"Spread-It": A Strategic Game of Competitive Diffusion through Social Networks

Shimon Ben-Ishay, Alon Sela, and Irad Ben-Gal
Dept. of Industrial Engineering, Tel Aviv University

***Abstract*— Diffusion of information is a key factor in many social and political situations. This work presents a strategic game called "Spread-It," which models the spread of information through social network structures. In the game, two competing parties must decide on the allocation and timing of their limited resources with the goal of increasing their influence in the network. Since present decisions affect the future level of network penetration, their effort allocation must be carefully planned. The work starts by defining the mathematical characteristics of the game, followed by analytical results derived by implementing several strategies for winning the game. Analyzing the experiments provides few observations regarding the role of influencers ('hubs') under different game conditions, the superiority of Monte Carlo tree search strategy over traditional game-tree search methods, and the budget required to guarantee the game's finalization.**

***Index Terms*— B2B Marketing, Game Theory, Information Diffusion, Political Games, Social Network**

I. Introduction

Spread-It is a proposed two-player zero-sum game aimed at analyzing strategic thinking in the art of acquiring influence through social networks. The game can be played on a virtual or physical board that represents an organization's social graph, where the set of vertices (nodes) is associated with members in the network, and the set of edges defines the connectivity between these members. Two opposing parties (players) compete on the board to increase their dominance, as a single choice of the organization. "Spread-It" can be played over any undirected graph representing the game's board, unlike other board games, e.g., Chess, Checkers and Othello, which are played on specific boards.

The reality that the game simulates can be seen as a case of two opposing attitudes trying to win a group of decision-makers. Such scenarios are seen, for example, in politics between two parties or in a bidding process where two suppliers are submitting competing proposals. Thus, the game can mimic two suppliers trying to win a bid within an organization by influencing different decision-makers to adopt their product / service. In large organizations, bids are applied regularly to choose between suppliers, and only one supplier is eventually chosen. Each of the two players represents an attitude (supplier), and the players are lobbyists of the supplier within the organization, trying to advance their attitude within the social graph of decision-makers. In each move, each player must decide on what node (key decision maker) to invest its convincing efforts, where each unit of 'convincing effort' is represented by a token, with different colors representing different attitudes for each of the lobbyists.

Influencing someone to adopt an alternative is seldom completed in a single interaction and is instead often accomplished through repeated efforts. This logic is embedded in the proposed game, where each player adds a single unit of influence (a colored token) to a chosen node during his/her turn. The token represents a limited (e.g., marketing) resource, such as budget or energy, required to influence the selected member. Then, when the total number of tokens in a node reaches the node's threshold, the node 'fires' or 'explodes' (we use both terms interchangeably), indicating that the associated member adopts the alternative of the player that gains the majority of tokens on the node. This adoption of an opinion by a node is named "an explosion" in the game's terminology. When an explosion occurs, first, all the tokens on a node are changed to the color of the winning party. This represents the loss of the effort of the losing party, and the effort invested by the winning party that increases its strength by winning its opponent. Furthermore, when an explosion occurs, after having the color of the losing party changed to that of the winning party, the tokens are distributed to the neighboring nodes. This process, represented by the "explosion," may repeat itself in a cascade manner, where one converted member influences another, thus representing the influence that the exploding node has on its neighbors toward adopting the alternative that it has.

As in many games, this game is a simplified set of rules that mimics a certain aspect of a highly complex reality. This simplification clearly implies limitations for the scenarios that the game represents. For example, the game does not represent the scenario in which an effort to influence opinions is performed through mass media such as the use of TV, radio or billboards, but only an effort to influence opinions through a personal one-to-one contact. Moreover, "Spread-It" falls under the category of sequential games (sometimes called *dynamic games*), where players move at different points of time and can observe some, if not all, the choices of other players before deciding upon their optimal response. This contrasts with *static games* (or *simultaneous game*) where players making their moves simultaneously. A broader discussion of the limitations of the game's model as well as the reality represented by its set of rules is discussed in subsection III.C.

The game includes two variations that represent opposite real-life scenarios: a Zero Loyalty variant and a Full Loyalty variant. In the Zero Loyalty variant, each node can flip and change its color (alternative) repeatedly without being "loyal" to any player, even the one that initiated the interaction with the node. This variant can represent a social dynamics case where people can unlimitedly change their opinions. Typically, this model is used to simulate the diffusion of ideas and opinions, such as the attitude towards a news event or the support of different political proposals, which may switch back and forth based on new information that is gathered along the process. In the Full Loyalty variant, each node can select an alternative only once. That is, once the node (person) has selected a color (adopting an alternative), it remains loyal to this alternative and does not change it further. This model can be used, for example, to study the diffusion of the adoptions of new products since these adoptions are typically associated with a purchase behavior or a significant investment and are not easily reversible.

This paper studies the two game variants and mainly focuses on the Zero Loyalty variant. It addresses some mathematical properties of the proposed model that rely on earlier games, such as the chip-firing game [1], and based on these works, it defines some mathematical properties that can be relevant in real-life scenarios. Several strategies and heuristics are studied and benchmarked to find potentially good influential policies in certain networks. Among these methods, a Monte Carlo tree search (MCTS) method is found to provide relatively better results. Moreover, several interesting observations can be made based on the empirical study. Analyzing the conducted experiments provides some unintuitive observations regarding the limited role of influencers ('hubs') in the situation of no customer loyalty to an alternative ('zero loyalty'); the superiority of probabilistic approaches (e.g., the MCTS) over traditional game-tree search methods; and the budget required to guarantee the game flow. Furthermore, the game provides a structured deterministic case, where the complex art of coalition formation can be studied, and demonstrates the difficulty of prediction in such scenarios where influence and decision-making is performed within a set of interacting social connections.

"Spread-It" was initially created as a board game in [2]. The game is freely available for non-commercial use and can be downloaded at http://tinyurl.com/md2tbke.

II. Background

This section describes various diffusion models through social networks. We use the following notation throughout the paper. Let $G = \langle V, E \rangle$ be a graph that represents the social network, where the set of vertices $V = \{v_1, v_2, \dots, v_n\}$ is associated with the individual users in the network, and the set of edges $E = \{e_{ij} = (v_{i,}, v_j) | v_{i,}, v_j \in V\}$ defines the connectivity between the users.

Models for the processes by which information and influence propagate through a social network have a long history in social science studies. These information diffusion models have been widely applied in a "word of mouth" (WOM) context [3]. In marketing, WOM has been acknowledged as a major influencer in the promotion of new products. Traditionally, WOM has been specified as information exchanged through face-to-face interactions; though more recently, the term has been extended to online or to technology-enabled information exchange between individuals. The mathematical properties of information spread through social networks have been studied in the well-known work of Kempe et al. [4]. The authors considered two basic models of information spread: the *Linear Threshold* (LT) model and the *Independent Cascade* (IC) model. These models represent the core ideas of information spread by which the Linear Threshold model captures social influence, and the Independent Cascade model captures retention loss. More formally, the LT model assumes that a node v can change from state 0, in which it is *non-infected* (or *inactive*), to state 1, in which it is *infected* (or *active*), and such a change depends on the mapping of weights and states of the node's infected neighbors, denoted by the subset δ_v. Thus, node v becomes infected and changes its state to 1 only if $\sum_{\delta_v} b_{v,u} \geq \theta_v$, where $\theta_v \sim U[0,1]$ is the threshold of v, and $b_{v,u}$ represents the social influence of the infected neighbor u on v. In the IC model, when node v becomes infected, it has a single chance to infect each 'currently non-infected' neighbor u. The infection attempt succeeds with probability $P_{v,u}$. Accordingly, the *influence maximization problem* (IMP) is defined as a selection of k *seed nodes* (the 'seed set') that maximizes the expected number of infected nodes by the end of the diffusion process. Kempe et al. [4] proved that the IMP problem under both IC and LT models is an NP-hard problem and showed that a greedy approach would reach a solution within 63% of the optimal bound. Unlike the "Spread-It" game with a Zero Loyalty variation, the LT model and the IC model do not consider recovery dynamics by which an infected node can reversely become uninfected.

