HeX and the single anthill: playing games with Aunt Hillary

J.M. Bishop, S.J. Nasuto, T. Tanay, E. B. Roesch and M. C. Spencer

Abstract In a reflective and richly entertaining piece from 1979, Doug Hofstadter playfully imagined a conversation between 'Achilles' and an anthill (the eponymous 'Aunt Hillary'), in which he famously explored many ideas and themes related to cognition and consciousness. For Hofstadter, the anthill is able to carry on a conversation because the ants that compose it play roughly the same role that neurons play in human languaging; unfortunately, Hofstadter's work is notably short on detail suggesting how this magic might be achieved[1]. Conversely in this paper - finally reifying Hofstadter's imagination - we demonstrate how populations of simple ant-like creatures can be organised to solve complex problems; problems that involve the use of forward planning and strategy. Specifically we will demonstrate that populations of such creatures can be configured to play a strategically strong - though tactically weak - game of HeX (a complex strategic game). We subsequently demonstrate how tactical play can be improved by introducing a form of forward planning instantiated via multiple populations of agents; a technique that can be compared to the dynamics of interacting populations of social insects via the concept of *meta-population*. In this way although, *pace* Hofstadter, we do not establish that a meta-population of ants could actually hold a conversation with Achilles, we do successfully introduce Aunt Hillary to the complex, seductive charms of HeX.

J. Mark Bishop
Goldsmiths, University of London, UK, e-mail: m.bishop@gold.ac.uk

[1] As Drew McDermott writes in the Cambridge Handbook of Consciousness [24], it is as if Hofstadter "*wants to invent a new, playful style of argumentation, in which concepts are broken up and tossed together into so many configurations that the original question one might have asked get shunted aside*".

1 Swarm Intelligence

In recent years, studies of the behaviour of social insects (e.g. ants and bees) and social animals (e.g. birds and fish) have proposed several new metaheuristics for use in collective intelligence. Natural examples of such 'swarm intelligence' - whereupon apparently intelligent behaviour is realised via various forms of social interaction - include fish schooling, birds flocking, ant colonies in nesting and foraging, bacterial growth, animal herding, brood sorting etc.

Communication – social interaction or information exchange – as observed in social insects is important in all forms of swarm intelligence. In the study of interaction in social insects, two key elements are the individuals and the environment, which results in two modes of interaction: the first defines the way in which individuals interact with each other and the second defines the interaction of individuals with the environment [6]. Interaction between individual agents is typically carried out via agent recruitment processes and it has been demonstrated that various recruitment strategies are deployed by ants [19] and honey bees [15, 32]. These recruitment strategies may be used, for example, to attract other members of the population to gather around one or more desired areas in the search space, either for foraging purposes or in order to facilitate a colony relocation to a better nest site.

It has been observed that recruitment strategies in social insects may take several forms: localised or global recruitment; one-to-one or one-to-many recruitment; and may operate stochastically or deterministically. The nature of information exchange also varies in different environments and with different types of social insects. Sometimes the information exchange is quite complex and, for example, might communicate data about the direction, distance and suitability of the target; or sometimes the information sharing is relatively simple, for example, a stimulation forcing a particular triggered action. Nonetheless, what all recruitment and information exchange strategies have in common is an ability to distribute useful information across their community [11].

Chemical communication through pheromones forms the primary method of recruitment in many species of ants, however in one species, *Leptothorax acervorum*, a 'tandem calling' mechanism (one-to-one communication) is used. In this process, on its return to the nest, a forager ant that has found the resource location physically recruits a single ant and, by this action, the location of the resource is physically publicised [26] to the population.

Swarm intelligence, as the study of metaheuristics inspired by natural collective intelligence, is a relatively new branch of artificial intelligence that realigns intelligence away from the individual towards the collective; its aim is to illustrate intelligent behaviour by considering individuals in a social context and monitoring their interaction with one another as well as with their environment. Natural examples of swarm intelligence systems are: fish-schooling, bird-flocking, animal herding, nesting and foraging in the social insects etc. In recent years, abstractions of such natural behaviour have motivated several new Swarm Intelligence heuristics. While in typical Swarm Intelligence algorithms only the syntactical exchange of information is considered, in many natural social interactions, it is not just syntactical information,

but also semantic rules and beliefs about how to process this information, that is exchanged [20].

The simple and often successful deployment of swarm based heuristics on traditionally difficult optimisation problems has generated significant interest (cf. Dorigo et al [13] [12]; Kennedy [21]); nonetheless, to date, they have merely been deployed on conceptually straightforward optimisation and regression problems.

This paper is organised in the following manner. In section (2) we introduce the game of Hex. In section (3) we introduce a Monte Carlo Stochastic Diffusion Search (MCSDS), a swarm intelligence algorithm for playing Hex based on a simple merger of Stochastic Diffusion Search (SDS) [4] and Monte Carlo methods [25]. Subsequently extending MCSDS in Section (4), we introduce a more sophisticated algorithm, Stochastic Diffusion Search applied to Trees (SDST); a novel swarm intelligence heuristic able to solve the complex and general problem of forward planning in a way analogous to Monte-Carlo Tree Search (MCTS) [1].

In SDS and MCSDS, direct one-to-one communication (which is similar to the tandem calling recruitment mechanism described earlier) is utilised[2]. In SDST, each individual agent processes information concerning a unique action without "awareness" of the way in which actions are being compared and combined. Yet the dynamics of the entire population of agents lead to a high level "reasoning" about successions of actions analogous to Monte-Carlo Tree Search (MCTS). In its functioning, SDST is argued to introduce a meta-level in the swarm intelligence paradigm.

Although some previous attempts have been made to apply decentralised methods to forward planning tasks, such methods did not reach the same degree of generality as SDST. For example Tesauro developed in 1989 a neural network program playing Backgammon (a non-deterministic finite two-person zero-sum game with perfect information) better than any other program (the program called Neurogammon won the backgammon competition of the First Computer Olympiad). However, Tesauro explicitly expressed in the introduction of [17] that "the game of backgammon in particular was selected because of the predominance of judgement based on static pattern recognition, as opposed to explicit look-ahead or tree-search computations."

By presenting SDST, our objective is to extend the applicability of parallel and distributed models of computation (and in particular SDS) to solve problems that were historically exclusively addressed with a sequential algorithmic approach requiring centralised control and access to the data. For the sake of simplicity and clarity, and because it is the problem for which it was originally conceived, SDST is presented in the context of combinatorial games (finite two-person zero-sum games with perfect information such as Chess). However, the discussion is entirely consistent with any planning task that can be represented as a tree of sequential decisions. Along the way we will illustrate how Hofstadter's 'Aunt Hillary' might beat her old friend the 'Ant Eater' at Hex.

[2] Although the recruitment behaviour of real ants is more complex than the behaviour in SDS, both are population-based and find their optima via agents communicating with each other.

