Stochastic Diffusion Search applied to Trees: a Swarm Intelligence heuristic performing Monte-Carlo Tree Search

Thomas Tanay and J. Mark Bishop[1]
Matthew C. Spencer and Etienne B. Roesch and Slawomir J. Nasuto[2]

Abstract. In this paper, we introduce Stochastic Diffusion Search applied to Trees (SDST), a swarm intelligence heuristic inspired from Stochastic Diffusion Search able to solve the complex and general problem of forward planning. 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.
This result is presented in the context of Abstract Platforms of Computation (APCs), a concept introduced in an accompanying paper in an attempt to clarify and broaden the notion of computation. In particular, the concept of APC is used to draw a distinction between classical sequential algorithmic models of computation and nature-inspired parallel distributed ones. It is argued that the understanding of (at least human) cognition requires the study of decentralised emergent systems (fundamentally parallel and distributed), whose computational properties cannot be reduced to their Turing power.

1 INTRODUCTION

In [12], we introduced the concept of Abstract Platforms of Computation (APCs) in order to characterise in a unified framework various computational paradigms and we discussed in this context the computational nature of both abstract and natural dynamical systems. The concept is defined in terms of intrinsic dynamics and customisation of constraints on these dynamics in a way that is consistent with other recent works. For example, [18] defines computation as "the processing of medium-independent vehicles according to rules", the notion of medium-independent vehicle being further detailed in the following way: "a given computation can be implemented in multiple physical media (e.g. mechanical, electromechanical, electronic, magnetic, etc), provided that the media possesses a sufficient number of dimensions of variation (or degrees of freedom) that can be appropriately accessed and manipulated."

Although our framework proposed in [12] based on the concept of APCs agrees in the broad sense with the taxonomy of computation proposed in [18], we do not believe that cognition is merely a computational phenomenon (as it is claimed to be in [18]: we may conclude that cognition is computation in the generic sense) even though the APC framework admits perfectly viable and a posteriori structurally isomorphic computational descriptions of cognitvce processes. Thus ultimately, one of the most important problems in cognitive science is to determine and characterise the set of dynamical systems that have the right computational properties to provide such descriptions. In the case of biological systems, cognition results from the activity of the brain which is inherently a parallel and distributed system. Surprisingly however, current parallel and distributed computational models (such as neural networks or swarm intelligence heuristics) appear rather limited when facing some complex problems. For example, artificial neural networks are particularly adapted to solve pattern recognition problems but they have difficulty in sequentially processing high arity predicates (they can be conceived as fundamentally building learning mappings in complex high dimensional Euclidean spaces). Consequently, the dominant paradigm in artificial intelligence has historically been and is still the sequential algorithmic one.

In the present paper, we introduce Stochastic Diffusion Search applied to Trees (SDST), a swarm intelligence heuristic (inherently parallel and distributed) inspired from Stochastic Diffusion Search (SDS) and able to solve the complex and general problem of forward planning in a way analogous to Monte-Carlo Tree Search (MCTS). 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 finite two-person zero-sum game with imperfect 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. Here, it is important to contrast this result with what constitutes a universality proof[3]. Indeed, proving that a model is Turing-equivalent consists in showing that any Turing machine can be simulated by an instance of this model. But in such a case, there

[1] Goldsmiths, University of London, UK, email: thomas.tanay@gmail.com
[2] University of Reading, UK

[3] Note that there does not currently exist any such proof for SDS, but for example neural nets [19] and the rule 110 cellular automaton [6] (which are parallel and distributed models) are proven to be Turing-equivalent.

exists an abstract level at which the instance in question can be described as implementing a sequential algorithmic APC. On the contrary in SDST, the solution to the problem faced is fundamentally emerging from the decentralised interaction of simple computational agents, and the functioning of the heuristic fundamentally cannot be abstracted to the functioning of a sequential algorithm APC (See Table 1 for an illustration of the different levels of abstraction). In practice, as described in section 3, SDST has been "programmed" from the intrinsic dynamics defining SDS (the alternation of test and diffusion phases in a population of communicating agents) by customising the test and diffusion protocols.

Table 1: Nature of the APCs in two example implementations of MCTS: SDST and a hypothetical implementation of classical MCTS via a cellular automaton (CA).

APC where MCTS occurs	SDST: Parallel Distributed	Classical MCTS: Sequential Algorithmic
Underlying APC	Digital computer: Sequential Algorithmic	Cellular automaton: Parallel Distributed

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.

