENCE OF MOVEMENTS

To reduce the play's space even further, we introduced a new relation between two plays called a movement. Such abstraction needs a complete match. A movement consists of two consecutive plays, both in different bricks. It starts capturing when the player positions the first house; then it moves the pointer to the place of the second house; at last it places the second house. The movement describes the displacement of going from the first house to the second one. To describe the complete match, we capture every displacement that occurs in a match. There are four different types of movements we defined for this research:

- **N and S movements:** Represented by the cardinal points, this type of movement is detected when there is a change of row. They are vertical movements.
- **E and W movements:** Represented by the cardinal points, this type of movement is detected when there is a change of column. They are horizontal movements.
- **Diagonal movements:** Represented by the cardinal points, this type of movement is detected when the change in the columns and rows are the same. (NW, SW, NE, and SE)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

- **Special movements:** This type of movement occurs when a player moves and next reaches a border. The name has a $+$ sign depending on the edge reached. For example, a movement NW^+ happens when the player places the next house in a brick that can be reached by going north, N, one time and then moves to the west as many times as required to reach the border. The change of rows and columns can be different; the player can move one row and nine columns, but they must end in a border. There are also movements like N^+W^+, which are when they reached a corner. Figure 3 shows an example of a NW^+.

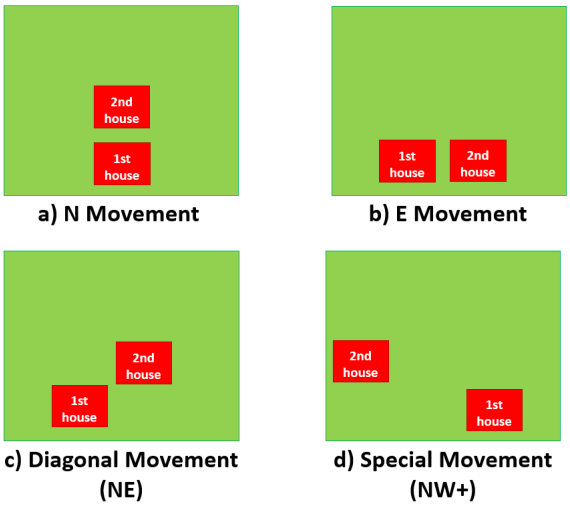

FIGURE 3. Images of how movements look in the vide ogame. The difference between a) and S movement is the order of 1st and 2nd house, same as b) and W movement. For the diagonal and special movements, the name will change depending on the cardinal direction.

If a movement does not belong to any of the classes listed above, it will be labeled “unknown". For example, if a player moves a different number of rows than columns, and it does not reach a border, the movement cannot be expressed using the symbols defined above, and so will be simply tagged “unknown". It follows that there might be strategies that cannot be expressed using our representation.

To express a match with movements, we can transform the ordered pairs from the abstract sequence into one of the four types of movements or an unknown one. However, a movement requires two plays; to define a complete match, we need to describe the first and last movements. The first play does not have a preceding one; we use a different attribute to represent it. The video game stores the first house play as an attribute with three different values: corner, border, or center, depending on the square it lies in. We use this attribute to define the first movement in this representation. The last play does not have a following one; to represent the final movement, we use a null one. However, we believe that the strategy is in the repeated sub-sequences of the match. Since both the first and last movements are unique, they will never be in a repeated sub-sequence. For that reason, we abstracted those two movements.

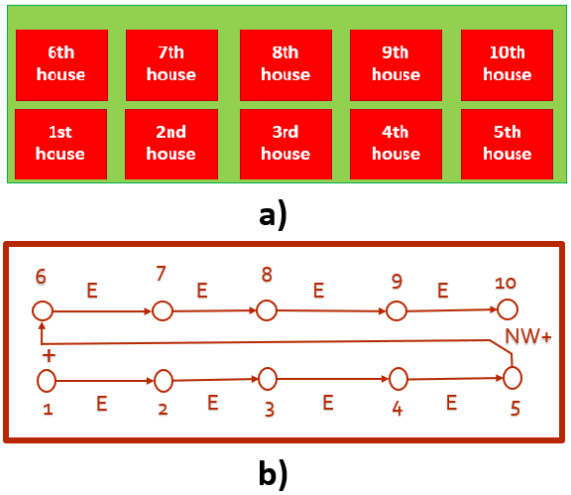

FIGURE 4. Example of Bottom-Left in our video game. Figure a) is how it would look in the video game. Figure b) shows how we interpret the movements. Each circle represents a house; their number is the order of placement. The arrows represent a movement, and their name is according to the groups we defined.

Taking advantage of the similarities between the HDP and the 2DBPP, we oriented this representation to detect all the movements of Bottom-Left (BL). This strategy is well known for the 2DBPP. In the video game, a player that uses this strategy begins placing a house on the bottom left corner of the land. The next movements consist of putting houses to the right of the previous one. When there is no more space for another one to the right, the player searches for the bottom-most left available position. The player repeats these actions until there is no more space to place another house. Figure 4 shows an example of how the BL strategy works for this video game and its representation in a movement sequence.

VI. EXPERIMENTS AND RESULTS

In this section, we describe the results of applying our method for discovering the strategy that a player followed in a match. Our experimentation had two parts validation and testing. It is worth mentioning that those names are not related to the terms used for machine learning. Figure 5 summarizes our approach to detect video game player strategies.

