Constraints (2016) 21:413–434
DOI 10.1007/s10601-016-9245-y

Multi-language evaluation of exact solvers in graphical model discrete optimization

Barry Hurley[1] · Barry O'Sullivan[1] · David Allouche[2] ·
George Katsirelos[2] · Thomas Schiex[2] ·
Matthias Zytnicki[2] · Simon de Givry[2]

Published online: 22 April 2016

Abstract By representing the constraints and objective function in factorized form, graphical models can concisely define various NP-hard optimization problems. They are therefore extensively used in several areas of computer science and artificial intelligence. Graphical models can be deterministic or stochastic, optimize a sum or product of local functions, defining a joint cost or probability distribution. Simple transformations exist between these two types of models, but also with MaxSAT or linear programming. In this paper, we report on a large comparison of exact solvers which are all state-of-the-art for their own target language. These solvers are all evaluated on deterministic and probabilistic graphical models coming from the Probabilistic Inference Challenge 2011, the Computer Vision and Pattern Recognition OpenGM2 benchmark, the Weighted Partial MaxSAT Evaluation 2013, the MaxCSP 2008 Competition, the MiniZinc Challenge 2012 & 2013, and the CFLib (a library

✉ Simon de Givry
degivry@toulouse.inra.fr

Barry Hurley
barry.hurley@insight-centre.org

Barry O'Sullivan
barry.osullivan@insight-centre.org

David Allouche
david.allouche@toulouse.inra.fr

George Katsirelos
george.katsirelos@toulouse.inra.fr

Thomas Schiex
thomas.schiex@toulouse.inra.fr

Matthias Zytnicki
matthias.zytnicki@toulouse.inra.fr

[1] Insight Centre for Data Analytics, University College Cork, Cork, Ireland

[2] MIAT, UR-875, INRA, 31320 Castanet Tolosan, France

of Cost Function Networks). All 3026 instances are made publicly available in five different formats and seven formulations. To our knowledge, this is the first evaluation that encompasses such a large set of related NP-complete optimization frameworks, despite their tight connections. The results show that a small number of evaluated solvers are able to perform well on multiple areas. By exploiting the variability and complementarity of solver performances, we show that a simple portfolio approach can be very effective. This portfolio won the last UAI Evaluation 2014 (MAP task).

Keywords Graphical model · Markov random field · Weighted constraint satisfaction problem · Integer linear programming · MaxSAT

1 Introduction

Graphical Models can concisely represent highly dimensional multivariate functions using a factorization into local functions. We consider discrete variables.

Constraint Networks and weighted variants such as Cost Function Networks (CFNs), aka (Weighted) Constraint Satisfaction Problems ((W)CSPs), aim at finding an assignment of all variables that minimizes a joint cost function defined as the sum of local functions (constraints being represented as functions with values in $\{0, \infty\}$). With Boolean variables, and a language restricted to clausal form, the (partial weighted Max)-SAT problem has the same target. Constraint Programming (CP), an extension of Constraint Networks including non-deterministic programming language features, can also easily capture these optimization problems by introducing cost variables [43].

In AI and statistics, probabilistic graphical models [29] use the same idea to concisely represent probability distributions over random variables. These models include Bayesian Networks and Markov Random Fields (MRFs). The problem of identifying a variable assignment that has maximum probability is called the *Maximum Probability Explanation* in Bayesian networks, or *Maximum A-Posteriori* (MAP) in MRF. This NP-hard problem has an extremely large application scope, e.g., in image processing or bioinformatics. By a simple ($-\log$) transformation, these problems can be reduced to CFNs.

Graphical Models can also be easily encoded as 0/1 Linear Programming (01LP) problems, a standard language for Operations Research (OR). We consider two encodings, including one based on the so-called *local polytope* [20, 30, 47], which has several interesting properties.

In this paper, we extract probabilistic and deterministic graphical models from various areas, each using a specific language. This covers competitions in MaxSAT, constraint programming, probabilistic inference and repositories in probabilistic image processing and cost function networks. We encode them in these underlying languages and close relatives, from AI (CFN, MaxSAT, MRF), CP, and OR (01LP). These benchmarks are traditionally used in competitions relying on a single language with dedicated solvers. We compare exact solvers which are all state-of-the-art for their own language on these encodings. We then define a novel portfolio hybrid solver exploiting them.

2 Combinatorial optimization languages

In this section we briefly describe the combinatorial optimization languages that will be used.

[CFN] Cost Function Networks, or Weighted Constraint Networks, extend Constraint Networks by using non-negative cost functions instead of constraints [38].

Definition 1 A Cost Function Network (CFN) is a triple (X, W, k) where $X = \{1, \dots, n\}$ is a set of n discrete variables, W is a set of non-negative functions, and k, a (possibly infinite) maximum cost. Each variable $i \in X$ has a finite domain D_i of values that can be assigned to it. A function $w_S \in W$, with scope $S \subseteq X$, is a function $w_S : D_S \mapsto \{\alpha \in \mathbb{N} \cup \{k\} : \alpha \leq k\}$, where D_S denotes the Cartesian product of all D_i for $i \in S$.

In CFNs, the cost of a complete assignment is the sum of all cost functions. A solution has cost less than k. Therefore a cost of k denotes forbidden assignments, used in hard constraints. A solution of minimum cost is sought.

[MRF] Markov Random Fields define a joint probability distribution. The terminology of *Graphical Models* (GMs) originally designates *probabilistic graphical models* such as Markov Random Fields (MRFs) and Bayesian Networks (BNs) [29]. In this paper, we restrict ourselves to MRFs because they do not impose any restriction on the local functions that can be used in the decomposition of the joint probability distribution (BNs use local conditional probabilities with a normalization requirement).

Definition 2 A discrete Markov Random Field (MRF) is a pair (X, Φ) where $X = \{1, \dots, n\}$ is a set of n random variables, and Φ is a set of potential functions. Each variable $i \in X$ has a finite domain D_i of values that can be assigned to it. A potential function $\phi_S \in \Phi$, with scope $S \subseteq X$, is a function $\phi_S : D_S \mapsto \mathbb{R} \cup \{\infty\}$.

The probability of a tuple $t \in D_X$ is defined as:

$$P(t) \propto \prod_{\phi_S \in \Phi} \exp(-\phi_S(t[S])) = \exp(-\sum_{\phi_S \in \Phi} \phi_S(t[S]))$$

where $t[S]$ denotes the restriction of t to the set of variables S. The additive potentials ϕ_S are called energies, in relation with statistical physics. Alternatively, multiplicative $\exp(-\phi_S(t[S])$ potentials can be used.

In this paper, we consider the MAP query that aims at finding a complete assignment of maximum probability (or equivalently, minimum energy).

[WPMS] Weighted Partial MaxSAT problems are CFNs restricted to Boolean domains and a language of weighted clauses [36].

Definition 3 A Weighted Partial MaxSAT (WPMS) instance is defined as a set of pairs $\langle C, w \rangle$ and an upper bound k. Each C is a clause and w is a number in $\mathbb{N} \cup \{k\}$, the *weight* of clause C. A clause is a disjunction of literals. A literal is a Boolean variable or its negation.

A clause with weight $\geq k$ is a *hard* clause, otherwise it is *soft*. The objective is to find an assignment to the variables appearing in the clauses that minimizes the sum of the weights of all falsified clauses, which should be of cost $< k$.

[01LP] A 0/1 Linear Program is defined by a linear objective function over a set of 0/1 variables to minimize under a conjunction of linear equalities and inequalities [25].

[CP] Constraint Programming problems are defined by a set of discrete variables and a set of constraints. The aim is to minimize the value of a given objective variable while satisfying all constraints [45].

3 Translations between formalisms

In this section we present encodings between graphical models represented in each of the AI/OR/CP languages we presented. We summarize in Table 1 for each input formalism the different translations used to produce every instance in the corresponding output formalism.

[MRF] Markov Random Field. With additive potentials, MRFs are essentially equivalent to CFNs except for the fact that they can use arbitrary real-valued potential functions instead of integer non-negative costs.[1] Additive MRFs can therefore be reduced to CFNs using a fixed decimal point representation of energies which are then scaled to integers and shifted to enforce non-negativity. This preserves optimal solutions.

Multiplicative MRFs can be transformed to additive MRFs using a simple ($-\log$) transform, and then to CFNs [17, 18]. Conversely, CFNs can be transformed to multiplicative MRFs (as in the UAI *MARKOV* format) by exponentiating costs.[2] Costs are all shifted by the same amount so that the largest multiplicative potentials are equal to 1. Hard costs ($\geq k$) are translated to a zero multiplicative potential (infinite energy) to preserve the ability to prune domain values based on constraint reasoning.