2 BACKGROUND

The work presented here rests on two pillars: the swarm intelligence metaheuristic for search and optimisation called Stochastic Diffusion Search (SDS) and the Monte-Carlo based search method for tree structures called Monte-Carlo Tree Search (MCTS). These two techniques are briefly described in the following subsections.

2.1 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 [20], mobile robot self-localisation [2], object recognition [11] and text search [3]. Additionally, a hybrid SDS and n-tuple RAM [1] technique has been used to track facial features in video sequences [11, 8]. Previous analysis of SDS has investigated its global convergence [13], linear time complexity [17] and resource allocation [14] 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 [13].

2.2 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" [4]. Since 2006, over 200 papers related to MCTS have been published, with applications ranging from computer Go to Constraints Satisfaction problems through Reinforcement Learning and Combinatorial Optimisation. [4] offers a complete survey of the published work on MCTS (until 2011) and argues that "it 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*. In this paper is introduced the idea to evaluate a game position by playing a great number of random games from that position, assuming that a good move must increase the expected outcome of the player[4]. 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 is introduced Upper Confidence bound applied to Trees (UCT), a method that "applies bandit ideas to guide Monte-Carlo planning". The crux in 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 exploit more 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 [5] 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 [9] 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 [4] 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

[4] This assumption is not necessarily a good one due to the distinction between random play and optimal play.

convergence to Minimax, i.e. it is assured to select the best move available if enough time is given (the convergence to Minimax can be very long in practice).

3 STOCHASTIC DIFFUSION SEARCH APPLIED TO TREES (SDST)

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. Conceptually, the application of SDS to game-tree exploration is a two step process. First, each node is being 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-tree—thus leading to the formation of a dynamically moving *metapopulation*[5] of agents.

3.1 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 2 (in SDS terms).

The operation of this algorithm is illustrated in Figure 2 on the small game-tree presented in Figure 1. 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).

Figure 2 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 2 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).

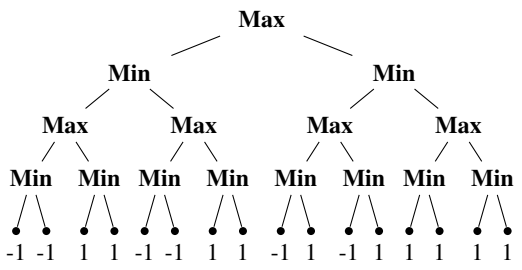
[5] The term was coined by Levins in [10] to describe the dynamics of interacting populations of social insects.

Table 2: First application of SDS to game-tree exploration: use of multiple populations of agents.

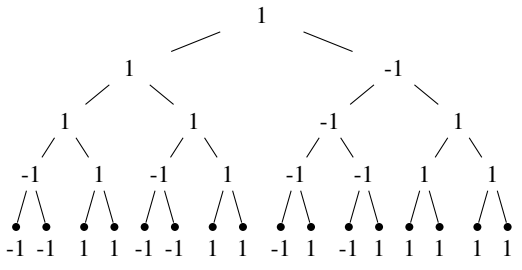
Initialisation During the initialisation phase, a local population of agents is generated for each node of the game-tree up to a fixed depth D. For each local population, agents' hypotheses are initialised to a possible move of the corresponding node.

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.



(a) Studied game-tree



(b) Minimax resolution

Figure 1: Studied game-Tree. The minimax resolution shows that Max is the winner if he plays optimally.