2 The Game of Hex

Hex is a combinatorial game that belongs to the family of connection games. It is an example of a game that has been solved in a non-constructive way: it is proved that the first player has a winning strategy, but the strategy in question is not known. The proof of this result is given in the following sections, after the rules of the game have been presented. The game of Hex was chosen for three main reasons:

- First, it is relatively simple and well-suited to perform rapid random game evaluations: the only action available to each player at each turn is to put one stone of his colour on the board.
- Second, the size of the board is changeable without modifying the nature of the game. This allows the algorithm to be tested on different sizes of game-trees.
- Third, it is mathematically elegant and has interesting properties in game theory.

2.1 Rules and History

The game of Hex was first invented in 1942 by the Danish scientist, artist and poet Piet Hein. It was then independently reinvented in 1947 by the American mathematician John Nash, while still a student at Princeton University. It was first known under the name *Polygon* in Denmark and was called after its creator *John* or *Nash* at Princeton University, before Parker Brothers marketed a version under the name *Hex* in 1952.

The game is played on a rhombic board covered with hexagonal cells. Each of the two players is associated with a colour (blue and red in the following) and two opposite sides of the board (for example top-right and bottom-left for blue and top-left and bottom-right for red). The rules are extremely simple: the two players alternatively place a stone of their colour on a single cell within the entire board and try to form a connected path between their two sides. The game ends when one of the two players managed to build such a connection. The usual size of the board is 11×11, but due to the relationship that Hex maintains with Go, the sizes 13×13 and 19×19 are also common. According to Sylvia Nasar's biography of John Nash *A Beautiful Mind*, he recommended 14×14 as the optimal size. Figure 1 shows a typical finished game: Red wins because he managed to connect his two sides of the board.

2.2 Game Theory

The proof of the existence of a winning strategy for the first player relies on two central points: the fact that there can be no draw in Hex, and the strategy stealing argument. A relatively simple proof of the first point is given in a paper from David Gale: "The game of Hex and the Brouwer fixed-point theorem" and is outlined in

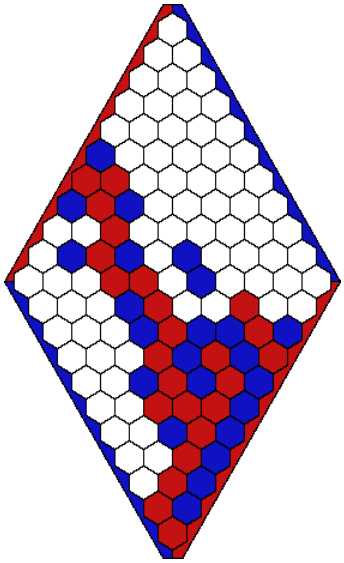


Fig. 1: A typical finished game at Hex (*From Wikimedia Commons*)

the next section. Interestingly, the same paper establishes an equivalence between this proof and the Brouwer fixed-point theorem, an important theorem in topology that states that for any continuous function f with certain properties there is a point x_0 such that $f(x_0) = x_0$.

2.2.1 One and only one winner

If the blue player is called x and the red player is called o, and if the two sides of the board corresponding to the blue player are called X and X' and the two sides corresponding to the red player are called O and O', Gale states what he calls the Hex theorem as follow:

Hex theorem: If every tile of the Hex board is marked either x or o, then there is either an x-path connecting regions X and X' or an o-path connecting regions O and O', but not both. [14]

In the original paper, Gale gives a very intuitive illustration of the theorem: "Imagine, for example, that the X-regions are portions of opposite banks of the river "O" (...) and that the x-player is trying to build a dam by putting down stones. It is quite clear that he will have succeeded in damming the river only if he has placed his stones in a way which enables him to walk on them from one bank to the other." In other words, the only way one has to prevent his opponent from winning is by winning himself and thus there is always a winner. Then Gale continues his analogy: "if the x-player succeeds in constructing a causeway from X to X', he will

in the process have dammed the river and prevented any flow from O to O'." In other words, if one of the players wins he prevents his opponent from winning at the same time and there can be only one winner. Although the theorem is intuitive, the proofs of the two results (the existence and the uniqueness of a winner) are rather delicate. The uniqueness directly follows from a fundamental result of topology called the Jordan Curve Theorem. This theorem asserts that every non-self-intersecting continuous loop divides the plane in an "interior" region and an "exterior" region so that any continuous path connecting a point of one region to a point of the other intersects with that loop somewhere. Although this theorem is also very intuitive, it is difficult to establish formally.

A constructive proof of the existence of a winner is given in [14]. It is outlined in the first part of the paper as follows (figure 2 is the corresponding figure on which colours have been added for clarity):

> "We consider the edge graph Γ of the Hex board to which additional edges ending in vertices u, u', v, v' have been added to separate the four boundary regions, as shown in the figure. We now present an algorithm for finding a winning set on the completely marked board. We shall make a tour along Γ, starting from the vertex u and following the simple rule of always proceeding along an edge which is the common boundary of an X-face and an O-face. Note that the edge from u has this property since it separates regions X and O. The key observation is that this touring rule determines a unique path; for suppose one has proceeded along some edge e and arrives at a vertex w. Two of the three faces incident to w are those of which e is the common boundary, hence one is an X-face, the other an O-face. The third face incident to w may be either an X-face or an O-face, but in either case there is exactly one edge e' which satisfies the touring rule."

The tour constituted by this algorithm is highlighted in yellow in figure 2 (it starts at vertex u at the bottom and ends at vertex v on the left). Two characteristics of this tour show that there necessarily is a winner:

1. It will never revisit any vertex. The reason for this is that by construction, the degree of every vertex of the tour is at most two. But since the degree of the first vertex of the tour is one (vertex u), the tour must constitute a simple path and end on a vertex of degree one. The only possibilities are u', v or v'.
2. It cannot end on vertex u' (top one). The reason of this is that the tour starting at u always keep blue cells on the left and red cells on the right while the edge ending at u has a red cell on its left and a blue cell on its right[3]. Hence the tour can only end on v or v'.

To conclude, one only needs to notice that if the tour ends on v the red cells on its right form a winning path for o and if it ends on v' the blue cells on its left form a winning path for x. Hence, there is always a winner at Hex.

