Evolution of Non-Deterministic Incremental Algorithms as a New Approach for Search in State Spaces

Hugues Juillé
Computer Science Department
Volen Center for Complex Systems
Brandeis University
Waltham, MA 02254-9110

Abstract

Let us call a *non-deterministic incremental algorithm* one that is able to construct any solution to a combinatorial problem by selecting incrementally an ordered sequence of choices that defines this solution, each choice being made non-deterministically. In that case, the state space can be represented as a tree, and a solution is a path from the root of that tree to a leaf. This paper describes how the simulated evolution of a population of such non-deterministic incremental algorithms offers a new approach for the exploration of a state space, compared to other techniques like Genetic Algorithms (GA), Evolutionary Strategies (ES) or Hill Climbing. In particular, the efficiency of this method, implemented as the Evolving Non-Determinism (END) model, is presented for the *sorting network* problem, a reference problem that has challenged computer science. Then, we shall show that the END model remedies some drawbacks of these optimization techniques and even outperforms them for this problem. Indeed, some 16-input sorting networks as good as the best known have been built from scratch, and even a 25-year-old result for the 13-input problem has been improved by one comparator.

1 INTRODUCTION

In the field of optimization and machine learning techniques, some very efficient and promising tools like *Genetic Algorithms* (GA), *Evolutionary Strategies* (ES), *Hill Climbing* or *Simulated Annealing* (SA) have been designed. Each of these techniques uses a set of operators to generate new states (offsprings) from a current set of states (parents). These operators are crossover and mutation for GA, recombination and mutation for ES ([Bäck et al., 1991]), and "jump" to a local neighboring state for classical Hill Climbing or SA.
These approaches appear to be very efficient for many NP-complete problems like the Traveling Salesman Problem (TSP), the design of VLSI circuits or the Job Shop Scheduling problem (JSP). However, when the subspace of valid solutions is defined by a set of complex constraints, the design of a good representation for a state, and therefore the design of useful operators, can be very difficult. For example, efficient GA or ES implementations for TSP or the Number Partitioning problem ([Ruml et al., 1995]) involve non-trivial operators in order to introduce some problem-specific knowledge. This paper presents another well-known problem for which there is only little information about the topology of the subspace of valid solutions: the *sorting network* problem. Hillis ([Hillis, 1992]) and Drescher ([Drescher, 1994]) used GA to tackle this problem. However, because no operator is known that restricts the search in the subspace of valid sorting networks, it is only at the cost of a very large population size that significant results have been achieved. On the contrary, the searching technique presented in this paper manages to restrict the exploration only to valid solutions. Indeed, the state space that is explored is described by a tree in which leaves correspond to valid solutions and a path from the root of that tree to a leaf represents the sequence of choices necessary to generate the corresponding solution. So, this searching technique requires the design of an *incremental algorithm* which is able to generate any valid solution represented in the tree. Such an algorithm begins at the root of the tree and, at each step of the incremental construction, selects a node among the children of the current node until a leaf is reached. Then, a population-based model we called Evolving Non-Determinism (END) simulates the evolution of a population of such incremental algorithms for which the selection of the successor to the current node is performed *uniformly randomly.* The laws that drive the evolution of this population of *non-deterministic incremental algorithms* are such that individuals whose first choices seem to be more promising to generate a good solution reproduce more than others.

This paper presents the END model and compares its efficiency to GA for the sorting network problem. This is the follow-up of an established problem for which several approaches have been used to try to improve some 25 years old results ([Belew & Kammayer, 1993, Drescher, 1994, Hillis, 1992, Levy, 1992, Parberry, 1991]). Actually, this problem was also the origin of an early paper [Tufts & Juillé, 1994] in which GA were used to try to replicate Hillis' experiments ([Hillis, 1992]) for the 16-input problem and in which some ideas of the END model were presented. Encouraging results described in this paper show how the END model both outperforms GA for this problem and even lets us expect that a broader field of applications can be tackled by this model.

This paper is organized as follows: Principles and parameters of the END model are presented in details in Section 2. Section 3 describes the sorting network problem and Section 4 analyzes GA and the END model approaches for this problem. Section 5 presents a summary and possible future research.

