Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem

Matthieu Basseur, Franck Seynhaeve, El-ghazali Talbi
LIFL / University of Lille,
Bat. M3, 59655 Villeneuve d'Ascq, France
{basseur,seynhaeve,talbi}@lifl.fr

Abstract - Multi-objective optimization using evolutionary algorithms has been largely studied in the literature. Here, we propose formal methods to solve some problems appearing frequently in the design of such algorithms. To evaluate the effectiveness of the introduced mechanisms, we apply them to the flow-shop scheduling problem. We propose a dynamic mutation Pareto Genetic Algorithm (GA) in which different genetic operators are used simultaneously in an adaptive manner, taking into account the history of the search. We present a diversification mechanism which combines sharing in the objective space as well as in the decision space, in which the size of the niche is automatically calculated. We propose also an hybrid approach which combines the pareto GA with local search. Finally, we propose two performance indicators to evaluate the effectiveness of the introduced mechanisms.

I. Introduction

The flow-shop problem is one of the numerous scheduling problems [16]. This problem has received a great attention since its importance in many industrial areas. The proposed methods for its resolution vary between exact methods such as the branch & bound algorithm, specific heuristics and metaheuristics. However, the majority of these works study the problem in its single criterion form and aim mainly to minimize the makespan.

The flow-shop problem can be presented as a set of N jobs $J_1, J_2, \ldots, J_N$ to be scheduled on M machines. Machines are critical resources: one machine can not be assigned to two jobs simultaneously. Each job J_i is composed of M consecutive tasks $t_{i1}, \ldots, t_{iM}$, where t_{ij} represents the j^{th} task of the job J_i requiring the machine m_j. To each task t_{ij} is associated a processing time p_{ij}. Each job J_i must be achieved before the due date d_i.

In our study, we are interested in permutation flow-shop problems where jobs must be scheduled in the same order on all the machines. We have to minimize two objectives:

- C_{max} : Makespan (Total completion time),
- T : Total tardiness.

The task t_{ij} is scheduled at the time s_{ij}. The two objectives can be computed as follow:

$$f_1 = C_{max} = Max\{s_{iM} + p_{iM} | i \in [1 \ldots N]\}$$

$$f_2 = T = \sum_{i=1}^{N} [max(0, s_{iM} + p_{iM} - d_i)]$$

In the R. L. Graham & al. [7] notation this problem can be defined by F/perm, $d_i/(C_{max}, T)$.

This problem is NP-hard. It cannot be solved by exact methods for medium and large instances. The number of feasible solutions, for a $N * M$ problem instance, is $N!$ solutions.

As many works on multi-objective optimization, the flow-shop scheduling problem set some difficulties. Here we try to design mechanisms to answer these difficulties and we apply those mechanisms to an existent algorithm previously proposed in our research group [15]. We also introduce performance indicators to evaluate the effectiveness of those mechanisms.

II. An Adaptive Pareto Genetic Algorithm

Whatever the problem, there exists many mutation and crossover operators. Many tests must be done on each operator in order to know its effectiveness. Moreover, the efficiency of an operator may change during the algorithm: an operator may offer a better convergence at the beginning of the GA, but this convergence may stop earlier than with another operator. The success of an operator may also depend on the instance of the problem.

So, we are interested in an adaptive Pareto GA, in which the choice of the operator is done dynamically during the search. In this work, we have applied the proposed mechanism to mutation operators. The goal of our method is to use simultaneously several mutation operators during the GA, and to change their ratio according to evaluation results from the respecting offsprings it produces. So the algorithm always uses the best operator more often than the other operators. This approach has already been used for mono-objective problems [8]. We have adapted this method to multi-objective optimization, as described below:

1. Create an initial population of individuals. Assign to each mutation operator M_1, ..., M_n the same ratio $P_{M_i} = 1/(n * p_{mutation})$ (n is the number of mutation operators and $p_{mutation}$ the mutation ratio).
2. Apply the GA. Mutation operators are selected in respect to their selection ratios.

