Concurrent product configuration and process planning: Some optimization experimental results

Paul Pitiot, Michel Aldanondo, Élise Vareilles

Concurrent product configuration and process planning: Some optimization experimental results

Paul Pitiot [a,b], Michel Aldanondo [b,*], Elise Vareilles [b]

[a] 3IL, CCI Rodez, Rodez, France
[b] Toulouse University, Mines Albi, France

1. Introduction

This paper concerns aiding mass customization, or, more accurately, how the two activities of product configuration and production planning can be achieved, optimized and computer supported in a concurrent way. An example relevant to the configuration and planning of a small aircraft illustrates our propositions all along the paper.

1.1. Concurrent configuration and planning

Product configuration can be defined as deriving the definition of a specific or customized product (through a set of properties, sub-assemblies or bill of materials, etc.) from a generic product or a product family [1] or [2]. Similarly, deriving a specific production plan (operations, resources to be used, etc.) from some kind of generic process plan, while respecting the product characteristics and the customer requirements, can define production planning with respect to product configuration [3] or [4]. As the decisions relevant to each of these two activities are closely dependent:

- decisions associated with the configuration of a product can have strong consequences on the planning of its production process (for example, a luxury finish requires additional manufacturing time),
- planning decisions can imply tough constraints to product configuration (for example, a given assembly duration prevents from using a particular kind of engine).

It is necessary to associate them in order to avoid inconsistencies and the traditional sequence "product configuration, then production planning". If many scientific works have been independently achieved on configuration or planning, as far as we know, scientific production is far less important when they are considered concurrently. Some initial ideas where proposed by Steward and Tate [5] and Schierholt [3]. More recent works can be found in [12] or [6].

1.2. Different requirements, two configuration/planning steps

Most of the times, configuration techniques support interactively the processing of customer requirements. This means that the consequences of each "elementary requirement" are computed and shown to the customer. By elementary requirement, we mean

* Corresponding author at: Centre Génie Industriel, Ecole des Mines d'Albi, 81013 Albi, France. Tel.: +33 563 49 32 34; fax: +33 563493183.
E-mail addresses: p.pitiot@aveyron.cci.fr (P. Pitiot), michel.aldanondo@mines-albi.fr (M. Aldanondo), elise.vareilles@mines-albi.fr (E. Vareilles).

a restriction of the domain of a variable involved in configuration (for example "plane capacity belongs to [6, 12]"), or in planning (for example "final assembly operation should be located in Italy"). As the goal of a company that uses concurrent configuration and planning techniques is to put on the market a product diversity that covers a large demand segment, the elementary requirements can become very diverse and numerous. The process in turn can be tricky and longer.

Each customer can be interested in different kinds of requirements, for example customer "A" can be mainly interested in the product "performance" (speed, altitude, etc.) while the product "comfort" (finish level, seat-space, etc.) may mostly attract customer "B". The idea is to limit the collection of requirements to those that mainly interest each customer. These requirements are named "non-negotiable requirements", while the remaining ones are named "negotiable requirements". Therefore, a first step interactively processes the non-negotiable requirements and then asks the software to complete autonomously the processing of the negotiable requirements in a second step. This autonomous computation can be achieved either with default values or with some multi-criteria optimization (cost, due-date, performances, etc.). This paper focuses on this last optimization issue. For paper clarity, we will only consider the two conflicting criteria cost and cycle time.

We therefore consider the concurrent configuration and planning process presented in Fig. 1 in two successive steps. Step1: interactive configuration and planning which processes non-negotiable requirements and provides a first solution space reduction. Step 2: response optimization which processes negotiable requirements and provides a Pareto front shown to the customer for a solution selection. This paper is mainly concerned by this second step.

1.3. Goal and organization of the paper

In a previous paper [7] we have proposed an original adapted evolutionary algorithm for this problem "CFB-EA" (for constrained filtering based evolutionary algorithm). However, the presentation was mainly descriptive and only some initial experimental results could be presented. Our objective in this paper is to prove that CFB-EA is a good candidate for optimizing concurrent configuration/planning problems. Thus, we propose to:

- evaluate the CFB-EA algorithm in detail. For that purpose, a survey of the scientific literature will help us identify a competitive evolutionary algorithm "FRB-EA" (for feasibility rules based evolutionary algorithm), in order to set up experimental comparisons,
- for a given problem, identify a size limit where exact optimization, a branch and bound (B&B) in our case, cannot be used due to computation duration and must be replaced by evolutionary computations (CFB-EA or FRB-EA) in our case.

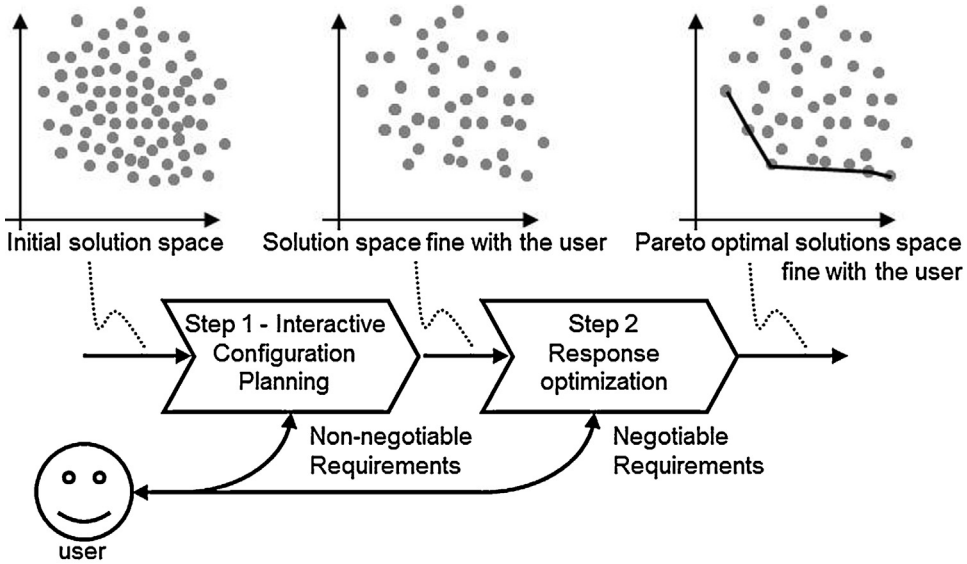
The paper is consequently organized as follows. In the next section, we describe how the previous two steps approach can be supported with constraint processing and discusses industrial and practical issues. In the third section, we formalize the optimization problem, review optimization techniques and finally detail the three optimization algorithms that will be used for experimentations. In the last section, we present experimental results that highlight the performance of CFB-EA and the limit where the exact approach should be replaced by evolutionary computations.

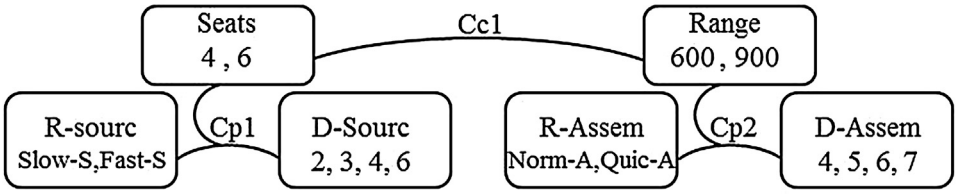
2. Optimizing configuration and planning considered as a CSP

2.1. Concurrent configuration and planning as a CSP

Many configuration or planning studies have shown that each of these two problems can be successfully processed when considered as a constraint satisfaction problem (CSP). As far as configuration is concerned, it has been explained by [8] or [9] while good surveys about planning can be found in [10] or [11]. Assuming a CSP for each problem, it is possible to bring them together in a single CSP and to process them concurrently in order to achieve support for concurrent configuration and planning.

