Assessing the Evolutionary Algorithm Paradigm to Solve Hard Problems

Ph. Preux* E-G. Talbi†

Abstract

Evolutionary Algorithms (EAs) are meta-heuristics based on the natural evolution of living beings. We wish to assess their use as intractable problems solver. In this paper, we first describe EAs suited to this task. We put the emphasis on the two main issues that are faced: representation of the solutions (and the related issue of operators), and the combined use of EAs with other search methods, leading to *hybrid EAs*. Grounded on our experience, we also strongly claim the need of a recognized (and used) framework to assess search methods.

Keywords: Evolutionary algorithms, Genetic algorithms, Hybrid algorithms

1 Introduction

Evolutionary Algorithms (EAs) have been imagined in the 60's in three places, with different goals in mind:

- in Germany, both H-P. Schwefel [Sch81] and H. Rechenberg [Rec73] proposed the evolution strategies (ES) to optimize real-valued functions,
- at UCSD, L.J. Fogel, A.J. Owens, and M.J. Walsh [FOW66] introduced the evolutionary programming (EP) techniques in the field of machine learning. Evolutionary programming has then been forgotten for 20 years, recently re-appearing,
- at the University of Michigan, J.H. Holland [Hol75] developed the genetic algorithms (GA) to model and understand the *adaptation in artificial and natural species*.

*Laboratoire d'Informatique du Littoral, BP 719, 62228 Calais Cedex, France, preux@lil.univ-littoral.fr

†Laboratoire d'Informatique Fondamentale de Lille, URA CNRS 369, Cité scientifique, 59655 Villeneuve d'Ascq Cedex, France, talbi@lifl.fr

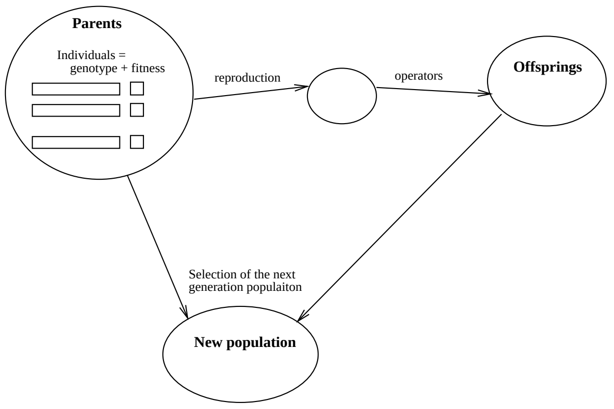
Parents
Individuals = genotype + fitness
reproduction
operators
Offsprings
Selection of the next generation populaiton
New population

Figure 1: The generic principle of evolutionary algorithms. At all stages, actions are performed stochastically

Though invented separately and having been used by totally separate communities, these three kinds of algorithms shared much resemblances. This point has recently been emphasized [SDB+93] when the three communities have become aware of each other. Among the three, GAs have gained much popularity and have been considered as function optimizers for quite a long time (first works on this issue is reported in A.D. Bethke's PhD [Bet81]). They have originally been designed in a very minimal fashion (*canonical GA*). From that point, they have greatly evolved while keeping the basic algorithm reproducing the cycle of life on a population of individuals: reproduction (involving *genetic operations*), selection/survival of the fittest (hence, death of the unfit). Each individual has a fitness which characterizes its ability to reproduce itself. The individual is defined by its genotype on which the operators are acting. This genotype is associated to the phenotypical (exhibited) traits of the individual which, when evaluated using an objective function, gives the individual's fitness (see figure 1). Individual genotypes are initialized at random. For us, EAs are algorithms based on this basic scheme. Individuals of the population may be any kind of data structure. Genetic operations may be any operation manipulating and changing individuals.

EAs have been successfully applied to numerical functions optimizations, specially massively multi-modal functions (see [Mic92]). Their application to the resolution of combinatorial optimization problems have also been studied. The traveling salesman problem is the first well-known work in the field. Scheduling problems (flow-shop, job-shop, and related design of timetables, or schedules for aircraft landing), graph coloring, graph partitioning, set partitioning, quadratic assignment, and SAT problems have been subject of many works with various success when tackling real-world sized problems. Our experience in using GAs to solve robot motion planning problems (robot with six or more degrees of freedom) [ATBM93][BATM93] shows that GAs are well adapted to search for solutions in high dimensionality search space with many optimal solutions.