3. Evaluate the fitness of each solution before and after applying the mutation.
4. For each mutation operator M_i, compute his average growth value $Progress(M_i)$.
5. According to the values of $Progress(M_i)$, adjust the P_{M_i}.
6. Return to step 2.

A similar algorithm could be defined with other operators (crossover, local search, etc.).

The progress of a mutation operator M_i applied on a solution I is:

$$\Pi(I_{M_i}) = \begin{cases} 0 & \text{if } I \text{ is dominated by } I_{M_i} \\ 1 & \text{if } I \text{ dominates } I_{M_i} \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

where I_{M_i} is the solution after mutation. $Progress(M_i)$ is the average progress of $\Pi(I_{M_i})$ computed with each solution modified by the mutation M_i. $\|M_i\|$ is the number of times where M_i is applied:

$$Progress(M_i) = \frac{\sum \Pi(I_{M_i})}{\|M_i\|}$$

We also adjust the P_{M_i} ratios according to the values of $Progress(M_i)$. In order to keep each operators, we use a value δ, which indicates the minimal ratio value of each operator:

$$P_{M_i} = \frac{Progress(M_i)}{\sum_{j=1}^{n} Progress(M_j)} * (1 - n * \delta) + \delta$$

Table I presents an example where we consider four mutation operators and compute their new mutation ratio according to their progress. Let S' be the solution obtained by applying a mutation M_i on a solution S. If S' dominates S we note $S' > S$, if S dominates S' we note $S' < S$, else we note $S \sim S'$. This example shows that the mutation M_3 give better results than other operators, so we give to this mutation the best ratio for the next generation.

Mutation	$S' > S$	$S \sim S'$	$S' < S$	P_{M_i}
M_1	1	4	6	0.23
M_2	0	0	7	0.05
M_3	3	7	1	0.43
M_4	4	0	9	0.29

TABLE I
EXAMPLE OF P_{M_i} COMPUTATION ($\delta = 0.05$)

Application to the Flow-Shop Problem

The coding used for the chromosomal representation of a solution is a permutation of jobs. A position of a job defines his scheduling order. For our application, we consider two mutation operators. The first operator is an exchange between two jobs (Fig. 1). The second one, the insertion operator, consists in choosing randomly two points of the chromosome, and make a circular permutation between these points (Fig. 2) [11].

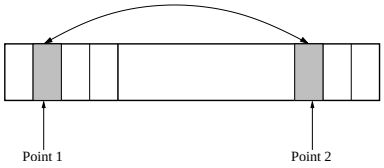

Fig. 1. Exchange operator

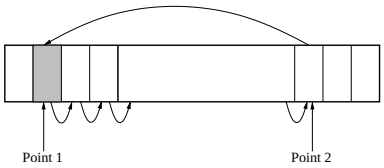

Fig. 2. Insertion operator

For the flow-shop problem, we observe that the insertion operator ratio performs better than the exchange operator ratio when the search is advanced, in many benchmarks. Experiments show that the adaptive algorithm performs better than the algorithm with only one operator (section V).

III. Diversification

In classical GAs, we can usually observe a diversity loss of the population, also called genetic drift. To face this drawback, many methods have been introduced: introduction of new randomly generated individuals, stochastic universal sampling [1], distance maintaining [14], crowding [14], neighborhood restriction [5] or sharing [6].

Here, we are interested in sharing which is used in many multi-objective GAs. In classical GAs, the solutions with best fitness are selected more often than the solutions with worst fitness. The sharing principle consists in the degradation of the fitness of individuals belonging to search regions with a high concentration of solutions. This process has the effect to favor the solutions diversity in the search space. The degradation of the fitness of an individual is realized thanks to a function sh called sharing function. The new fitness $f'(x)$ is equal to the original fitness f divided by the sharing counter $m(x)$ (niching counter) of the individual:

$$f'(x) = \frac{f(x)}{m(x)} \text{ with } m(x) = \sum_{y \in pop} sh(dist(x, y))$$

Let d be the distance between x and y. The sharing function sh is defined as follows:

$$sh(d) = \begin{cases} 1 - \frac{d}{\sigma_{share}} & \text{if } d < \sigma_{share} \\ 0 & \text{otherwise} \end{cases}$$

The constant σ_{share} designates the non-similarity threshold (niche size), i.e the distance from which two individuals x and y are not considered as belonging to the same niche. We can observe two different sharing, depending on how the distance between two individuals is computed.