This concurrent process and the supporting constraint framework present three main interests. Firstly, constraints linking configuration and planning can be processed in both directions (from product configuration to process planning, for example: a large flying range requires a specific tank assembly resource–from planning to configuration, for example: a given assembly duration forbids such cabin layout). Secondly, product and planning requirements can be processed in any order, avoiding the already mentioned sequence: "configure product then plan its production" [12]. Thirdly, the CSP framework is well suited for interactive





Seats	R-Sourc	D-Sourc
4	Slow-S	4
4	Fast-S	2
6	Slow-S	6
6	Fast-S	3

Cp1 (Seat, R-Sourc, D-Sourc),

Seats	Range
4	900
6	600
6	900

Cc1(Seats, Range)

Range	R-Assem	D-Assem
600	Norm-A	5
600	Quic-A	4
900	Norm-A	7
900	Quic-A	6

Cp2 (Range, R-Assem, D-Assem)

processes, thanks to constraint filtering techniques, see [13] for discrete constraints, [14] for numerical ones and [15] for piecewise functions, whilst also being adapted to optimization thanks to the various problem-solving techniques available (see [16]). In order to highlight the benefits of filtering capabilities, as explained in [10], we assume infinite capacity planning. This means that we consider that production is launched according to each customer order and that the production capacity is adapted accordingly.

In order to illustrate the problem under discussion, we consider a very simple example dealing with the configuration of a small plane and the planning of its production process. The constraint model is shown in Fig. 2. The plane is defined by two product variables: number of seats (noted Seats, with two possible values 4 or 6) and flight range (noted Range, with possible values 600 or 900 km). A constraint Cc1 forbids a plane with 4 seats and a range of 600 km. The production process contains two sequential operations: sourcing and assembling. (noted Sourc and Assem). Each operation is described by two process variables: the resources that can be used and the relevant operation duration. For sourcing, the variables are the resource (noted R-Sourc, with possible resources "Slow-S" and "Fast-S") and duration (noted D-Sourc, with possible durations 2–4 and 6 weeks). For assembling, the variables are the resource (noted R-Assem, with possible resources "Norm-A" and "Quic-A") and the duration (noted D-Assem, with possible durations 4–7 weeks). Two constraints linking product and process variables modulate configuration and planning possibilities: one linking seats with sourcing, Cp1 (Seats, R-Sourc, D-Sourc), and a second one linking range with assembly, Cp2 (Range, R-Assem, D-Assem). The allowed combinations of each constraint are shown in the three tables of Fig. 2. The process cycle time is the sum of the two operation durations obtained with a numerical constraint (cycle time = D-Sourc + D-Assem).

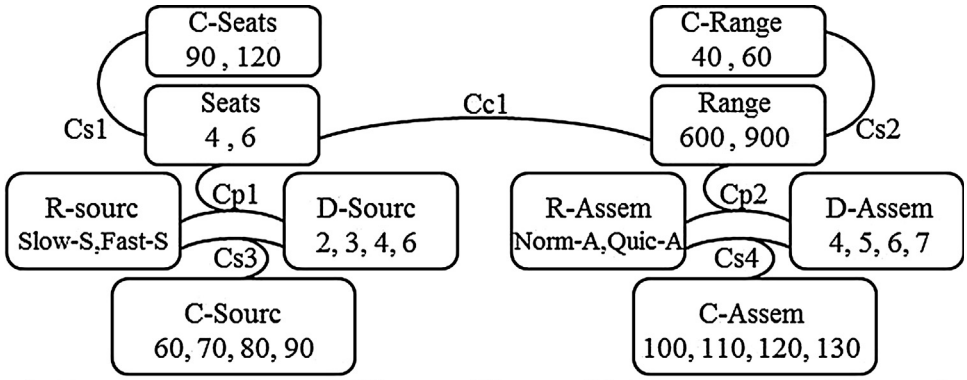
Without taking constraints into account, this model shows a combinatory of 4 for the product (2×2) and 64 for the production process $(2 \times 4) \times (2 \times 4)$, providing a combinatory of 256 (4×64) for the whole problem. When constraints are taken into consideration, 12 solutions emerge (for product and production process), with a cycle time belonging to the interval [7,13]. During the first step of interactive configuration and planning, the user can input any requirement and filter the constraints of the problem. For example, a product requirement "Range = 900″ will imply a cycle time belonging to [8,13], while a process requirement of "cycle time $<$ 10″ will imply at least a fast sourcing resource (R-Sourc, = "Fast-S"). Once this first step has been achieved, all the requirements that interest the user (the non-negotiable ones) are taken into account and many solutions can remain. Therefore finding the optimal solutions is the next natural step.

2.2. Optimizing configuration and planning concurrently

The addressed optimization problem corresponds with a constrained multi-criteria optimization problem. A first specificity of this concurrent configuration/planning problem is that the solution space can be large. It is reported in [17] that a configuration solution space of more than 1.4×10^{12} is required for a car-configuration problem. When planning is added, the combinatorial structure can become very large. In the last section, we will conduct experimentations with a problem space of 10^{17}. A second specificity is that the size of the problem depends on the quantity of negotiable requirements. A customer with many and detailed requirements will have very few negotiable requirements leading to a small optimization problem. Conversely, a customer with low or weak expectations will generate a large optimization problem. A third specificity is that the problem is not hardly constrained, there are always many solutions as most companies must propose many solutions to their customers. In our experimentations the ratio constrained combinatory/unconstrained combinatory varies from 10% down to 0.001%.

A similar remark to the one in the previous concurrent configuration and planning section proves relevant for optimization. We found many research works independently dealing with the optimization of each activity. Many configuration optimization studies show that evolutionary approaches are the most appropriate, see for example [18] or [46]. The literature dealing with production planning optimization is prolific and many techniques can be used, as summarized in the recent small survey [19]. But, as far as we know, very little can be found considering them concurrently. Furthermore, the second problem specificity (quantity of remaining negotiable requirements) allows considering either exact optimization (for small problem instances) or meta-heuristics (for larger instances) for investigations.

Various criteria can characterize a solution. On the product configuration side, there are, for example, performance and product cost. On the production planning side, the criteria could be cycle time and process cost. Most of the time, product performance is case-dependent, as explained in [20] and can characterize various product parameters, such as speed, power, energy consumption, etc. Cycle time matches the ending date of the last production operation of the configured product. Cost is at least the sum of the product cost and the process cost [21]. As these criteria are in conflict, it is better for decision aiding to offer the customer a set of possible compromises in the form of a Pareto Front (for example, performance/cost, performance/time or time/cost) rather than a single solution that aggregates criteria. For



C-Seats	Seats
90	4
120	6

Cs1 (C-Seats, Seats)

C-Range	Range
40	600
60	900

Cs2 (C-Range, Range)

C-Sourc	R-Sourc	D-Sourc
60	Slow-S	4
80	Fast-S	2
70	Slow-S	6
90	Fast-S	3

Cs3 (C-Sourc, R-Sourc, D-Sourc),

C-Assem	R-Assem	D-Assem
100	Norm-A	5
120	Quic-A	4
110	Norm-A	7
130	Quic-A	6

Cs4 (C-Assem, R-Assem, D-Assem)

Fig. 3. Concurrent configuration and planning model to be optimized.

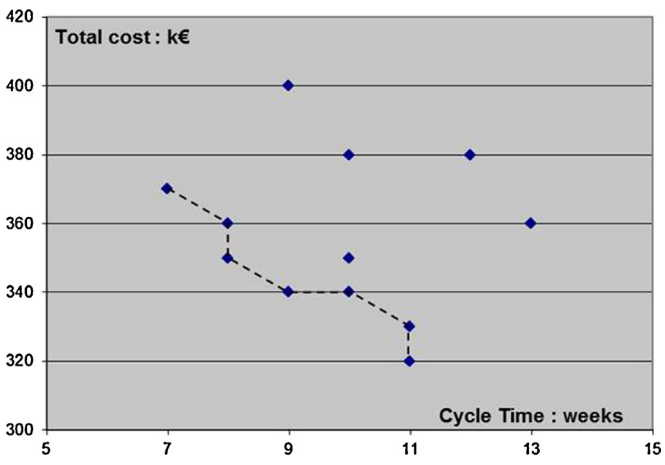
clarity, as already said we assume only the two criteria, cycle time and total cost.

In order to complete our example, we consider now a time/cost compromise, and define total cost and cycle time criteria as shown in Fig. 3. For cost, each product variable and each process operation is associated with a cost parameter and a relevant cost constraint: (C-Seats, Cs1), (C-Range, Cs2), (C-Sourc, Cs3) and (C-Assem, cs4) and the total cost is obtained with a numerical constraint. The cycle time is unchanged.