In this paper, we will restrict ourselves to symbolic intractable problems. We will first focus on the use of EAs as a basic search algorithm. We will consider their weaknesses and provide some answers. These weaknesses are two-folds: first, the representation of solutions, and the operators that are used should be designed in close relationship with the structure of individuals; second, a criticism of the way EAs explore the research space will lead us to hybridize them. Aiming at assessing the potential of EAs for combinatorial optimization, we will emphasize the need for an experimental framework to be able to compare works that are done in the field using different search algorithms.

The remainder of the paper is organized as follows. In section 2, we will present the background about the utilization of EAs to solve combinatorial optimization problems. In section 2.1, we will discuss issues with regards to the structure of individuals which represent solutions of the problem that is to be solved, as well as the genetic operations (known as *genetic operators*) that build new solutions to be tested.

Unfortunately, as we will see later (see section 2.2), when one tries to compare his results with other people's results, he soon faces a cruel disappointment: there is no benchmarks which is used by every one; it is almost impossible to understand precisely what the other people have done and the results they have really achieved. For instance, using stochastic algorithms, it is quite useless to say that a given solution has been found if we do not know how often it is found if we want to assess the usability of these algorithms. Further, points of interest lie in the ratio between the number of sampled points with regard to the number of points in the research space.

When trying to discuss the pros and cons of a search method, reading related bibliography often turns out to read statements such as: "my algorithm $\mathcal{X}$ obtains good result when solving the problem $\mathcal{Y}$" which seems a real performance according to the author's faith, while keeping silent that there exists an algorithm $\mathcal{Z}$ that solves $\mathcal{Y}$ much more easily! However, we think that no algorithm (or class of algorithms/heuristics) can be better than any algorithm on a wide spectrum of problems (and recent theoretical results comfort this feeling — see [WM95]). Each algorithm or class of algorithms explores and exploits the research space in its own way; it will perform well for certain problems, and bad for (much more) other problems. Hence, we think that we should combine several search algorithms to make the search more efficient, or to have higher odds to find good optima. This combination of several algorithms, or *hybridization*, will be detailed in section 3, being one of the key-points of our works.

2 EAs for NP problem solving

Since their origins, genetic algorithms[1] have been designed as manipulating binary strings by the way of operators (mainly *crossover* and *mutation* in actual implementations) acting mechanically on them. By the term *mechanically*, we mean that these operators act without regards to the fitness of individuals. Theoretical arguments have been "obtained" stating that the binary representation was the most efficient representation that could be used in a GA (see [Hol75] for the first presentation of points known as "schemata processing", "N^3 argument", or "building-block hypothesis", and [Gol89] for a comprehensive discussion of these results). Even though, we, and many others, are very skeptical about these "results", we briefly discuss how a binary representation may been used.

Using a binary representation, some NP problems have been tackled for which this representation is very natural (graph partitioning problem, maximum clique size, ...). We briefly review and discuss De Jong & Spears study of the SAT problem [DS89].

In the individual, each variable of the expression to be satisfied is associated a bit. The fitness of the individual may be trivially computed by evaluating the boolean expression with the valuation of variables as stated in the individual. However, this evaluation leads to a 0/1 result which is quite poor to guide the algorithm towards a solution: the objective function returns 0 for almost all points of the space. Hence, the authors propose an other objective function which gives more interesting results which considers the clauses of the expression to satisfy and rates the evaluation according to the number of clauses that are satisfied by themselves. This function gives better results. However, the problems that are studied are quite small (maximum 105 variables). This restricts the interest of this work: as a canonical NP-complete problem, any NP-complete problem may be reduced to SAT. However, the reduction of an interesting, realistic problem to SAT quickly involves many hundreds or thousands variables. The proposed method does not appear scalable to such large problems. Furthermore, we think that SAT is not suited to EA techniques due to its "needle in a haystack" nature: EAs may efficiently search a space if it can grasp on clues to attain points of high fitness.

If binary representation naturally fits to SAT and some other problems, for lots of other problems, the use of a binary representation is rather counter-intuitive. Most authors who have experimented the resolution of symbolic problems with GAs claim that one should use the most natural representation to represent individuals (see [Dav91]). Furthermore, they also claim that one should define new operators, suited to the representation and to the problem itself. The representations that have been used range from a simple list of numbers to any kind of data structure, including arrays, variable size lists, or even Lisp functions in

[1]we speak here of genetic algorithms because evolution strategies have not substantially been applied to any non-numerical problem, and evolutionary programming has not been studied during 20 years. Hence, all the works in the field has been based on genetic algorithms, without even knowledge of the existence of the other two algorithms.

the sub-field of genetic programming [Koz92].

