Towards hybrid evolutionary algorithms

Ph. Preux

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

E-G. Talbi

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

Abstract

Metaheuristics have received considerable interest these recent years in the field of combinatorial optimization. However, the choice of a particular algorithm to optimize a certain problem is still mainly driven by some sort of devotion of its author to a certain technique rather than by a rationalistic choice driven by reason. Hybrid algorithms have shown their ability to provide local optima of high quality. Hybridization of algorithms is still in its infancy: certain combinations of algorithms have experimentally shown their performance, though the reasons of their success is not always really clear. In order to add some rational to these issues, we study the structure of search spaces and attempt to relate it to the performance of algorithms. We wish to explain the behavior of search algorithms with this knowledge and provide guidelines in the design of hybrid algorithms. This paper briefly reviews the current knowledge we have on search spaces of combinatorial optimization problems. Then, we discuss hybridization and present a general classification of the way hybridization can be conducted in the light of our knowledge of the structure of search spaces.

Key words: Evolutionary algorithms, Genetic algorithms, Hybrid algorithms, Parallel meta-heuristics, Combinatorial Optimization.

1 Introduction

Metaheuristics have received considerable interest in the recent years in the field of combinatorial optimization (see Osman and Laporte's 1996 review [28] which contains more than 1400 references). Even if some papers regularly

pretend that a certain algorithm beats all the others on a problem, most researchers are rather confident that only hybrid algorithms can actually lead to really interesting performance in the field. Combinations of simulated annealing [8], tabu search [17], and genetic algorithms [27] have provided very powerful search algorithms (see C. Reeves' book [35], or V. J. Rayward-Smith & *al*'s book [33] for a recent account of this field). However, hybriding algorithms is sort of an art: which algorithms should be combined?, how should they be combined?, how should each algorithm be written? are some of the crucial issues. To date, there has not yet been any attempt to provide a global view of the art of hybridization, nor guidelines to help in the choice of how to hybridize and which algorithms should be combined to obtain the best result on a certain instance of a problem. Such an ambitious goal is far beyond the scope of this paper though we wish to make a step towards this direction. We have begun to study the structure of some $\mathcal{NP}$-hard problems as well as the way local search algorithms are working. We have also made our best to work towards a taxonomy of hybrids and the relation between the structure of a problem and the performance of hybrids. The goal of this paper is to present our current state of reflexion on these issues. Though this classification is centered around evolutionary algorithms, in its spirit, it is not restricted to hybrid algorithms having an evolutionary algorithm in it. We exhibit an important class of hybrids which have been rather poorly studied to date. We discuss it in the light of recent works we have been doing which are very promiseful and raise many questions both at the conceptual level and at the implementation level.

The paper is organized as follows. First, we will review the work dealing with the structure of combinatorial optimization problems. We will take some time to review our own recent work that aims at filling the gap between the knowledge we have on the structure of problems and search algorithms. Then, we will present a taxonomy of hybrid algorithms that tries to encompass all published work to date in the field and provide a unifying view of it.

2 Structure of $\mathcal{NP}$-hard problems

It is a fact that the structure of search spaces of $\mathcal{NP}$-hard problems is very poorly known. However, we think that in order to really be able to have a scientific approach of the design of search algorithms, such a knowledge is required. This knowledge would help the designer of an algorithm to optimize a certain instance of a certain problem to choose the most suited combination of algorithms to accomplish this task. It is clear that the study of the structure of search spaces is very difficult. Today, there are only a very few papers that address this issue and their actual utilization to guide the design of algorithms

has nearly never been done. We have been working for some time on this problem of investigating the structure of $\mathcal{NP}$-hard problems. In the same time, we have studied the behavior of iterated local search algorithms in detail and tried to relate to how they wander in a search space during their action. We thus try to fill the gap between these two issues. The problem is complex because there is a feedback between these two issues: the algorithm structures the problem in a certain and important way, and this structure implies a certain behavior of the algorithm.