[WPMS] Weighted Partial MaxSAT. As weighted partial MaxSAT is a CFN with Boolean variables and a language of clauses, thus a WPMS instance is already a CFN. For a CFN, we consider two encodings to WPMS based on CSP to SAT encodings: the *direct* encoding [6], and the *tuple* encoding encoding introduced by Bacchus [7]. WPMS costs are non-negative integers and the WCNF format allows to express an upper bound that will be used to represent k, preserving the ability to prune.

Direct encoding: for each variable i with domain size $|D_i| > 2$, we use one proposition $d_{i,r}$ for each value $r \in D_i$. This proposition is true iff variable i is assigned the value r. To ensure that exactly (At Least and At Most One) one value is used for each variable, we encode *At Most One* (AMO) with hard clauses $(\neg d_{i,r} \vee \neg d_{i,s})$ for all $i \in \{1, \ldots, n\}$ and all $r < s$, $r, s \in D_i$, as well as *At Least One* (ALO) with one hard clause $(\bigvee_r d_{i,r})$ for each i. Boolean variables are directly encoded as propositions and do not require AMO/ALO clauses. Then, for each cost function $w_S \in W$ and each tuple $t \in D_S$ with $w_S(t) > 0$, we have a clause $(\bigvee_{i \in S} \neg d_{i,t[i]})$ with weight $w_S(t)$, where $d_{i,t[i]}$ denotes the proposition associated with assigning to variable i the value that it has in tuple t.

Tuple encoding: it encodes domains as in the direct encoding. We have a proposition $d_{i,r}$ for each variable/value pair representing $i = r$, along with AMO/ALO clauses that enforce that each variable is assigned exactly one value (for non-Boolean variables). Nullary and unary cost functions are also represented as soft clauses exactly as in the direct encoding.

For each cost function w_S, $|S| > 1$, and each tuple $t \in D_S$ with $w_S(t) < k$ we have a proposition $p_{S,t}$. For non-zero cost $w_S(t) > 0$, we have the soft clause $(\neg p_{S,t})$ with weight $w_S(t)$. This represents the cost to pay if the tuple t is used. For every variable $i \in S$, we have a hard clause $(d_{i,t[i]} \vee \neg p_{S,t})$. These clauses enforce that if tuple t is used, its values $t[i]$ must be used. Then, for each variable $i \in S$ and each value $r \in D_i$, we have hard

[1] Rational costs are also used in [11].

[2] Script available at genoweb.toulouse.inra.fr/degivry/evalgm/scripts/wcsp2markov.py

Table 1 Summary of translations between formalisms and possible issues

In/Out	MRF	CFN	WPMS	01LP	CP
	(UAI)	(WCSP)	(WCNF)	(LP)	(MINIZINC)
MRF	-	$-\log(prob)$	Through CFN	Through CFN	Through CFN
CFN	$\exp(-cost)$	-	Direct/ tuple encod.	Direct/ tuple encod.	Extra cost[a] vars & *table* constraints
WPMS	Through CFN[b]	Direct trans.	Direct encod. only	Through CFN[c]	Extra cost[d] vars & reified logical *or*
CP	Through CFN	Decomposed objective & global constraints[e]	Through CFN	Through CFN[f]	-

[a] Cannot represent large costs ($> 2^{31}$) using a single domain. CFLib benchmarks were manually translated, avoiding table constraints except for ProteinDesign and SPOT5

[b] Cannot represent large clauses ($>$ 23 literals in our case) using complete tables

[c] No tuple encoding

[d] Cannot represent large costs ($> 2^{31}$) using a single domain

[e] Cannot represent large domains in extension ($d > 1,000$) and non-decomposable objectives (requiring cost functions with $> 10^6$ tuples)

[f] This translation is far from being optimal, e.g., linear constraints will be first decomposed in ternary cost functions

clauses $(\neg d_{i,r} \vee \bigvee_{t \in D_S, t[i]=r, w_S(t)<k} p_{S,t})$ that enforce that if a value $r \in D_i$ is used, one of the allowed tuples $t \in D_S$ such that $t[i] = r$, $w_S(t) < k$ must be used.

On CSP, it is known that Unit Propagation (UP) on the tuple encoding enforces arc consistency in the original CSP (the set of values that are deleted by enforcing AC have their corresponding literals set to false by UP) [7].

We express the asymptotic complexities of the two encodings in terms of the total number of tuples of cost 0 (t_0), k (t_k) or other (t_r) in the problem. For the direct encoding, this is $O(nd^2 + t_k + t_r)$, while for the tuple encoding this is $O(nd^2 + a(t_0 + t_r))$, where n is the number of variables, d is the maximum domain size, and a is the maximum cost function arity. The hidden big-O constants are larger for the tuple encoding, which has an additional linear factor a. In our experiments (see Table 2 in Section 4.1), we found that the tuple encoding is typically much larger, more than can be accounted for by the hidden constants. Hence it appears that our benchmark instances have many more tuples with zero cost than with infinite (k) cost ($t_0 >> t_k$).

[01LP] 0/1 Linear Programming. The 01LP encodings of CFNs are similar to those for WPMS, using 0/1 variables. The additional expressivity of linear constraints enables further simplifications. These translations are used to generate 01LP in CPLEX "LP" format.

Direct encoding: AMO/ALO clauses are replaced by one linear constraint per non-Boolean variable $i \in X$: $\sum_{r \in D_i} d_{i,r} = 1$. For each cost function w_S, the soft clause encoding of a tuple t with non-zero soft cost $0 < w_S(t) < k$ is replaced by a linear constraint $\sum_{i \in S}(1 - d_{i,t[i]}) + p_{S,t} \geq 1$ that forces the value of $p_{S,t}$ to 1 if the tuple t is used.

This $p_{S,t}$ variable appears in the objective function, with a coefficient $w_S(t)$. If t has cost k or above, a constraint $\sum_{i\in S}(1-d_{i,t[i]}) \geq 1$ is used and no term appears in the objective function.

Tuple encoding: the same encoding as above is used for domains and for zero and unitarity cost functions. For each cost function w_S, $|S|>1$, for each variable $i \in S$, each value $r \in D_i$, a constraint $\sum_{t\in D_S,t[i]=r,w_S(t)<k} p_{S,t} = d_{i,r}$ enforces that a value (i,r) is used iff a tuple t s.t. $t[i]=r$ and $w_S(t)<k$ is used. The same 0/1 variable $p_{S,t}$ appears in the objective function with a $w_S(t)$ coefficient if $0 < w_S(t) < k$.

This encoding has been proposed by Koster in [30] to encode Partial Constraint Satisfaction Problems. Since all $d_{i,r}$ are 0/1 variables, the constraints enforce that the $p_{S,t}$ are also integral. We therefore relax the integrality constraint on $p_{S,t}$ variables.

Assuming there are no costs in $\{0,\infty\}$, for each cost function w_S, each variable i, and each value $r \in D_i$, by summing the linear constraints $\sum_{i\in S}(1-d_{i,t[i]})+p_{S,t} \geq 1$ from the direct encoding over all tuples $t \in D_S$ such that $t[i]=r$, we found:

$$\begin{aligned}
M|S| - \sum_{j\in S\setminus\{i\}} \frac{M}{|D_j|}(\sum_{s\in D_j} d_{j,s}) - Md_{i,r} + \sum_{t\in D_S,t[i]=r} p_{S,t} &\geq M\\
M|S| - \sum_{j\in S\setminus\{i\}} \frac{M}{|D_j|}(1) - Md_{i,r} + \sum_{t\in D_S,t[i]=r} p_{S,t} &\geq M\\
M|S| - \frac{M(|S|-1)}{d} - Md_{i,r} + \sum_{t\in D_S,t[i]=r} p_{S,t} &\geq M\\
M(|S|-1) - \frac{M(|S|-1)}{d} - Md_{i,r} + \sum_{t\in D_S,t[i]=r} p_{S,t} &\geq 0
\end{aligned}$$

$$\text{Thus,} \sum_{t\in D_S,t[i]=r} p_{S,t} \geq M(d_{i,r} - (|S|-1)(1-\frac{1}{d}))$$

with $d = \max_{j\in S\setminus\{i\}} |D_j|$ and $M = |D_{S\setminus\{i\}}|$, the Cartesian product of all domains of S except D_i. If $|S|=2$, then $M=d$, and $M(d_{i,r}-(|S|-1)(1-\frac{1}{d}))$ is either negative ($d_{i,r}=0$) or equal to 1 ($d_{i,r}=1$). Therefore, the direct encoding can be seen as a relaxation of the tuple encoding.