We now turn to discuss and exemplify these "non canonical GAs" to solve NP-hard problems.

2.1 Non canonical GAs

Initially, the most studied NP-hard problem has been the Traveling Salesman Problem [Gre87]. Very naturally, each town being numbered by an integer, a valid tour is a permutation of N integers, where N is the number of towns involved in the problem. Hence, an individual has been encoded as a list of N integers. It was then felt as obvious that standard binary operators were neither useful, nor valid, and that new operators ought to be devised, suited to this representation. Various recombination operators have been proposed which were dedicated to permutation of integers (CX, OX, and PMX operators [MGSK88]). For mutation, swapping two integers in the list gives a new valid tour. A recombination operator has also been proposed, the edge recombination operator [WSS91], which has yielded the best known results on the TSP with GAs to date). Edge recombination preserves edge information when producing an offspring. However, this operator does not act directly on the individuals. Instead, an "edge map" is build with two individuals to recombine and this data structure is used to create the two offsprings. This illustrates one general principle that is used for certain problems, the *indirect encoding*, that is, individuals are first decoded giving a new data structure which is used to produce offsprings.

A debate between using direct or indirect encoding is running with regards to the job-shop-scheduling problem (JSP for short). For the JSP, using a direct encoding scheme, an individual represents a schedule, whereas using an indirect encoding scheme, an individual should be decoded to obtain a schedule which may then be evaluated. Recombination may act either on the individuals themselves, or individuals may be decoded to produce a schedule on which recombination act. This decoding may be non-deterministic (see [DPT95] for a review of some usable encodings).

Of course, using such elaborate representations (either direct or indirect), it becomes clear that specific operators have to be used in order to take advantage of the information that are provided in the individuals. These *enhanced* operators are generally dedicated to a particular problem. For instance, addressing the JSP, Nakano and Yamada have used the Giffler and Thompson algorithm [GT69] to perform recombination with great success based on a direct representation [NY92].

Actually, it is today widely recognized among EA users that the representation has to be chosen in full accordance with operators. It should be noted that if the recombination operator acts blindly (without regards to the individuals it manipulates), then the representation should preferably be chosen with the building-block hypothesis as a guideline, whereas this hypothesis may be forgotten if recombination "analyzes" the individuals it manipulates. Briefly speaking, the building-block hypothesis aims at explaining how GAs work by saying that parts of genotypes that provide high fitness to the individuals to which they belong (building-blocks) are combined (by the recombination operator) to build still fitter individuals.

Even using non binary representations and well-tuned operators, there has not been to date evidences that GAs alone can actually perform better than otherly known methods to solve NP problems. However, some works have been performed on algorithms combining several search methods, these algorithms being qualified as *hybrid*. These algorithms already proved very efficient both when applied on academic problems, and industrial problems. Before going any further on hybrid algorithms, we wish to briefly point out the difficulty to assess search methods objectively.

2.2 Real assessment

We have felt that it is quite a hard task comparing results of different authors to asses the efficiency of any algorithm. Two main issues may be raised:

1. how can we objectively discuss the relative strength of 2 different algorithms? Because of the importance with regards to how a heuristic searches a research space (in other words for EAs, which operators are used), the performance of the same basic algorithm may vary dramatically. For their part, formal analysis are not relevant to address this issue today.

2. which problems should we use for benchmarks? These problems should be used by most researchers. Furthermore, we should use problems that, at least on average, are not particularly suited to one kind of search heuristic because of some idiosyncrasies of the research space. The nature of research space landscapes might provide useful analysis tools (see [Jon95] for a recent step towards this direction).

Aiming at comparing our results to others of the EA community, we have met huge problems to fill in a comparative summary table (see table 1) including some important parameters.

In EAs, many interacting factors have rather unknown effects (and non-additional effects) on the algorithm performance such as the population size, the reproduction and selection schemes, the constitution of the initial population. Further, to be able to compare stochastic algorithms, we are interested in the density of probability to find a given optimum and the number of iterations (or number of evaluated points which is somewhat more difficult to know) needed to find an optimum. This number of evaluations should be compared with the number of points in the research space.

Reference	MT10 × 10				MT20 × 5			
	best	number of sampled points	pop size	prob. success	best	number of sampled points	pop size	prob. success
[NY92]	930				1184			
[DP92]	938				1178			
[FRC93]	949				1189			
[JW94]	937				1174			
[Soa94]	997	20000	100					
[DPT95] (ours)	953	500000	250		1180	500000	250	

Table 1: Comparing results from different authors is not easy! According to what we have found in several articles, we display all found characteristics. We have not provided the probability of success for our results because we have not realized enough experiments to obtain meaningful figures. MT$xx \times yy$ means Muth and Thompson JSP problem made of xx jobs of yy operations [MT63].