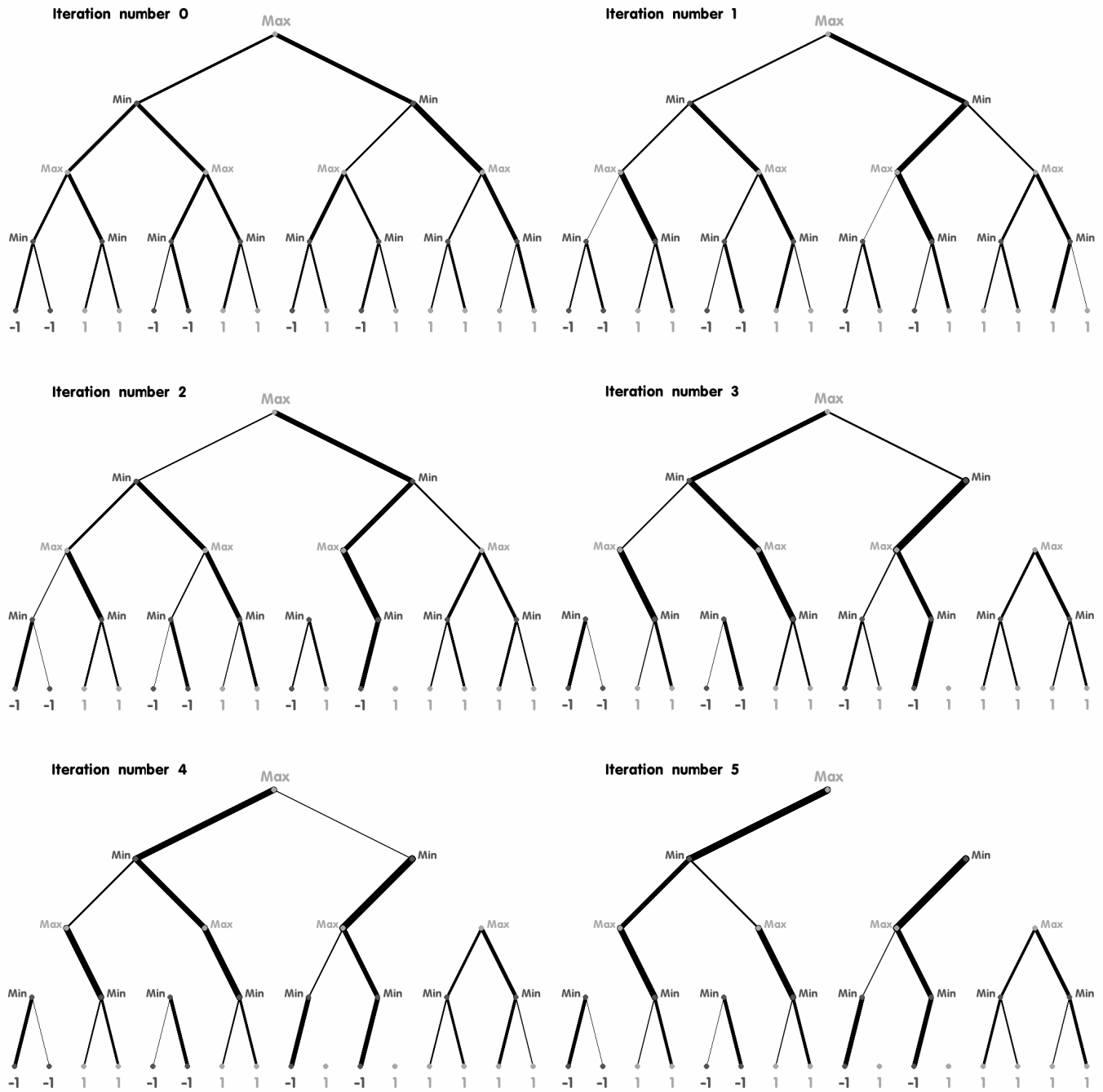


Figure 2: Illustration of the algorithm presented in Table 2: Evolution of the distribution of the agents in the different nodes of the studied game-tree (first 5 iterations shown, total number of agents = 175). Each branch has an area proportional to the number of agents in the parent node population supporting the move corresponding to the child node population.

3.2 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 2).
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 3.

Table 3: 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 3 on the studied game-tree. As for the previously discussed algorithm, a majority of agents in the root node population first points toward the right move (best move in a purely Monte-Carlo sense) before reorienting toward the left move (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 3: at iteration 12 the metapopulation stabilises in the left part of the game-tree.

4 DISCUSSION

In the previous sections, we have introduced Stochastic Diffusion Search applied to Trees (SDST), a swarm intelligence heuristic performing forward planning. SDST is very similar to classical Monte-Carlo Tree Search (MCTS) algorithms in its functioning, but is conceptually 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 homogeneous agents with limited computational capacities. This distinction can be expressed differently by saying that classical MCTS and SDST are implemented on Abstract Platforms of Computation of fundamentally different nature (sequential algorithmic vs. parallel distributed). Importantly, the facts that classical MCTS could be implemented on a Turing equivalent decentralised system (such as a cellular automaton), or that SDST is being executed on a digital computer are not relevant. The concept of APC allows layered levels of abstraction, and what matters is the nature of the APC at the level at which the forward planning problem is being solved. This last remark suggests that the broad notion of computability that emerged in recent works ([12], [18]) needs new tools to be studied and in particular, a characterisation of computational systems in terms of their Turing power is not sufficient any more.