A. DATABASE

Our sample consists of volunteers between 20-60 years with at least a degree in high school. We obtained 113 matches with scores from 4 to a maximum of 24. With a confidence level of 95% and an error of 8%, we calculated the size of the sample (n) we need as follows:

$$n = \frac{1.95^2 \cdot 0.2 \cdot 0.8}{0.08^2} \approx 97 \tag{2}$$

Since our database contains more than 97 users, we can say that the highest score for our video game is 24 ± 2 with a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

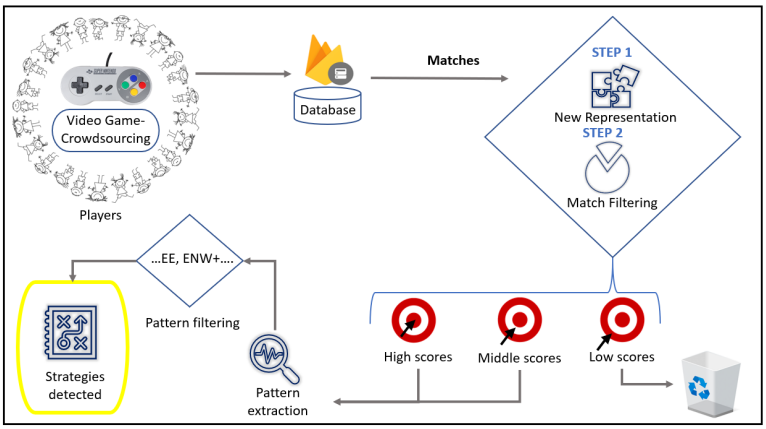

FIGURE 5. The approach begins by collecting matches from our Video game-crowdsourcing in a database. Then, we transform each match into our representation and perform a filter using Pareto's principle. The next step is to obtain patterns with Sequitur. Finally, we perform pattern filtering to select the strategies that characterize the best players.

95% confidence interval, meaning that it exists the possibility that the strategies we find are not the best.

The next step is to find the best players. To identify them, we ordered them by the score the players got. According to Pareto's principle, 20% of the causes generate 80% of the consequences, and the other 80% produce 20% [54] Appealing to this principle, we defined a threshold to distinguish the high score matches; 20% of our matches have a high score, belonging to the best players, and 80% a low one. The threshold we found indicates the person with the lowest score of the high score group; it is 19 points. The amount of positive samples is 24, belonging to the top 20% of the scores. Those matches correspond to the ones where the associated player placed more houses, the objective of the video game.

The worst matches are not considered for the analysis; they belong to the low 20% of the 113. They performed an average of 6 movements; the best matches did 20. Analyzing the sequences with the lowest scores, we saw a few repeated sub-sequences. Meaning that they did not follow a strategy according to our definition; their movements can be considered random. Those matches can generate noise in the negative class, leading to false conclusions. Therefore, the negative group consists of 60% players above the lowest and below the best; they are 64. Figure 6 shows the division of all the matches we collected.

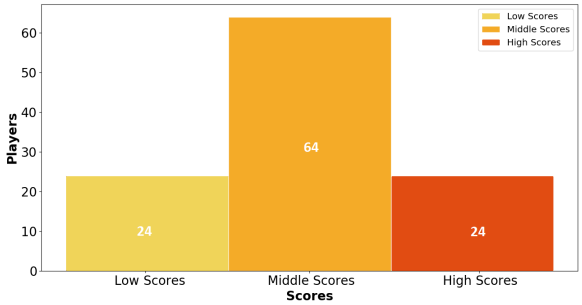

FIGURE 6. Database collected from the Video game-Crowdsourcing platform.

B. VALIDATION EXPERIMENT

This experiment validates the methodology for distinguishing a particular strategy on a match. The objective is to prove that we can describe a strategy using sub-sequences. To confirm the strategy, we need to recognize how the users played their match. We use synthetic matches to know their plays. They consist of one played following the actions of a known strategy, for example, Bottom-Left. We crafted those matches following the strategy carefully to obtain a high score.

TABLE 2. Bottom-Left style strategies sequences and amount of matches in the synthetic class that uses them.

Strategy	Sequence	Matches
Bottom-Right	$W-W-NE^{+}-W-W-NE^{+}...$	4
Bottom-Left	$E-E-NW^{+}-E-E-NW^{+}...$	3
Top-Right	$W-W-SE^{+}-W-W-SE^{+}...$	3
Top-Left	$E-E-SW^{+}-E-E-SW^{+}...$	2
Left-Bottom	$N-N-S^{+}E-N-N-S^{+}E...$	1

We decided to use strategies similar to Bottom-Left (BL) for two reasons. First, we designed the representation for detecting this strategy; a sequence can describe all its movements. Second, we want the experiment to simulate players that use different strategies. BL has variants by changing the starting position and the directions. Table 2 shows the movement sequences of the variants we used of the Bottom-Left of figure 7.

In this type of strategy, we can find two main repeated sub-sequences or patterns, presented in table 3. The Straight patterns in Bottom-Left are the movements to the right or E; it can be a single movement or more than one. The Change of Line pattern happens when the player cannot place another house to the E; it is a NW^{+} movement. We can also find this pattern with Straight ones surrounding it, for example, $E-NW^{+}-E$.