3 Hybrid EAs

EAs more or less simulates a natural process[2]. As such, they possess a certain dynamic which is inherent to the process, regardless of details related to individual representation or the way individuals are acted upon by operators. One basic property of this process is that the population it acts on is said to "converge", that is to become more and more uniform[3] [Pre94]. Starting with a random population, all the individuals become grossly identical after a certain amount of time. The uniformization of genotypes is correlated to a stabilization of the mean fitness of the population. If we sketch the evolution during time of the mean fitness of the population, we see very clearly that this stabilization is quite fast (see figure 2).

3.1 Sequential hybridization

A fundamental practical remark is that after a certain amount of time, the population is quite uniform and the fitness of the population is no longer decreasing, the odds to produce fitter individuals being very low (250 generation on figure 2). That is, the process has fallen into a basin of attraction from which it has a (very) low probability to escape.

This point leads us to raise two issues:

1. once fallen in a basin, the algorithm is not able to know if it has found the optimal point, or if it has fallen into a local optimum. Hence, we need to find ways to escape the optimum in order to try to find an other optimum

[2]as far as we understand this natural process, and we are able to simulate it

[3]we leave aside niching techniques that aim at avoiding this uniformization and permit an EA to find, and maintain, multiple optima during the same evolution.

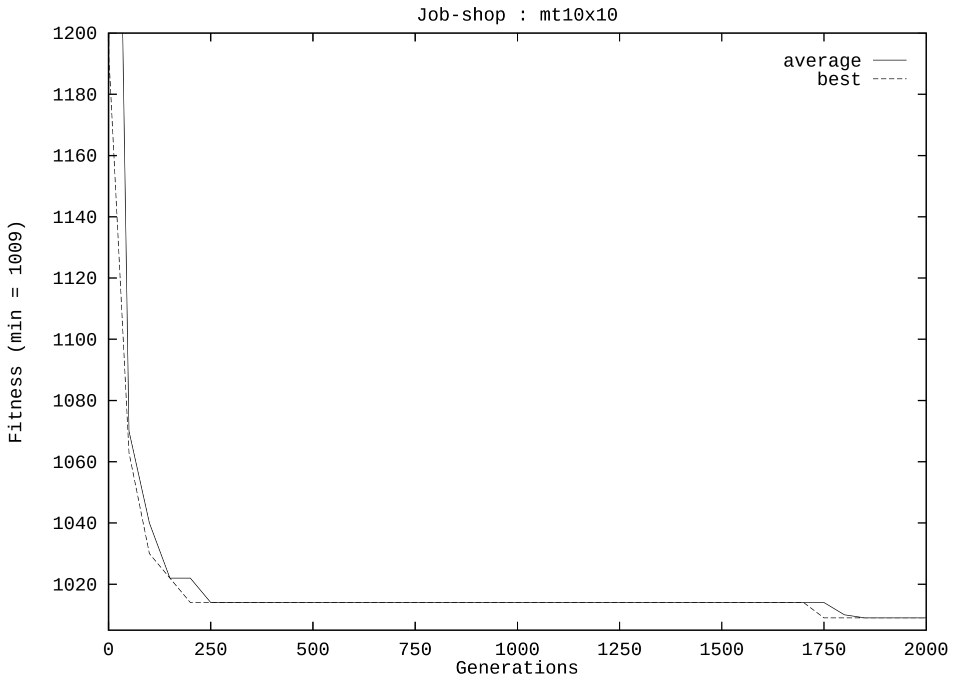

Figure 2: Evolution during time of the mean fitness of the population of a genetic algorithm solving a JSP (MT10 $\times$ 10). We have represented the mean fitness of the population (plain line) as well as the fitness of the best individual found so far (dashed line). This evolution is typical, regardless of the problem that is solved, the representation of individuals, or the operators that are used. This evolution is a fundamental property of the process at work in GAs.

2. the exploitation of the already found basin of attraction to find as efficiently as possible the optimal point in the basin

The first point may be solved by restarting the EA with a new population hoping that the EA will not fall into the same basin. Eshelman [Esh91] reports enhanced results using such a technique. This is quite natural since restarting is equivalent to performing several runs. Hence the odds to find the optimum are multiplied by the number of runs.

