Your model

Trim level
Expression+  £16,600

Engine & gearbox
ENERGY TCe 130
Included

Your style

Colour
Caster white
Included

Wheels
18" DAKOTA alloy wheels
Included

Upholstery
Goth upholstery
Included

Your extras

Options
Included

Happy with your configuration?

2 YOUR CAR SUMMARY

RESTART YOUR CONFIGURATION

All-New MEGANE
Expression+ ENERGY TCe 130

Total price: £16,600

Load information: Car and image prices may vary according to optional equipment selected. Due to continuous product development prices and options shown may not always reflect the latest Renault data. Please check with your dealer for the latest information.
The Configurator as a Computational Problem

▶ in the applet
  ▶ configure parts of car (≈ 160 parts, some hidden, $10^{26}$ combinations)
  ▶ constraints on combinations (mostly marketing, $10^{21}$ possible cars)
  ▶ after partial choice by user, check for legal car (and minimal price, restrict future choices, take back choices, . . . )

▶ slightly abstracted
  ▶ input: CNF formula and partial assignment
  ▶ question: is there satisfying assignment?
  ▶ clearly NP-hard
  ▶ so is there a SAT/CSP-solver in this flash-applet? NO, because...
  ▶ not enough computational power
The Configurator as a Computational Problem

- in the applet
  - configure parts of car (∼ 160 parts, some hidden, $10^{26}$ combinations)
  - constraints on combinations (mostly marketing, $10^{21}$ possible cars)
  - after partial choice by user, check for legal car (and minimal price, restrict future choices, take back choices, ...)

- slightly abstracted
  - input: CNF formula and partial assignment
  - question: is there satisfying assignment?
  - clearly NP-hard
The Configurator as a Computational Problem

- in the applet
  - configure parts of car (≈ 160 parts, some hidden, $10^{26}$ combinations)
  - constraints on combinations (mostly marketing, $10^{21}$ possible cars)
  - after partial choice by user, check for legal car (and minimal price, restrict future choices, take back choices, . . .)

- slightly abstracted
  - input: CNF formula and partial assignment
  - question: is there satisfying assignment?
  - clearly NP-hard

- so is there a SAT/CSP-solver in this flash-applet? NO, because . . .
  - not enough computational power
  - want guaranteed response time
Knowledge compilation and d-DNNF

- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
Knowledge compilation and d-DNNF

- Knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- Here: encode constraints on cars in d-DNNF
Knowledge compilation and d-DNNF

- Knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- Here: encode constraints on cars in d-DNNF

\[
\bigvee \bigwedge x \bigwedge \neg y \bigvee \neg y
\]

- d-DNNF (deterministic decomposable negation normal form)
Knowledge compilation and d-DNNF

▶ knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
▶ here: encode constraints on cars in d-DNNF

\( \lor \land \land x \lor \neg y \land \lor \neg y \lor x \)

▶ d-DNNF (deterministic decomposable negation normal form)
  ▶ negation normal form: \( \neg \) only on inputs
Knowledge compilation and d-DNNF

- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF

\[
\text{\begin{tikzpicture}
\node (1) at (0,0) {$\lor$};
\node (2) at (-2,-2) {$\land$};
\node (3) at (2,-2) {$\land$};
\node (4) at (1,-4) {$\lor$};
\node (5) at (-2,-4) {$\neg$};
\node (6) at (2,-4) {$\neg$};
\node (7) at (0,-6) {$y$};
\node (8) at (-1,-6) {$\neg$};
\node (9) at (1,-6) {$x$};
\node (10) at (0,-8) {$y$};
\draw (1) -- (2);
\draw (1) -- (3);
\draw (2) -- (4);
\draw (2) -- (5);
\draw (3) -- (6);
\draw (4) -- (7);
\draw (4) -- (8);
\draw (4) -- (9);
\draw (6) -- (10);
\draw (8) -- (10);
\draw (9) -- (10);
\end{tikzpicture}}
\]

- d-DNNF (deterministic decomposable negation normal form)
  - negation normal form: $\neg$ only on inputs
  - decomposable: children of $\land$ have disjoint variables
Knowledge compilation and d-DNNF

- knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
- here: encode constraints on cars in d-DNNF

\begin{center}
\begin{tikzpicture}
  \node (v1) at (0,0) {$\lor$};
  \node (v2) at (-1,-1) {$\land$};
  \node (v3) at (1,-1) {$\land$};
  \node (x) at (0,-2) {$x$};
  \node (v) at (-1,-3) {$\lor$};
  \node (v4) at (1,-3) {$\lor$};
  \node (n) at (0,-4) {$\neg$};
  \node (y) at (-1,-5) {$y$};
  \node (x2) at (1,-5) {$x$};
  \node (y2) at (0,-6) {$y$};
  \draw (v1) -- (v2);
  \draw (v1) -- (v3);
  \draw (v2) -- (x);
  \draw (v3) -- (v);
  \draw (v) -- (v4);
  \draw (v4) -- (y);
  \draw (v4) -- (x2);
  \draw (n) -- (y2);
\end{tikzpicture}
\end{center}

- d-DNNF (deterministic decomposable negation normal form)
  - negation normal form: $\neg$ only on inputs
  - decomposable: children of $\land$ have disjoint variables
  - deterministic: every assignment satisfies at most one child of $\lor$
knowledge compilation: encode complicated solutions spaces into format that allows efficient reasoning
here: encode constraints on cars in d-DNNF

\[ \neg x \lor y \land \neg y \land x \lor \neg x \land y \land \neg y \lor \neg x \land \neg y \]

d-DNNF (deterministic decomposable negation normal form)
- negation normal form: \( \neg \) only on inputs
- decomposable: children of \( \land \) have disjoint variables
- deterministic: every assignment satisfies at most one child of \( \lor \)
- generalizes other data structures like OBDDs
Probabilistic Embedding

- Probability of having a mapping that maps edges on edges?
- Models the problem on probabilistic databases
- $\#P$-hard
Probabilistic Embedding

- Probability of having a mapping that maps edges on edges?
- Models problem on probabilistic databases
- #P-hard
Probabilistic Embedding

▶ probability of having mapping that maps edges on edges?
▶ models problem on probabilistic databases
▶ #P-hard
Probabilistic Embedding

- probability of having mapping that maps edges on edges?
- models problem on probabilistic databases
- #P-hard
The Practical Approach: Grounding

- Write down all potential matches, answer “yes” if and only if one of them is among 
  \{af, ad, df\}, \{bd, bc, dc\}, \{cd, ce, de\}.
- Translate to DNF:
  \((x \land af) \lor (x \land ad) \lor (x \land df)) \lor (x \land bd) \lor (x \land bc) \lor (x \land cd) \lor (x \land cd) \lor (x \land ce) \lor (x \land de)).
- Compute probability with model counter (typically CNF).
The Practical Approach: Grounding

- write down all potential matches, answer “yes” if and only if one of them there

\[\{af, ad, df\}, \{bd, bc, dc\}, \{cd, ce, de\}\]
The Practical Approach: Grounding

- write down all potential matches, answer “yes” if and only if one of them there
  \[
  \{af, ad, df\}, \{bd, bc, dc\}, \{cd, ce, de\}
  \]

- translate to DNF
  \[
  (x_{af} \land x_{ad} \land x_{df}) \lor (x_{bd} \land x_{bc} \land x_{cd}) \lor (x_{cd} \land x_{ce} \land x_{de})
  \]
The Practical Approach: Grounding

- write down all potential matches, answer “yes” if and only if one of them there

\[ \{af, ad, df\}, \{bd, bc, dc\}, \{cd, ce, de\} \]

- translate to DNF

\[ (x_{af} \land x_{ad} \land x_{df}) \lor (x_{bd} \land x_{bc} \land x_{cd}) \lor (x_{cd} \land x_{ce} \land x_{de}) \]

- compute probability with model counter (typically CNF)
Model Counters: Exhaustive DPLL

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)

- basis of state of the art model counters
Model Counters: Exhaustive DPLL

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)
- basis of state of the art model counters

Theorem (Huang, Darwiche ’05)

*Traces of runs of exhaustive DPLL are d-DNNF.*
Model Counters: Exhaustive DPLL

- Exhaustive DPLL
  - backtracking search to find/count solutions of CNF
  - caching (reuse counts of subformulas already solved)
  - components (independent formulas treated independently)
- basis of state of the art model counters

Theorem (Huang, Darwiche ’05)

Traces of runs of exhaustive DPLL are d-DNNF.

