Optimization of Product Instantiation using Integer Programming

Pim van den Broek
Department of Computer Science
University of Twente
Enschede, the Netherlands
pimvdb@ewi.utwente.nl

Abstract—**We show that Integer Programming (IP) can be used as an optimization technique for the instantiation of products of feature models. This is done by showing that the constraints of feature models can be written in linear form. As particular IP technique, we use Gomory cutting planes. We have applied this technique to a test suite of feature models from the literature, and found that the Gomory cutting planes can be used to improve the feature models. We discuss a number of applications: analysis of feature models, resolving configuration errors and optimization of product instantiation.**

Keywords-feature models; integer programming; product instantiation; linear constraints

I. Introduction

To analyze properties of software product lines which are specified by feature models, it is customary to map feature models to other data structures: Benavides et al. [1] use Constraint Satisfaction Problems, Batory [2] uses Logic Truth Maintenance Systems and Satisfiability Solvers, Czarnecki and Kim [3] use Binary Decision Diagrams and Van den Broek and Galvão [4] use Generalized Feature Trees. A recent survey of these analysis methods has been given by Benavides et al. [5]. When one wants to obtain an optimal product of the feature model, i.e. a product that best suits a number of criteria, each of the approaches mentioned above supports the possibility to iterate through the set of all products; however, none of them supports optimization without iterating through all products. This means that optimization is not possible when the number of products is prohibitively high. In [1], for instance, it is reported that for some moderate-sized feature models it is impossible to determine the number of products by just counting them. Therefore, there is a need for finding optimal products by an optimization procedure which does not iterate through all products.

The main contribution of this paper is the observation that the constraints of feature models can be written in a linear form. This opens up the possibility of optimization using integer programming [6]. Integer programming (IP) is an extension of linear programming (LP), where the additional constraint is added that the solutions are binary, i.e. the variables take only the values 0 and 1. In this paper, we consider the method of Gomory cutting planes [7]. When we apply the simplex algorithm to obtain an optimal solution of a linear programming problem, and we obtain a solution which is non-integer, Gomory cutting plane method provides a new constraint which is satisfied by all integer solutions, but cuts off the obtained non-integer solution. The simplex algorithm (or, more efficiently, its dual) may then proceed to find a new optimal solution which also satisfies the cutting plane constraint. This process continues until, in a finite number of steps, an integer solution is found.

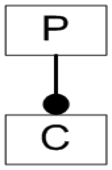
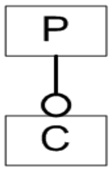
We have applied this procedure to the test suite of feature models given by Segura et al. [8]. We applied the simplex algorithm to obtain the products with maximum resp. minimum number of features. For 21 of the 24 feature models the simplex algorithm obtained both optimal products, or found that no products do exist. For 3 of the 24 models the simplex algorithm obtained a non-integer solution; a single Gomory cutting plane was sufficient to obtain the optimal products. Remarkably, in all three cases the cutting planes gave information with which the feature model could be "improved". For the improved feature models the simplex algorithm obtained the optimal products. So, after improving the feature models, the simplex algorithm did not give any non-integer solution.

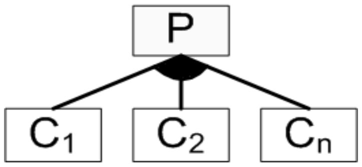
We will give several possible applications of our approach. We will show how to obtain the set of dead features and the set of fully mandatory features of a feature model. We will show how the method of White et al. [9], which determines the smallest set of changes which transforms an incorrect specification into a valid product, can be improved. As a new application, we discuss how, using our approach, an optimizing product configurator can be designed.

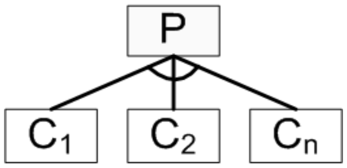
This paper is organized as follows. In the next section we present the linear form of the constraints of feature models. In section 3 we present our approach to derive optimal products of feature models using IP with Gomory cutting planes. In section 4 we apply our approach to the test suite of feature models in [8]. Section 5 discusses applications and section 6 concludes the paper.

