When Ants Play Chess
(Or Can Strategies Emerge From Tactical Behaviors?)

Alexis Drogoul
LAFORIA, Boîte 169, Université Paris VI
4, Place Jussieu 75252 PARIS CEDEX 05
drogoul@laforia.bip.fr

Abstract

Because we think that plans or strategies are useful for coordinating multiple agents, and because we hypothesize that most of the plans we use are build partly by us and partly by our immediate environment (which includes other agents), this paper is devoted to the conditions in which strategies can be viewed as the result of interactions between simple agents, each of them having only local information about the state of the world. Our approach is based on the study of some examples of reactive agents applications. Their features arebriefly described and we underline, in each of them, what we call the *emergent strategies* obtained from the local interactions between the agents. Three examples are studied this way: the eco-problem-solving implementations of Pengi and the N-Puzzle, and the sociogenesis process occuring in the artificial ant colonies that compose the MANTA project. We then consider a typical strategical game (chess), and see how to decompose it through a distributed reactive approach called MARCH. Some characteristics of the game are analysed and we conclude on the necessity to handle both a global strategy and local tactics in order to obtain a decently strong chess program.

1. Introduction

In military science as well as in everyday life, building a *strategy* usually means making a plan of coordinated actions for reaching a particular goal. It often refers to situations in which (1) the goal to reach is possibly conflicting with the goal of another agent and (2) different resources must be employed, sometimes concurrently, in order to reach it. A strategy always relies on two strong assumptions: having a view as global as possible of the current situation, and making sure that the resources will perform the desired actions. That is why military have always payed a great attention to develop intelligence services, in order to obtain as much information as they can, and to establish strong hierarchies, in order to be obeyed on the battlefield. But that is also why famous battles have been lost. The best example could be that of the battle of Waterloo, during which the late arrival of Grouchy, due to both a lack of information and a misunderstanding of orders, has ruined the whole strategy established by Napoleon and its generals. In that respect, the disadvantages of a long term strategy are quite clear (note that these critics can be applied to *plans*, in the AI acceptation of the term, see (Brooks 1987)): (1) relying on too global information prevents it from being truly adaptive to unpredictible details and it is always difficult

to backtrack when the resources are already engaged in their respective actions; (2) assuming that the resources are dependable prevents it from being robust to their failures. We can then assume that, without taking external conditions (like weather) into account, the main reasons for which a strategy fails are constituted by local failures at the resources level, be they facing an unexpected event or badly performing their part of the strategy. By analogy with the military terms, we will call these local behaviors *tactical* behaviors.

However, battles are not always lost. And, when they are lost by one of the participants, they are likely to be won by its opponents. In most cases, like in Waterloo from the Allieds' point of view, the success results both from the pre-established strategy and from *tactical* behaviors that were not included in it, but whose occurence has favourized it. In these cases, the victory can be viewed as the consequence of a set of tactical behaviors, some of them having been generated by a strategy, and some of them not.

But what would an independent observer not aware of both strategies notice ? He would see a sequence of actions, coordinated or not, and would reconstruct his own vision of the strategy employed by the winner. In the worst case (see Figure 1), this vision of the battle could be totally irrelevant to the real strategy.

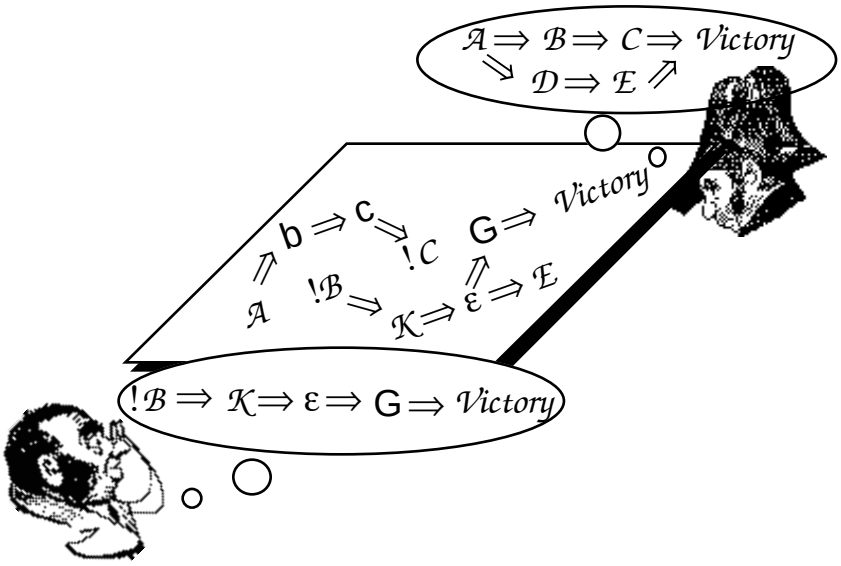

Fig 1 - The strategist (time t-1), the battlefield (time t) and the observer (time t+ 1)

