The Fitness Function And Its Impact On Local Search Methods

D. Duvivier, Ph. Preux,
C. Fonlupt and D. Robilliard
Université du Littoral,
Laboratoire d'Informatique du Littoral,
BP 719, 62228
Calais Cedex, France,
duvivier@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— **The fitness function is generally defined rather straightforwardly in evolutionary algorithms (EA): it is simply the value of the function to optimize. In this paper, we argue and show that embedding more information in the fitness function leads to a significant improvement of the quality of the local optima that are reached. The technique is developed here on $\mathcal{NP}$-hard problems and demonstrated on the Job-Shop-Scheduling problem. The technique is first used in a mere steepest descent hill-climber in order to assess its usefulness. Then, it is shown that its use in an EA also improves its performance in terms of the quality of solutions that are found.**

Keywords— **combinatorial optimization, evolutionary algorithms, objective function, job-shop scheduling problem**

I. INTRODUCTION

Hill-climbers are general local iterative search methods which mainly based their decision on the fitness of the current point and the fitness of the neighboring points. While the choice of the neighbor to visit next is not a trivial issue, it gets still worse when all neighbors are equally fitting, that is when the hill-climber has reached a "plateau", or when it has found a local optimum. This paper proposes a way to overcome this problem. We illustrate the technique on $\mathcal{NP}$-hard problems.

First, we need to define exactly what we call a "plateau" :

Definition 1 (plateau) Given σ a point in the search space, and v a value taken in the range of values of the criterion C. Given $\mathcal{S}$, the set of points σ' in the neighborhood of the current solution σ.

Considering $\mathcal{X}$ a subset of $\mathcal{S}$ defined by $\sigma'' \in \mathcal{X}$ iff $C(\sigma'') = v$, $\mathcal{X}$ is a plateau iff it contains at least 2 elements (i.e. $\mid \mathcal{X} \mid > 1$).

When solving a combinatorial optimization problem, we typically want to optimize one quantity, the main objective of the search (length of a tour in the Traveling Salesman Problem (TSP), time to complete a certain set of jobs in the Job-shop Scheduling Problem (JSP) for example). Most often, when these problems are tackled with hill-climbers (either very simple hill-climbers or more sophisticated ones like Tabu search or evolutionary algorithms), the objective function that is used to decide which point to visit next is only related to the quantity to optimize. For problems that have been extensively studied, we are aware of some good on-side properties that accompany a good tour for the TSP or a good schedule for the JSP. Though they might appear to be redundant with regards to the main quantity that is being optimized, these on-side properties might prove useful when the main objective is not sufficient to discriminate among neighbors of the current point. When a hill-climber is on a plateau, these subsidiary properties might help to choose the next point to visit. This technique can also be used to escape from a local optimum: when the main quantity can not be improved any further, guiding temporarily the search using an other quantity can be fruitful. Obviously, this quantity has to be relevant and has to be "more accurate" with regards to the other quantity, that is, it has to provide new information.

In the case of the TSP, we know that the shortest path is likely to be composed of edges among the shortests. Thus, when confronted with a set of neighbors, it can be fruitful to choose one in which the average length of edges is shorter than the other ones. Using this subsidiary property will typically provide better tours at the end of the search. We have applied this basic idea to an other $\mathcal{NP}$-hard problem [6]: the Job-Shop Scheduling Problem.

In this paper, we discuss various "on-side" properties that can be used to improve the choice of the neighbor to go to. As was suggested in [4], [5], [3], we show that taking this information into account significantly improves the quality of the points that are found. To enable a better understanding of what is going on in the algorithm, we develop our idea with a steepest descent hill-climber using a mutation-like operator. In the last section of this paper, we show that the approach is also successful when embedded in an evolutionary algorithm.

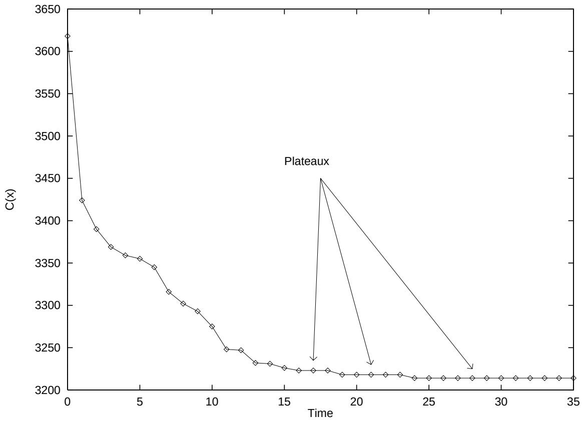

Fig. 1. The evolution of $C(x)$ during a typical run of a steepest descent algorithm along the iterations. x can be regarded as the current point during a steepest descent hill-climbing or a Tabu search, or as the best point of the population of an evolutionary algorithm.

