Incremental Co-evolution of Organisms: A New Approach for Optimization and Discovery of Strategies

Hugues Juillé

Brandeis University, Computer Science Department
Volen Center for Complex Systems, Waltham, MA 02254-9110, USA
hugues@cs.brandeis.edu

Abstract. In the field of optimization and machine learning techniques, some very efficient and promising tools like Genetic Algorithms (GAs) and Hill-Climbing have been designed. In this same field, the Evolving Non-Determinism (END) model presented in this paper proposes an inventive way to explore the space of states that, using the simulated "incremental" co-evolution of some organisms, remedies some drawbacks of these previous techniques and even allow this model to outperform them on some difficult problems.
This new model has been applied to the *sorting network problem*, a reference problem that challenged many computer scientists, and an original one-player game named *Solitaire*. For the first problem, the END model has been able to build from "scratch" some sorting networks as good as the best known for the 16-input problem. It even improved by one comparator a 25 years old result for the 13-input problem. For the Solitaire game, END evolved a strategy comparable to a human designed strategy.

Keywords: Evolutionary optimization, Simulated co-evolution, Sorting networks.

1 Introduction

Simulation of the rules of life is an attractive and intuitive approach for the design of optimization tools or for machine learning. The evolution of a population of organisms is ruled by some operators that allow the exploration of the state space and a selection mechanism that works with respect to an objective function. Then, this simulated evolution may allow the emergence of specialized organisms or some complex behaviors. Genetic Algorithms are a perfect example of such a simulated evolution.
The aim of this paper is to describe an inventive model, called Evolving Non-Determinism (END), which proposes a new approach to explore the space of states. In fact, the END model can be seen as many simple competing organisms that interact locally one with each other and that become more and more complex with time, evolving using specialization. At each generation, they are evaluated according to a fitness function that allows most promising organisms to survive selection and to spread over.

The END model is extremely different from other well-known techniques: Genetic Algorithms (GAs), Hill-Climbing or Simulated Annealing (SA) in the way the information about the landscape (or the topology) of the space of states is used. This allows the END model to outperform these techniques in the case of problems for which there is only little information about the gradient of the landscape or for which state representation is problematic. Indeed, the main drawback of GAs is that crossover and mutation operators used to evolve genes may create genes that correspond to an invalid solution. For some problems, this

drawback can be such that the population size has to be very important to counterbalance this undesirable property. For Hill-Climbing and SA, some operators have also to be designed in order to evolve solutions by finding some new solutions in their neighborhood. The design of such operators may be very elaborate for some problems. Unlike these techniques, the END model doesn't care about solution representation or local neighboring solutions since solutions are generated incrementally and are always valid. In fact, we shall see that the space of states can be represented by a tree and that a solution is a path from the root of this tree to a leaf.

The END model has been applied on two difficult "real-life" optimization problems. Encouraging results described in this paper, let us expect that a broader field of applications can be tackled by this model. The first problem is the follow-up of an established problem for which several approaches ([1, 5, 10]) have been used to try to improve some 25 years old results concerning sorting networks [8]. Actually, this problem was also at the origin of an early paper [13] in which GAs were used to try to replicate Hillis's experiment ([5]) for the 16-input problem and in which some ideas of the END model were presented. The second problem is a very simple one-player game for which the player tries to find a strategy to get a score as large as possible.

This paper is organized as follows: Principles and parameters of the model are presented in details in Section 2 along with a comparison with other optimization techniques. Results for the two real-life problems are described in Section 3. Section 4 presents a summary and possible future research.

2 Evolving Non-Determinism

2.1 Principles

2.1.1 Description of Organisms

The END model simulates the co-evolution of a population of N organisms. These organisms live in a grid world, wrap-around, for which there are as many slots as organisms. Each slot is occupied by an organism and every organism works as a non-deterministic Turing machine since it makes random decisions.

2.1.2 Representation of the Space of States

The class of problems we propose to study is such that the space of states can be represented by a *tree of solutions*. The leaves of this tree correspond to all possible solutions to the problem and any solution is uniquely defined by a path from the root to the leaf to which this solution is attached. Such a path can be described by an ordered sequence of choices, each choice corresponding to the node that has been picked while incrementally generating the path. A problem for which the space of states can be represented by such a tree is called an *incremental problem* in the following of the paper.

Moreover, we assume the following properties for the tree of solutions:

- A *fitness* can be assigned to each leaf (and therefore to each solution),
- For each internal node, children nodes are *correct and fair*. That is, no leaf corresponds to a non-valid solution and there are no useless choice in the description of a solution.

In the following of the paper, we shall also use the *partial solution* term to specify the first choices (or the *prefix*) of the description of a solution.