2 EVOLVING NON-DETERMINISM

2.1 PRINCIPLES

In this paper, it is assumed that the search state space can be represented by a tree. Leaves of this tree represent valid solutions and internal nodes represent partial solutions (or steps necessary to reach valid solutions). There is a well-known AI algorithm for search in directed graphs and trees called *Beam search* ([Winston, 1984]). Beam search examines in parallel a number of nearly optimal alternatives (the *beam*). This search algorithm progresses level by level in the tree of states and it moves downward only from the best w nodes at each level. Consequently, the number of nodes explored remains manageable: if the branching factor in the tree is atmost b then there will be atmost wb nodes under consideration at any depth. The END model is similar to this algorithm in the sense that each individual of the population can be seen as one alternative in the beam. Moreover, a *fitness* (or score) must be assigned to internal nodes of the tree in order to determine which nodes will be explored further and which nodes will be ignored by the search algorithm. Here, beam search uses heuristics to score the different alternatives and to select alternatives that are most promising, i.e., the nodes with largest scores. However, this approach assumes the existence of a score operator used to evaluate the nodes.

The new idea proposed by the END model is to estimate the score of a given internal node by doing a random sampling from this node. That is, a path is constructed incrementally and randomly until a leaf (or valid solution) is reached. Then, the score of this solution is directly computed according to the problem objective function. If the problem is to find a shortest path, the score can be the length of the path necessary to reach the corresponding leaf in the tree where the different edges are eventually weighted according to a cost function. If the problem is to find a strategy to play a game, the score can be the evaluation of the game configuration defined by the leaf. Finally, the score of this non-deterministically and incrementally constructed solution is assigned to the initial node.

In fact, the END model performs the search by driving the evolution of the population of incremental algorithms according to the following procedure:

1. Initially, the search is restricted to the first level of the tree and each individual of the population randomly selects one of the first-level nodes.
2. Each individual scores its associated node (or alternative) by doing a random sampling as discussed above.
3. A *reproduction* round is operated in the population. The purpose of this operation of reproduction is to simulate the Darwinian mechanism of natural selection and survival of the fittest.
4. A test is performed and the result determines whether the level of search is increased by one or not. In the affirmative, each individual selects uniformly randomly one of the children of its associated node and this node becomes the new alternative assigned to the individual.
5. The search stops if no new node can be explored; otherwise it continues with step 2.

So, the simulated evolution is a sequence of competitive rounds. In the END model, the level of search in the tree is called the *commitment degree* since it corresponds to a commitment to the first choices of the incremental construction of an optimal solution.
The next two sections describe the reproduction operation and the management of the commitment degree.

2.2 REPRODUCTION

The mechanisms used for the reproduction operation have been inspired by the parallel architecture used for the implementation of the END model. Indeed, the current implementation of the END model uses a Maspar MP-2 parallel computer. This system is a SIMD 2D wrap-around mesh (i.e., a torus) architecture with a number of processor elements (PEs) ranging from 1K to 16K (our configuration is composed of 4K PEs).

So, our population has been modeled as a 2D wrap-around mesh where an individual is assigned to each point of intersection of the mesh and, therefore, has four neighbors.

Then, the reproduction operation is performed in the following way:

- Each individual compares its score with the score of the individuals in its neighborhood. This neighborhood is composed of the individuals whose Hamming distance from the first individual is lesser than a given value of the parameter: *radius*.
- If one individual in the neighborhood has a better score than all the others then it is copied instead of the current individual. If several individuals in the neighborhood have an identical score which is better than all the others then one is selected uniformly randomly and copied instead of the current individual. If no individual in the neighborhood is better than the current individual, then this individual remains unchanged.

So, if a given node of the current level of search in the tree is more likely to lead to a more promising solution then the reproduction operation allows individuals associated to this node to reproduce more than other individuals (therefore focussing the beam).

2.3 MANAGEMENT OF THE COMMITMENT DEGREE

Because of the reproduction operation, alternatives that seem to be more promising are represented by a larger and larger number of individuals. Ultimately, if one waits until all the individuals in the population agree on the same alternative before continuing the search on the next level, this means that the beam is reduced to the exploration of a single alternative. On the contrary, if the level of search is increased too frequently, the beam can become very wide. Indeed, in that case, many different alternatives are represented in the population and each alternative is represented by a small number of individuals. So, the disappearance of a promising alternative during the reproduction operation because of a non-favourable random sampling score evaluation is more likely.