With regards to the second point, it is experimentally clear that the exploitation of the basin of attraction that has been found may be more efficiently performed by an other algorithm than by an EA. Hence, it is much more efficient to use a local search algorithm such as a hill-climbing or a tabu search (see figure 3). This schema of algorithm is qualified *hybrid*. More precisely, we distinguish different kinds of hybridization: this one we qualify *sequential hybridization* (SH). Basically, this phrase means that a set of algorithms is applied one after an other, each using the output of the previous as its input, acting in a pipeline fashion (see figure 3). The sequential hybridization may use a greedy algorithm to generate a good initial population for the EA (see figure 3(b)).

We have performed experiments on the graph partitioning problem using the TABU algorithm exploiting the result found by a GA [TMS94]. This greatly enhances the search

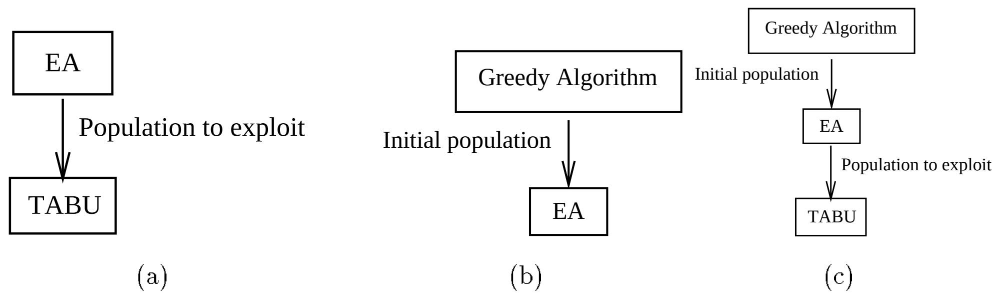

Figure 3: Sequential Hybridization. Three instances of this hybridization scheme are represented. There may be more than three algorithms to be pipelined.

performed either by the GA, or the TABU alone (see table 2 column 5 compared to columns 2, 3, and 4). The problem of SH lies on deciding when to stop one algorithm and trigger the next one. Waiting for the stabilization of the search fitness is feasible. However, there may be a time before actual stabilization when the process is already engaged in a basin which it can not escape. Triggering the next algorithm from that earlier moment might prove more efficient. Furthermore, as shown in figure 2, because of the still acting mutation, there may be further improvements of the solution after a relatively long stable phases (around generation 1750 on this example). Triggering the next algorithm during the first stabilization may lead to miss this further improvement of the evolutionary search. This leads to open issues which we are currently concerned of.

We now turn to other ways of hybridization.

3.2 Parallel hybrid EAs

According to the size of problems we wish to study, it is rather tempting to consider parallel implementations of EAs. Hence, there are several ways to parallelize the basic EA. However, care should be taken to distinguish mere parallel implementations of sequential EAs from implementations of parallel EAs. The former aims at keeping the essence of the sequential search whereas the latter forms a new sub-class of EAs behaving differently from the sequential one. A detailed presentation of parallel GAs is well beyond the scope of the present article (see [Tal95] for a comprehensive treatment of this issue). Rather, we will focus here on the parallel models as various kinds of hybridization of the basic EA. We introduce a classification of these models, each class being thought of as addressing different issues.

3.2.1 Parallel synchronous hybridization

The most immediate idea that comes to mind is to use some search method as one operator of the EA. In this case, instead of using a blind operator acting regardless of the fitness of the original individual and the operated one, we use an operator which is a search algorithm that considers the individual as the origin of its search, applies itself, and finally replaces the

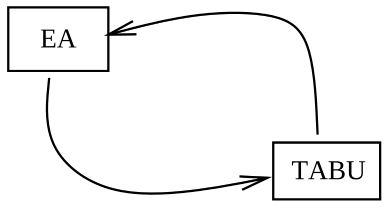

Figure 4: Parallel Synchronous Hybridization. For instance, a TABU search is used as a mutation operator in an EA.

original individual by the enhanced one[4]. The search algorithm may be a simple hill-climber, a TABU search, or even some kind of simulated annealing algorithm. We call this kind of hybridization a *parallel synchronous hybridization* (PSH).

However, the use of such hybridized algorithm may lead to severe problems of premature convergence, short-cutting to some extent the exploration phases.

We have obtained good results on the graph partitioning problem hybridizing a GA with a TABU search (see table 2 column 6).

We consider PSH as a first step towards hybridization. In several occasions, PSH has provided better results than other methods on difficult problems.