The notion of fitness landscape is used to get some intuition about the structure of search spaces. This notion was imagined by Sewall Wright in the field of dynamics of population in biology [44]. It is also known as "energy landscape" in statistical mechanics. This notion aims at providing some insight into the problem structure and what an iterated local search algorithm "sees" when exploring it. It is made of plateaus, valleys, peaks, canyons, ... Formally, the definition is: let,

- $\mathcal{E}$ the set of all points of the search space,
- ω the operator that is used by the search algorithm,
- $G = (\mathcal{E}, E)$, the graph whose vertices are the points of the search space. An edge links points x and x' if x' can be obtained by applying ω on x,
- $\mathcal{L}$ is the geometrical object obtained by assigning the quality as an altitude to the vertices of G.

$\mathcal{L}$ is the landscape of the problem. When minimizing, the algorithm searches for points of very low altitude in this landscape.

We have recently investigated the landscape associated to the Traveling Salesman Problem. Using the 2-change operator, we [15] have found that all local optima are gathered in a structure called "massif central" by S. Kauffman [21,22], or "big valley" by Kirkpatrick and Toulouse [23]. The massif central contains only a very small amount of the whole search space: 1 point out of $O(n^{2n/3})$. This structure somehow dilutes when using the city-exchange operator which takes two cities in a tour and exchanges them [14]. This dilution of the massif central is the reason why, as it is exprimentally known, city-exchange provides local optima of lower quality than 2-change. The existence of this massif central has led us to design an efficient hybrid algorithm that first finds the massif central and then explores it in order to find better local optima [30]. The existence of such a topology of the landscape has been suggested for the flow-shop scheduling problem [45], and the graph k-coloring problem [18]. If this might be a general property of combinatorial problems having no constraint involved in it and using a suitable operator, the landscape of constrained problem such as the 0-1 Knapsack is not alike: in this problem, local optima are located on the border of a "crater" lying between

feasible and non feasible solutions [16].

In our mind, this knowledge of the structure of search spaces can be used, and should be used, as a guideline in the choice of a search algorithm (it should be clear to the reader that it is not a matter of choosing an algorithm for a given structure of space: the use of an algorithm implies a certain structure by itself: these two issues are strongly correlated). We now turn to a proposition of taxonomy of hybrid algorithms and we relate them to the structure of the search space in order to distinguish among them.

3 A Taxonomy of hybrid algorithms

Evolutionary Algorithms (EAs) more or less simulates a natural process [1]. As such, they possess a certain dynamics which is inherent to the process, regardless of details related to solution representation or the way solutions 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 [2] [31]. Starting with a random population, all its 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 1).

3.1 Sequential hybridization

A fundamental and 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 generations on figure 1). 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,

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

[2] 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.