Clearly, for the search to be efficient, these two extremes must be avoided. This goal is reached by using a strategy that manages the level of search. In fact, the purpose of this strategy is to drive the search by deciding when the commitment degree is incremented. In the current implementation, two strategies have been designed.

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 the number of individuals that correspond to better alternatives can 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 disorder measure evaluates the width of the beam. To compute this measure, each individual of the population counts the number of individuals among its four nearest neighbors that correspond to a different alternative than itself. Then, this number is summed over all the members of the population and the result is the disorder measure. This measure reflects the degree of convergence of the population. If a large part of the population corresponds to a few alternatives then this measure is small because most of the individuals are identical. On the other hand, if many alternatives are explored, and each alternative is represented by a few individuals, then the disorder measure is large. As individuals of the population focus on most promising alternatives because of the reproduction operation, the disorder measure decreases. When this measure reaches an arbitrarily fixed threshold (which is a parameter of the model), one considers that the width of the beam is small enough and the search can continue on the next level of the tree. The drawback of this strategy is that it can take a long time for the disorder measure to reach the given threshold if several alternatives lead to equivalent optimal solutions. This problem doesn't appear with the first strategy. Most of the time, a combination of these two strategies offers a good compromise.

2.4 PARAMETERS OF THE MODEL

The description of the END model in the previous section shows that it is characterized by the following parameters:

Population size : If the number of individuals in the population increases then the width of the beam can also be larger without decreasing the efficiency of the search. Therefore, the size of the explored state space is directly related to the size of the population.

Neighborhood used for reproduction : The size of the neighborhood for which the fitness of individuals is compared during a reproduction round drives the dynamics of evolution of the population. Indeed, if the radius of this neighborhood is large then the search focusses quickly on the best individuals and discards apparently less promising alternatives. On the contrary, a small radius allows the search to converge slowly. Therefore, this parameter manages the tradeoff between exploration and exploitation.

Management of the commitment degree : As it is described in the previous section, this parameter also plays an important rôle in managing the balance between exploration and exploitation.

The above description of the influence of these parameters on the search has been confirmed experimentally in [Juillé, 1994].

2.5 IDEA OF THE END APPROACH

Another approach to describe how the END model works is to 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 corresponding to a particular subset of this partition. Then, the reproduction process allows alternatives for which the score (or fitness) evaluation by random sampling is better on average to be represented by a larger number of individuals than other alternatives. Such alternatives 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 partitionned into smaller sub-domains and, therefore, details of the landscape take more and more importance. Schraudolph and Belew ([Schraudolph & Belew, 1992]) implemented a similar idea for GA 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, take a fitness function 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 it is shown in [Juillé, 1995], the landscape of the space of states for the sorting network problem is of this kind.

However, the ability of the END model to maintain diversity by managing the balance between exploration and exploitation allows it to be an efficient search algorithm.

3 THE SORTING NETWORK PROBLEM

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 has received much attention since it admits 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 network research is in [Knuth, 1973]).
There is a convenient graphical representation of sorting networks as shown in figure 1, which is a 10-input sorting network (from [Knuth, 1973]). 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 exchanges 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 largest element migrates to the bottom line.

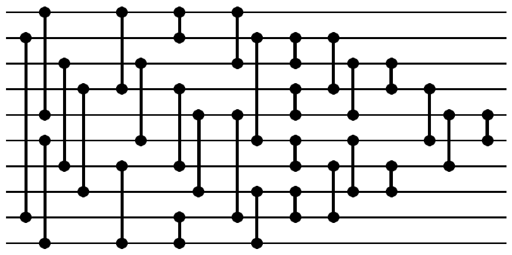

Figure 1: A 10-input sorting network using 29 comparators and 9 parallel steps.

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

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

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 [Parberry, 1991]. 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 [Knuth, 1973] 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 [Knuth, 1973] 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. E. Knuth, and independently K. E. Batcher, found a new method and designed a sorting network 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 [Green, 1969] and [Knuth, 1973]).

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.