Total cost = C-Seats + C-Range + C-Sourc + C-Assem.
Cycle time = D-Sourc + D-Assem

The twelve previously-mentioned solutions are shown in Fig. 4, with the Pareto front bringing together the optimal ones. In this figure, all solutions are present.

As previously stated, when non-negotiable requirements are processed during interactive configuration and planning, some of these solutions are removed. Once all these requirements are processed, the optimization step identifying the Pareto front can be launched in order to propose a set of optimal solutions to the customer.



2.3. Practical and operational issues

2.3.1. Industrial issues

All industrial companies producing in assemble to order (ATO) or make to order (MTO) have to face the problem of configuring the product and also providing the customer with a delivery date. Most of the time, product configuration is achieved thanks to configuration techniques but three kinds of solutions can be found regarding delivery date computations. The first one or simplest one is to provide a standard duration that is more or less adapted to a small number of product and process key factors. The second one or most complicated one takes into account inventory levels and the availability of resources. They are very difficult to maintain and require a strong integration with ERP. The one used in this paper places itself between the two previous ones. It assumes infinite resource capacity and infinite inventory levels but respects production and sourcing cycle time.

It is also important to note that the problem, even if it is a strict ATO, can be very different according to the quantity and the cost of sub-assemblies and components. A simple product gathering 15 cheap components (as a personal computer for example) requires much less optimization processing than a complex product gathering hundreds of components and large customized sub-assemblies (as for example the cement production plant discussed in [47]). Needless to say that this issue totally differs depending on the kind of configured products: a computer of 1 k€, a car of 50 k€, a boat of 500 k€ or a plane of 5 M€. Therefore, if a 1 h process sounds a maximum for processing the requirements of a computer or a car, a night of computation would probably not be a problem for the configuration of a boat or a plane.

2.3.2. Web site issues

Most companies proposing products with customization possibilities show a configuration application on their web sites.

All car manufacturers have a configuration software on their web site, as it often is with PC providers. In this way, the customer can configure his product himself. However, most of the time there is no production planning issues and no delivery date processing associated with such configuration software as far as we know. Web sites seem oriented toward selling and do not consider production issues. If delivery dates are proposed, they are most of the time based on standard durations.

The computation time necessary to provide some optimization results can probably explain this lack of optimization possibilities on the web. For business to customer (B to C) in particular, this needs some kind of information exchange in order to be able to send the optimization results after the end of the session. For business to business (B to B) this sounds much more manageable and requires a light customer/supplier relationship.

2.3.3. Software provider issues

Many software relevant to configuration and planning are available on the market and most of the ERP systems (like for example Oracle™ or SAP™) propose planning and configuration modules.

With their integrated solutions, ERP editors provide sophisticated optimization solutions that take into account inventory levels and the availability of resources (second solution of previous industrial issues). However, optimization requires the computation of solutions and each solution corresponds with the selection of specific components, sub-assemblies, manufacturing or sourcing resources. Therefore, complex queries that are time consuming become necessary and tend to reserve this kind of optimization techniques for industrial situations where gaining less than 1% efficiency, cost or due date is important. Here are some examples that can correspond with this situation: airplane cabin layout [22], factory configuration [21], railway interlocking system [23] or telephone switching system [45]. These are complex and costly products mainly sold on a B to B basis.

Many configurator editors that are not ERP providers (like for example Tacton™, Bigmachines™ or Caméléon™) provide optimization solutions. But as they do not manage any production data (inventory levels and resource capacity), the previous optimization of concurrent configuration and planning proves almost impossible for them. By cons they can propose either adapted standard durations or the one we propose. We have seen some specifically design software modules showing a behavior very similar to the one we propose.

2.3.4. Conclusion relevant to practical issues

The previous discussion shows that concurrent configuration and planning with the relevant optimization problem has clearly become a subject that interests both industrial companies and software providers. But, as explained in a detailed survey about configuration software [24], it is always very delicate to know about configuration software possibilities. Furthermore software vendors do not communicate much on how their solutions work and handle specific cases. Likewise industrial companies are rather secret about their knowledge about product quotation and delivery dates policy. However, many authors agree that such configuration and planning software are key element for mass customization development in industry.

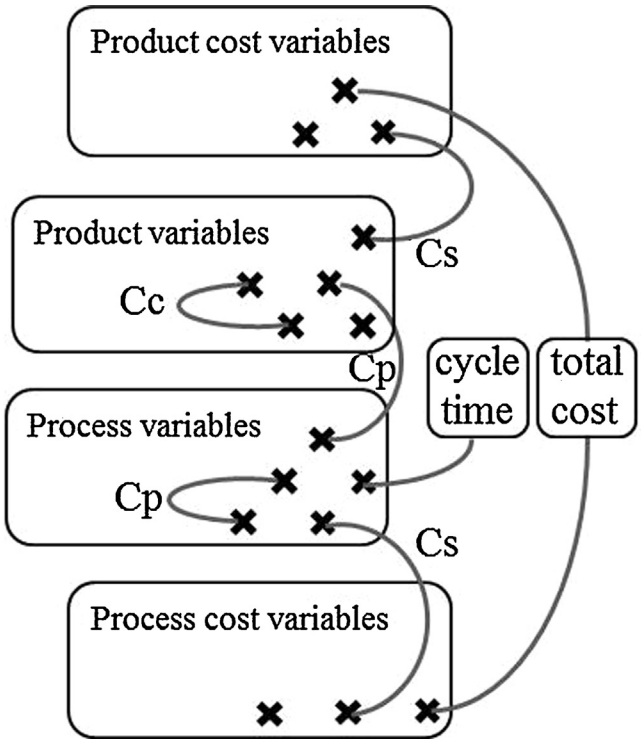
3. Optimization problem and evolutionary algorithm proposals

This section begins with a definition of the optimization problem. This is followed by a literature review of constrained evolutionary approaches. This bibliography study enables us to select an EA approach (feasibility rule in our case) that will be used for comparisons during the experiments. The last section will present the three optimization techniques: CFB-EA, FRB-EA and BB.

3.1. Definition of the optimization problem

As explained previously, the first step leads to the reduction of the initial feasible space (left graph of Fig. 1) to a restrained area (center graph of Fig. 1). This corresponds to the filtering of the customer's non-negotiable requirements. The filtering system provides domain bounds for every criteria variable (minimal and maximal values for total cost and cycle time). The restrained area contains solutions corresponding to the different remaining decisions to be taken. But this area also contains unfeasible solutions, due to the constraints of the problem. The aim of the optimization process is to find a selection of solutions close to the Optimal Pareto front (right graph of Fig. 1). The problem of the example of Fig. 3 is generalized as the one shown in Fig. 5.

The constrained optimization problem (O-CSP) is defined by the quadruplet $\langle V, D, C, f\rangle$ where V is the set of decision variables, D the set of domains linked to the variables of V, C the set of constraints on variables of V and f the multi-valuated fitness function. Here, the aim is to minimize both total cost and cycle time. The set V gathers: the product variables and the process variables resource (we assume that duration is deduced from product and resource). The set C gathers: only configuration constraints (C_c) and process constraints (C_p). The variables operation durations and cycle time are linked with a numerical constraint that does not impact solution definition and therefore does not belong to V and C. The same applies to the product/process cost variables and total cost, which are linked with cost constraints (C_s) and total cost constraints. The filtering system allows dynamically updating of the domain of all these variables with respect to the constraints. The variables belonging to V are all symbolic or at least discrete. Duration and cost variables are numerical and continuous. Therefore, constraints are discrete (C_c), numerical (cycle time and total cost) or mixed (C_p and C_s). Discrete constraints filtering is



processed using a conventional arc consistency technique (see [13]) while numerical constraints are processed using bound consistency (see [25]).

3.2. Overview of constrained optimization approaches

In their original versions, EAs were designed to deal with large unconstrained search spaces [26]. Therefore, they need to be adapted to take into account the constraints of the problem. Many research studies try to integrate constraints in EA. C. Coello Coello proposes a wide state-of-the-art study of these methods on (C. Coello Coello web site [44]) and a synthetic overview in [27]. In this part, we summarize the current tendencies in the resolution of constrained optimization problems using EAs. Six kinds of methods deal with this problem: penalty functions (PF), stochastic ranking (SR), ε-constrained (EC), multi-objective concepts (MO), feasibility rules (FRs) and special operators (SO).