II. OBJECTIVE FUNCTION

A. Basics

In this paper, we consider minimization problems defined as:

given a quantity $C(x)$, find a solution $X_\circ$ in the set $\mathcal{D}$ such that $X_\circ = \min_{x\in\mathcal{D}}\{C(x)\}$.

Of course, the results presented in this paper can be straightforwardly applied to a maximization problem.

Using an iterative search algorithm to solve a problem of this kind, the usual idea is to design the objective function $\mathcal{F}$ as returning the value of the quantity to optimize for the point under consideration x:

$$\mathcal{F}(x) = C(x)$$

We have plotted the evolution of $C(x)$ during a typical run of a steepest descent algorithm minimizing $C(x)$ (see Fig. 1). The plot exhibits several plateaux. On each plateau, the algorithm has difficulties to be guided in the neighborhood of the current solution. In this situation, using other information to guide the search can be useful. This extra information (secondary criterion) serves to discriminate points that are seemingly identical (identical from the point of view of the main criterion $C(x)$) though being truly different.

We have applied this method to solve different $\mathcal{NP}$-hard problems such as the JSP, and the TSP.

In order to simplify the study of this technique and to be able to assess its effect without multiplying the parameters, we have chosen a very simple algorithm: the steepest descent hill-climber (SDHC for short). Once the technique is validated on the SDHC, we show that it can be used in a non-deterministic hill-climber (NDHC for short) and in an EA and improves the quality of the points that are reached.

In the SDHC, there are two main parameters: the operator and the selection function. The operator swaps two operations and reschedule operations that needs to be rescheduled. Using this "simple" operator, we can study the selection function itself. The objective function is then the only information that guides the heuristic in the search space. One of the main goals of embedding several (secondary) criteria is to improve the objective function to perform a better choice among the neighbors of the current solution.

B. The discriminating power of a criterion

A (secondary) criterion C' is useful if it can discriminate points that have the same value for the main criterion C. This "discriminating power" is thus an important characteristic of a criterion and a proper definition is required.

We introduce a value denoted $\varphi_\sharp(C')$ that embeds the "discriminating power" of a certain criterion C':

Definition 2 (discriminating power)
We introduce $\phi_\sharp(C')$, as the number of different values that are taken by the criterion C' on a plateau of the main criterion C.
Then, we define $\varphi_\sharp(C')$ as the average number of $\phi_\sharp(C')$ on a set of solutions.
This latter value is called the discriminating power of C'.

A criterion C' such that $\varphi_\sharp(C') \approx 1$ is not a good criterion since – on average – there is only one value of C' on each plateau of C: this means that C' can not be used to guide the heuristic on the plateaus of C.

III. APPLICATION TO THE JOB-SHOP-SCHEDULING PROBLEM

Let us now turn to the use of this technique on an example, namely the Job-shop Scheduling Problem. There are several variants of the JSP (see for example [2], [13], [15], [8]). We consider here the simple JSP where J jobs, each composed of M operations are to be realized on M machines. Each operation must be realized on a single machine. Each job has an operation which has to be performed on each machine. A schedule indicates at each time slot and on each machine, the operation being currently processed.

The makespan of a schedule is the total time of realization of the J jobs, it is defined as follows:

Definition 3 (makespan) Let us denote $E_m(x)$ the completion time of the last operation performed on machine m according to the schedule x. Then we define the makespan $C_{\max}(x)$ of the schedule:

$$C_{\max}(x) = \max_{1 \leq m \leq M} E_m(x)$$

The objective of the JSP considered here is to minimize the makespan.

Let us define two terms that are in common use in the JSP community:

Definition 4 (busy-schedule) A schedule x such that $C_{\max}(x) = \text{makespan}(x)$ is called a busy-schedule if, for each time slot t where $0 \leq t \leq C_{\max}(x)$ there is at least one operation running on a machine.

Definition 5 (active schedule) A schedule is said to be active iff none of its operations may be started earlier without modifying the order of execution of the operations on the machines.

It is known that the optimal schedule is active [7].