A trivial extension of the IC or the LT model is to introduce a competitive setting for the diffusion of influence through the social network. In real-world cases, for example, there is a competition among firms providing products or services that compete for the same market share. These firms compete for strategic members (influencers), aiming to maximize the adaptation of their products. These products are usually pricey, and in many cases, it is unlikely that a consumer will purchase more than one competing product within a short period of time.

In recent years, several competitive influence propagation models have been studied. Bharathi et al. [5] proposed an extension to the IC model and showed that the last player to select the seed set can obtain at least a $(1 - \frac{1}{e})$-approximation to the optimal strategy. Carnes et al. [6] studied the available strategies of a company trying to introduce a new product into an existing market where a competing product already exists. Such a setting turns the problem into a *Stackelberg game* [7]. Clark and Poovendran [8] introduced a model called *Dynamic Influence in Competitive Environments* (DICE) and showed that the IC and the LT models can be derived as special cases of DICE. Like the proposed "Spread-It" game (with a Zero Loyalty variation), a node in the DICE model can switch between adopted alternatives over time. The second variant of DICE (simultaneous-move game) considers a game of *incomplete information*, in which neither player can observe the other's moves. This setting is different from the proposed "Spread-It" model, in which a *perfect information* game is assumed, i.e., all players have access to the game information and know exactly at each time what are the influence efforts that are made by their opponent. This scenario not only allows a better strategic planning of the moves, but also fit the reality to a certain degree.

Influence was further studied through a game theoretic approach. In this direction, Alon et al. [9] introduced a model of diffusion of influence through a social network and studied the relation between the network diameter and the existence of pure Nash equilibria. Tzoumas et al. [10] generalized the model proposed by Alon et al., and considered a setting where each player tries to infect a set of k seed nodes. While in the proposed "Spread-It" model, each player can only influence (by putting a color token) one node at a time in every turn. Another difference between Tzoumas et al.'s and Alon et al.'s models vs. "Spread-It" is that in the former models, each node can have at most one alternative, i.e., a node that has adopted a particular alternative cannot alter its decision later on, while in the latter model (the "Spread-It" with Zero Loyalty variation), each node can flip and change its color repeatedly without being "loyal" to any alternative. Furthermore, in Alon et al.'s model, if two players compete for the same node at the same time, they "cancel out" each other, and the node is removed from the game (colored gray), while in "Spread-It," such conflicts are allowed, i.e., different players can compete for the same node at the same time, as often happens in real-life settings.

III. The "Spread-It" Model

"Spread-It" is a game played on a finite graph. For purposes of simplicity, we have chosen to focus on a finite, connected, undirected graph without self-loops or parallel edges. Every vertex (v) in the graph has a threshold value k_v, which is set to $\deg(v)$ the degree of v unless otherwise defined. Full information is assumed, and k_v is thus pre-specified and known to both players. When the threshold is reached, the node "fires" (sends) one token to all its connected nodes. The players' objective is to gain influence through the network by turning all (or the majority of) the nodes to their color. In this paper, we focus on a two-player game that represents a competitive dual diffusion through social networks. As indicated above, two game variants are considered, as follows:

— In the *Zero Loyalty* variant, a member (vertex) can change its alternative (color) throughout the game, and a particular vertex in the graph can thus fire several times.

— In the *Full Loyalty* variant, a member (vertex) can select an alternative (color) and fires only once. Once the node fires, it is colored by the winning color (according to a majority rule), and the players continue competing for the remaining vertices (if such exist), while the colored node remains unchanged.

A. *"Spread-It" Rules and Notations*

Let $G = \langle V, E \rangle$ be a finite connected undirected graph without loops with n vertices and m edges, in which each vertex has a predefined threshold value k_v. The function $T: V \to \mathbb{N}$ is a token configuration. $T_b(v)$ is the number of *black* tokens on vertex v, and $T_r(v)$ is the number of *red* tokens on vertex v. Two players (black and red) play the following game:

1) Initially, there are no tokens on the vertices: $T_b(v) = 0, T_r(v) = 0 \; for \; all \; v \in V$.
2) Each player receives a designated, equivalent number of tokens equal to $m - \frac{n}{2}$ (in subsection III.D.2, an explanation for this number of tokens is given).
3) Players alternate turns, each of which consists of two phases. In the *first phase*, the player puts his colored token on any vertex on the graph. If a vertex v has as many tokens as its threshold $(T_b(v) + T_r(v) \geq k_v)$, it fires (or "explodes"). In the *second phase*, following the explosion, the firing vertex distributes one token of the winning color to each of its neighbors.
4) The player with the largest number of tokens at the time that the vertex reaches its threshold "wins" all the tokens on this vertex, coloring all of them. If the numbers of tokens on a node are equal and the threshold is reached, the color of the last token placed on the node defines the winning color.
5) Formally, when v explodes, T_c is modified to a configuration T'_c such that:
$$T'_c(u) = \begin{cases} T(v) - k_v & if \; u = v, \\ T_c(u) + 1 & if \; uv \in E(G), \\ T_c(u) & otherwise; \end{cases} \quad (1)$$
where c is the winning color (black or red) on v. Firing vertex v means that we decrease $T(v)$ by k_v tokens, and increase $T_c(u)$ by 1 for each neighbor u of v.
6) In the Full Loyalty variant of the game, the exploded vertex is marked (fixed) by the winning color and becomes inactive, which means that no player can place any more tokens on this colored vertex anymore.
7) In the Zero Loyalty variant, the game is won when both players have finished their initial tokens and one player's tokens are a majority or when the game reaches the infinite topple ("endless explosions") and only one color of tokens is on the graph, giving that player a clear victory. In the Full Loyalty variant, the game is won by the player having the majority of vertices on the board with his/her color.

Explosion Order. When an explosions cascade occurs after a player action, it follows an order defined by the dominance of that player. Technically, these vertices are sorted by the difference between the player's colored tokens vs. the other colored tokens in descending order. Accordingly, the first exploded vertices are the ones where the player has the largest majority. Such ordering gives the player an advantage. An alternative order is to assign an index to each vertex in the graph such that if several vertices meet the criterion for explosion, they explode by their index order.

B. *Examples of Game Play*

Below, two examples are proposed - one per each "Spread-It" variant.

1) *Zero Loyalty variant.*

The Zero Loyalty variant is presented in Fig. 1.

(a) The initial board state - each vertex is identified by an id. The vertex threshold is denoted by the number at the center of the vertex. The number of red tokens on a vertex is denoted by the number at the bottom left-hand side of the vertex. The number of black tokens on a vertex is denoted by the number at the bottom right-hand side of the vertex. **(b)** Black places his/her token on $v1$. **(c-d)** Red places his/her token on $v1$. An "explosion chain" starts on $v1$ since it reaches its threshold (2). Because Red was the last player to put a token on $v1$, he/she is the winner of this vertex. $v1$ explodes and fires one red token to each of its neighbors: $v2$ and $v3$. **(e)** Black places his/her token on $v3$. Now, $v3$ has reached its threshold of 2 and explodes. When $v3$ explodes, it fires one black token to each of its neighbors: $v1$ and $v2$. **(f)** The "explosion chain" continues. $v2$ reaches its threshold of 2. Black is the winner of this vertex as he/she was the last player to put a token on $v2$. $v2$ explodes and fires one black token to each of its neighbors: $v1$ and $v3$. **(g)** The "explosion chain" continues. $v1$ reaches its threshold of 2. Black is the winner of this vertex as he/she has the majority of tokens in the vertex (2 vs. 0). The game ends. One obtains an infinite chain of Black explosions. The winner is Black, who has won all the tokens in the game. Red has lost all of his/her tokens.