2.1.3 Incremental Co-evolution of Organisms

The simulated co-evolution is a sequence of competition rounds. At each round, each organism generates a path from the root to a leaf in the tree of solutions associated to the problem at hand. This path is built incrementally, choosing uniformly randomly a child for each node encountered.
Then, each organism is seen as the member of a *species* and a *fitness* is assigned to it:

- The prefix of the solution an organism generated defines the species to which it belongs. The length of this prefix is defined by a parameter called *commitment degree*. The value of this parameter equals 1 at the beginning of the simulated co-evolution and its increasement is managed thanks to a global strategy.
- The fitness is defined as the fitness of the solution this organism has generated.

According to the value of the fitness, a selection is performed in the population of organisms. This selection is operated as follows:

- At each slot, the fitness of the occupying organism is compared to the fitness of organisms in the neighborhood.
- Then, the organism with the highest fitness in this neighborhood is copied into the slot providing this fitness is better than the one of the organism currently in the slot. If several organisms have the same fitness, one of them is picked randomly.
- Otherwise, the slot occupant remains unchanged (and has eventually been copied into slots of its neighborhood).

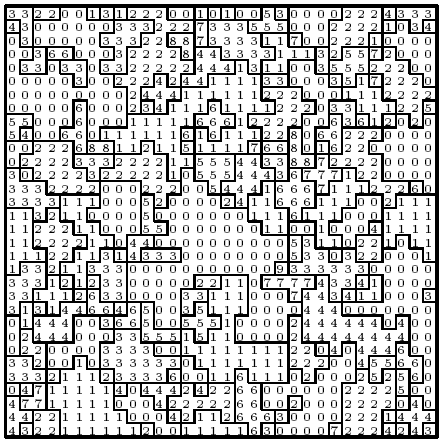
The idea of this selection is that an organism with a higher fitness probably made some better choices for the first choices of its solution. Clearly, this simulated co-evolution of competing species allows such organisms to duplicate themselves and to take the place of worse organisms.

Now that the selection is completed, for the next round, every organism builds another solution. However, organisms keep the same prefix as their previous solution and build a new solution beginning with this prefix. This process can be seen as a backtrack in the tree of solutions and the incremental building of a new random path from this point. Therefore, the species organisms belong to doesn't change. Then, another round of selection is performed.

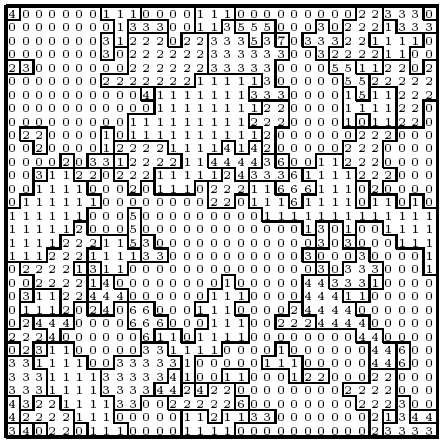
Figure 1 describes how such a population of competing species evolves. This simulation was performed in a 1024 slots world. There are 10 species competing; each species is represented by a number from 0 to 9. The problem tackled in this simulation is the sorting of integers: $\{0, \ldots, 9\}$ and the fitness is defined as the number of ascents in the sequence generated by organisms. This problem is used in [6] to analyze performance of the END model. Looking at this simulated co-evolution, we can see that, as rounds go off, the species represented by '0' takes more and more importance and it almost dominates all the other species after 16 rounds.

It is easy to understand that, proceeding in that way, organisms corresponding to best first choices are theoretically stronger than others during the selection step and therefore duplicate more often. So, their species take more and more importance in the total population. If we let this scenario run indefinitely, we can expect a species to overcome all other species and to be the only one in the population.

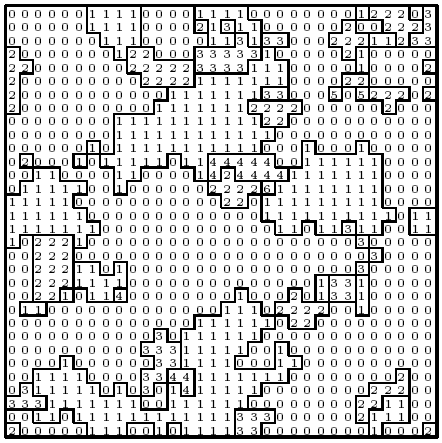
That is why the *commitment degree* parameter has been introduced. Increasement of the commitment degree corresponds to the specialization of organisms since it defines the length of the prefix used to identify species of organisms.