In this somewhat extreme case, the strategist is unuseful and the strategy (or the plan), as understood by the observer, emrges from the coordination of local tactical behaviors performed by resources that are not aware of their place in this strategy (for the good reason that it is not written anywhere).
Because we do think that plans or strategies are useful for coordinating multiple agents, and because we hypothesize that most of the plans we use everyday are build

partly by us, and partly by our immediate environment (which includes other agents), this paper will be devoted to the conditions in which strategies can be viewed as the result of interactions between simple agents, each of them having only local information about the state of the world. We do not claim that neither all strategies nor all plans can be viewed this way, but we rather think that such an approach can constitute a very constructive lower bound for planning or search, by helping these two fields to take local behaviors into account and by eventually providing them with new tools (for finding local heuristics, for example).

Our approach will be based on the study of some examples of reactive agents applications (see (Ferber & Drogoul 1992) for an exhaustive definition). Their features will be briefly described and we will underline, in each of them, what we call the *emergent strategies* obtained from the local interactions between the agents. Three examples will be studied this way: the eco-problem-solving implementations of Pengi and the N-Puzzle, and the sociogenesis process occuring in the artificial ant colonies that compose the MANTA project.
We will then consider a typical strategical game (chess), and see how to decompose it through a distributed reactive approach called MARCH. Some characteristics of the game will be analysed and we will conclude on the necessity to handle both a global strategy and local tactics in order to obtain a decently strong chess program.

2. Previous works on emergent strategies

2.1 Pengi

Description of the Game

Pengi is based on a video game called Pengo. A penguin (the player) is moving around in a maze made up of ice cubes and diamonds, and inhabited by bees. The penguin must collect all the diamonds while avoiding being stung by a bee or crushed by a cube. It can kill bees by kicking cubes at them. Bees can alos kill the penguin by kicking cubes at it or stinging it. They move around in a semi-random way: a bee may sometimes choose to get away from the penguin instead of chasing it. An ice cube can be pushed unless there is another cube adjacent to it in the direction it is pushed. Then it slips along until it hits a cube, a diamond or the border of the screen. Any animal met during its move is crushed.
Therefore, the strategy usually followed by the player is clearly twofold: try to collect as much diamonds as possible while avoiding to be crushed by an ice cube or stung by a bee.

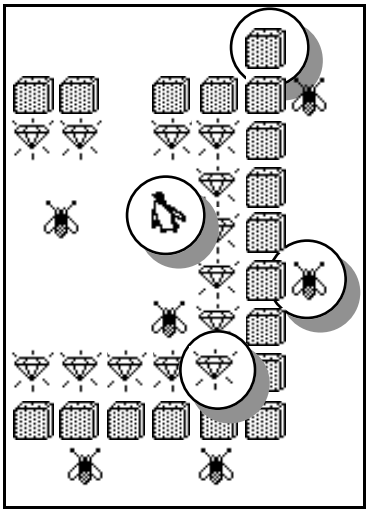

Fig 2 - The Pengi game

The Eco-Problem-Solving Implementation of Pengi

Our implementation of Pengi (different from that of (Agre & Chapman 1987)) has been realized within the Eco-Problem-Solving framework (see (Ferber & Jacopin 1989) or (Ferber & Drogoul 1992) for a definition of it, and (Drogoul & al. 1991) for a more precise description of EcoPengi), in which the agents are provided with a local perception of their environment, a satisfaction behavior for reaching their own individual goal and a flight behavior used in case of aggression. All the agents are viewed as totally autonomous, which means that they are not directed by a pre-established strategy. In that respect, their behaviors are:

- Diamonds are satisfied and cannot flee (i.e. they do not have any behaviors).
- Ice cubes do not perform any satisfaction behavior. When being attacked (by a bee or a penguin), they flee in the opposite direction of the aggression and when this location is free, send themselves a flee-message from their old location. During its flight, a cube can enter the perception area of the penguin, and then attack it, or crush an animal when entering its location.
- Bees own a parametrable perception area (the distance at which they see the penguin). Their satisfaction behavior is twofold: if the penguin moves around in this area, they take its position as goal and begin to use an incremental satisfaction behavior (Delaye & al. 1990), which consists in moving each time in the direction of the goal. If not, they randomly choose a goal in this area. They possess two aggressive behaviors: the first one against the ice cubes, the second against the penguin when they see it. They just tell them to flee.
- The satisfaction behavior of the penguin consists in reaching and eating the nearest diamond. While not satisfied, it uses the same incremental satisfaction behavior as the bees. Its flight behavior is generated by an external aggression coming either from a bee or an ice cube. This behavior is twofold: (1) fleeing a cube just prompts the penguin to find an adjacent location perpendicular to the cube's moving direction (if there is none, it flees in the same direction until it finds a safe location), and (2) fleeing a bee consists in getting away from it as far as possible (or, if the bee stands on the other side of an ice cube adjacent to the penguin, in kicking it at the bee). Pushing a cube is the unique aggressive behavior of the penguin.