The synthetic class consists of 13 matches using a variant of Bottom-Left; table 2 shows the distribution by strategy. The reason for that amount is because, to test it with people,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

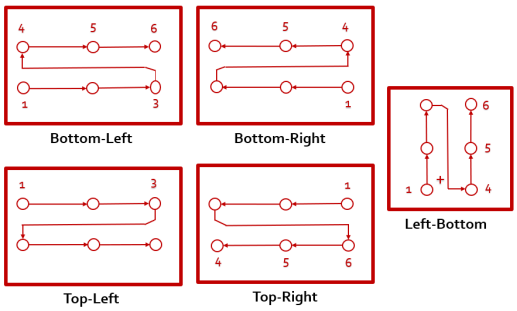

FIGURE 7. Five different variations of the strategy of Bottom-Left.

TABLE 3. Bottom-Left strategies patterns.

Pattern	Examples
Straight	E, W, S, N
Change of Line	$NE^+, NW^+, SE^+, SW^+, S^+E$

we have 24 positive and 64 negative matches from players. That means we need to find strategies in a proportion of $24/64$, meaning that the synthetic matches should be 24 or less. The times we used a strategy in the synthetic match, as table 2 indicates, was to show that we can find strategies that appear few times in the positive class or what is the least amount of matches we need from a strategy to detect it as high scoring. Because we found that limit by repeating strategies from 1 to 4 times, and our amount is less than 24, adding more matches does not provide extra information to this experiment.

Since we do not know the strategy of the actual high score matches from our database, we have to substitute the positive class. Putting the synthetic matches instead of the positive group allows us to have a control group for this experiment. It will serve us to prove that the strategies we find are the ones with the highest scores.

1) Extracting Patterns with Sequitur

Using patterns, we can define a strategy for a given match as a sequence of plays with elements in common. Those happen to be repeated sub-sequences of plays. Finding patterns or repeated sub-sequences is a job that several algorithms can fulfill [32], [55], [56]. Depending on the task, some of them can perform better than the others.

Sequitur is an algorithm that extracts hierarchical structures from a text where the context does not matter [24]. It analyzes a sequence of characters and builds grammatical rules in linear time. The rules obtained belong to sub-sequences in the text that appear at least twice. By applying it to a match, those rules happen to be the repeated sub-sequences that belong to it.

To detect strategies from the matches, we use a version of Sequitur available for python; it receives a string as input and outputs rules for each repeated sub-sequence. Our input must represent each movement in the sequence with one character; if not, the results of Sequitur might not be accurate. Since we have 21 different movements, we can express each one with a distinct character.

To use Sequitur, we need sequences of movements from the matches; Sequitur can analyze one string at a time. We need to decide on a method to obtain patterns of the synthetic class. One option is to apply Sequitur to a sequence composed of all the matches; the other to all of them individually. The first method can find patterns that appear once in more than one match, additionally to the ones that happened at least twice in just one. However, in practice, it finds more redundant patterns, ones that can be described by others. For example, if we have two patterns, $E-E-E$, and $E-E$, the first one is redundant because we can explain it with the second one.

Since both methods can define the same amount of strategies, but the second one uses fewer patterns and avoids redundancy, we applied Sequitur to the individual matches for our experiments. In total, this method obtained 19 different patterns for the 13 sequences from the synthetic group; in table 4, we show some of the patterns, avoiding redundant ones.

TABLE 4. Patterns obtained by applying Sequitur to the synthetic matches individually. We show 8 of the 19 we got, avoiding redundancy.

Index	Patterns
1	$W-W$
2	$E-E$
3	$N-N$
4	SE^+-W-W
5	$E-E-NW^+$
6	$W-W-NE^+$
7	SW^+-E-E
8	$N-S^+E$

2) Detecting Strategies

A match can be described as a short sequence of plays and patterns, whereas a strategy is a description of the process to achieve one match. In this research, we detect strategies manually by combining patterns. A more sophisticated method could achieve different results. In this validation experiment, we know the strategies we want to detect; our goal is to verify that the patterns we found can build them.

The 8 patterns shown in table 4 resemble those found in the sequences of table 2. Before we detect a strategy, we need to reconstruct a match using the patterns we found. For example, we can build a BL strategy using patterns number 2 and 5 of table 4 as in equation (3), where the symbol $*$ indicates that the pattern appears one or more times.

$$EE^*NW^+EE^*NW^+EE^*...NW^+EE^* \qquad (3)$$

Using equation (3), we can create the algorithm 1 to play a match using a BL strategy. Additionally, with the patterns of table 4 we can create an algorithm for all the strategies we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

used in the synthetic matches, meaning that we can detect a strategy by finding sub-sequences

Algorithm 1: Bottom-Left Algorithm

```
→ Start in the Bottom-Left corner.
while Land has empty space do
    → Place House
    if There is space to the right then
        move E
    else
        move NW+
```

3) Comparison with Negative class

One of the goals of the research is to find strategies that lead to high scores. The comparison between classes shows how players with low scores deviated from a high score player. We perform the analysis in two steps. The first one is to detect the difference in the strategies players use in each class. The second is to find the patterns that characterize the best players. It will show us what sub-sequences players make more frequently in the synthetic class than in the negative.

To detect the differences, we applied Sequitur to the 64 negative matches. In total, we got 23 patterns. If we consider the relation of patterns by matches, the negative class has 0.36 $patterns/match$, whereas the synthetic has 1.46. To understand the difference between strategies, we analyzed the sequences of the negative class. We can divide them into two groups according to their sequence, random, or patterns with random changes. The first group performs all of its moves such that there is no repeated sub-sequence. The second group has some patterns, but they have many intermediate random movements between them. Therefore, we can't explain the match without going through specific details. According to this test, a high score depends on following a strategy faithfully, avoiding random movements.