- so model counters can also compile: sharpSAT ⇝ Dsharp
- bounds on d-DNNF
  ≈ bounds on grounding approach to probabilistic embedding
Lower Bounds for d-DNNF

Theorem (Bova, Capelli, M., Slivovsky, IJCAI 2016)

There are CNF-formulas whose smallest d-DNNF encoding has exponential size.
Communication Complexity

Alice and Bob want to evaluate $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$

- Alice knows assignment to $x_1, \ldots, x_n$
- Bob knows assignment to $y_1, \ldots, y_n$

**Definition:** Communication Complexity $\text{CC}(f)$

- minimal number of bits that Alice and Bob have to exchange

**Examples:**
- $\text{CC}(\text{PARITY}) = 1$
- $\text{CC}(\text{EQUAL}(x_1, \ldots, x_n, y_1, \ldots, y_n)) = n$

- well studied, many variations, techniques and results
Communication Complexity

- Alice and Bob want to evaluate $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- Alice knows assignment to $x_1, \ldots, x_n$, Bob knows assignment to $y_1, \ldots, y_n$

**Definition: Communication Complexity $CC(f)$**

minimal number of bits that Alice and Bob have to exchange
Alice and Bob want to evaluate $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$

Alice knows assignment to $x_1, \ldots, x_n$, Bob knows assignment to $y_1, \ldots, y_n$

**Definition:** Communication Complexity $CC(f)$

minimal number of bits that Alice and Bob have to exchange

**Examples**

- $CC(PARITY) = 1$
- $CC(EQUAL(x_1, \ldots, x_n, y_1, \ldots, y_n)) = n$
Alice and Bob want to evaluate $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$

Alice knows assignment to $x_1, \ldots, x_n$, Bob knows assignment to $y_1, \ldots, y_n$

Definition: Communication Complexity $CC(f)$

minimal number of bits that Alice and Bob have to exchange

Examples

$CC(PARITY) = 1$

$CC(EQUAL(x_1, \ldots, x_n, y_1, \ldots, y_n)) = n$

well studied, many variations, techniques and results
Assume small OBDD for $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$. Cut OBDD in the middle. Alice sends the node she reaches on her half of the assignment. Therefore, $\log(\text{# nodes in middle}) \geq \text{CC}(f)$. 

10/16
communication protocol from small OBDD:

- assume small OBDD for $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- cut OBDD in the middle
- Alice sends node she reaches on her half of assignment
communication protocol from small OBDD:

- assume small OBDD for $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$
- cut OBDD in the middle
- Alice sends node she reaches on her half of assignment

$\Rightarrow \log(\# \text{ nodes in middle}) \geq CC(f)$
Generalization to d-DNNF

Connection (Bova, Capelli, M, Slivovsky ’16)

Functions with high multipartition communication complexity have no small d-DNNF.
Generalization to d-DNNF

Connection (Bova, Capelli, M, Slivovsky ’16)
Functions with high multipartition communication complexity have no small d-DNNF.

Theorem (essentially Duris et al. ’04)
There are CNF with high multipartition communication complexity.
Generalization to d-DNNF

Connection (Bova, Capelli, M, Slivovsky ’16)

Functions with high multipartition communication complexity have no small d-DNNF.

Theorem (essentially Duris et al. ’04)

*There are CNF with high multipartition communication complexity.*

- lets us show lower bounds for d-DNNF “easily” by using CC literature
- several variations of this for other data structures from knowledge compilation
Graphs and CNF

- assign graph to every CNF formula

\[ (\neg x_1 \lor x_2) \land (x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4) \]
Graphs and CNF

- assign graph to every CNF formula

\[(\neg x_1 \lor x_2) \land (x_2 \lor \neg x_3) \land (\neg x_2 \lor \neg x_3 \lor x_4)\]

- idea: if graphs are “easy”, maybe we can compile efficiently
Observation

Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

Proof (sketch).

Assume that CNF is monotone, so all clauses $x \lor y$

Compute for root $x$

d-DNNF $C_x,0$, $C_x,1$ for solutions when $x$ is 0/1

▶ only edge $xy$:

$C_x,0 = \neg x \land y$ and $C_x,1 = x$

▶ otherwise $x$ with children $y_1,...,y_s$

$C_x,o = \neg x \land C_{y_1},1 \land \cdots \land C_{y_s},1$

$C_x,1 = x \land (C_{y_1},0 \lor C_{y_1},1) \land \cdots \land (C_{y_s},0 \lor C_{y_s},1)$

compilation result $C = C_x,0 \lor C_x,1$
Observation
Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

Proof (sketch).
Assume that CNF is monotone, so all clauses $x \lor y$
### Observation

Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

### Proof (sketch).

Assume that CNF is monotone, so all clauses $x \lor y$

Compute for root $x \times d$-DNNF $C_{x,0}$, $C_{x,1}$ for solutions when $x$ is 0/1
Observation

Every 2CNF whose graph is a tree can be compiled into a d-DNNF of linear size.