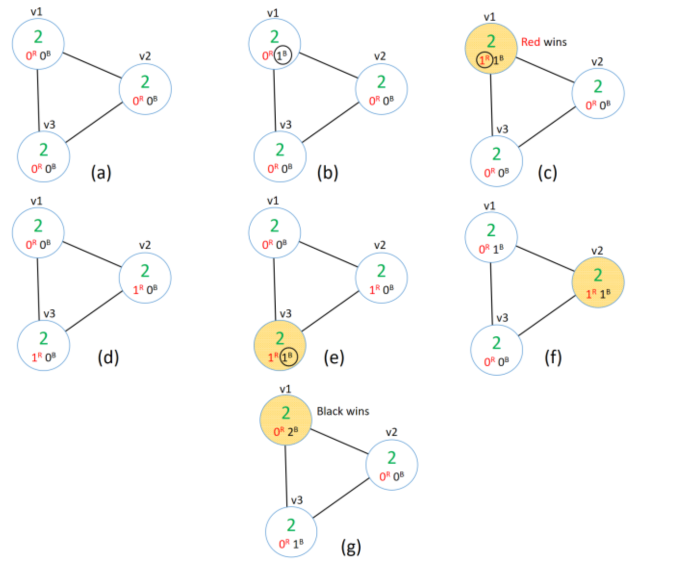

Fig. 1 An example of a Zero Loyalty game played on a small board. In this example, some nodes ($v1$) change their alternatives.

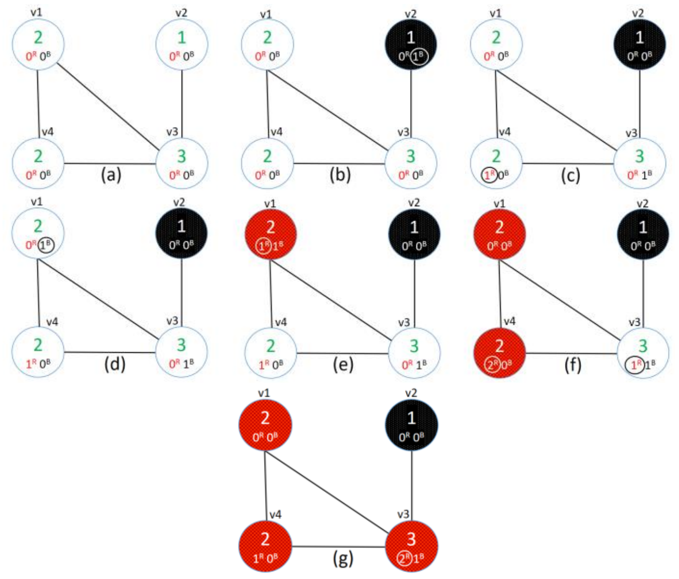

Fig. 2 An example of Full Loyalty game played on a small board.

2) *Full Loyalty variant.*

The Full Loyalty variant is presented in Fig. 2.

(a) The initial board state. **(b)** Black places his/her token on $v2$. The vertex explodes since its threshold is 1. Black wins this vertex and "occupies" it. The vertex is now Black forever, and the players compete on all vertices except for $v2$. **(c)** Red places his/her token on $v4$. **(d)** Black places his/her token on $v1$. **(e)** Red places his/her token on $v1$. Red wins the vertex and "occupies" it. The players then compete on all vertices except for $v2$ and $v1$. **(f)** An "explosion chain" starts. $v4$ explodes in Red. While $v1$ is the neighbor of $v4$, this vertex is out of the game at this stage (since it is already occupied). It does not receive a token. One token remains in the "exploded" $v4$. **(g)** The "explosion chain" continues. $v3$ explodes in Red. Red wins this vertex and occupies it. All the vertices are "occupied" by the players. Red has won the majority of vertices in the game and is declared as the winner.

C. "Spread-It" and Real-World Influence Diffusion Scenarios

This subsection presents some of the properties of "Spread-It" and explains how the game model fits (to a certain extent) real-world scenarios of influence diffusion through social networks.

— **Vertex coloring and node "explosion".** In the game, when a vertex reaches its threshold, all the tokens change their colors to the majority's color before the vertex fires. Such a rule represents a situation where a few players (e.g., companies) are trying to convince a decision maker in an organization to adopt their alternative (e.g., product or service). When enough persuasion efforts have been invested, the decision maker threshold is reached, implying that the he reached a decision. Then, all efforts made by the non-winning player become obsolete. The "explosion" indicates that the person not only has adopted the winning alternative but also become an agent of that alternative and now influences his/her neighbors toward this alternative. A more realistic approach will consider the case where the nodes might, or might not, accept the influence following the "explosion" by some probability value, but this probabilistic approach is outside the scope of this paper and could be considered in future research.

— **The winner takes all.** When a player wins a vertex, all efforts made by the losing player on that specific vertex become practically useless; moreover, these efforts serve the winning player. An equivalent scenario in real life occurs when two software companies compete for a new customer in a request for information (RFI) process. Each of the companies allocates efforts to answer the RFI, proposing new features and methods (effort is represented by tokens). When the customer decides to adopt one of the proposals, the winning company profits from the effort of the loosing company as well, e.g., by using parts of the features that are now integrated by the client in a single RFP (request for proposal) or by the publicity of such a win against a company that invested in the competition. However, even in the general case, the winning company gains a new customer, which improves not only its cash flow but also its competitiveness in comparison to the losing party.

— **The relation between the node's threshold and its degree.** Selecting the node's threshold to be greater than or equal to its degree represents a scenario in which the required effort to win a decision maker (node) is proportional to his influence (reflected by the node degree). For example, winning a bid in a large influential company requires, in most cases, greater effort compared to winning a bid in a smaller company. Naturally, the winning results (financial gains) are usually higher when winning a bid in a larger company than in a smaller one. Nevertheless, it is not at all certain that the relationship between influence and ease of access is linear as applied in the game. The linear relation between influence and ease of accessibility is modeled in the game's dynamics mainly to provide an easy (and functioning) set of rules that apply to the game. In the Experiments section (see subsection V.F) we examine the case where the node's threshold is smaller than the node's degree and analyzed its effects on the player's choices.

— **Token distribution.** Let us note that the distributions of tokens at the beginning of the game are equal for both players. This setting represents an underlying assumption that the two competing players have relatively equivalent alternatives. For example, the products of the two firms are comparable in terms of price, features, branding and quality, such that the main factor that influences the purchasing decision depends on their strategic effort to influence the decision maker. Of course, this is not the case in many real scenarios, where one product outperforms the other in one or more aspects. Nonetheless, such a scenario where one product clearly outperforms the other is less relevant for the game since the customer's choice is clearer.

— **Influencing several nodes at a time**. Note that each player can use a single token per turn. This implementation results in a simpler game rule, speeds up the game and maintains a continuous game flow. An alternative is to allow players to use more than a single token at each turn. Note, however, that under such a scenario, before the player places its next token, it is necessary to apply the 'firing rules' on each node which is about to explode until a stable state of the game is reached. Following an experimentation of this alternative, we have concluded that applying a rule where a player may use more than one token in each turn tends to result in an abrupt game which is much harder to follow and to analyze, particularly for human players.

D. "Spread-It" and the Chip-firing Game

This section considers an important related model to "Spread-It" known as the Chip-Firing Game (CFG) [1]. The CFG provides a broad base for better understanding the "Spread-It" game and its various characteristics. The model provides insights and properties regarding some of the "Spread-It" features. However, the games are inherently different, as discussed below.

CFG is a widely used model in physics, economics, computer science and other science domains to illustrate dynamical systems. For example, in physics, CFG is often used to describe the phenomenon of self-organized criticality [11]; in economics and computer science, CFG was proposed as a model for resource distribution systems [12]. The classic CFG model is a solitaire game played on a graph [1]. The graph or game board is comprised of a set of interconnected vertices, some of which have multiple chips. The "degree" of a vertex is defined as the number of edges that directly connect it to other vertices. When the number of chips is greater than or equal to the degree of the vertex, the vertex "fires" its chips, and such firing sends one chip on a vertex to each of its connected neighboring vertices (see Fig. 3). Leftover chips remain in the original vertex.

The game is played by firing a single vertex at each time step until a stable configuration is reached (that is, until no vertex has more chips than its degree) or until it is determined that stability cannot be achieved. The game is said to be in a stable configuration if none of the vertices are in the process of firing. One trivial example of a stable configuration is the empty configuration, in which none of the vertices contain any chips.