Behavior of the System

As the behavior of the penguin is goal-directed (moving towards the nearest diamond), it has to be aware of the position of the diamonds that it did not eat before. These information are provided to it by means of gradients, generated by the diamonds and propagated by the patches. Although it is computed in a distributed manner (see Figure 3), it could appear as an endeavour to give the penguin a global perception of the game. But what is important to understand is that these gradients do not replace any strategy, because: (1) they provide alternate paths to each diamond and do not force the penguin to follow one of them; (2) they do not provide the penguin with any indication on the coordination of actions that it will have to perform to reach a diamond (for example, when meeting bees on the path); (3) they only provide the penguin with the position of the nearest diamond, without paying attention to the other diamonds' positions.

Thanks to this propagation, the penguin just needs, at each time, to ask its patch gradient value, and its incremental satisfaction behavior will consist in choosing the adjacent location whose gradient value is the smallest.

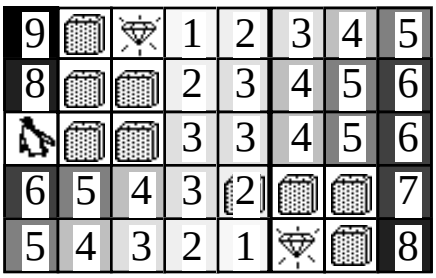

Fig 3 - Two diamonds gradient

When looking at Pengi running, an observer will then face a system that plays (and wins, in most of the cases) the game without defining any *a priori* strategy - which he does not necessarily know. He will see the penguin moving around the board, catching diamonds, avoiding (and sometimes killing) bees and eventually stopping. He will then hypothesize that an intelligent system, hidden somewhere behind the game, has successfully planned the penguin's activity. And he will have this approach because we know that it is difficult to reason about situations like this without a centralized mindset (Resnick 1992). He will then *naturally* build out an *a posteriori* strategy from the sequence of actions of the penguin and attribute it to the omnipotent entity.

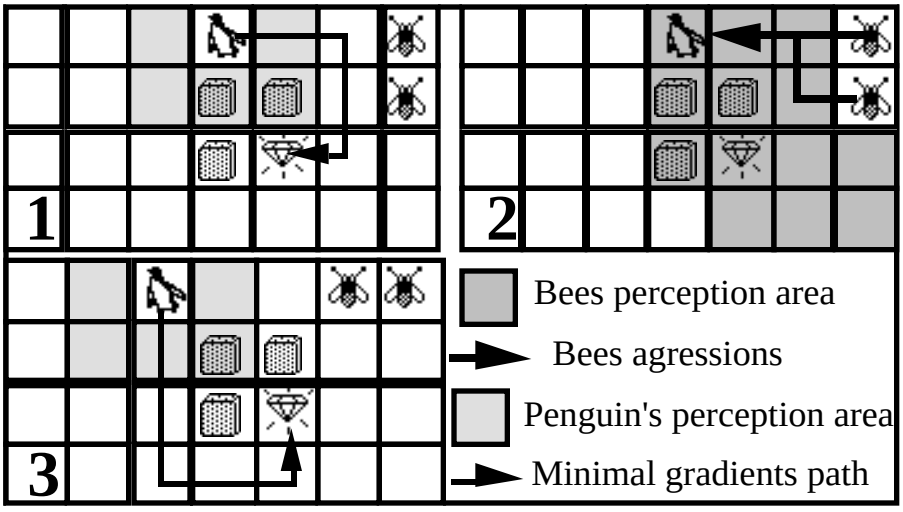

Fig 4 - a short-term emergent "plan"

But the long-term strategy as well as the short-term plans (like that on Figure 4) used by the penguin are just artificial reconstructions of the observer's mind, who credits the system with an intelligence that it does not own. For instance, in the situation

above, one could think that the penguin has planned a path to the diamond, path that would prevent it from being caught by the bees. But the way the final path has been chosen is only due to the reactivity of the penguin to the bees, and to the alternate paths provided by the gradient propagation.
Pengi allows us to underline three features of the *emergent strategies*: (1) They rarely lead to optimal solutions (the number of moves of the penguin is often greater than the number of moves of a human player); (2) They can be observed but difficultly formalized automatically as a set of symbolic representations (see the discussion in (Wavish 1991) about representing emergent behaviours); (3) They are then difficult to use again (which is quite normal, in a way, in a very changing environement).