PF: Penalty functions are the traditional way to integrate violation of constraints in an objective function (deb 2000). For each individual, its fitness value is calculated according to the level of constraints violation. The aim is to decrease the fitness of unfeasible solutions in order to favor the selection of feasible solutions. The main drawback of such an approach is that it requires a careful fine-tuning of the weights (penalty factors) needed to aggregate the violation of different constraints and these values are highly problem-dependent [28]. Even if some improvements have been proposed (for example: dynamic, adaptive, co-evolved, fuzzy-adapted), penalty functions are, without extremely careful fine-tuning, highly sensitive to premature convergence due to local optima.

SR: Stochastic ranking [28] was designed to deal with the shortcomings of penalty functions (over-and under-penalization due to unsuitable values for the penalty factors). SR uses a bubble-sort-like process to rank the solutions in the population and an additional parameter that allows a pseudo-random selection of some unfeasible solutions to build the next generation. SR is very competitive while requiring a low number of fitness-function evaluations, but it needs a ranking process.

EC: The ε-constrained method [29], uses a relaxation mechanism that considers as feasible any solution with a sum of constraint violations under the limit ε. The minimization of the sum of constraint violations precedes the minimization of the objective function. In this approach and its spin-offs, such as the α-constrained method [30], the careful fine-tuning of α and ε remains the main shortcoming.

MO: Recently, approaches using multi-objective concepts have been popular in the literature. They consider constraint violation as a single objective function. Gong and Cai [31] merges the original objective function with each constraint and gets as much objectives as constraints. Ray et al. [32] considers two objective functions: the original objective function and the sum of constraint violation. The first method is limited to problems with a low number of constraints, due to the lack of performance of the multi-objective algorithm. The difficulty in the second method is to elaborate an effective algorithm that preserves the diversity of individuals.

FRs: Feasibility rules were originally proposed for selection in binary tournaments in [33]. It consists in three rules that allow classification of the solution: (i) when comparing two feasible solutions, the one with the best fitness function is selected; (ii) when comparing a feasible and an unfeasible solution, the feasible one is selected; (iii) when comparing two unfeasible solutions, the one with the lowest sum of constraint violations is chosen. The absence of user-defined parameters is one of its main advantages. However, it may also lead to premature convergence [34]. Nowadays, it is a very popular constraints-handling approach because of its simplicity and ability to be coupled to any sort of EA.

SO: The special operators class groups together methods that try to deal only with feasible individuals, such as repairing methods, preservation of feasibility methods [35] or operator, which move solutions within a specific region of interest inside the search space, as, for example, the boundaries of the feasible region [36]. Generally, these methods are known to be efficient on non-over-constrained problems (i.e. a feasible solution can be obtained in a reasonable amount of time to be able to generate a population of solutions). Our CFB-EA approach belongs to this family and aims at preserving the feasibility of the individuals during their construction. Kowalczyk in [37] proposed initial ideas for using constraint consistency to prevent inconsistent individuals. But their paper did not provide any details or experimentation. We adopt this idea and focus on specific evolutionary operators that prune search space using constraint filtering. The main difference between our approach and others previously developed is that we do not have any unfeasible solutions in our population or archive, while other approaches try to exploit the information contained in unfeasible solutions to improve optimization.

By looking for the most adequate concurrent optimization approach, we have come to the following conclusions: (i) with regard to penalty functions: after some unsuccessful tests and because of our preference of a minimal intervention of the expert for the tuning of the parameters, we did not retain this kind of method for our study, (ii) with regard to multi-objective functions: as our problem deals with multiple constraints, we did not retain this kind of method, mainly for performance reasons, (iii) with regard to stochastic ranking and ε-constrained methods: they are rather close to feasibility rules, but as they both require an extra probability parameter, we did not consider them and put feasibility rules in our short list, (iv) with regard to special operators: we think that it is better to avoid inconsistent individuals (as CFB-EA does) rather than trying to repair them, thus we did not adopt this approach, (v) with regard to the feasibility rule: with a low number of parameters to tune and a behavior close to penalty functions, we decided to consider this approach for our experiments. The following sections will therefore deal with the three optimization techniques: constraint-filtering-based approach (CFB-EA), branch and bound (BB) and feasibility-rule approach (FRB-EA). Next section describes them briefly.

3.3. Description of the optimization algorithms used for experimentation

3.3.1. Constraint-filtering-based evolutionary algorithm (CFB-EA)

The proposed evolutionary algorithm is based on SPEA2 [38] with an added constraints filtering process that avoids unfeasible individuals (or solutions) in the archive. SPEA2 is a useful Pareto-based method founded on the preservation of a selection of best solutions in a separate archive (archive on generation t is noted At and population Pt). Six parameters are required: size of archive (Narch), size of population (Npop), number of generation (T) (or any stopping criterion), probabilities for crossover (P_{cross} for individual selection, crossover of approximately half of the chromosome) and mutation (P_{mut} for individual selection/P_{mutg} for gene selection) operators. The archive size mainly impacts the diversity of solutions; while population size defines how many solutions are generated at each generation. This provides the following six-step approach:

1. Initialization of individual set (P_0) that respects the constraints (thanks to filtering),
2. Fitness assignment (balance of Pareto dominance and solution density),
3. Selection of individuals and archive update (A_t),

4. Stopping criterion test (time, nb. of generations or performance),
5. Individuals selection for crossover and mutation operators (binary tournaments),
6. Individuals crossover and mutation that respect constraints (thanks to filtering) to generate next generation (P_t)
7. Return to step 2.

For initialization, crossover and mutation operators, each time an individual is created or modified, every gene (decision variable of V) is randomly instantiated into its current domain. To avoid the generation of unfeasible individuals, the domain of every remaining gene is updated by constraint filtering. As filtering is not fool proof, inconsistent individuals can be generated. In this case, a limited backtrack process is launched to solve the problem. For full details please see [7]. Notice that we have designed the evolutionary operator so as to preserve diversity (random selection of gene and state); thus, if an individual become unfeasible – and in order to preserve individual consistency – we backtrack on previous choices until an individual becomes feasible again. Our approach also has the advantage of not needing any additional parameter tuning for constraint handling.

3.3.2. Branch-and-bound procedure using filtering system (BB)

The key idea of the branch and bound algorithm is to explore a search tree, but using a cutting procedure that stops exploration of a branch when a better branch has already been found. The first tool needed is a splitting procedure that corresponds to the selection of one variable of the problem and to the instantiation of this variable with each possible value. The second tool is a node-bound evaluation procedure. The filtering process is used to achieve this task with a partial instantiation and is able to evaluate if the partial instantiation is consistent with the constraints of the problem. If this is the case, it can provide the lower bound of each criterion cycle time and cost. When the search reaches a leaf of the search tree, or complete instantiation, the filtering system gives the exact evaluation of the solution. Thus, the values of leaf solutions can be used to compute the current Pareto front and then to cut remaining unexplored branches that are dominated by any aspect of the Pareto front solution (e.g. the upper bounds of the leaf solution dominate the minimal bounds of the branch to cut). This provides the following steps:

1. Select a remaining un-instantiated variable and split on each possible value.
2. Evaluate each value with constraints filtering system.
3. Cutting of every Pareto-dominated branch by a complete solution that has already been found.
4. Return to step 1 on the lowest node value until there are no more unexplored branches.

3.3.3. Feasibility–rules-based evolutionary algorithm (FRB-EA)

Two possible implementations of feasibility rules (FRs) in an EA can be investigated. The original one was to incorporate FRs in the selection process. In order to avoid unfeasible individuals, we prefer to use feasibility rules while comparing pair-wise solutions during the fitness assignment. In the SPEA2 method, the Pareto dominance concept is used to compare and evaluate solutions. We added FRs in this Pareto dominance in the following way: (i) when comparing two feasible solutions, the classic Pareto dominance relation is used; (ii) when comparing a feasible and an unfeasible solution, the feasible one will dominate; (iii) when comparing two unfeasible solutions, the one with the lowest sum of constraint violation will dominate. This leads to the following dominance comparison algorithm:

1. If one solution of the two solutions compared is unfeasible, it is dominated.
2. If both are unfeasible, the one with the highest sum of constraint violation is dominated.
3. If both are feasible, the classic Pareto dominance algorithm defines which one is dominated.