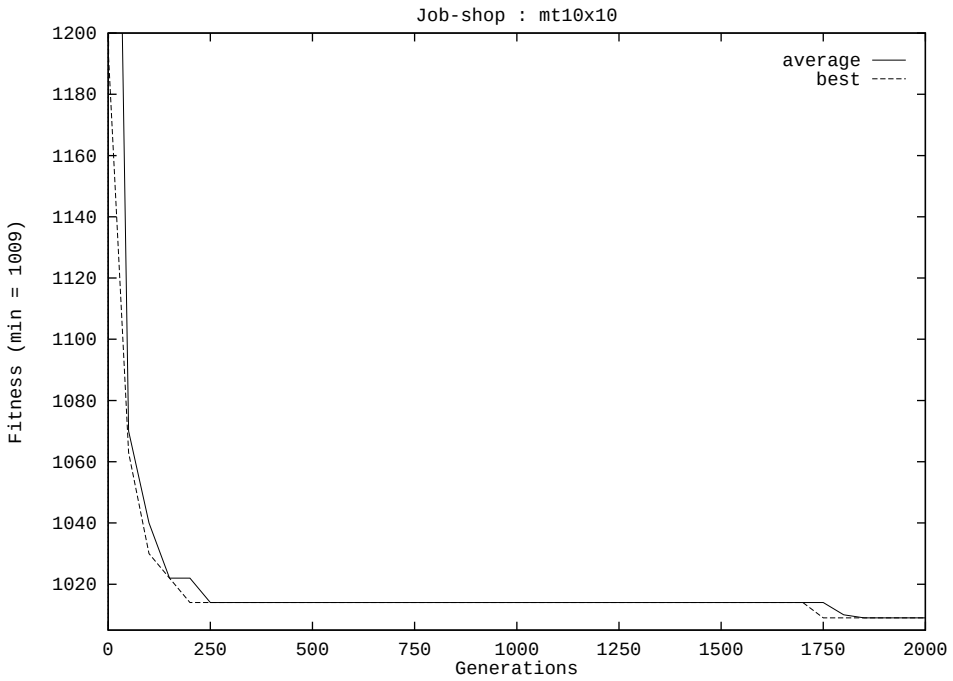

Fig. 1. Evolution during time of the mean fitness of the population of a genetic algorithm solving an instance of the Job-Shop Schduling Problem (JSP, on instance MT10 $\times$ 10). We have represented the mean fitness of the population (plain line) as well as the fitness of the best solution found so 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 Evolutionary algorithms.

(2) the exploitation of the already found basin of attraction has to be realized in order 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 [11] 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 tabu search (see figure 2). 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. The sequential hybridization may use a greedy algorithm to generate a good initial population for the EA (see figure 2).

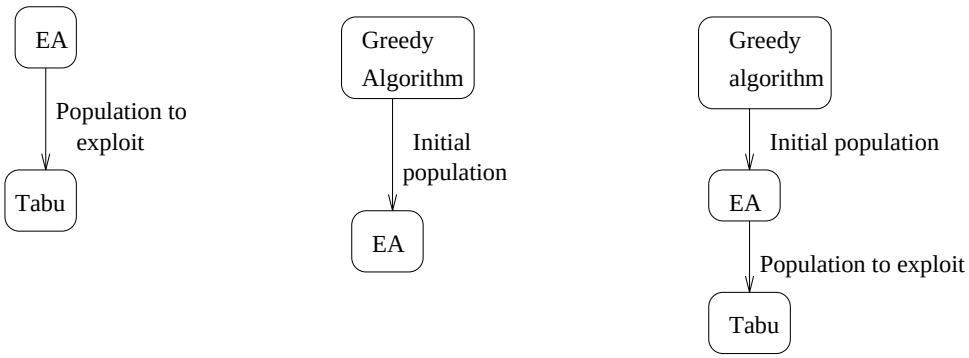

Fig. 2. Sequential Hybridization. Three instances of this hybridization scheme are represented. There may be more than three algorithms to be pipelined.

Many authors have used the idea of sequential hybridization. In [26], the authors introduce simulated annealing to improve the population obtained by an EA. In [25], the proposed algorithm starts from simulated annealing and uses EAs to enrich the solutions that have been found. In [10], two genetic algorithms are pipelined to solve a problem of routing of macro-cell layouts.

We have performed experiments on the graph partitioning problem using the tabu search algorithm exploiting the result found by a EA [41]. This greatly enhances the search performed either by the EA, or the tabu search alone (see table 1 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 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 1, 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.

Our recent work [30] has also exhibited an other strategy combining a multi-start hill-climber and a degenerate genetic algorithm (performing only recombination), the resulting population being optimized again, for a certain number of cycles, by this sequential hybrid. The idea behind it is that the multi-start hill-climber based on the 2-change operator reaches the border of the massif central. Then, recombination is used to work inside the massif central generating new points which are not 2-optimal, and approaching a bit more the center of the massif central. These solutions feed the multi-start hill-climber which provides new, and better 2-optimal solutions, and so on.

We now turn to hybridizations based on parallelism.

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 [39,34] 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 original individual by the enhanced one[3]. The search algorithm may be a simple hill-climber, a tabu search, or even some kind of simulated annealing algorithm [4]. We call this kind of hybridization a *parallel synchronous hybridization* (PSH) because the different algorithms are precisely synchronized.

