Climbing Up NP-Hard Hills

D. Duvivier* and Ph. Preux** and E-G. Talbi***

Abstract. Evolutionary algorithms are sophisticated hill-climbers. In this paper, we discuss the ability of this class of local search algorithms to provide useful and efficient heuristics to solve NP-hard problems. Our discussion is illustrated on experiments aiming at solving the job-shop-scheduling problem. We focus on the components of the EA, pointing out the importance of the objective function as well as the manner the operators are applied. Experiments clearly show the efficiency of local search methods in this context, the need of hybrid algorithms, as well as the very good performance obtained by simple hill-climbing algorithms. We also show that our results compare favorably with other published results.

1 Introduction

Our work aims at assessing what EAs are good for in the context of combinatorial optimization with regards to other search methods (both exact and non exact). We think that stochastic local search algorithms can bring something to this field. Exact methods are limited in the size of the problems they can handle because of their time/space requirements. We are perfectly aware of some limitations in the theoretical results to assess solutions found with local searches. However, we think that hybrid algorithms might provide some useful methods, able to handle large instances, and able to relax in some measure these theoretical limitations.

To achieve our goal, we concentrate ourselves on the resolution of the job-shop-scheduling problem (JSP), the traveling salesman problem (TSP), and the quadratic assignment problem (QAP), and we investigate how the evolutionary paradigm may be used (see [BPT96] for the QAP). Though sophisticated ones, EAs are hill-climbers: given a collection of points of the search space, the EA forms a new collection and keeps, deterministically or not, the best points among these two collections. The discriminating feature of EAs from a "standard" hill-climber is the use of recombination operators. Hence, comparing an EA with a standard hill-climber using only mutation-like (asexual) operators may shed some lights on the importance of recombination as well as its efficiency. However, care has to be taken when drawing conclusions on the efficiency of recombination because of the mixed effect of mutation and recombination.

In this paper, we will restrict ourselves to one problem, namely the job-shop-scheduling on which we have obtained some significant results. There are several

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

** Laboratoire d'Informatique du Littoral, 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

variants of the JSP (see [P94]). We consider here the simple JSP where J jobs, each composed of M operations are to be realized. There are also M machines. Each operation may be realized on a single machine. Each job has an operation which has to be performed on each machine. Then, the aim of the search is to find a planning of occupation of the machines that minimizes the total makespan, that is the total time of realization of the J jobs. This planning indicates, for each time slot, the operation it is processing. A planning is said to be active iff none of its operation may be started earlier resulting in a reduction of the overall makespan. It is known that the optimal planning is active [GT69]. A critical operation is one that, if started earlier, would reduce the makespan. Hence, critical operations are good target for local optimization of a planning. In section 2, we discuss the encoding of solutions. In section 3, we detail the algorithms we have used so far. Section 4 gives our current results compared with the best results known so far. To be able to compare our results with others, we used the ORLIB benchmarks [Orl], that is a set of about 150 instances of size ranging from simple instances to very difficult ones. Finally, we will discuss these results and give our perspectives.

2 Encoding schemes

In this section, we discuss the encodings of solutions that we have used in our algorithms. Basically, two classes of encodings may be considered for the JSP. They are qualified either as direct, or indirect.

2.1 Indirect encoding

When using an indirect encoding, the data structure is not a planning. Given a data structure, a decoder is required to express the planning. According to the information that is present in the data structure, the decoder has more or less work to do to be able to derive a planning. Constraints are handled by the decoder and guarantee the validity of the planning that is derived. We use a table of $J \times M$ entries. Each job is assigned a class of markers. All the markers associated to one job have the same tag (the job number in our example, below). The markers are then shuffled in the array. The decoder considers this array from the "left to the right". For each entry, it schedules as soon as possible the next not-yet scheduled operation of the job associated to the marker that has been found.

2.2 Direct encoding

In the case of a direct encoding, the data structure actually represents a planning. Hence, no decoder is required. We use a two-dimensional encoding, one dimension ranging among jobs, the other one among machines. For each machine, we have the list of operations that are performed with their time slot. For each job, we have the list of operations tagged with their time slot. This encoding makes it easier to detect available time slots.

3 Algorithms

We have mainly worked with three classes of primitive algorithms: "standard" hill-climber, tabu search and evolutionary algorithms. We have also investigated hybrid algorithms tinkered with these three algorithms as building-blocks. Let us now turn to a detailed presentation of these algorithms.