The continuous relaxation of the tuple encoding is known in the MRF field as the *local polytope* [20, 47, 50]. This polytope is interesting for several reasons. First, the dual of the local polytope is exactly the Optimal Soft Arc Consistency (OSAC) LP for CFN described in [11, 12]. This polytope underlies also convergent message-passing bounds [20, 50] used for MRF optimization. Ignoring possible value pruning (by node consistency or substitutability [19]), OSAC and therefore the local polytope bound too, are known to be stronger than any other soft arc consistency [11]. Second, the dual variables of this polytope can be directly interpreted as the amount of cost that is shifted by arc consistency so-called Equivalence Preserving Transformations [13]. Therefore, existing soft arc consistencies that iteratively change blocks of costs can be analyzed as fast incremental approximate Block Coordinate Descent algorithms aiming at solving this dual LP [37]. This result establishes a strong link between 01LP solvers using the local polytope encoding and CFN/MRF solvers using soft arc consistencies or convergent message passing: in absence of pruning, the LP bound will always be at least as strong as the soft arc consistency bounds.

The significance of this connection is further strengthened by a recent result showing that the local polytope (or its dual) are "universal" in the sense that any LP can be translated *in linear time* in a graphical model whose local polytope has the same optimum as the original LP [44]. Progress in solving this polytope (exactly or approximately by soft arc consistencies or message passing) and in solving a general LP are therefore tightly linked.

[CP] Constraint Programming. In [43], a translation of CFNs into crisp CSPs has been proposed. In this transformation, the decision variables of the CFN are preserved and every cost function is reified into a constraint whose scope is augmented by one extra variable, representing the assignment cost. This reification of costs into domain variables transforms a CFN in a crisp CSP with more variables and increased arities. Typically, unary and binary cost functions are converted into TABLE constraints of arity two and three respectively. Another extra cost variable encodes the objective function, connected by a SUM constraint to all other cost variables. All the cost variables are non-negative integers with the same initial upper bound k as provided in the WCSP format. The same approach applies to WPMSs, using reified Boolean expressions instead of TABLE constraints to encode hard and soft clauses. The resulting CSP models are expressed in the MINIZINC [39] CP language.[3]

The converse translation of CP models with a cost variable into a CFN (and then MRFs and WPMSs) that does not use cost variables is a complex task.[4] It requires identifying local[5] cost functions, starting from the objective variable, while removing intermediate cost variables. We implemented a corresponding prototype in NUMBERJACK[6] [22] reading the low-level FLATZINC format [39]. Global constraints are decomposed into ternary cost functions in extension (tables with costs in $\{0, \infty\}$, see [1]), requiring small input domain sizes.

4 Graphical model evaluation

We have collected a set of benchmarks and performed experiments using state-of-the-art solvers coming from several research areas.

4.1 Collection of benchmarks

To gather an extensive set of benchmarks representing optimization problems from various areas, we collected problems from different sources including deterministic (CFN, MaxCSP, WPMS), probabilistic (MRF, BN), as well as CP collections. Each collection contains several categories of instances, each category corresponding to a specific class of problems.

[MRF]: the Probabilistic Inference Challenge (PIC) 2011 benchmark set[7] and the (5-ary) genetic linkage analysis problem [18] from the Uncertainty in Artificial Intelligence

[3] A 1-hour time limit was used to translate MINIZINC2 to FLATZINC, readable by CP solvers.

[4] Directly lifting a CP model, with its cost variable, to a CFN would be of limited value since all AC in CFN are known to enforce AC on cost functions representing hard constraints.

[5] We restrict the size of cost functions to be less than 10^6 tuples in our implementation.

[6] http://numberjack.ucc.ie/

[7] http://www.cs.huji.ac.il/project/PASCAL

(UAI) 2008 Evaluation[8] were taken in UAI *MARKOV* format with multiplicative potentials. This PIC challenge on approximate inference in probabilistic graphical models is dedicated to a variety of queries and we only considered the MAP/MPE query. We used a subset of the instances available in PIC 2011, excluding Alchemy, CSP, Promedas, and ProteinProtein.[9] These problems have been translated to CFNs in WCSP format (then to WPMS, 01LP, CP) using a $(-\log)$ transform followed by fixed decimal point representation with 2-digit precision after the decimal point (the precision is constrained by CP solvers that typically accept only 32-bit integers).[10]

[CVPR]: the Computer Vision and Pattern Recognition (CVPR) OpenGM2 benchmark[11] [27] contains binary and ternary MRF instances in HDF5 format with additive potentials. We excluded Brain, Knott, and MatchingStereo/ted-gm instances because of their size ($> 1GB$), and ModularityClustering because it came from outside the computer vision community. ColorSeg, MatchingStereo, PhotoMontage have integer energies, directly defining non-negative costs. For the others, we used 8-digit precision after the decimal point.

[CFLib]: the CFLib[12] is a collection of CFN and MaxSAT problems expressed in WCSP format. We extracted problems that are directly available in the WCSP format and further translated them into dedicated MINIZINC models manually. The extracted benchmarks include combinatorial auctions [35], CELAR/GRAPH radio-link frequency assignment problems [10], Mendelian error correction problems on complex pedigrees [46], computational protein design problems [3] (with 2-digit precision), SPOT5 satellite scheduling problems [8], and uncapacitated warehouse location problems [33, 34].

[MaxCSP]: all binary CSP categories with table constraints and at least one inconsistent instance (BlackHole, Langford, Quasi-group Completion Problem, Graph Coloring, random Composed, random 3-SAT EHI, and random Geometric, excluding pure random categories) from the CSP 2008 Competition[13] were translated from XCSP2.1/XML format to CFNs (as MaxCSPs) where allowed (resp. forbidden) tuples have zero (resp. unit) cost. We set $k = 1,000$.

[WPMS]: weighted partial MaxSAT instances coming from the MaxSAT 2013 Evaluation[14], including crafted MIPLib, DIMACS Max Clique, and industrial WPMS instances, have been directly encoded as CFNs, each clause being encoded as a cost function with just one non-zero cost tuple. Translation to MRF (resp. CP) was restricted to instances with small-arity clauses (resp. with 32-bit costs, excluding the WPMS/Upgradeability category).

[8] http://graphmod.ics.uci.edu/uai08/Evaluation/Report/Benchmarks

[9] Alchemy and Promedas were solved by TOULBAR2 in less than 1 sec. each. CSP instances came from CFLib. ProteinProtein is already present in CVPR under the name of *Protein Prediction* ProteinInteraction.

[10] The resulting WCSP instances were translated back to UAI instances (with *_digit2* extension) in order to optimize the same objective function.

[11] http://hci.iwr.uni-heidelberg.de/opengm2

[12] http://costfunction.org/benchmark

[13] http://www.cril.univ-artois.fr/CPAI08 and http://www.cril.univ-artois.fr/~lecoutre/benchmarks.html

[14] http://maxsat.ia.udl.cat:81/13/benchmarks/

Table 2 Number of instances and their total compressed (gzipped) size per format for each benchmark resource

Benchmark	Nb.	UAI	WCSP	LP (direct)	LP (tuple)	WCNF (direct)	WCNF (tuple)	MINIZINC
MRF	319	187MB	475MB	2.4G	2.0GB	518MB	2.9GB	473MB
CVPR	1461	430MB	557MB	9.8GB	11GB	3.0GB	15GB	N/A
CFN	281	43MB	122MB	300MB	3.5GB	389MB	5.7GB	69MB
MaxCSP	503	13MB	24MB	311MB	660MB	73MB	999MB	29MB
WPMS	427	N/A[a]	387MB	433MB	N/A	717MB	N/A	631MB
CP	35	7.5MB	597MB	499MB	1.2GB	378MB	1.9GB	21KB
Total	3026	0.68G	2.2G	14G	18G	5G	27G	1.2G

[a]Only WPMS/MaxClique and WPMS/MIPLib (except mod008) could be translated in UAI format for a total of 8.8MB gzipped size

[CP]: we extracted a selection of CFN-decomposable CP problems from the MiniZinc Challenges 2012 &2013.[15] Only the smallest instances in FastFood, Golomb, and OnCallRostering categories could be decomposed in WCSP format using less than 1GB per instance (resp. 1, 3, and 3 instances per category).

Together, these benchmark resources contain problems offering a large variety in terms of size, maximum arity or domain size and cost range. WPMS and CVPR categories have the highest number of variables (close to 1 million variables for WPMS/TimeTabling, half a million for CVPR/PhotoMontage and ColorSeg). The WPMS benchmark also has the largest arities (a weighted clause on 580 variables appears in Haplotyping). For the other benchmarks, maximum arity varies from 2 to 5. Graph connectivities are usually very small for MRF&CVPR (often based on grid graphs where vertices represent pixels in images) and WPMS benchmarks. MRF/ObjectDetection, CFN/ProteinDesign, MaxCSP/Langford, and CVPR/Matching have complete graphs. MRF/ ProteinFolding has the largest domain size (503 values). Most CVPR instances have very large cost ranges (8-digit precision), whereas MaxCSP instances contain only 0/1 costs. The emphasis between optimization and feasibility also varies a lot among the problems: almost all deterministic GM categories, except MaxCSPs and CFN/CELAR, contain forbidden (k) tuples in their cost functions. On the contrary, probabilistic GMs usually have no forbidden tuples (except for MRF/Linkage and DBN).

