Automatically Selecting Inference Algorithms for Discrete Energy Minimisation

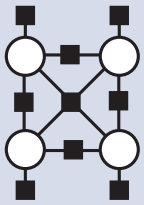
Paul Henderson • Vittorio Ferrari

THE UNIVERSITY of EDINBURGH informatics

ALVIN

Problem

MAP inference in graphical models (GMs)



many algorithms...
GM characteristics affect which is best

we **automatically select best algorithm** for a given GM

Prediction Tasks

best-and-fastest
lowest energy

good-and-fastest
>98% variables correct

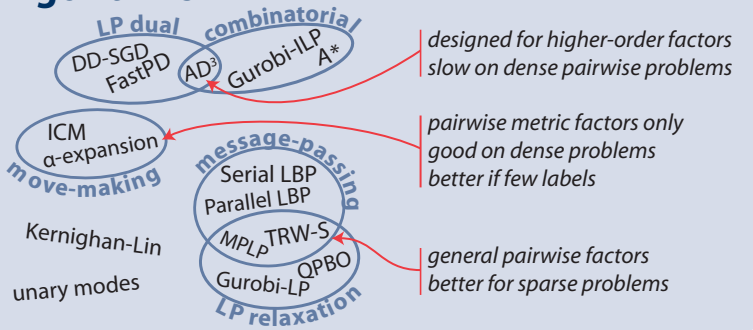
using 1-of-N random forests

Baselines

naïve
use most-often best algorithm

strong
use best for superclass
(pairwise, higher-order, partitioning)

Algorithms

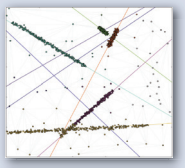


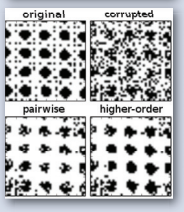
Dataset

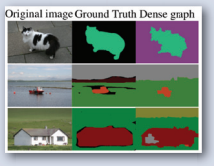
340 GMs from **32 problem classes**

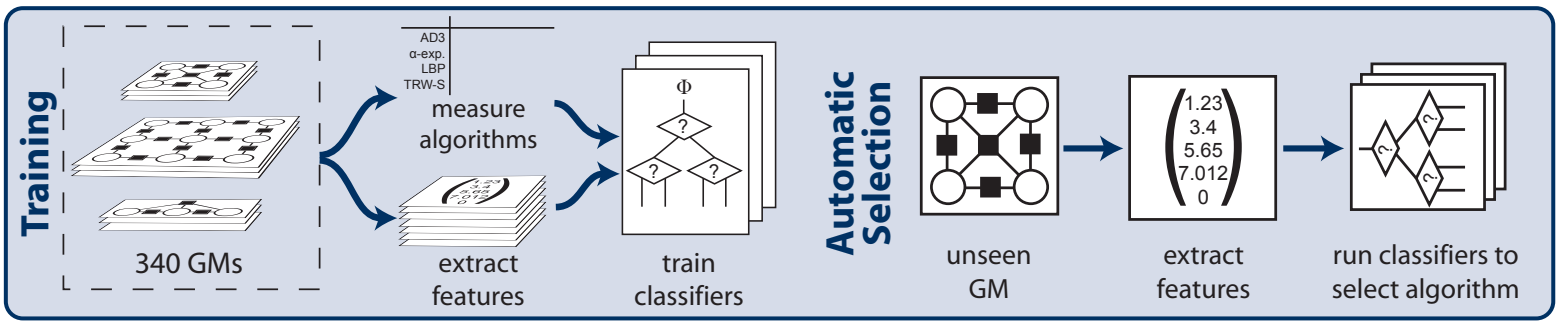
OpenGM2 + four others

50:50 train / test split

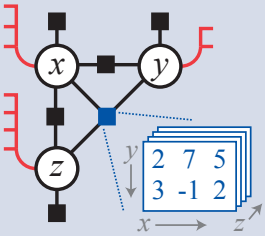








Features



instance size

$|V| = 3$
$|F| = 6$
$L_{min} = 2$
$L_{max} = 4$
$L_{mean} = 3$

structural features

order	density
2	$\frac{2}{\binom{3}{2}} = \frac{2}{3}$
3	$\frac{1}{\binom{3}{3}} = 1$
4	0

energy features

- fractions of pairwise factors that...
  (i) are submodular (ii) satisfy metric axioms
- influence of each factor order on final energy

Accuracy

mean fraction of variables matching best labelling

fraction of GMs correctly classified

predicting the best-and fastest	**62%**	**97.1%**
naïve baseline	30%	75.4%
strong baseline	36%	95.6%
predicting the good-and-fastest	**69%**	**96.4%**
naïve baseline	31%	75.3%
strong baseline	28%	87.5%

- *outperforms even the strong baseline*
- *generalises to problems not seen during training*

Confusion Matrix

• good-and-fastest • pairwise

true \ predicted	AD³	α-exp	BPS	FPD	ICM	KL	LBP	QPBO	TRW-S	UM
AD³	0	0	0	0	2	0	0	0	0	0
α-exp	0	5	1	1	0	0	0	0	2	0
BPS	0	0	1	0	2	0	1	0	1	0
FPD	0	0	1	19	0	0	0	0	0	0
ICM	0	0	0	0	16	1	0	0	3	0
KL	0	0	0	0	0	24	0	0	0	0
LBP	0	0	0	0	3	0	4	0	0	1
QPBO	0	0	3	0	0	0	0	3	0	0
TRW-S	0	1	0	6	1	0	1	0	6	0
UM	0	1	0	1	0	0	0	1	0	0

- *learns subtle distinctions such as whether to use TRW-S, FastPD, or α-expansion*

Speed

	speed-up	matching labels
run every algorithm and take the best	1×	100%
predict then run selected best-and-fastest algorithm	47×	97.1%
predict then run selected good-and-fastest algorithm	88×	96.4%

- *labellings very similar to running every algorithm*
- *much lower computational expense*