2.2 The N-Puzzle

Another example of *emergent strategy* can be found in our work on a distributed approach to the N-Puzzle (for a complete description of the solving system see (Drogoul & Dubreuil 1991)). The well known N-puzzle problem consists of a square board containing N square tiles and an empty position called the "blank". Authorized operations slide any tile adjacent to the blank into the blank position. The task is to rearrange the tiles from some random configuration into a particular goal configuration.
Our implementation of the N-Puzzle considers each tile as an autonomous agent, with its own goal and its own perception area. As the world in which these agents move is much more constrained than that of Pengi, the tiles are provided with more sophisticated local heuristics than the penguin. They primarily help them to find a path to their goal without disturbing too much the previously satisfied tiles and without generating loops of aggression. Like Pengi, these behavioral heuristics are based on the computation of gradient fields, both to the blank location of the puzzle and to the goal of the current tile (see (Drogoul & Dubreuil 1993) for details), with the possibility for a tile to temporarily lock its patch in order to protect its position.
Although the first aim of this project was to prove, from the point of view of the computation cost, the interest of a distributed approach in a domain essentially reserved to classical search, we were surprised to see the emergence of truly original strategies in the solving of small problems within the puzzle, like that of the placement of the corner tiles. The task is to satisfy a tile whose goal is a corner of the puzzle without shifting the satisfied tiles that make up the row or the column including this corner. Figure 5 focuses on the problem met by tile $\mathcal{A}$ and shows the moves of the fleeing tiles, in order to see how it works.

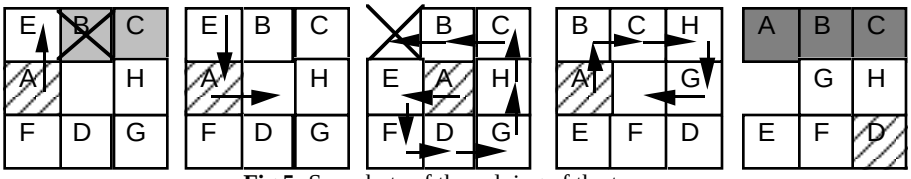

Fig 5- Snapshots of the solving of the top row

If we comment these stages from the tiles viewpoint, we have, in the beginning, $\mathcal{B}$ and $\mathcal{C}$ that are satisfied, and $\mathcal{A}$ that tries to reach its goal. $\mathcal{A}$ then attacks $\mathcal{E}$. The patch of $\mathcal{B}$ is provided as a constraint during this attack. $\mathcal{E}$ finds no unlocked patch for fleeing. It

then unlocks all the patches and attacks $\mathcal{A}$ (the patch of $\mathcal{B}$ is still a constraint). $\mathcal{A}$ flees on the blank. Since it remains unsatisfied, it attacks $\mathcal{E}$ again, now with its goal as constraint. $\mathcal{E}$ then attacks $\mathcal{F}$, which attacks $\mathcal{D}$, and so on until $\mathcal{B}$ flees on the blank. $\mathcal{A}$ then slides on its previous patch and attacks $\mathcal{B}$. In this case, there are no valid constraints. $\mathcal{B}$ then attacks $\mathcal{C}$, which prefers to slide onto its goal rather than onto the blank. Therefore, it attacks $\mathcal{H}$, which attacks $\mathcal{G}$, which flees onto the blank. $\mathcal{A}$ now satisfies itself and the row is solved.

The previous comment describes how the system works, from the point of view of the programmer. It seems quite difficult to understand it as a clear strategy for satisfying the corner tiles. However, the constant repetition of this behavior during all the experiments gave us the idea to formalize it symbolically, like a naive observer would do, and to try it by hand. And it appeared to be a very powerful and universal "operator" for completing rows and columns, and eventually solving the whole puzzle, even for a human player. The idea is to:

1) Correctly place all the tiles of the row/column except the last one (called $\mathcal{T}n$), which is quite easy to do.

2) Slide $\mathcal{T}n$ towards its goal without shifting the previous ones.

3) When arrived in front of its goal, slide together $\mathcal{T}n$ and the tile lying on its goal (called $\mathcal{T}i$) backwards (like $\mathcal{A}$ and $\mathcal{E}$ on figure 5).

4) Then slide $\mathcal{T}i$ on any patch except the goal and current patch of $\mathcal{T}n$. This will necessarily shift the previously satisfied tiles.

5) Eventually slide $\mathcal{T}n$ towards its goal while replacing the previous tiles.

Although this algorithm is evidently sub-optimal, it is much simpler than any of the known methods proposed for completing the N-puzzles. It appears to be a good example of the alternate strategies that can be obtained by using a distributed approach to problem solving.