Table 2 reports the number of instances per benchmark resource and its gzipped size for the seven formulations. The UAI format appears to be the most compact to express local functions as tables. It relies on a *complete* ordered table of costs which does not require describing tuples whereas the other formats explicitly describe tuples associated to non-zero costs. The price to pay for this conciseness is the inability of the UAI format to represent large arity functions with a few non-zero costs (such as large weighted clauses). As seen before, the tuple encoding is usually larger than the direct one, except for MRF/CVPR LPs where the local polytope is a good choice since there are almost no zero costs. CP instances

[15] http://www.minizinc.org/challenge2012/results2012.html and http://www.minizinc.org/challenge2013/results2013.html

benefit from global constraints in the MINIZINC language, which are decomposed in large tables in the other formats.

4.2 Experimental settings

We compared state-of-the-art MRF solvers[16] DAOOPT[17] [42] (using its 1-hour settings), winner of PIC 2011, and TOULBAR2[18] [18, 34] (including Virtual Arc Consistency (VAC) as preprocessing [11], dominance rule pruning [19], and hybrid best-first search [2]), winner of MaxCSP 2008 and UAI 2010 & 2014 Evaluations, against WPMS MAXHS[19] solver [14, 15], winner of crafted WPMS MaxSAT 2013, the CP solver GECODE,[20] winner of MiniZinc Challenges 2012, and IBM-ILOG CPLEX 12.6.0.0 (using parameters EPAGAP, EPGAP, and EPINT set to zero to avoid premature stop).

All computations were performed on a single core of AMD Opteron 6176 at 2.3 GHz and 8 GB of RAM with a 1-hour CPU time limit.[21]

4.3 Experimental results

The number of instances solved in less than 1 hour, excluding translation times between formats, is available in Table 3. Resource-based cactus plots are shown in Fig. 1.[22] Beyond the number of problems solved and the mean CPU time on solved instances reported in this table, we refine our analysis in two ways. First, we summarize the evolution of lower and upper bounds for each algorithm over all instances in Fig. 2.

Specifically, for each instance I we normalize all costs as follows: the initial lower bound produced by TOULBAR2 (before VAC) is 0; the best – but potentially suboptimal – solution found by any algorithm is 1; the worst solution is 2. This normalization is invariant to translation and scaling. Additionally, we normalize time from 0 to 1 for each pair of algorithm A and instance I, so that each run finishes at time 1. This time normalization is different for different instances and for different algorithms on the same instance. A point $\langle x, y \rangle$ on the lower bound line for algorithm A in Fig. 2 means that after normalized runtime x, algorithm A has proved on average over all instances a normalized lower bound of y and similarly for the upper bound. We show both the upper and lower bound curves for all algorithms evaluated here, except GECODE which produces no meaningful lower bound before it proves optimality. In order for the last point of each curve to be visible, we extend all curves horizontally after 1.0. Additionally, on the right of Fig. 2, we show the same curves but excluding instances that took less than 5 seconds to solve with a simple version of TOULBAR2 that does not use either VAC preprocessing or hybrid best first search, for a final set of 1208 instances. We remove those easy instances because the runtime tends to

[16] MPLP2 http://cs.nyu.edu/~dsontag version 2 (using 2.10^{-7} gap thres.) was tested but the results are not presented in Section 4.3 as it was dominated in most categories by TOULBAR2.

[17] https://github.com/lotten/daoopt open source version 1.1.2, not including the closed source and unavailable convergent message-passing bound tightening used in the PIC challenge.

[18] http://www.inra.fr/mia/T/toulbar2 version 0.9.8, parameters *-A -V -dee -hbfs*.

[19] http://www.maxhs.org version 2.51, no parameter.

[20] http://www.gecode.org/ version 4.4.0, using free search.

[21] Using parameter *-pe parallel_smp 2* on a SUN Grid Engine to ensure half-load of the cores on the cluster.

[22] More detailed results are available at http://genoweb.toulouse.inra.fr/~degivry/evalgm.

Table 3 Number of problems solved in less than 1 hour (N/A if RAM usage or 32-bit limit prevented encoding). In parentheses, mean CPU time in seconds on solved instances ('-' if none). Bold is best. The first column contains the category name followed by s: nb. of instances, d: max. dom. size, a: max. arity

Problem/s/d/a	DAOOPT	TOULBAR2	CPLEX	CPLEX$_{tuple}$	MAXHS	MAXHS$_{tuple}$	GECODE
MRF/319/503/5	151	**226**	156	210	118	72	1
(UAI)	(584.39)	**(93.80)**	(111.88)	(82.18)	(98.68)	(1509.93)	
DBN/108/2/2	60	**81**	65	69	38	2	0
	(626.79)	**(192.42)**	(124.66)	(155.12)	(366.15)	(1748.65)	(-)
Grid/21/2/2	5	0	**15**	1	8	0	0
	(1223.67)	(-)	**(120.90)**	(3354.21)	(557.01)	(-)	(-)
ImageAlignment/10/93/2	10	**10**	0	9	0	0	0
	(754.96)	**(5.27)**	(-)	(88.41)	(-)	(-)	(-)
Linkage/22/7/5	17	14	16	**22**	20	20	0
	(576.94)	(364.73)	(365.09)	**(21.99)**	(52.62)	(124.04)	(-)
ObjectDetection/37/21/2	0	0	0	0	0	0	0
	(-)	(-)	(-)	(-)	(-)	(-)	(-)
ProteinFolding/21/503/2	0	**21**	10	9	2	0	0
	(-)	**(20.24)**	(169.28)	(176.17)	(268.51)	(-)	(-)
Segmentation/100/21/2	59	**100**	50	100	50	50	1
	(460.33)	**(0.29)**	(0.06)	(3.35)	(7.38)	(22.54)	(1509.93)
CVPR/1461/20/3	1274	**1340**	382	1332	483	1038	N/A
(HDF5)	(481.02)	**(22.81)**	(179.96)	(8.70)	(355.71)	(58.83)	
ChineseChars/100/2/2	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	N/A
ColorSeg/21/12/2	0	**15**	0	5	0	0	N/A
	(-)	**(1340.56)**	(-)	(190.33)	(-)	(-)	
GeomSurf/600/7/3	555	**600**	382	600	387	321	N/A
	(509.01)	**(0.96)**	(179.96)	(2.89)	(183.53)	(63.15)	
InPainting/4/4/2	0	**2**	0	1	0	0	N/A
	(-)	**(325.72)**	(-)	(339.90)	(-)	(-)	
Matching/4/20/2	4	**4**	0	3	0	0	N/A
	(319.24)	**(3.20)**	(-)	(765.25)	(-)	(-)	
MatchingStereo/2/20/2	0	0	0	0	0	0	N/A
	(-)	(-)	(-)	(-)	(-)	(-)	
ObjectSeg/5/8/2	0	4	0	**5**	0	0	N/A
	(-)	(2292.28)	(-)	**5**	0	0	
PhotoMontage/2/7/2	0	0	0	0	0	0	N/A
	(-)	(-)	(-)	(-)	(-)	(-)	
ProteinInteraction/8/2/3	0	0	0	**3**	0	2	N/A
	(-)	(-)	(-)	**(60.80)**	(-)	(1019.63)	
SceneDecomp/715/8/2	715	**715**	0	715	96	715	N/A
	(460.20)	**(0.07)**	(-)	(1.11)	(1049.83)	(54.20)	
CFN/281/300/3	211	**256**	245	238	228	210	141
(WCSP)	(768.93)	**(109.84)**	(33.85)	(34.00)	(14.91)	(121.20)	(224.77)

Table 3 (continued)