To find the patterns that characterize the synthetic matches, we perform two tests. The first one is Fisher's exact test, and the second is a permutation test. To build the contingency table for Fisher's exact test, we found if each pattern appeared in the matches divided by class. We present an example of this table for one pattern in table 5. Using a significance level of 0.05, 12 out of the 19 patterns approve this test. With them, we can describe three strategies: Bottom-Right, Bottom-Left, and Top-right.

TABLE 5. Contingency table for pattern $W - W - NE^{+}$.

	Positive	Negative	Total
Appeared	3	2	5
Not Appeared	10	62	72
Total	13	64	77

To do the permutation test, we compare the average frequencies; the average times a pattern appears in all sequences of a group; of both classes using the 19 we got by applying Sequitur to the synthetic group. First, we calculate the average frequency for each pattern in both classes; then, with a permutation test, we confirm if there is a significant difference between both groups for the times a player uses a pattern. The permutation test makes a re-sample for both classes; it takes all the frequencies, sorts them, and re-assigns them randomly to the matches. Our test performs $10,000$ permutations. For each one, we calculate the difference between the average frequency of both classes. To test if there is a significant difference, we use a 95% two-tail test. We present an example of a permutation test for one pattern in figure 8.

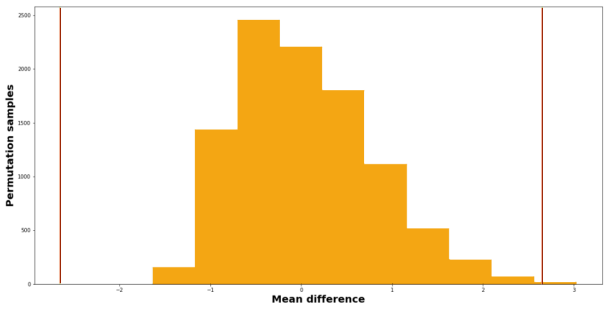

FIGURE 8. Permutation test graph for one pattern. The red lines represent the original difference of the average frequencies.

The test passed 14 patterns of the 19 we had; all of them belong to a strategy we used for the synthetic matches. The first six are the Straight patterns with W or E movements. The other ones are Change of Line patterns that belong to Bottom-Left, Bottom-Right, or Top-Right. The strategies that this test could not detect were used two times or less in the synthetic matches.

Both tests arrive at the same conclusion, even though the first one approves fewer patterns. They suggest that using one of the three strategies more frequently used in the synthetic class will lead to a high score.

C. TESTING EXPERIMENT

After validating our methodology with synthetic matches, we performed the test with real ones. The objective is to detect patterns from the best players to explain their strategy. The patterns we got allowed us to detect several strategies, some of them existing in the literature. The groups we use in this experiment come from the initial division of the 113 players. The experiment concludes by comparing the negative class with the positive as in the validation experiment.

D. EXCTRACTING PATTERNS WITH SEQUITUR

Following the same strategy as in the validation experiment, we applied Sequitur to all the 24 matches from the positive class. The algorithm found 24 patterns. Those patterns will serve us to detect strategies and compare the positive class with the negative in the following sections.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103930, IEEE Access

E. DETECTING STRATEGIES

Using the 24 patterns, we detected five strategies presented in figure 9. From them, we can see that three are variants of Bottom-Left. For the other two, we could not find a similar version in the literature; we named them Snail and Rodent; algorithms 2 and 3 implement their steps.

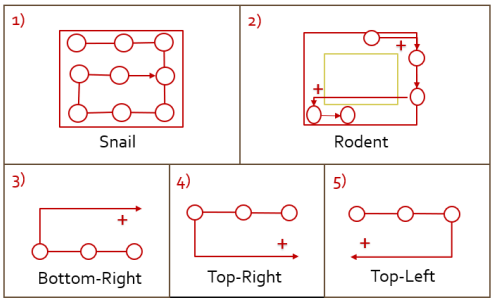

FIGURE 9. Strategies detected with patterns from players.

To produce algorithm 2, we used some of the 24 patterns, as we did to build BL strategy in the validation experiment. That algorithm can produce a match using the Snail strategy. It starts placing houses in one direction until it reaches an obstacle; then, it turns in one direction to continue putting houses. This process repeats, always turning in the same direction until there is no space where to turn. Figure 9 1) is the result of this algorithm, where each circle is a house and the lines are a move.

Algorithm 2: Snail Algorithm

```
→ Start in the Top-Right corner.
Directions = [E, S, W, N]
Current = 0
while Land has empty space do
    → Place House
    if There is space in Directions[Current] then
        move Directions[Current]
    else
        Change Current direction.
```

The second algorithm we found is Rodent. This algorithm will focus on filling the external borders first. Then it uses another strategy to fill the center. It does not matter the order in which the borders are filled. In figure 9 2) the player is placing houses (circles) in different positions at the border; each arrow is a move to where the next house will be. The green square in the middle represents where the player will not put a house until there is no place left in the borders.

We constructed algorithm 3 using some of the 24 patterns we extracted. It selects one of the four borders of the land and fills it. Then, it chooses any other border and continues until the four are full. At last, to fill the center, the player uses a different strategy. It is important to notice that Snail is a variant of Rodent. However, a Rodent strategy might start at the Top border and continue with the Bottom one; it also uses different strategies to fill the center.