Sharing function can be computed in the decision space, i.e the chromosomal representation of an individual. So, the distance d_1 between two so-

lutions x and y is the minimal number of mutations we must applied on x to have y.

This function can be computed too in the objective space, i.e fitness of individuals. The distance d_2 is equal to $|f_1(x) - f_1(y)| + |f_2(x) - f_2(y)|$.

Here we use a combination of these two approaches [12], [14]. The function sh takes the following form:

$$sh(d_1, d_2) = \begin{cases} 1 - \frac{d_1(x,y)}{\sigma_1} & \text{if } d_1(x,y) < \sigma_1 \\ & \text{and } d_2(x,y) \geq \sigma_2 \\ 1 - \frac{d_2(x,y)}{\sigma_2} & \text{if } d_1(x,y) \geq \sigma_1 \\ & \text{and } d_2(x,y) < \sigma_2 \\ 1 - \frac{d_1(x,y)d_2(x,y)}{\sigma_1\sigma_2} & \text{if } d_1(x,y) < \sigma_1 \\ & \text{and } d_2(x,y) < \sigma_2 \\ 0 & \text{otherwise} \end{cases}$$

where σ_1 and σ_2 are respectively the phenotypic and genotypic niche sizes, and d_1 and d_2 are respectively the phenotypic and genotypic distance between two solutions x and y.

Computation of σ_{share} in objective space

The effectiveness of the sharing principle mainly depends on the parameter σ_{share} which must be set carefully. In fact, diversification becomes inefficient with a low value of σ_{share}, but the progression speed of the front become too small when this value is too high. So, to obtain a good diversification, we have to make tests on each instance of problem to find a good approximation of σ_{share}, and this value must be set for each objective.

Some works propose methods to approximate σ_{share}. Some rough guidelines for setting these parameters in single-objective cases are given by K. Deb [3]. He suggests fixing σ_{share} at some known minimal separation between desired optima. Horn applied this mechanism to the multi-objective case [9] where σ_{share} must be computed to spread solutions over all the Pareto front, with the hypothesis that the solutions are ideally distributed on the front.

Let N be the population size. So, we can have an approximation of σ_{share} area:

$$Area_{niche}[\sigma_{share}] \approx \frac{Area_{pareto}}{N}$$

where $Area_{niche}[\sigma_{share}]$ is the area of the intersection of a niche and the Pareto front. In a n-dimensional space volume, a niche is a hypersphere of dimension n and the Pareto front is a space of dimension n-1. Then their intersection is a space of dimension n-1 which can be approximated by $(\sigma_{share})^{n-1}$. So, in two dimensions, $\sigma_{share} \approx Area_{pareto}/N$.

Fonseca & Al. [4] approximated $Area_{pareto}$ by a hyper-cube of dimension n defined by the extreme values of the front (Fig. 3). The front area is less than half of the hyper-cube surface.

We consider that the computation of the distance is based on Euclidian distance. M_j and m_j are

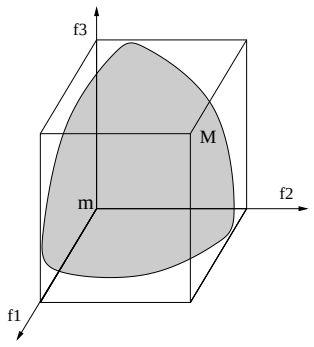

Fig. 3. Space occupied by the front approximated by an hyper-cube: Example with three objectives

respectively the maximum and the minimum value for the objective j. Then the maximum bound for this area is:

$$Area_{Pareto} \leq \sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} (M_j - m_j)$$