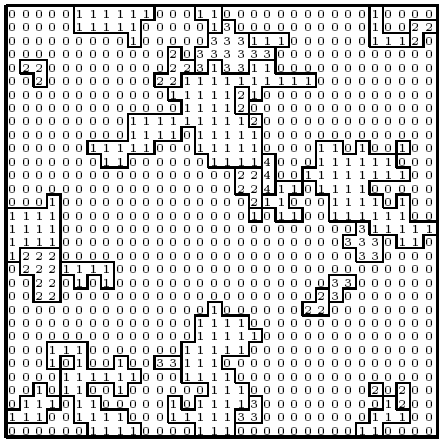
Step 5 - Disorder = 1794



Step 8 - Disorder = 1390



Step 12 - Disorder = 1046



Step 16 - Disorder = 812

Figure 1: *Simulation of the co-evolution of 10 competing species in a 1024 slots world.*

This parameter is managed by a global meta-level strategy discussed in the next section.

2.1.4 Strategy to Manage the Commitment Degree

In fact, there are no rules to find the best strategy. For our experiments, we designed two different strategies.

The first one is the simpler since the commitment degree is increased every n rounds, where n is fixed. n has to be chosen astutely so that good species have time to grow and to reach a significant size. The problem with this strategy is that the value of n is difficult to estimate *a priori*.

The second strategy uses a measure of the state of the model called *disorder measure*. The idea of this measure is to have a way to detect when a species overcomes others to a degree such that we can consider that this species corresponds to the first best choices and such that, after increasing the commitment degree, specializations of this overcoming species will be significantly represented. This measure is defined as the sum over all the slots of the number of organisms in the neighborhood that belong to a different species than the one of the occupying organism. This value tends to decrease as some species dominates the others. When this disorder measure reach a given threshold, the commitment degree is increased.

The drawback of this strategy is that it can take a long time for the disorder measure to reach the given threshold if there are some different first choices that are equivalent. This problem doesn't appear with the first strategy. That's why a combination of these two strategies offers often a good compromise.

Thus, as the commitment degree increases, organisms commit themselves in these earliest choices which seem to be the most attractive.

Finally, when the commitment degree can't be increased because it equals the length of the found out solution, the simulation stops.

2.2 Parameters of the model

From the description of the model in the previous section, it appears that a few parameters manage the model. These parameters are the following:

Population size: as this size grows, the number of solutions generated at each round increases. The size of the "sample" is more important and it is more likely that species corresponding to optimal solutions don't disappear because of a too small number of representatives.

Neighborhood used for selection: Unformally, the intuition behind this parameter is the following: If this neighborhood is large then we can expect that when a good solution is found out, it propagates more easily in the population. Therefore, the convergence to a good solution can be fast. However, a large neighborhood may forbid the discovery of a better solution since it shrinks the space of explored solutions. It is not the case when a small neighborhood is used. But, the drawback of the later case is that convergence is slower.

Management of the commitment degree evolution: the strategy used to schedule increasements of the commitment degree is very important. Indeed, it must be such that the number of representatives of "interesting" species at the next round is sufficient to expect them to overcome other species with a high probability.

All these parameters are analyzed experimentally in [6].

2.3 Comparison to other Optimization Techniques

As we have already said, our model doesn't exploit information about local gradient of the landscape. Indeed, once a solution is built, only first choices of this solution are used for the next round. That means that we don't try to get some better solution in the neighborhood of this solution, using some gradient information as it is the case for Simulated Annealing or Hill-Climbing.
This remark doesn't apply to Genetic Algorithms. However, the drawback of GAs are the operators used to evolve in the landscape: cross-over or mutation may make very difficult the search of an optimal solution if the new genotype doesn't correspond to a valid solution. In the case of some problems, like the two presented in this paper, this drawback is not negligible and good efficiency can only be reached if an extremely large population of genes is used.

To understand a little more easily how the END model works, let us make the following analogy: Children of the root of the tree of solutions can be seen as a partition of the space of states, each child (or species) corresponding to a particular sub-set of this partition. Then, the selection process allows species that generate a better solution on average (regarding the fitness) to dominate other species. Such species correspond to the domains of the space of states for which the mean value for the fitness is larger. Therefore, at this stage, details and gradient of the landscape of the space of states are not considered. Then, as the commitment degree increases, each domain is partitioned into smaller subdomains and, therefore, details of the landscape take more and more importance. Schraudolph and Belew ([12]) implemented a similar idea for GAs by tracking the convergence of the population to restrict subsequent search using a *zoom operator*.

Of course, it is easy to define a landscape such that this strategy doesn't work. For example, define a fitness such that the optimal correspond to a peak located in a region with a very low fitness and for which another region, far from this optimal peak, has a high average value. This strategy will be inclined to find out a local optimum in the region of high average fitness.
As we shall see later in this paper, the space of states for the sorting network problem has this kind of landscape.

However, the END model also has the ability to maintain a certain *degree of diversity*. This degree of diversity is directly related to the strategy used to manage the evolution of the commitment degree and it allows the model to not converge too quickly.