Definition 6 (critical operation) Given a schedule x, a critical operation is one that, if delayed, would increase the makespan of x (if reordering the operations is not allowed).

Hence, critical operations are good targets for local optimization of a schedule. The number of critical operations found in a schedule x is denoted $C_{\text{op}}(x)$.

We also define the function H_2 of a schedule x after [9]:

$$H_2 = \sum_{m=1}^{M} E_m^2(x)$$

where $E_m(x)$ is the completion time of the last operation performed on machine m according to the schedule x.

A. Basic objective function

For the simple JSP, aiming at minimizing the makespan, the usual idea is to design the objective function as returning the makespan of a schedule (*e.g.* $C(x) = C_{\max}(x)$):

$$\mathcal{F}_1(x) = C_{\max}(x)$$

B. Discriminating power of different criteria for the JSP

Several criteria may be used as a clue of the goodness of an individual. A lot of work has still to be done, but we can discuss here criteria that have shown usefulness when incorporated into the objective function.

We apply a SDHC on five instances of the simple JSP so as to compute the value of $\varphi_\sharp$ of two criteria along the walks (*cf.* Table I). The criteria that are used are H_2 as well as C_{op}.

TABLE I

This table gives the mean value and the standard deviation of $\phi_\sharp$ for two criteria as measured along walks of the SDHC using $\mathcal{F}_1$. For each instance, 500 experiments were performed starting from random initial solutions.

Instance	$\phi_\sharp(H_2)$		$\phi_\sharp(C_{\text{op}})$	
	Aver ($\varphi_\sharp$)	Std-dev	Aver ($\varphi_\sharp$)	Std-dev
mt10x10	3.92	0.56	2.60	0.34
mt20x5	6.39	1.16	4.69	0.88
ta01	6.85	1.31	3.89	0.75
abz7	27.69	5.93	20.53	4.98
ta31	33.19	6.67	23.03	5.29

The value of $\varphi_\sharp$ is greater than one for the two studied criteria. This means that these criteria can discriminate points on the plateaux of the makespan. So they can be useful in the objective function. Furthermore, $\varphi_\sharp(H_2)$ is always greater than $\varphi_\sharp(C_{\text{op}})$, indicating a higher discriminating power for H_2 than for $\mathcal{C}_{\text{op}}$.

C. Correlation between criteria

In order to determine which criterion might be interesting to incorporate into the objective function, we have computed the correlation between each criterion and the makespan. Thus, we know which criterion is relevant with the goodness of a point. The correlation is computed over a set of points drawn at random and over the set of local optima reached at the end of the walks. The SDHC is started from each of the points drawn at random and uses the objective function ($\mathcal{F}_1$) which is only based on the makespan. This gives us at most 500 local optima.

The figures are displayed in Table II. The correlation between $C_{\max}$ and H_2 is very high, while the correlation between $C_{\max}$ and C_{op} is rather loose for points drawn at random (inexistent when considering local optima only). So, according to these values, H_2 appears to be a good criterion to incorporate in our objective function while C_{op} might provide some indication.

D. Different objective functions

In the previous section, we have introduced several criteria that seem to be interesting to incorporate in the objective function. Let us now define several functions so as to evaluate the usefulness of these criteria while solving the JSP. In the following functions, we

TABLE II

THIS TABLE GIVES THE AVERAGE VALUE OF CORRELATIONS BETWEEN DIFFERENT CRITERIA ON THE FIVE INSTANCES. FOR EACH INSTANCE 500 POINTS ARE SAMPLED.

Correlation	Cmax/H2	Cmax/Cop	H2/Cop
Random points	0.862	0.364	0.304
Local optima	0.866	0.094	0.052

define K_i (where $1 \leq i \leq 3$) so that the makespan remains the most important criterion (main criterion): a variation of the makespan is always greater than the sum of the variations of the other criteria (secondary criteria) for all possible busy-schedules of a given instance of the JSP.

● First of all, we can incorporate the H_2 heuristic in the objective function. We call this function $\mathcal{F}_2(x)$:

$$\mathcal{F}_2(x) = K_1 \times C_{\max}(x) + H_2(x)$$

K_1 is an upper bound for $H_2(x)$ over all the possible busy-schedules of the current instance of JSP.