2.3 The Sociogenesis Process in MANTA's colonies

The MANTA project, described in (Drogoul & al. 1991b; Drogoul & Ferber 1992), is an application of the EthoModelling Framework (EMF) to the modelling and simulation of the social organization in an ant colony. EMF is based on the principles of multi-agent simulation, which means that each organism of the population, ants, cocoons, larvae and eggs, is represented as an agent whose behavior is programmed with all the required details. Our aim with MANTA is to test hypotheses about the emergence of social structures from the behaviors and interactions of each individual.
The foundation (or sociogenesis) process can be observed in many species of ants. In the species Ectatomma ruidum, whose societies are the natural counterparts of MANTA's simulated ones, the foundation is said semi-claustral : the foundress sometimes leaves her nest to provide the food which is necessary for itself and, above all, for the larvae (in a claustral foundation the queen provides food to the larvae by using her own internal reserves) . Furthermore, the queen generally continues to forage after the emergence of the very first workers.
Sociogenesis is a remarkable challenge at the queen level. As a matter of fact, the queen, which can be considered as an agent intended to behave in a multi-agent system in coordination with other agents (the future workers), has to face alone, during the first stages of the society, a very complex situation, in which it has to take

care of the whole brood as well as going and find food. So we were interested in the strategy of survival used by the queens, which appear to be quite the same in all natural colonies.

The experiments have been conducted with a model that reproduces the exact laboratory conditions in which natural sociogeneses have been studied. A variable amount of food that depends on the number of ants present in the colony is provided every day (of the simulation timescale) at the entrance of the nest. All the experiments have been tested with the same parameters, and the only source of unpredictability is to be found in the default random walk of the ants.

An experiment begins by putting a queen agent alone in an empty nest and let it evolve (lay eggs, which become larvae, cocoons, and eventually workers). We stop it whenever one of these two situations is reached: (1) the queen dies (by starving); (2) more than six workers are born. The first signifies the death of the colony and the failure of the sociogenesis. The second situation has been chosen as a good estimation of success, with respect to what happens in natural colonies. As a matter of fact, a natural colony that reaches this stage ordinarily keeps on growing, which means that the task of reaching around six workers constitutes the most difficult part of sociogenesis.

Now that we have defined the stop criterions of the simulation, we can take a look at the results. In Table 1 are tabulated the results in terms of successes and failures. Failure cases are clustered in seven categories, each of them indicating the composition of the population when the queen dies.

Results	Composition	Number	%
Failures with	Eggs	4	6,15%
	Eggs, Larvae	10	15,38%
	Larvae	19	29,23%
	Eggs, Larvae, Cocoons	3	4,62%
	Larvae, Cocoons	7	10,77%
	Eggs, Cocoons	1	1,54%
	Larvae, Workers	2	3,08%
Total Number of Failures		46	70,77%
Total Number of Successes		19	29,23%
Total Number of Experiments		65	100,00%

Table 1 - Successes and Failures of the sociogeneses

First point, the proportion of failures (70,77%) appears in these experiments to be close to that observed for natural sociogeneses in laboratory conditions. Second point, the situations in which the foundation of the colony is likely to fail can be obviously characterized by the fact that larvae are part of the population. As a matter of fact, cases of failure in the presence of larvae represent 89,13% of the total number of failures. Why is it so? The simplest explanation that can be provided is, from a behavioral point of view, that larvae must be cared, carried and nourished whereas the other agents composing the brood just need to be cared and carried. The presence of larvae then generates, at the population level, a very important need in food, propagated by the larvae by means of stimuli. Therefore, the agents that can bring

back food to the colony (namely the queen and the workers) will have to do it more often, simply because their feeding behavior will be much more frequently triggered than before. It does not cause too much problems as long as many workers are available, because other tasks still have a good probability to be executed. But, in the early stages of the foundation in which the queen is alone, it quickly results in preventing it from doing anything else, even keeping enough food to feed itself. Moreover, the feeding behavior is probably the most dangerous behavior of all, because food is often far away from the brood, which is aggregated near humid places and can then be neglected during a long time, and, more simply, because food can begin to run short, thus obliging the queen to stay outside the nest (or near its entry) until more food is supplied.

However, we have also conducted successful experiments. And, given the constraints stated above, we are going to see which *emergent strategy* is employed by the queen and the first workers to succeed in the foundation of the colony.

The colony whose demographical evolution is depicted on Figure 6 provides us with a typical example of the evolution of the population in our artificial colonies and can advantageously be compared to the curves obtained in the natural sociogeneses. The curve in light gray, which represents the population of eggs, is very unequal, although it is possible to detect a regularity in its sudden falls, which approximatively occur every fourty days. These falls appear to be synchronized with the peaks of the second curve (in gray), which represents the population of larvae. It is apparent, when loooking at other populations, that there is an interplay between the populations of eggs and larvae at the colony level. What is interesting, however, is that this interplay has not been coded in the system. As it could be seen as a particular side effect of a special initial configuration, we have represented, in the six panels of Figure 7, the same curves obtained with six other colonies (from C1 to C6).