Proof (sketch).

Assume that CNF is monotone, so all clauses $x \lor y$
Compute for root $x$ d-DNNF $C_{x,0}$, $C_{x,1}$ for solutions when $x$ is 0/1
- only edge $xy$: $C_{x,0} = \neg x \land y$ and $C_{x,1} = x$
- otherwise $x$ with children $y_1, \ldots, y_s$
  \[
  C_{x,o} = \neg x \land C_{y_1,1} \land \cdots \land C_{y_s,1}
  \]
  \[
  C_{x,1} = x \land (C_{y_1,0} \lor C_{y_1,1}) \land \cdots \land (C_{y_s,0} \lor C_{y_s,1})
  \]

compilation result $C = C_{x,0} \lor C_{x,1}$
Graph Decomposition Techniques and Knowledge Compilation

generally, assigning (hyper)graphs to CNF gives tractable classes
Graph Decomposition Techniques and Knowledge Compilation

generally, assigning (hyper)graphs to CNF gives tractable classes

- \( \beta \)-acyclicity
- disjoint branches
- \( \gamma \)-acyclicity

Parameter of the hypergraph

Parameter of the incidence graph

(Samer, Szeider '10; Paulusma, Slivovský, Szeider '13; Fisher, Makowsky, Ravve '08; Slivovský, Szeider '14; Capelli, M, Durand '14; Brault-Baron, Capelli, M '15; Sæther, Telle, Vatshelle '14, Bova, Capelli, M, Slivovský '15)
Treewidth vs. Cliquewidth

- d-DNNF size for formula $F$
  - treewidth $k$ FPT: $2^k |F|$
  - cliquewidth $k$ not FPT: $|F|^k$

- FPT size far preferable

Question: Can DNNF size for cliquewidth be improved to FPT?

Theorem (roughly, M ’16)
- Cliquewidth $k$ formulas generally require DNNF size $n^k$
  - proved with communication complexity approach
  - hard instances random systems of linear equations
Treewidth vs. Cliquewidth

- d-DNNF size for formula $F$
  - treewidth $k$ FPT: $2^k |F|$
  - cliquewidth $k$ not FPT: $|F|^k$
- FPT size far preferable

**Question**
Can DNNF size for cliquewidth be improved to FPT?
Treewidth vs. Cliquewidth

- d-DNNF size for formula $F$
  - treewidth $k$ FPT: $2^k|F|$
  - cliquewidth $k$ not FPT: $|F|^k$
- FPT size far preferable

**Question**
Can DNNF size for cliquewidth be improved to FPT?

**Theorem (roughly, M ’16)**

*Cliquewidth* $k$ formulas generally require DNNF size $n^k$
Treewidth vs. Cliquewidth

- d-DNNF size for formula $F$
  - treewidth $k$ FPT: $2^k|F|$
  - cliquewidth $k$ not FPT: $|F|^k$
- FPT size far preferable

**Question**
Can DNNF size for cliquewidth be improved to FPT?

**Theorem (roughly, M ’16)**

**Cliquewidth** $k$ formulas generally require DNNF size $n^k$

- proved with communication complexity approach
- hard instances random systems of linear equations
data structures from artificial intelligence with many connections to other areas
  - database theory
  - (communication/circuit) complexity
  - graph theory
  - lower bounds for algorithms
data structures from artificial intelligence with many connections to other areas
  ▶ database theory
  ▶ (communication/circuit) complexity
  ▶ graph theory
  ▶ lower bounds for algorithms

some things I could not talk about
  ▶ factorized databases
  ▶ constant delay enumeration algorithms (with updates)
  ▶ qualitative differences between counting algorithms
Conclusion

- data structures from artificial intelligence with many connections to other areas
  - database theory
  - (communication/circuit) complexity
  - graph theory
  - lower bounds for algorithms
- some things I could not talk about
  - factorized databases
  - constant delay enumeration algorithms (with updates)
  - qualitative differences between counting algorithms

Thank you for your attention!