3.1 Multi-start Hill-climber

The hill-climber is considered as a very simple heuristic that does not require heavy computational resources. A hill-climber is very easy to design. However, it may easily be trapped in a local optimum. We also use a variant of the basic hill-climber, namely a multi-start hill-climber. Starting at a random point, the hill-climber tries to move uphill. When, after a certain amount of attempts, it always fails to climb, the current best point is recorded and the climbing is restarted from another point in the space, chosen at random. To be more precise, it performs the algorithm displayed in figure 1.

```
current_schedule := random_schedule
best_ever_found_schedule := current_schedule
best_schedule := current_schedule
for i := 1 to number_of_periods do
  for j := 1 to threshold do
    candidate_schedule := perform move from current_schedule
    if candidate_schedule is better than current_schedule then
      current_schedule := candidate_schedule
    fi
  done
  if current_schedule is better or equal to the best_schedule then
    best_schedule := current_schedule
    if current_schedule is better than the best_ever_found_schedule then
      best_ever_found_schedule := current_schedule
    fi
  else /* Trigger a new experiment */
    best_schedule := current_schedule := random_schedule
  fi
done
output ("I have found " best_ever_found_schedule)
```

Fig. 1. The multi-start hill-climber algorithm that we use.

The delicate part in using a hill-climber lies in the choice of the move operator. Using an indirect encoding, the basic move turns out to be a swap of markers. Accordingly, using a direct encoding, the basic move is an exchange of two operations. Using one encoding or the other, the solutions obtained are equivalent.

3.2 TABU search

The TABU search is a hill-climber which is able to escape from local optima [Glo89a, Glo89b]. The TABU search has only been used with the direct encoding. The basic move swaps two operations of two different jobs. We use an aspiration criterion which accepts a movement even though it is taboo if it leads to the best ever found planning. The size of the taboo list varies dynamically during the run according to [HW95].

3.3 Evolutionary algorithm

As the TABU, the EA only works with the direct encoding. The operators, the objective function, the basic scheme of the algorithm have been tailored for the resolution of the JSP and are described below.

Encoding The encoding of solutions should fit with the problem, as well as the operators that act on them. In this regard, an encoding providing as much information as possible, or at least as much as useful, is worthy. An indirect encoding clearly lacks a lot of information. Thus, we use the direct encoding in the EA. Using this encoding, we can detect the critical operations of a planning.

Recombination operator We use a recombination operator largely inspired by the GA/GT crossover introduced in [NY92]. This operator relies on Giffler and Thompson's algorithm [GT69] to produce an offspring given two parents. The principle of the GT algorithm is:

- $C \leftarrow$ first operation (which is not yet scheduled) of each job
- compute ECT of operations $\in C$
- repeat
  1. select $O^* \in C$ which has minimum ECT
  2. $G \leftarrow$ operations $\in C$ sharing the same machine with O^* and conflicting with O^* (processing overlaps)
  3. choose an operation $O^{**} \in G$ at random
  4. schedule O^{**} as soon as possible according to ECTs
  5. update C and ECTs
- until all operations are scheduled

where C is a "cut", G is a set of operations conflicting with O^*, and ECT means Earliest Completion Time.

A property of this GA/GT operator is that it always produces active plannings.

The mutation operator The mutation performs a swap of two operations. To lead to a valid planning, the swap should only consider operations on one machine chosen at random and re-schedule all the operations that are not yet scheduled. This mutation operator always produces semi-active plannings which, in general, are not active.

The objective function For the simple JSP, aiming at optimizing the makespan, the usual idea is to design the objective function as returning the makespan of a planning/solution. However, close observations of the behavior of the EA in this case (it is still more salient in the case of the hybrids described below) led us to work further the objective function. Initially, our criterion to stop the EA was a null standard deviation of the performance of the members of the population, that is all individuals have the same makespan. However, this does not involve that all individuals are the same planning. As long as there is (genotypic) diversity in the population, the population can evolve. Thus the halting criterion is not the good one. However, it is not easy to assess the equivalence of two plannings. Thus, it is difficult to assess the diversity of the population when all individuals have the same makespan. The number of critical operations may be used as a clue of the goodness of an individual: the less critical operations it has, the better it is. Hence, to distinguish two individuals, we used the sufficient (but not necessary) condition: if two plannings have a different number of critical operations, they are different. Hence, we use an objective function which combines the makespan and the number of critical operations. Using this new objective function, the results have been improved substantially.