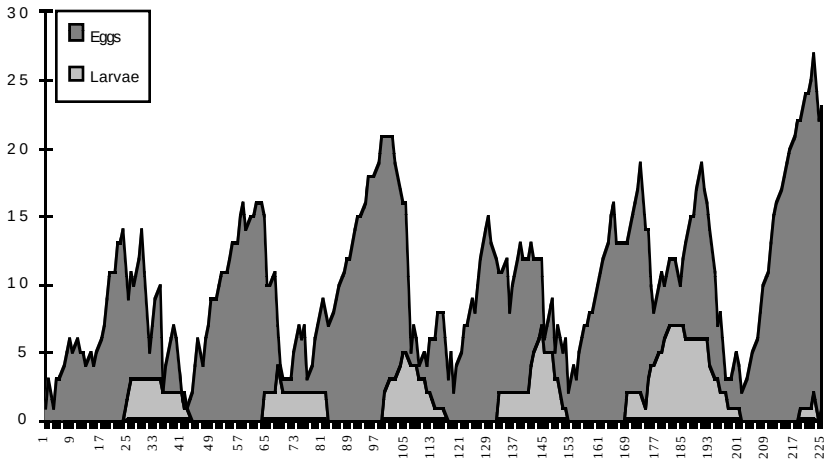

Fig 6 - Populations of eggs and larvae in colony C9

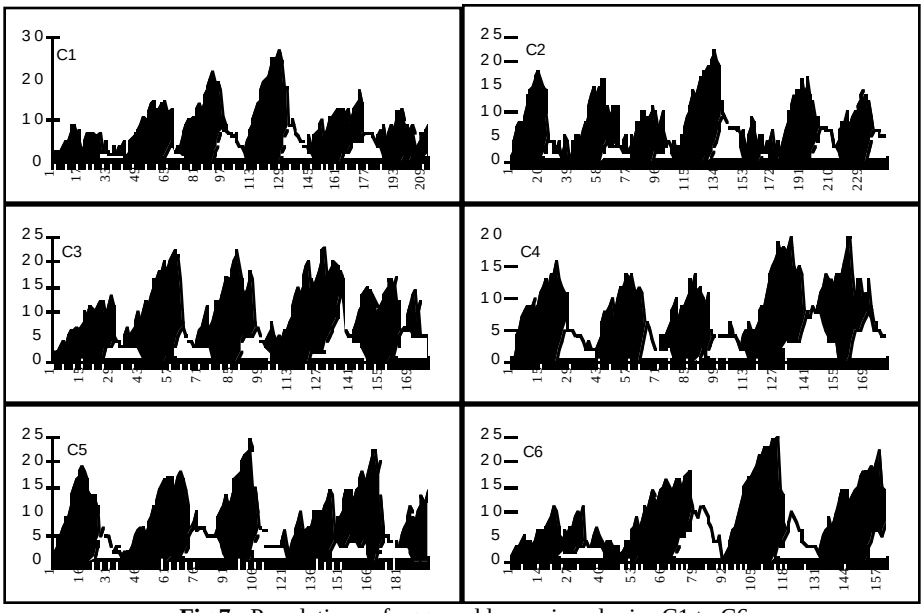

Fig 7 - Populations of eggs and larvae in colonies C1 to C6

All these diagrams clearly show that we obtain the same kind of population fluctuations, with small variations, in different cases. These fluctuations appear to be closely related to those observed in real colonies. As we did not put any global data in the simulations, we can obviously assume that these macro-properties are produced by the behaviors and interactions of the agents at the micro-level. And it is fascinating to see that only a few rules at the individual level can generate such complex patterns at the collective level. From a behavioral point of view, the comments that can be made about these curves are close to those already made for the failure cases, except that the queen succeeds in caring, carrying and feeding all the brood agents.

But an observer could probably credit the queen with much more cognitive capabilities than it is really provided with. As a matter of fact, the queen acts during the first months of foundation as if it anticipates that it will not be able to take care of all the brood at the same time and its behaviors could be interpreted as parts of the following strategy:

1) lay as much as possible eggs until the first larvae appear.
2) when the number of larvae reaches a sufficient threshold, neglect the previous eggs or convert them into alimentary eggs (which can be used to feed the larvae).
3) take care of the larvae until the first cocoons appear.
4) when the number of cocoons reaches a sufficient threshold, neglect the last larvae and begin to lay eggs.
5) try to lay as much as possible eggs while taking care of the cocoons, until the first workers appear.
6) when all the cocoons have become workers, start all over again (but now the workers are able to help the queen in foraging).

A strategy that could be summarized by the following sentence: “never deal with more than two types of brood agents at the same time”. And it is very interesting to see that this long-term strategy is simply generated by the interactions between the behaviors of the queen and the needs of the brood agents.

