Aiding Interactive Configuration and Planning: A Constraint and Evolutionary Approach

Paul Pitiot[1], Michel Aldanondo[1], Elise Vareilles[1], Paul Gaborit[1], Meriem Djefel[1,2], and Claude Baron[2]

[1] Toulouse University Mines Albi, [2] Toulouse University INSA, France
{paul.pitiot,michel.aldanondo,elise.vareilles}@mines-albi.fr,
{paul.gaborit,meriem.djefel}@mines-albi.fr,
{meriem.djefel,claude.baron}@insa-toulouse.fr

Abstract. This communication aims to propose a two step interactive aiding system dealing with product configuration and production planning. The first step assists interactively and simultaneously the configuration of a product and the planning of its production process. Then a second step complete the two previous tasks thanks to a constrained multi-criteria optimisation that proposes to the user a set of solutions belonging to a Pareto front minimizing cost and cycle time. The first section of the paper introduces the problem. The second one proposes a solution for the first step relying on constraint filtering for both configuration and planning. The following ones propose an evolutionary optimisation process and first computation results.

Keywords: product configuration, process planning, constraint satisfaction problem, evolutionary algorithm.

1 Introduction

This paper presents an integrated support tool which at first allows interactive configuration of a product and interactive planning and scheduling of its production process, and then minimizes conflicting criteria cost and cycle time. The configuration of a private aircraft will be used as an example to illustrate our research work.

In literature, most of the research into product configuration and production planning treats them independently. However, the decisions of product configuration obviously have strong consequences on the planning of its production process (for example, a luxury finish requires at least two additional months. On the other hand, planning decisions can provide hard constraints to product configuration (for example, such assembly duration forbids the use of such a kind of engine). Therefore, we propose to associate these two problems so that (i) the consequences of each decision of product configuration can be propagated toward the planning of its production process and (ii) the consequences of each process planning or scheduling decision can be propagated towards the product configuration. As we target interactive assistance in order to allow some kind of "what if" operating mode, we need to be able to

H. Papadopoulos, A.S. Andreou, and M. Bramer (Eds.): AIAI 2010, IFIP AICT 339, pp. 238–245, 2010.

show the consequences of each user's elementary requirement. A user's elementary requirement can be defined as a restriction of the domain of a variable involved in configuration (for example, number of seats belongs to [6, 12]) or in planning (for example, due date is prior to 31/10/2010). We consequently do not intend to process all the requirements simultaneously in a single shot to get a solution for both problems but rather to progressively lead the user to a solution for both product configuration and planning of its production.

In the field of configuration, many authors, among whom [1] or [2], showed that product configuration can be efficiently modeled and aided when it is considered as a Constraints Satisfaction Problem (CSP). In a same way, authors interested in planning and scheduling as [3] or [4] have shown that theses problems could be also modeled and aided when considered as a CSP. We therefore propose to consider configuration and planning problems as two constraint satisfaction problems. We assume that a constraint based model of a generic product and the 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. We also limit the scope of this paper to infinite capacity planning. To support interactive assistance, we only use the filtering or constraint propagation capabilities of the CSP framework. We finally link the two problems (configuration and planning) and the coupling constraints proposed in [5] together to propagate the consequences in both directions.

In the previous system, a product can be entirely configured and its production process entirely planned. "Entirely" means to restrict the solution space to a single solution, each problem variable having a single value. But we are not interested in this operating mode. We assume that it is possible to decompose the set of user's requirements in two sub-sets: non-negotiable requirements and negotiable ones. Our idea is to process interactive configuration and planning with the first sub-set or requirements only (non negotiable) and achieve a first reduction of the solution space. Remaining variable affectations (remaining solution space) are kept for multi-criteria considerations in the second step of our proposition.

