Concurrent product configuration and process planning, towards an approach combining interactivity and optimality

Paul Pitiot, Michel Aldanondo, Élise Vareilles, Paul Gaborit, Meriem Djefel, Sabine Carbonnel

Concurrent product configuration and process planning, towards an approach combining interactivity and optimality

Paul Pitiot[ab], Michel Aldanondo[a]*, Elise Vareilles[a], Paul Gaborit[a], Meriem Djefel[a] and Sabine Carbonnel[a]

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

In mass customisation, defining concurrently the configured product and the planning of the associated production process is a key issue in the customer/supplier relationship. Nevertheless, few studies propose supporting the decision-maker during the resolution of this significant problem. After studying the decision-maker's needs and problem characterisation (modelling and scale aspects), we propose in this paper a two-step approach with the aid of some tools. The first step allows the customer or internal requirements to be captured interactively with a constraint-based approach. However, this step does not lead to one single solution, e.g. there are many uninstantiated remaining decision variables. In this paper, we suggest adding an original optimisation step to complete this task. Thus, the contribution of the study is twofold: first, methodologically to define a new two-step approach that meets industrial needs; and second, to provide adapted tools especially for the optimisation step. The optimisation step, using a multi-criteria constrained evolutionary algorithm, allows the user to select their own cost/cycle time compromise among a set of Pareto optimised solutions. A conventional evolutionary algorithm is adapted and modified, with the inclusion of filtering processing, in order to avoid generating invalid solutions. Experimentations are described, and a comparison is made with a branch-and-bound approach that outlines the interest in the propositions.

Keywords: product configuration; process planning; constraint filtering; evolutionary algorithm; optimisation

1. Introduction

During negotiations with a customer for a customised product, the ability to offer an optimised set of solutions that meets customer requirements is a major challenge for an industrial decision-aiding system. This is especially the case when requirements include, concurrently, product aspects (components, options, performance, cost, functionalities . . .) and production process aspects (tasks, resources, cost, cycle time . . .). The goal of this paper is to present a two-step aiding system with the following features: first, the possibility – interactively and concurrently – to configure a product and to plan the production process involved to obtain this product; and second, to minimise conflicts between criteria such as performance, cost, and cycle time. Our aim is twofold: first, methodologically to define a new two-step approach that meets industrial needs (for interactivity and optimisation); and second, to provide effective software tools for this approach, especially for the optimisation step. An example dealing with the customisation of a light aircraft runs throughout the paper. This initial section introduces the problem and the organisation of the paper.

1.1 *Product configuration and production planning issues*

Product configuration can be described as defining a specific or customised product (through a set of properties, or bill of materials, etc.) from a generic product or a product family while respecting specific customer requirements (Mittal and Frayman 1989). Production planning can be considered as defining a specific production plan (a set of scheduled operations, resources to be used, quantity needed, etc.) from some kind of generic process plan while taking into account product characteristics and customer requirements (Barták *et al*. 2010). Although product configuration and production planning activities, along with the relevant supporting software, are known to be key elements in the development of mass customisation (Pine 1993), most of the research dealing with these two domains is strongly independent, and scientific results deal either with configuration or with planning. Consequently, the two

*Corresponding author. Email: michel.aldanondo@mines-albi.fr

activities are often considered in sequence: product configuration, then production planning. However, decisions relevant to each of these activities are closely dependent on the following:

- 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 provide tough constraints to product configuration (for example, a given assembly duration forbids the use of a particular kind of engine).

Therefore, iterations will become inevitable in order to fulfil customer requirements. For example, once a plane is fully configured or defined, planning can come up with a delivery time that is too late, requiring modifications in the configuration of the plane, thus causing iteration in the process. In order to avoid these time-consuming iterations and to be able to consider product and planning requirements concurrently, we propose associating them in a single problem. This would allow, for example, the two requirements: 'plane capacity = 12 seats' and 'delivery time = 4 months' to be considered before any others (speed, flight range, cost, etc.).

As the constraint satisfaction problem (CSP) framework has been successfully used for many configuration and planning studies (section 2), we will consider configuration and planning problems as two CSPs and will associate them. As we seek to provide interactive assistance, we shall need to compute, and show to the user, the consequences of each 'user's elementary requirement'. An elementary requirement is a restriction of the domain of a variable involved in configuration (for example 'plane capacity belongs to Clevenger *et al*. (2005) and Hvam *et al*. (2002)'), or in planning (for example 'due date is prior to 31/10/2011'). We shall consequently use the filtering capabilities of the CSP to reflect user requirements through the network of constraints and lead the user progressively towards a solution for both configuration and planning.

1.2 *Optimising product configuration and production planning*

In a similar way, optimisation work is very often carried out separately for configuration and planning. Given our problem, we must consider the optimisation of the complete problem of both configuring and planning. In this situation, various criteria can characterise a solution:

- on the product configuration side, we should consider product performance (speed, range . . .) and product cost;
- on the planning side, we should consider production cycle time (or customer delivery date) and production process cost.

We are consequently dealing with a multi-criteria optimisation problem. As these criteria are in conflict, it is better 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.

Moreover, if the two aspects (product and process) are taken into account, the size of the constrained search space is considerably increased. The optimisation tool used has to deal with a huge number of possible solutions. Consequently, we will use – and adapt to constrained problems – an Evolutionary Algorithm that has the advantage of avoiding the aggregation of criteria and can provide solutions on a Pareto front in an efficient way.

1.3 *Proposal for a two-step approach*

The two processes, (1) interactive configuration and planning and (2) configuration and planning optimisation, can be used independently. However, as explained in the introduction, we propose a two-step aiding system.

Our idea is that a product and associated production process can require a large number of variables in order to be entirely described, and that the customer or the system user is only interested in a small subset of these variables that correspond to their requirements. For example, a customer can be mainly interested in flight range and cost, and not in speed and delivery time. We assume that the user is able to decompose their requirement set into two sub-sets: non-negotiable requirements and negotiable ones. Our proposal is to process interactive configuration and planning with the first sub-set of requirements only (non-negotiable ones) and achieve a first reduction in the solution space. Remaining variable affectations (negotiable requirements) are kept for multi-criteria optimisation and provide solutions belonging to Pareto fronts that minimise cost, cycle time, or performance. With these Pareto fronts, the user can finish the process by selecting the solution that fits their specific compromise.

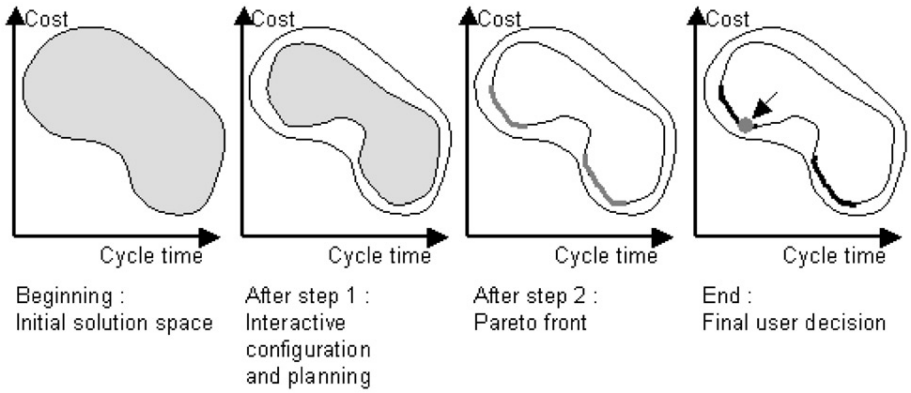

Figure 1. Proposed optimal configuration and planning process.