The lower bound is the hypotenuse connecting the extreme values. Then in two dimensions, we have:

$$Area_{Pareto} \geq \sqrt{|M_1 - m_1|^2 + |M_2 - m_2|^2}$$

When $n \geq 3$, the lower bound approximation is more difficult. So, when $n \geq 3$, we use only the upper bound of $Area_{Pareto}$ to approximate σ_{share}:

$$\sigma_{share} \leq \sqrt[n-1]{\frac{\sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} (M_j - m_j)}{N}}$$

In two dimensions, we have an upper and a lower bound for the surface occupied by the front:

$$\frac{\sqrt{|M_1 - m_1|^2 + |M_2 - m_2|^2}}{N} \leq \sigma_{share} \leq \frac{|M_1 - m_1| + |M_2 - m_2|}{N}$$

We consider that the objectives are normalized between 0 and 1. So, we have two approximations for σ_{share}: $2/N$ and $\sqrt{2}/N$.

For our application, we use the Manhattan distance notion. This distance has been chosen to favor a diagonal front [9] and to simplify the combinatorial complexity of our algorithm. So, with normalized values of the front, $Area_{Pareto}$ is a constant equal to 2 (the Manhattan distance between the point (0,1) and the point (1,0)). So, we take $2/N$ as value of σ_{share}.

In our application, we use combined sharing and the automatic computation of σ_{share}. Results show that the set of solutions obtained are more diversified, and the space explored is larger than without these mechanisms (section V).

IV. An Hybrid Approach

In our initial study of the flow-shop problem [15], we combined GA with local search in order to refine the search. The idea is first to run the GA in order to get a first approximation of the Pareto frontier and then to apply a local search on every non-dominated solution. The local search has the merit to improve the solutions, but the progresses realized appears only for problems of important size.

Our idea is to hybridize our GA by a memetic algorithm. The memetic algorithm we use is similar to a GA, but its mutation operators are replaced by a heuristic. The heuristic we use is a complete local search in order to keep the advantages of the GA (exploration) and to accelerate the algorithm convergence.

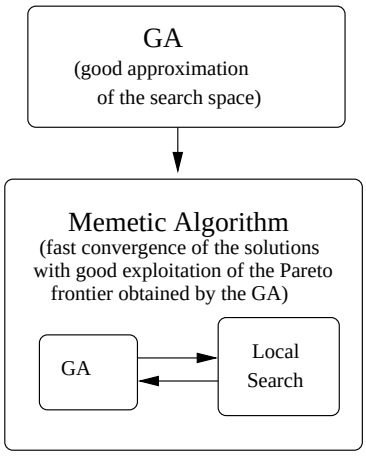

Fig. 4. GA + Memetic algorithm: the method

The effect of memetic algorithm on the Pareto front obtained by the GA is schematized on Fig. 5.

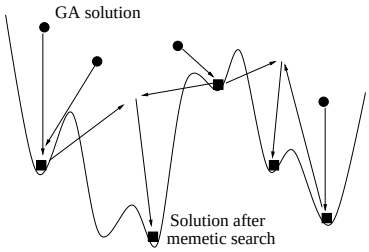

Fig. 5. GA + Memetic method: the effect

The convergence is very fast with this type of memetic algorithm, but each generation of the algorithm takes a long time in comparison to the time taken by a single GA generation. So, we suggest to realize only a few generations of memetic search after the GA (Fig. 4).

Results obtained on the flow-shop scheduling problem show that the memetic method applied after a GA offers better results than only the GA or memetic search. Results obtained on many instances show the effectiveness of this mechanism (section V).

GAs progression is too slow when solutions become good. Hybridization by local search offers a fast convergence, but the Pareto population are not really exploited. This mechanism offers a fast convergence with a good exploitation of the Pareto population.