In most industrial cases, the resulting products configured are characterized by criteria such as performance and product cost while relevant production plans are associated with cycle time and production cost. A solution is always a compromise of somehow contradictory criteria. In this presentation we only consider two criteria: cost (product cost and production cost) and cycle time for production planning. Hence, the next step is to find solutions that belong to the Pareto front (time/cost) among the solution space restricted during the first step. Multi-criteria optimization techniques and more accurately Evolutionary Algorithm (EA) (see [6] and [7]) have the advantage to avoid the aggregation of criteria and can provide solutions on a Pareto front in a rather simple way. Thanks to an evolutionary approach, the second step will perform a second reduction of the solution space for both product and plan and provide solutions on the Pareto front. Finally, the user can finish the process by selecting the solution that fits his specific time/cost compromise. The next section describes our proposition for the first step with an example, then our proposition for the second step with an evolutionary approach is detailed.

2 Configuration and Planning Models and Constraint Processing

The configuration model (left part of figure 1) gathers product descriptive variables (for example: aircraft range, number of engines, type of finish…) and product cost variables (finish cost, engine cost…) that are either symbols or discrete numbers. Configuration constraints (for example black solid lines that can link aircraft range and engine type together) and cost definition constraints (for example grey solid lines that can link engine type and number of engines with engine cost) correspond most of the time with discrete tables showing allowed combinations of allowed values. In this discrete problem, the associated CSP is discrete and the filtering provided by arc consistency technique [8] allows interactive configuration and cost estimation.

The planning model (right part of figure 1) gathers a set of planning operations (like manufacturing, assembling…) linked with ordering constraints. Each operation is defined with three operation temporal variables (starting date, ending date, possible duration) and eventually resource variables (required resource, quantity of required resource). We assume that the three temporal variables are real variables defined with intervals while resource type is symbolic and resource quantity a real variable. The cost of an operation is a real variable that depends on the resource type or quantity and the operation duration. As we consider planning with infinite capacity of resource, the constraints are as follows. Ordering constraints between operations (if task Y is after task X then starting date of Y is greater than or equal to ending date of X) and operation duration constraints (ending date equals starting date plus possible duration) are numerical constraints (black solid lines). The constraints that link possible duration with required resource and/or quantity of required resource and/or cost are mixed constraints (black and grey solid lines). Our numerical constraints are simple calculations (+,-,*, /, =, >, <), therefore they respect the hypothesis of bound Consistency proposed by Lhomme in [9]. Based on interval arithmetic, bound consistency is gathered with arc consistency and allows interactive planning and scheduling.

A mixed constraint between product description variables and planning operation variables (black doted lines) allows to propagate decision consequences from configuration to planning and from planning to configuration. [5] can be referred for more details. The first global criterion, i.e. total cost, is calculated thanks to a numerical constraint as the sum of product cost and operation cost (grey doted lines of figure 1). The second one, the production cycle time corresponds with the ending date of the last operation.

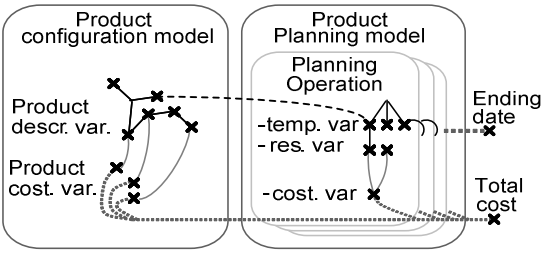

Fig. 1. Configuration and planning model

3 Optimization Problem to Assign

3.1 Definition of the Optimization Problem

The first interactive filtering step leads to the restriction of the initial feasible space (noted (a) on figure 2) to a restrained area (noted (b) on figure 2). This corresponds to the filtering of a customer's non-negotiable requirements. The filtering system provides domain bounds for every criteria variable (minimal and maximal values for cost and cycle time). The restrained area contains solutions corresponding to different remaining decisions to fulfil according to the remaining requirement. Notice that this area also contains unfeasible space where there is no solution due to the constraints of the problem to solve. The aim of the optimization process is to find a selection of the closest solutions to the Optimal Pareto front. Various metaheuristics may be used to solve this problem. In this research work, we focus on evolutionary algorithms for their ability to propose multiple solutions while solving a multiobjective problem. But classical evolutionary algorithms have to be adapted to take into account the constraints of the problem.