● The number of critical operations may be used as a clue of the goodness of an individual: it is commonly held that the less critical operations a schedule contains, the better the schedule is. So we define an objective function based on this criterion. We call this function $\mathcal{F}_5(x)$:

$$\mathcal{F}_5(x) = K_2 \times C_{\max}(x) + C_{\mathrm{op}}(x)$$

K_2 is an upper bound for $C_{\mathrm{op}}(x)$ over all the possible schedules of the current instance of JSP.

● To assess whether the added information is relevant and does help the algorithm, we introduce a function where the additional criterion is replaced by a random noise. We call this function $\mathcal{F}_4(x)$:

$$\mathcal{F}_4(x) = K_3 \times C_{\max}(x) + \mathrm{Random}(K_3)$$

where $\mathrm{Random}(K_3)$ is a pseudo-random number generator (uniform distribution) that returns integer numbers between 0 and $K_3 - 1$.

● We can also combine two criteria in the same objective function. Hence, we use an objective function which combines the makespan, the H_2 heuristic, and the number of critical operations. We call this function $\mathcal{F}_3(x)$:

$$\mathcal{F}_3(x) = K_2 \times K_1 \times C_{\max}(x) + K_2 \times H_2(x) + C_{\mathrm{op}}(x)$$

● In all functions $\mathcal{F}_1$ up to $\mathcal{F}_5$, the makespan is the most important criterion. Finally, we define a function in which the main criterion is H_2. We call this function $\mathcal{F}_0(x)$:

$$\mathcal{F}_0(x) = K_0 \times H_2(x) + C_{\max}(x)$$

where K_0 is an upper bound for $C_{\max}(x)$ over all the possible busy-schedules of the current instance of JSP.

In the following section, we have set K_i to the maximum value of each criterion:

- $K_0 = D+1$ where D is the sum of the durations of all the operations of the instance: it is easy to see that the makespan of busy-schedules is always smaller than this value,
- $K_1 = M \times D^2$: this value is an immediate consequence of the definition of the previous value,
- $K_2 = J \times M$: there are $J \times M$ operations in the current instance of the JSP, so that there are no more than $J \times M$ critical operations,
- $K_3 = K_0$: using this value, the number given by Random(K_3) are of the same order of magnitude than the makespan.

In fact it is possible to define these values precisely, but this is useless for our application: all we need is to define upper-bounds of the criteria so that the makespan remains the main criterion.

IV. RESULTS

We use the benchmarks of the ORLIB [1]. In our test-suite, the size of the instances ranges from 10×10 to 30×15. Our SDHC has been tested with the six objective functions $\mathcal{F}_0$ to $\mathcal{F}_5$. For each instance, a set of 500 initial solutions are drawn at random. Then, the six functions are tested on this set of solutions. The results we have obtained are displayed in Table III.

For every instance of our test-suite, $\mathcal{F}_3$ provides the best results in terms of the quality of the local optima that are found. Clearly, the more information embedded in the objective function, the better the results. The worst objective function is generally $\mathcal{F}_1$ (where $C_{\max}$ is the only criterion). $\mathcal{F}_4$, the function which includes a random noise, performs at least as well as $\mathcal{F}_1$, generally better. $\mathcal{F}_0$ where $C_{\max}$ is not the main criterion, competes rather well with $\mathcal{F}_1$. The comparison between the results of $\mathcal{F}_2$ (where H_2 is the only secondary criterion) and the results of $\mathcal{F}_0$ (where H_2 is the main criterion) shows that H_2 is a good secondary criterion but a bad main criterion.

The objective functions may be sorted according to their average performance as follows :

$$\mathcal{F}_3 > \mathcal{F}_2 > \mathcal{F}_5 > \mathcal{F}_4 \geq \mathcal{F}_0 \geq \mathcal{F}_1$$

where $\mathcal{F}_3 > \mathcal{F}_2$ means that $\mathcal{F}_3$ performs better than $\mathcal{F}_2$.

TABLE III

This table sums up our results using the SDHC. The instances are identified with their name in the Orlib. The second line gives the size of the instance. The third line gives the value of the optimum if it is known, a range of it when it is unknown. Then for each function, the table gives the average value, the standard deviation, the minimum value and the maximum value of the makespan over 500 experiments. See the main text for the definition of the different functions.