The resulting process is shown in Figure 1 with a cycle/cost compromise. The first step can be considered as a kind of partially interactive configuration and planning, while the second one is an autonomous optimisation process that terminates the job. In order to deal with negotiable requirements, industrial configuration software frequently proposes default values or static preference ordering. Our purpose is to avoid these weak solutions by minimising performance, cycle time, and cost.

As we are dealing with mass customisation, the response time of the aiding system must be considered. A response time of 1 s, for processing each elementary requirement, is generally accepted by users and achieved by configuration and planning software. Consequently, it can be derived that step 1 – with various attempts – lasts around 1 h. For the optimisation task of step 2, the response time unit is closer to an hour. We therefore have a rough order of magnitude of 1 h for each step. It is clear that such a problem can address different kinds of configured products: a personal computer of €1000, a car of €50,000, a boat of €500,000 or a plane of €5,000,000. Therefore, if a 1 h process sounds an adequate goal for a PC or a car, it is probable that a night of computation will not be a problem for the configuration of a plane.

The proposed aiding system provides concurrent product configuration and production planning, which makes it possible to take into account product and planning constraints and requirements simultaneously, thus avoiding iterations. The proposed two-step approach first interactively respects the customer's detailed requirements and then, in an autonomous way, proposes optimal compromises (performance, cycle time, cost).

1.4 *Problem restriction and organisation of the paper*

As we have presented initial ideas relevant to concurrent configuration and planning in Aldanondo *et al*. (2010), the main contribution of this paper is to the second step (optimisation) where an evolutionary algorithm will be adapted so as to take constraints into account. However, an example of the full problem must be considered in order to make some comparisons with a standard branch-and-bound optimisation approach.

The new decision-aiding approach described in this paper needs to provide suitable support for decision-makers who negotiate directly with customers to define what product will be made (configuration) and in what terms (process). Therefore, our method involves the presence of generic models for product (configuration), production process (planning) and coupling. For clarity, we have limited the scope of this paper to: (1) problems with only two criteria: cycle time and cost (grouping product and process cost); (2) problems that can be described on a single level of abstraction (issues dealing with composite configuration (Sabin and Freuder 1996) or multi-level planning with constraints (Mouhoub and Sukopan 2005) will not be addressed); and (3) infinite capacity planning, where we consider that production is launched according to each customer order, and production capacity is adapted accordingly.

The rest of the paper is organised as follows. The second section discusses related works. The third presents the constraint model supporting concurrent configuration and planning with the aircraft example. The fourth section is concerned with the evolutionary algorithm including specific constraint-based operators. The branch-and-bound

procedure that is used for comparison is also explained. Finally, detailed experimentation results and relevant discussions are proposed.

2. Related works

2.1 *Works relevant to product configuration and production planning with CSP*

In the field of configuration, many authors, including Mittal and Frayman (1989), originally, and later (Mailharro 1998, Junker 2006, Aldanondo and Vareilles 2008) showed that configuring a product can be efficiently modelled and aided when considered as a Constraints Satisfaction Problem (CSP). In a similar way, authors, such as Dechter *et al*. (1991) or Barták *et al*. (2010), interested in planning and scheduling have shown that these problems could also be modelled and aided when considered as a CSP. As explained above, we will concentrate on the filtering possibilities of the constraint approaches in order to provide interactivity. When dealing with the association of the two problems, significant works have been published on associating product and process design. Axiomatic Design (Suh 1990) and Design Structure Matrices (Lindemann 2007) have proposed some kinds of causality models that link product and production process. Dealing more specifically with configuration, Jiao *et al*. (2007) and Zhang *et al*. (2009) have proposed interesting generic modelling approaches that extend product configuration towards process configuration. However, these works did not examine how decision consequences could be propagated between product configuration and production planning. Propagation is addressed in Stewart and Tate (2000) and Aldanondo *et al*. (2010), where axiomatic design and constraint propagation are mixed, a feature that constitutes one of the bases of our proposal.

2.2 *Works relevant to optimisation of product configuration and production planning*

The first specificity of the optimisation problem under consideration is that the solution space is very large. It is reported in Amilhastre *et al*. (2002) that a configuration solution space of more than 1.4×10^{12} is required. It is clear that when planning is added, the combinatorial structure becomes very large indeed. Another aspect lies in the fact that the shape of the solution space is not continuous and, in most cases, shows many singularities. Furthermore, the multi-criteria aspect and the need for Pareto optimal results are also significant problem characteristics. These points explain why most of the articles published on this subject, such as Li *et al*. (2006), Hong *et al*. (2008), and Zhou *et al*. (2008) consider genetic or evolutionary approaches to deal with this problem. As our configuration and planning problem is considered as a CSP, our optimisation process should be able to take constraints into account. Clevenger *et al*. (2005) and Coello Coello's (2011) website describe various evolutionary approaches that handle constraints. These works will be detailed in Section 4 in order to introduce our optimisation proposals.

2.3 *Works relevant to our two-step proposal*

Very few works consider this issue directly. However, existing works dealing with the sequential association of configuration and/or planning with optimisation are linked either with studies relevant to the mass-customisation business process or with studies that associate interactive configuration and autonomous configuration completion. In the first case, some authors, coming mainly for the operation management community, such as Forza and Salvador (2002) or Hvam *et al*. (2002), discuss various ways to organise configuration, production, and planning activities. In the second case, the problems addressed are more relevant to the constraint community. Amilhastre *et al*. (2002) and Ullman (2007), for example, deal mainly with backtracking avoidance in interactive configuration or optimisation in autonomous configuration. As far as we know, there is no work that exactly matches our two-step proposal, although we have seen demos of configuration software solutions that are very close to our proposals.

3. Concurrent configuration and planning with constraint processing

We therefore assume that a constraint-based model of a generic product and a same kind of model for a generic production plan can be established, and we restrict configuration and planning tasks to the instantiation of these two models. The goal of this section is to describe, for configuration and for planning: each problem,

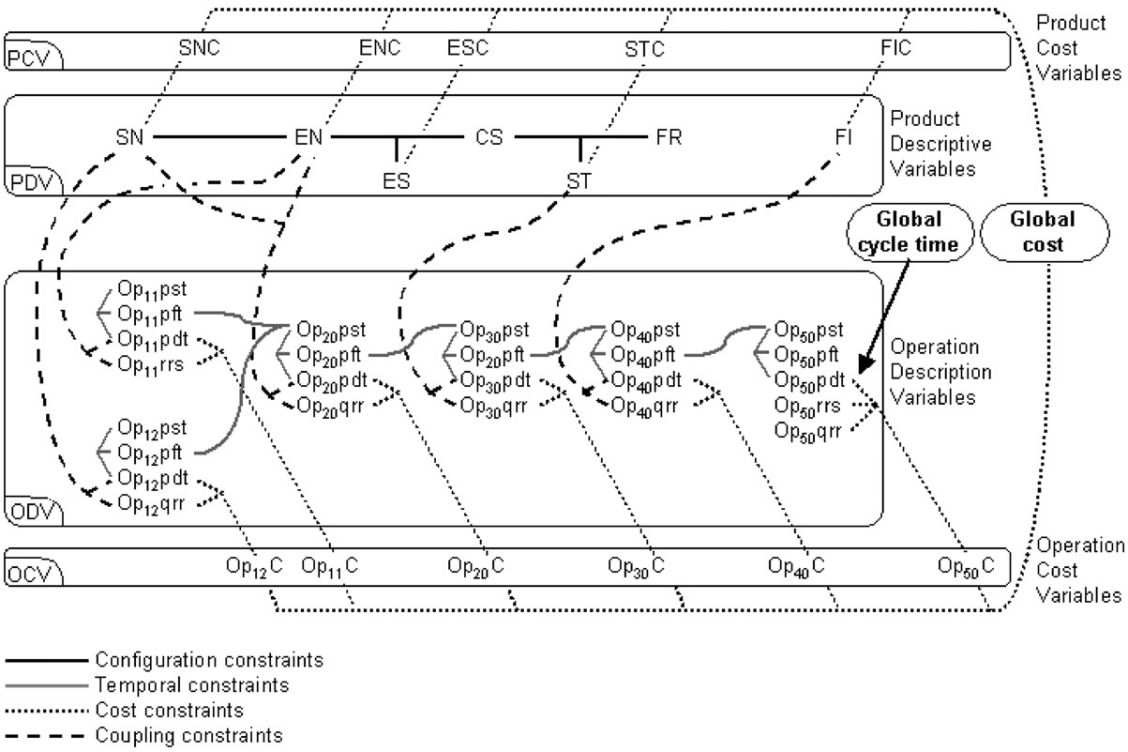

Figure 2. Configuration and planning constraint model.