[3] The rigorous proof is not based on this left-right consideration: "this would involve getting into the quite complex notion of orientation, which is not needed for our proof." [14]

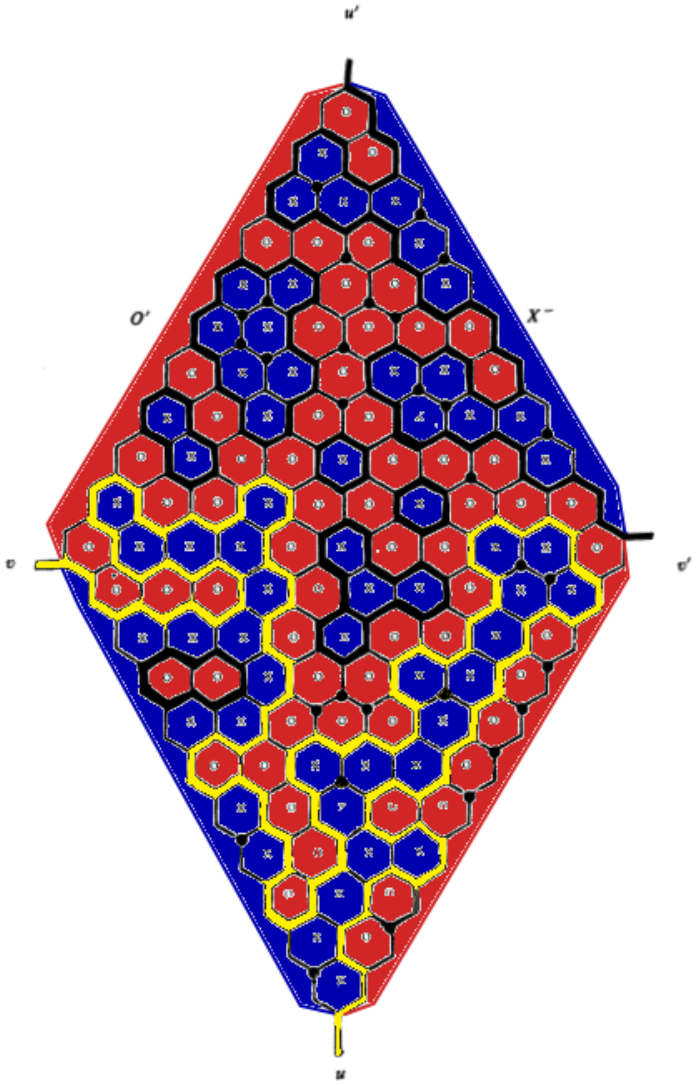


Fig. 2: Illustration of the tour followed by Gale's algorithm on a full Hex board (starting at u and finishing at v in yellow). The red cells on its right form a winning path for o. (*From [14], colours added.*)

2.2.2 The Strategy Stealing Argument

The absence of draw at Hex has an interesting consequence if one remembers Zermelo's theorem: according to [17], Zermelo's theorem states that "in Chess, either White can force a win, or Black can force a win, or both sides can force a draw." Now it can be maintained that in Hex, either Blue can force a win or Red can force a win. In fact, an ingenious reductio ad absurdum from John Nash called the strategy stealing argument proves that this is the first player who has a winning strategy. The argument goes as follow: let one suppose that there exists a winning strategy for the second player. In this case, the first player can steal this strategy to build his own winning strategy in the following way. First he places one of his stones anywhere on the board and let the second player play as if he was the first one. Then he follows the second player's winning strategy until either the game finishes, or the winning strategy tells him to play the move he played first. In the second case, he just plays anywhere and starts following the winning strategy again the next turn. In this situation both players have a winning strategy which is contradictory and the initial hypothesis that the second player has a winning strategy is false. It is important to note here that the stealing strategy argument only holds because it is never a disadvantage to play a move at Hex. However this is not always the case: in Chess there are situations called *Zugzwang* in which every move leads to a worse and often lost position (they happen most of the time in late endgames).

Of course, the strategy stealing argument is non-constructive and the winning strategy for the first player is not known for boards bigger than 9×9 cells. Yet, in practical play, it appears that playing first does constitute a great advantage. To compensate this bias, the swap rule allows the second player to choose between either playing normally, or taking the first player's position after his first move. This re-equilibrates the game because in this case the first player should play neither the strongest moves (such as the centre of the board) because the second player would switch its position with him, nor the worst moves (such as the two cells in the acute angles of the rhombus) because the second player would leave them to him. In the presence of the swap rule this is the second player who has a winning strategy since he can choose between taking the first player's move if it is a winning one or leaving it if it is a loosing one (although this information is not known in practical play).

3 Playing games with Aunt Hillary

The work presented in this section rests on two pillars: a swarm intelligence metaheuristic for search and optimisation called Stochastic Diffusion Search (SDS) and the Monte-Carlo search method. These two techniques are briefly outlined in the following subsections.

3.1 Stochastic Diffusion Search (SDS)

SDS is an efficient probabilistic swarm intelligence global search and optimisation technique that has been applied to diverse problems such as site selection for wireless networks [35], mobile robot self-localisation [3], object recognition [5] and text search [4]. Additionally, a hybrid SDS and n-tuple RAM [2] technique has been used to track facial features in video sequences [5, 16]. Previous analysis of SDS has investigated its global convergence [31], linear time complexity [30] and resource allocation [27] under a variety of search conditions.

SDS is based on distributed computation, in which the operations of simple computational units, or agents are inherently probabilistic. Agents collectively construct the solution by performing independent searches followed by diffusion of information through the population. SDS relies on two principles: partial evaluation of hypotheses and direct communication between agents. The SDS algorithm is characterised by three phases: Initialisation, Test and Diffusion—the test and diffusion phases are repeated until a Halting criterion is reached. During the initialisation phase each agent formulates a hypothesis, i.e. chooses a potential solution in the search space. During the test phase each agent partially evaluates its hypothesis: agents for which the partial evaluation is positive become active, and the others become inactive. During the diffusion phase, agents exchange information by direct communication: each inactive agent X contacts an agent Y at random. If Y is active, X takes its hypothesis, otherwise X formulates a new hypothesis at random (procedure called passive recruitment). In practice, a halting criterion needs to be defined to stop the algorithm running; the properties of convergence of SDS led to the definition of two criteria, a weak and a strong version [31].

3.2 Monte-Carlo Stochastic Diffusion Search (MCSDS)

The starting point in applying Monte Carlo methods to SDS is the simulation of random games a great number of times. The suitability of Hex to perform random game simulations was one of the main reasons to select this game as a study case. Indeed, a random game just consists in alternatively placing red and blue stones on the board until it is full. Although this is relatively simple, some care has to be given to fill the board with a uniform distribution or the evaluation of the moves could be biased.

A simple improvement that can be given to the standard Monte-Carlo algorithm is to sample the evaluation of the cells by applying a 'multi-armed bandit' analogy': the moves that tend to give good results at the beginning of the evaluation should receive more attention than the moves that appear to be bad. This can be achieved by applying SDS.

First, a population of agents with hypotheses about the best move to play is initialised. Second, the hypotheses are tested by performing a random game simulation: if the outcome is a win the agent becomes active, otherwise stays inactive.

Third, every inactive agent selects at random another agent for communication. If the selected agent is active, the first agent copies its hypothesis, but if the selected agent is inactive, the first agent chooses a new hypothesis at random (passive recruitment strategy).

It is important to notice that this 'improvement' only concerns the speed of the process; theoretically the value attributed to each move is the same as in the standard version. Indeed, the value of a move is still assimilated to the probability that it leads to a win given random play, and this probability is not changed by the way evaluation is balanced between the different moves. Hence using SDS in this way has exactly the same 'level of play' as standard Monte-Carlo - it simply uses SDS as an efficient Swarm Intelligence resource allocation technique.