Algorithm 3: Rodent Algorithm

```
→ Start in any corner.
Borders = [Left, Right, Top, Bottom]
Current = 0
while At least one border has empty space do
    → Place House
    if There is space in the current border then
        Place house in that border.
    else
        Change current border.
Use strategy to fill the center.
```

F. COMPARISON WITH NEGATIVE CLASS

In the validation phase, we analyzed the difference between the synthetic and negative groups in two steps. The first one consisted of finding the differences in the negative sequences. Since the negative group does not change, the results remain the same. Regarding the second one, the procedure for Fisher's exact test and the permutation test is equal, but in this case, we used the 24 patterns we got from the positive class in this experiment.

Fisher's exact test approved 6 patterns; with them, we can describe completely one strategy, Snail. Besides, some of the patterns belong to the strategies of Rodent and Bottom-Right. This test suggests that these two strategies can give good results when used to complement other strategies. For example, the Rodent strategy needs to do something different to fill the center of the land. With the permutation test, 12 patterns were approved; they belong to three of the five strategies: Snail, Bottom-Right, and Rodent.

The remaining two strategies that this test did not detect might be present in both classes. However, players in the positive one could add different movements to complement the strategy; as we saw when we analyzed the sequences, causing the number of patterns for one strategy to drop. Another possibility is that the number of movements players in the negative class used of those strategies is close to the amount from the positive group, resulting in a similar average frequency. However, the test can confirm that using any of the three strategies that approved the test as the main one will lead to a high score.

VII. DISCUSSION

One of the contributions of this research is the method to represent a match of the video game. Our sequences of plays are a way in which pattern extractors can find relevant moves a player makes. Other research can use it to represent matches from video games that behave similarly.

The validation experiment tested our methodology to extract patterns and detect strategies. It proves our process can identify the strategy a player follows in a match. The reason

for that is because we could detect all the strategies we used to build the synthetic matches.

The validation experiment allows us to confirm that the strategies we found in the testing experiment are the ones followed by the best players. In the testing experiment, we found that the strategies of Snail, Bottom-Right, and Rodent lead to a high score. Our results suggest that if we use any of these three strategies to solve the HDP, we will have a good solution. However, we need to create the algorithm and compare it with others to confirm it.

Although, our results might be biased because of the sample size we have. This means that there is a chance the strategies we found are not the best or there might be others. To avoid this, future research might need to increase the amount of matches to perform the analysis.

VIII. CONCLUSIONS

The model of Video game-Crowdsourcing can collect solutions from people with different skills. An important step to analyze strategies from this model is problem selection. It defines the problem that the crowd will solve and the strategies we can extract. While developing the solution model, we took into account two aspects. First, we considered the problem we chose and how to make it for people to enjoy it. Also, we oriented our solution model to find strategies for the problems of interest we selected.

A limitation related to Video game-Crowdsourcing is that it needs thorough dissemination for people to know the video game. This approach uses entertainment as the method to attract people. However, some people might lose interest because entertainment is not something they can measure, like getting paid. To attract more players, we need a previous phase to promote the video game and let people give it a chance to entertain them. Another limitation of our work is the abstractions to the original problem and the player's match. Each step is a simplification of the real problem and might provoke that our results are not faithful to our goal. For example, a player can make a movement we do not recognize, which avoids our model to find a strategy. Nevertheless, they allowed us to obtain strategies and prove that finding repeated sub- sequences is one method for finding strategies.

An interesting finding of this study was that people use strategies existing in the literature to get high scores. One reason could be because they had applied it to another problem. Another reason is that it came to them as they were solving the video game.

Also, it is important to note that the objective of this study was to find the strategies to get the best solutions. For a different application, it might be relevant to know the path that people took to get a low score. The methodology to achieve it is the same, but instead of finding patterns for high score players, we need to apply Sequitur to the low score players and make the analysis. Nevertheless, in this study, negative cases served us to contrast the matches with the positive ones.

Future research for this study should focus on four things. The first is to create algorithms that use the patterns we found and compare the solutions with existing techniques to solve the HDP or similar ones like the 2D Bin Packing Problem. The second is to expand the database we build to improve our results. The third branch for research to focus on is trying different methods to extract patterns and detect strategies. The last one is on modifying the abstraction of the HDP to include more features in the strategies.

The results of our study are promising, suggesting that we might find new strategies that could lead to high score solutions for problems of interest. Even with a few matches, we could detect strategies that exist in the literature. So, this motivates the continuation of our study so that with major deployment researchers in algorithmics can find strategies that lead to better solutions to problems of interest.

REFERENCES

[1] Y. Hajjaji, W. Boulila, I.R. Farah, I. Romdhani and A. Hussain, "Big data and IoT-based applications in smart environments: A systematic review," *Computer Science Review,* vol.39, 2021, DOI: 10.1016/J.COSREV.2020.100318.

[2] Z. Xu, Y. Liu, N. Y. Yen, L. Mei, X. Luo, X. Wei and C. Hu, "Crowdsourcing Based Description of Urban Emergency Events Using Social Media Big Data," in *IEEE Transactions on Cloud Computing,* vol. 8, no. 2, pp. 387–397, 1 April-June 2020, DOI: 10.1109/TCC.2016.2517638.