PSH has led to many application of genetic algorithms to the effective optimization of $\mathcal{NP}$-hard problems. On the TSP, [43] uses the 2-opt and the Lk-opt heuristics to perform local optimization of tours. [12] discusses this approach on the quadratic assignment problem, [13] on the graph coloring problem. In [5], a PSH algorithm has been used for the multi-constraint knapsack problem and the set covering problem. A heuristic operator which utilizes problem-specific knowledge is incorporated into the genetic operators. We only mean to provide some examples here. The review is far from being thorough. We have obtained good results on the graph partitioning problem hybridizing a GA with a tabu search (see table 1 column 6).

However, this type of hybrids is subject to two important remarks. First, the algorithm generally meets severe problems of premature convergence, short-cutting to some extent the exploration phases. Second, more fundamentally, the utility of the evolutionary algorithm is clearly questioned. As illustration of

[3] This kind of operators is qualified *lamarckian*.

	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 1
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 [41] for details.

our purpose, on the JSP [9], a mere multi-start hill-climber based on the same operator as the mutation generally outperforms the hybrid: the multi-start hill-climber is faster, very easy to design, and provides solutions of higher quality. This observation clearly questions the efficiency of the recombination operator. To be useful at all, recombination should be carefully designed, rather than acting blindly on genotypes as was done in the early days of the application of GAs on combinatorial optimization problems. However, some parallel synchronous hybrids have given interesting results (*e.g.* see [43]).

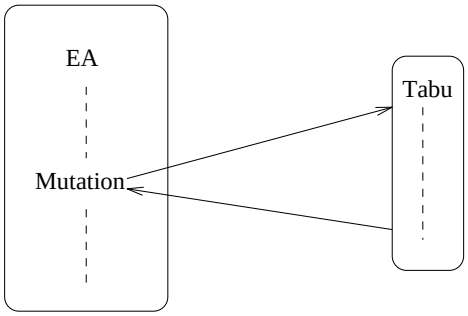

Fig. 3. Parallel Synchronous Hybridization. For instance, a tabu search is used as a mutation operator in an EA.

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 three kinds of cooperation:

global in which all the algorithms search the same research space. The goal is here to explore the space more thoroughly. A heterogeneous global PAH algorithm based on EA and tabu search has been proposed in [6] to solve a network design problem. The population of the EA is asynchronously updated by a parallel tabu search algorithm. The best solutions found by the multiple tabu search algorithms build an elite population. Such a strategy enables the different cooperating algorithms to explore the search space in their own way, each having its own point of view on the problem (each algorithm has its own landscape). The results of the different searches is then confronted.

partial in which 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 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. An immediate interest of this approach is to split a large problem into sub-problems.

functional in which the algorithms solve different problems. An example of such an approach has been developed in [1] to solve the quadratic assignment problem (QAP). A parallel tabu search is used to solve the QAP, while a genetic algorithm achieves a diversification task, which is formulated as an optimization problem. A frequency memory stores information relative to all solutions visited during the tabu search. The genetic algorithm refers to the frequency memory to generate solutions being in unexplored regions (see figure 4).

The so-called "island model" of parallel genetic algorithms [42] is a PAH and can be any of these three types. However, it has been mostly implemented as a global PAH.

Whether homogeneous or heterogeneous, global, partial or functional, 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:

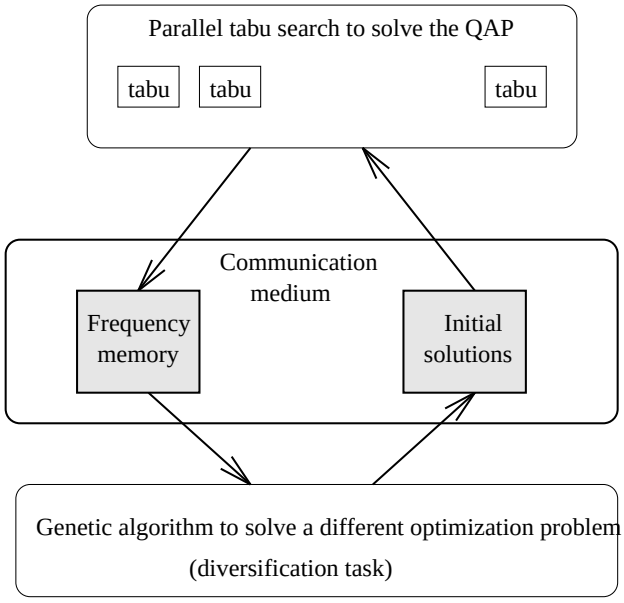

Fig. 4. Parallel Asynchronous Heterogeneous Functional Hybridization. Several search algorithms solve different problems.

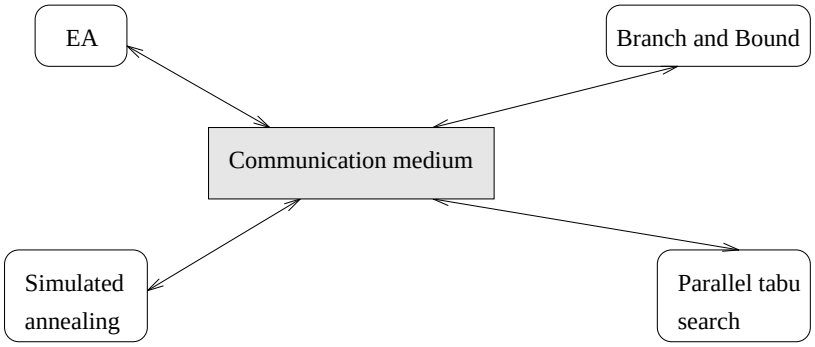

Fig. 5. Parallel Asynchronous Heterogeneous Hybridization. Several search algorithms cooperate, co-adapt, and co-evolve a solution.

- 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 building blocks that can be combined in different ways:

(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.

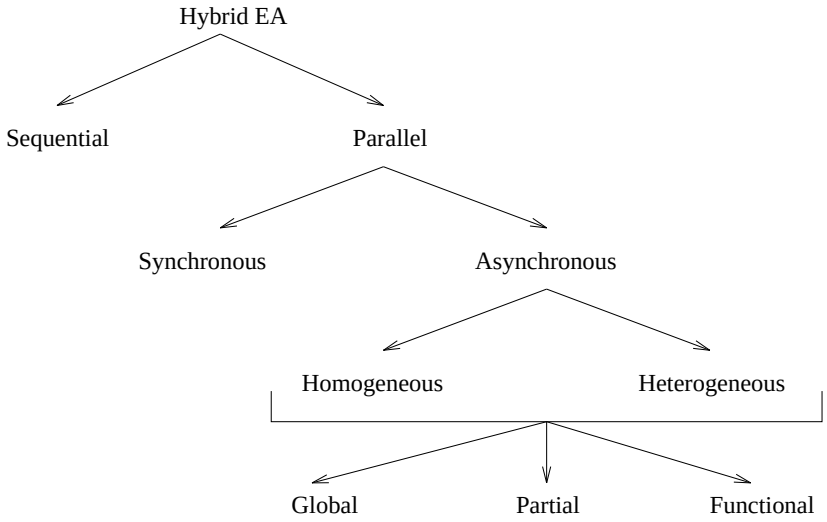

Fig. 6. A classification of hybrid EAs.

(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 [19]. The search algorithm is a GA. Each GA evolves individuals of a species 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) [24] which proves very efficient at solving big sized SPP in real world applications (airline crew scheduling).

The classification that is proposed also encompasses algorithms that are not related to EAs.

Let us consider Karp's partitioning heuristic for the Traveling Salesman Problem [20]. Once the problem has been partitioned, each sub-problem is solved independently of each others. Then, the tours that has been obtained are glued together to form a valid hamiltonian tour, solution to the problem. This is thus a S+PAH+S hybrid.