3.3 Hex and the single ant hill

Hofstadter [18] imagines Ant Hillary and the Ant Eater conversing as follows:

> ANTEATER: . . . Aunt Hillary and I have conversations for hours. I take a stick and draw trails in the moist ground, and watch the ants follow my trails. Presently, a new trail starts getting formed somewhere. I greatly enjoy watching trails develop. As they are forming, I anticipate how they will continue (and more often I am wrong than right). When the trail is completed, I know what Aunt Hillary is thinking, and I in turn make my reply.

In this paper we do not, *pace* Hofstadter, illustrate Aunt Hillary in communication with the Ant Eater; we do however, successfully introduce Aunt Hillary (AH) to the complex, seductive charms of HeX. To do this we simply need to show that Aunt Hillary can perform the Monte-Carlo SDS Hex algorithm outlined in Section 3.2.

In nature, ants produce numerous different pheromones, each with its own distinct purpose. For example, ants secrete pheromones to attract mates, to signal danger to the colony, or to give directions about a location. Ant Hillary deploys pheromones to represent information and, in this way, can play a game of Hex against the Ant Eater (AE) as follows.

Aunt Hillary maintains a group of red ants to identify hypotheses about the potential moves available at the current stage of the game; a *partial evaluation* of each move is constituted by the result of a random game performed assuming a hypothesized move has been played; after which, SDS diffuses 'successful' hypotheses through the population. After a number of these hypothetical plays have been evaluated, red ants will tend to cluster around moves that have the highest probabilities of leading to a win. This process constitutes the simplest application of MCSDS to the problem of computer game-playing and is easily implemented as a simple 'ant algorithm' that Aunt Hillary can physically enact.

Aunt Hillary first divides the ants she will deploy in the game into three types, defining *hypothetical-moves, random-plays and actual-moves*:

- a population of k red 'H-ants' - each of which maintain a *Hypothesis* suggesting Aunt Hillary's next best move onto an unoccupied position on the board; each H-ant carries a unique pheromone representing that H-ant's hypothesis;
- associated with each H-ant are two equal sized groups of g[4] black and white 'R-ants'; each R-ant representing a potential legal *Random* play from either Aunt Hillary [black] or Ant Eater [white]. Each R-ant is uniquely associated with a particular hypothesis by carrying that H-ant's pheromone. Once a 'hypothetical move' has been made by Aunt Hilary, successive deployment of associated (black and white) R-ants thus demarcate positions on the board with successive *random moves* played by the Ant Eater and Aunt Hillary;
- two equal sized groups of g black and white 'M-ants'; each M-ant representing an actual played *Move* from either Aunt Hillary [black] or Ant Eater [white]. Black and white M-ants thus demarcate positions on the board with the successive *actual moves* played by either Aunt Hillary or the Ant Eater.

Aunt Hillary initially allocates the red H-ants with a random hypothesis, i.e. a hypothetical 'next-play' selected randomly from the current set of legal moves available on the board. When it is her turn to move, Aunt Hillary 'thinks' until her thinking time is up, after which, she moves to the position defined by the most popular hypothesis (i.e. the hypothesis carried carried by most red H-ants).

Aunt Hillary subsequently makes this move by positioning a black M-ant to this board position. The Ant Eater then makes its play and AH marks its move by positioning a white M-ant to demarcate that board location. After a finite sequence of such turns - as Nash proved - either Aunt Hillary will have won (established a continuous path [linking AH's M-ants] from left to right across the board) or the Ant Eater will have won (established a continuous path [linking AE's M-ants] from the top to the bottom of the board).

In performing her 'thinking' Aunt Hillary merely iterates the following two-step parallel procedure until her 'thinking time' is over and a move is obliged.

1. *Evaluate Monte-Carlo games*. For each of the [red H-ant] hypotheses Aunt Hillary performs a Monte-Carlo simulation and plays a 'random game'. To do this, the population of black and white R-ants [associated with each focal H-ant hypothesis] take turns to play randomly selected legal moves - whereby a (black / white) R-ant walks to a random position (unoccupied by either an R-ant associated [with this hypothesis] or a previously played M-ant - until either Aunt Hillary has won (established a continuous path [linking AH M-ants, the focal H-ant and black R-ants associated with the focal ant's hypothesis] from left to right across the board) or the Ant Eater has won (established a continuous path [linking AE M-ants and the white R-ants associated with each focal hypothesis] from the top to the bottom of the board). If the random game resulted in a win for the Ant Eater then the focal H-ant hypothesis is deemed *inactive*; if the result was a win for Aunt Hillary the focal hypothesis is *active*.

   In performing each random move, a R-ant randomly walks around the board for a random time period (t) after which it stops at the first position not occupied by either an ant baring the same hypothesis pheromone or a M-ant (demarcating a previously played

[4] Group size = g = (board area DIV 2) + 1.

move).

2. *Diffuse hypotheses*. Each *inactive* red H-ant randomly selects another H-ant; if that H-ant is active, it transfers its hypothesis (active H-ant → inactive H-ant) otherwise the inactive H-ant selects a new hypothesis at random from the unoccupied board positions. To physically perform such 'hypothesis diffusion', each H-ant [hypothesis] merely moves randomly around the board for a random time period (t) after which it continues to move randomly until it alights onto a position occupied by another red H-ant. If this H-ant is *active*, it stays at this position; if it is *inactive*, it continues to move randomly stopping as soon as it alights onto a position not occupied by a previous move (as demarcated by the presence of an M-ant)[5]. In other words, if diffusion did not occur the H-ant selects a new hypothesis at random by simply randomly moving around the board for a random time period (t), after which it stops at the first position not occupied by a M-ant.

NB. As the pheromones carried of each of the H-ants (and their associated group of R-ants) are distinct, the Monte-Carlo games for all of the hypotheses and diffusions processes can be evaluated in parallel; *thus Aunt Hillary's 'thinking' is characterised by a seething mass of randomly moving ants*.

3.4 Analysis

While both the standard Monte-Carlo program and MCSDS show relatively good strategical sense and always perform moves that increase the overall chance to win, both play poor tactically. In early evaluation trials, it was established that the MCSDS algorithm offered good performance on a 7×7 Hex board against a naive random opponent, but poor tactical play against a more skilled opponent.

An illustration of one of the tactical weakness of MCSDS concerns situations called 'bridges'. In Hex, a bridge situation occurs whenever a player cannot stop the other player from connecting two groups of stones in one move because there are two ways to do it (see figure 3) . When a bridge is formed for one player, the best tactic for the other player is to play somewhere other than closing the bridge. However, the MCSDS algorithm (as deployed by Aunt Hillary) is unable to reliably play this way; to do so would require the anticipation of potential next moves from the opponent (see figure 3).