Finally, an analogy may also be done between the END model and a classical Artificial Intelligence heuristic search technique: *Beam Search* [2]. In this technique, a number of nearly optimal alternatives (the beam) are examined in parallel and some heuristic rules are used to focus only on promising alternatives, therefore keeping the size of the beam as small as possible. This technique proceeds like the END model in the sense that the search space is described by a directed graph in which each node is a state and each edge represents the application of an operator that tranforms a state into a successor state. A solution is a path from an initial state to a goal state.
Therefore, the problem modeling is very similar for the two techniques. However, the main difference is that the END model doesn't use some heuristic rules but only a fitness function that quantifies the degree of quality of a solution. Beam search uses heuristic rules to prune the set of alternatives at each step of the incremental building of a path to a goal state. This pruning is performed by the END model during selection but it is performed in an *auto-adaptive* way: no heuristic is provided to the model.
In fact, the END model has two important advantages compared to such an approach:

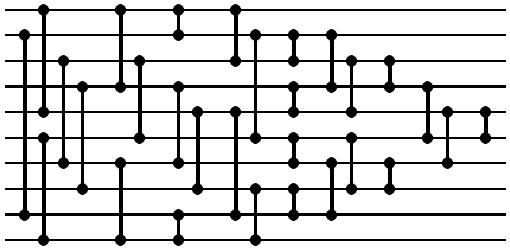


Figure 2: *A 10-input sorting network using 29 comparators and 9 parallel steps.*

- It doesn't need *a priori* knowledge of the topology of the space of states (no heuristics).
- A strategy is evolved by the model itself to find out an optimum, by eliminating "unpromising" species very soon and by maintaining diversity to keep attractive species. Therefore, an auto-adaptive behavior emerges while the model is searching for the optimum.

3 Results for Two Difficult Real-Life Problems

3.1 Sorting Networks

3.1.1 Presentation

An *oblivious comparison-based algorithm* is such that the sequence of comparisons performed is the same for all inputs of any given size. This kind of algorithm received much attention since they allow an implementation as circuits: comparison-swap can be hard-wired. Such an oblivious comparison-based algorithm for sorting n values is called an n-input *sorting network* (a survey of sorting networks research is in [8]).
There is a convenient graphical representation of sorting networks as the one in figure 2 which is a 10-input sorting network (from [8]). Each horizontal line represents an input of the sorting network and each connection between two lines represents a *comparator* which compares the two elements and exchange them if the one on the upper line is larger than the one on the lower line. The input of the sorting network is on the left of the representation. Elements at the output are sorted and the larger element is on the bottom line.
Performance of a sorting network can be measured in two different ways:

1. Its *depth* which is defined as the number of parallel steps that it takes to sort any input, given that in one step disjoint comparison-swap operations can take place simultaneously. Current upper and lower bounds are provided in [10]. Table 1 presents these current bounds on depth for $n \leq 16$.
2. Its *length*, that is the number of comparison-swap used. Optimal sorting networks for $n \leq 8$ are known exactly and are presented in [8] along with the most efficient sorting networks to date for $9 \leq n \leq 16$. Table 2 presents these results.
   The 16-input sorting network has been the most challenging one. Knuth [8] recounts its history as follows. First, in 1962, Bose and Nelson discovered a method for constructing sorting networks that used 65 comparisons and conjectured that it was best possible. Two years later, R. W. Floyd and D.

inputs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Upper	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9
Lower	0	1	3	3	5	5	6	6	7	7	7	7	7	7	7	7

Table 1: *Current upper and lower bounds on the depth of n-input sorting networks.*

E. Knuth, and independently K. E. Batcher, found a new way to approach the problem and designed a sorting networks using 63 comparisons. Then, a 62-comparator sorting network was found by G. Shapiro in 1969, soon to be followed by M W. Green's 60 comparator network (see [4] and [8]).

inputs	1	2	3	4	5	6	7	8
comparators	0	1	3	5	9	12	16	19
inputs	9	10	11	12	13	14	15	16
comparators	25	29	35	39	45*	51	56	60

Table 2: *Best upper bounds currently known for length of sorting networks. Previously, the best known upper bound for the 13-input problem was* 46

3.1.2 Previous works

Most of the previous works on sorting networks focussed on the 16-input problem.

The first person who used optimization technics to design sorting networks is W. Daniel Hillis. In [5] and [9], he used GAs and then co-evolution to find a 61-comparator, only one more sorting exchange than the construction of Green. However, Hillis considerably reduced the size of the search space since he initialized genes with the first 32 comparators of Green's network. Indeed, since the pattern of the last 28 comparators of Green's construction is not really intuitive, one can think that a better solution exists with the same initial 32 comparators.

In a previous paper, overviewing the END model [13], a 60 comparator sorting network was found out with the same initial conditions as Hillis, that is as good as the best known. Two attempts were also done on the 15-input problem and the model was able to find two 56 comparators sorting networks, again as good as the best known, with no initial conditions.