In addition to the main result, our work introduces a meta-level in the Swarm Intelligence paradigm: 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 (SDST is asymptotically equivalent to Minimax). Interestingly, the concept of metapopulation (a population of populations) exists in biology to refer to the dynamical coupling that appears between different populations of social insects [10].

Finally, the work presented here takes on its full meaning only if one recognises that it might have some interesting insights to provide about 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) [16]. Also in SDS, contrary to most of the other swarm intelligence heuristics[6], the meaning is embedded in the entire population instead of being simply supported by individual agents. This property is due to the partial evaluation of solutions: in the case of string matching for example, 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).

[6] Ant Colony Optimisation also shares this property

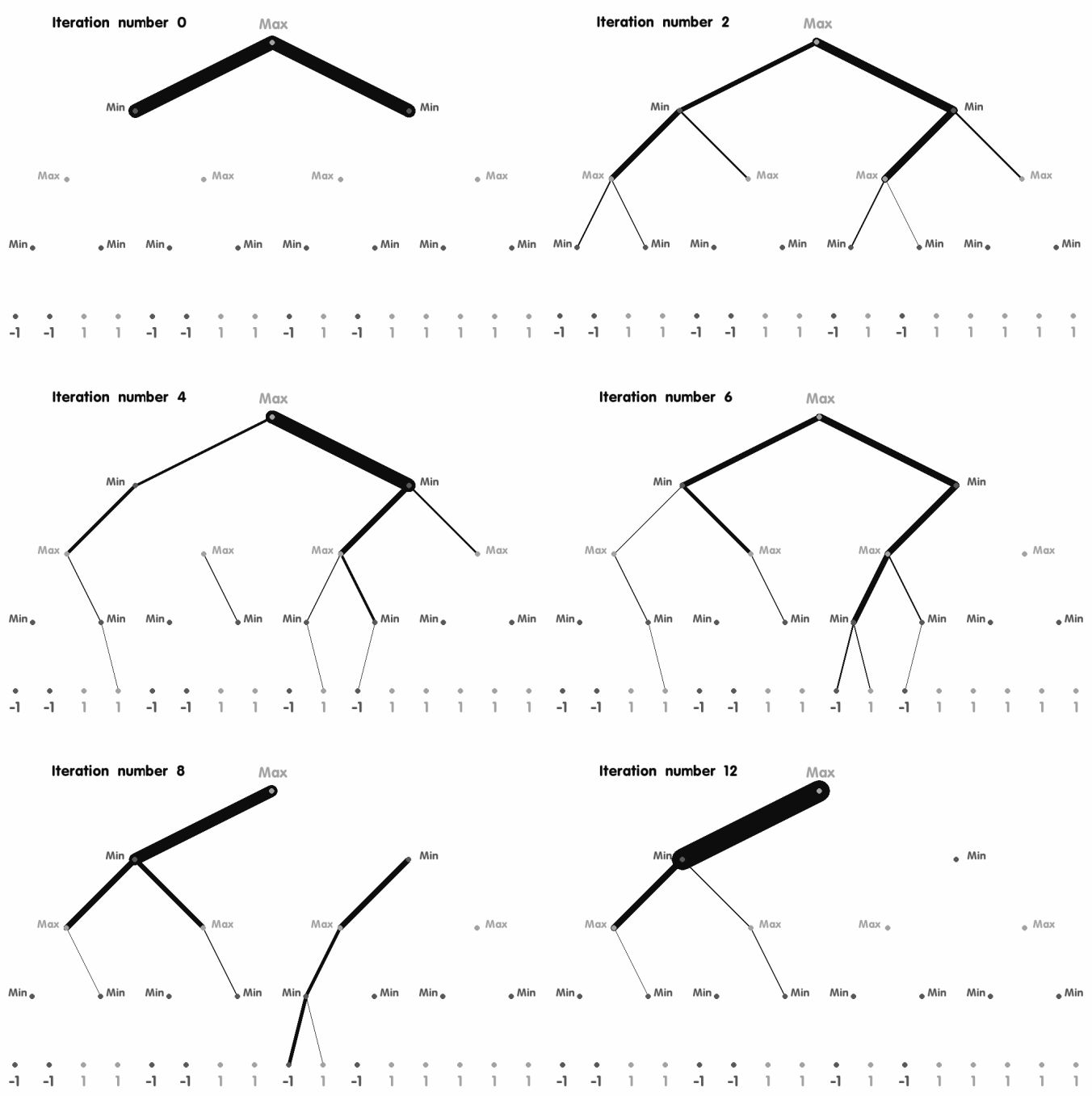


Figure 3: Illustration of SDST: Evolution of the distribution of the agents in the entire game-tree (iterations 0, 2, 4, 6, 8 and 12 shown, total number of agents = 100). Each branch has an area proportional to the number of agents in the parent node population supporting the move corresponding to the child node population.