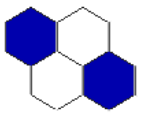
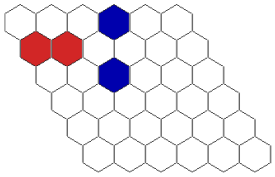
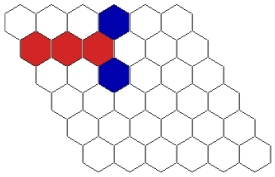


Fig. 3: A bridge for Blue: Red cannot stop Blue from connecting the two blue cells on the next move.

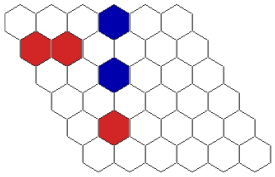
[5] A process isomorphic to asynchronous passive recruitment SDS [9].



(a) An hypothetical game situation. It is Red to move.



(b) The move that the standard Monte-Carlo program would perform: it increases its chances to win if the rest of the game is played randomly.



(c) Example of a better tactical move.

Fig. 3: Illustration of the weakness in tactic peculiar to the standard Monte-Carlo program.

To improve the tactical play, a metapopulation [23] of ants is deployed to enable a Monte-Carlo informed SDS to tactically explore the game tree[6]: Stochastic Diffusion Search applied to Trees (SDST).

3.5 Monte-Carlo Tree Search (MCTS)

MCTS "is a recently proposed search method that combines the precision of tree search with the generality of random sampling" [7]. To date, over 350 papers related to MCTS have been published, with applications ranging from computer Go to Constraints Satisfaction problems through Reinforcement Learning and Combinatorial Optimisation. Browne et al. [7] offer a complete survey of the published work on MCTS (until 2011) and conclude that MCTS "has already had a profound impact on Artificial Intelligence (AI) approaches for domains that can be represented as trees of sequential decisions, particularly games and planning problems".

MCTS has originally been developed in the context of computer game playing and finds its roots in B. Abramson's 1990 paper *Expected-outcome: a general model of static evaluation* [1]. This paper introduces the central Monte Carlo theme to

[6] A 'game tree' is a directed graph whose nodes are positions in a game and whose edges are moves. The complete game tree for a game is the game tree starting at the initial position and containing all possible moves from each position; a n-ply game tree describes all possible move/counter-move combinations to a depth of n moves.

evaluate a game position by playing a large number of random games from that position, assuming that a good move must increase the expected outcome of the player[7]. The second decisive step in the development of MCTS was the publication in 2006 of Kocsis and Szepesvári's paper *Bandit based Monte-Carlo Planning*. In this paper the 'Upper Confidence bound applied to Trees' (UCT) heuristic is introduced; a method that "applies bandit ideas to guide Monte-Carlo planning". The crux of UCT is to choose the moves to be evaluated at each node of the game-tree according to the information already collected during previous evaluations, in order to better exploit the most promising areas of the tree. Standard MCTS consists in iteratively building a "search-tree" (the root node of which is the current position) and is outlined in [8] as a succession of four phases: Selection, Expansion, Simulation and Backpropagation. In practice, the four phases are repeated until a given computational budget is spent (usually the time), at which point a decision is made and a move is played.

The moves to be evaluated are first chosen in the existing search-tree from the root in a way that balances between exploration of the available moves and exploitation of the most promising ones (*selection*): the policy used to choose the moves during this phase is called the "tree policy" and this is where [22] introduced the analogy between a node of the search-tree and a multi-armed bandit.

When a leaf of the search-tree is reached, the rest of the game is played up to a final state (*simulation*). The policy used during this phase is called the "default policy" and can be purely random in the simplest implementations of MCTS. The first move chosen by the default policy is then added to the search-tree (*expansion*).

Finally, the statistics of each node crossed during the selection phase are updated according to the outcome of the simulated game (*backpropagation*).

The way MCTS works is rather intuitive and it is argued in [7] that "the forward sampling approach is, in some ways, similar to the method employed by human game players, as the algorithm will focus on more promising lines of play while occasionally checking apparently weaker options." An important property of MCTS is its asymptotic convergence to Minimax, i.e. it is assured to select the best move available if enough time is given (albeit the convergence to Minimax can take a very long in practice).

4 Stochastic Diffusion Search Applied to Trees (SDST)

Conceptually, the application of SDS to game-tree exploration is a two step process. First, each node is attributed a distinct and independent *local population* of agents to solve the problem of move selection on that node. Second, a reallocation policy is used to move the uncontacted agents toward more interesting regions of the game-

[7] This assumption is not necessarily a good one due to the distinction between random play and optimal play - see analysis of standard Monte-Carlo methods (and MCSDS) in Section (3.4).

tree—thus leading to the formation of a dynamically moving *metapopulation*[8] of agents[9].

4.1 First step: use of multiple populations of agents

The first step toward implementing SDST is to use SDS to solve the "exploration-exploitation dilemma" appearing during the selection phase of MCTS at each node of the search-tree. An algorithm detailing this idea is given in Table 1 (in SDS terms).

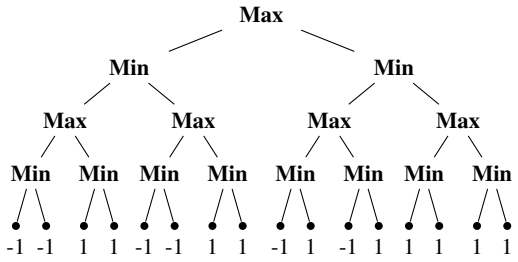
Table 1: First application of SDS to game-tree exploration: use of multiple populations of agents.

Initialisation	During the initialisation phase, a local population of agents is generated for each node of the game-tree up to a fixed depth D. For each local population, agents' hypotheses are initialised to a possible move of the corresponding node.
Test	During the test phase, a complete hypothesis is formed for each agent in the local population corresponding to the root node (later called root node population). This is done by combining agents from different local populations in a way analogous to the selection phase in MCTS: for each agent X in the root node population, an agent Y in the local population pointed by X's hypothesis is selected. Then an agent in the local population pointed by Y's hypothesis is selected, etc, until depth D is reached. Once a hypothesis is formulated, a simulation is run (in the MCTS sense) and the activities of the agents forming the hypothesis are updated according to the node they belong to (step corresponding to the backpropagation in MCTS): if the simulation leads to a win for Max, the agents in populations corresponding to Max's nodes become active and the agents in populations corresponding to Min's nodes become inactive (if it leads to a loss, it is the contrary).
Diffusion	During the diffusion phase, each local population acts independently, i.e. a diffusion phase is undertaken in the sense of Standard SDS without communication with other local populations.

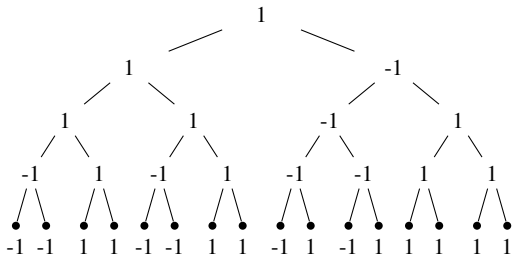
[8] The term was coined by Levins in [23] to describe the dynamics of interacting populations of social insects.