Ryan [11] applied a Genetic Programming approach to the problem of 9-input sorting networks. Kim Kinnear also used GP, but in the area of adaptive sorting algorithms, to find an algorithm to sort n elements in $O(n^2)$ time [7].

Ian Parberry ([10]) worked on the optimal depth problem and found out optimal values for 9 and 10-input sorting networks using an efficient exhaustive search executed on a supercomputer.

More recently, Gary L. Drescher ([3]) designed a novel fitness function for evolving sorting networks, using GAs. His approach finds quickly and reliably 60-comparator networks, as compact as the best known, again starting with the first 32 comparators of Green's construction.

3.1.3 Non-Deterministic Sorting Network Algorithm

The aim of this non-deterministic algorithm is to generate incrementally and randomly some valid sorting networks. Each organism runs this algorithm (see

```
Begin with an empty or a partial network
DO BEGIN
    Compute the set of significant comparators
    IF (set of significant comparators IS NOT empty)
        Pick one of these comparators randomly and add
        it to the existing partial network
    END_IF
UNTIL (set of significant comparators is empty)
/* A valid and fair sorting network has been generated */
```

Figure 3: Non-deterministic algorithm run by each organism.

figure 3), making some random choices until a valid sorting network is found. A run of this algorithm corresponds to the incremental construction of a path in the tree representing all valid and fair sorting networks; that is, valid sorting networks with no useless comparators.
Valid sorting networks are built using the *zero-one principle*:

Zero-one principle: If a network with n input lines sorts all 2^n sequences of 0's and 1's into nondecreasing order, it will sort any arbitrary sequence of n numbers into nondecreasing order.

Therefore, we only consider all 2^n possible inputs instead of the $n!$ permutations of n distinct numbers to incrementally build a sorting network.

The fitness of a sorting network is defined as its length. However, for the selection process, ties are broken using the depth of the sorting networks. In that way, we also generate some efficient sorting networks regarding the number of parallel steps.

3.1.4 Results

Experiments were performed on a Maspar MP-2 parallel computer. The configuration of our system is composed of $4K$ processors elements (PEs). In the maximal configuration a MP-2 system has $16K$ PEs. Each PE has a 1.8 Mips processor, forty 32-bit registers and 64 kilobytes of RAM. The PEs are arranged in a two-dimensional matrix.
This architecture is particularly well-adapted for our model since it is designed as a two-dimensional (grid) world. Each PE simulates one organism if we want to study a $4K$ population, but it can also simulate more organisms if we want a larger population.

Results for the 16-input problem initialized with the first 32 comparators of Green's sorting network will not be presented. The last version of the model is able to evolve a sorting network as good as the best known with a $4K$ population size and a success rate of almost 100% within 5 to 10 minutes. This time performance is comparable to Drescher's results ([3]) who used a 64-node CM-5 computer. Actually, the interesting comparison between his GAs approach and the END model is that:

- GAs evolved a population of 2^{19} $(= 524,288)$ sorting networks (compared to a 4,096 population size for END),
- 29 to 100 generations are enough for GAs to find the optimum but 150 to 200 generations are often required for END.

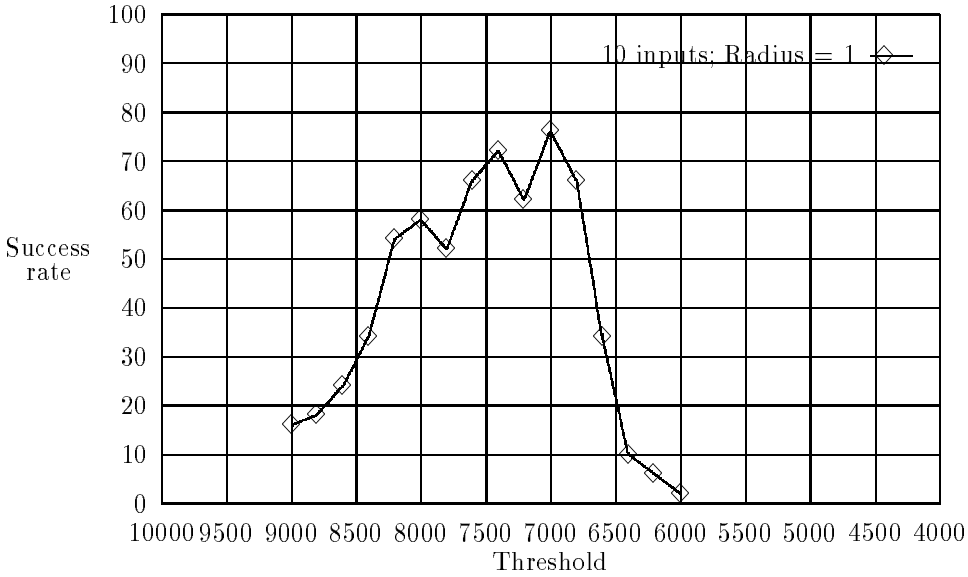


Figure 4: *Success rate for the 10-input sorting network problem versus the threshold value for the disorder measure strategy.*

In the following, only results for evolved sorting networks from scratch (*ie:* with no initialization) are described. In order to increase the probability of finding good sorting networks, we used a $64K$ population size, each processor of the Maspar simulating 16 organisms.