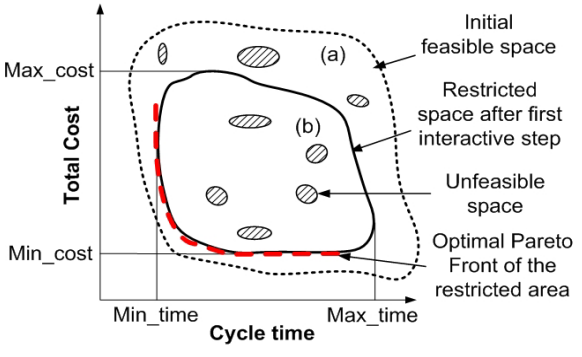

Fig. 2. Illustration of the reduced search space to optimize (b) from the initial feasible space (a) after the filtering process on non-negotiable requirements. Optimization process aims finding the Pareto-Optimal solutions in this restricted area.

3.2 Formal Definition of Optimization Problem

The constrained optimization problem (O-CSP) is defined by the quadruplet <V, D, C, f >, 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 minimize both cost and cycle time. The set V gathers: the product descriptive variables and the resources variables (required resource and choice or quantity of required resource). Constraints, as seen in section 2, link the variables of V with all the variables of the two problems. The filtering system allows dynamically computing modification on the domain of variables. This OCSP is a difficult problem to solve. The existing methods to handle constraints in EA are often computationally expensive. The next section briefly presents existing.

3.3 Overview of Existing Constrained Optimisation Approach

Initially, EA deal with large combinative unconstrained problem. They are used to solve multiple problems like project planning or project configuration. But real world problems are generally constrained. Many research studies try to integrate constraints in EA. C. Coello Coello comprehensively surveys the state of the art of these methods [10] (more than eight hundred references). It seems that classical EA performance decreases while the number of constraints increases. Constraints imply feasible and un-feasible areas in search space. Then the ratio between the size of feasible and un-feasible space guides the choice of specific methods to handle constraints in EA [11]. Four kinds of methods deal with this problem: penalty functions [12], repair methods [13], approaches that separate objectives and constraints (Multiobjective Optimization (MO) techniques) [14] and specific representations or operators [15], [16].

The first one, penalty functions, is the most common way to integrate violation of constraints in the objective function. For each individual, the level of violation of constraint is added to the fitness function during the evaluation phase. The aim is to increase the probability of selection for feasible or near feasible solutions. 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. The repair 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 a performing repairing algorithm that preserves the diversity of individuals. The same problem appears with MO approaches. These ones integrate the satisfaction of each constraint (or a group of constraints) as a specific objective.

Finally, the specific operators or representations approaches aim at preserving the feasibility of the individuals during their construction. Kowalczyk previously proposed in [15] the use of constraint consistency to prevent variable instantiations which are not consistent with the constraints of the problem. However, his study suggests that the supplementary computing time needed for constraint propagation may be very expensive in comparison with optimization process.

In this paper, we will focus on specific evolutionary operators that prune search space using constraints filtering (with our own filtering system named Cofiade while Kowalczyk used Ilog-solver, a commercial software, see [17] for a software comparison). The arc-consistence is less time consuming than other CSP solving system but it doesn't guaranty global coherence of the model.

3.4 Proposed Approach

Overview of modified EA used. EA used is adapted from the SPEA2 method [18] with classical evolutionary steps (initialisation, evaluation, selection, stopping criterion and perturbation operators). It is one of the most useful Pareto-based methods founded on preservation of a selection of best solutions in a separate archive. It includes performing evaluation strategy witch bring a well-balanced population density on each area of search space, and uses an archive truncation process that preserve boundary solution. It ensures both a good convergence speed and the preservation of diversity of

solutions. Zitzler shows that the maximal complexity of the overall algorithm is $O(N_{pop}^3)$ with N_{pop} the number of solutions in current population. We completed this method with specific evolutionary operators (initialisation, uniform mutation and uniform crossover) presented in next sections. Finally, the stopping criterion is a fixed number of generations.

Initialisation operator. Thanks to this operator, a well-diversified set of initial individuals is obtained. For each individual to create, every gene (decision variable) is randomly instantiated into its current domain. In order to avoid the generation of unfeasible individuals, the domain of every gene is dynamically updated by filtering after each instantiation. If an individual is incoherent, a limited backtrack process cancels one of the previous choice, then the individual is filtered again until the values of the remaining variable are consistent with the constraints. If the backtrack limit is reached, the individual is abandoned to bound the number of backward step and thus the computational time spent by the filtering. This process (random instantiation then filtering) is repeated until all the genes of every individual are instantiated.