[9] The initial motivation for the work on SDST was to extend the applicability of Stochastic Diffusion Search (SDS) to more complex search spaces, and combinatorial games were chosen as a first study case. Then, Monte-Carlo Tree Search (MCTS) came naturally as a good framework for several reasons. First, MCTS does not rely on domain knowledge but rather on a large number random game simulations and the notion of random game simulation fits well with the concept of partial evaluation in SDS. Second, the strength of MCTS relies on the tree policy balancing between exploration of the search space and exploitation of the promising solutions and SDS is a metaheuristic precisely conceived to solve this "exploration-exploitation dilemma" in the management of the computational resources. Finally, MCTS has proven very successful in a wide range of problems—not only game playing—and is still under active study.

The operation of this algorithm is illustrated in Figure 5 on the small game-tree presented in Figure 4. The studied game-tree has been specifically designed to reveal the ability of the algorithm to converge to minimax and escape local optima: while a monte-carlo evaluation of the left and right moves for Max at the first ply would respectively lead to 50% and 75% chances to win—thus suggesting that the right move is better—the minimax resolution of the game-tree actually shows that, *if the players play optimally*, the left move leads to a win for Max (whatever Min plays at the second ply, the right move for Max at the third ply leads to a win) while the right move leads to a loss (if Min plays his left move at the second ply, whatever Max plays for the third ply leads to a loss with Min playing the left move at the fourth ply).



(a) Studied game-tree



(b) Minimax resolution

Fig. 4: Studied game-Tree. The minimax resolution shows that Max is the winner if he plays optimally.

Figure 5 shows that during iterations 1 and 2, most of the agents in the root node population point toward the right move. Then during iterations 3 and 4, the selection of Min's left moves at plies 2 and 4 changes this tendency and at iteration 5 all the agents in the root node point toward Max's left move—the best move in the minimax sense. Figure 5 simply illustrates that, as any other MCTS with a different tree policy, the algorithm presented here converges to minimax (provided that every non-terminal node of the game-tree is being attributed a population of agents).

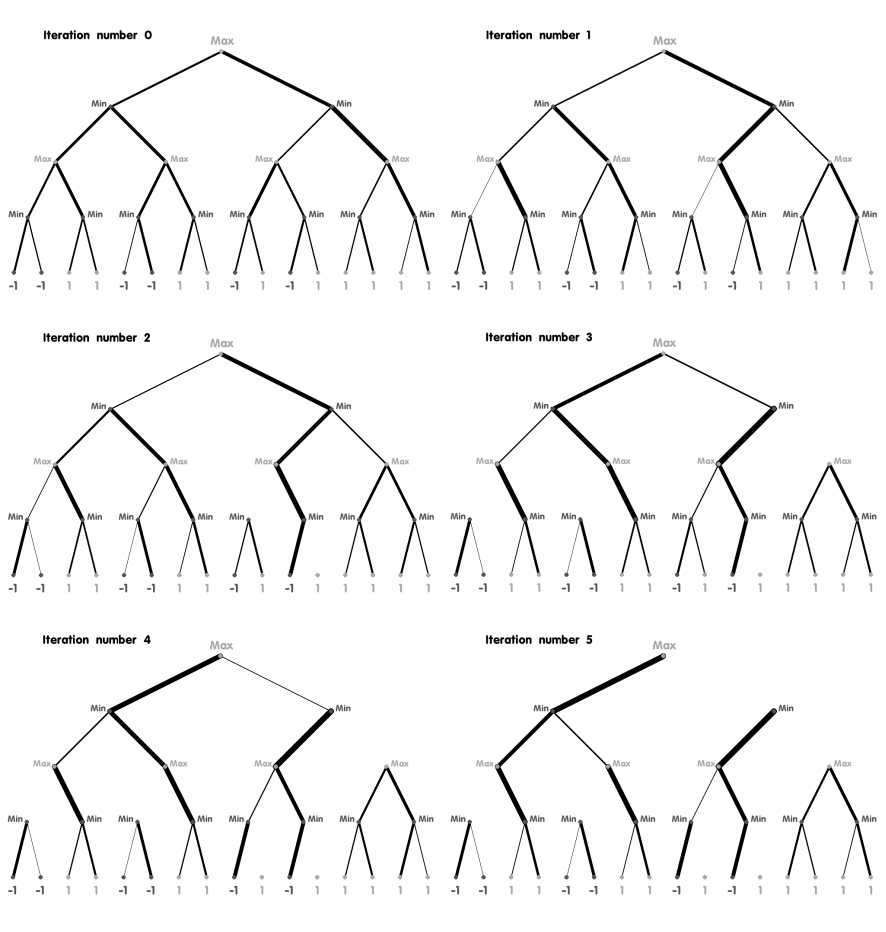


Fig. 5: Illustration of the algorithm presented in Table 1: Evolution of the distribution of the agents in the different nodes of the studied game-tree (first 5 iterations shown, total number of agents = 175). Each branch has an area proportional to the number of agents in the parent node population supporting the move corresponding to the child node population.

4.2 Second step: use of a reallocation policy

Although the previously discussed algorithm is shown to solve the problem of game-tree exploration, it suffers from two main drawbacks. First, the number of studied nodes in the game-tree and the number of agents per node need to be fixed manually in a very artificial way. Second, a uniform repartition of the agents in the initialisation phase rapidly leads to many agents being uncontacted in some branches (for example, all the agents on the right side of the tree become useless after the fifth iteration in Figure 5).

These drawbacks can be solved with the use of a reallocation policy where agents are scattered in the tree from the root node and uncontacted agents are backscattered toward parent nodes. SDST uses such a reallocation policy, defined naturally as described in Table 2.

Table 2: Stochastic Diffusion Search applied to Trees (SDST).

Initialisation During the initialisation phase, all the agents are allocated to the root node population and their hypotheses are selected randomly among the available moves.

Test During the test phase, complete hypotheses are formed. For each agent X in the root node population, an agent Y in the local population pointed by X's hypothesis is selected. Then an agent in the local population pointed by Y's hypothesis is selected, etc, until the local population pointed by the last agent is empty. Once a hypothesis is formulated, a simulation is run and activities of the agents forming the hypothesis are updated.

Diffusion For each local population, the diffusion phase is divided in three subphases:

1. *Backscattering:* the agents that were not contacted to form a hypothesis go back in the parent node population. In order to preserve the hypotheses distribution among the different moves in the parent node population, a backscattered agent chooses its new hypothesis not randomly but by copying the hypothesis of a chosen agent in that population.
2. *Scattering (by active recruitment):* every active agent X selects another agent Y at random; if Y is inactive, it is sent in the local population pointed by X's hypothesis. Similarly to the backscattering subphase, in order to preserve the hypotheses distribution in the host node population, the scattered agent selects its new hypothesis not randomly but by copying the hypothesis of a chosen agent in that population (if there are no agents at all in the host node population, then the new hypothesis is chosen randomly).
3. *Internal diffusion (by passive recruitment):* every inactive agent X selects another agent Y at random; if Y is active, X takes Y's hypothesis.