A few of the CFG properties can provide insights into the dynamics of the "Spread-It" model. For example, the total number of tokens on each vertex, regardless of their colors, can reveal when the game has become stable. This point is further discussed later. Additional background on chip-firing games can be found in [13], [14].

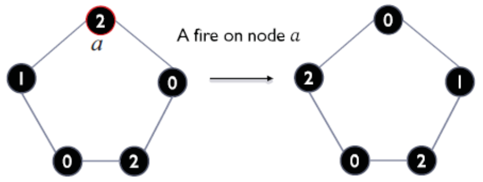

Fig. 3 Example of CFG "Firing."

1) CFG vs. "Spread-It: Similarities and Differences.

At the early stages of the "Spread-It" game in the Zero Loyalty variant, one can notice certain characteristics that distinguish both games:

— *Infinity*: "Spread-It" can reach a state of infinity (a series of explosions in vertices) without reaching any stable state.

— *Periodicity*: A series of recurring explosions in certain graph structures and game configurations that are worth further examination.

The classic CFG is similar to "Spread-It" with several changes:

(a) CFG is a solitaire game; "Spread-It" is a two-player game.
(b) CFG uses one color of tokens; "Spread-It" uses two colors.
(c) In CFG, the explosion order in a given turn does not matter as the same game state is eventually reached. In "Spread-It," the order matters as two different colored tokens are used. To decide whom the winning player is at a given moment, the order of the explosions is important.

2) CFG Properties Used in "Spread-It" Game.

In most CFG models, including the classic one (Björner et al. model [1]), the threshold of each vertex v is set to be the degree of v, $deg(v)$. By applying this threshold to any vertex in the "Spread-It" graph, one can infer the following properties:

Infinity. Based on CFG theory, it is known that reaching a state of infinity depends on a number of factors:

— N – The total number of tokens on the game board (one can ignore the color of the tokens and consider them as a single color).

— n – The number of vertices in graph G.

— m – The number of edges in graph G.

Both n and m are static and known in advance to both players. N keeps changing in every turn as each player adds a token to the graph.

If the total number of tokens (N) is more than $2m - n$, then the procedure cannot terminate. With this number of tokens, one node will have at least as many as its degree. If $N \leq 2m - n$, then a terminating position can occur. If the number of tokens is less than m, then the game always terminates. Furthermore, if $m \leq N \leq 2m - n$, then the game may or may not terminate, depending on the original configuration of the tokens. At every moment of the "Spread-It" game, one can calculate these parameters and determine, for example, whether the game is in an infinite state and declare the winner. One can deduce that if both players play at least m turns, the game enters a stage in which it can end in an infinite topple. This can be a problematic situation for a computer agent. For example, a tree-search based agent generates a game tree to find the best next move, and if one of the tree nodes (game states) will lead to an infinity state, then the tree generation process may not terminate. To prevent such cases, these CFG properties can be used to identify the "*infinity zone*" and thus create a mechanism to stop the tree generation at this step.

Token Distribution. The use of CFG properties enables the determination of the number of tokens each player will receive at the beginning of the game according to the characteristics presented above. Each player is given exactly $m - \frac{n}{2}$ tokens. If more tokens are given, the game will be endless (since, if the total number of tokens is greater than $2m - n$, the game is in an infinity state). If the number of tokens (two colors together) is less than m, then one can reach the point where the players have used all their chips, but there is no explosion at all on the board. By the distribution choice we have made, we let the game be in the infinite or finite state. This depends on each player's moves.

Note that the above properties do not hold if the vertex's threshold is not equal to its degree, which is a valid option of the "Spread-It" game. Please refer to Appendix A for an example demonstrating such a case.

The "Spread-It" game is a challenging problem for a computerized agent. The methods used to build such an agent are presented in the next section.

IV. Methods

A. AI Game Agents

This subsection describes the search methods that were studied in order to find a good game strategy that wins "Spread-It." All these methods seek a promising seeding of nodes by tokens with respect to some decision criteria. A direct implementation of a brute-force search in "Spread-It" on a large network (hundreds or thousands of nodes) is computationally prohibitive. The main reasons are the game complexity and the lack of an adequate evaluation function. However, by solving "Spread-It" on smaller networks (of a few dozen nodes), one can assess some of the strengths and weaknesses of various game strategies. Moreover, optimal game strategies over small networks may be used as a benchmark for testing other searching techniques. For these small networks, an Alpha-Beta algorithm was utilized [15]. For larger networks (over 30 nodes), where the branching factor is larger, the tree search methods (such as Alpha-Beta) are no longer a tractable strategy. Accordingly, a Monte Carlo tree search (MCTS) method, which was previously implemented to master the game Go [16], was utilized in "Spread-It." Along with these two methods, a Dummy strategy (using a random player) was also implemented mainly for a benchmark purpose. The following subsections elaborate on each of these agents.

1) Alpha-Beta Agent. Turn-based games can be represented as a "game tree." Tree nodes represent network (board) situations, while branches are a possible board configuration following the players' actions. The tree leaves represent possible game endings and contain information on the value (utility) to each of the players given the related ending. One theoretical method for searching for the best action is to generate the entire game tree for a given position and select the action that results in the highest utility. The problem with this approach is that the game trees for most of the board games, including "Spread-It," are usually too large to be searched entirely in a reasonable time. A common approach for performing searches on game trees is based on the Minimax algorithm along with an optimization scheme [15]. The Alpha-beta pruning is a common optimization scheme of the Minimax algorithm. It limits the search space by avoiding searching sub trees of moves which won't be selected, allowing for a relatively more powerful look-ahead search. In addition to the Alpha-beta pruning, iterative deepening strategy has been used as an enhancement for this agent.

A vital requirement for game tree searches is the use of an *evaluation function* (heuristic) by which the Minimax algorithm operates. The heuristic function directs the search process by scoring various states of the game. Note that, since a perfect evaluation function is not given and often does not exist in a real setting, we inspected a handful of heuristics, including some combinations of functions. In particular, the following heuristics were studied:

— *Parity heuristic* - Captures the token difference between the current player and opponent player.

— *Stability heuristic* - Each player's tokens are classified into three categories:

(a) Stable: Tokens that cannot be flanked (alternate color) in the next game steps.

(b) Semi-Stable: Tokens that might be flanked in the next game steps.

(c) Unstable: Tokens that can be flanked in the next action.

Weights are associated with each of the three categories, typical weights can be: w_s=1 for stable tokens (T_s), w_{us}=-1 for unstable tokens (T_{us}) and w_{se}=0 for semi-stable tokens (T_{se}). The sum $T_s \cdot w_s + T_{us} \cdot w_{us} + T_{se} \cdot w_{se}$ defines the final stability value for each player.

— *Hubs heuristic* - A common assumption relates the number of hubs in a region to the player's stability in that region. In other words, it assumes a positive correlation between the number of hubs owned by a player and the probability of that player to win the game. Following this approach, a score is given to each node in the graph according to the player's chances of winning that node (e.g., a high score was given to a node that is about to explode during the next move and the player is the winner of that node). Accordingly, each node's score is multiplied by the PageRank value of that node, prioritizing hubs. The sum of the nodes' scores defines the final Hubs value for each player.

— *Mobility Heuristic* - The number of possible next moves. One can assess the player's potential mobility, taking into account the number of beneficial and non-beneficial moves.

— *General heuristic* - Two factors are used to rate the overall state of the game: the "counting factor," which counts the number of majority nodes a player has and the "remaining factor," which calculates the player's remaining tokens in each node required for it to "explode." The remaining factor is given a larger weight.

— *Linear Combination (LC) heuristic* - The LC consists of linear combinations of all the other heuristics and looks as follows: $LC\ Score = w_1 \cdot H_1 + w_2 \cdot H_2 + \ldots + w_j \cdot H_j$, where H_j is the score of a particular heuristic, and w_j is the associated weight assigned to this heuristic score. To learn the weights, a learning algorithm was utilized, following Mitchell's model of a computer agent for the game of checkers [17]. The learned weights determine the relative importance of the various heuristics. A detailed description of the implemented approach can be found in Appendix B.