Before presenting results, let us come back to a previous observation. As it has been said in section 2.3, the landscape of the state space for the sorting network problem is such that optimal solutions don't correspond to the species which perform well for lower value of the commitment degree. That means that if we let the model evolve until there are only a few species remaining then the probability that the best solution be found out is very low. This behavior can be oberved in figure 4 which shows the evolution of the success rate as the value of the threshold decreases. First the success rate increases but, once the diversity degree is forced to be lower by decreasing the threshold value (the disorder measure strategy is used), the success rate also decreases.

This explains why one needs to maintain diversity to be able to find out an optimal solution. And of course, the larger the population size, the higher the probability of success.

We have been interested to tackle two different sorting network problems:

1. The 13-input problem because of the large gap between the best known result (46 comparators) and the 12-input one (39 comparators).
2. The 16-input problem because it is the most challenging one.

For the 13-input sorting network problem, we ran the END model 6 times with different values for the selection neighborhood radius (between 1 and 5) and the threshold. Each run was completed in about 8 hours. For 2 runs, we got a sorting network better than the best known. That is, the END model discovered two sorting networks using only 45 comparators, one comparator less than the best current known. Moreover, those two sorting networks use 10 parallel steps

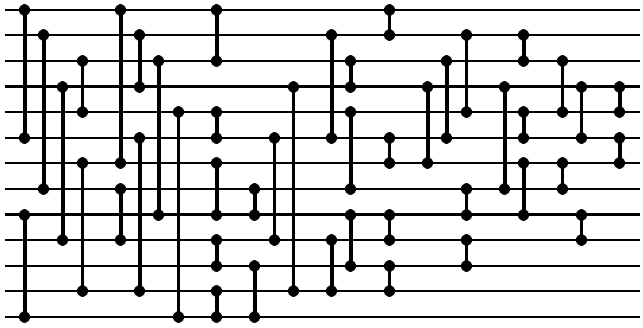


Figure 5: *A 13-input 45-comparators sorting network using 10 parallel steps.*

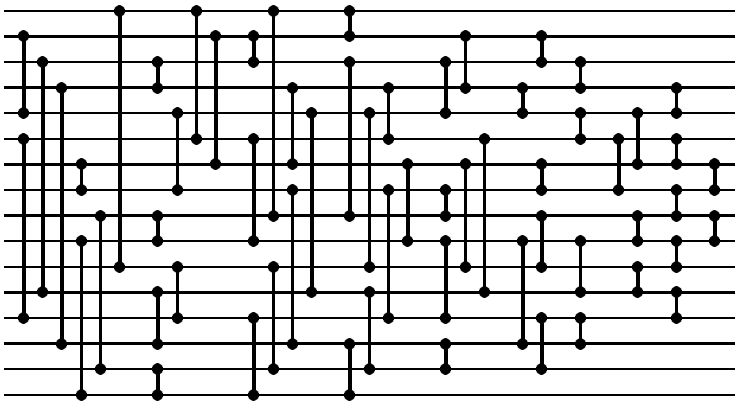


Figure 6: *A 16-input 60-comparators sorting network using 10 parallel steps.*

which is very good since to get smaller delay time one often has to add one or two extra comparator modules ([8]) and the best known delay for the 13-input problem is 9. Figure 5 presents one of these two 13-input sorting networks.

For the 16-input sorting network problem, we ran the END model 3 times. Each run was completed in a period of 48 to 72 hours[1]. For 2 runs, a 60 comparator sorting network was found out, each of them using 10 parallel steps. This is as good as the best human-built sorting network designed by M. W. Green. Those sorting networks were entirely designed from scratch by the END model (*ie:* there was no initial comparators as it was the case the previous times this problem was tackled using computers). Figure 6 presents one of these two 16-input sorting networks.

[1] A new algorithm that doesn't use the zero-one principle but that maintains the set of unsorted vectors using a list of masks has been recently implemented. This algorithm allowed us to divide execution time for the 16-input problem by a factor of about seven. Now, we can get reliable results within an execution time of 12 hours for a run. Using the maximal configuration for the Maspar, a run would take about 3 hours.