Basic scheme of the EA We have observed that, because we are typically using a high rate of mutation and a sophisticated recombination operator, the benefit of the application of the crossover is often destroyed by the mutation applied afterwards. Hence, we use a different scheme of application of the operators. More precisely, the EA performs as follows:

- initialize the population of plannings at random and make them all active
- while completion criterion is not fulfilled do
  - reproduce
  - apply operators on offsprings
  - operated offsprings form the next population

The operators are applied as follows:

1. pick up two individuals χ_1 and χ_2 in the population
2. perform cross-over on them and stochastically keep the best of the two resulting individuals: χ_1'
3. perform mutation on one individual among χ_1 and χ_2 choosing one or the other at random. This results in χ_2'
4. put χ_1' and χ_2' in the population of offsprings

This non-conventional scheme of application of operators has proven to perform better.

3.4 Hybrid algorithms

Various hybridization schemes are possible (see [PT95]). We have currently experimented one scheme on the JSP, namely the synchronous hybridization which uses a

search algorithm as a mutation operator. Only direct encodings have been used in the case of hybrid algorithms (all algorithms are then working on the same representation).

We have used a simple hill-climber (HC) as the search algorithm in the hybrid denoted EA+HC below. It iterates a certain amount of moves, starting from a planning which is the individual it is applied on. The amount of moves that are allowed while there is no improvement is set to 200. When this threshold is reached, the algorithm continues to move as long as it climbs up.

We use populations of 300 individuals, stochastic universal sampling reproduction strategy, generational elitist selection. GA/GT is applied with probability 0.6, swap mutation is applied to all the individuals, and the hill-climber to 5 % of the population.

4 Results

4.1 Performance of our algorithms

We now turn to the results we have obtained with previously reviewed algorithms (*cf.* table 1). In our benchmark suite, we use the Muth and Thompson's instances, the 8 Carlier's instances, as well as 6 Lawrence's instances, and 5 Taillard's instances of big size. The size of the instances ranges from 6×6 to 100×20.

Generally speaking, the hybrid EA+HC always performs better than the EA alone. The hybrid has found the optimal planning for some instances (MT6 $\times$ 6, MT10 $\times$ 10, car1, car2, car3, car4, car6, car7, la01, la02, la03, la04, and la05) while the EA did only find the optima of the MT6 $\times$ 6, la01, and la05 which are found by all the algorithms.

For their parts, the hybrid EA, MSHC, and Tabu all achieve the same kind of results, lying within a few percents from the optimal planning. When considering the distribution of points found in a series of experiments with a given algorithm, it clearly appears that the standard deviation of optimal plannings is very different from one algorithm to another, and from one series of instances to another. The MSHC generally has a low to moderate standard deviation, the EA and the hybrid EA+HC a moderate one, the TABU heuristic quite a high one. Furthermore, for all heuristics, the standard deviation obtained on the Carlier's instances is much higher than the standard deviation obtained for Muth and Thompson's and Lawrence's instances. Hence, on the test-suite, the MSHC always finds the same quality of makespans, while the TABU finds various quality of plannings.

Concerning the mean time of execution of the various algorithms, the very short time of execution of the MSHC is noteworthy. The EAs (both pure and hybrid) have moderate times of execution. For the TABU, no such conclusion might be easily drawn with regards to the size of the instance for the MT's. It should be pointed out that the EA+HC was run during 2000 generations. However, the best planning that the EA+HC is able to find in a run is usually synthesized during the first hundreds generation. Hence, that means that the EA+HC can be stopped after a few hundreds of generations without losing, most of the times, the best solution that it would have been able to find in the run. It also means that we can perform 10, or 20 runs in the time of 1, and have much more odds to obtain better solutions while using the

Instance	Size	optimum	EA	EA+HC	MSHC	TABU
mt6x6	6x6	55	55 (0)	55 (0)	55 (0)	55 (0)
mt10x10	10x10	930	953 (2)	930 (0)	947 (2)	930 (0)
mt20x5	20x5	1165	1180 (1)	1178 (1)	1175 (<1)	1165 (0)
car1	11x5	7038	7101 (<1)	7038 (0)	7038 (0)	7038 (0)
car2	13x4	7166	7576 (6)	7166 (0)	7166 (0)	7166 (0)
car3	12x5	7312	7594 (4)	7312 (0)	7312 (0)	7399 (1)
car4	14x4	8003	8423 (5)	8003 (0)	8003 (0)	8003 (0)
car5	10x6	7702	8047 (4)	7720 (≈0)	7738 (≈0)	7720 (≈0)
car6	8x9	8313	8699 (5)	8313 (0)	8313 (0)	8313 (0)
car7	7x7	6558	6680	6558 (0)	6558 (0)	6558 (0)
car8	8x8	8264	8407 (2)	8264 (0)	8294 (≈0)	8294 (≈0)
la01	10x5	666	666 (0)	666 (0)	666 (0)	666 (0)
la02	10x5	655	711	655 (0)	655 (0)	655 (0)
la03	10x5	597	624	597 (0)	597 (0)	597 (0)
la04	10x5	590	620	590 (0)	590 (0)	590 (0)
la05	10x5	593	593 (0)	593 (0)	593 (0)	593 (0)
abz7	20x15	(655, 665)	757	685	682	682
la36	15x15	1268	1408 (11)	1292 (<2)	1281 (1)	1291 (<2)
ta21	20x20	(1539, 1658)	1968	1721	1725	1712
ta31	30x15	(1764, 1766)	2098	1907	1807	1774
ta41	30x20	(1859, 2026)	2551	2262	2143	2117
ta51	50x15	2760	3225 (15)	2909 (5)	2760 (0)	2776 (<1)
ta71	100x20	5464	6065 (10)	5723 (<5)	5744 (5)	5609 (<3)