This method is simple to implement and allows for comparative study. Furthermore, some promising improvements have been proposed for a FR-EA as a distribution-based stopping criterion [39], or for various diversity preservation mechanisms [40,41]. These proposals highlight the interest of this approach.

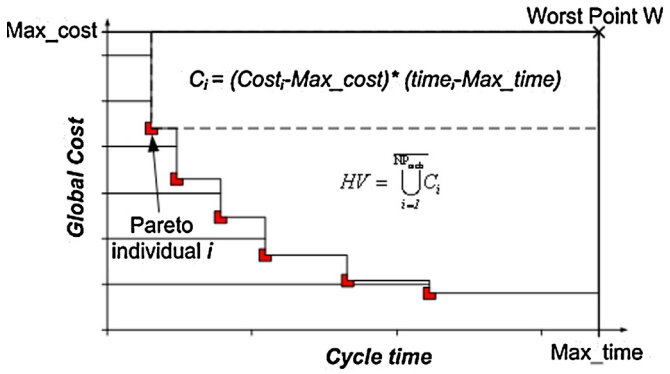
4. Experimental results and discussions

The aim of this section is to compare concepts that allow integration of constraints handling in an evolutionary optimization process (filtering vs. feasibility rules) in the context of our two-step aiding process. A comparison with an exact approach (BB) is also made in order to estimate the limit where the exact approach should be replaced by evolutionary computations. We want to define the limits of each approach, according to the size of the problem, its constraints level and the difficulty of parameter setting. The size of the problem to solve varies greatly according to the number of choices made in the first step of the proposed aiding approach. The constraint level of the model is also a key point when selecting an optimization technique. It regulates the ratio between feasible and unfeasible space. Finally, we assume that the user is not an expert in optimization techniques, so the difficulty of tuning optimization parameters is essential in our context.

The optimization algorithms were implemented in C++ programming language and interacted with the filtering system coded in Perl language. All tests were done using a laptop computer powered by an Intel core i5 CPU (2.27 GHz, only one CPU core is used) and using 2.8 Go of ram.

For a multi-objective aiding problem, the user expects an efficient and diversified set of solutions in a reasonable lapse of time. To evaluate the algorithm results, we used the hypervolume metric defined by Zitzler [42] and illustrated in Fig. 6. It measures the hypervolume of space dominated by a set of solutions. It thus allows both convergence and diversity proprieties to be evaluated (the fittest and most diversified set of solutions is the one that maximizes hypervolume).

In first section, we study the models used for experimentation. Then, we compare the three approaches (BB, FR-EA and CFB-EA) on various small instances. In a third section, the results of FRB-EA and CFB-EA on two sizes of model (small and large) and for three constraints levels are compared. Finally, these results are discussed in the last experimental section.



4.1. Models used for experimentation

Every model used in these experiments derives from a large real-world-like model named "full_aircraft" corresponding with a small commercial aircraft (between 4 and 12 passengers). This model has been established during the four years project ATLAS project funded by the French National Research Agency (ANR) whose objective was to study interactions between product design and project planning in aerospace industry within the Aerospace Valley industry cluster (http://www.aerospace-valley.com/). Some aspects of this model can be consulted in [7] and [6].

The model used in this paper groups 92 variables (symbolic, integer or float variables) linked by 67 constraints (compatibility tables, equations or inequalities). Among these variables, we find 21 decision variables that will be manipulated by the optimization algorithms (chromosome in EA). Twelve variables, with an average of six possible values for each, describe the main product aspects. In order to have a general idea, these variables are: SN (seat quantity), AN (armchair quantity), SC (seat comfort level), AC (armchair comfort level), FA (cabin finish level), FO (cockpit finish level), OE (on-board electronics), FE (flight electronics), CZ (cargo zone size), CS (cruising speed), FR (flight range), EN (engine category). Nine variables, with an average of nine possible values for each (in fact, the combination of 3 resource types and 3 resource quantities), describe the nine production operations that compose the production process. These operations are: Op11 (structure sourcing), Op12 (components sourcing), Op21 (structure manufacturing), Op22 (components assembling), Op31 (structure assembling), Op41 (cabin assembling), Op51 (body painting), Op61 (final testing), Op71 (delivery).

When constraints are not considered, this leads to a number of possible combinations (around $10^{18} \approx 6^{12} \times 9^{9}$ for "full_aircraft" model). For a first illustration, Fig. 7 shows a random sampling of this model (solutions are small dark points) and the Pareto front obtained with one run of CFB-EA (gray rhombuses). This figure also presents a sampling of a model named "red4", based on a reduction of the initial "full_aircraft", in order to simulate a reduction made by the user in the first step of our approach and to test our algorithm on a small instance.

According to the choices made during the first step of the proposed aiding procedure, the size of the optimization problem can vary considerably. We first evaluate the size where an exact algorithm, BB in our case, could be used. To modulate the size of the model, we have simulated five sets of user non-negotiable requirements that instantiate various variables on the full aircraft problem that provide five models to optimize (named red1 to red5 according to growing size). In model red1, on the product model size: 6 product variables have been instantiated and the 6 others have been reduced from 6 to 3 possible values (unconstrained combinatory is around $729 \approx 1^{6} \times 3^{6}$); on the process model size: 3 process variables have been instantiated and the 6 others have been reduced from 9 to 3 possible values (unconstrained combinatory is around $729 \approx 1^{3} \times 3^{6}$). The combinatory of the unconstrained model red1 is therefore around $0.53 \times 10^{6} \approx 729 \times 729$. For the other models (red2 to red5), the process model is unchanged and the product model size is increased thanks to modifications in the number of instantiated variables (6–2) and the number of variables with 3 possible values (6–10), as follows: red 1 ($1^{6} \times 3^{6}$), red2 ($1^{5} \times 3^{7}$), red3 ($1^{4} \times 3^{8}$), red4 ($1^{3} \times 3^{9}$), red5 ($1^{2} \times 3^{10}$). Relevant unconstrained combinatory are shown in the Table 1.

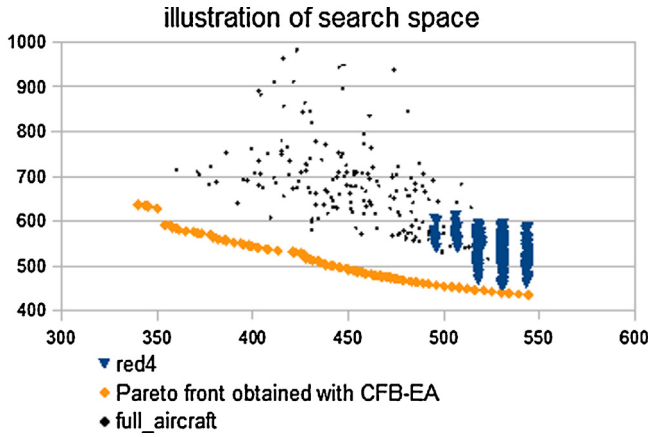


Fig. 7. Search space and illustration of a Pareto front obtained with CFB-EA.

The second goal is to investigate the impact of the constraint level for EA algorithms. We therefore generated three models derived from the model red5 with three constraint levels, by modulating the number of allowed combinations in the constraints belonging to the set C (configuration constraints Cc and process constraints Cp). These models, named red5_MC1 to red5_MC4, have a constraint level (ratio constrained combinatory/unconstrained combinatory) of around: 10% for red5_MC1, 5% for red5_MC2, 2% for red5_MC3 and 0.5% for red5_MC4, as shown in Table 1. These ratios and the estimated number of solutions were obtained with a random sampling of every model (average for one thousand individuals randomly generated).

Table 1 recapitulates the considered models. Without taking into account constraints, the combinatory goes from half a million to 43 million for small models (red1 to red5) and is around 10^{18} for the "full_aircraft" model. The presence of constraints leads to a lot of unfeasible solutions and Table 1 illustrates how constraints reduce the number of feasible solutions. The hypervolume obtained with the BB is also shown (unknown for full_aircraft models).