Ant colonies are also a homogeneous global PAH [7]. In this approach, a population of insects (generally "ants") are metaphorically asynchronously solving a TSP. Ants are traveling on the edges of the graph corresponding to the instance to solve. They communicate by leaving some substance on their path ("pheromone") that other ants detect and tend to follow at their turn.

3.2.4 Parallel hybrid implementations

Parallel hybridization schemes ideally provide novel ways to parallelize EAs by providing parallel models of the algorithms. 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 various types of parallel architectures:

- Massively parallel SIMD machine: Maspar MP-1 of 16k processors,
- Parallel MIMD machine: reconfigurable network of 128 transputers,
- Homogeneous processor farm: 16 DEC Alpha,
- Heterogeneous network of 120 workstations: PC/Linux, Sun4/Sunos, Sun/Solaris, ...

The easiness to use parallel and distributed architectures has been acknowledged for this hybrid model. We are experimenting with a functional heterogeneous PAH to solve the QAP [2]. A GA is cooperating with a parallel tabu search algorithm. The GA has been implemented on a massively parallel 16K processor Maspar MP-1 following a cellular synchronous model while the parallel tabu algorithm are running on a 16 Dec-alpha farm and a large network of heterogeneous workstations (fig.7).

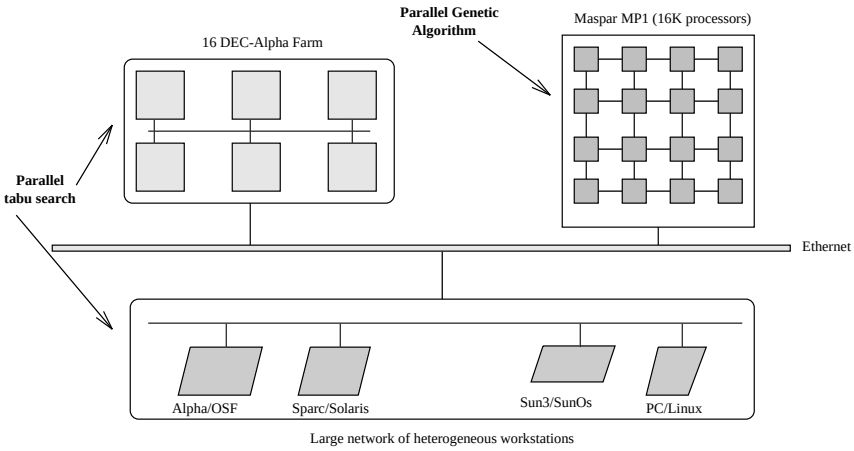

Fig. 7. Parallel implementation of functional heterogeneous PAH algorithms.

Parallel hybrid algorithms fall into three categories depending on whether the number and/or the location of work (tasks, data) depend or not on the load state of the target machine:

- **non-adaptive**: this category represents parallel algorithms in which both

the number of tasks of the application and the location of work (tasks or data) are generated at compile time (static scheduling). The allocation of processors to tasks (or data) remains unchanged during the execution of the application regardless of the current state of the parallel machine. Most of the proposed algorithms belong to this class.

An example of such an approach for parallel tabu search is presented in [29]. The neighborhood is partioned into equal size partitions depending on the number of workers, which is equal to the number of processors of the parallel machine. In [2], a parallel GA is proposed, where the number of tasks generated is equal to the population size which is fixed at compile time.

When there are noticeable load or power differences between processors, the search time of the non-adaptive approach presented is derived by the maximum execution time over all processors (presumably on the most highly loaded processor or the least powerful processor). A significant number of tasks are often idle waiting for other tasks to complete their work.

- **semi-adaptive**: to improve the performance of parallel non-adaptive algorithms, dynamic load balancing must be introduced [29][3]. This class represents algorithms for which the number of tasks is fixed at compile-time, but the location of work (tasks, data) is determined and/or changed at run-time. Load balancing requirements are met in [29] by a dynamic redistribution of work between processors. During the search, each time a task finishes its work, it proceeds to a work-demand. Dynamic load balancing through partition of the neighborhood is done by migrating data.