2) Monte Carlo Tree Search Agent. In "Spread-It," the size of the board (graph) is theoretically unlimited. However, conventional game-tree search algorithms (such as Alpha-Beta) are suitable for games with a fixed network size. We found that, when the graph size was larger than approximately 25-30 nodes, conventional search techniques became computationally limited. To address this challenge, we used a Monte Carlo tree search (MCTS) procedure. The MCTS method has been used in games with large branching factors, such as Go [18], [19]. In general, the MCTS method determines the most promising action in any given situation by simulating a large series of random moves (simulations). By relying on the results gathered from multiple simulated games, the algorithm builds a partial game tree until some predefined constraint (e.g., time or the number of iterations) is reached. At this point the search is stopped and the node with the most promising statistics is selected. During the tree-building process, nodes are added iteratively to the search tree and the algorithm maintains the visit count and the win count for each node. MCTS inclines toward high-scoring moves over low-scoring moves and therefore more time is spent in examining optimal moves. The UCT (Upper Confidence Bound for Trees) algorithm that was originally formalized by Kocsis and Szepervari [20] is the most common basis for MCTS implementation and was applied in this study as a game agent. The following four phases are performed per search iteration:

a) *Selection*: Selecting the best node to start exploration, until the agent come across a node with unexplored child nodes or it reaches the leaf node. In the case of the UCT algorithm, the selection strategy is based on the UCB1 multi-armed bandit algorithm [21]. Thus, from node v, a child node i is selected to maximize:

$$UCT = \overline{x_i} + c \cdot \sqrt{\frac{\ln(n_v)}{n_i}}$$

where $\overline{x_i}$ denotes the average score (i.e., the wining rate) of node i, n_i is the number of times node i has been visited, n_v is the number of times the (parent) node v has been visited and c is an exploration constant.

b) *Expansion*: If the current selected node has unexplored child nodes, select a child and add it to expand the tree.

c) *Rollout*: simulate the game till the end from the new selected node(s).

d) *Backpropagation*: Results received by the simulations are backpropagated through the selected nodes to update their statistics.

By repeating these four phases iteratively, the search tree is gradually constructed.

3) Dummy (Random) Agent. This Dummy agent places tokens on random nodes in every turn and is being used mainly for a benchmark purpose.

V. Experiments and Results

This section describes the experiments used to rate the overall performance of the implemented AI agents, as well as the effecting scenarios. Over 22,000 experiments of complete games were executed using the "Spread-It" Engine ("SPRITE") and analyzed later.

The section is organized as follows. Subsection V.A presents the experimental settings. Subsection V.B describes the testing methodology. Subsection V.C exhibits the results of the Zero Loyalty variant. Subsection V.D exhibits the results of the Full Loyalty variant. Subsection V.E examines the role of central nodes. Subsection V.F examines the relation between the node's degree and its threshold. Subsection V.G summarizes some of the main observations.

A. Experimental Settings

- *Game Engine:* an engine named "SPRITE" (**Spr**ead-**It** **E**ngine) was developed to simulate the "Spread-It" game. The engine draws the graph layout, calculates the network measures, and generates synthetic graphs for the experiments.

- *Environment*: The tests were performed on 4 Intel Xeon 2.27-GHz Quad-Core virtual machines (VM), each with 8 GB of RAM.

- *Graph Data Set*: A total of 15 synthetic undirected graph networks were generated. The primary focus was on real-world networks. Each graph (game board) was taken from a different network class: Small-World Graph, Random Graph and Preferential Attachment Scale-Free Graph (Barabási–Albert model [22]). Five graphs were generated for each network class with numbers of nodes and edges ranging from 14 to 88 nodes and 19 to 258 edges.

- *Parameters*: The following parameters were set for all test cases:

1) The starting player is black.
2) The chosen evaluation function (heuristic) for the Alpha-Beta algorithm was the Linear Combination (LC) heuristic described above. This heuristic was selected since it achieved the most promising results in the preliminary experiments and theoretically generalizes the proposed heuristics. A complete report of the results of the different Alpha-Beta heuristics is given in Appendix C.
3) The Linear Combination heuristic weights were set as follows: $LC\ Score = w_1 \cdot H_{parity} + w_2 \cdot H_{stability} + w_3 \cdot H_{mobility} + w_4 \cdot H_{hubs} + w_5 \cdot H_{general},\ where\ w_1 = 0.27,\ w_2 = -0.08,\ w_3 = 0,\ w_4 = 0.315,\ w_5 = 0.495.$ These weights were obtained by the learning algorithm described in IV.A.1 and Appendix B.
4) The learning rate for finding the optimal set of weights for the LC heuristic was set to $\eta = 0.05$.
5) The Alpha-Beta algorithm's search depth was set to 4.
6) The number of Monte Carlo tree search iterations was set

to 1000.

7) The exploration constant of the MCTS UCB1 formula was set to $c = \sqrt{2}$.
8) The MCTS rollout (simulation) policy used uniformly random move choices.
9) Time limit for each agent was set to 2.5 seconds.
10) The threshold of each vertex in any graph was set to the degree of the vertex. For the case where the threshold is not equal to the degree, see subsection V.F.

B. Testing Methodology

The performances of the different AI agents were evaluated against each other, and the number of wins, losses and draws were counted for each AI agent, where w counts the number of black wins, l counts the number of red (the opponent) wins (or black losses), and d counts the number of draws where the game is infinite and where there is no convergence to one player's color (i.e., there is no winner). Based on experimentation, the following rates were calculated for each game method:

— *Win rate (WR)*: The percentage of games won over the total number of games, given by $\frac{w}{w+l+d}$.

— *Draw rate (DR)*: The percentage of draws over the total number of games, given by $\frac{d}{w+l+d}$.

— *Win Loss ratio (WLR)*: This ratio ignores the draws and represents the ratio between the number of wins to the total number of wins and losses, given by $\frac{w}{w+l}$.

The results are shown with their 95% confidence interval along with the win percentage (WR).

C. "Spread-It" Zero Loyalty Results

TABLE I shows the performance of each agent against all the other agents over different graph boards. The first column lists the starting (black player) agent, while the second column lists the red player agent (six agents' combinations). The rest of the columns (WR, DR, WLR) represent the performance of the black player. Each set of players is matched up 500 times – 250 times as the first player (black) and 250 times as the second player (red). These games were tested over 15 different graphs played for a total of 22,500 games ($6 \cdot 250 \cdot 15$). A summary of the performance results is presented in TABLE II.

TABLE I
ZERO LOYALTY VARIANT – AGENT RESULTS

Black Player	Red Player	WR%	DR%	WLR%
Alpha-Beta	MCTS	31.0 ± 1.48	4.1	32.4
Alpha-Beta	Random	97.5 ± 0.49	0.1	97.9
MCTS	Alpha-Beta	70.1 ± 1.46	4.2	74.0
MCTS	Random	99.6 ± 0.20	0	99.6
Random	MCTS	0.7 ± 0.26	0.1	0.7
Random	Alpha-Beta	4.0 ± 0.62	1.1	4.0

TABLE II
ZERO LOYALTY VARIANT – WIN RATE SUMMARY, OVER ALL GAMES PLAYED

	MCTS	Alpha-Beta	Random
Win Rate (WR%)	85.3 ± 0.80	64.1 ± 1.08	2.38 ± 0.34

It is clear that the MCTS approach is by far the most powerful stand-alone agent, as it wins (WR) ~99% against the Random player and ~70% WR against the Alpha-Beta agent. The Draw Ratio (DR) for MCTS is approximately 4% against the Alpha-Beta agent. As expected, the Random agent is clearly the weakest strategy with only 4% wins against the Alpha-Beta and less the 1% against the MCTS agent. The Alpha-Beta agent has a win rate of 31% against the MCTS agent and 97% WR against the Random player.

As described in subsection "Graph Data Set," the graph classes included Small-world graphs, Random graphs and Scale-Free graphs. TABLE III below shows the win rate (WR) for each agent per each graph class against all other agents. The name of the agent listed in the first column is the starting (black) player. While seemingly there are no meaningful differences between these three graph families, the MCTS approach obtains the highest win rate in the Barabàsi-Albert graph class at $\sim 87\%$. It is possible that one could have seen greater differences between the three graph classes with larger graphs (having hundreds of nodes or more), but the high game's complexity makes it difficult to test such a scenario.