each associated constraint model, and each relevant interactive constraint propagation process. Their association is explained in Aldanondo *et al*. (2010) where more details can be found. The constraint model of the aircraft example presented in Figure 2 is progressively detailed throughout the section.

3.1 *Configuration problem, relevant constraint model, and example*

In this paper, we consider that a generic product can be described by a set of properties or a set of components, or a mix of both, as proposed in Aldanondo and Vareilles (2008). Therefore, the proposed configuration model gathers a first sub-set of configuration variables called PDV for 'product description variables'. PDV variables can be associated with product properties (e.g. flight range, types of finish ...) or component type (for example: tank reference, engine reference ...). The definition domains of these variables are either symbols (for example: type of finish ...) or discrete numbers (e.g. flight range ...). The configuration constraints that link these PDV variables show the allowed combinations of variable values. They are shown in the upper part of Figure 2 with black plain lines.

Each PDV variable can have an influence on the product cost and therefore can be associated with variables belonging to a second sub-set of configuration variables called 'product cost variables' (PCV). These variables (last letter 'C' notation) are defined on a real domain. Configuration cost constraints link PDV with PCV variables and are shown in the upper part of Figure 2 with black dotted lines.

For the whole configuration model, gathering variables belonging to PDV and PCV, all the constraints describe allowed the combination of variable values and are discrete. There is no calculus or numerical constraint. The associated CSP is therefore discrete, and the filtering provided by arc consistency techniques, synthesised in Bessiere (2006), allows interactive configuration and an estimation of product cost variables.

Our example deals with a simplified version of a configuration of a small plane. The plane is characterised by seven product description variables (PDV subset): SN (seat number), EN (engine), CS (cruising speed), FR (flight range), FI (finish reference), ES (engine setting), and ST (supplementary tank) reference. Three configuration

constraints show allowed combinations between: plane size and engine (SN, EN); engine and speed that imply an engine setting (EN, CS, ES); speed and flight range with a supplementary tank (CS, FR, ST). The plane is characterised by five PCV associated with product variables with cost constraints: SNC (seat number), ENC (engine), FIC (finish), ESC (engine setting), and STC (supplementary tank). This part of the model is shown in the upper part of Figure 2.

3.2 *Planning problem and relevant constraint model*

We consider that a generic production process can be described with a set of planning operations (supplying, manufacturing, assembling, etc.) linked to anteriority constraints. Each operation is defined with:

- three operation temporal variables: possible starting time (pst), possible finish time (pft), possible duration (pdt), defined on a real domain;
- two operation resource variables: required resource (rrs) defined on a symbolic domain, quantity of resource (qrr) defined on integer domain;

which belong to a first sub-set of planning variables called 'operation description variables' (ODV). Two kinds of planning constraints exist. The first kind is temporal constraints, which gathers:

- for each operation (Op_i), a duration constraint explaining that finish time equals start time plus duration: $Op_i pft = Op_i pst + Op_i pdt$;
- for each sequence of two operations ($Op_i - Op_{i+1}$), a precedence constraint explaining that next operation start time ($Op_{i+1}pst$) is larger or equal than previous operation finish time ($Op_i pft$): $Op_{i+1}pst \geq Op_i pft$.

These temporal constraints are numerical and shown in the lower part of Figure 2 with plain grey lines. The second group of constraints corresponds to resource constraints that link operation resource variables (rrs, qrr) with the duration of the operation (pdt). These are mixed constraints that show allowed combinations of values.

Among these five kinds of variables, only the two resource variables, resource (rrs) and quantity (qrr), can influence the production process cost. They can be associated with 'operation cost variables', which are denoted as 'Op_iC', defined on a real domain and belong to the subset OCV. Production-cost constraints, which are also mixed constraints, link ODV with OCV variables. They are shown in the lower part of Figure 2 with black dotted lines

For the whole planning model, gathering variables belonging to ODV or OCV, cost constraints are discrete, while temporal constraints are numerical. As these numerical constraints are elementary calculations ($+$, $-$, $\times$, $/$, $=$, $>$, $<$), they respect the hypothesis of bound consistency proposed in Lhomme (1993). Based on interval arithmetic, bound consistency is associated with discrete arc consistency and allows interactive planning and estimation of production cost variables.

In our example, the production process of the plane involves six operations. Each operation Op_i is characterised by the six variables (Op_ipst, Op_ipft, Op_ipdt, Op_irrs Op_iqrr, Op_iC) with the relevant resource and cost constraints. The production process begins with two simultaneous operations, supplying Op_{11} and manufacturing Op_{12}, followed by assembling Op_{20}, tank assembling Op_{30}, finishing Op_{40}, and finally packaging and delivery Op_{50}. This part of the model is shown in the lower part of Figure 2.

3.3 *Associated problems and optimisation criteria*

The global problem gathers configuration and planning models using coupling constraints. A coupling constraint links at least one variable of the configuration model with at least one variable of the planning model. Any variable belonging to PDV can belong to a coupling constraint, while only resource variables (Op_irrs, Op_iqrr) or the duration variable (Op_ipdt) of ODV can. These constraints are shown in the centre of Figure 2 with black broken lines. They are either discrete or mixed constraints with no calculus or numerical constraint; therefore, arc consistency techniques allow a configuration decision to be propagated towards planning, and in the opposite way, a planning decision towards configuration.

In terms of Pareto criteria, the global cost and the global duration can be defined as follows. The global cost (GC) is the sum of all product cost and operation cost variables (PCV and OCV). Global cycle time (GT) corresponds to the earliest possible finishing time of the last operation of the production process.

In our example, the association of the two models requires five coupling constraints. The last operation, packaging and delivery (Op_{50}), is not influenced by the plane configuration. The coupling constraints link: supplying (Op_{11}) with engine (EN), manufacturing (Op_{12}) with seat number (SN), assembling (Op_{20}) with both engine and seat number (EN, SN), tank assembling (Op_{30}) with supplementary tank reference (ST), finishing (Op_{50}) with finish (FI). The global cost GC is obtained with the following numerical constraints:

$$GC = (SNC + ENC + FIC + ESC + STC) + \left(Op_{11}C + Op_{12}C + Op_{20}C + Op_{30}C + Op_{40}C + Op_{50}C\right).$$

The global cycle time corresponds to the finishing time of the packaging and delivery operation: $GT = Op_{50}pft$.