3.2.2 Parallel asynchronous hybridization

We now turn to an other kind of hybridization on which we are extensively working today, the *parallel asynchronous hybridization* (PAH). Basically, the PAH scheme involves several algorithms performing a search of the research space (or a sub-space of it), and cooperating to find an optimum. Intuitively, PAH will ultimately perform at least as well as one algorithm alone, more often perform better, each algorithm providing information to the others to help them.

We distinguish two different types of PAH:

- **homogeneous** in which case all the cooperating algorithms are the same
- **heterogeneous** in which case different algorithms are used

From an other point of view, we can also distinguish two kinds of cooperation:

- **global** all the algorithms search the same research space. The goal is here to explore the space more thoroughly
- **partial** the problem to be solved is decomposed into sub-problems, each one having its own search space. Then, each algorithm is dedicated to the search of one of these spaces. Speaking in general terms, the sub-problems are all linked with each other, thus

[4]This kind of operators is qualified *lamarckian*

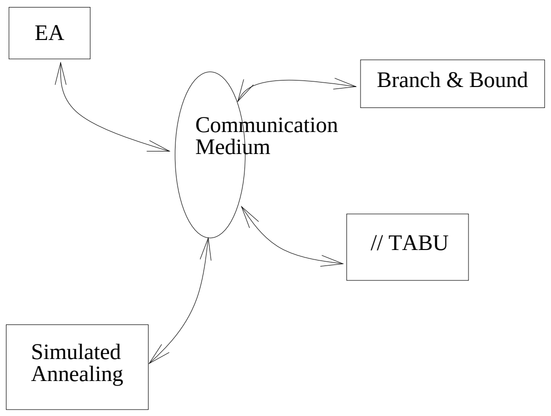

Figure 5: Parallel Asynchronous Heterogeneous Hybridization. Several search algorithms cooperate, co-adapt, and co-evolve a solution.

	GA	HC	TABU	SH GA/TABU	PSH GA/HC
Solution quality (min)	13.37	19.37	7.37	7.37	
Solution quality (max)	22.37	44.37	18.37	13.37	
Solution quality (mean)	18.67	30.47	12.67	11.97	13.45
Solution quality (deviation)	7.07	63.88	18.41	6.41	
Search time (sec.)	438.7.9	995	1031.7	913.1	3407

Table 2: Graph partitioning with GAs, Hill-Climber (HC), TABU, and 2 kinds of hybrid EAs (Benchmark: Split a binary tree of 127 vertices into 4 partitions). The figures are averaged over 10 experiments. See [TMS94] for details.

involving constraints between optima found by each algorithm. Hence, the algorithms communicate in order to respect these constraints and build a global viable solution to the problem.

Whether homogeneous or heterogeneous, global or partial, the key questions in the PAH lie in the design of the communication medium which allows the different algorithms to exchange information. Issues that should be addressed are:

- which individuals to exchange?
- when?
- how should they be handled by the other algorithms?

3.2.3 General hybridization schemes and applications

We have presented a classification of hybridizations. However, these hybridizations should be regarded as primitives that can be combined in different ways:

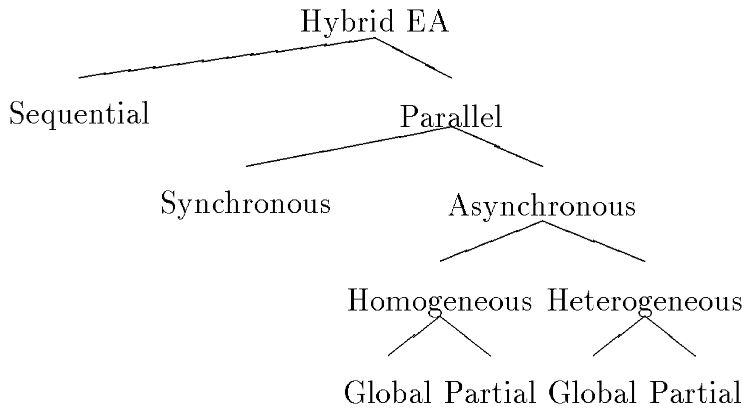

Figure 6: A classification of hybrid EAs.

1. P(S+A)H : a parallel asynchronous hybrid where EAs use search algorithm as operators
2. S+PAH, or S+PSH : a parallel hybrid where the population is initialized using the result of a search method
3. S+P(S+A)H : a combination of both previous schemes
4. ... : actually, any thoughtful combination of previously presented schemes

An application of partial homogeneous PAH has been done for the JSP [HMW90]. The search algorithm is a GA. Each GA evolves individuals of a specie which represent the process plan for one job. Hence, there are as many cooperating GAs that there are jobs.The communication medium collects fitted individuals from each GA, and evaluates the resulting schedule as a whole, rewarding the best process plans.