II. Linear Constraints for Feature Models

In this section we discuss linear constraints for feature models. An overview is given in table 1. The column labeled CSP in this table shows the conventional mapping of feature models to Constraint Satisfaction Problems, taken from [5].

	relationship	CSP	Linear constraints
mandatory		$P = C$	$P = C$
optional		IF $(P = 0)$ $C = 0$	$C \leq P$

or		IF $(P > 0)$
		$1 \leq \Sigma_i C_i \leq n$
		ELSE
		$C_1 = C_2 = .. = C_n = 0$
			$\forall i \in [1..n]: C_i \leq P$
			$\Sigma_i C_i \geq P$

alternative		IF $(P > 0)$ $\Sigma_i C_i = 1$
		ELSE
		$C_1 = C_2 = .. = C_n = 0$
			$\Sigma_i C_i = P$

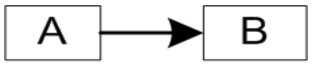
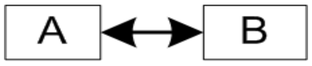
requires		If $(A > 0)$ $B > 0$	$A \leq B$
excludes		If $(A > 0)$ $B = 0$	$A + B \leq 1$

TABLE I. MAPPING FROM FEATURE MODELS TO LINEAR CONSTRAINTS

The last column shows our mapping of feature models to linear constraints. To each feature there corresponds a binary variable; a product of the feature model corresponds to a valuation of this set of variables; a value 1 indicates that the feature is present in the product, a value 0 indicates that the feature is not present in the product.

- Mandatory features: Since a mandatory child C of parent P is present if and only if P is present, the constraint is $P = C$.
- Optional features: The only constraint on the presence of a child C is the presence of its parent P; this can be linearly expressed as $C \leq P$.
- Or group of features: The constraints for an or-group of n features can be expressed in linear form by $C_1 \leq P$, $C_2 \leq P$,..,$C_n \leq P$ and $C_1+C_2+..+C_n \geq P$. The first n constraints state that if the parent P is absent ($P = 0$), all children should be absent. The last constraint states that if the parent P is present ($P = 1$) then at least one of the children $C_1, C_2, .., C_n$ should also be present.
- Alternative group of features: The condition for an alternative-group of features can be expressed in linear form by $C_1+C_2+..+C_n = P$. It states that if the parent P is not present ($P = 0$) none of the children should be present, and that if the parent is present ($P = 1$), exactly one of the children should be present.
- Requires constraint: The linear constraint $A \leq B$ expresses that if A is present, B is present as well.

- Excludes constraint: The linear constraint $A + B \leq 1$ expresses that A and B cannot be both present.

Using these rules, a feature model can be mapped to a set of linear constraints. To this set the constraint $R = 1$, where R is the root feature, should be added; this expresses that the root feature should be present in every product. The products of the feature model correspond to the binary solutions (0 or 1) for the feature variables.

III. Integer Programming

Given a set of linear constraints of a feature model and a linear expression in the feature variables which is called the goal-function, the problem of finding the binary solution of the constraints for which the goal-function is maximal (or minimal) is an integer programming problem. The problem which arises by ignoring the condition that the solution be binary, is called the linear relaxation (LR) of the problem, and is a LP problem. The LR can be solved using the Simplex Algorithm. This algorithm provides the optimal value of the goal-function, and a set of values for the feature variables for which this optimal value is attained. These values of the feature variables will be non-negative, but not necessarily equal to 0 or 1.

Due to the feature model constraints, for each feature F we have $F \leq 1$. This is because the value of the root is equal to 1, and $C \leq P$ for each child feature C of a parent feature P.

If the solution of the LR is an integer solution, then this solution is the optimal solution of the IP problem. If the solution of the LR is a non-integer solution, then there are multiple ways to proceed [6]. We have chosen for Gomory's cutting plane method [7]. Given a non-integer solution of the LR, this method provides a new constraint, called a Gomory cutting plane, such that all integer solutions satisfy it, but the obtained non-integer solution does not. This new constraint, can then be added to the set of constraints. The simplex algorithm (or, more efficiently, its dual) may then proceed to find a new optimal solution which satisfies the updated constraint set. This process continues until, in a finite number of steps, an integer solution is found; this integer solution is the opimal solution of the original IP problem. An example of a Gomory cutting plane is given in the next section.