The definition of these coupling constraints completes the model and allows the representation in Figure 2 of the global constraint model associating configuration and planning. In terms of model size, each configuration variable (PDV) has a domain size between 4 and 6, while each process operation (ODV) has an average combinatory (required resource × resource quantity) of between 3 and 25. Without taking constraints into account, the solution space of the product model is 5184, and the planning model is 96,000. The size of the global problem model is 497,664,000. This model can be consulted and interactively used at http://cofiade.enstimac.fr/cgi-bin/cofiade.pl select model 'Aircraft-CSP-EA-10'.

4. Optimisation problem and evolutionary algorithm proposals

This section first deals with the optimisation problem; then the modified evolutionary algorithm is presented in detail. A short presentation of the branch-and-bound procedure, required for experimental computations, concludes this section.

4.1 *Definition of the optimisation problem*

The previous first step leads to the restriction of the initial feasible space (first graph of Figure 1) to a restrained area (second graph of Figure 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 global cost and global cycle time). The restrained area contains solutions corresponding to different remaining decisions to fulfil. But this area might also contain unfeasible solutions owing to the constraints of the problem. The aim of the optimisation process is to find a selection of solutions close to the optimal Pareto front (third graph of Figure 1). To solve this problem, we focus on evolutionary algorithms for their ability to propose multiple solutions while solving a multiobjective problem. But classic evolutionary algorithms have to be adapted to take into account the constraints of the problem.

The constrained optimisation 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 each variable of V, C the set of constraints on variable of V, and f the multi-valuated fitness function. Here the aim is to minimise both global cost and global cycle time. The set V gathers the product descriptive variables and the operation resource variables. Constraints link the variables of V. The filtering system dynamically updates the domain of these variables with respect to the constraints. This O-CSP is a difficult problem to solve. The existing methods to handle constraints in EA are often computationally expensive.

4.2 *Overview of constrained optimisation approaches*

Initially, EAs deal with large combinative unconstrained problems, but real-world problems are generally constrained. Many research studies have tried to integrate constraints in EA. Coello Coello (2002, 2011) proposes a wide state-of-the-art study of these methods and a brief overview. Four kinds of methods deal with this problem: penalty functions (Richardson *et al*. 1989), repairing methods (Salcedo-Sanz 2009), approaches that separate objectives and constraints (Multiobjective Optimisation techniques for example proposed by Clevenger *et al*. 2005), and specific representations or operators (Michalewicz and Nazhiyath 1995, Kowalczyk 1997).

Penalty functions are the most common way to integrate violation of constraints in an objective function. For each individual, the level of violation of constraint is added to the fitness function. The main drawback of such an approach is that the boundary between feasible and unfeasible regions is usually difficult to grasp. Furthermore, it requires the definition of the weights needed to aggregate the violation of different constraints.

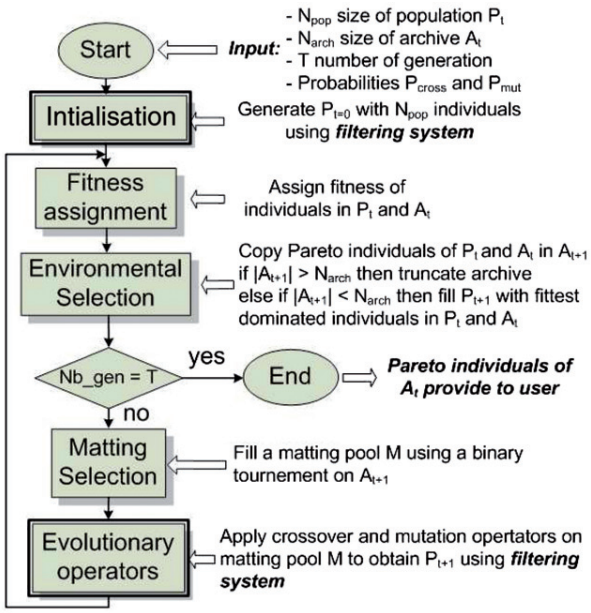

Figure 3. Adapted SPEA2 algorithm.

Repairing methods only try to deal with feasible individuals. As soon as an unfeasible individual is generated, a specific operator redirects it towards the feasible space. The difficulty is thus to elaborate an effective repairing algorithm that preserves the diversity of individuals. The same problem appears with Multiobjective Optimisation approaches. These integrate the satisfaction of each constraint (or a group of constraints) as a specific objective.

Finally, the specific operators or representation approaches aim at preserving the feasibility of the individuals during their construction. Kowalczyk (1997) proposed first ideas relevant to the use of constraint consistency to prevent variable instantiations that are not consistent with the constraints of the problem. In this paper, we accept this idea and focus on specific evolutionary operators that prune search space using constraint filtering.

4.3 *Proposed approach for a constraint-based evolutionary algorithm*

4.3.1 *Overview of modified EA*

The EA used here is adapted from the SPEA2 method (Zitzler *et al.* 2001) with classic evolutionary steps illustrated on Figure 3. It 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 denoted A_t and population P_t). It ensures both a good convergence speed and the preservation of diversity of solutions. Five parameters are required: size of archive (N_{arch}), size of population (N_{pop}), number of generations (T) (or any stopping criterion), probabilities for crossover (P_{cross}), and mutation (P_{mut}) operators.

After initialisation, SPEA2 includes an efficient evaluation strategy that brings a well-balanced density of population on each area of search space. The calculation of fitness is based on the Pareto dominance between individuals (relative performance of a solution) and the measurement of solution density. The information on density makes it possible to discern individuals with identical scores according to the density of individuals in the surrounding area. Following fitness assignment, the step of environmental selection updates individuals of the archive. The archive is filled with a selection of best individuals in the population and the previous archive. Finally, if the stopping criterion is not reached, the mating selection step selects N_{pop} individuals in the archive to constitute the mating pool. This is done by a binary tournament between individuals of the archive. Crossover and mutation operators will use this mating pool to build the next generation. See Zitzler *et al.* (2001) for more details on SPEA2 calculations.

For our situation, the initialisation and evolutionary operators steps (framed with a bold line on Figure 3) have been modified to interact with a filtering system in order to build or modify the individuals according to the constraints of the problem (e.g. to keep them in the feasible space). The following sections detail the initialisation, crossover, and mutation operators that interact with the filtering system.

4.3.2 *Initialisation operator*

This operator provides a well-diversified set of initial individuals. The pseudo-algorithm below details its general behaviour. For each individual to create, every gene (decision variable) is randomly instantiated into its current domain. To avoid the generation of unfeasible individuals, the domain of every remaining gene is dynamically updated by filtering after each instantiation. If an individual is inconsistent, a limited backtrack process cancels one of the previous choices; then the individual is filtered again until the values of the remaining variable are consistent. If the backtrack limit is reached, the individual is abandoned to bound the computational time spent by the filtering. This process (random instantiation then filtering) is repeated until all the genes of every individual are instantiated.

Pseudo-algorithm of initialisation operator:
Input: Empty initial population P_0
Required size of population N_{pop}
Start
While (size(P_0) < N_{pop})
Select an individual not finished
Select an uninstantiated variable on it
Select a value in variable domain
Ask for a filtering of individual
If (consistent instantiation) **then**
Update domains of remaining genes
Else
While (individual is unfeasible) **do**
If (Backtrack counter limit reach) **then**
Restart with empty individual
Else
Randomly select an instantiated variable
Uninstantiate it
Increase Backtrack counter
Ask for a filtering of individual
EndIf
EndWhile
EndIf
if (every gene is instantiated) **then**
Add individual to the population P_0
Endif
EndWhile
End

4.3.3 *Uniform mutation operator*

This operator introduces a random perturbation on the evolutionary process that allows escape from sub-optimal areas. It modifies the instantiation of some genes on individuals selected according to the mutation probability P_{mut} among mating pool M. For each randomly selected individual, the randomly selected genes are uninstantiated. The filtering system updates the domain of these variables according to the instantiation of other genes. Finally, the mutation of the selected genes itself is achieved in the same way as during the initialisation (instantiation, filtering and backtrack limit). The generated individuals are added to the next generation P_t.