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

4 Comparison of GA and END approaches

4.1 GA approach

4.1.1 Description

As in [Drescher, 1994] and [Hillis, 1992], the intuitive representation of sorting networks is that each genome encodes a sorting network as a sequence of pairs, each pair representing a comparator. Then, crossover is performed by exchanging groups of comparators between two mating individuals. A mutation consists in modifying one of the two indices defining a comparator. The fitness of individuals is then scored using the following criteria:

- Using the *zero-one principle* ([Knuth, 1973]), it is sufficient to test the 2^n possible binary input vectors (where n is the number of inputs of the sorting network) to determine if a sorting network is correct. Hillis ([Hillis, 1992]) and Drescher ([Drescher, 1994]) used the ratio of correctly sorted binary vectors to score the fitness of individuals in the population.

- Some comparators in the representation can be non-significant because they don't reduce the size of the unsorted vector set. This criterion is used by Drescher to allow convergence towards short networks. The technique used by Hillis is a little different because of his representation of sorting networks as pairs of chromosomes. Shorter networks are created when identical comparators occur at the same position in the two chromosomes of a pair.

Using this representation for networks, one can see that it is highly probable that crossover or mutation creates offsprings outside the space of correct sorting networks. For example, let us study the space of states in the very simple case of a 4-input sorting network. Table 3 presents, for networks of a given length, the proportion of valid sorting networks among all possible networks (restricted to networks for which no two consecutive comparators are identical) that constitute the state space explored by GA.

As the length of networks increases, the ratio of valid sorting networks increases since adding a comparator cannot transform a valid sorting network into an incorrect one. However, this ratio becomes very small when one comes close to the optimal length. In that case, the probability that crossover or mutation improves individuals is very low and, as is shown in the next section, the population size has to be large to counterbalance this undesirable property.

4.1.2 Results

Hillis and Drescher both tackled the 16-input sorting network problem. However, the size of the search space is considerably reduced since their population is initialized with the first 32 comparators of Green's network and this "prefix" is protected from changing. Indeed, since there are no regular pattern for the last 28 comparators of Green's construction, one can think that a better solution exists with the same initial 32 comparators. Moreover, since there are only 151 remaining unsorted vectors after this initial construction, the fitness can be computed within reasonable time.
Details of GA implementation are not relevant here and are described in [Drescher, 1994] and [Hillis, 1992]. Results along with population size and number of generations for Hillis and Drescher's experiments are presented in table 4.

Drescher's GA evolved sorting networks as compact as the best known. However, in all experiments, a very large population size is used to reach these results.
The next section presents experiments for the END model approach and shows that this model is able to tackle even much more complex instances of the sorting network problem.

4.2 END approach

4.2.1 Description

A non-deterministic incremental algorithm (see figure 2) is run by each individual of the population to generate incrementally valid sorting networks. 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*. So, only the 2^n possible binary input vectors need to be considered (instead of the $n!$ permutations of n distinct numbers).

The fitness of a sorting network is defined as its length. However, for the reproduction operation, ties are broken using the depth of sorting networks. In that way, efficient sorting networks are generated regarding the number of parallel steps.

```
Begin with an empty or a partial network N
DO BEGIN
  Compute the set S of significant comparators
  IF (S is not empty) THEN
    Pick randomly a comparator from S
    and append it to N
  END_IF
UNTIL (S is empty)
/* Now N is a valid and fair sorting network */
```

Figure 2: Non-deterministic incremental algorithm run by each individual.

4.2.2 Results

Experiments were performed on a Maspar MP-2 parallel computer. The configuration of our system is composed of $4K$ processors elements (PEs). The peak performance of this system for 32-bit integer computation is $17,000$Mips. In the maximal configuration a MP-2 system has $16K$ PEs and a peak performance of $68,000$Mips. Each PE simulates one individual if one wants to study a $4K$ population, but it can also simulate several individuals to evolve 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 ([Drescher, 1994]) presented in section 4.1.2. Actually, the interesting comparison between his GA approach and the END model is that:

- GA 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 GA to find the optimum but 150 to 200 generations are often required for END.

To show the efficiency of the END model for the sorting network problem, the construction of networks from scratch, i.e., without any initialization, has been studied. So, there is no restriction on the search in the space of states. Then, the END model has been able to find all best known upper bounds for the length of sorting networks (for the number of inputs in the range of 9 to 16) and even improved the upper bound for the 13-input problem.

Table 5 presents parameters and results of experiments for the 13-input and the 16-input problems [1].