  However, the degree of parallelism in this class of algorithms is not related to load variation in the target machine: when the number of tasks exceeds the number of idle nodes, multiple tasks are assigned to the same node. Moreover, when there are more idle nodes than tasks, some of them will not be used.
- **adaptive**: *Parallel adaptive programs* are parallel computations with a dynamically changing set of tasks. Tasks may be created or killed as a function of the load state of the parallel machine. A task is created automatically when a node becomes idle. When a node becomes busy, the task is killed. As far as we know, the only work along this line is [40].

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. Landscape analysis, and hybridization schemes constitute the central issues of this work. Among other hybridization schemes, parallel asyn-

chronous 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.

References

[1] V. Bachelet, Z. Hafidi, Ph. Preux, and E-G. Talbi. Diversifying tabu search by genetic algorithms. Technical Report LIL-97-13, Laboratoire d'Informatique du Littoral, Calais, Laboratoire d'Informatique du Littoral, Calais, October 1997.

[2] V. Bachelet, P. Preux, and E. G. Talbi. Hybrid parallel heuristics for the quadratic assignment problem. In *Parallel Optimization Colloquium POC96*, Versailles, France, Mar 1996.

[3] P. Badeau, M. Gendreau, F. Guertin, J-Y. Potvin, and E. Taillard. A parallel tabu search heuristic for the vehicle routing problem with time windows. *RR CRT-95-84, Centre de Recherche sur les Transports, Université de Montréal*, Dec 1995.

[4] D. E. Brown, C. L. Huntley, and A. R. Spillane. A parallel genetic heuristic for the quadratic assignment problem. *Proc. of the Third Int. Conf. on Genetic Algorithms, San Mateo, USA*, pages 406–415, 1989.

[5] P. C. Chu. *A genetic algorithm approach for combinatorial optimization problems*. PhD thesis, University of London, 1997.

[6] T. G. Crainic, A. T. Nguyen, and M. Gendreau. Cooperative multi-thread parallel tabu search with evolutionary adaptive memory. *2nd Int. Conf. on Metaheuristics, Sophia Antipolis, France*, July 1997.

[7] M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on Evolutionary Computation*, 1(1), 1997.

[8] Kathryn A. Dowsland. Simulated annealing. In *[35]*, pages 20–69. 1995.

[9] D. Duvivier, Ph. Preux, and E-G. Talbi. Climbing-Up $\mathcal{NP}$-Hard Hills. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature (PPSN'96)*, pages 574–583. Springer-Verlag, Lecture Notes in Computer Science vol. 1141, September 1996.

[10] H. Esbensen. A macro-cell global router based on two genetic algorithms. In *Proc. of the European Design Automation Conference*, pages 428–433, 1994.

[11] Larry J. Eshelman. The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In *[32]*, pages 265–283, 1991.

[12] Charles C. Fleurent and Jacques A. Ferland. Genetic hybrids for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics*, 16:173–188, 1994.

[13] Charles C. Fleurent and Jacques A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operation Research*, 1995.

[14] C. Fonlupt, D. Robilliard, and Ph. Preux. A comparison of the 2-opt-move and the city-swap operators for the TSP. In *Proc. Evolution Artificielle*, Lecture Notes in Computer Science, Nimes, France, October 1997. Springer-Verlag.

[15] C. Fonlupt, D. Robilliard, Ph. Preux, and E-G. Talbi. Fitness landscape and performance of meta-heuristics. In *Proc. Meta-Heuristics'97 (MIC'97)*, Sophia-Antipolis, France, July 1997.

[16] A. Fréville and G. Plateau. Heuristics and reduction methods for multiple constraint 0-1 linear programming problems. *European Journal of Operation Research*, 24:206–215, 1986.

[17] F. Glover. Tabu search. In *[35]*, chapter 3, pages 70–150. 1995.

[18] A. Hertz, B. Jaumard, C.C. Ribeiro, and W.P. Formosinho. Local optima topology for the k-coloring problem. *Discrete Applied Mathematics*, 49:257–280, 1994.