4.3.4 *Uniform crossover operator*

This operator allows random and uniform shuffling of the genes of two individuals (named parents) selected in the mating pool M according to P_{cross}. The pseudo-algorithm on the next page describes its general behaviour.

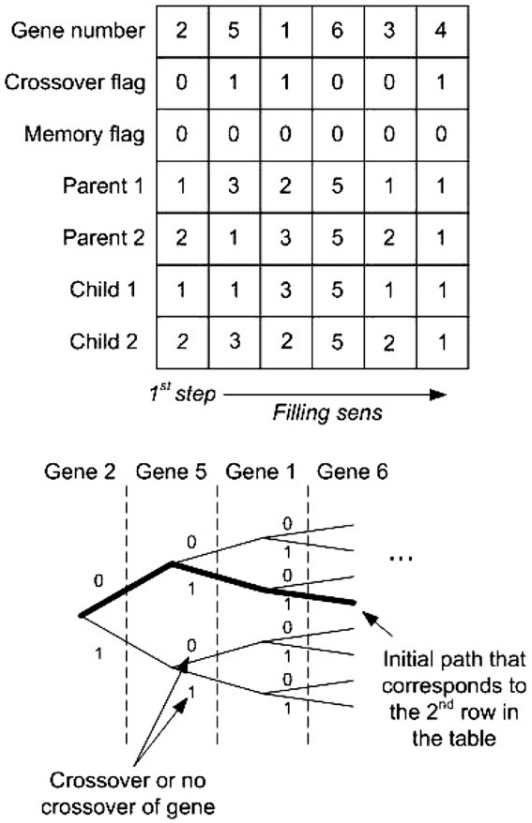

Figure 4. Crossover table and tree.

To achieve this task, a crossover table (represented on the upper part of Figure 4) makes it possible to select which genes will be exchanged between parents. The crossover corresponds to a selected path on a binary tree (represented on the lower part of Figure 5), where each branch is linked to the crossover of a particular gene. An instantiation of the crossover table (lines that correspond to child) is equivalent to the selection of a path on the crossover tree.

First, the two first lines of the table are randomly filled in (the first line corresponds to the order of treatment of genes, and the second one selects which genes will be crossed). The order of treatment of a specific gene is chosen randomly to avoid the dominance of genes by their position in the chromosome. The initialisation of crossover flags corresponds to a random selection of a path in the graph.

The system tries to achieve this random crossover. At every gene instantiation, the filtering system updates the domain of remaining genes. The crossover table is initially filled in identically for both children, but if an individual becomes unfeasible, a specific backtrack is carried out by changing a crossover flag in their specific table. A supplementary flag (third line) is added to the table that memorises the unfeasible path on the tree in case of backtrack. Then, a backtrack counter limits the number of backtracks. If the backtrack limit is reached, the corresponding child is abandoned. Finally, every feasible child is added to the next generation.

Pseudo-algorithm of crossover operator:
Input: P_{cross} Crossover probability
M mating pool with N_{pop} individuals
Start
For(all individuals in M) **do**
If(Random$(0, 1) < P_{\text{cross}}$) **then**
Randomly select another Parent of M
Create corresponding crossover table

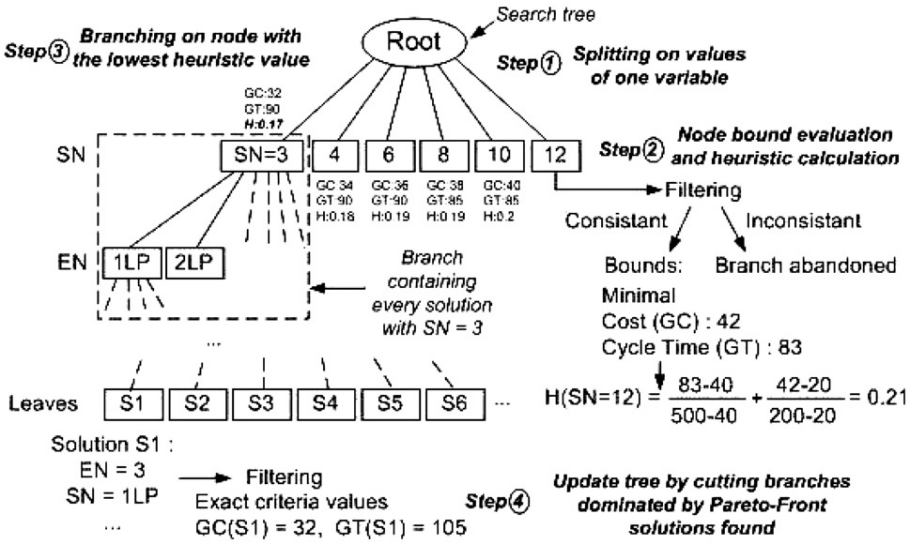

Figure 5. Search tree of BB algorithm.

```
        Fill randomly with gene number the 1st line
        Fill according to P_cross with 0 or 1 the flags of 2nd line
        Fill systematically with 0 memory the flags of 3rd line
        While (crossover done for both children) do
          Apply next crossover according to 2nd line (1 cross/0 no cross)
          Ask for a filtering of both individuals
          If(a child is unfeasible) then
            If(Backtrack counter limit reach) then
              Abandon corresponding child
            Else
            While (memory flag of current gene = 1) do
              Set memory flag of current gene to 0
              Go back to previous gene
              EndWhile
              Set memory flag of current gene to 1
              Switch crossover flag of current gene
            EndIf
          EndIf
        EndWhile
      EndIf
    EndFor
  End
```

4.4 *Branch-and-bound procedure using filtering system*

The branch-and-bound (BB) method is a classic systematic algorithm for finding optimal solutions to various optimisation problems. This method was first proposed in Land and Doig (1960) for linear programming. It consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless solutions are discarded by using the upper and lower estimated bounds of the quantity being optimised. The key idea of the BB 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 (e.g. during search-tree exploration, if the lower bound for some tree-node A is greater than the

upper bound for some other node B, then A may be safely discarded from the search). So, a branch-and-bound procedure requires three tools: a splitting procedure, a bound-evaluation procedure and a heuristic.

The splitting procedure, given a set S of candidates, returns two or more smaller sets (S_1, S_2, etc. . . .) whose union covers S. Note that the minimum of $f(x)$ over S is $\mathrm{Min}(v_1, v_2\}$, where each v_i is the minimum of $f(x)$ within S_i. This step is called branching, since its recursive application defines a tree structure whose nodes are the subsets of S. For our problem, the splitting procedure corresponds to the selection of one variable of the problem and to the instantiation of this variable for each possible value. For example, the first split could be on seat-number variable (SN) and generates six child nodes as illustrated on Figure 5, step 1.

The second tool needed is a node-bound evaluation procedure. The filtering process is used to achieve this task. With a partial instantiation (e.g. a node of the search tree, for example the node (SN = 12) on the tree on Figure 5, step 2), the filtering system is able to evaluate if the partial instantiation is consistent with the constraints of the problem, and, if this is the case, to provide the lower bound of each criterion. Notice that these lower bounds are optimistic values for each criterion taken separately. When the search reaches a leaf of the search tree (e.g. a complete instantiation, see step 4 on Figure 5), 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).