[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 designed and implemented. This algorithm improves the execution time by a factor of about seven for the 16-input problem. Now, reliable results can be obtained within an execution time of 12 hours for this problem. Using the maximal configuration for the Maspar, a run would take about 3 hours.

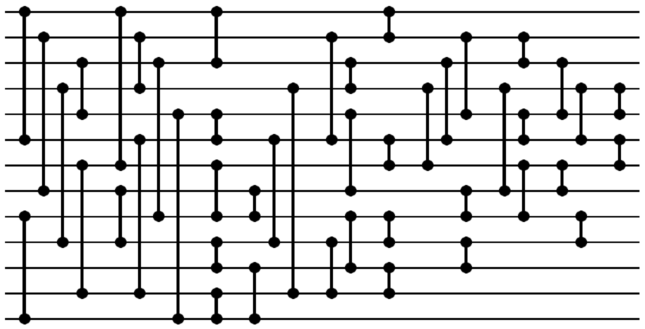
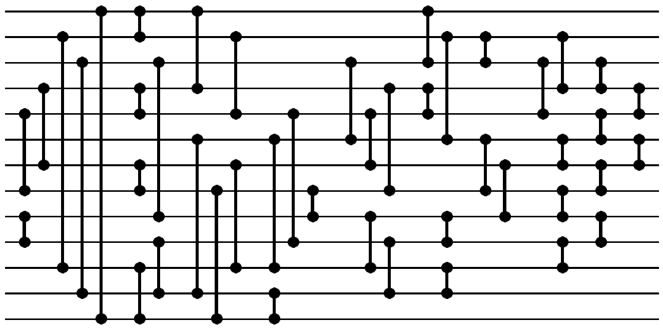

Figure 3: Two 13-input 45-comparator sorting networks using 10 parallel steps.

For the 13-input problem, the END model discovered two sorting networks using only 45 comparators (presented in figure 3), one comparator less than the best current known. Moreover, these two sorting networks use 10 parallel steps which is very good since to get smaller delay time one often has to add one or two extra comparator modules ([Knuth, 1973]) and the best known delay for 13-input sorting networks is 9.

For the 16-input sorting network problem, two 60 comparator sorting networks have been evolved from scratch, each of them using 10 parallel steps. This is as good as the best human-built sorting network designed by M. W. Green. Figure 4 presents one of these two 16-input sorting networks.

5 CONCLUSION AND FUTURE RESEARCH

This paper presented a new and very promising search algorithm. By using a population-based approach for the search in the state space and by constructing solutions incrementally, this model outperforms GA in the case of problems for which the constraints that define the sub-space of valid states are complex and result in a topology for which no useful operator is known to explore this sub-space efficiently. We were interested in studying the sorting network problem because it is a very challenging problem and results of experiments could be compared to the GA approach that have been used by Hillis and Drescher. Moreover, analysis of this problem also revealed that the topology of the sub-space of valid sorting networks makes the use of crossover and mutation harmful.

In another field, board games can also be considered as a subset of this class of "topologically" complex problems since rules of such games define valid configurations of the board and these valid configurations often represent only a very small subset of the whole set of possible configurations. In [Juillé, 1995], an example of a board game is presented to show how a strategy is evolved by the END model to play this game. Moreover, the END model is intrinsically highly parallelizable and scalable. Using a 2-D mesh architecture where each processor simulates one or several individuals, it is possible to evolve a very large population.

The END model could also be enhanced by adding some features like:

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

Finaly, we are currently working to replace the global strategy for the commitment degree management by a local strategy that would be managed by the individuals of the population themselves.

Acknowledgements

I am pleased to acknowledge the support of the W. M. KECK Foundation for the Volen National Center for Complex Systems. I would like to thank Jordan Pollack, Patrick Tufts and Martin Cohn for their valuable advice and discussions. Thanks also to the NSF whose grant allowed the Brandeis Computer Science Department to get 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 for her constant curiosity.

References

[Bäck et al., 1991] Bäck, T., Hoffmeister, F., & Schwefel, H.-P. (1991). A survey of evolution strategies. In Belew, R. K. & Booker, L. B. (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2–9, San Mateo, California. Morgan Kaufmann.

[Belew & Kammayer, 1993] Belew, R. K. & Kammayer, T. (1993). Evolving æsthetic sorting networks using developmental grammars. In Forrest, S. (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California. Morgan Kauffmann.