Instance		mt 10x10	mt 20x5	ta01	abz7	ta31
Size		10x10	20x5	15x15	20x15	30x15
Optimum		930	1165	1231	656	$\frac{1764}{1766}$
$\mathcal{F}_0$	Aver	1073	1362	1465	779	2156
	St.D	40	44	42	17	45
	Mini	958	1238	1371	735	1999
	Maxi	1073	1515	1605	846	2305
$\mathcal{F}_1$	Aver	1085	1351	1467	782	2150
	St.D	39	46	46	18	43
	Mini	991	1228	1344	721	2036
	Maxi	1214	1506	1626	853	2300
$\mathcal{F}_2$	Aver	1069	1331	1440	770	2119
	St.D	37	42	44	18	45
	Mini	974	1229	1338	725	1990
	Maxi	1205	1452	1609	839	2278
$\mathcal{F}_3$	Aver	1068	1328	1439	770	2118
	St.D	37	43	45	19	44
	Mini	974	1229	1337	727	1990
	Maxi	1205	1452	1609	839	2278
$\mathcal{F}_4$	Aver	1083	1349	1459	782	2150
	St.D	41	45	44	18	44
	Mini	971	1217	1352	729	2036
	Maxi	1207	1488	1632	861	2300
$\mathcal{F}_5$	Aver	1081	1347	1464	778	2144
	St.D	39	46	46	19	43
	Mini	971	1222	1344	728	2021
	Maxi	1214	1506	1626	853	2295

We have embedded the objective functions $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ in a simple (non-deterministic) hill-climber (NDHC for short), on the same set of initial solutions. The results are similar to the results of the SDHC (see Table IV), but NDHC needs less evaluations than SDHC.

The objective functions are sorted according to their average performance as follows :

$$\mathcal{F}_3 > \mathcal{F}_2 > \mathcal{F}_1$$

Clearly, these results show that our approach is relevant and actually useful on real problems.

V. APPLICATION TO AN EVOLUTIONARY ALGORITHM

In order to demonstrate that this approach can be used in different hill-climbers, we have embedded the same set of objective functions in an evolutionary algorithm and run it on the same set of instances.

TABLE IV

This table sums up our results using the NDHC. The instances are identified with their name in the Orlib. The second line gives the size of the instance. The third line gives the value of the optimum if it is known, a range of it when it is unknown. Then for each function, the table gives the average value, the standard deviation, the minimum value and the maximum value of the makespan over 500 experiments.

Instance		mt 10x10	mt 20x5	ta01	abz7	ta31
Size		10x10	20x5	15x15	20x15	30x15
Optimum		930	1165	1231	656	$\frac{1764}{1766}$
$\mathcal{F}_1$	Aver	1084	1349	1460	781	2150
	St.D	42	43	47	19	47
	Mini	991	1239	1351	738	2020
	Maxi	1233	1509	1600	853	2339
$\mathcal{F}_2$	Aver	1068	1330	1442	774	2129
	St.D	39	48	45	19	44
	Mini	955	1211	1354	732	2020
	Maxi	1180	1473	1606	836	2271
$\mathcal{F}_3$	Aver	1067	1325	1440	774	2126
	St.D	38	46	44	19	43
	Mini	951	1215	1334	732	2020
	Maxi	1180	1470	1606	836	2267

We have designed the EA as discussed in [4] and we refer the reader to this reference for any further detail. Our EA uses direct encoding, this means that the data structure actually represents a schedule. We use a recombination operator based on the GA/GT crossover introduced in [12]. The mutation performs a swap of two operations and re-schedule all the operations that needs to be rescheduled (due to the swap). Clearly, our objective here is not to tune the EA to obtain the best results. Rather, we simply aim at showing that the same kind of improvements on the quality of the solutions that are reached can be obtained. Thus, we use standard population size, rate of application of operators and a ranking selection plus elitism. We will not go into more details about it because it would require a thorough presentation of the coding of solutions and a description of the operators that are out of the scope of this paper.

The results that were obtained to date are still partial with regards to those presented earlier on the SDHC (see Table V). However, it is again very clear that the extra information does help the algorithm to find better solutions. In all cases except one, the $\mathcal{F}_3$ function leads to the best performance. However, the performance of $\mathcal{F}_2$ with regards to the mere $\mathcal{F}_1$ is not so striking as for the SDHC.

TABLE V

THIS TABLE SUMS UP OUR RESULTS USING AN EA. THE SECOND LINE GIVES THE SIZE OF THE INSTANCE. THE THIRD LINE GIVES THE VALUE OF THE OPTIMUM IF IT IS KNOWN, A RANGE OF IT WHEN IT IS UNKNOWN. THEN FOR EACH FUNCTION, THE TABLE GIVES THE AVERAGE VALUE AND THE STANDARD DEVIATION OF THE MAKESPAN OVER 10 RUNS.