V. Performance Evaluation

The evaluation of the performance of the algorithm has been realized on some Taillard benchmarks [13], extended to the bi-objective case [15][1]. We have evaluated our previous algorithm on the same benchmarks. The table II describes the results obtained:

- *Problem* is the instance treated.
- *Dim* is the dimension of the problem. NxM denotes an instance of the flow-shop problem with N jobs and M machines.
- *NB gen* is the number of generations of the algorithms made.
- *UB* is the upper bound obtained for the Makespan in single-objective studies.
- M_1 and T_1 are the best Makespan and Tardiness obtained by the basic algorithm.
- M_2 and T_2 are the best Makespan and Tardiness obtained with the introduced mechanisms.
- Dev_1 and Dev_2 are the deviation of the two algorithms regarding *UB*.

For the Makespan, we obtain the same deviation than our previous algorithm for the three first instances, but we improve it for the other instances. Our deviation fluctuates between 0% and 1.19%. According to the tardiness, we improve our previous results for all the instances except ta_20_20_01.

These results give us an idea on the progress made, but only on the extremities of the front. Proper comparison of two multi-objective optimization algorithms is a complex issue. Several different solutions have been proposed in recent years. Solutions quality can be assessed in different ways. Some approaches compare the obtained front with the true Pareto front [17]. Others approaches evaluate a front with a reference point [10]. Some performance measures dont use any reference point or front to evaluate an algorithm [18].

Here, we have to compare two different algorithms, without knowning the true Pareto front. We propose two complementary types of performance indicators: the contribution and the entropy. The contribution indicator quantifies the domination between two sets of non-dominated solutions. The entropy indicator gives an idea about the diversity of the solutions found.

Contribution

The contribution of a set of solutions PO_1 relatively to a set of solutions PO_2 is the ratio of non-dominated solutions produced by PO_1.

- Let C be the set of solutions in $PO_1 \cap PO_2$.
- Let W_1 (resp. W_2) be the set of solutions in PO_1 (resp. PO_2) that dominate some solutions of PO_2 (resp. PO_1).
- Let L_1 (resp. L_2) be the set of solutions in PO_1 (resp. PO_2) that are dominated by some solutions of PO_2 (resp. PO_1).

[1] The bi-objective benchmarks and the results obtained are available on the web (**http://www.lifl.fr/~basseur**).

Problem	Dim	Nb gen	UB	M_1	Dev_1	M_2	Dev_2	T_1	T_2
ta_20_5_01	20x5	50000	1278	1278	0%	**1278**	**0%**	453	**452**
ta_20_5_02	20x5	50000	1359	1359	0%	**1359**	**0%**	491	**491**
ta_20_10_01	20x10	80000	1582	1586	0.25%	**1586**	**0.25%**	1508	**1224**
ta_20_10_02	20x10	80000	1659	1674	0.9%	**1672**	**0.78%**	1342	**1275**
ta_20_20_01	20x20	200000	2297	2330	1.43%	**2308**	**0.48%**	1062	**1097**
ta_50_5_01	50x5	200000	2724	2735	0.4%	**2729**	**0.18%**	3629	**3364**
ta_50_10_01	50x10	200000	3037	3126	2.93%	**3063**	**0.85%**	6653	**4636**
ta_50_20_01	50x20	300000	3886	3990	2.67%	**3933**	**1.19%**	11379	**7667**

TABLE II
PERFORMANCE EVALUATION (NEW RESULTS IN BOLD)

- Let N_1 (resp. N_2) be the other solutions of PO_1 (resp. PO_2): $N_i = PO_i \setminus (C \cup W_i \cup L_i)$.
- Let PO^* be the set of Pareto solutions of $PO_1 \cup PO_2$. So, $\|PO^*\| = \|C\| + \|W_1\| + \|N_1\| + \|W_2\| + \|N_2\|$.

The contribution of the algorithm PO_1 relatively to PO_2 is stated as[2]:

$$C(PO_1/PO_2) = \frac{\frac{\|C\|}{2} + \|W_1\| + \|N_1\|}{\|PO^*\|}$$

For example, we evaluate the contribution of the two sets of solutions PO_1 of PO_2 of Fig. 6: solutions of PO_1 (resp. PO_2) are represented by circles (resp. crosses). We have $C(PO_1, PO_2) = 0.7$ and $C(PO_2, PO_1) = 0.3$.