Table 1. This table sums up our results. The instances are identified with their name in the Orlib.The second column gives the size of the instance. The third one gives the value of the optimum if it is known, a range of it when it is unknown. The fourth column gives the best result obtained using the evolutionary algorithm. The fifth column gives the result obtained with the hybrid EA+hill-climber (EA+HC), the sixth with the multi-start hill-climber (MSHC), the seventh with the TABU. In (), we have indicated the percent from the optimum the best found planning lies in. All algorithms have been run from 10 to 20 times.

same amount of computational resources. However, the remark should be tempered by the fact that, as noted earlier, our objective function has changed, resulting in a change in the halting criterion. Thus, the genotypic diversity of the population is now better taken into account to stop the algorithm when the "natural" evolution of the population has actually ended.

It is noteworthy that for big instances, the research space is huge and only a small fraction of it is visited by the algorithm.

It is also noteworthy that using EAs with a population 2, or 4 times bigger does not improve the quality of best found plannings. The rapid loss of diversity probably accounts for this fact.

We would like to put the emphasis on the performance achieved by the (multi-start) hill-climber. The best found plannings are either the optimal planning or very close to it. This prompts us with two remarks. First, the design of the hill-climber

is very simple while the design of the EA is not so easy (notably because of the recombination operator). Second, the basic movement of the MSHC is similar to the mutation that was used in the EA. Hence, this questions the efficiency of the GA/GT operator.

4.2 Comparison with other works

In table 2, we have summarized the results obtained by several authors on the Muth and Thompson's instances, mainly using evolutionary algorithms. We have only taken into consideration works with pure EAs with random initial population. (However, in some of these works, the initial plannings are generated at random, and made active before going any further.)

Reference	Algo-rithm	MT 6×6	MT 10×10	MT 20×5
Carlier and Pinson [CP89]	B&B	55	930	1165
Nakano and Yamada [NY91]	EA	**55**	965	1215
Nakano and Yamada [NY92]	EA	55	**930**	1184
Dorndorf and Pesh [DP92]	–	55	938	1178
Fang, Ross and Corne [FRC93]	EA	–	949	1189
Juels and Wattenberg [JW94]	EA	–	937	1174
Soares [Soa94]	EA	58	997	–
Kobayashi *et al* [KOY95]	EA	–	930	–
Our results (best obtained results)	EA	55	937	1178
Optimum		**55**	**930**	**1165**

Table 2. This table displays the results obtained by different authors on the Muth and Thompson instances. The first column gives the references to the work. The second column indicates the kind of algorithm that have been used. The three subsequent columns give the results that are reported (best found makespan) for the 3 instances. [NY91] uses an indirect encoding. [NY92] uses a direct encoding with the GA/GT operator. [JW94] uses the indirect encoding. They use an other recombination operator.

The Branch-and-Bound (B&B) is an exact method. Carlier and Pinson [CP89] have obtained the optimal solutions using this method.

Using an EA with an indirect representation, Nakano et Yamada [NY91] have obtained the optimal solution to the MT6×6. Nakano et Yamada [NY92] have obtained the optimal solution to MT10×10. They had turned to a direct representation and the GA/GT as the recombination operator. The optimal planning of the MT20×5 has not been found using the same algorithm. It should be noted that the optimal planning of the MT10×10 has only been found in 4 experiments out of 600. [KOY95], using a new operator (a crossover coupled with a hill-climber), have obtained the optimum of the MT10×10 51 times out of 100 experiments, that is much more often than Nakano and Yamada. It should also be emphasized that among the reported results, only Kobayashi *et al* [KOY95] work solely with active plannings.