IV. Application to a Test Suite

We have applied our approach to a test suite of 24 feature models in [8]. These feature models are meant to test whether an analysis method correctly determines whether feature models are void or non-void. To test the voidness of a feature model with IP, any goal-function can be used. We chose to maximize the goal-function $GF = F_1+..+F_N$, so we determine, if the feature model has products, the product with the maximum number of features. It turns out that the LR gives the correct result for 22 feature models, and that for 2 feature models it gives a fractional result. For both cases where a fractional result was obtained, we determined a Gomory cutting plane[7]; after adding it to the constraints, the correct (non-fractional) result was obtained.

We then performed the same test by minimizing the given goal function, i.e. by determining the product with the lowest number of features. Again there were fractional solutions in two cases, for which one Gomory cutting plane was needed. In total 3 feature models led to fractional solutions in the two tests. They are depicted in figure 1.

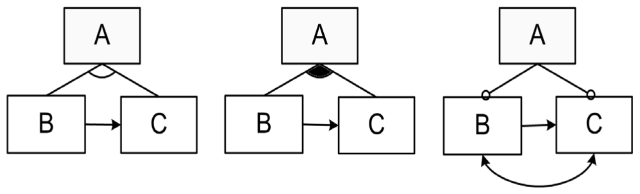


Figure 1. The feature models of the test suite with fractional solutions

What is apparent, is that all three feature models contain redundancies. In all cases, the Gomory cutting plane could be used to simplify the feature model. The simplified feature models are shown in figure 2.

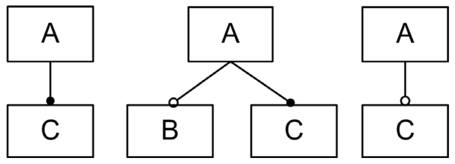


Figure 2. The simplified feature models of figure 1

So, the disadvantage of IP that the LR may give fractional solutions, is turned into an advantage, in that fractional solutions are used to obtain information to improve the feature model. It remains to be determined whether this just happens to be true for the small feature models in the test suite, or whether this is true in general.

We performed the tests with our own implementation of the simplex algorithm in the functional programming language Miranda [10]. We implemented Bland's rule [11], which guarantees termination; we used unbounded integers and rational numbers instead of real numbers, which guarantees exact computations.

It will be instructive to consider one of the cases in more detail. We consider the first of the feature models in figure 1, which is the only feature model which gave fractional results for both tests. The constraints for this feature model are:

$$A = 1$$
$$A - B - C = 0$$
$$B - C \leq 0$$

The first step toward their solution is to introduce an auxiliary variable D, with $D \geq 0$, defined by $D = C–B$, and to write the constraints as equalities:

$$A = 1$$
$$A - B - C = 0$$
$$B - C + D = 0$$

The simplex algorithm, when maximizing the goal-function $GF = A + B + C$, transforms these into

$$A = 1$$
$$B + D/2 = 1/2$$
$$C - D/2 = 1/2$$

From this form of the constraints the fractional optimal solution A=1, B=1/2, C=1/2, D=0 is obtained. From the second constraint, since B must be integer, it follows that D/2 – 1/2 must be integer. Since D is non-negative we obtain the Gomory cutting plane $D/2 - 1/2 \geq 0$, which can be written as $D \geq 1$. This constraint can be added to the constraint set, and cuts off the earlier fractional optimal solution. However, instead of continuing with the simplex algorithm, we turn to the feature model and find that the new constraint reads $C–B \geq 1$ which implies C = 1 and B = 0; so B is a dead feature and C is a mandatory feature. The improved feature model is the first feature model in figure 3.

In the same way the other two feature models of figure 3 are obtained as improvements of the feature models of figure 2. For the second feature model we obtain the Gomory cutting plane $A–2C \leq -1$, from which we conclude that C=1; for the second feature model we obtain the Gomory cutting plane $B \leq 0$, which implies B = 0. When in the test suite of 24 feature models the feature models of figure 2 are replaced by the feature models of figure 3, the LR for the two tests give the correct solution in all cases.