[3] Z. Nouri, H. Wachsmuth and G. Engels, "Mining Crowdsourcing Problems from Discussion Forums of Workers," in *Proc. of the 28th International Conference on Computational Linguistics,* Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 6264–6276, DOI: 10.18653/v1/2020.coling-main.551.

[4] J. Füller, K. Hutter and N. Kröger, "Crowdsourcing as a service – from pilot projects to sustainable innovation routines," *International Journal of Project Management,* vol. 38, no. 2, pp. 183–195, 2021, DOI: 10.1016/j.ijproman.2021.01.005.

[5] L.Tan, H. Xiao, K. Yu, M. Aloqaily and Y. Jararweh, "A blockchain-empowered crowdsourcing system for 5G-enabled smart cities," *Computer Standards & Interfaces,* vol. 76, pp. 103517, 2021, DOI: 10.1016/j.csi.2021.103517.

[6] M. Peña-Chilet, G. Roldán, J. Perez-Florido, F. M. Ortuño, R. Carmona, V. Aquino, D. Lopez-Lopez, C. Loucera, J. L. Fernandez-Rueda, A. Gallego, F. García-Garcia, A. González-Neira, G. Pita, R. Núñez-Torres, J. Santoyo-López, C. Ayuso, P. Minguez, A. Avila-Fernandez, M. Corton, M. A. Moreno-Pelayo, M. Morin, A. Gallego-Martinez, J. A. Lopez-Escamez, S. Borrego, G. Antiñolo, J. Amigo, J. Salgado-Garrido, S. Pasalodos-Sanchez, B. Morte, The Spanish Exome Crowdsourcing Consortium, A. Carracedo, A. Alonso and J. Dopazo "CSVS, a crowdsourcing database of the Spanish population genetic variability," *Nucleic Acids Research,* vol. 49, no. 49, pp. D1130–D1137, 2021, DOI: 10.1093/nar/gkaa794.

[7] B. Morschheuser, J. Hamari, J. Koivisto and A. Maedche, "Gamified crowdsourcing: Conceptualization, literature review, and future agenda,' *International Journal of Human-Computer Studies,* vol. 106, pp. 26–43, 2017, DOI: 10.1016/j.ijhcs.2017.04.005.

[8] L. E. P. Estrada, D. Groen and J. E. Ramirez-Marquez, "A serious video game to support decision making on refugee aid deployment policy," *Procedia Computer Science,* vol. 108, pp. 205—214, 2017, DOI: 10.1016/j.procs.2017.05.112.

[9] J. S. Kim, M. J. Greene, A. Zlateski, K. Lee, M. Richardson, S. C.Turaga, M. Purcaro, M. Balkam, A. Robinson, B. F. Behabadi, M. Cam-pos, W. Denk and H. S. Seung, "Space-time wiring specificity supports directionselectivity in the retina", *Nature,* vol. 509, pp. 331—336, 2014, DOI: 10.1038/nature13240.

[10] S. Mavandadi, S. Feng, F. Yu, S. Dimitrov, R. Yu and A. Ozcan, "Biogames: A platform for crowd-sourced biomedical image analysis and telediagnosis," *Games for Health Journal,* vol. 1, pp. 373–376, 2012, DOI: 10.1089/g4h.2012.0054.

[11] L. Chittaro, R. Ranon and L. Leronutti, "VU-Flow: A visualization tool for analyzing navigation in virtual environments," *IEEE Transactionson*

Visualization and Computer Graphics, vol. 12, no. 6, pp 1475-–1485, Nov.-Dec. 2006, DOI: 10.1109/TVCG.2006.109.

[12] A. Drachen and A. Canossa, "Analyzing spatial user behavior in computergames using geographic information systems," in *Proc. 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era,* New York, NY, USA: ACM Press, 2009, pp. 182-–189, DOI: 10.1145/1621841.1621875.

[13] O. Cavadenti, V. Codocedo, J.-F. Boulicaut and M. Kaytoue, "What did I do wrong in my moba game? Mining patterns discriminating deviant behaviours," in *Proc. IEEE International Conference on Data Science and Advanced Analytics (DSAA),* Montreal, QC, Canada, 2016, pp. 662-–671, DOI: 10.1109/DSAA.2016.75.

[14] N. Y. Khameneh and M. Guzdial, "Entity embedding as game representation," in *Proc. 2020 Experimental AI in Games Workshop,* October 2020.

[15] E. Carlini and A. Lulli, "Analysis of Movement Features in Multiplayer On-line Battle Arenas," *Journal of Grid Computing,* vol. 17, no. 1, pp. 45-–57, 2019, DOI: 10.1007/s10723-018-9470-2.

[16] N. Ross, E. Keedwell and D. Savic, "Human-derived heuristic enhancement of an evolutionary algorithm for the 2d bin-packing problem," in *Lecture Notes in Computer Science,* vol. 12270, Springer, Cham, September 2020, pp. 413—427, DOI: 10.1007/978-3-030-58115-2_29.

[17] G. N. Yannakakis and J. Togelius, "A Panorama of Artificial and Computationa Intelligence in Games," *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 7, no. 4, pp. 317-–335, Dec. 2015, DOI: 10.1109/TCIAIG.2014.2339221.

[18] A. Fernandez-Ares, A. M. Mora, J. J. Merelo, P. Garcia-Sanchez and C. Fernandes, "Optimizing player behavior in a real-time strategy game using evolutionary algorithms (CEC)," in: *2011 IEEE Congress of Evolutionary Computation,* New Orleans, LA, USA, 2011, pp. 2017—2024, DOI: 10.1109/CEC.2011.5949863.