D. Levine has used a P(S+A)H scheme in his PhD to solve set partitioning problems (SPP) [Lev94] which proves very efficient at solving big sized SPP in real world applications (airline crew scheduling).

3.2.4 Parallel hybrid implementations

Parallel hybridization schemes ideally provides novel ways to parallelize EAs by providing parallel models of the algorithms. Hence, instead of merely parallelizing and finely tuning a sequential algorithm which has, though important, however limited capabilities to be parallelized, parallel hybrids are inherently suited to parallel computer environments. Furthermore, it should be pointed out and emphasized that the PAH proposes natural ways to efficiently implement algorithms on heterogeneous computer environments which is recognized as a very tough problem today.

As a first step, we have implemented a parallel asynchronous homogeneous hybrid GA on a network of workstations [Duv94]. The evaluation of this tool is on its way. The easiness to use such kind of parallel architecture has been acknowledged. Today, we are experimenting with heterogeneous PAH to solve QAP [Bac95]. A GA is cooperating with a multi-TABU algorithm, that is a set of TABU searching concurrently. To solve the QAP, a binary representation of individuals is ideally suited. Hence, the GA has been implemented on a massively parallel 16K processor Maspar MP-1 following a cellular synchronous model while the TABUs are running concurrently on a 16 Dec-alpha farm (actually, any network of any number of workstations may be used). As raised above, the work concentrates here on the study of the communication medium.

4 Conclusion

Studying EAs to solve combinatorial optimization problems, we have presented our current work in the field. We have argued for the use of hybrid algorithms. Our early experimental results clearly show that this may prove as a fruitful research path. Representations, operators, and hybridization schemes constitute the central issues of this work. Among other hybridization schemes, parallel asynchronous hybrid algorithms are strongly appealing for many reasons:

- following our natural insight, cooperation between individuals proves, on the long run, an efficient strategy.
- asynchronicity adds an other level of stochasticity which has not been thoroughly explored to date. This implies non determinism. The lack of global control is naturally replaced by the cooperation of methods.
- this model ideally meets the challenge of the implementation on parallel computer environments.

We have also briefly discussed the difficulty that we meet when we want to assess objectively the power of EAs. We strongly urge for the definition of a recognized framework that would be used by everyone in the field. Large size meaningful benchmarks are required. Reflections on what may be an objective assessment of heuristics is also required.

References

[ATBM93] J. M. Ahuactzin, E. G. Talbi, P. Bessière, and E. Mazer. *Geometric reasonning for perception and action*, chapter Using genetic algorithms for robot motion planning, pages 84–93. Number 708 in Lectures Notes in Computer Science. Springer-Verlag, 1993.

[Bac95] Vincent Bachelet. Algorithmes génétiques hybrides pour le QAP, 1995. DEA thesis, Laboratoire d'Informatique Fondamentale de Lille (à paraître).

[BATM93] P. Bessière, J. M. Ahuactzin, E. G. Talbi, and E. Mazer. Global planning with local methods: The ariadne's clew algorithm. In *IEEE Int. Conf. on Intelligent Robots and Systems IROS-93*, Yokohama, Japan, Jul 1993. IEEE Press.

[Bet81] A. D. Betkhe. *Genetic Algorithms as Function Optimizers.* PhD thesis, University of Michigan, Ann Arbor, MI, USA, March 1981. Dissertation Abstracts International 41(9), 3503B (University Microfilms No 8106101).

[Dav87] Lawrence Davis, editor. *Genetic Algorithms and Simulated Annealing.* Research Notes in Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, USA, 1987. ISBN: 0-934613-44-3.

[Dav91] Lawrence Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, USA, 1991. ISBN: 0-442-00173-8.

[DP92] U. Dorndorf and E. Pesh. Evolution-based learning in a job shop scheduling environment. Technical Report RM 92-019, Rijksuniversiteit Limburg, The Netherlands, 1992.

[DPT95] D. Duvivier, Ph. Preux, and E-G. Talbi. Application des algorithmes génétiques au problème du job-shop-scheduling. Technical report, Laboratoire d'Informatique du Littoral, 1995.

[DS89] Kenneth A. De Jong and William Spears. Using genetic algorithms to solve NP-complete problems. In *[Sch89]*, pages 124–132, 1989.