Problem/*s*/*d*/*a*	DAOOPT	TOULBAR2	CPLEX	CPLEX$_{tuple}$	MAXHS	MAXHS$_{tuple}$	GECODE
Auction/170/2/2	169	170	170	170	**170**	170	113
	(663.04)	(93.10)	(0.03)	(0.14)	**(0.03)**	(121.16)	(231.55)
CELAR/16/44/2	4	**14**	0	3	0	0	0
	(598.72)	**(279.00)**	(-)	(560.44)	(-)	(-)	(-)
Pedigree/10/28/3	4	**10**	5	9	10	6	0
	(373.43)	**(10.58)**	(44.28)	(57.27)	(190.49)	(99.28)	(-)
ProteinDesign/10/198/2	4	**9**	0	7	0	4	0
	(597.46)	**(13.40)**	(-)	(298.88)	(-)	(477.72)	(-)
SPOT5/20/4/3	(309.04)	(40.44)	**16**	12	6	5	0
	6	4	**16**	12	6	5	0
Warehouse/55/300/2	24	49	**54**	37	42	25	28
	(1752.42)	(163.23)	**(142.57)**	(6.46)	(6.78)	(92.83)	(197.39)
MaxCSP/503/50/2	176	**398**	219	75	249	233	6
(XCSP)	(603.56)	**(386.08)**	(152.73)	(876.84)	(76.21)	(538.93)	(115.39)
BlackHole/37/50/2	10	10	**30**	10	10	10	0
	(222.19)	(0.08)	**(141.91)**	(2.22)	(0.30)	(2.78)	(-)
Coloring/22/6/2	17	17	**17**	16	14	14	4
	(319.29)	(11.39)	**(7.14)**	(72.33)	(17.67)	(50.80)	(171.61)
Composed/80/10/2	26	**80**	80	37	80	73	0
	(543.73)	**(0.13)**	(4.48)	(1667.07)	(79.81)	(1383.72)	(-)
EHI/200/7/2	0	**179**	0	0	1	0	0
	(-)	**(773.86)**	(-)	(-)	(3078.96)	(-)	(-)
Geometric/100/20/2	92	**95**	65	0	89	84	0
	(755.46)	**(134.57)**	(419.39)	(-)	(31.52)	(138.98)	(-)
Langford/4/29/2	2	**2**	2	1	2	2	2
	(272.24)	**(0.12)**	(38.79)	(0.03)	(0.32)	(2.19)	(2.97)
QCP/60/9/2	29	15	25	11	**53**	50	0
	(496.31)	(143.49)	(54.94)	(263.83)	**(121.82)**	(242.80)	(-)
WPMS/427/2/580	11	197	269	N/A	**321**	N/A	28
(WCNF)	(536.35)	(110.33)	(109.76)		**(168.67)**		(243.39)
Haplotyping/100/2/580	N/A	1	18	N/A	**44**	N/A	0
		(784.32)	(679.90)		**(674.01)**		(-)
MIPLib/12/2/93	2	3	3	N/A	**3**	N/A	3
	(365.31)	(102.39)	(49.85)		**(9.47)**		(28.61)
MaxClique/62/2/2	9	33	38	N/A	**40**	N/A	24
	(574.36)	(209.07)	(229.33)		**(362.26)**		(280.38)
PackupWeighted/99/2/177	N/A	53	**99**	N/A	99	N/A	0
		(167.82)	**(0.72)**		(7.14)		(-)
PlanningWithPref/29/2/372	N/A	7	11	N/A	**28**	N/A	1
		(515.22)	(751.65)		**(65.82)**		(0.03)

Table 3 (continued)

Problem/*s*/*d*/*a*	DAOOPT	TOULBAR2	CPLEX	CPLEX$_{tuple}$	MAXHS	MAXHS$_{tuple}$	GECODE
TimeTabling/25/2/36	N/A	0	0	N/A	**7**	N/A	0
		(-)	(-)		**(1020.73)**		(-)
Upgradeability/100/2/77	N/A	100	**100**	N/A	100	N/A	N/A
		(12.43)	**(0.84)**		(2.73)		
CP/35/*163/4*	9	16	2	7	18	14	**26**
(MINIZINC)	(387.13)	(354.57)	(0.99)	(584.10)	(145.94)	(400.03)	**(138.55)**
AMaze/6/17/4	0	3	0	4	**6**	5	4
	(-)	(279.71)	(-)	(998.46)	**(12.00)**	(161.25)	(176.91)
FastFood/6/*5/2*	1	1	1	1	1	1	**6**
	(200.32)	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)	**(14.22)**
Golomb/6/*163/3*	0	3	0	0	3	1	**6**
	(-)	(44.97)	(-)	(-)	(117.34)	(78.01)	**(111.17)**
OnCallRostering/5/*89/4*	1	2	1	2	**3**	3	2
	(253.25)	(27.27)	(1.98)	(47.44)	**(162.22)**	(362.19)	(75.13)
ParityLearning/7/20/4	7	7	0	0	5	4	**7**
	(432.94)	(663.51)	(-)	(-)	(343.24)	(907.40)	**(248.10)**
VRP/5/100/4	0	0	0	0	0	0	**1**
	(-)	(-)	(-)	(-)	(-)	(-)	**(255.45)**
Total	1832	**2433**	1273	1862	1417	1567	202
	(534.34)	**(107.26)**	(123.70)	(57.35)	(191.45)	(143.45)	(219.37)
Nb. of 1st position	0	**16** [1]	7 [3]	3 [5]	9 [2]	0	4 [4]
Nb. of best solution	2209 [2]	**2562 [1]**	1355 [5]	1300 [6]	1626 [4]	1706 [3]	229[7]
Nb. of single best sol.	57 [4]	88 [2]	43 [5]	**95 [1]**	80 [3]	1 [7]	13 [6]
Zscore (time)	135.37 [6]	**57.84 [1]**	102.97 [3]	104.89 [4]	90.73 [2]	122.88 [5]	136.58 [7]
Zscore (cost)	63.00 [3]	**26.25 [1]**	59.24 [2]	69.92 [4]	80.55 [5]	108.76 [7]	100.55 [6]
Borda-score	89.40 [5]	**182.50 [1]**	129.60 [2]	102.78 [4]	114.37 [3]	59.54 [7]	60.64 [6]
Borda-score (norm)	2.08 [5]	**4.24 [1]**	3.01 [2]	2.86 [3]	2.66 [4]	1.65 [7]	1.84 [6]

be dominated by whatever preprocessing technique each solver uses. This means that the optimal solution is reported near the end of the search, although it is early in absolute terms.

Note that this plot highlights different aspects of the solvers' behavior than the cactus plots and should be interpreted in conjunction with those.

In the second part of our analysis, we compute global measures that try to compensate for the very different cardinalities of the categories. For each instance, we compute two Z-scores,[23] one for the CPU time and another for the cost of the best solution found at the deadline. In the extreme case where a solver is the only one able to solve an instance (resp. is not able to solve it), we use a score of -4 (resp. 4). A mean Z-score is then computed for each category and the sum of all mean Z-scores is reported in Table 3.

To take into account the CPU time and cost in a common measure, we also computed Borda scores, following the MiniZinc Challenge's approach. For each instance, and each pair of solvers, a reward in [0, 1] is granted to each solver as follows: if a solver reports a better cost than the other, it is granted a reward of 1 (and 0 for the other). For identical costs, if t_0 and t_1 are the CPU time for two solvers denoted 0 and 1, the solver i will receive

[23] The Z-score of a value x in a set of values is $\frac{x-\mu}{\sigma}$ where μ is the mean of the set and σ its standard deviation.