Uniform mutation operator. This operator introduces a random perturbation on the evolutionary process that allows escaping from sub-optimal areas and thus the exploration of search space. It modifies the instantiation of some genes on individuals selected according to the mutation probability. Every individual and their genes selected for the mutation are beforehand chosen randomly. The individuals are selected then selected genes are un-instantiated. The filtering system updates the domain of these variables according to the instantiation of others genes. Finally, the mutation of the selected genes is achieved in the same way as during the initialisation (instantiation, filtering and backtrack limit). When every gene is instantiated, the mutation process end and the individuals are added to the next generation.

Uniform crossover operator. This operator allows to shuffle randomly and uniformly the genes of two individuals (named parents) selected according to the crossover probability. As for the mutation operator, couple of parents and crossover of their genes are randomly selected before the crossover operation. To achieve this task, a crossover table (table of crossover flags for every gene) is filled. It allows to select beforehand which genes will be exchange between parents. Indeed, the crossover corresponds to a selected way on a binary tree where each branch is linked to the crossover of a particular gene. An instantiation of the crossover table is equivalent to the selection of a path on the crossover tree. Notice that position of a specific gene is chosen randomly to avoid dominance of genes by their position in chromosome. During crossover operation, the system tries to achieve random crossover. At every gene instantiation, the filtering system updates the domain of remaining genes. The crossover table is initially filled identically for both child, but if an individual become unfeasible, a specific backtrack is done by changing some crossover flag in the table. A supplementary flag is added to the crossover table that memorise unfeasible way on the tree in case of backtrack. Then a backtrack counter limits the number of backward steps. If the backtrack limit is reach, the corresponding child is abandoned. Finally, every feasible child is added to the next generation.

4 Preliminary Results

We perform various tests on the aircraft problem[1] (42 variables, ten of whom are taken into account in EA, 42 constraints, about two millions feasible solutions) to evaluate the proposed approach. Various evolutionary settings were investigated. In most runs, the modified EA was able to provide very good solutions, near Pareto-optimal solutions. Figure 3 presents the Pareto-optimal front and the archive founded after one run of the proposed EA (10 generations, population size: 20, archive size: 21, crossover rate: 0.7, mutation rate: 0.3). During this run, 71.4% of the optimal Pareto front is reach for a computation time of 1730 seconds (around 28 minutes). This result is very interesting, because it clearly shows that in a reasonable amount of time, it is possible to propose a set of solutions that permit the user to decide about his own compromise cost/cycle time. This is much more accurate than solutions frequently proposed in industrial configuration software that are based on the selection of default values that try to minimize either cost or cycle time.

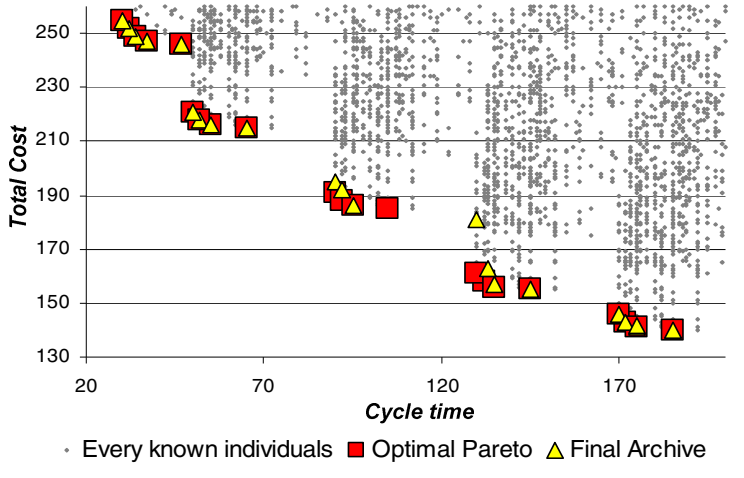

Fig. 3. Detail of archive resulting from one run of modified EA (triangles) and best known Pareto Front (squares) and every individual founded for this problem (points).