Finally, the last tool needed is a heuristic that allows selection of the next node to explore. The heuristic used is the lowest normalised aggregation of lower bounds of each remaining node. For example, if node A has (GC_A, GT_A) as lower bounds for global cost (GC) and global cycle time (GT), its heuristic value will be:

$$H(\mathrm{A}) = \frac{\mathrm{GT}_A - \mathrm{Min}_{GT}}{\mathrm{Max}_{GT} - \mathrm{Min}_{GT}} + \frac{\mathrm{GC}_A - \mathrm{Min}_{GC}}{\mathrm{Max}_{GC} - \mathrm{Min}_{GC}}. \tag{1}$$

where Min_i and Max_i are minimal and maximal values of each criterion i.

This behaviour is illustrated in Figure 5 with the third step (branching on node with lowest heuristic value). Among the six possible values, the node (SN = 3) has the lowest heuristic value (H(SN = 3) = 0.17). The algorithm goes on with this node, and the other nodes are saved on an ordered list of remaining nodes (according to their heuristic value).

5. Experimental results and discussions

We performed various tests on the aircraft problem. The optimisation algorithms were implemented in C++ programming language and interacted with the filtering system coded in Perl language. The completed system was designed to work with a multiple parallel filtering system (to take advantage of using a population of parallel solutions to filter). However, in order to evaluate the performance of the algorithm clearly, we used only one filtering system for the results presented in this paper. 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 GB of RAM. These tests compared the behaviour of our constrained EA algorithm with the exact BB algorithm.

In a first sub-section, two model sizes and two constraint levels were considered in order to qualify our EA proposition. Neither user reduction nor an initial configuration/planning process was considered for these first experiments. Then, optimisation following a user reduction, corresponding to the process in Figure 1, is studied in a second sub-section.

For a multiobjective 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 and Thiele (1998) and illustrated in Figure 6. It measures the hypervolume of space dominated by a set of solutions. It thus allows evaluation of both convergence and diversity properties (the fittest and most diversified set of solutions is the one that maximises hypervolume).

5.1 *Impact of the model size and level of constraint without user reduction*

Different versions of the aircraft model were investigated in order to evaluate the impact of the model size and the constraint density on the algorithm performance.

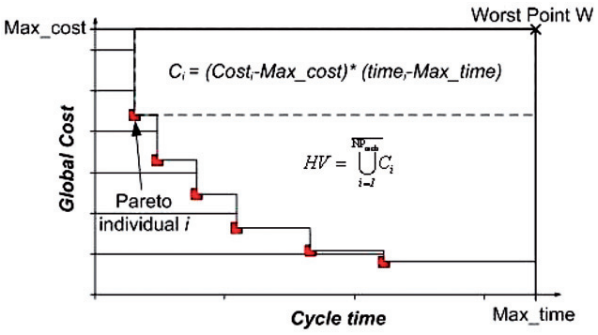

Figure 6. Example of hypervolume linked to a Pareto front.

Regarding model size, the model presented in Section 2 was reduced by the suppression of the last task and the reduction in possible values for the type of resource on the first task. Thus, algorithm behaviour (EA and BB) was evaluated on a full model or on a small one characterised by:

- full model: 497,664,000 solutions without constraints;
- small model: 6,220,800 solutions without constraints.

Density of constraints: density of constraints is linked with the number of allowed combinations between problem variables. To evaluate the impact of constraints density, two versions of the aircraft model were produced: one with a weak density of constraints (20% of possible combinations between configuration variables were prohibited) and the other with a higher density of constraints (50% of possible combinations between configuration variables were excluded). This provides four models characterised by:

- full model: around 47,600,000 feasible solutions for the low-constrained model (9.5% of search space), around 12,288,000 feasible solutions for the high-constrained model (1.9% of search space)
- small model: around 595,000 feasible solutions for the low-constrained model, around 153,000 feasible solutions for the high-constrained model.

Those numbers are evaluated by a sampling of search space (percentage of solutions feasible obtained with a random sampling of around 50,000 solutions). The curves in Figure 7 illustrate the evolution of the hypervolume dominated by the Pareto front solutions (the horizontal axis corresponds to the time in seconds) for each model investigated:

- small model in the upper part of Figure 7 and full model in the lower part of the same figure;
- low-constrained model in the left part of Figure 7 and high-constrained model in the right part of the same figure.

Notice that the curves that represent EA performance are average results for 30 executions. Two sets of evolutionary settings are used to obtain these results:

- For the small models, we used evolutionary settings adapted to the small search space: population size: 50; archive size: 40; P_{mut}: 0.4; P_{cross}: 0.8. The ending criterion used is a limit of time (algorithm stopped if time consumed after each generation is greater than 1800s, half an hour).
- For the full models, we adapt settings for a wider search: population size: 150; archive size: 100; P_{mut}: 0.4; P_{cross}: 0.8. The ending criterion used is a limit of time that corresponds to the time consumed by the BB algorithm.

Notice that parameter settings are one of the main drawbacks to 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 allows a quick improvement of 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.

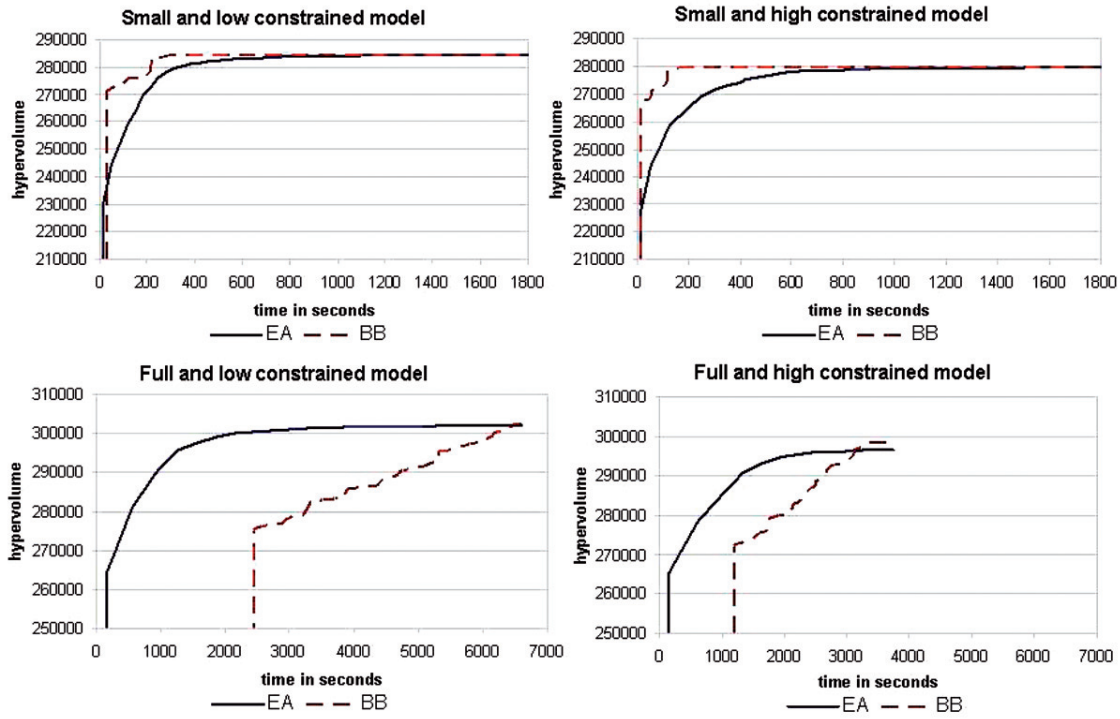

Figure 7. Experimentation with different model sizes and constraint densities.