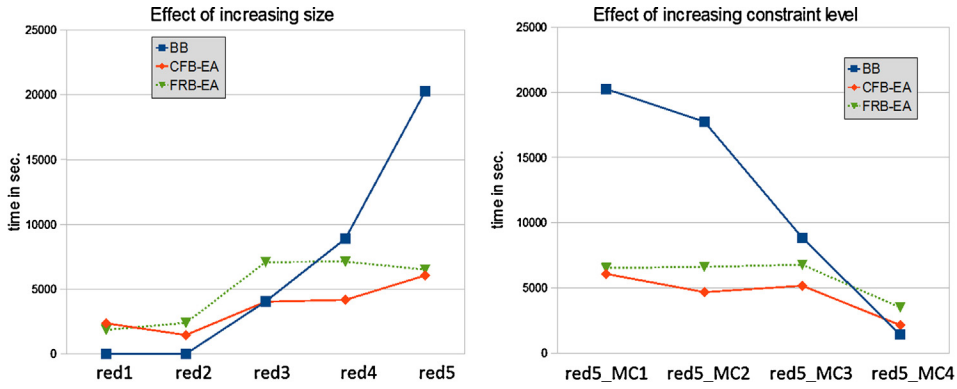


Fig. 8. Experimentations with different model sizes and constraints levels.

Table 2
Comparison CFB-EA, FRB-EA and BB on small models.

Model	Optimal HV	BB		CFB-EA			FRB-EA		
		Total time	Time to reach optimal	Average time	Time std_dev	Average HV	Average time	Time std_dev	Average HV
red1	1,23,576	16	14	2379	0.63	1,23,576	1874.1	0.40	1,23,576
red2	1,29,782	797	19	1476	0.38	1,29,782	2411.2	0.53	1,29,736
red3	1,41,914	7836	4063	4048	0.44	1,41,914	7107.7	0.29	1,41,655
red4	1,54,383	17,612	8908	4186	0.33	1,54,383	7141.6	0.49	1,53,380
red5	1,67,189	42,103	20,257	6069	0.40	167,189	6522	0.36	1,67,189

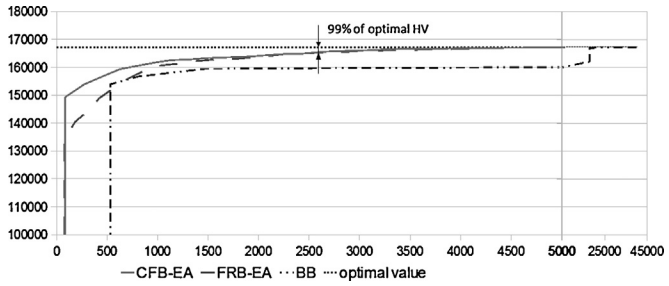
Table 3
Comparison CFB-EA, FRB-EA and BB according to constraints level.

Model	Optimal HV	BB		CFB-EA			FRB-EA		
		Total time	Time to reach optimal	Average time	Time std_dev	Average HV	Average time	Time std_dev	Aver. HV
red5_MC1	1,67,189	42,103	20,257	6069	0.40	1,67,189	6522	0.36	1,67,189
red5_MC2	1,67,189	36,898	17,753	4674	0.26	1,67,189	6613	0.46	1,67,162
red5_MC3	1,67,189	17,708	8846	5162	0.22	1,67,189	6775	0.31	1,66,415
red5_MC4	1,38,714	7034	1407	2134	0.32	1,38,714	3501	0.23	138,714

4.2. A comparison with an exact approach on small size models

As already said, a first goal of these experiments is to compare the behavior of the EA methods with an exact approach for small instances, BB in our case. The curves in Fig. 8 and the corresponding values given in Tables 2 and 3 illustrate the results of the three algorithms when the number of variables (red1 to red5 on left part on Fig. 8 and Table 2) or the constraint level (red5_MC1 to red5_MC4 on right part of Fig. 8 and Table 3) increase. The vertical axis corresponds to the time spent (i) reaching optimal Pareto Front with BB algorithm or (ii) reaching optimal Pareto Front with CFB-EA and FRB-EA.

Notice that while the times shown for the BB in Fig. 8 and in Table 2 and 3 are the times needed to reach the optimal, the BB did not stop at this moment. It had to cut every remaining branch to be sure that it reached optimal, and this can take a significant time. Curves in Fig. 9 and corresponding values summarized in Table 2 illustrate the evolution of the hypervolume dominated by the Pareto front solutions (the horizontal axis corresponds to the time in seconds) for each approach investigated on model red5 (or red5_MC1). The curves representing an EA performance are average results for ten executions. The evolutionary settings used to obtain these results were: Population size: 80, Archive size: 100, P_{mut}: 0.3, P_{mutg}: 0.2, P_{cross}: 0.8. The ending criterion used is to reach the optimal HV value (previously computed using the BB algorithm) or a stagnation of the HV for more than 2 h.



Clearly, parameter settings are one of the main drawbacks for an efficient use of evolutionary algorithms. The setting of population and archive sizes is a compromise between speed convergence and final performance. Indeed, a small population and archive size leads to a quick improvement in solution quality but it also increases the time needed to reach the optimal Pareto front. Whereas larger population and archive sizes reduce the convergence speed but allow the optimal Pareto front to be found quickly.

For the small models (left part of Fig. 8), the BB algorithm reaches optimal Pareto front much faster compared with EA performance (few seconds on red1 and red2 models against some thousand seconds for EAs). On the other hand, EAs greatly outperform the BB algorithm on larger models (time needed to reach optimal HV: 6069 s for CFB-EA, 6522 s for FRB-EA and 20257 s for BB). CFB-EA and FRB-EA, after a rapid improvement of

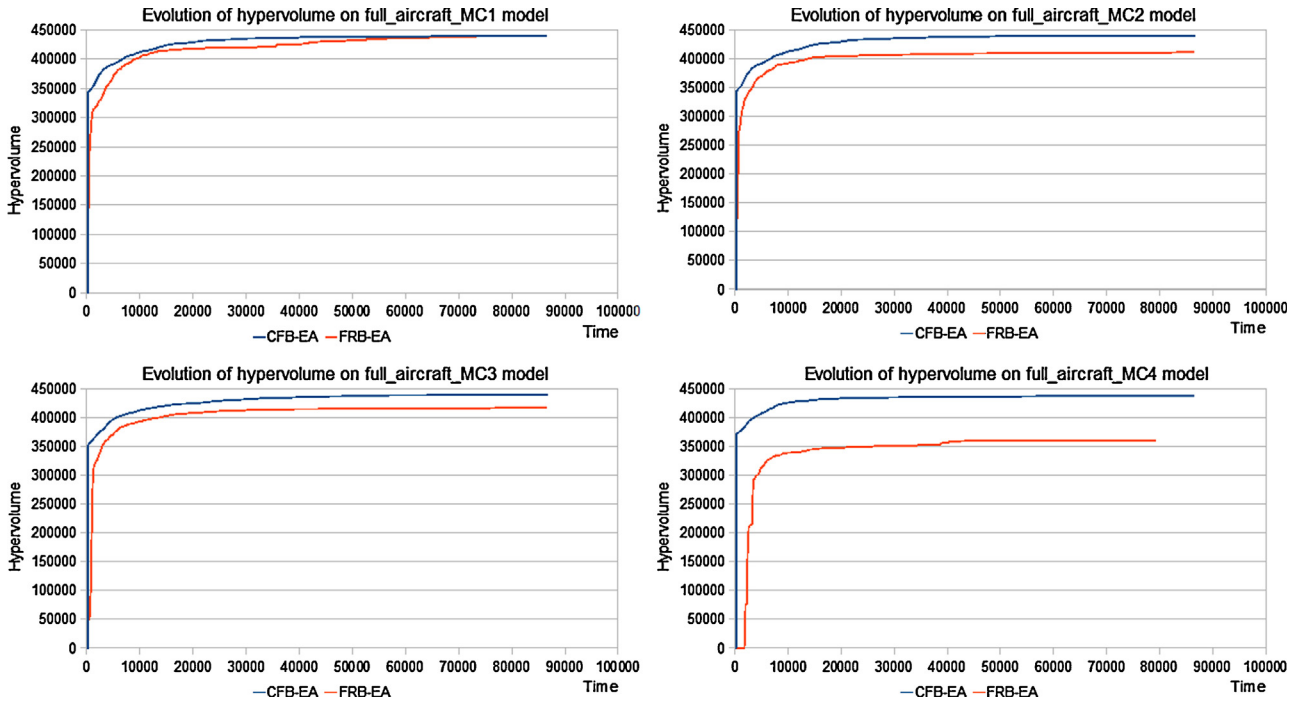


Fig. 10. Evolution of hypervolume with CFB-EA and FRB-EA on full models.

hypervolume (for example on model red5, 99% of optimal HV is reached in 2600 s for CFB-EA and 2830 s for FRB-EA), stagnate and converge very slowly to the optimal Pareto front (less than 0.5% of improvement on the last 3000 s). Indeed, this is one of the weak points of classic EA approaches. A mimetic algorithm (coupling of an EA with a local search approach) is actually prospected to improve this aspect.