[19] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *Proc 2009 IEEE Symposium on Computational Intelligence and Games,* Milan, Italy, 2009, pp. 140—147, DOI: 10.1109/CIG.2009.5286483.

[20] V. Svatoň, J. Martinovič, K. Slaninová and T. Bureš,"Improving strategy in robot soccer game by sequence extraction," *Procedia Computer Science,* vol. 35, pp. 1445—1454, 2011, DOI: 10.1016/j.procs.2014.08.204.

[21] J. Pfau, A. Liapis, G. Volkmar, G. N. Yannakakis and R. Malaka, "Dungeons & Replicants: Automated Game Balancing via Deep Player Behavior Modeling," in *Proc. 2020 IEEE Conference on Games (CoG),* Osaka, Japan, 2020, pp. 431—438, DOI: 10.1109/CoG47356.2020.9231958.

[22] T. A. Litman, "Affordable-accessible housing in a dynamic city: Why and how to increase affordable housing development in accessible locations," *Victoria transport policy institute,* March 16 2021, [Online]. Available: https://www.vtpi.org/aff_acc_hou.pdf.

[23] J. Michalek, R. Choudhary and P. Papalambros, "Architectural layout design optimization," *Engineering Optimization,* vol. 34, no. 5, pp. 461—484, Sep. 2002, DOI: 10.1080/03052150214016.

[24] C. G. Nevill-Manning and I. H. Witten, "Identifying Hierarchical Structure in Sequences: A linear-time algorithm," *Journal of Artificial Intelligence Research,* vol. 7, no. 1, pp. 67-–82, 1997, DOI: 10.1613/jair.374.

[25] S. Jakobs, "On genetic algorithms for the packing of polygons," *European-Journal of Operational Research,* vol. 88, no. 1, pp. 165–181, Jan. 1996, DOI:10.1016/0377-2217(94)00166-9.

[26] L. von Ahn, "Games with a purpose," *Computer*, vol. 39, no. 6, pp. 92–94, June 2006, DOI: 10.1109/MC.2006.196.

[27] L. von Ahn, R. Liu and M. Blum, "Peekaboom: a game for locating objects in images," in *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI),* Montreal, Canada, April 2006, pp. 55-–64, DOI: 10.1145/1124772.1124782.

[28] T. Ivanjko, "Crowdsourcing image descriptions using gamification: a comparison between game-generated labels and professional descriptors," *42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO),* Opatija, Croatia, 2019, pp. 537-541, DOI: 10.23919/MIPRO.2019.8756841.

[29] K. Xin, S. Zhang, X. Wu and W. Cai, "Reciprocal Crowdsourcing: Building Cooperative Game Worlds on Blockchain," *IEEE International Conference on Consumer Electronics (ICCE),* Las Vegas, NV, USA, 2020, pp. 1–6, DOI: 10.1109/ICCE46568.2020.9043129.

[30] I. Lykourentzou, F. Ahmed, C. Papastathis, I. Sadien and K. Papangelis "When Crowds Give You Lemons: Filtering Innovative Ideas using a Diverse-Bag-of-Lemons Strategy," in *Proc. of the ACM on Human-Computer Interaction,* 2018, vol. 2, pp. 1–23, DOI: 10.1145/3274384.

[31] J. Juul, "Half-real: Video Games Between Real Rules and Fictional Worlds," Ann Arbor, MI, USA: MIT Press, 2005.

[32] E. Carlini, A. Lulli and L. Ricci, "TRACE: Generating traces from mobility models for distributed virtual environments," *Lecture Notes in Computer Science,* vol. 10104, pp. 272-–283, May 2017, DOI: 10.1007/978-3-319-58943-5_22.

[33] L. Cañete-Sifuentes, R. Monroy, M. A. Medina-Pérez, O. Loyola-González and F. Vera Voronisky, "Classification Based on Multivariate Contrast Patterns," *IEEE Access,* vol. 7, pp. 55744–55762, 2019, DOI: 10.1109/ACCESS.2019.2913649.

[34] S. C. Bakkes, P. H. Spronck and G. van Lankveld, "Player behavioural modelling for video games," *Entertainment Computing,* vol. 3, no. 3, pp. 71–79, 2012, DOI: 10.1016/j.entcom.2011.12.001.

[35] M. B. Johns, N. D. Ross, H. A. Mahmoud, E. C. Keedwell, D. J. Walker and D. A. Savic, "Augmented evolutionary intelligence: Combining human and evolutionary design for water distribution network optimisation," in *Proc. Genetic and Evolutionary Computation Conference (GECCO),* New York, NY, USA, 2019, pp. 1214-–122, DOI: 10.1145/3321707.3321814.

[36] S. Liu, S. J. Louis and M. Nicolescu, "Using CIGAR for finding effective group behaviors in RTS game," in *Proc. 2013 IEEE Conference on Computational Inteligence in Games (CIG)*, Niagara Falls, ON, Canada, 2013, pp. 1–8, DOI: 10.1109/CIG.2013.6633652.

[37] G. Synnaeve and P. Bessière, "A Bayesian model for opening prediction in RTS games with application to StarCraft," in *Proc. IEEE Conference on Computational Intelligence and Games (CIG'11),* Seoul, Korea (South), 2011, pp. 281–288, DOI: 10.1109/CIG.2011.6032018.