For the small models (first two curves), the BB algorithm reaches the optimal Pareto front much faster compared with EA performance (290 s for the low-constrained model and 163 for the high-constrained one. See Table 1 for more detail). Both algorithms start with a lapse of time where performance is null. For the BB algorithm, this corresponds to the time needed to reach a first leaf on the search tree, while for the EA, it corresponds to the time consumed to constitute the initial population. The EA, after a rapid increase in hypervolume, stagnates and converges very slowly to the optimal Pareto front (less than 0.5% of improvement on the last 400 s). This is indeed one of the weak points of classic EA.

On the other hand, the EA greatly outperform the BB algorithm on the full model. For example, on the low-constrained model, the BB algorithm took 20 times longer to reach a good set of solutions (less than 0.5% of the optimal hypervolume) for the complete model compared with the small one, while the EA needed only five times longer (for a model 20 times larger).

The impact of constraints density could also be analysed. Table 1 shows values obtained on these models. As can be seen, the BB algorithm performance is improved when the density of constraints is high. 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 EA performance moves in the opposite way. The more the model is constrained, the more the random crossover operation will have to backtrack to find feasible solutions, and thus the time needed by the algorithm will be consequent.

The main conclusion of this section is that the proposed EA algorithm is clearly better than the BB algorithm when the size of the problem increases. A second conclusion is that an increase in the density of constraints tends to limit the tendency of the previous conclusion. In the next section, we therefore consider only the full problem with a low level of constraint and user reduction that is our ultimate goal.

5.2 *Optimising a full problem with user reduction*

Our ultimate goal is to associate configuration and planning with optimisation of the compromise cycle/cost. As previously explained, the user specifies non-negotiable requirements in a first interactive phase. The first step

Table 1. Comparison of EA and BB according to model sizes and constraint densities.

Time (s)	Percentage of time consumed by BB algorithm	Hypervolume for hypervolume with BB algorithm	Hypervolume for hypervolume with BB algorithm then with EA	Relative standard deviation of hypervolume for the 30 runs	EA/BB deviation (%)	Deviation from optimal value (%)	Time (s)	Percentage of time consumed by BB algorithm	Hypervolume for hypervolume with BB algorithm	Hypervolume for hypervolume with BB algorithm then with EA	Relative standard deviation of hypervolume for the 30 runs	EA/BB deviation (%)	Deviation from optimal value (%)
Small and low constrained model							Small and high constrained model						
72	25	272,647	252,330	4.83	−7.45	11.25	41	25	267,798	228,031	9.22	−14.85	0.18
145	50	275,910	263,899	4.13	−4.35	7.18	82	50	271,063	251,126	5.75	−7.36	0.10
290	100	284,323	279,208	1.05	−1.80	1.80	164	100	279,294	261,793	4.22	−6.27	0.06
1812	625	284,243	0.05	−0.03	0.03	1814	1106	279,294	0	0	0		
Full and low constrained model							Full and high constrained model						
1619	25	0	297,486	0.60	100	1.58	835	25	0	278,399	2.06	100	6.73
3237	50	278,894	301,208	0.30	8.00	0.35	1670	50	275,590	293,264	0.57	6.41	1.75
6474	100	302,254	301,903	0.16	−0.12	0.12	3340	100	298,498	296,447	0.50	−0.69	0.69

Table 2. Comparison of EA and BB on a full problem with user reduction.

Time (s)	Percentage of time consumed by BB algorithm	Hypervolume for hypervolume with BB algorithm	Hypervolume for hypervolume with BB algorithm then with EA	Relative standard deviation of hypervolume for the 30 runs	EA/BB deviation (%)	Deviation from optimal value (%)
2996	25	0	189,064	0.37	100	0.33
5991	50	0	189,427	0.35	100	0.14
12,060	100	189,697	189,530	0.34	−0.09	0.09

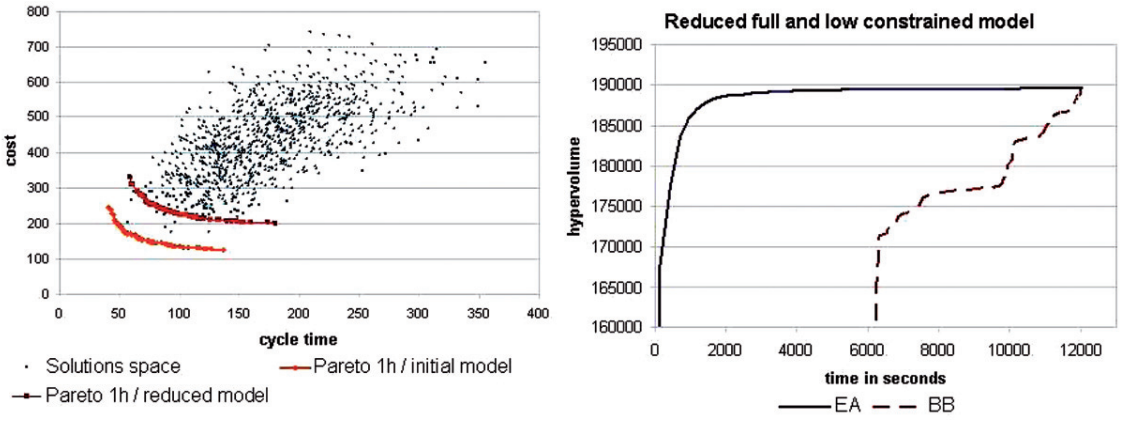

Figure 8. Experimentation on a full problem with user reduction.

suggests that the user defines these non-negotiable requirements by selecting the required values for some decision and asks for a filtering of the model. The previous full model is considered, and the following reductions on the variables are made:

- SN: seat number: two values are kept instead of six;
- FI: finish: two values are kept instead of four.

For this model, the number of feasible solutions is around 6,350,000 solutions. The initial search-space shape (space of existing solutions for the whole problem without any restriction requested by the user) is represented in the left part of Figure 8 with the Pareto front provided by the EA algorithm after 1 h. In the same figure, the Pareto front obtained by the EA algorithm after 1 h on the reduced model is also shown. In Table 2 and in the right part of Figure 8, the evolution of the hypervolume of EA and BB algorithms on the reduced problem are shown.

When comparing the two Pareto fronts, it is clear that the user's requirements exclude the most economical solutions. The hypervolume progression shows that it is possible rapidly to obtain a first idea of the possible cycle/cost compromises that can be proposed to the user.

As shown by the hypervolume curves in Figure 8, a reduction made by the user can have a huge impact on BB algorithm performance while AE is relatively unaffected by this operation. For the BB algorithm, reductions made by the user suppress some branches on the search tree, so its execution would be faster. However, with the reduction proposed, the optimal Pareto front is in a highly dense area of the search space (a large number of solutions exist in this area as shown in the search spaces of Figure 8). This aspect greatly impacts the performance of the BB algorithm because a large number of solutions seem to be interesting, but considerable backtracking occurred before it reached the leaves of the search tree. Therefore the search tree needs to be developed to a considerable depth to find the optimal set of solutions.

These results match our expectations, because they lead to the conclusion that, with EA, it is possible to propose, in a reasonable amount of time, a set of solutions that permits the user to decide on their own cost/cycle time

compromise, whatever the reduction made by the user. Therefore, the two-step process described in this paper can doubtless be considered as being of significant assistance to achieve optimal configuration and planning.

6. Conclusions

In this paper, we have presented an efficient tool that is able to assist product configuration and production process planning concurrently, using an interactive constraint-filtering system and an evolutionary optimisation system. This aiding tool operates with respect to a two-phase process. The first phase consists of an interactive configuration of the product and the relevant production process. This configuration is carried out using a set of non-negotiable variables on which the user can make choices. The second phase involves optimising the previous result regarding two antagonist criteria, which, in our case, are the cost and cycle time of the product.