V. Applications

In this section we present some possible applications of our approach.

A. *Analysis of Feature models*

Suppose for some feature model we want to determine the set of dead features, i.e. the set of features which do not occur in any product. We maintain a set Candidates of features which might eventually be dead. We initialize this set to contain all features. As goal-function, we take the sum of all feature-variables of the set Candidates, and obtain the product for which this goal-function is maximal. The features of this product are not dead, and they are deleted from the set Candidates. This is repeated until the optimal value equals 0 or Candidates is empty. The features which are still in Candidates are the dead features. In pseudo-code:

```
Set Candidates := set of all features
REPEAT
  Goal G = sum of variables of features in
                              Candidates;
  Product P = product for which G is maximal;
  Boolean Stop = Optimal Value = 0 OR Candidates = ∅
  IF Not Stop THEN
      Set Features = set of features of P
      Candidates := Candidates -- Features
UNTIL Stop
```

Likewise we can determine the set of fully mandatory features, i.e. the set of features which occur in every product:

```
Set Candidates := set of all features
REPEAT
  Goal G = sum of variables of features in
                              Candidates;
  Product P = product for which G is minimal;
  Set Features = set of features of P
  Set Remaining = Candidates -- Features
  Boolean Stop = #Candidates = #Remaining
  IF Not Stop THEN
      Candidates := Remaining
UNTIL Stop
```

B. *Configuration Errors*

White et al. [9] have given a method to determine the smallest set of changes to an incorrect configuration which turns it into a correct one. Here a change is either the selection of a non-selected feature or the deselection of a selected feature. The authors formulate the problem as a constraint satisfaction problem, where for each feature they introduce 3 new variables and 2 new constraints (in a simplified version, they introduce for each feature 2 new variables and 1 new constraint).

Using IP, the problem can be solved without introduction of new variables and new constraints. Let $F_1, .. , F_k$ be the features which are selected in the incorrect configuration, and let $F_{k+1}, .. , F_N$ be the non-selected features. Let the goal-function be $GF = F_1 + .. + F_k - F_{k+1} - . . - F_N$. Maximizing this goal-function with IP gives the product which resembles the incorrect specification as good as possible, i.e. the incorrect configuration can be changed into it with a minimal number of changes.

In [8], a sequence of 71 feature models with configurations are given, as a test sequence to test the correctness of algorithms which determine the validity of the configurations. We have applied IP to this test sequence, not just to determine the validity of the configuration, but to determine the most resembling product, using the goal-function mentioned above. With this approach, the LR obtained the correct result in all cases except for the feature models of figure 2, which required, again, the addition of one Gomory cutting plane.

C. *Optimization of Product Instantiation*

State-of-the-art product configurators perform a dialogue with a user, who can choose which features he/she wants as part of his/her product [12,13,14]. The configurators take care not to present the user with options which cannot be realized anymore, due to constraints from previous choices. Also, the user may undo decisions when they prevent other choices to be made [15].

What is not possible, up to now, is to take into account the importance for the user of the decisions he makes. Some features he might want to have more than others, and some he wants more to be absent than others. Here we envisage a configurator which determines the importance for the user of the presence/absence of features, and then determines the optimal configuration, i.e. the configuration which matches the user's wishes as good as possible. No configuration errors can arise in this approach; however, the configured product may not satisfy all the user's wishes, but the deviation will be as low as possible. The problem being an optimization problem, the best way to attack is, of course, with an optimization approach, like the one advocated in this paper.

Let α_F be equal to 1 if the user wants feature F in the product, and equal to –1 if he/she does not want the feature in the product. Let β_F denote the degree of importance the user assigns to the product conforming to his wish (either presence or absence) regarding feature F. This degree of importance may range from 0, denoting indifference, to 1, denoting maximal importance. Let the goal-function be

$$GF(F_1,..,F_n) = \alpha_{F1} * \beta_{F1} * F_1 + .. + \alpha_{Fn} * \beta_{Fn} * F_n.$$

Applying IP to maximize the goal-function GF will yield the product which matches the user's wishes as good as possible.