[38] S. Ontañón, K. Mishra, N. Sugandh and A. Ram, "Case-based planning and execution for real-time strategy games," *Lecture Notes in Computer Science,* vol. 4626, pp.164—178, 2007, DOI: 10.1007/978-3-540-74141-1_12.

[39] Letter Symbols for Quantities, ANSI Standard Y10.5-1968. J. Ortega, N. Shaker, J. Togelius, G. N. Yannakakis, "Imitating human playing styles in Super Mario Bros," *Entertainment Computing*, vol. 4, no. 2, pp. 93—104, 2013, DOI: 10.1016/j.entcom.2012.10.001.

[40] N. van Hoorn, J. Togelius, D. Wierstra and J. Schmidhuber, "Robust player imitation using multiobjective evolution," in *Proc. IEEE Congress on Evolutionary Computation,* Trondheim, Norway, 2009, pp. 652–659, DOI: 10.1109/CEC.2009.4983007.

[41] J. Muñoz, G. Gutierrez and A. Sanchis, "Towards imitation of human driving style in car racing games," in: *Believable Bots: Can Computers Play LikePeople?*, Springer-Verlag Berlin Heidelberg: Hingston, Philip, 2012, pp.289-–313, DOI: 10.1007/978-3-642-32323-2_12.

[42] I. V. Karpov, J. Schrum and R. Miikkulainen, "Believable bot navigation viaplayback of human traces," *Believable Bots: Can Computers Play LikePeople?*, Springer-Verlag Berlin Heidelberg: Hingston, Philip, 2012, pp.151–170, DOI: 10.1007/978-3-642-32323-2_6.

[43] A. J. Fernández-Ares, P. García-Sánchez, A. M. Mora and J. J. Merelo, "Adaptive bots for real-time strategy games via map characterization," in *Proc. IEEE Conference on Computational Intelligence and Games (CIG),* Granada, Spain, 2012, pp. 417–721, DOI: 10.1109/CIG.2012.6374185.

[44] C. Gabbe, "Local regulatory responses during a regional housing shortage: An analysis of rezonings in Silicon Valley," *Land Use Policy,* vol. 80, pp. 79-–87, Jan. 2019, DOI: 10.1016/j.landusepol.2018.09.035.

[45] J. Huang, G. Q. Shen and H. W. Zheng, "Is insufficient land supply the root cause of housing shortage? Empirical evidence from Hong Kong," *Habitat International*, vol. 49, pp. 538-–546, Oct. 2015, DOI: 10.1016/j.habitatint.2015.07.006.

[46] A. A. Fakere, O. Arayela and C. O. Folorunso, "Nexus between the participation of residents in house design and residential satisfaction in Akure, Nigeria," *Frontiers of Architectural Research,* vol. 6, no. 2, pp. 137-–148, June 2017, DOI: 10.1016/j.foar.2017.02.003.

[47] M. Markatou, "Urban Planning and Greening Practices: A Case For Neighborhood Development in a Typical Urban Area," *Journal of Environmental Science and Engineering B,* vol. 9, pp. 189-–199, 2020, DOI: 10.17265/2162-5263/2020.05.004.

[48] K. Mouratidis, "Neighborhood characteristics, neighborhood satisfaction, and well-being: The links with neighborhood deprivation," *Land Use Policy,* vol. 99, Dec. 2020, DOI: 10.1016/j.landusepol.2020.104886.

[49] S. S. Wong and K. C. Chan, "EvoArch: An evolutionary algorithm for architectural layout design," *Computer-Aided Design,* vol. 41, no. 9, pp. 649-–667, Sep. 2009, DOI: 10.1016/j.cad.2009.04.005.

[50] M. Beyaz, T. Dokeroglu and A. Cosar, "Robust hyper-heuristic algorithms for the offline oriented/non-oriented 2D bin packing prob-

lems," *Applied Soft Computing,* vol. 36, pp. 236-–245, Nov. 2015 DOI: 10.1016/j.asoc.2015.06.063.

[51] R. P. Abeysooriya, J. A. Bennell, A. Martinez-Sykora, "Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation," *International Journal of Production Economics,* vol. 195, pp. 12-–26, Jan. 2018, DOI: 10.1016/j.ijpe.2017.09.014.

[52] A. G. Hansson, "Promoting planning for housing development: What can Sweden learn from Germany?," *Land Use Policy,* vol. 64, pp. 470-–478, May 2017, DOI: 10.1016/j.landusepol.2017.03.012.

[53] C. S. Chan, "Cognitive processes in architectural design problem solving," *Design Studies,* vol. 11, no. 2, pp. 60-–80, April 1990, DOI: 10.1016/0142-694X(90)90021-4.

[54] R. Dunford, Q. Su and E. Tamang, "The Pareto Principle," *The Plymouth Student Scientist,* vol. 7, no. 1, pp. 140–148.

[55] B. Negrevergne, A. Termier, M. C. Rousset and J. F. Méhaut, "Para Miner: A generic pattern mining algorithm for multi-core architectures," *Data Mining and Knowledge Discovery,* vol. 28, pp. 593-–633, 2014, DOI: 10.1007/s10618-013-0313-2.

[56] T. Uno, M. Kiyomi and H. Arimura, "LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets," in *Proc. IEEE ICDM Workshop on Frequent Itemset Mining Implementations,* 2004.