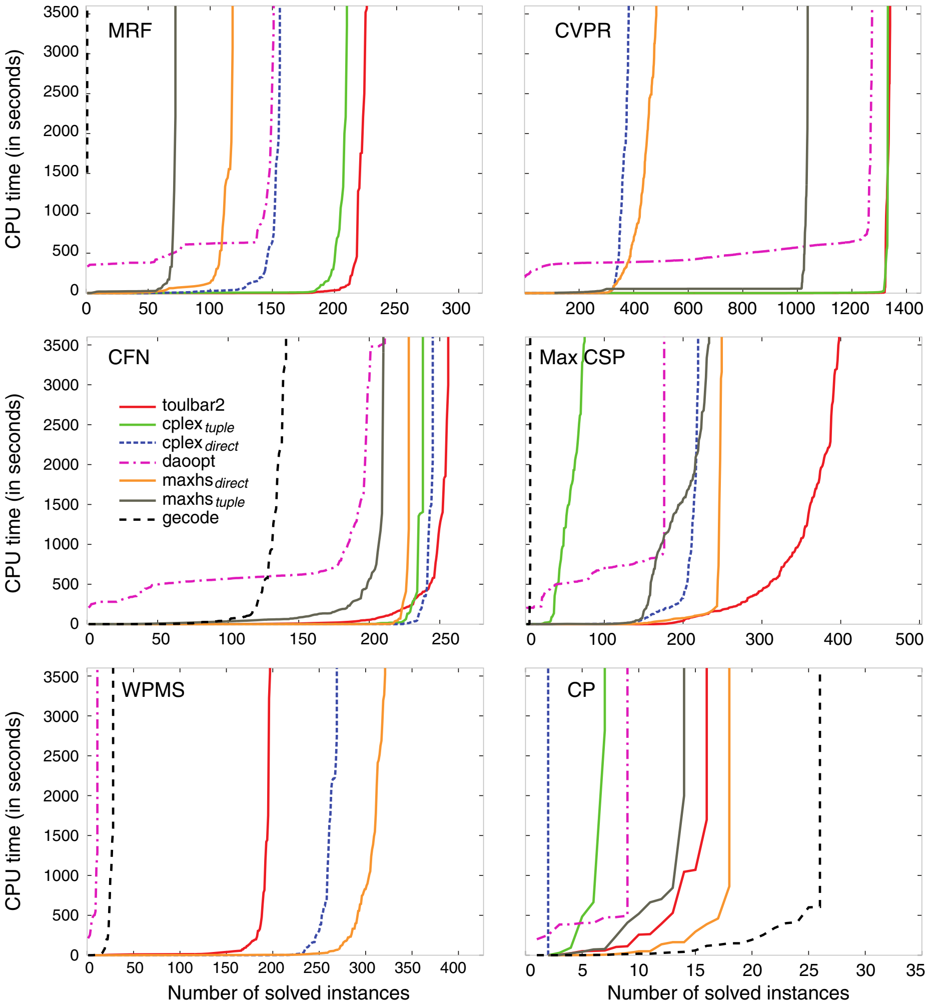

Fig. 1 Cactus plots for MRF, CVPR, CFN, MaxCSP, WPMS, and CP benchmark resources

a reward of $\frac{t_{|i-1|}}{t_0+t_1}$, favoring the fastest solver. A mean Borda score is computed for each category and the sum of mean scores reported.

The tuple encoding and CP approaches are not applicable to WPMS and CVPR, respectively. Quite fairly, these measures penalize these approaches for this limitation. To see if this penalty was enough to explain the scores of these approaches, we also report the Borda score normalized by the number of applicable categories (this optimistically assumes that these approaches would work as well on these inaccessible benchmarks as on the rest of the benchmarks). The only change is a swap of the order between CPLEX with the tuple encoding and MAXHS with the direct encoding.

As expected, for each source of benchmarks, the best solver (in terms of number of solved instances) is usually a solver that is dedicated to this type of problems (i.e., TOULBAR2 for CFN, MAXHS for WPMS, GECODE for CP). However, some solvers, such as MAXHS,

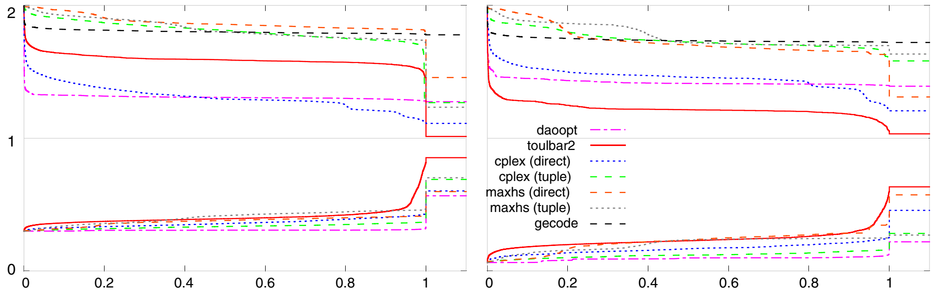

Fig. 2 Normalized lower and upper bounds on all instances (*left*) and a set of 1208 hardest instances (*right*)

CPLEX, and TOULBAR2, performed well on several resources, respectively solving to optimality 2,043, 2,313, and 2,433 instances among a total of 3,026 (using the best encoding on each category for MAXHS and CPLEX). Using the number of solved instances per category, breaking ties by best mean CPU time on solved instances, these three solvers won the first position on 9, 10, and 16 categories respectively, among 43 categories. Looking at cactus plots in Fig. 1, TOULBAR2 and CPLEX dominate on MRF&CVPR, followed by DAOOPT. They also dominate on CFN, followed by MAXHS. TOULBAR2 performed well on MaxCSP. MAXHS and GECODE dominate on their own category (resp. WPMS and CP).

In terms of extreme size and solving difficulty, the CVPR/ColorSeg/colseg-cow4 instance defines the largest search space ($d^n = 2^{829,440}$) completely solved by TOULBAR2. MRF/ObjectDetection is the smallest totally unsolved category ($d^n \leq 2^{264}$). We now consider each benchmark resource, highlighting unexpected results.

[MRF]: on MRF/Linkage [28] (maximum number of variables $n = 1289$, maximum domain size $d = 7$), CPLEX$_{tuple}$, followed by MAXHS, got the best results, showing their suitability for non-binary (max. arity $a = 5$) cost functions with forbidden tuples. The tuple encoding is the only 01LP encoding usually considered in MRFs. Surprisingly, CPLEX with the direct encoding was the best on the Grid category ($n = 6400, d = 2$), benefiting from a large number of *zero-half cuts*. DAOOPT did not perform as well as for the PIC 2011 Evaluation. One explanation is the missing problem reformulation feature used in the PIC challenge [42], a piece of code which is not available in source or binary format. Another explanation is that DAOOPT spends more time finding good upper bounds (using local search in preprocessing) than on the optimality proof (as Fig. 2 seems to show). CP solvers performed poorly on MRFs due to the large costs, resulting in huge domains for cost variables in the CFN-to-CP translation.

[CVPR]: on CVPR/Scene Decomposition, using a superpixel model [27] with fewer variables ($n = 208, d = 8$), TOULBAR2 solved all 715 instances in 0.07 second each on average compared to 1.11 for CPLEX$_t$. The good performance of TOULBAR2 on CVPR instances is largely due to its virtual arc consistency initial problem reformulation [11]. On these problems, it offers a tight lower bound in much less time than LP on the tuple encoding. This encoding was always better for CPLEX, consistent with the ubiquity of the local polytope formulation as a linear relaxation for MRFs. The tuple encoding also improved the performance of the MaxSAT solver (see MAXHS$_t$ results in Table 3) on two categories (ProteinInteraction, SceneDecomp).

[CFN]: TOULBAR2 clearly dominates on CELAR ($n = 458, d = 44$), Pedigree ($n = 10017, d = 28$), and ProteinDesign ($n = 18, d = 198$), whereas CPLEX with direct encoding, followed by MAXHS, performed the best on Operations Research problems Auction ($n = 246, d = 2$) and Warehouse ($n = 1100, d = 300$). The 01LP tuple encoding still performed quite well when the problem size remains relatively small ($n \times d \leq 20,000$), otherwise memory errors sometimes occurred, as on the largest Warehouse instances (*capa-b-c-m*). GECODE performed relatively well on Auction and Warehouse, solving three large instances (*capmo-3-4-5* with $n = 200, d = 100$).

[MaxCSP]: MAXHS performed well on MaxCSP due to its ability to quickly solve all the satisfiable (zero cost optimum) instances that remained in the Geometric ($n = 50, d = 20$) and QCP ($n = 264, d = 9$) categories, thanks to its embedded MINISAT solver. The good results obtained by DAOOPT can similarly be explained by its initial stochastic local search procedure [42], finding good initial upper bounds especially on satisfiable or random instances like EHI ($n = 315, d = 7$) and Geometric. TOULBAR2 won the first position on four MaxCSP categories, especially on EHI random category, thanks to its new hybrid best-first search strategy [2] which *simulates* restarts with memory. Surprisingly, the tuple encoding was always dominated by the direct encoding here.

[WPMS]: large clause arities make the tuple encoding or the use of exhaustive tables in UAI format space intractable. While MAXHS dominates the scene, it is interesting to notice the ability of CPLEX to outperform MAXHS on two categories (Upgradeability and PackupWeighted). In PackupWeighted ($n = 25554, d = 2$), CPLEX can be up to one order of magnitude faster than MAXHS. GECODE was the fastest solver to find and prove optimality on 11 MaxClique ($n = 3321, d = 2$) instances, whereas MAXHS won this category by solving 40 instances among 62.

[CP]: CP instances are difficult to translate into GMs with local functions and small domains: 10 instances among 35 MINIZINC instances could not be translated for space reasons. Moreover the translation is not appropriate for LP solvers (linear constraints are decomposed), explaining the poor performance of CPLEX. Here, GECODE performed the best in most of the cases. However, MAXHS, performed the best on two categories: Amaze and OnCallRostering. Similarly, DAOOPT was faster than GECODE on the most difficult ParityLearning instance (52_26_6.3). DAOOPT solved all the instances in preprocessing thanks to its complete bucket/variable elimination [16], with a memory space usage below 529MB (induced width less than 25), smaller than its limits (4GB and $i = 35$-bound) [42].

