APPENDIX: SUPPLEMENTARY MATERIAL
Clamping Improves TRW and Mean Field Approximations

In this Appendix, we provide:

- Details of all methods used for selecting a variable to clamp.
- Additional experimental details and results.
- Additional discussion on greedily selecting a variable to clamp.

8 Details of all methods for selecting a variable to clamp

8.1 Earlier methods: maxW and Mpower

The maxW and Mpower heuristics introduced by Weller and Jebara (2014b) were defined as follows.

8.1.1 maxW

This is the simplest method yet it can be very effective. Assign a clamp score $s(i)$ to each variable i by setting $s(i) = \sum_{j \in \mathcal{N}(i)} |W_{ij}|$. Pick the variable with highest score.

8.1.2 Mpower

Form matrix M defined by $M_{ij} = \frac{1}{n-1} \tanh \left| \frac{W_{ij}}{4} \right|$. The $\tanh$ term is inspired by the effect of cycles in Lemma 5 from Weller et al., 2014, which was derived using loop series methods (Sudderth et al., 2007; Chertkov and Chernyak, 2006), and dividing by $n-1$ is in order to ensure that row sums are < 1 and hence $\sum_{k=1}^{\infty} M^k$ is convergent. Note that $[M^k]_{ii}$ is the sum over all paths of length k from i to i of the product of the modified edge weights along the cycle.

To compute the sum over all k, evaluate $(I-M)^{-1} - I$ and examine diagonal terms. However, this overcounts all cycles, and in particular it includes relatively high value terms coming from paths simply from i to any neighbor j and back again, along with all powers of these. In order to discard these, compute clamp score $s(i)$ as the ith diagonal term of $(I-M)^{-1} - I$ minus $s_i/(1-s_i)$, where s_i is the ith diagonal term of M^2. Pick the variable with highest score.

See (Weller and Jebara, 2014b, Supplement) for more details.

8.2 New methods

We introduced the following new methods.

8.2.1 frustCycles and strongCycles

For frustCycle, the goal is to try to identify at least one frustrated cycle composed of edges (i, j) with high absolute weight $|W_{ij}|$. The method builds on an algorithm introduced by Sontag et al. (2012). strongCycles works in the same way but also takes into consideration balanced cycles. These approaches are the first to examine the sign of edge weights W_{ij} (in order to identify strong frustrated cycles) rather than just their absolute value $|W_{ij}|$.

For both methods, the value of removing a cycle is estimated using a cycScore heuristic from Lemma 5 of Weller et al., 2014, which uses the loop series method (Chertkov and Chernyak, 2006; Sudderth et al., 2007) to attempt to estimate the extent of error, i.e. $A(\theta) - \tilde{A}(\theta)$, caused by the cycle. This estimate will be positive for a balanced cycle and negative for a frustrated cycle, see §5.2. Algorithm 1 provides an outline of the methods.

8.2.2 Strip to the core

To strip a model to its core, simply iteratively remove variables with degree 1 until no more remain, see Figure 1. This is typically run as a pre-processing step before applying other clamp selection heuristics. When removing variables, care must be taken to keep track of the original variable indices for those that remain. For all methods above, we first strip to the core.

Algorithm 1 frustCycles / strongCycles methods to select a variable to clamp

Input: Edge weight model parameters $\{W_{ij}\}$
Output: Clamp scores $s(i)$ for each variable, variable to clamp

Similarly to Mpower, set all $M_{ij} \leftarrow \tanh\left|\frac{W_{ij}}{4}\right|$.
Construct a maximum weight spanning tree T using weights $\{M_{ij}\}$.
Initialize all variables to have clamp score $s(i) \leftarrow 0$.
for all edges of the model not in T **do**
    Consider the edge together with T, which creates a cycle C including the edge
    For the cycle C, compute cycScore $\leftarrow \log(1 + \prod_{(i,j)\in C} \tanh\frac{W_{ij}}{4})$ // note the signed W_{ij} here, see (Weller et al., 2014, Lemma 5)
    For strongCycles, add cycScore to all vertices in C
    For frustCycles, add cycScore to all vertices in C only if cycScore< 0, i.e. a frustrated cycle
end for
Set all $s(i) \leftarrow |s(i)|$.
Pick the variable with highest score.