3. A Multi-Agent Reactive Chess Program: MARCH

3.1 Why a Distributed Chess ?

One cannot talk about strategy without talking about chess. Chess appears to be the best example of two-players game in which handling a global strategy is viewed as absolutely necessary for winning. For that reason, we decided to use a distributed approach to implement a chess program, in order to see whether or not local tactical behaviors could really replace a global strategy. Of course, our aim has never been to obtain a chessmaster. And we did not get one. We simply wanted to program a decent player, able to defend, attack and occasionally mate against average human players, whilst staying as simple as possible. The result, called MARCH (for Multi-Agent Reactive CHess), overpassed much of our expectations on many points (its simplicity/effiency ratio, its good behavior in critical situations, etc.), but appeared to be extremely limited on some others (in sacrificing too much pieces, for example), and we believe some of its drawbacks to be due to a lack of certain cognitive capacities. Since MARCH is, as far as we know, the first attempt to apply a multi-agent approach to chess, it is certainly too soon to draw any definitive conclusions from it. But we are convinced that this first steps towards a multi-agent chess program will be useful as a starting point to further works in the domain, which could reconciliate reactive and rationnal agents.

3.2 Description of MARCH

As for the N-Puzzle, our approach consists in viewing each chessman as an autonomous agent, with its own behavior and its own perception area. We then have six different types of agents: eight pawns, two knights, two bishops, two rooks, one queen and one king. Each of these agents knows the directions in which it can move, the place it is lying on and its material value (1 for pawns, 3 for knights and bishops, 4 for rooks, 10 for the queen, and infinite for the king).
Each place of the chessboard knows the chessman that is lying on it and holds two values called *whiteStrength* and *blackStrength.*. Il also holds the two differences between these values, respectively called *whiteDiff* and *blackDiff.*
A turn consists in:

(1) asking each chesspiece to:

(a) propagate a constant value (namely 10) on the places it directly threatens (generally those on which it could move, except for the pawns), thus increasing the whiteStrength or the blackStrength of the place, depending on the piece’s color.

(b) inform the enemy’s pieces it directly threatens that they are threatened by itself.

(2) asking each threatened pieces to propagate its material value (by adding it to the appropriate strength) on the places situated between itself and the pieces that threaten itself.

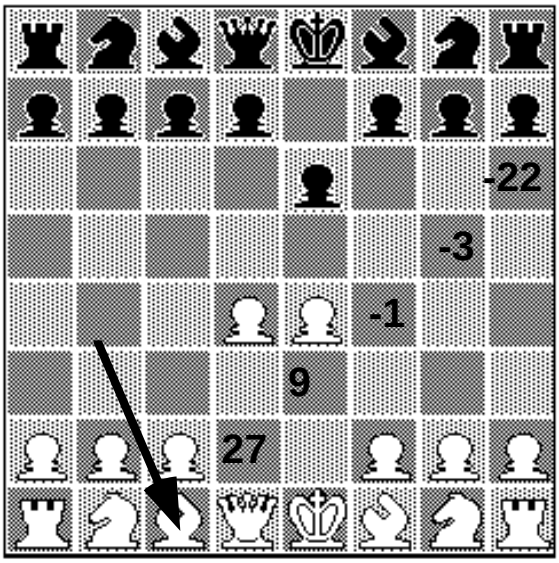

Fig 9 - An example of marks given by the white bishop to the places on which it could move

(3) asking each piece to give a mark to each place onto which it could move. This mark is computed as follows:

(a) it is firstly equal to the whiteDiff or blackDiff of the place, depending on the piece color.

(b) if there is any enemy piece on the place, its material value, multiplied by two, is added to the mark.

(c) the material value of each enemy piece it would threaten when located on this place is added to the mark, as well as the material value of each allied piece it would protect (except that of the king).

(d) Finally, the material value of the piece and the whiteDiff or black of its place are removed from the mark.

Note that some pieces may mark the places in a different way: the king always rejects places on which the opponent's strength is greater than zero (which means an enemy piece threatens the place); pawns add to the mark a bonus which corresponds to the line number of the place (from 3 or 4 at the beginning to 8 in the enemy lines), because they are inclined to move forwards.

(4) And finally choosing randomly a piece among the pieces whose mark are the greatest and asking it to move on the related place.

An example of marking is shown on Figure 9. The first place gets 27 from the white bishop, which corresponds to its whiteDiff (40 because it is protected by the King, the Queen, the Knight and the bishop), minus the whiteDiff of the bishop's place (10,

because it is protected by the queen) and minus the bishop's material value (3). The last place gets -22, which corresponds to its whiteDiff (-10, because it is protected by the bishop but threatened by a black knight and a black pawn), plus the material value of the threatened enemy pieces (1, because there is only a pawn), minus the whiteDiff of the bishop's place (10) and its material value (3).