With either the tuple or direct encoding, CPLEX was able to be the best in at least one category per benchmark resource (except for CP) showing very good robustness. For probabilistic models, the tuple encoding is the ideal choice since the emphasis is on optimization (essentially no tuple with cost k). In this case, the tuple formulation offers a strong bound, an essential source of pruning. In several cases however, thanks to its incremental soft arc consistencies and strong virtual arc consistency preprocessing, TOULBAR2 outperformed CPLEX on such problems. These results can be analyzed in the light of the known relations between LP and soft arc consistencies [11]: thanks to pruning by node consistency and substitutability and to their efficiency and strong incrementality, soft arc consistencies seem capable of outperforming LP by finding a better trade off than LP in the compromise between tightness and computational cost on the local (universal) polytope.

In other benchmarks however, the direct encoding is always preferable. This could be explained by the better conciseness of the encoding on benchmarks with many 0 costs, as shown in Table 2, and to some extent by the lesser pruning of the optimization bound in the presence of hard constraints. This encoding seems essentially ignored in the MRF community.

Overall, this shows that significant speedups can be achieved by exploiting encodings to different optimization languages.

5 Exploitation: a portfolio approach

Solver portfolios [21, 23, 32] aim to exploit this diversity by replacing a single solver with a set of complimentary solvers and a mechanism for selecting a subset to use on a particular problem. By making decisions at an instance specific level, it is possible to make significant performance gains over any of the individual component solvers. Solver portfolios have been highly successful in constraint programming [4, 24, 41], satisfiability [26, 51], MaxSAT [5], and many more fields. For an extensive survey of the wide-range of literature on the algorithm selection problem, we refer the reader to [32].

The majority of modern portfolio approaches employ some form of machine learning to take the role of the selection model. To enable this involves a training phase whereby for a reference set of instances, a domain-specific feature description, a candidate set of algorithms, and a performance metric are defined. Feature descriptions for each instance and performance data of each algorithm on each instance are recorded. The machine learning model is built such that the performance metric is maximized on this training data. Subsequently, to apply this trained model to a new test instance at runtime, first the feature description must be computed and passed to the model to make a solver selection. The chosen solver is then applied to the problem instance.

5.1 Graphical model instance features

To describe graphical model instances, we consider the following feature set: i) the input file size, ii) the CPU time to read the instance, iii) an initial upper bound on the solution, iv) the time to compute the initial upper bound, v) the number of variables, vi) the number of cost functions. The ratio of vii) unary, viii) binary, and ix) ternary cost functions, i.e. the fraction of the total number of cost functions of each arity. x) The ratio of cost functions which have arity 4 or greater. Finally, a number of statistics such as the mean, standard deviation, coefficient of variation, minimum, and maximum for xi) domain size, and xii) cost function arity. By no means does this list constitute a comprehensive list of features for graphical models, nevertheless in initial evaluations these proved effective and have the benefit of being relatively cheap to compute.

Table 4 presents the Gini importances [9][24] of the above features according to a decision tree classifier aiming to predict the fastest solver. The most important features are the ratio of binary cost functions, the minimum domain size, and the value of the initial upper bound.

5.2 Machine learning offline evaluation results

Table 5 presents an offline evaluation of a simple portfolio approach based on 6 solvers from Sec. 4.3. We consider a subset of the benchmarks and the solvers such that all the instances could be translated to all the solvers, i.e., we exclude the WPMS and CP benchmarks, and the GECODE solver.

[24] The normalized total reduction brought by the feature.

Table 4 Gini importances of features

Feature	Gini importance	Feature	Gini importance
Ratio of binary cost functions	0.14445	Arity coefficient of variation	0.07546
Minimum domain size	0.13928	Arity std. deviation	0.07149
Initial UB	0.10988	Arity mean	0.06555
Time to read	0.09211	Num. variables	0.04304
Time UB	0.08393	Num. cost functions	0.03875
File size	0.08305	Mean dom. size	0.01634

The portfolio is built using LLAMA [31], with 10-fold stratified cross validation. This involves splitting the dataset into 10-equally sized folds with an equal distribution of the best solver across folds. For brevity, we present results only for the best performing regression, classification, and clustering methods, plus the Random Forest classifier. The *Virtual Best Solver* (VBS) corresponds to an oracle deciding the best solver for each instance. The table lists the mean (std. dev.) CPU time on the solved instances, the number of instances solved to optimality in less than 1 hour, the number of times each solver was the fastest. In addition, the misclassification penalty shows the contribution of each solver to the portfolio, i.e., the number of instances that were not solved by any other solver, and, where another one solved the instance, the additional CPU time needed by the next best solver. From these statistics alone, it is clear that each of the component solvers (except MAXHS$_{tuple}$) play a valuable contribution to the portfolio both in terms of being able to solve more instances, and reducing the overall CPU time needed. Additionally, each of the portfolio methods are able to outperform the single best solver and close most of the gap to the virtual best solver.

Table 5 Summary of portfolio approaches sorted by decreasing number of problems solved over the 2,564 instances

Solver	Solved time (sec.)		Num.	Num.	Misclass. pen.	
	Mean	Std. dev.	solved	best	solved	total time
VBS(6)	93.0	385.1	2,321			
M5P regression	91.5	376.1	2,298			
J48 classification	84.7	368.1	2,294			
Random Forest	74.6	327.6	2,279			
k-means clustering	66.9	301.4	2,259			
TOULBAR2	105.2	408.3	2,220	1,863	224	28,000.1
CPLEX$_{tuple}$	55.4	316.6	1,852	27	3	10,345.3
DAOOPT	535.1	340.1	1,812	3	0	3,236.8
MAXHS$_{tuple}$	140.0	414.5	1,551	3	1	8.4
MAXHS	199.0	565.4	1,078	208	4	9,261.4
CPLEX	127.7	433.4	1,002	217	36	14,381.9

Table 6 Offline evaluation of the UAI 2014 portfolio on 2,564 instances

Solver	Solved time (sec.)		Num.	Num.	Misclass. pen.	
	Mean	Std. dev.	solved	best	solved	total time
VBS(5)	63.5	276.3	2,315			
UAI'14 portfolio	71.8	312.4	2,276			
INCOP+TOULBAR2	87.6	361.2	2,227	352	23	82,616.2
TOULBAR2	105.2	408.3	2,220	1,449	13	56,339.3
CPLEX$_{tuple}$	55.4	316.6	1,852	27	6	9,584.7
MPLP2	66.2	424.6	1,537	198	0	1,183.3
CPLEX	127.7	433.4	1,002	289	46	13,276.0

5.3 The UAI 2014 portfolio

A specific portfolio was developed and submitted to the UAI 2014 Inference Competition (MAP task). It was built from five constituent solvers: i) TOULBAR2, ii) a version of TOULBAR2 taking a starting solution from an initial run of the INCOP [40] local search solver, iii) the Message Passing Linear Programming MPLP2 solver [48, 49], iv) CPLEX using the direct encoding, and v) CPLEX with the tuple encoding. These solvers were selected based on their complementary performances in previous empirical evaluations. Table 6 presents the results of an offline evaluation of this portfolio.[25]

The effectiveness of this multi-language portfolio was independently verified in the UAI 2014 Inference Competition, achieving two first places in the MAP task under both the 20 and 60 minute timeouts.[26] Three of the portfolio's component solvers were submitted to the same competition as independent entries. The two 01LP encodings performed extremely well on certain instances but extremely poorly on the remaining.[27] Based on the competition's overall evaluation metric, the cumulative sum of a solver's rank on each instance, the 01LP encodings did not rank high overall but were the top-ranked solvers in a number of cases. Likewise, the INCOP+TOULBAR2 solver was the highest ranked in some cases but ranked in mid-field in many others.[28] The UAI'14 portfolio solver was highly successful in deciding when to run these solvers or not, achieving first place overall. This independent empirical evaluation supports the findings demonstrated in this paper, that significant speedups can be achieved by exploiting various encodings to related languages.

6 Conclusions

Our empirical results demonstrate the effectiveness of a number of solvers on various graphical model formats, where no single solver consistently dominates the results.

[25] https://github.com/9thbit/uai-proteus used a Random Forest classifier and an older version of TOULBAR2 version 0.9.7, with no parameter. Here we report the results using the same settings as in Sec. 4.3, INCOP+TOULBAR2 corresponds to TOULBAR2 using an extra parameter *-i* for the initial INCOP starting solution phase.

[26] See MAP/Proteus entry at http://www.hlt.utdallas.edu/~vgogate/uai14-competition/leaders.html.

[27] See MAP/MIP-UAI and MAP/MIP-T-UAI entries.

[28] See MAP/IncTb entry.

Rather, the best solver depends on each problem category, bringing to light the respective strengths, robustness and weaknesses of each solver family. They highlight the efficacy of encoding a problem to a related language and exploiting complementary solving technologies.

We demonstrate that it is possible to exploit these complementary strengths using a portfolio approach, built on this knowledge won the UAI 2014 Evaluation. We hope that our proposed collection of benchmarks, readily available in many formats, will enrich the various competitions in CP, AI, and OR, leading to more robust solvers and new solving strategies.