Not every feature needs to be present in the goal-function; absence of a feature F means that the user is indifferent about the presence of F ($\beta_F = 0$). If the user wants feature F, then the presence of F in the product contributes β_F to the goal function, and if the user does not want feature F, then the presence of F in the product contributes $-\beta_F$ to the goal function. To elicitate the proper degrees of importance of the features from the user, techniques from the field of multi-criteria decision making [16] can be used.

VI. Conclusion

An optimization problem is a problem of determination the maximum (or minimum) of a function of several variables subject to a number of constraints. In the case of feature models, determination of an optimal product is such an optimization problem. Up till now, to the best of our knowledge, the only optimization procedures which have been applied to feature models are procedures which cycle through the whole set of products, as for instance in [9]. In this paper, we have shown that the constraints of feature models can be written in linear form, which opens the possibility to apply optimization procedures from the domain of integer programming to feature models. Application on a test suite of (small) feature models has revealed that relaxation to linear programming gives immediate solutions in most cases and that in the cases where non-integer optimal solutions are obtained, Gomory cutting planes can be used to improve the feature models. We have discussed several applications: analysis of feature models, resolving configuration errors and optimization of product instantiation.

References

[1] D. Benavides, P. Trinidad and A. Ruiz-Cortés, "Automated Reasoning on Feature Models", in: O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, Lecture Notes in Computer Science 3520, Springer-Verlag Berlin Heidelberg, pp. 491-503, 2005.

[2] D. Batory, "Feature Models, Grammars, and Propositional Formulas", in: H. Obbink and K. Pohl (eds.): Software Product Lines Conference 2005, Lecture Notes in Computer Science 3714, Springer-Verlag Berlin Heidelberg, pp. 7-20, 2005.

[3] K. Czarnecki and P. Kim, "Cardinality-based Feature Modeling and Constraints: A Progress Report", in: Proceedings of the International Workshop on Software Factories, OOPSLA 2005.

[4] P. van den Broek and I. Galvão, "Analysis of feature models using generalised feature trees". In: D. Benavides, A. Metzger and U. Eisenecker (eds.), 3th International Workshop on Variability Modelling of Software-intensive Systems, ICB-Research Report 29, University of Duisburg-Essen, pp. 29-36, 2009.

[5] D. Benavides, S. Segura and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: a literature review". Information Systems, in press, 2010.

[6] G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, New York, 1988.

[7] R.E. Gomory, "Outline of an Algorithm for Integer Solutions to Linear Programs", Bulletin of the American Mathematical Society 64, pp. 275-278, 1958.

[8] S. Segura, D. Benavides and A. Ruiz-Cortés, "FaMa test suite v1.2", Technical Report ISA-10-TR-01, Applied Software Engineering group, University of Seville, Spain, 2010.

[9] J. White, D. Benavides, D.C. Schmidt, P. Trinidad, B. Dougherty and A. Ruiz-Cortés, "Automated Diagnosis of Feature Model Configurations", Journal of Systems and Software 83, pp 1094-1107, 2010.

[10] D. Turner, "Miranda: a non-strict functional language withpolymorphic types", in: Functional Programming Languages and Computer Architecture, Lecture Notes in Computer Science Vol 201, J.-P. Jouannaud (ed.), Springer-Verlag, pp. 1-16, 1985.

[11] R. Bland, "New finite pivoting rules for the simplex method", Mathematics of Operations Research 2, pp. 103-107, 1977.

[12] R. Buhrdorf, D. Churchett, C. Krueger, "Salion's Experience with a Reactive Software Product Line Approach". In: Proceedings of the 5th International Workshop on Product Family Engineering, Lecture Notes in Computer Science Vol. 3014, Springer-Verlag, pp. 317-322, 2004.

[13] D. Beuche, "Variant Management with Pure:: variants". Tech. rep., Pure-Systems GmbH, http://www.pure-systems.com, 2003.

[14] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés, "FAMA: Tooling a framework for the automated analysis of feature models". In: Proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS2007), 2007.

[15] A. Nöhrer and A. Egyed, "Conflict resolution strategies during product configuration". In: Proceedings of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS2010), 2010.

[16] E. Triantaphyllou, Multi-criteria decision making methods: a comparative study, Kluwer Academic Publishers, 2000.