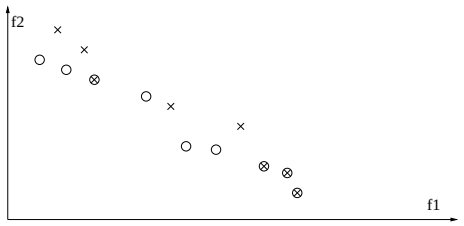

Fig. 6. Example of contribution (C=4, W_1=4, W_2=0, N_1=1, N_2=1)

Entropy

Let PO_1 and PO_2 be two sets of solutions.

- Let PO^* be the set of optimal Pareto solutions of $PO_1 \cup PO_2$.
- Let N_i be the cardinality of solutions of $PO_1 \cup PO^*$ which are in the niche of the i^{th} solution of $PO_1 \cup PO^*$. The size of the niche is defined in section III.
- Let n_i be the cardinality of solutions of PO_1 which are in the niche of the i^{th} solution of $PO_1 \cup PO^*$.
- Let C be the cardinality of the solutions of $PO_1 \cup PO^*$.
- Let $\gamma = \sum_{i=1}^{C} \frac{1}{N_i}$ be the sum of the coefficients affected to each solution. The more concentrated is a region of the solution space, the lower will be the coefficients of its solutions.

[2]Let us remark that $C(PO_1/PO_2) + C(PO_2/PO_1) = 1$.

Then, the following formula is applied to evaluate the entropy E of PO_1, relatively to the space occupied by PO^*:

$$E(PO_1, PO_2) = \frac{-1}{log(\gamma)} \sum_{i=1}^{C} \left(\frac{1}{N_i} \frac{n_i}{C} \log \frac{n_i}{C}\right)$$

The two previous indicators have been applied to the fronts P_1 obtained with the previous algorithm and our fronts obtained with the introduced mechanisms P_2 (Table III). These tests show that the contribution and the entropy of P_2 are always better than our previous results. In particular, our contribution reaches 1 for instances with 50 jobs.

	Contribution		Entropy	
Problem	P_1/P_2	P_2/P_1	P_1/P_2	P_2/P_1
ta_20_10_01	0.19	0.81	0.79	0.96
ta_20_10_02	0.31	0.69	0.88	0.92
ta_20_20_01	0.19	0.81	0.79	0.97
ta_50_5_01	0	1	0.72	0.94
ta_50_10_01	0	1	0.76	0.97
ta_50_20_01	0	1	0.70	0.96

TABLE III
PERFORMANCE EVALUATION (MULTI-OBJECTIVE INDICATORS)

Convexity

We also evaluate the interest of the Pareto approach by analyzing the landscape of the Pareto front in terms of convexity. In fact, it is well known that aggregation approaches generate only supported solutions i.e those which are on the convex hull of the set of solutions (Fig. 7).

We compute the part of supported solutions obtained on each front PO^* (Table IV, with S the set of solutions which are on the convex hull of PO^*). The convex hull is computed by incremental method whose complexity is $O(p.log(p))$ where p is the number of solutions [2].

The proportion of supported solutions is small. This show that we may loose a large part of Pareto optimal solutions by solving the flow-shop scheduling problem with an aggregation approach.

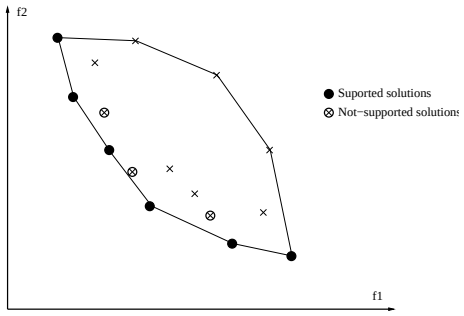

Fig. 7. Convexity

Problem	$\|PO^*\|$	$\|S\|$	Convexity
ta_20_10_01	36	8	0.222
ta_20_10_02	17	8	0.471
ta_20_20_01	30	7	0.233
ta_50_5_01	12	5	0.417
ta_50_10_01	22	7	0.318
ta_50_20_01	44	12	0.273