Instance		mt 10x10	mt 20x5	ta01	abz7
Size		10x10	20x5	15x15	20x15
Optimum		930	1165	1231	656
$\mathcal{F}_1(x)$	Aver	986	1210	1328	719
	St.D	17	16	22	5
$\mathcal{F}_2(x)$	Aver	987	1216	1310	719
	St.D	15	18	23	15
$\mathcal{F}_3(x)$	Aver	978	1203	1317	706
	St.D	17	23	12	6

VI. DISCUSSION AND PERSPECTIVES

In this paper, we have focused our attention on the objective function used by hill-climbers in general, evolutionary algorithms being one special case of this general class of meta-heuristics. We have concentrated ourselves on the problem which is raised when the search reaches a plateau of the fitness landscape. Driven by its objective function, the hill-climber is then totally unable to grasp on some information to guide itself towards the next point to visit. We have proposed a method to alleviate this problem which relies on the use of on-side properties that go along with the goodness of a solution with regards to the function being optimized. We have embedded this technique in two kinds of hill-climbers and demonstrated its usefulness on the simple JSP. We think that this technique can be used on any $\mathcal{NP}$-hard problem on which some knowledge is available (that is at least all the "well-known" $\mathcal{NP}$-hard problems). Furthermore, there is no reason why this scheme would not be applied to any optimization problem. Even for satisfiability problems, it is possible to define secondary criteria (see [10] for example). We have also shown that this technique can also be embedded into evolutionary algorithms and still improves the quality of the local optima that are found. Furthermore, this technique does not require a lot of implementation work and the computational cost can be low. (Obviously, this depends on the kind of on-side properties that are used and the cost to compute them.)

An other application of this technique in evolutionary algorithms would be to measure, and thus help maintain, the diversity of the population, the key-point of a good search for an evolutionary algorithm. It is not always obvious to figure out whether two individuals are different or not: at the genotype level, two individuals might seem different though representing two really identical individuals (*e.g.* when individuals are graphs, checking the equivalence of two graphs can not always be efficiently done); in the same time two individuals having the same fitness can be truly distinct. Using extra information as these on-side properties would help discriminate among the individuals with the same value for their main objective but different with regards to their genotypes.

REFERENCES

[1] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990. available via anonymous ftp on mscmga.ms.ic.ac.uk:/pub.

[2] J. Carlier and P. Chrétienne. *Problèmes d'ordonnancement, modélisation, complexité, algorithmes*. Masson, 1988. ISBN: 2-225-81275-6.

[3] D. Duvivier, Ph. Preux, C. Fonlupt, D. Robilliard, and E-G. Talbi. Impact de la fonction objectif sur les performances des algorithmes itératifs de recherche locale. In *Premier congrès de la Société Française de Recherche Opérationnelle et Aide à la Décision (ROADF'98)*, page 137, Paris, France, January 1998.

[4] D. Duvivier, Ph. Preux, and E-G. Talbi. Climbing up NP-hard hills. In *[16]*, pages 574–583, September 1996.

[5] D. Duvivier, Ph. Preux, and E-G. Talbi. Genetic algorithms applied to the job-shop scheduling problem. In *FUCAM'96, Workshop on Production Planning and Control*, Mons, Belgique, September 1996. 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).

[6] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the theory of NP-Completeness*. W.H. Freeman and Company, 1979. ISBN: 0-7167-1045-5.

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

[8] GOThA. Les problèmes d'ordonnancement. *Operations Research*, 27(1):77–150, 1993.

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

[10] Kenneth A. De Jong and William M. Spears. Using genetic algorithms to solve NP-complete problems. In *[14]*, pages 124–132, 1989.

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

[12] Ryohei Nakano and Takeshi Yamada. A genetic algorithm applicable to large-scale job-shop problems. In *[11]*, pages 281–290, 1992.

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

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

[15] E. Taillard. *Recherches itératives dirigées parallèles*. PhD dissertation, Ecole Polytechnique Fédérale de Lausanne, 1993.

[16] H-M. Voight, W. Ebeling, I. Rechenberg, and H-P. Schwefel, editors. *Proc. of the Fourth Conf. on Parallel Problem Solving in Nature*, Berlin, Germany, 1996. Springer-Verlag, Berlin. Lecture Notes in Computer Science, vol. 1141.