5 Conclusions

In this paper, we present an efficient aiding system for coupled product configuration and project planning, using an interactive constraint filtering system then a modified evolutionary optimization system. The modified EA is based on interaction with the filtering system that prunes the search space and thus reduces the search effort by limiting it to the feasible individuals. Standard evolutionary operators are adapted to take advantage of filtering. First experiments indicate that this method is well adapted

[1] Aircraft model could be investigated on-line, select model Aircraft-CSP-EA at : http://cofiade.enstimac.fr/cgi-bin/cofiade.pl

for an interactive aiding system. It generates near-optimal Pareto-solutions in reasonable computing time. It allows the user to decide efficiently about his cost/cycle-time compromise when dealing simultaneously with configuration and planning.

References

1. Soininen, T., Tiihonen, T., Männistö, T., Sulonen, R.: Towards a General Ontology of Configuration. AIEDAM 12(4), 357–372 (1998)
2. Junker, U.: Handbook of Constraint Programming, ch. 26. Elsevier, Amsterdam (2006)
3. Dechter, R., Meiri, I., Pearl, J.: Temporal Constraint Satisfaction Problems. Artificial Intelligence (49), 61–95 (1991)
4. Barták, R., Salido, M., Rossi, F.: Constraint satisfaction techniques in planning and scheduling. J. Intell. Manuf. 21, 5–15 (2010)
5. Vareilles, E., Aldanondo, M., Djefel, M., Gaborit, P.: Coupling interactively Product and Project Configuration: a Proposal using Constraints Programming. In: IMCM PETO 2008, Copenhagen, Denmark (2008) ISBN: 978-87-90855-12-3
6. Li, B., Chen, L., Huang, Z., Zhong, Y.: Product configuration optimization using a multiobjective GA. I.J. of Adv. Manufacturing Technology 30, 20–29 (2006)
7. Chelouah, R., Baron, C., Zholghadri, M., Gutierrez, C.: Meta-heuristics for System Design Engineering. Studies in Computational Intelligence, vol. 203, pp. 387–423. Springer, Heidelberg (2009)
8. Bessiere, C.: Handbook of Constraint Programming, ch. 3. Elsevier, Amsterdam (2006); Richardson, J.T., Palmer, M.R., Liepins, G., Hilliard, M.: Some guidelines for GA with penality functions. In: Schaffer, J.D. (ed.) Proc. of 3rd int. conf. on G.A., pp. 191–197 (1989)
9. Lhomme, O.: Consistency techniques for numerical CSPs. In: IJCAI 1993, Chambéry France, pp. 232–238 (1993)
10. Computer science department: CINVESTAV, `http://www.cs.cinvestav.mx/~constraint/`
11. Coello Coello, C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
12. Richardson, J.T., Palmer, M.R., Liepins, G., Hilliard, M.: Some guidelines for GA with penality functions. In: Schaffer, J.D. (ed.) Proc. of 3rd int. conf. on G.A., pp. 191–197 (1989)
13. Salcedo-Sanz, S.: A survey of repair methods used as constraint handling techniques in evolutionary algorithms. Computer science review, 175–192 (2009)
14. Clevenger, L., Ferguson, L., Hart, W.E.: Filter-based Evolutionary Algorithm for Constrained Optimization. Evolutionary Computation 13(3), 329–352 (2005)
15. Kowalczyk, R.: Constraint Consistent Genetic Algorithms. In: Proc. of IEEE conf. on evolutionary computation, pp. 343–348 (1997)
16. Michalewicz, Z., Nazhiyath, G.: Genocop III: A co-evolutionary algorithm for numerical optimization with non linear constraints. In: Fogel, D.B. (ed.) Proc. of the second IEEE conf on evolutionary computation, pp. 647–651 (1995)
17. Vareilles, E., Carbonnel, S., Djefel, M., Aldanondo, M., Rochet, S., Auriol, G., Baron, C.: Coupling Product and Project Configuration with Constraints: a CSP Software Comparison. In: International Conference SKIMA 2009 (2009)
18. Zitzler, E., Laumanns, M., Thiele, L.: 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)