This system of marking allows each piece to make a careful compromise between security and opportunism, defense and attack. Security is represented by the whiteDiff and blackDiff values, which indicate how much allied pieces are protecting the place and how much enemy pieces are threatening it. Opportunism and attack are represented by the possibilities of capture or threatens a place offers. Defense is represented firstly by whiteDiff or blackDiff, because threatened pieces will artificially increase the strengths of the places between them and their aggressor, thus attracting proportionaly to their material value allied pieces that could cover them, and secondly by the possibilities of protecting other pieces.

3.3 Experiments

The experiments with MARCH were firstly conducted by making it play two hundred games against an average human opponent. The program won 57 times, lost 83 times and obtained 60 draws. Almost all its defeats occured during the first stages of the game and a majority of its victories during the last stages, which means that it is quite bad in opening. However, once its pieces have been correctly deployed, it can play very well and perform clever attacks. Its main drawbacks are a certain predilection towards the systematic exchange of pieces (which partly explains the high number of drawn games), and a very poor defense in "good" situations (like openings).

The second set of experiments was conducted against the GNU Chess program, which is a quite strong player. MARCH did not win any of the fifty games played against it, and obtained only three draws. Once again, almost all its defeats occured during the first stages of the games.

The preliminary conclusions that we may draw from these experiments are as follows:

(1) We have demonstrated that a multi-agent reactive system can *play* chess with a level approximatively equivalent to that of an average human player (i.e. who learned chess in his youth and who does not find the time to train more than once a week...).

(2) Some emergent strategies (sequences of actions that seem to be coordinated) can be observed when the system plays (and essentially when it attacks), but they remain partial and do not last during all the game.

(3) When facing a strong strategist (like GNU chess), the program is not able to react in a coordinated way and quickly gets trapped.

(4) Obtaining good openings is a very difficult challenge, because the environment of the agents is totally open and not enough threatened by the opponent to make them react intelligently.

4. Conclusion

As we were saying above, MARCH represents the first step towards a strong multi-agent chess program. Our vision is that keeping only reactive agents in it will not allow MARCH to progress significantly. We have certainly explored, with this program, the limits of the emergent strategies that can be obtained from the interactions between reactive agents. Playing chess requires more than a punctual coordination between two pieces at a given time.

We do not say that it is not possible to obtain long-term emergent strategies with reactive systems: the example of the sociogenesis proves the contrary. But the difference between the agents that compose an ant colony and the agents that compose the chessboard is that the latter have to face a real strategy whose application is not restricted by any unexpected events or resources failures.

We have shown in this paper that, in many domains, a global strategy could be advantageously replaced by a set of tactical behaviors. This kind of results, although they have to be confirmed by further researches, constitute in our sense an interesting lower bound for studying the influence of individual behaviors on the collective behaviors and the interplays that occur between them.

Bibliography

(Brooks 1987) **R. Brooks**, "Planning is just a way of avoiding figuring out what to do next", MIT Working Paper 303, 1987.

(Delaye & al. 1990) **C. Delaye, J. Ferber & E. Jacopin** "An interactive approach to problem solving"*in* Proceedings of ORSTOM'90, November 1990.

(Drogoul & al. 1991) **A. Drogoul, J. Ferber, E. Jacopin** "Viewing Cognitive Modeling as Eco-Problem-Solving: the Pengi Experience", LAFORIA Technical Report, n°2/91

(Drogoul & Ferber 1992) **A. Drogoul & J. Ferber,** "Multi-Agent Simulation as a Tool for Modeling Societies: Application to Social Differentiation in Ant Colonies", *in* Proceedings of MAAMAW'92 (forthcoming "Decentralized AI IV").

(Drogoul & al. 1992b) **A. Drogoul, J. Ferber, B. Corbara, D. Fresneau,** "A Behavioral Simulation Model for the Study of Emergent Social Structures" *in* Towards a Practive of Autonomous Systems, F.J. Varela & P. Bourgine Eds, pp. 161-170, MIT Press.

(Drogoul & Dubreuil 1992) **A. Drogoul & C. Dubreuil** "Eco-Problem-Solving model: Results of the N-Puzzle", in (Werner & Demazeau 1992), pp 283-295.

(Drogoul & Dubreuil 1993) **A. Drogoul & C. Dubreuil** "A Distributed Approach to N-Puzzle Solving", to appear in the proceedings of the 13th DAI Workshop.

(Ferber & Drogoul 1992) **J. Ferber & A. Drogoul**, “Using Reactive Multi-Agent Systems in Simulation and Problem Solving”, in “Distributed Artificial Intelligence: Theory and Praxis”, L. Gasser eds.

(Wavish 1992) **P. Wavish** "Exploiting Emergent Behaviour in Multi-Agent Systems" *in* (Werner & Demazeau 1992).

(Werner & Demazeau 1992) **E. Werner & Y.Demazeau,** "Decentralized AI 3", North-Holland, June 1992.