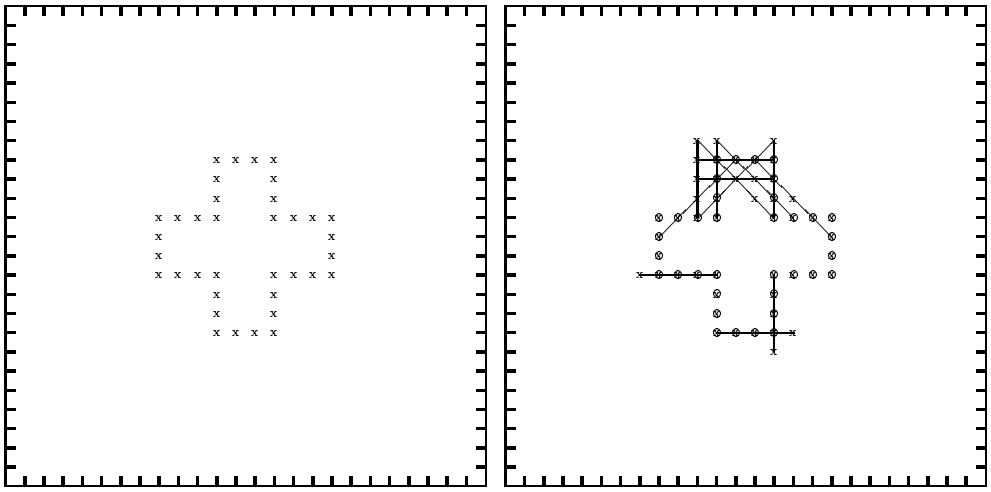


Figure 7: *The initial configuration and a possible configuration after 13 moves for the Solitaire game. For a clearer picture, the grid layout is not drawn but is represented by the rules on the border.*

3.2 A One-Player Game: Solitaire

3.2.1 Presentation of the game

This second problem is an original one and no one has published about it. Therefore, it is not possible to make some comparison. However, it is an interesting problem since it shows how difficult the modelling of a problem can be for other classical optimization techniques.

To play this game, you only need a piece of paper with a grid layout and a pencil. First, the player draws a fixed initial pattern composed of crosses, like the left picture in figure 7. The rule is that a cross can be added at an intersection of the grid layout if and only if it allows the drawing of a line composed of 5 crosses that do not overlap another line. This line may however be the continuation of another line or may cross another line. That is, the new line can share at most one common cross with another line. This new line can be drawn vertically, horizontally or diagonally.
The right picture in figure 7 shows a possible configuration of this game after a few moves. Crosses of the initial pattern are circled to be identified more easily. Now, the goal of this game is simply to draw as many lines as possible!

If this game is played by hand, one can see that a good strategy is difficult to elaborate. After a few plays, a score of 70-80 lines is relatively common. However, to reach 90 lines is less obvious and a score greater than 100 lines is exceptional.

Moreover, it can be proved that the maximum number of moves for this game is finite. However, no tight upper bound has been established for this optimum.

This game is interesting because it is a typical example of a problem wich seems impossible to code using an optimization technique like GAs or Hill-climbing. However, using an incremental approach, the description of solutions to this problem becomes obvious.

3.2.2 Results

The most recent result is a 117 lines game configuration. This result has been found out using a population of $4K$ organisms and required 30 hours of computation on the Maspar.
Figure 8 shows the final configuration along with the configuration of the game after the first 40 moves. It is very interesting to observe that most of the moves at the beginning of the play are located in the same area of the game board: this is the same as our best own strategy for hand-playing! This strategy is the result of the evolution of the population of organisms as a whole.

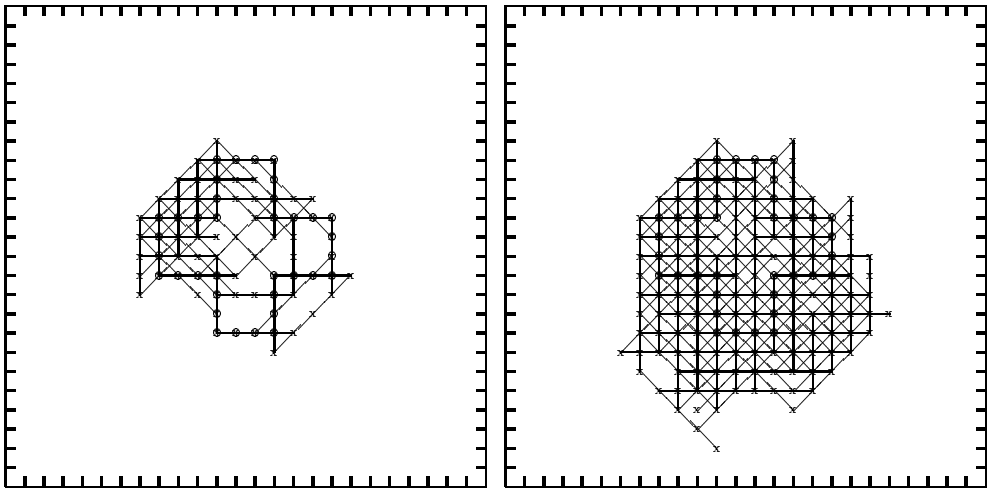


Figure 8: *The best configuration found out for the Solitaire game, using 117 lines (on the right); and the configuration for this best play after 40 moves (on the left).*