TABLE III
ZERO LOYALTY VARIANT – AGENT RESULTS GROUP BY GRAPH CLASS

Agent	Random (WR%)	Watts-Strogatz (WR%)	Barabàsi-Albert (WR%)
Alpha-Beta	65.3 ± 1.86	63.7 ± 1.88	61.3 ± 1.90
MCTS	79.4 ± 1.58	82.5 ± 1.48	86.9 ± 1.32
Random	2.1 ± 0.56	3.0 ± 0.66	3.0 ± 0.66

D. "Spread-It" Full Loyalty Results

The Full Loyalty variation represents the case where members do not change their alternative throughout the game; therefore, a particular node in the graph can explode only once during the game. As seen in TABLE IV and TABLE V, the superiority of the MCTS method is maintained throughout the Full Loyalty variation as well, and reached a higher win rate compared to the Zero Loyalty variant results presented in TABLE I: ~74% WR vs. ~70% WR when the agent competed against the Alpha-Beta agent. Furthermore, it seems that the Alpha-Beta agent reaches considerably worse scores compared to the results of the Zero Loyalty game. Both the Alpha-Beta and MCTS agents in the Full Loyalty variation obtained higher draw rates.

TABLE IV
FULL LOYALTY VARIANT – AGENT RESULTS

Black Player	Red Player	WR%	DR%	WLR%
Alpha-Beta	MCTS	13.6 ± 1.09	17.6	16.5
Alpha-Beta	Random	91.1 ± 0.91	7.8	98.9
MCTS	Alpha-Beta	74.7 ± 1.39	14.1	87.0
MCTS	Random	99.6 ± 0.20	0.2	99.8
Random	MCTS	0.2 ± 0.14	0.4	0.2
Random	Alpha-Beta	1.8 ± 0.42	8.9	1.9

TABLE V
FULL LOYALTY VARIANT – WIN RATE SUMMARY, OVER ALL GAMES PLAYED

	MCTS	Alpha-Beta	Random
Win Rate (WR%)	87.5 ± 0.74	50.0 ± 1.13	1.0 ± 0.22

TABLE VI shows the win rate (WR) for each agent over each graph class against all other agents. As can be seen, the MCTS agent obtains again (as seen for the Zero Loyalty variant in TABLE III) the highest win rate (WR%) over the Barabàsi-Albert graph class with ~ 95% WR. The Alpha-Beta agent achieved the highest WR over the Watts-Strogatz graph family with ~56% and the lowest WR over the Random graph family with ~37%.

TABLE VI
FULL LOYALTY VARIANT – AGENT RESULTS GROUP BY GRAPH CLASS

Agent	Random (*WR*%)	Watts-Strogatz (*WR*%)	Barabàsi-Albert (*WR*%)
Alpha-Beta	37.3 ± 1.89	56.7 ± 1.94	51.7 ± 1.95
MCTS	84.7 ± 1.41	83.0 ± 1.47	95.0 ± 0.85
Random	0.3 ± 0.21	1.7 ± 0.50	1.2 ± 0.42

E. *Are Central Nodes important in "Spread-It"?*

The following subsection inspects the influence of the central nodes on the modus operandi of an agent and its winning rate. From the experiments mentioned above, we checked whether a particular agent (MCTS or Alpha-Beta) preferred nodes with higher centrality metrics over other nodes. This scenario is strongly related to real-world phenomena in which companies or individuals compete for the influencers in social networks. The common convention holds that investing more efforts to gain control over influencers is an effective strategy that leads to better influence through the social network [23], [24].

A representative scale-free graph (Barabási–Albert model with 24 nodes and 70 edges) was selected from the graph data set mentioned above. This graph class was chosen since the topology of a scale-free network is dominated by a few highly connected nodes (often called "hubs") while the rest of the nodes having very few connections. The games where the black agent (first player) was the MCTS or the Alpha-Beta were analyzed. The percentage of times a particular node was chosen by an agent over all the other agent's selections was recorded. Different centrality metrics such as PageRank, eigenvector, etc. were tested in order to rank the nodes. However, no meaningful ranking differences were found in comparison with the degree centrality metric. In this analysis, we focused on the top 20% of nodes based on the degree (threshold) centrality metric. For this group of nodes we summed up the percentage of times they were selected.

Fig. 4 shows the percentage of times each agent selected the top 20% nodes in two game sets (i) *all games* and (ii) *games where the agent won* for the Zero Loyalty variant of the game. Fig. 5 presents the results of the same analysis for the Full Loyalty variant. It is notable that the MCTS agent in the Zero Loyalty variant does not have a strong preference for the top 20% nodes over the other nodes. In both game sets, the top 20% of the nodes were selected by the MCTS agent approximately in ~20% of the times, thus, in accordance with their overall percentage in the population, as expected from a random selection of nodes. Interestingly, the Alpha-Beta agent in the Zero Loyalty variation, chose the central nodes less frequently than could been expected if the choice was purely random. In contrast, in the Full Loyalty variant, the MCTS agent had a stronger preference for the top 20% of the nodes (selecting them ~30% of the times). The Alpha-Beta agent in the Full Loyalty variant selected the top 20% nodes approximately ~20% of the times, while in the Zero Loyalty variant, these central nodes where only chosen ~9% of the times.

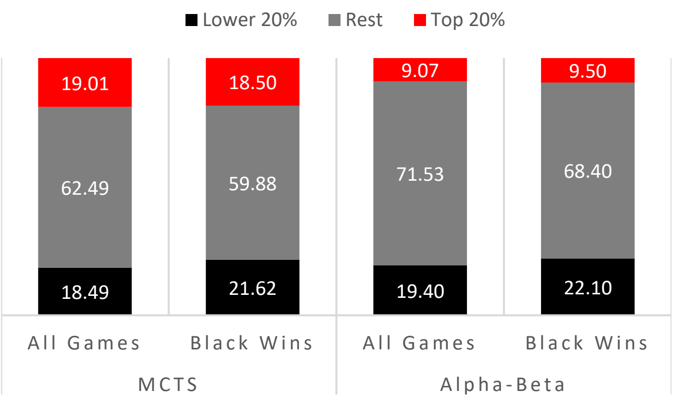

Fig. 4 Zero Loyalty – Number of times high central nodes (top 20%) and low central nodes (bottom 20%) were chosen by the MCTS vs. Alpha-Beta agents.

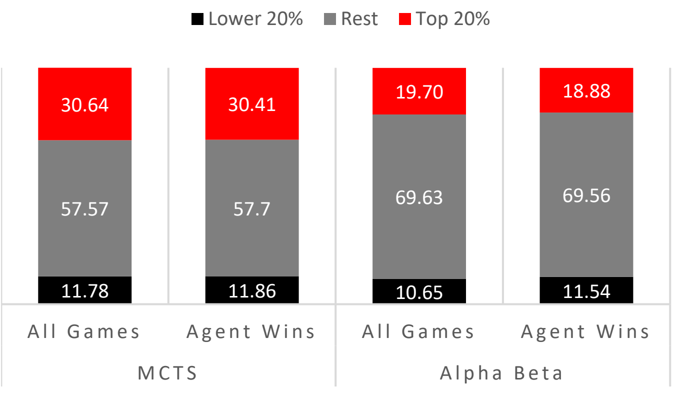

Fig. 5 Full Loyalty - Number of times high central nodes (top 20%) and low central nodes (bottom 20%) were chosen by the MCTS vs. Alpha-Beta agents.