[19] Philip Husbands, Frank Mill, and Stephen Warrington. Genetic algorithms, production plan optimisation and scheduling. In *[37]*, pages 80–84, 1990.

[20] R. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman in the plane. *Mathematics and Operations Research*, 2:209–224, 1977.

[21] Stuart A. Kauffman. Adaptation on rugged fitness landscapes. In *[38]*, pages 527–618. 1988.

[22] Stuart A. Kauffman. Principles of adaptation in complex systems. In *[38]*, pages 619–712. 1988.

[23] S. Kirkpatrick and G. Toulouse. Configuration space analysis of travelling salesman problems. *J. Physique*, 46:1277–1292, August 1985.

[24] 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.

[25] F. T. Lin, C. Y. Kao, and C. C. Hsu. Incorporating genetic algorithms into simulated annealing. *Proc. of the Fourth Int. Symp. on AI*, pages 290–297, 1991.

[26] S. W. Mahfoud and D. E. Goldberg. Parallel recombinative simulated annealing: A genetic algorithm. *Parallel computing*, 21:1–28, 1995.

[27] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, A Bradford Book, 1996.

[28] Ibrahim H. Osman and Gilbert Laporte. Metaheuristics: a bibliography. *Annals of Operational Research*, 63:513–628, 1996.

[29] C. S. Porto and C. Ribeiro. Parallel tabu search message-passing synchronous strategies for task scheduling under precedence constraints. *Journal of heuristics*, 1(2):207–223, 1996.

[30] Ph. Preux, D. Robilliard, and C. Fonlupt. Fitness landscapes of combinatorial problems and the performance of local search algorithms. Technical Report LIL-97-12, Laboratoire d'Informatique du Littoral, Calais, France, Laboratoire d'Informatique du Littoral, Calais, October 1997. (submitted).

[31] Philippe Preux. étude de l'uniformisation de la population des algorithmes génétiques. In J-M. Alliot, E. Lutton, E. Ronald, and M. Schoenauer, editors, *Actes de Évolution Artificielle'94*, pages 19–28, Toulouse, France, September 1994. Cépaduès.

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

[33] V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors. *Modern Heuristic Search Methods.* John Wiley and Sons, 1996.

[34] Colin Reeves. Genetic algorithms. In *[35]*, pages 151–196. 1995.

[35] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems.* Advanced Topics in Computer Science. Mc Graw-Hill, 1995.

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

[37] H-P. Schwefel and R. Männer, editors. *Proc. of the First Parallel Problem Solving in Nature*, Lecture Notes in Computer Science, vol 496. Springer-Verlag, Berlin, 1991.

[38] Daniel L. Stein, editor. *1988 Lectures in Complex Systems.* Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley Publishing Company, 1989. SFI Studies in the Science of Complexity, Lectures Vol. I, ISBN: 0-201-52015-4.

[39] E. G. Talbi. *Parallélisme et applications irrégulières*, chapter Algorithmes génétiques parallèles : Techniques et applications, pages 29–48. Hermes, France, 1995.

[40] E. G. Talbi, Z. Hafidi, and J-M. Geib. Parallel adaptive tabu search for large optimization problems. In *2nd Metaheuristics International Conference MIC'97*, Sophia-Antipolis, France, July 1997.

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

[42] Reiko Tanese. Distributed genetic algorithms. In *[36]*, pages 434–439, 1989.

[43] Nico L. J. Ulder, Emile H. L. Aarts, Hans-Jürgen Bandelt, Peter J. M. van Laarhoven, and Erwin Pesch. Genetic local search algorithms for the traveling salesman problem. In *[37]*, pages 109–116, 1990.

[44] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proc. 6th Congress on Genetics*, volume 1, pages 356–366, 1932.

[45] Takeshi Yamada and Colin Reeves. Permutation flowshop scheduling by genetic local search. In *2nd IEE/IEEE International Conference on genetic Algorithms In Engineering Systems: Innovations and Applications*, Glasgow, UK, 1997.