We have described 4 heuristics so far: maxW, Mpower, frustCycles and strongCycles, all of which first strip to the core. In addition, recognizing that MF makes the assumption that all variables are independent, which may be poor when edge strengths are strong irrespective of cycles being present, we also use a maxW0 heuristic which does not first strip to the core, for a total of five.

8.2.3 TRE methods

All methods so far use only edge weights. We add TRE versions of all the above (see §5.3), hence this gives 10 heuristics.

8.2.4 Meta-heuristics for clamping

Now we have described 10 heurstics, each of which performs better or worse in different contexts. For MF, which always yields a lower bound for $A(\theta)$, and TRW, which always yields an upper bound, we may 'probe' by trying all these heuristics and then pick the one that yields the best improvement in $\tilde{A}(\theta)$. That is, for MF, take the one that yields $\max \tilde{A}_M^{(i)}(\theta)$; for TRW, take the one that yields $\min \tilde{A}_T^{(i)}(\theta)$. For Bethe, if the model is attractive, then $\tilde{A}_B^{(i)}(\theta)$ is a lower bound and we may similarly pick the heuristic that delivers $\max \tilde{A}_B^{(i)}(\theta)$. We call these meta-heuristics *pseudo-greedy*. In addition, we can do the same 'full greedy' process, where we try to clamp *all* possible variables then pick the best. We call this full meta-approach *greedy*.

For Bethe on mixed models, we can't know in advance if we have an over- or under-estimate, so we cannot do exactly the same thing. Instead, we explored performance achieved by picking the variable that gave best improvement in: (i) TRW; (ii) TRW-MF, that is the gap between the two; and (iii) MF. Of these, the greedy-TRW heuristic was the most successful, and it is this version of greedy (and correspondingly, pseudo-greedy) that we report for Bethe on mixed models.

9 Additional experimental details and results

For all inference methods, we used the open source libDAI library (Mooij, 2010), with the following parameters:

For MF,
`MF[tol=1e-7,maxiter=10000,damping=0.0,init=RANDOM,updates=NAIVE]`

For Bethe,
`HAK[doubleloop=1,clusters=BETHE,init=UNIFORM,tol=1e-7,maxiter=10000]`
This is guaranteed to converge to a stationary point of the Bethe free energy (whereas BP may not converge).

For TRW,
`TRWBP[updates=SEQFIX,tol=1e-7,maxiter=10000,logdomain=0,nrtrees=1000,...`
`damping=0.25,init=UNIFORM]`

Note that, particularly for MF and Bethe methods, we may obtain local (rather than global) optima. Because of this, we

might occasionally observe results that get worse with clamping, even where theory shows that the global optimum can only improve. For MF, initializing the optimization of each clamped problem to the solution of the parent problem removes this concern, though we did not find this necessary in practice: empirically it appeared sufficient to initialize using the same random seed each time. For Bethe, we see no easy way to avoid this issue without using expensive methods such as those of Weller and Jebara (2014a).

Models. All grids are toroidal, so all variables have degree 4. Random 4-regular graphs are randomly generated s.t. all variables still have exactly degree 4, though the structure is random. All random Erdös-Renyi models have edge probability s.t. the average degree is 4. Note that the complete graphs have far fewer variables (just 10 or 15) but are much more densely connected (with roughly the same number of edges as the corresponding models with degree 4) and have higher treewidth.

9.1 Additional results

We first provide plots of number of clamps vs. error for all runs, then the same plots but zoomed in so that the Bethe results are easier to see; then plots of runtime ($\log$ scale) vs. error for all runs. Note that sometimes clamping makes the subsequent optimization problems easier to solve, hence the total time with clamping is occasionally *lower* than without, while also being significantly more accurate (for example, see TRW performance with mixed $[-6, 6]$ for the complete graph on 10 variables in Figure 14).

Next in Figure 18, we show the distribution of signed error $\tilde{A}_B(\theta) - A(\theta)$ for Bethe on all our mixed models, showing the bias toward overestimation suggested by the discussion in §5.2.