The first phase is based on constraint-filtering algorithms that allow the user to interactively manipulate the configuration and planning models, and to process their non-negotiable requirements. The main interest in this approach is to enable the propagation of configuration decisions towards planning and, in the opposite direction, the propagation of planning decisions towards configuration. Thus, configuration and planning are achieved interactively, simultaneously, and progressively. The traditional sequence 'configure the product first and then plan its production process' is no longer a necessity. Consequently, time-consuming iterations between configuration and planning are greatly reduced.

The second phase is based on a modified EA that allows the constraints to be taken into account during the optimisation process. The modified EA uses the filtering system to prune the search space and thus to reduce search efforts by limiting it to the feasible individuals. Standard evolutionary operators are adapted to take advantage of the filtering. In particular, we propose an original limited backtracking crossover operator. Our proposals have been compared with a BB algorithm. Initial experiments indicate that this method seems to be very well adapted for optimising configuration and planning concurrently. It generates near-optimal Pareto solutions in a reasonable computing time. The comparison of both performance and diversity of Pareto fronts is ensured by the hypervolume metric.

The combination of these two phases means that our aiding tool is able to configure the best solutions and to give an idea of what they are to the user. It allows the user to decide efficiently about his cost/cycle-time compromise when dealing simultaneously with configuration and planning. This promising work introduces a large number of prospective studies: scale aspects, evolutionary parameter tuning, comparison with other constrained evolutionary approaches, and also problems grouping more than two objectives. For scale aspects or larger problems, we are currently developing an iterative optimisation 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 optimisation process. For the tuning of EA parameters, we could consider the possibility of an automated setting with a variable population and archive size. For comparison with other constrained evolutionary approaches, according to the density of constraints and the model size, we have included a study of penalty functions in our list of future works. Finally, configuration and planning problems taking into account objectives such as performance, risk, or quality will also be considered.

References

Aldanondo, M. and Vareilles, E., 2008. Configuration for mass customization: how to extend product configuration towards requirements and process configuration. *Journal of Intelligent Manufacturing*, 19 (5), 521–535.

Aldanondo, M., Vareilles, E., and Djefel, M., 2010. Towards an association of product configuration with production planning. *International Journal of Mass Customisation*, 3 (4), 316–332.

Amilhastre, J., Fargier, H., and Marquis, P., 2002. Consistency restoration and explanations in dynamic csps – application to configuration. *Artificial Intelligence*, 135 (1–2), 199–234.

Barták, R., Salido, M., and Rossi, F., 2010. Constraint satisfaction techniques in planning and scheduling. *Journal. of Intelligent Manufacturing*, 21 (1), 5–15.

Bessiere, C., 2006. Constraint propagation. *In*: F. Rossi, P. Van Beek and T. Walsh, eds. *Handbook of constraint programming*. Amsterdam: Elsevier, 29–70.

Clevenger, L., Ferguson, L., and Hart, W.E., 2005. Filter-based evolutionary algorithm for constrained optimization. *Evolutionary Computation*, 13 (3), 329–352.

Coello Coello, C., 2002. Theoretical and numerical constraint-handling techniques used with EAs: a survey of the state of art. *Computer Methods in Applied Mechanics and Engineering*, 191 (11–12), 1245–1287.
Coello Coello, C., 2011. List of references on constraint-handling techniques used with evolutionary algorithms: CINVESTAV, www.cs.cinvestav.mx/~constraint/.
Dechter, R., Meiri, I., and Pearl, J., 1991. Temporal constraint satisfaction problems. *Artificial Intelligence*, 49, 61–95.
Forza, C. and Salvador, F., 2002. Managing for variety in the order acquisition and fulfilment process: the contribution of product configuration systems. *International Journal of Production Economics*, 76 (1), 87–98.
Hong, G., *et al*., 2008. Identification of the optimal product configuration and parameters based on individual customer requirements on performance and costs in one-of-a-kind production. *International Journal of Production Research*, 46 (12), 3297–3326.
Hvam, L., Riis, J. and Malis, M., 2002. A multi-perspective approach for the design of configuration systems. *In*: *ECAI configuration workshop proceedings*, Lyon, France, 56–62.
Jiao, J., Zhang, L., and Pokharel, S., 2007. Process platform planning for variety coordination from design to production in mass customisation manufacturing. *IEEE Transactions on Engineering Management*, 54 (1), 112–129.
Junker, U., 2006. Configuration. *In*: F. Rossi, P. Van Beek and T. Walsh, eds. *Handbook of constraint programming*. Elsevier: Amsterdam, 835–875.
Kowalczyk, R., 1997. Constraint consistent genetic algorithms. *In*: *IEEE evolutionary computation conference proceedings*, 343–348.
Land, A.H. and Doig, A.G., 1960. An automatic method of solving discrete programming problems. *Econometrica*, 28 (3), 497–520.
Lhomme, O., 1993. Consistency techniques for numerical CSPs. *In*: *IJCAI 1993 proceedings,* Chambery France, 232–238.
Li, L., *et al*., 2006. Product configuration optimization using a multiobjective genetic algorithm. *International Journal of Advanced. Manufacturing Technology*, 30 (1–2), 20–29.
Lindemann, U., (2007). A vision to overcome 'chaotic' design for X processes in early phases. *In*: *Engineering design conference proceedings,* Paris, 231–232.
Mailharro, D., 1998. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 12 (4), 383–397.
Michalewicz, Z. and Nazhiyath, G., 1995. Genocop III: a co-evolutionary algorithm for numerical optimization with non linear constraints. *IEEE evolutionary computation conference proceedings*, 647–651.
Mittal, S. and Frayman, F., 1989. Towards a generic model of configuration tasks. *In*: *IJCAI 1989 proceedings,* Detroit, MI, 1395–1401.
Mouhoub, M. and Sukopan, A., 2005. A new temporal CSP framework handling composite variables and activity constraints. *In*: *Conference on tools with artificial intelligence proceedings,* Hong Kong, 143–149.
Pine, B., 1993. *Mass customisation – the new frontier in business competition*. Boston: Harvard Business School Press.
Richardson, J.T., *et al*., 1989. Some guidelines for genetic algorithms with penalty functions. *In*: *Conference on genetic algorithms proceedings*, Fairfax, VA, 191–197.
Sabin, D. and Freuder, E., 1996. Configuration as composite constraint satisfaction. *In*: *Artificial intelligence and manufacturing research planning workshop proceedings*, AAAI, 37–44.
Salcedo-Sanz, S., 2009. A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer Science Review*, 3 (3), 175–192.
Steward, D. and Tate, D., 2000. Integration of axiomatic design and project planning. *In*: *Axiomatic design conference proceedings*, Boston, 285–289.
Suh, N., 1990. *The principles of design*. Oxford University Press.
Ullman, J.R., 2007. Partition search for non-binary constraint satisfaction. *Information Sciences*, 177 (18), 3639–3678.
Zhang, L., *et al*., 2009. Supply chain configuration with coordinated product, process and logistics decision: an approach based on Petri nets. *International Journal of Production Research*, 47 (23), 6681–6706.
Zhou, C., Zhihang, L., and Chuntao, L., 2008. Customer-driven product configuration optimization for assemble-to-order manufacturing enterprises. *International Journal of advanced manufacturing technology*, 38 (1–2), 185–194.
Zitzler, E. and Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms – a comparative case study. *In*: *Conference on parallel problem solving from nature proceedings*, Amsterdam, 292–301.
Zitzler, E., Laumanns, M. and Thiele, L., 2001. SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. *In*: *Evolutionary methods for design, optimisation and control with application to industrial problems proceedings,* Athens, 95–100.