[Drescher, 1994] Drescher, G. L. (1994). Evolution of 16-number sorting networks revisited. Personal communication.

[Green, 1969] Green, M. W. (c.1969). Some improvements in nonadaptive sorting algorithms. Technical report, Stanford Research Institute, Menlo Park, California.

[Hillis, 1992] Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, C. et al. (Eds.), *Artificial Life II*. Addison Wesley.

[Juillé, 1994] Juillé, H. (1994). Evolving non-determinism: An inventive and efficient tool for optimization and discovery of strategies. Draft paper.

[Juillé, 1995] Juillé, H. (1995). Incremental co-evolution of organisms: A new approach for optimization and discovery of strategies. To appear in the proceedings of the Third European Conference on Artificial Life.

[Knuth, 1973] Knuth, D. E. (1973). *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison Wesley.

[Levy, 1992] Levy, S. (1992). *Artificial Life: the Quest for a New Creation*. Pantheon Nooks.

[Parberry, 1991] Parberry, I. (1991). A computer-assisted optimal depth lower bound for nine-input sorting networks. *Mathematical Systems Theory*, 24:101–116.

[Ruml et al., 1995] Ruml, W., Ngo, J. T., Marks, J., & Shieber, S. (1995). Easily searched encodings for number partitioning. To appear in the Journal of Optimization Theory and Applications.

[Schraudolph & Belew, 1992] Schraudolph, N. N. & Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9:9–21.

[Tufts & Juillé, 1994] Tufts, P. & Juillé, H. (1994). Evolving non-deterministic algorithms for efficient sorting networks. Poster, Artificial Life IV Conference. First 60-comparator results.

[Winston, 1984] Winston, P. H. (1984). *Artificial Intelligence*. Addison-Wesley. Second edition.

Table 3: Ratio of valid sorting networks in the space of all possible networks

Network length	4	5	6	7	8	9	10
Size of state space	750	3,750	18,750	93,750	468,750	2,343,750	11,718,750
Valid sorting networks	0	12	840	11,580	105,000	776,412	5,097,960
Ratio	0.0	0.0032	0.0448	0.12352	0.22400	0.331269	0.435026

Table 4: Results for Hillis and Drescher's GA experiments

	W. David Hillis	Gary L. Drescher
Population size	65,536	524,288
Number of generations	up to 5,000	29 to 100
Results	61 comparators using co-evolution of parasites	60 comparators, 100% success for 10 consecutive runs, 6 constructions use 10 parallel steps (like Greens's sorting network)
Parallel computer used	64K processor CM-1	64-node CM-5
Execution time	100 to 1,000 generations per minute	5 to 18 minutes

Table 5: Results for the END model experiments for the 13-input and the 16-input problem

	13-input problem	16-input problem
Population size	65,536 each PE simulates 16 individuals	65,536 each PE simulates 16 individuals
Number of generations	160 to 250	300 to 500
Neighborhood radius for reproduction	3 or 4	5
Results	Number of runs: 6 For 2 runs: 45 comparators	Number of runs: 3 For 2 runs: 60 comparators
Execution time	about 8 hours for each run	about 48 to 72 hours for each run

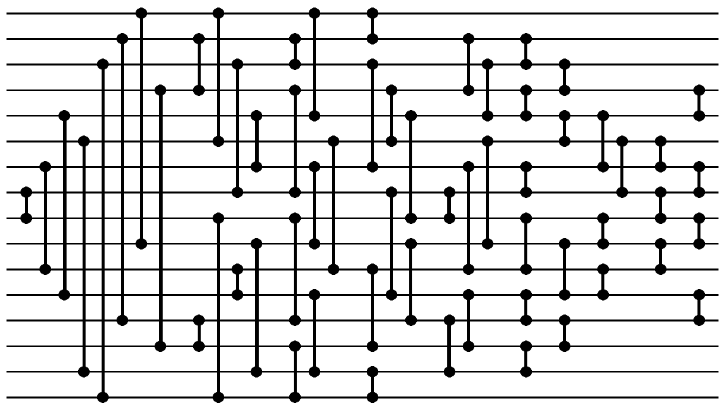

Figure 4: A 16-input 60 comparator sorting networks using 10 parallel steps.