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 binding of features belonging to the same object*" [16]. In addition to giving a new theoretical solution to the binding problem [15], the ability to allocate efficiently and dynamically the cognitive resources to the search task has been proposed as a model for neural attention [7].

In their survey [4], 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 into the swarm intelligence paradigm, we believe that SDST indeed manage to "provide further insights (...) into how intelligent behaviour can arise from simple statistical processes."

ACKNOWLEDGEMENTS

We would like to thank the John Templeton Foundation for supporting this research, and the reviewers for their useful comments.

REFERENCES

[1] I. Aleksander and T.J. Stonham, 'Guide to pattern recognition using random-access memories', *Computers and Digital Techniques, IEE Journal on*, **2**(1), 29–40, (1979).
[2] P.D. Beattie and J.M. Bishop, 'Self-localisation in the 'SENARIO' Autonomous Wheelchair', *Journal of intelligent & robotic systems*, **22**(3), 255–267, (1998).
[3] J.M. Bishop, 'Stochastic Searching Networks', in *Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313)*, pp. 329–331. IET, (1989).
[4] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, 'A survey of monte carlo tree search methods', *Computational Intelligence and AI in Games, IEEE Transactions on*, **4**(1), 1–43, (2012).
[5] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, '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).
[6] Matthew Cook, 'Universality in elementary cellular automata', *Complex Systems*, **15**(1), 1–40, (2004).
[7] K. De Meyer, J.M. Bishop, and S.J. Nasuto, 'Attention through Self-Synchronisation in the Spiking Neuron Stochastic Diffusion Network', *Consc. and Cogn*, **9**(2), 81–81, (2000).
[8] H.J. Grech-Cini and G.T. McKee, 'Locating the mouth region in images of human faces', volume 2059, (1993).
[9] L. Kocsis and C. Szepesvári, 'Bandit based Monte-Carlo Planning', *Machine Learning: ECML 2006*, 282–293, (2006).
[10] R. Levins, 'Some demographic and genetic consequences of environmental heterogeneity for biological control', *Bulletin of the ESA*, **15**(3), 237–240, (1969).
[11] R. Linggard, DJ Myers, and C. Nightingale, 'The Stochastic Search Network', in *Neural Networks for Images, Speech, and Natural Language*, 370–387, Chapman & Hall, Ltd., (1992).
[12] C. S. Matthew, J. M. Bishop, J. S. Nasuto, E. B. Roesch, and T. Tanay, 'Abstract platforms of computation', *AISB*, (2013). submitted.
[13] S. Nasuto and M. Bishop, 'Convergence analysis of stochastic diffusion search', *Parallel Algorithms and Applications*, **14**(2), 89–107, (1999).
[14] S.J. Nasuto, *Resource Allocation Analysis of the Stochastic Diffusion Search*, Ph.D. dissertation, University of Reading, 1999.
[15] S.J. Nasuto and J.M. Bishop, 'Neural Stochastic Diffusion Search Network-a theoretical solution to the binding problem', in *Proc. ASSC2, Bremen*, volume 19, (1998).
[16] S.J. Nasuto, J.M. Bishop, and K. De Meyer, 'Communicating neurons: A connectionist spiking neuron implementation of stochastic diffusion search', *Neurocomputing*, **72**(4), 704–712, (2009).
[17] S.J. Nasuto, J.M. Bishop, and S. Lauria, 'Time complexity analysis of the stochastic diffusion search', *Neural Computation*, **98**, (1998).
[18] Gualtiero Piccinini and Andrea Scarantino, 'Information processing, computation, and cognition', *Journal of biological physics*, **37**(1), 1–38, (2011).
[19] Hava T Siegelmann and Eduardo D Sontag, 'On the computational power of neural nets', *Journal of computer and system sciences*, **50**(1), 132–150, (1995).
[20] R.M. Whitaker and S. Hurley, '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).