As expected, these tests show that BB algorithm must be reserved for small size models where it is more accurate than evolutionary approaches. But this report must be moderated according to the constraint level as showed on right part of Fig. 8. When the problem is more constrained, BB reaches faster the optimal Pareto front, Table 3 shows values obtained on these models. This is because the filtering allows more branches to be cut on the search tree, in such way that the algorithm reaches leaf solutions and, consequently, optimal solutions more quickly. The EAs performance is more stable according to this criterion except for the more constrained model (red5_MC4). This behavior is probably links to the fact that this model has a very limited number of possible solutions and an optimal HV easier to reach.

A first conclusion of this section is that the proposed EA algorithms are clearly better than the BB algorithm when the size of the problem increases. A second conclusion is that an increase in the constraint level tends to very slightly limit the tendency of the previous conclusion. A last important conclusion with respect to our problem, EAs are much less sensible to modifications of problem size and constraint level than BB algorithm. Therefore, in the next section, we consider only the large model named full_aircraft gathering thirty variables with various level of constraint.

4.3. Comparison EA on a real-world-like problem

The goal of this section is to compare on larger problems only the two EA algorithms as BB is unsuitable on this scale. They are evaluated on the full_aircraft model and on three models derivate from this one with various constraint levels. Curves on Fig. 10 and corresponding values of Table 4 illustrate the results of the two algorithms when the constraint level increases (MC1 lowest constraint level up to MC-4 highest one). The vertical axis corresponds to the hypervolume reach by each algorithm. The curves are average results for ten executions of each EA. Evolutionary settings used to obtain these results: population size: 80, archive size: 100, P_{mut}: 0.3, P_{mutg}: 0.2 for CFB-EA and 0.1 for FRB-EA, P_{cross}: 0.8. The ending criterion used is a strict time limit of one day (86,400 s).

These tests show that CFB-EA and FRB-EA belong to the same range of performance. On the full_aircraft model (full_aircraft_MC1), difference with FRB-EA is mainly on convergence speed and quality of the Pareto Front. With CFB-EA, hypervolume increases quickly and constantly (99% of final hypervolume is reached in 30,000 s against 54,000 s for FRB-EA). Those characteristics, especially the diversity of individuals, are promising for a future hybridization with a local search. When the constraints level increases (from around 10% of feasible solutions to less than 0.1%), a gap between final performance of both EA can be seen. On the more constrained model (full_aircraft_MC4), CFB-EA hypervolume is around 15% better than the one obtained with FRB-EA. In fact CFB-EA and FRB-EA are clearly two different ways to take into account constraints in evolutionary optimization process. Table 5 gathers various indicators that allow comprehension of the behavior of each approach. The main difference is the utilization of the computing time.

In CFB-EA, around 98–99% of the computing time is spent to both construct and evaluate solutions using filtering. Indeed, as we manipulate only feasible solutions, lots of backtracking and

Table 5
Precision on behavior of EA approaches.

Model	CFB-EA		
	Average duration of a generation	Total number of solutions generated	% of time spend to generate and evaluate solutions
full_aircraft	570	14,577	98
full_aircraft_MC2	604	13,703	99
full_aircraft_MC3	504	16,439	98
full_aircraft_MC4	624	13,269	99
Model	FRB-EA		
	Average duration of a generation	Total number of solutions generated	% of time spend to evaluate solutions
full_aircraft	229	36,044	88
full_aircraft_MC2	240	34,314	88
full_aircraft_MC3	262	31,447	89
full_aircraft_MC4	220	34,440	86

filtering are needed to obtain feasible solutions especially during crossover operation (around 20,000 backtracks and 4,00,000 filtering needed to generate around 15,000 solutions). That leads to long generations (around 600 s for each generation). In FRB-EA, only evaluation of solution thanks to filtering algorithm represents around 88% of the computing time. As there is no backtracking (unfeasible solution are kept on evolutionary process), FRB-EA generates more than twice solutions compared to CFB-EA, but more than half of these solutions are unfeasible.

In the FRB-EA implemented in our work, those solutions are most of the time unused in the evolutionary process; because during the selection process, if an unfeasible solution is compared to a feasible one, we systematically keep the feasible one. Consequently, the time spent to generate those solutions is lost without any benefit for the optimization process. That is also why the hypervolume increases slower in this approach. This aspect is already known by authors of feasibility rules principles: if no further mechanisms are adopted to preserve diversity (particularly paying attention to the need to keep infeasible solutions in the population), this approach will significantly increase the selection pressure [43].

To conclude this comparison, it seems that proposed CFB-EA approach is relevant for the range of concurrent configuration and planning problems addressed in this article (size and constraints level). CFB-EA enables to propose, in a reasonable amount of time, a set of solutions that permits the user to decide about his own cost/cycle time compromise. A last operational advantage lies in the fact that CFB-EA is robust with regard to evolutionary parameters setting. We have tried various parameters settings with both EAs, and CFB-EA was really constant in its performance compared to FRB-EA.

5. Conclusions

The goal of this article has been to propose a detailed evaluation of an evolutionary algorithm, called CFB-EA for Constraint filtering based evolutionary algorithm, that deals with concurrent product configuration and production planning. The problem has been presented and modeled as a constraint satisfaction problem, and then a two-steps approach, gathering an interactive configuration and planning process followed by a multi-criteria optimization, has been presented with a simple example. Once the optimization problem highlighted, a detail survey of evolutionary algorithms that handle constrained problem permits us to identify the most suitable competing optimizing approach (FRB-EA for feasibility rule based evolutionary algorithm). An exact branch and bound procedure (BB) is also recalled for small instances.

For small instances, from 12 to 15 decision variables with 3 values for each (0.2–2.10^6 solutions), BB is globally better than EA approaches. Logically the proposed EA works better when the size of the problem gets larger compared to BB, however the tendency goes to the opposite when the problem tends to be more constrained. As the size of the optimization problem is directly dependent of the quantity of negotiable requirements (first step interactive configuration/planning), an interesting result is that it is possible to propose a kind of limit that can trigger the selection of BB or EA optimization. Of course this limit is specific to the addressed problem. For these small instances, it must also be pointed out that FRB-EA and CFB-EA have similar performances (much less than 2 h) with a very low sensitivity to problem sizes and constraint levels.

When the problem gets larger, BB cannot be considered. On a problem of 21 decision variables (12 product variables with 6 values and 9 process variables with 9 values), when the constraint level is low (solution space between 10^{15} and 10^{17}), CFB-EA and FRB-EA perform very closely, when the constraint level increases (solution space between 10^{13} and 10^{14}) CFB-EA is a little better. In terms of convergence speed, CFB-EA reaches around 90% of the hypervolume in less than 3 h and 99% in less than 10 h. The low sensitivity of CFB-EA with respect to constraint level can be also noticed.

That leads us to consider that the CFB-EA approach is competitive for this kind of configuration/planning problems even if some further improvements could be investigated for both EA approaches. Therefore the two steps process object of this paper can be considered with no doubt, as a significant assistance for optimal configuration and planning achievement. It allows the user to decide efficiently about his cost/cycle-time compromise when dealing simultaneously with configuration and planning.

These promising results introduce some prospective studies: convergence speed, evolutionary parameter tuning and also problems with more than two objectives. For convergence speed or larger problems, we are currently developing an iterative optimization process that aims to reduce considerably the time required to obtain a near-optimal Pareto front using a kind of zoom on a specific area selected by the user during the optimization process. For parameters tuning, we are thinking of considering the possibility of an automated setting with a variable population and archive size. Finally, configuration and planning problems taking into account several objectives such as performance, risk, or sustainable aspects are in our short list.