SDST is illustrated in figure 6 on the studied game-tree. As for the previously discussed algorithm, a majority of agents in the root node population first points toward the 'move right' (the best move in a purely standard Monte-Carlo sense) before reorienting toward the 'move left' (the best move in the minimax sense). However, the distribution of the agents in the entire metapopulation is now dynamically regulated: most of the agents diffuse in the right part of the game-tree in the first four iterations, and then diffuse back to the left part of the tree in the following iterations. Also, only the regions of interest are visited: for example the entire region after Max's right move at the first ply and Min's right move at the second ply is ignored because the entire subtree leads to a win for Max (no agent becomes active in Min's node population to send inactive agents in this area).

Under normal conditions, an equilibrium between the scattering and backscattering forces eventually appears, leading to a statistically stable metapopulation. A very interesting property of SDST is that this equilibrium depends on the number of

agents used. Asymptotically if enough agents are used, the equilibrium is equivalent to minimax. This is the case of the simulation presented in figure 6: at iteration 12 the metapopulation stabilises in the left part of the game-tree.

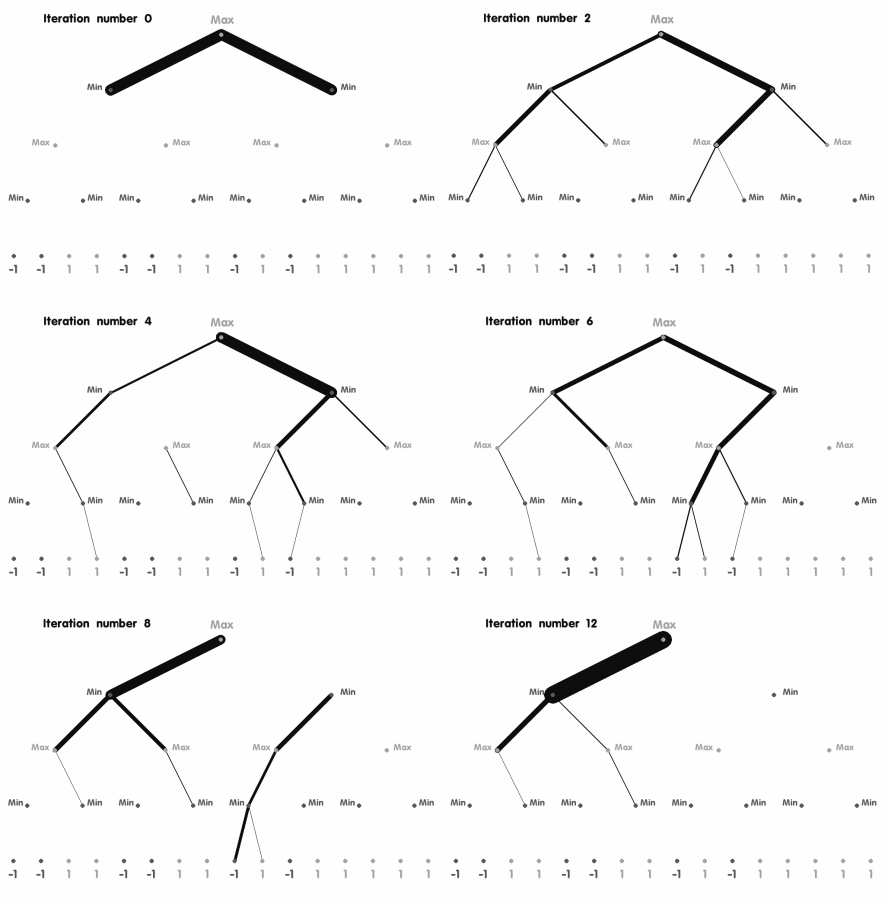


Fig. 6: Illustration of SDST: Evolution of the distribution of the agents in the entire game-tree (iterations 0, 2, 4, 6, 8 and 12 shown, total number of agents = 100). Each branch has an area proportional to the number of agents in the parent node population supporting the move corresponding to the child node population.

5 Conclusion

In the first half of this paper, we demonstrated that a simple application of Monte-Carlo methods to Stochastic Diffusion Search could enable a population of agents (very simple ant-like creatures; our eponymous Aunt Hillary) to play a *strategically informed* game of Hex, although it was also demonstrated that such a system is incapable of *tactically informed* play.

To improve tactical play, it is necessary to facilitate a form of forward planning; the latter half of the paper describes how this can be achieved using (a) a metapopulation of simple ant-like agents to represent a minimax game tree and a novel swarm intelligence heuristic to explore this representation and identify good tactical moves. This heuristic we term Stochastic Diffusion Search applied to Trees (SDST).

SDST is very similar to classical Monte-Carlo Tree Search (MCTS) algorithms in its functioning, but conceptually is radically different. While classical MCTS requires a central processing unit executing the algorithm in a sequential way (with a permanent and complete access to the data), the problem solving ability in SDST emerges from the collaboration of a swarm of homogeneous *ant-like* agents with limited computational capacities.

In addition to developing SDST, our research introduces *meta-level processing* to the Swarm Intelligence paradigm as SDST relies on emergence – both at the level of the agents forming local populations and at the level of the local populations forming a dynamically moving metapopulation. Individual agents are themselves unable to compare the different moves available to them, but their *interaction* leads to the exploitation of the most promising branches at each node of the game-tree. Similarly, local populations have a weak level of play when taken independently (branches are chosen without tactical sense), but their *interaction* makes a high level of play emerge as SDST is asymptotically equivalent to Minimax. Interestingly, the concept of metapopulation (a population of populations) has been used in biology since 1969 to refer to the dynamical coupling that appears between different populations of social insects [23].

Thirdly, the work presented herein takes on its full meaning only if one recognises that it potentially offers interesting insights relating to cognition. In fact, SDS has already been proposed as a model for neural activity: the one-to-one communication makes it a plausible candidate, and there exists a connectionist spiking neuron version of SDS called NESTER (for NEural STochastic nEtwoRk) [29]. Also in SDS, contrary to most of the other swarm intelligence heuristics[10], semantic interpretation (meaning) is embedded in the entire population of the swarm instead of being simply supported by individual agents[11]. In the neural model NESTER, this property leads to the synchronisation of the firing of neurons at convergence; "*hence in this model oscillatory behaviour may be a result of, rather than a cause of, the*

[10] Ant Colony Optimisation also shares this property

[11] This property is due to the partial evaluation of solutions: in the case of string matching for example, as discussed by Nasuto [27], the position of the solution after convergence is indicated by the formation of a cluster of agents, possibly dynamically fluctuating; in the case of a partial match, agents will keep exploring the text while the cluster will globally stay on the best match.

binding of features belonging to the same object" [29]. Furthermore, in addition to offering a novel theoretical solution to the binding problem [28], this ability to efficiently and dynamically allocate cognitive resources in a cognitive search task has been proposed as a model for neural attention [10].