Acknowledgments We are grateful to the GenoToul (Toulouse) Bioinformatic platform for providing us computational support for this work. This work is supported by Science Foundation Ireland (SFI) Grant 10/IN.1/I3032. The Insight Centre for Data Analytics is supported by SFI Grant SFI/12/RC/2289.

References

1. Allouche, D., Bessiere, C., Boizumault, P., Givry, S., Gutierrez, P., Loudni, S., Métivier, J., & Schiex, T. (2012). Decomposing global cost functions. In *Proceedings of AAAI*.
2. Allouche, D., de Givry, S., Katsirelos, G., Schiex, T., & Zytnicki, M. (2015). Anytime hybrid best-first search with tree decomposition for weighted CSP. In *Proceedings of CP* (pp. 12–28).
3. Allouche, D., Traoré, S., André, I., Givry, S., Katsirelos, G., Barbe, S., & Schiex, T. (2012). Computational protein design as a cost function network optimization problem. In *Proceedings of CP* (pp. 840–849).
4. Amadini, R., Gabbrielli, M., & Mauro, J. (2015). A Multicore Tool for Constraint Solving. In *Proceedings of IJCAI* (pp. 232–238).
5. Ansótegui, C., Malitsky, Y., & Sellmann, M. (2014). MaxSAT by Improved Instance-Specific Algorithm Configuration. In *Proceedings of AAAI* (pp. 2594–2600).
6. Argelich, J., Cabiscol, A., Lynce, I., & Manyà, F. (2008). Encoding Max-CSP into partial Max-SAT. In *Proceedings of ISMVL* (pp. 106–111).
7. Bacchus, F. (2007). GAC via unit propagation. In *Proceedings of CP* (pp. 133–147).
8. Bensana, E., Lemaître, M., & Verfaillie, G. (1999). Earth observation satellite management. *Constraints*, *4*(3), 293–299.
9. Breiman, L., Friedman, J., Stone, C.J., & Olshen, R.A. (1984). *Classification and regression trees*: CRC press.
10. Cabon, B., de Givry, S., Lobjois, L., Schiex, T., & Warners, J. (1999). Radio link frequency assignment. *Constraints*, *4*, 79–89.
11. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., & Werner, T. (2010). Soft arc consistency revisited. *Artificial Intelligence*, *174*, 449–478.
12. Cooper, M., de Givry, S., & Schiex, T. (2007). Optimal soft arc consistency. In *Proceedings of IJCAI* (pp. 68–73).
13. Cooper, M.C., & Schiex, T. (2004). Arc consistency for soft constraints. *Artificial Intelligence*, *154*(1-2), 199–227.
14. Davies, J., & Bacchus, F. (2011). Solving MAXSAT by solving a sequence of simpler SAT instances. In *Proceedings of CP* (pp. 225–239).
15. Davies, J., & Bacchus, F. (2013). Exploiting the power of MIP solvers in MaxSAT. In *Proceedings of SAT* (pp. 166–181).
16. Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, *113*(1–2), 41–85.
17. Fargier, H., Lang, J., Martin-Clouaire, R., & Schiex, T. (1995). A constraint satisfaction framework for decision under uncertainty. In *Proceedings of the 11th International Conference on Uncertainty in Artificial Intelligence*. Montréal.
18. Favier, A., Givry, S., Legarra, A., & Schiex, T. (2011). Pairwise decomposition for combinatorial optim. in graphical models. In *Proceedings of IJCAI* (pp. 2126–2132).

19. de Givry, S., Prestwich, S., & O'Sullivan, B. (2013). Dead-end elimination for weighted CSP. In *Proceedings of CP* (pp. 263–272).
20. Globerson, A., & Jaakkola, T. (2007). Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Proceedings of NIPS* (pp. 553–560).
21. Gomes, C.P., & Selman, B. (2001). Algorithm Portfolios. *Artificial Intelligence*, *126*(1-2), 43–62.
22. Hebrard, E., O'Mahony, E., & O'Sullivan, B. (2010). Constraint Programming and Combinatorial Optimisation in Numberjack. In *Proceedings of CP-AI-OR* (pp. 181–185).
23. Huberman, B.A., Lukose, R.M., & Hogg, T. (1997). An economics approach to hard computational problems. *Science*, *275*(5296), 51–54.
24. Hurley, B., Kotthoff, L., Malitsky, Y., & O'Sullivan, B. (2014). Proteus: A hierarchical portfolio of solvers and transformations. In *Proceedings of CP-AI-OR* (pp. 301–317).
25. Jünger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleyblank, W., Reinelt, G., Rinaldi, G., & Wolsey, L. (Eds.) (2010). *50 years of integer programming 1958–2008*: Springer.
26. Kadioglu, S., Malitsky, Y., Sellmann, M., & Tierney, K. (2010). ISAC – Instance-specific algorithm configuration. In *Proceedings of ECAI* (pp. 751–756).
27. Kappes, J., Andres, B., Hamprecht, F., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., & Rother, C. (2015). A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, *115*(2), 155–184.
28. Kishimoto, A., & Marinescu, R. (2013). Recursive best-first and/or search with overestimation for genetic linkage analysis. In *Proceedings of CP workshop on constraint based methods for bioinformatics*.
29. Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*: The MIT Press.
30. Koster, A. (1999). Frequency assignment: Models and algorithms. Ph.D. thesis.
31. Kotthoff, L. (2013). LLAMA: leveraging learning to automatically manage algorithms. Tech. Rep. arXiv:1306.1031.
32. Kotthoff, L. (2014). Algorithm Selection for combinatorial search problems: a survey. *AI Magazine*, *35*(3), 48–60.
33. Kratica, J., Tośic, D., Filipović, V., & Ljubić, I. (2001). Solving the simple plant location problem by genetic alg. *RAIRO*, *35*(1), 127–142.
34. Larrosa, J., de Givry, S., Heras, F., & Zytnicki, M. (2005). Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In *Proceedings of IJCAI* (pp. 84–89).
35. Larrosa, J., Heras, F., & de Givry, S. (2008). A logical approach to efficient max-sat solving. *Artificial Intelligence*, *172*(2-3), 204–233.
36. Li, C.M., & Manyà, F. (2009). Maxsat. In *Handbook of satisfiability, chap. 19*: IOS Press.
37. Meltzer, T., Globerson, A., & Weiss, Y. (2009). Convergent message passing algorithms: a unifying view. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence* (pp. 393–401): AUAI Press.
38. Meseguer, P., Rossi, F., & Schiex, T. (2006). Soft constraints processing, In Rossi, F., van Beek, P., & Walsh, T. (Eds.) *Handbook of constraint programming, chap. 9*: Elsevier.
39. Nethercote, N., Stuckey, P., Becket, R., Brand, S., Duck, G., & Tack, G. (2007). *MiniZinc: Towards a standard CP modelling language*, (pp. 529–543).
40. Neveu, B., Trombettoni, G., & Glover, F. (2004). Id walk: A candidate list strategy with a simple diversification device. In *Proceedings of CP* (pp. 423–437).
41. O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., & O'Sullivan, B. (2008). *Using case-based reasoning in an algorithm portfolio for constraint solving*: Irish Conference on Artificial Intelligence and Cognitive Science.
42. Otten, L., Ihler, A., Kask, K., & Dechter, R. (2012). Winning the PASCAL 2011 MAP challenge with enhanced AND/OR branch-and-bound. In *NIPS DISCML Workshop*.
43. Petit, T., Régin, J., & Bessière, C. (2000). Meta constraints on violations for over constrained problems. In *Proceedings of ICTAI* (pp. 358–365).
44. Prusa, D., & Werner, T. (2015). Universality of the local marginal polytope. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(4), 898–904.
45. Rossi, F., van Beek, P., & Walsh, T. (Eds.) (2006). *Handbook of constraint programming*: Elsevier.
46. Sánchez, M., de Givry, S., & Schiex, T. (2008). Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints*, *13*(1–2), 130–154.
47. Schlesinger, M. (1976). Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, *4*, 113–130.

48. Sontag, D., Choe, D., & Li, Y. (2012). Efficiently searching for frustrated cycles in MAP inference. In *Proceedings of UAI* (pp. 795–804).
49. Sontag, D., Meltzer, T., Globerson, A., Weiss, Y., & Jaakkola, T. (2008). Tightening LP relaxations for MAP using message-passing. In *Proceedings of UAI* (pp. 503–510).
50. Werner, T. (2007). A linear programming approach to max-sum problem. *Pattern Analysis and Machine Intelligence*, *29*(7), 1165–1179.
51. Xu, L., Hutter, F., Hoos, H.H., & Leyton-Brown, K. (2008). SATzilla: Portfolio-based algorithm selection for SAT. In *Journal of artificial intelligence research* (pp. 565–606).