References

[1] S. Mittal, F. Frayman, Towards a generic model of configuration tasks, in: Proceedings of IJCAI, 1989, pp. 1395–1401.
[2] T. Soininen, J. Tiihonen, T. Männistö, R. Sulonen, Towards a general ontology of configuration, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12 (4) (1998) 357–372.
[3] K. Schierholt, Process configuration: combining the principles of product configuration and process planning, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 15 (5) (2001) 411–424.

[4] P.T. Helo, Q.L. Xu, S.J. Kyllonen, R.J. Jiao, Integrated vehicle configuration system connecting the domains of mass customization, Computers in Industry 61 (1) (2010) 44–52.
[5] D. Steward, D. Tate, Integration of axiomatic design and project planning, in: Proceedings of first International Conference on Axiomatic Design, Cambridge, USA, (2000), pp. 285–289.
[6] L.L. Zhang, E. Vareilles, M. Aldanondo, Generic bill of functions, materials, and operations for SAP2 configuration, International Journal of Production Research 51 (2) (2013) 465–478.
[7] P. Pitiot, M. Aldanondo, E. Vareilles, P. Gaborit, M. Djefel, S. Carbonnel, Concurrent product configuration and process planning, towards an approach combining interactivity and optimality, International Journal of Production Research 51 (2) (2013) 524–541.
[8] D. Mailharro, A classification and constraint-based framework for configuration, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12 (4) (1998) 383–397.
[9] U. Junker, Chapter 24, Configuration, in: Handbook of Constraint Programming, Elsevier, 2006, pp. 835–875.
[10] R. Barták, M. Salido, F. Rossi, Constraint satisfaction techniques in planning and scheduling, Journal of Intelligent Manufacturing 21 (1) (2010) 5–15.
[11] M. Koubarakis, Chapter 19, Temporal CSPs, in: Handbook of Constraint Programming, Elsevier, 2006, pp. 663–697.
[12] M. Aldanondo, E. Vareilles, M. Djefel, Towards an association of product configuration with production planning, International Journal of Mass Customisation 3 (4) (2010) 316–332.
[13] C. Bessiere, Chapter 3, Constraint propagation, in: Handbook of Constraint Programming, Elsevier, 2006, pp. 29–70.
[14] F. Benhamou, L. Granvilliers, Chapter 16, Continuous and interval constraints, in: Handbook of Constraint Programming, Elsevier, 2006, pp. 569–603.
[15] M. Aldanondo, E. Vareilles, K. Hadj-Hamou, P. Gaborit, Aiding design with constraints: an extension of quad tree in order to deal with piecewise functions, International Journal of Computer Integrated Manufacturing 21 (4) (2008) 353–365.
[16] S.C. Brailsford, C.N. Potts, B.M. Smith, Constraint satisfaction problems: algorithms and applications, European Journal of Operational Research 119 (1999) 557–581.
[17] J. Amilhastre, H. Fargier, P. Marquis, Consistency restoration and explanations in dynamic CSPs – application to configuration, Artificial Intelligence 135 (2002) 199–234.
[18] L. Li, L. Chen, Z. Huang, Y. Zhong, Product configuration optimization using a multiobjective GA, International Journal of Advanced Manufacturing Technology 30 (2006) 20–29.
[19] H. Missbauer, R. Uzsoy, Optimization models of production planning problems, Planning Production and Inventories in the Extended Enterprise, vol. 151, Springer, 2011, pp. 437–507.
[20] E. Vareilles, M. Aldanondo, P. Gaborit, Evaluation and design: a knowledge based approach, International Journal of Computer Integrated Manufacturing 20 (7) (2007) 639–653.
[21] L. Hvam, S. Pape, M.K. Nielsen, Improving the quotation process with product configuration, Computers in Industry 57 (7) (2006) 607–621.
[22] M. Kopisch, A. Günter, Configuration of a passenger aircraft cabin based on conceptual hierarchy, constraints and flexible control, Lecture Notes in Computer Science 604 (1992) 421–430.
[23] A. Falkner, G. Fleischanderl, Configuration requirements from railway interlocking stations, in: Proceeding of the IJCAI Configuration Workshop, Seattle, USA, 2001.
[24] D. Sabin, R. Weigel, Product configuration frameworks – a survey, IEEE Intelligent Systems 13 (4) (1998) 42–49.
[25] O. Lhomme, Consistency techniques for numerical CSPs, in: Proceedings of IJCAI, 1993, pp. 232–238.
[26] T. Bäck, Evolutionary Algorithms in Theory and Practice, Oxford University Press, New York, 1996.
[27] E. Mezura-Montes, C. Coello Coello, Constraint-handling in nature-inspired numerical optimization: past, present and future, Swarm and Evolutionary Computation 1 (4) (2011) 173–194.
[28] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 4 (3) (2000) 284–294.
[29] T. Takahama, S. Sakai, N. Iwane, Constrained optimization by the epsilon constrained hybrid algorithm of particle swarm optimization and genetic algorithm, Lecture Notes in Artificial Intelligence 3809 (2005) 389–400.
[30] T. Takahama, S. Sakai, Constrained optimization by alpha-constrained genetic algorithm (alpha GA), Systems and Computers in Japan (35) (2004) 11–22.
[31] W. Gong, Z. Cai, A multiobjective differential evolution algorithm for constrained optimization, in: Proceedings of IEEE Congress on Evolutionary of Congress on Evolutionary Computation (CEC'2008), Hong Kong, (2008), pp. 181–188.
[32] T. Ray, H.K. Singh, A. Isaacs, W. Smith, Feasibility driven evolutionary algorithm for constrained optimization, in: Constraint-Handling in Evolutionary Optimization, vol. 198, Studies in Computational Intelligence Series, 2009, 145–165.
[33] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering (186) (2000) 311–338.
[34] E. Mezura-Montes, C.A. Coello Coello, E.I. Tun-Morales, Simple feasibility rules and differential evolution for constrained optimization, in: Proceedings of the 3rd Mexican International Conference on Artificial Intelligence (MICAI'2004), Lecture Notes in Artificial Intelligence No. 2972, 2004, pp. 707–716.
[35] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer-Verlag, 1996.
[36] M. Schoenauer, Z. Michalewicz, Evolutionary computation at the edge of feasibility, in: Proceedings of the 4th Parallel Problem Solving from Nature, Berlin, Lecture Notes in Computer Science 1141 (1996) 245–254.
[37] R. Kowalczyk, Constraint consistent genetic algorithms, in: Proceedings of IEEE Conference on Evolutionary Computation, Indianapolis, USA, (1997), pp. 343–348.
[38] E. Zitzler, M. Laumanns, L. Thiele, SPEA2. Improving the Strength Pareto Evolutionary Algorithm, Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, 2001.
[39] K. Zielinski, R. Laur, Stopping criteria for differential evolution in constrained single-objective optimization, in: Advances in Differential Evolution, Springer, 2008, pp. 111–138.
[40] J. Brest, V. Zumer, M.S. Maucec, Self-adaptative differential evolution algorithm in constrained real-parameter optimization, in: Proceedings of 2006 IEEE Congress on Evolutionary Computation (CEC'2006), Vancouver, BC, Canada, (2006), pp. 919–926.
[41] L. Cagnina, S. Esquivel, C. Coello-Coello, A bi-population PSO with a shake-mechanism for solving constrained numerical optimization, in: Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC'2007), Singapore, (2007), pp. 670–676.
[42] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms – a comparative case study, in: Proceedings of 5th International Conference on Parallel Problem Solving from Nature, Springer Verlag, 1998, pp. 292–301.
[43] E. Mezura-Montes, C.A. Coello Coello, A simple multimembered evolution strategy to solve constrained optimization problems, IEEE Transactions on Evolutionary Computation 9 (1) (2005) 1–17.
[44] C. Coello Coello, Comput. Sc. Dep CINVESTAV, http://www.cs.cinvestav.mx/~constraint/.
[45] G. Fleishanderl, G. Friedrich, A. Haselbock, H. Schreiner, M. Stumptner, Configuring large systems using generative constraint satisfaction, IEEE Intelligent Systems 13 (4) (1998) 59–68.
[46] G. Hong, D. Xue, Y. Tu, Rapid identification of the optimal product configuration and its parameters based on customer-centric product modeling for one-of-a-kind production, Computers in Industry 61 (3) (2010) 270–279.
[47] L. Hvam, Mass customisation of process plants, configuration, International Journal of Mass Customisation 1 (4) (2010) 445–462.