Finally, in their survey of Monte Carlo Tress Search [7], Browne et al. concluded that:

> "Over the next five to ten years, MCTS is likely to become more widely used for all kinds of challenging AI problems. We expect it to be extensively hybridised with other search and optimisation algorithms and become a tool of choice for many researchers. In addition to providing more robust and scalable algorithms, this will provide further insights into the nature of search and optimisation in difficult domains, and into how intelligent behaviour can arise from simple statistical processes."

Although it was not conceived for practical AI purposes, we believe that SDST pertains to the type of hybridised algorithm Browne et al. had in mind. In particular, by integrating MCTS with the swarm intelligence paradigm of Stochastic Diffusion Search, we believe that SDST indeed manages to "provide further insights (...) into how intelligent behaviour can arise from simple statistical processes."

Acknowledgements The central argument presented herein was developed under the aegis of Templeton project 21853, *Cognition as Communication and Interaction*. The initial development of SDST was extracted from the unpublished MSC Dissertation from Tanay [33] and from Tanay et al [34]. This work was originally presented by Bishop at the PT-AI conference St. Antony's College, Oxford, 22nd-23rd September, 2013.

References

1. Abramson, B.: Expected-outcome: A general model of static evaluation. Pattern Analysis and Machine Intelligence, IEEE Transactions on **12**(2), 182–193 (1990)
2. Aleksander, I., Stonham, T.: Guide to pattern recognition using random-access memories. Computers and Digital Techniques, IEE Journal on **2**(1), 29–40 (1979)
3. Beattie, P., Bishop, J.: Self-localisation in the 'SENARIO' Autonomous Wheelchair. Journal of intelligent & robotic systems **22**(3), 255–267 (1998)
4. Bishop, J.: Stochastic Searching Networks. In: Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313), pp. 329–331. IET (1989)
5. Bishop, J.: The Stochastic Search Network. In: R. Linggard, D. Myers, C. Nightingale (eds.) Neural Networks for Images, Speech, and Natural Language, pp. 370–387. Chapman & Hall, Ltd. (1992)
6. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature **406**, 3942 (2000)
7. Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. Computational Intelligence and AI in Games, IEEE Transactions on **4**(1), 1–43 (2012)
8. Chaslot, G., Bakkes, S., Szita, I., Spronck, P.: Monte-carlo tree search: A new framework for game ai. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, pp. 216–217 (2008)
9. De Meyer, K.: Foundations of stochastic diffusion search. Ph.D. thesis, University of Reading, UK. (2003)

10. De Meyer, K., Bishop, J., Nasuto, S.: Attention through Self-Synchronisation in the Spiking Neuron Stochastic Diffusion Network. Consc. and Cogn **9**(2), 81–81 (2000)
11. De Meyer, K., Nasuto, S., Bishop, J.: Stochastic diffusion optimisation: the application of partial function evaluation and stochastic recruitment in swarm intelligence optimisation. In: A. Abraham, C. Grosam, V. Ramos (eds.) Swarm intelligence and data mining, vol. 2. Springer Verlag (2006)
12. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. thesis, Milano: Politecnico di Italy (1992)
13. Dorigo, M., Maniezzo, V., Colorni, A., Dorigo, M., Dorigo, M., Maniezzo, V., Maniezzo, V., Colorni, A., Colorni, A.: Positive feedback as a search strategy. Tech. rep., Technical Report No. 91-016, Politecnico di Milano, Italy (1991)
14. Gale, D.: The game of hex and the brouwer fixed-point theorem. The American Mathematical Monthly **86**(10), 818–827 (1979)
15. Goodman, L.J., Fisher, R.C.: The Behaviour and Physiology of Bees. CAB International, Oxon, UK (1991)
16. Grech-Cini, H., McKee, G.: Locating the mouth region in images of human faces. In: SPIE-The International Society for Optical Engineering, Sensor Fusion VI, vol. 2059 (1993)
17. Hart, S.: Games in extensive and strategic forms. Handbook of Game Theory with Economic Applications **1**, 19–40 (1992)
18. Hofstadter, D.: Godel, Escher, Bach: An Eternal Golden Braid. Basic Books (1979)
19. Holldobler, B., Wilson, E.O.: The Ants. Springer-Verlag (1990)
20. Kennedy, J.F., Eberhart, R.C., Shi, Y.: Swarm intelligence. Morgan Kaufmann Publishers, San Francisco ; London (2001)
21. Kennedy J. & Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (IV), pp. 1942–1948 (1995)
22. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo Planning. Machine Learning: ECML 2006 pp. 282–293 (2006)
23. Levins, R.: Some demographic and genetic consequences of environmental heterogeneity for biological control. Bulletin of the ESA **15**(3), 237–240 (1969)
24. McDermott, D.: xartificial intelligence and consciousness. In: M.M. Zelazo P.D, E. Thompson (eds.) The Cambridge Handbook of Consciousness. Cambridge University Press (2007)
25. Metropolis, N., Ulam, S.: The monte carlo method. Journal of the American Statistical Association **44**(247), 335–341 (1949). DOI 10.1080/01621459.1949.10483310. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1949.10483310. PMID: 18139350
26. Moglich, M., Maschwitz, U., Holldobler, B.: Tandem calling: A new kind of signal in ant communication. Science **186**(4168), 1046–1047 (1974)
27. Nasuto, S.: Resource Allocation Analysis of the Stochastic Diffusion Search. Ph.D. thesis, University of Reading (1999)
28. Nasuto, S., Bishop, J.: Neural Stochastic Diffusion Search Network-a theoretical solution to the binding problem. In: Proc. ASSC2, Bremen, vol. 19 (1998)
29. Nasuto, S., Bishop, J., De Meyer, K.: Communicating neurons: A connectionist spiking neuron implementation of stochastic diffusion search. Neurocomputing **72**(4), 704–712 (2009)
30. Nasuto, S., Bishop, J., Lauria, S.: Time complexity analysis of the stochastic diffusion search. Neural Computation **98** (1998)
31. Nasuto, S., Bishop, M.: Convergence analysis of stochastic diffusion search. Parallel Algorithms and Applications **14**(2), 89–107 (1999)
32. Seeley, T.D.: The Wisdom of the Hive. Harvard University Press (1995)
33. Tanay, T.: Game-tree exploration using stochastic diffusion search. Tech. rep., goldsmiths, University of London (2012)
34. Tanay, T., Bishop, J., Nasuto, S., Roesch E.B. &Spencer, M.: Stochastic diffusion search applied to trees: a swarm intelligence heuristic performing monte-carlo tree search. In: Proceedings of the AISB 2013: Computing and Philosophy symposium,'What is computation?' (2013)
35. Whitaker, R., Hurley, S.: An agent based approach to site selection for wireless networks. In: Proceedings of the 2002 ACM symposium on Applied computing, pp. 574–577. ACM (2002)