F. *The Effect of a Threshold Unequal to the Node's Degree.*

In all the experiments performed so far, the node's threshold was set to be equal to the node's degree. In this experiment, we inspected how the agent node's selection is affected when there are nodes with a threshold that is not equal to their degree. Specifically, we aimed at investigating whether an agent would prefer a more 'central' node (i.e., a higher-degree node) with a lower firing threshold. This implies that a lower "investment" (i.e., lower number of tokens) is required to potentially win this node, although the "profit" remains the same, since the exploded node distributes relatively more tokens than the threshold to its neighbors. In the case where the node's threshold is lower than its degree, new tokens are created ex nihilo for the agent who won at that node, while in the case where the node's degree is equal to the node's threshold the "energy conservation" is maintained in the system (as no new tokens are created besides those on the board). To test this hypothesis, we studied the same scale-free graph mentioned in subsection V.E above. From this graph, we have selected the two most central nodes (based on their degree-centrality metric) with a degree equal to 10 and a threshold equal to 5 that was arbitrarily determined to be equal to half of the degree. For each agent (MCTS or Alpha-Beta) we executed 500 games on the graph board with the agents playing against the other agent and the Random agent. The percentage of times each node with the reduced threshold (from the two selected ones) was selected by an agent was recorded and compared with the "regular" case where the selected nodes had a threshold equal to their degree.

Fig. 6 and Fig. 7 present the results for the Zero Loyalty and the Full Loyalty variants respectively. It is evident that the dominant agent from the previous experiments (MCTS) clearly prefers nodes with higher degree than their threshold (18% of the agent moves) in comparison with the previous case in which the threshold is equal to the node degree (7.7% of the agent moves). A similar observation was obtained for the Alpha-Beta agent. In the Full Loyalty variant of the game, both agents still preferred the nodes where the degree is higher than the threshold over nodes where these parameters are equal. Note that the selection of nodes with a reduced threshold in the Zero Loyalty variant was substantially higher (18% vs. 7.7% and 11.2% vs. 4.1%) in comparison to the Full Loyalty variant where these differences were less noticeable (17.8% vs. 12.5% and 11.5% vs. 9.6%).

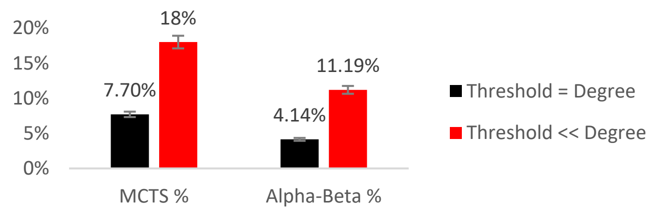

Fig. 6 Zero Loyalty – Ratio of selection of nodes when the cost of capturing a node is reduced (Threshold << Degree) or kept as it is (Threshold = Degree) for both types of agents.

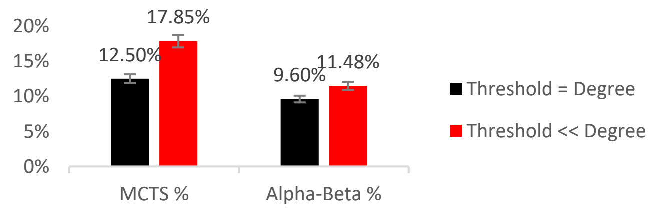

Fig. 7 Full Loyalty - Ratio of selection of nodes when the cost of capturing a node is reduced (Threshold << Degree) or kept as it is (Threshold = Degree) for both types of agents.

G. *Conclusions and Main Observations*

An interesting observation from the study is related to the role of central nodes (hubs) in the "Spread-It" game. While influencing the central hubs seem to be a good strategy in general as indicated in the literature (e.g., [23], [24]), this is not necessarily the case for the "Spread-It" game, and does not always results in a win. One reason why the MCTS method achieved a higher winning rate with respect to other agents is since the MCTS does not use any predefined rules. Instead, it adapts itself to the current network state and decides on the next moves based only on that. This observation provides some intuition on how decision-makers should act in situations represented by the "Spread-It" game in the Zero Loyalty variant - making decisions based on current states without predefining any strategy a priori. The superiority the MCTS method are also due to the relatively limited span of the Alpha-Beta agent. For this method, a depth limit needs to be set, thus, the entire development of the tree could not be fully anticipated in advance.

As a general note, it is quite reasonable and intuitive to believe that an allocation of budget to influential nodes would yield a high rate of winning. Nevertheless, in this study it was found that such a strategy does not yield a high winning rate in certain cases. When analyzing the nodes chosen by the winning strategy with respect to their influence (measured by their degree), it was found that the winning strategy has no preference for allocating tokens to nodes with higher centrality measures in the Zero Loyalty variant of the game, while in the Full Loyalty variant, in which extensive flips in the system's state are not possible, there was a clear preference by the winning agents towards the central nodes. These results contradict a common belief by which the nodes' influence can be derived from the network topology itself, relying mainly on their centrality measures and less on the dynamics of spreading. Indeed, in many simplified dynamics of spread (such as the Full Loyalty), the centrality measures of a node provide good predictions for its influence, but this is not necessarily the case under some realistic information spread dynamics [25]. Some of these dynamics, including the change of opinion that might occur in scenarios associated with products that compete in markets with low customer loyalty, are captured by the "Spread-It" model.

Another observation emerges from the similarity between "Spread-It" and the CFG. The CFG is based on the Abelian sandpile model [11], which is often used for studying self-organized criticality in dynamical systems that consist of a sequence of cascades. As in the Zero Loyalty version of the game, small perturbations in the system might result in significant changes of the entire system's states. This chaotic phenomenon might explain the need for an extended look-ahead estimation of future system states, as implemented by the MCTS method.

Finally, let us note that some aspects of the "Spread It" game, as a model of influence spread in social networks, should be highlighted. These aspects do not relate to the implemented AI solutions per-se but more to the contribution of the game itself as an abstract model of influence.

VI. Summary And Future Work

This work introduces a game, called "Spread-It," that captures influence spread dynamics within a social network. Several AI agents are analyzed to find an algorithm with a high wining rate in the game. A Monte Carlo tree search (MCTS) method is found to obtain the best winning rate among the studied agents. Not only that the MCTS method obtains a better winning rate, but also it reveals some interesting properties of the winning strategy. One such property is related to the role of central influencing nodes ("hubs") in the network. Conventional influence measures are often determined by the network topology. These measures include the node's Degree, its PageRank, and the Eigenvector Centrality. These are commonly used as valid measures for the nodes' potential influence within the network. Yet, they are found to be less effective in the scenario of flipping opinions dynamics, as characterized by the Zero Loyalty "Spread-It" model.

The reason for their inefficiency in the Zero-Loyalty game, is related to the fact that nodes might flip their loyalty several times, influencing the entire system state and generating cascades of influence trajectory changes that are long and hard to predict.

Future work on “Spread-It” could extend the current frame to a multi-player game or a game played not only on undirected graphs but also on directed ones. On such a digraph, the player can face new game phenomena, such as a “sink node” (having a zero *out-degree*) that ‘swallows’ tokens. One can also consider a probabilistic framework where probabilities are assigned to graph edges, by which tokens are fired on the node’s neighbors. In this paper, we limit our study of the game to a static network where the network’s structure does not change over time. A more complex approach will look at “dynamic” networks, where the topology of the network evolves over time. Another property to study in the future is networks with different modularity (i.e., networks with a presence of communities); these networks can be generated and benchmarked using, for example, the method developed by Lancichinetti et al. [26].

“Spread-It” is a simple game designed for studying the interactions involved in processes of collective decision-making. In such cases, different players try to influence decision-makers with known (and sometimes hidden) social connections. Flips in the opinion of a majority can then occur suddenly and unpredictably. These flips can be found in parliament decision processes, in daily commercial agreements or in jury trials verdicts, just to name few examples. While the unlimited flipping of opinions, as it exists in the theoretical model, is somewhat unrealistic, sequential flips of opinions can be found in several real-life cases. Since group’s decision-making processes have become the norm in many social studies, the implications of this work can be related to several group decisions processes.

Appendix A

Fig. 8 gives an example where the threshold of a vertex is not equal to its degree. In such cases the properties mentioned in subsection III.D.2 do not hold. In this graph,

— The degree of each vertex is 2 while the threshold is 1.
— The number of tokens in the graph (N) is 1.
— The number of nodes in the graph (n) is 3.
— The number of edges in the graph (m) is 3.