Using an EA, only Nakano and Yamada [NY92] and [KOY95] have obtained the optimum of the MT10 $\times$ 10. No one has obtained the optimum of the MT20 $\times$ 5.

5 Discussion and perspectives

In this paper, we have compared different stochastic meta-heuristics on a test-suite composed of job-shop-scheduling instances, ranging from small size to big size. The meta-heuristics are hill-climbers, TABU search and evolutionary algorithms. Best plannings that were found, standard deviation, times to obtain this planning are discussed for 4 algorithms. The hybrid algorithm achieves better results than the pure EA. Furthermore, a simple multi-start hill-climber often obtains very good solutions. For this kind of algorithms, the trade-off between its design and its efficiency is then very good.

Given the results obtained with the pure EA and the hill-climber, we are prompted to question the efficiency of the recombination operator as long as this is the real difference between these two algorithms. This observation is radically different from the one we obtained on the set partitioning problem where the EA worked much better (both with regards to the solution that was obtained and the time to obtain it) than several kinds of hill-climbers (see [Tal93]). Hence, further studies on the role of the recombination are required.

As it was observed in our experiments, different classes of problem hardness seem to exist. Very pragmatically, we would like to say that an instance is easy if it can be solved optimally with a simple algorithm. This way, we optimize the trade-off between the time of design of the algorithm and the quality of solutions it is able to find. Furthermore, the multi-start hill-climber has also proved to be very fast to find optimal solutions on a series of instances.

In the forthcoming future, we will continue our study of hybridization. We will shortly have a hybrid EA+TABU. Further, we wish to experiment with the hybridization scheme discussed in [PT95] relying on the cooperation of search methods acting concurrently.

References

[BPT96] Vincent Bachelet, Philippe Preux and El-Ghazali Talbi. *Proc. of the Parallel Optimization Colloquium*, Versailles, France, March 1996.

[CP89] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, **35**(2):164–176, 1989.

[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.

[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 Stephanie Forrest, editor. *Proc. of the Fifth International Conference on Genetic Algorithms*, Urbana-Champaign, IL, USA, July 1993. Morgan Kaufmann, San Mateo, CA, USA, pages 488-493.

[Glo89a] F. Glover. Tabu search — part I. *ORSA Journal of Computing*, **1**(3):190–206, 1989.

[Glo89b] F. Glover. Tabu search — part II. *ORSA Journal of Computing*, **2**(1):4–31, 1989.

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

[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.

[HW95] Alain Hertz and Marino Widmer. *La méthode TABOU appliquée aux problèmes d'ordonnancement.* Automatique, Productique, Informatique Industrielle, **29**(4–5), pages 353–378, 1995.

[KOY95] Shigenobu Kobayashi, Isao Ono, and Masayuki Yamamura. An efficient genetic algorithm for job shop scheduling problems. In Larry J. Eshelman, editor. *Proc. of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, USA, July 1995. Morgan Kaufmann, San Mateo, CA, USA, pages 506–511.

[NY91] Ryohei Nakano and Takeshi Yamada. Conventional genetic algorithm for job shop problems. In Richard K. Belew and Lashon B. Booker, editors. *Proc. of the Fourth International Conference on Genetic Algorithms*, La Jolla, CA, USA, July 1991. Morgan Kaufmann, San Mateo, CA, USA, pages 474–479.

[NY92] Ryohei Nakano and Takeshi Yamada. A genetic algorithm applicable to large-scale job-shop problems. In R. Männer and B. Manderick, editors. *Proc. of the Second Conf. on Parallel Problem Solving in Nature*. Elsevier Science Publishers, Amsterdam, 1992, pages 281–290.

[Orl] Orlib. available via anonymous ftp on `mscmga.ms.ic.ac.uk:/pub`.

[P94] Bernard Penz. *Constructions agrégatives d'ordonnancements pour des job-shops statiques, dynamiques et réactifs.* PhD dissertation, Université de Grenoble I, 1994.

[PT95] Ph. Preux and E-G. Talbi. Assessing the evolutionary algorithm paradigm to solve hard problems. In *Constraint Processing, workshop on really hard problem solving*, September 1995. An extended version is available as report LIL-95-4 from the Laboratoire d'Informatique du Littoral (`ftp://ftp-lil/pub/preux/papers/lil-95-4.ps.gz`)

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

[Tal93] E. G. Talbi. *Allocation de processus sur les architectures parallèles à mémoire distribuée.* PhD thesis, Institut National Polytechnique de Grenoble, Mai 1993.

This article was processed using the LaTeX macro package with LLNCS style