[Duv94] David Duvivier. Algorithmes génétiques sur calculateurs parallèles, 1994. DEA thesis, Laboratoire d'Informatique Fondamentale de Lille.

[Esh91] Larry J. Eshelman. The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In *[Raw91]*, pages 265–283, 1991.

[For93] Stephanie Forrest, editor. *Proc. of the Fifth International Conference on Genetic Algorithms*, Urbana-Champaign, IL, USA, July 1993. Morgan Kaufmann, San Mateo, CA, USA.

[FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Adaptation.* Wiley, New York, 1966.

[FRC93] Hsiao-Lan Fang, Peter Ross, and Dave Corne. A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In *[For93]*, pages 488–493, 1993.

[Gol89] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[Gre87] John J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In *[Dav87]*, pages 42–60. 1987.

[GT69] B. Giffler and G. L. Thompson. Algorithms for solving production scheduling problems. *Operations Research*, 8:487–503, 1969.

[HMW90] Philip Husbands, Frank Mill, and Stephen Warrington. Genetic algorithms, production plan optimisation and scheduling. In *[SM91]*, pages 80–84, 1990.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems.* Michigan Press University, Ann Arbor, MI, 1975.

[Jon95] Terry Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, University of New Mexico, Santa Fe, NM, USA, May 1995.

[JW94] Ari Juels and Martin Wattenberg. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. Technical Report csd-94-834, University of California, 1994.

[Koz92] John R. Koza. *Genetic Programming.* A Bradford Book. MIT Press, Cambridge, MA, USA, 1992. ISBN: 0-262-11170-5.

[Lev94] David Levine. *Parallel Genetic Algorithms for the Set Partioning Problem.* PhD thesis, Argonne National Laboratory, Maths and Computer Science Divivsion, May 1994. report ANL 94/23.

[MGSK88] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–85, 1988.

[Mic92] Zbigniew Michalewicz. *Genetic Algorithm + Data Structure = Evolution Programs.* Springer-Verlag, 1992.

[MM92] R. Männer and B. Manderick, editors. *Proc. of the Second Conf. on Parallel Problem Solving in Nature.* Elsevier Science Publishers, Amsterdam, 1992.

[MT63] J. F. Muth and G. L. Thompson. *Industrial scheduling.* Prentice-Hall, Englewood Cliffs, N.J., 1963.

[NY92] Ryohei Nakano and Takeshi Yamada. A genetic algorithm applicable to large-scale job-shop problems. In *[MM92]*, pages 281–290, 1992.

[Pre94] Philippe Preux. étude de l'uniformisation de la population des algorithmes génétiques. In *Actes de Évolution Artificielle'94*, Toulouse, France, September 1994.

[Raw91] Gregory J. E. Rawlins, editor. *Workshop on the Foundations of Genetic Algorithm and Classifier*, Bloomington, IN, USA, 1991. Morgan Kaufmann, San Mateo, CA, USA.

[Rec73] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog Verlag, Stuttgart, 1973.

[Sch81] Hans-Paul Schwefel. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[Sch89] J.D. Schaffer, editor. *Proc. of the Third International Conference on Genetic Algorithms*, Bloomington, IN, USA, 1989. Morgan Kaufmann, San Mateo, CA, USA.

[SDB+93] William M. Spears, Kenneth De Jong, Thomas Bäck, David B. Fogel, and Hugo de Garis. An overview of evolutionary computation. In *Proc. of the First European Conf. on Machine Learning*, 1993.

[SM91] H-P. Schwefel and R. Männer, editors. *Proc. of the First Parallel Problem Solving in Nature.* Springer-Verlag, Berlin, 1991.

[Soa94] C. Soares. Evolutionary computation for the job-shop scheduling problem. Technical Report UU-CS-1994-52, Utrecht University, The Netherlands, December 1994.

[Tal95] E-G. Talbi. Algorithmes génétiques parallèles : Techniques et applications. *Ecole CAPA95, à paraitre dans Parallélisme et applications irrégulières, edition Hermes, Cauterets, France*, Fev 1995.

[TMS94] E. G. Talbi, T. Muntean, and I. Samarandache. Hybridation des algorithmes génétiques avec la recherche tabou. In *Evolution Artificielle EA94*, Toulouse, France, Sep 1994.

[WM95] David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[WSS91] Darrell Whitley, Timothy Starkweather, and Daniel Shaner. The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. In *[Dav91]*, chapter 22, pages 350–372. 1991.