Interactive configuration with constraints consistency and recommendation

Hélène Fargier
Pierre-François Gimenez
Jérôme Mengin

IRIT-CNRS
University of Toulouse

12th June 2018
Complex products

Complex, highly customizable products (combinatorial domains)
→ **cars**, computers, travels, kitchens. . .
→ number of possibilities exponential in the number of configuration variables
→ all products aren’t feasible (like a convertible car with a sunroof)
The constraints are hard: some products are infeasible

They come from:

- technical limitations (no sunroof on a convertible car)
- commercial considerations (no leather wheel on a lower-end car)
- stock variability (out-of-stock item)
- etc.

Renault Master: $10^{21}$ cars, $10^{16}$ feasible cars
Interactive configuration process

Product construction: the interactive configuration process

- the user chooses a configuration variable
- the configurator proposes possible values
- the user chooses a value for this variable

This process continues until the product is fully defined

Every proposed value must lead to a possible vehicle, but it’s an NP-hard problem! Two techniques:

- constraints propagation [Wal72]
- compilation [AFM02]
At each step of the interactive configuration, there is a partial, ongoing configuration

Recommendation $= \text{recommend}$, given a partial configuration $u$, a value for a variable $\text{Next}$

A good recommendation is:
- accurate
  $\rightarrow$ the user is willing to accept
- quick
  $\rightarrow$ on-line application
- We have a sales history from Renault, no other information → no information about the user
- The user chooses the variables one by one → the order of the variables is unknown
- There are constraints on allowed configurations → we use the *SaLaDD* compiler [Sch15]
- The sales history products may or may not satisfy the constraints
Recommendation in interactive configuration not very studied

Two categories of tools:
- $k$-nearest neighbours [CGO$^+$02]
- Bayesian network

Goal: experiment and compare these methods
Outline

1. Context and issue
2. Algorithms
   1. based on Bayesian networks
   2. based on \( k \)-nearest neighbours
3. Experiments
4. Conclusion
Bayesian networks represent a probability distribution on the configurations by means of a direct acyclic graph (DAG) and probability tables

- Each node is a variable
- An edge between $A$ and $B$ means that the probability of $A$ depends on the value of $B$ (and vice-versa)

```
Bayesian network

Manual ← Engine ← #pass. ← Colour ← Leather → Rim
```
How to recommend with a Bayesian network?

Probability $p(o)$ that a car $o$ will be bought

Our recommendation is based on:

$$\text{argmax}_{x \in \text{Next}} p(\text{Next} = x \mid \text{Assigned} = u)$$

$\text{Next}$ is the configuration variable chosen by the user, $u$ the partial configuration

We assume the sales history are a representative sample of future user choices

Two phases:

- Learn a Bayesian network from the sales history off-line
  $\rightarrow$ constraints aren’t taken into account during the learning
- Recommend a value of the conf. variable on-line
  $\rightarrow$ the learning isn’t critical, the inference is
Naive Bayesian network: special case of Bayesian network with strong assumptions of independence

+ inference is quick

— roughly approximates the real probability distribution (less accurate)
3 algorithms based on $k$-nearest neighbours

Instead of using the whole sample, they use previous sales similar to the current one

The 3 algorithms process these neighbours in a different way
Three algorithms

Among the $k$-nearest neighbours of the current partial configuration

**Weighted Majority Voter:** each neighbours votes with a weight proportional to its similarity with the current configuration

**Naive Bayes voter:** uses the neighbours to learn a naive Bayesian network. No learning is possible off-line

**Most popular choice:** computes the most probable completion of the current configuration and recommend the value of $\text{Next}$ in it
Experimental protocol

10 folds cross-validation: history sales split into a training set and a test set

- **Training set:** Bayesian networks learning / neighbours searching
- **Test set:** for each item we simulate a configuration session
  For each recommendation for `Next`, we compare the recommended value with the value really chosen
  → Only one possible value: no evaluation
  → Recommended = chosen: success, else: failure

We measure the success rate and the recommendation time w.r.t. the number of assigned variables
Datasets from Renault

Experiments made on i5 processor at 3.4GHz, using one core
All algorithms are written in Java

- dataset “Renault-44”
  - 44 variables
  - 14786 examples, 8252 examples consistent with the constraints
  - 70.80% recommendations are trivial

- dataset “Renault-48”
  - 48 variables
  - 27088 examples, 710 examples consistent with the constraints
  - 71.73% recommendations are trivial

- dataset “Renault-87”
  - 87 variables
  - 17715 examples, 8335 examples consistent with the constraints
  - 46.89% recommendations are trivial
Error rate w.r.t. the number of assigned variables

Experiment on *Renault-44*: 44 variables, 14786 examples including 8252 examples consistent with the constraints
Recom. time w.r.t. the number of assigned variables

Experiment on *Renault-44*: 44 variables, 14786 examples including 8252 examples consistent with the constraints
Error rate w.r.t. the number of assigned variables

Experiment on *Renault*-48: 48 variables, 27088 examples including 710 examples consistent with the constraints
Recom. time w.r.t. the number of assigned variables

Experiment on Renault-48: 48 variables, 27088 examples including 710 examples consistent with the constraints
Experiment on *Renault-44*: 44 variables, 14,786 examples including 8,252 examples consistent with the constraints.
Summary

- Constraint compilation is usable on-line
- $k$-nearest neighbours and Bayesian networks are accurate and fast enough
- Naive Bayesian network is adapted when execution time is more critical than accuracy
- The presence of constraints reduces the accuracy


David L Waltz. 
Generating semantic descriptions from drawings of scenes with shadows. 
1972.