4 Conclusion

This paper presented an inventive and very promising technique. By using a new approach for the search in the state space and by constructing solutions incrementally, this model can outperform actual techniques. However, it should be noticed that when the topology of the search space for a given problem is well-known and its gradient is appropriate for Hill-Climbing or SA, these techniques are more efficient than the END model regarding execution time. Indeed, the approach of the END model is a statistical one which progressively takes into account details of the landscape; no gradient information is used. Moreover, analysis of the sorting network problem also reveals that the topology of the sub-space of valid networks makes the use of crossover and mutation critical. Therefore, it is difficult to take advantage of the "building block" mechanism exploited by GAs.

The aim of this paper was to present the parameters of the model. We focussed only on some elementary operators for selection and solution generating in order to identify clearly their importance for the efficiency of the model. The model described in this paper can be easily enhanced with some new features like:

- Allowing the use of some heuristics for solution generating to reduce the number of potential extensions at each node of the tree of solutions.
- Managing a local memory for each organism that would memorize its "past" and would allow learning.
- Each organism could look for a local optimum before selection rounds (using SA for example). When possible, this technique allows a faster convergence.

Moreover, we are currently working to replace the global strategy for the commitment degree management by a local strategy that would be managed by the model itself.

A very interesting advantage of the END model is that, intrinsically, it is highly parallelizable and its performance is related almost linearly to the size of the parallel system used.

Finally, the very encouraging results let us think about improving this approach and applying it on even more challenging problems like:

- multi-player games,
- problem solving,
- mechanical discovery of heuristics or theorems,

for which each organism would be an elementary and naive game player or problem solver. At each step, organisms would take a decision randomly but, as the result of the evolution of the population, some "high-level strategies" could be expected. This last idea needs however to be worked on.

5 Acknowledgments

I would like to thank Jordan Pollack, Patrick Tufts, Martin Cohn and Jacques Cohen for their advice and discussions. I would like also to thank Roger Gallier who challenged me one day with the Solitaire problem. Thanks also to the NSF whose grant allowed the Brandeis Computer Science to buy a Maspar computer. Finally, I want to thank my wife, Anne, for the moral support she provided me while I was working on this project and her constant curiosity.

References

[1] Richard Belew and Thomas Kammeyer: *Evolving Aesthetic Sorting Networks using Developmental Grammars.* In Proceedings of the Fifth International Conference of Genetic Algorithms.

[2] Roberto Bisiani: *Beam Search.* In Encyclopedia of Artificial Intelligence, Vol. 2, Second Edition, John Wiley & Sons, 1992.

[3] Gary L. Drescher: *Evolution of 16-Number Sorting Networks Revisited.* Submitted.

[4] Milton W. Green: *Some Improvements in Nonadaptive Sorting Algorithms.* Stanford Research Institute. Menlo Park, California.

[5] W. Daniel Hillis: *Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure.* In Artificial Life II, Langton, et al, Eds. Addison Wesley, 1992, pp. 313-324.

[6] Hugues Juillé: *Evolving Non-Determinism: An Inventive and Efficient Tool for Optimization and Discovery of Strategies.* Draft paper, Computer Science Departement, Brandeis University, 27 pp.

[7] Kim Kinnear: *Generality and Difficulty in GP: Evolving a Sort.* In Proceedings of the Fifth International Conference on Genetic Algorithms, S. Forrest, Morgan Kaufmann Publishers, 1993.

[8] Donald E. Knuth: *The Art of Computer Programming: Volume 3 - Sorting and Searching.* Addison Wesley, 1973.

[9] Steven Levy: *Artificial Life: the Quest for a New Creation.*Pantheon Books, 1992.

[10] Ian Parberry: *A Computer-Assisted Optimal Depth Lower Bound for Nine-Input Sorting Networks.* In Mathematical Systems Theory, No 24, 1991, pp. 101-116.

[11] Conor Ryan: *Pygmies and Civil Servants.* In Advances in Genetic Programming, Kim Kinnear, Ed. MIT Press, 1994.

[12] Nicol N. Schraudolph and Richard K. Belew: *Dynamic Parameter Encoding for Genetic Algorithms.* In Machine Learning, Vol. 9, 1992, pp. 9-21.

[13] Patrick Tufts and Hugues Juillé: *Evolving Non-Deterministic Algorithms for Efficient Sorting Networks.* Submitted.