According to the “infinity” property: if $N > 2m - n$ (i.e. if $N > 3$), then the game reaches a state of infinity (by series of explosions). In the graph configuration below, the property does not hold, since if $N \geq 1$ the game is infinite without reaching a stable state. Accordingly, the token distribution $m - \frac{n}{2}$ property does not hold either.

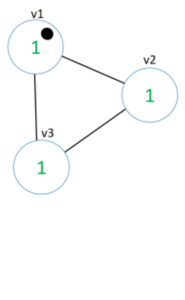

Fig. 8 Example of “Spread-It” graph where the node’s threshold is not equal to its degree.

Appendix B

The following outline summarizes how the learning algorithm works for finding the optimal weights for the Linear Combination (LC) heuristic. The learner has to learn a function V, called the *target function*, which is the ideal function for choosing the best move in each game state. The target function assigns a numerical score to any given game state, where higher scores are assigned to better game states and vice versa. For the final game states, the value of the target function is $V(s) = 100$ if the agent won the game; $V(s) = -100$ if the agent lost the game; and $V(s) = 0$ if the game ends in a draw, where s is the game state. For the intermediate states, an *approximation function* $\hat{V}$ is used, reflecting the real value of the game state. The LC heuristic function is an approximation of the target function V. The LC weights are modified throughout the learning process, converging the function to the desired state. In order to learn effectively, the agent needs feedback from a critic procedure. In this case the feedback is given indirectly by the value of a function called the training function V_{train}. For every state s, the training function can be expressed in the following way: $V_{train}(s) = \hat{V}(Successor(s))$, where $Successor(s)$ denotes the next game state following s for which it is again the learner's turn to move. The weights are recalculated every turn by using the Least Mean Square (LMS) technique which can be viewed as performing a stochastic gradient-descent search throughout the hypothesis space (weight values) to minimize the error E. This error represents the error made by the sum of the squared differences between the target function and the training function. Each turn, for every single training example, each weight is recalibrated using the following function: $W_i = W_i \cdot \eta \cdot (V_{train} - \hat{V}(s)) \cdot H_i$ where η is a small constant called the *learning rate* that moderates the size of the weight update; H_i is the *i*-th score of a particular heuristic, and W_i is the associated weight assigned to this heuristic score, as explained in section IV.

Twenty simulations were executed, where each simulation consists of 10,000 games. In each game the learner agent played against a second copy of itself on a random synthetic graph made up of 15-25 nodes (the size of the graphs was limited due to time constraints). Following each simulation, the weights vector, the number of wins, losses and draws were recorded. Of the twenty simulations, eight simulations which achieved at least a 70% wins rate by the first learner were selected. The average of the 8 final weight vectors was used to determine the final weight for each heuristic H.

Please refer to [17] for further details of the algorithm and the learning model.

Appendix C

TABLE VII shows how each Alpha-Beta agent with a specific heuristic performed against all the other agents including the MCTS and the Random agents (7 combinations per each heuristic) on different graph boards. The name of the Alpha-Beta agent (with the specific heuristic) listed in the agent column is the starting (black) player, while the second is the red player. The second and third columns show the win rate (WR%)

of an agent in the two game variants. Each player competed in 100 games per heuristic combination (a total of 700). These games were tested over 15 different graphs (a total of $15 \cdot 700 = 15{,}000$).

For further details on each heuristic, see subsection IV.A.1.

TABLE VII
ALPHA-BETA HEURISTICS RESULTS

Agent Heuristic	Zero Loyalty WR%	Full Loyalty WR%
Parity	37.7 ± 0.93	50.9 ± 0.96
Stability	14.1 ± 0.67	10.9 ± 0.60
General	38.4 ± 0.93	48.3 ± 0.96
Hubs	24.6 ± 0.82	28.6 ± 0.86
Lin Com.	49.4 ± 0.96	62.2 ± 0.93
Mobility	23.9 ± 0.82	21.8 ± 0.79

One can observe from these results that the Linear Combination (LC) heuristic (with the Alpha-Beta agent) achieved the highest win rate (WR%) in both game variants. In the Zero Loyalty variant, the LC heuristic reached ~50% WR, whereas other heuristics obtain win rates of under 40%. In the Full Loyalty variant, the LC agent reached ~62% WR, whereas other heuristics obtained win rates of under 50%.

REFERENCES

[1] A. Björner, L. Lovász and P. W. Shor, "Chip-firing games on graphs," *European Journal of Combinatorics,* vol. 12, pp. 283-291, 1991.

[2] A. Sela, "Information Spread in Social Networks (Unpublished Doctoral dissertation)," Tel-Aviv, 2017.

[3] B. L. Bayus, "Word of Mouth-the Indirect Effects of Marketing Efforts," *Journal of advertising research,* vol. 25, pp. 31-39, 1985.

[4] D. Kempe, J. Kleinberg and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[5] S. Bharathi, D. Kempe and M. Salek, "Competitive influence maximization in social networks," in *International Workshop on Web and Internet Economics*, 2007.

[6] T. Carnes, C. Nagarajan, S. M. Wild and A. Van Zuylen, "Maximizing influence in a competitive social network: a followerś perspective," in *Proceedings of the ninth international conference on Electronic commerce,* 2007.

[7] X. He, A. Prasad, S. P. Sethi and G. J. Gutierrez, "A survey of Stackelberg differential game models in supply and marketing channels," *Journal of Systems Science and Systems Engineering,* vol. 16, pp. 385-413, 2007.

[8] A. Clark and R. Poovendran, "Maximizing influence in competitive environments: a game-theoretic approach," in *Decision and Game Theory for Security*, Springer, 2011, pp. 151-162.

[9] N. Alon, M. Feldman, A. D. Procaccia and M. Tennenholtz, "A note on competitive diffusion through social networks," *Information Processing Letters,* vol. 110, pp. 221-225, 2010.

[10] V. Tzoumas, C. Amanatidis and E. Markakis, "A game-theoretic analysis of a competitive diffusion process over social networks," in *Internet and Network Economics*, Springer, 2012, pp. 1-14.

[11] P. Bak, C. Tang and K. Wiesenfeld, "Self-organized criticality: An explanation of the 1/f noise," *Physical review letters,* vol. 59, p. 381, 1987.

[12] J. Desel, E. Kindler, T. Vesper and R. Walter, "A simplified proof for a self-stabilizing protocol: A Game of Cards," *Information Processing Letters,* vol. 54, pp. 327-328, 1995.

[13] C. Godsil and G. F. Royle, Algebraic graph theory, vol. 207, Springer Science & Business Media, 2013.

[14] N. L. Biggs, "Chip-firing and the critical group of a graph," *Journal of Algebraic Combinatorics,* vol. 9, pp. 25-45, 1999.

[15] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial intelligence,* vol. 6, pp. 293-326, 1976.

[16] B. Brügmann, "Monte carlo go," Technical report, Max Planck Institute of Physics, 1993.

[17] T. M. Mitchell, "Machine Learning," in *Machine Learning*, McGraw-Hill Education, 1997, pp. 1-15.

[18] A. Levinovitz, *"The Mystery of Go, the Ancient Game That Computers Still Can't Win", WIRED,* 2014. [Online]. Available: https://www.wired.com/2014/05/the-world-of-computer-go. [Accessed: 09- Sep- 2017].

[19] E. Lasker, Go and Go-moku, Courier Corporation, 2012.

[20] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006*, Springer, 2006, pp. 282-293.

[21] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning,* vol. 47, no. 2-3, pp. 235-256, 2002.

[22] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, pp. 509-512, 1999.

[23] J. Goldenberg, S. Han, D. R. Lehmann and J. W. Hong, "The role of hubs in the adoption process," *Journal of Marketing,* vol. 73, pp. 1-13, 2009.

[24] E. Katz and P. F. Lazarsfeld, Personal Influence, The part played by people in the flow of mass communications, Transaction Publishers, 1955.

[25] D. J. Watts and P. S. Dodds, "Influentials, networks, and public opinion formation," *Journal of consumer research,* vol. 34, pp. 441-458, 2007.

[26] A. Lancichinetti, S. Fortunato and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E,* vol. 78, p. 046110, 2008.