TABLE IV
CONVEXITY OF THE SET OF SOLUTIONS OBTAINED WITH OUR ALGORITHM

VI. Conclusion and Perspectives

In this study, we have introduced general mechanisms for multi-objective optimization. We have proposed an Adaptive Pareto Genetic Algorithm where:

- Different genetic operators are used simultaneously in an adaptive manner,
- A combined sharing is applied as well in the objective space as in the decision space,
- The size of the niche in the objective space is computed automatically,
- An hybrid approach combining Pareto GA with a memetic approach is introduced.

Moreover, we have introduced two indicators adapted to the performance evaluation of multi-objective optimization algorithms: contribution and entropy.

The proposed approach has been tested successfully on flow-shop scheduling problems.

Interest of each mechanism is showed on the web at **http://www.lifl.fr/~basseur**.

The mechanisms defined here will be included in the PARADISEO platform, which is a parallel and distributed object-oriented programming environment for multi-objective optimization developed by our research group.

Another perspective of this work is a more general study of landscape (convexity, continuity,...) of multi-objective problems to adapt the design of such algorithms to the instance of the problem.

References

[1] J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of international conference on Genetic Algorithms and their application*, pages 101–111, 1985.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter Computational Geometry, pages 886–915. MIT Press and McGraw-Hill Book Company, jun 1990.

[3] K. Deb. Genetic algorithms in multimodal function optimization. Master's thesis, University of Alabama, 1989.

[4] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms 1: A unified formulation. Technical Report 564, University of Sheffield, UK, January 1995.

[5] K. Fujita, N. Hirokawa, S. Akagi, S. Kimatura, and H. Yokohat. Multi-objective optimal design of automotive engine using genetic algorith. In *Proceedings of the 1998 ASME Design Engineering Technical Conference*, Atlanta, Georgia, U.S.A.

[6] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multi-modal function optimisation. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[7] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*, volume 5, pages 287–326. 1979.

[8] T.-P. Hong, H.-S. Wang, and W.-C. Chen. Simultaneous applying multiple mutation operators in genetic algorithm. *Journal of Heuristics*, 6(4):439–455, sep 2000.

[9] J. Horn and N. Nafplioti. Multiobjective optimization using the niched pareto genetic algorithm. Technical Report IlliGAL 93005, Urbana, IL, 1993.

[10] A. Jaszkiewicz. On the performance of multiple objective genetic local search on the 0/1 knapsack problem. a comparative experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznan University of Technology, Poznan, Poland, jul 2000.

[11] T. Murata and H. Ishibuchi. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Trans. on System, Man, and Cybernetics—Part C: Applications and Review*, 28(3):392–403, 1998.

[12] J. Rowe, K. Vinsen, and N. Marvin. Parallel GAs for Multiobjective Functions. In J. T. Alander, editor, *Proceedings of the Second Nordic Workshop on Genetic Algorithms and Their Applications (2NWGA)*, pages 61–70, Vaasa, Finland, aug 1996. University of Vaasa.

[13] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operations Research*, 64:278–285, 1993.

[14] E. G. Talbi. Metaheuristics for multiobjective combinatorial optimization : state of the art. Technical report, LIFL, University of Lille, France, 2000.

[15] E. G. Talbi, M. Rahoual, M. H. Mabed, and C. Dhaenens. A hybrid evolutionary approach for multicriteria optimization problems : Application to the flow shop. In E. Zitzler et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 416–428. Springer-Verlag, 2001.

[16] V. T'kindt and J. C. Billaut. *Multiobjective scheduling (in french)*. Presses universitaires de Tours, 2000.

[17] D. A. V. Veldhuizen and G. B. Lamont. On measuring multiobjective evolutionary algorithm performance. In *In 2000 Congress on Evolutionary Computation, Piscataway, New Jersey*, volume 1, pages 204–211, Jul 2000.

[18] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3, nov 1999.