Finally, in Figure 19, we provide plots showing performance of each heuristic - this indicates how often each one picks the same variable to clamp as pseudo-greedy at each specific clamp step.

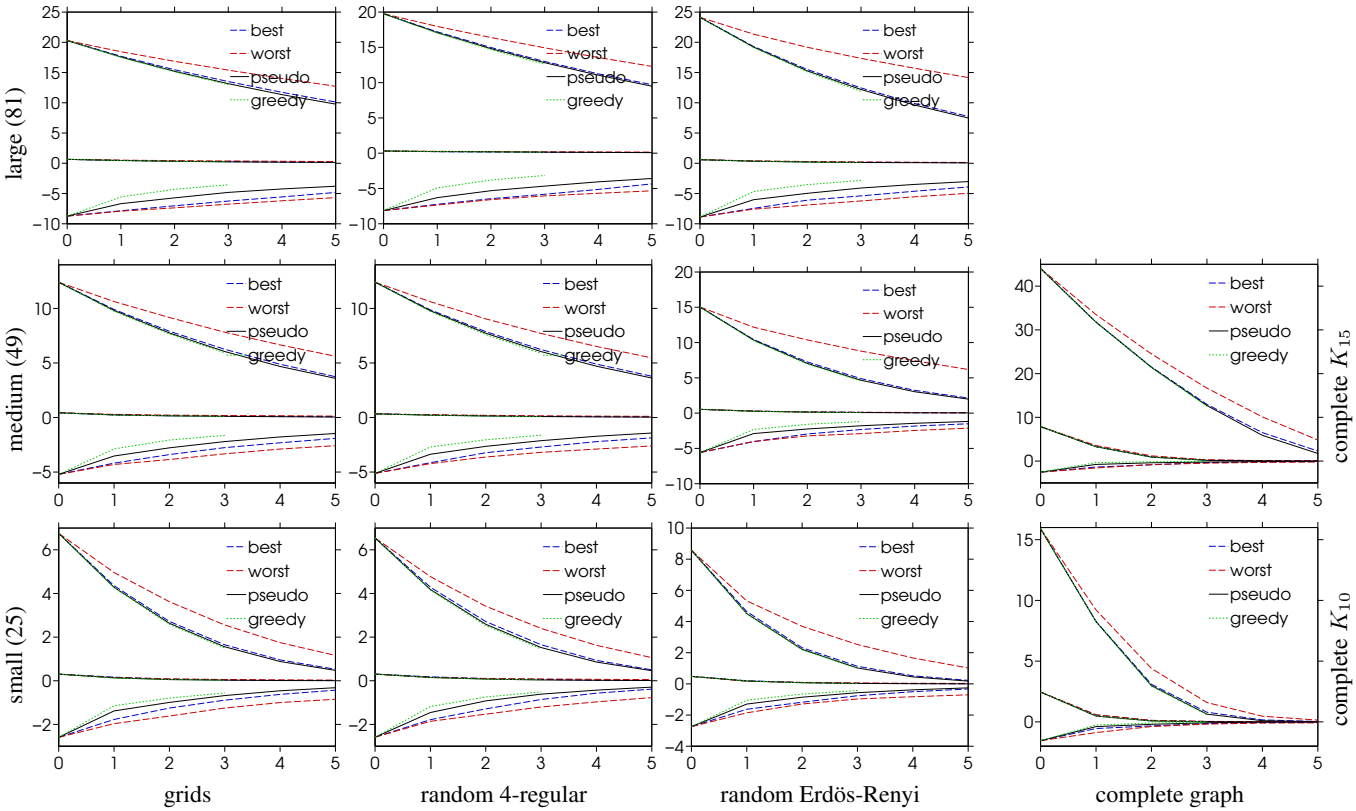

Figure 6: Mixed $[-6, 6]$

10 Additional discussion on greedily selecting a variable to clamp

An interesting question is whether greedily picking the one variable that gives best error improvement and repeating say k times is optimal, i.e. will it result in error as low as if instead, we try all possible *sequences* of clampings up to k long. It becomes computationally expensive to try this but we ran experiments out to 3 clampings. We observed that iterating a greedy search is *not* optimal, in that the full optimization does perform better, but only by a very slight margin on the models we tried.

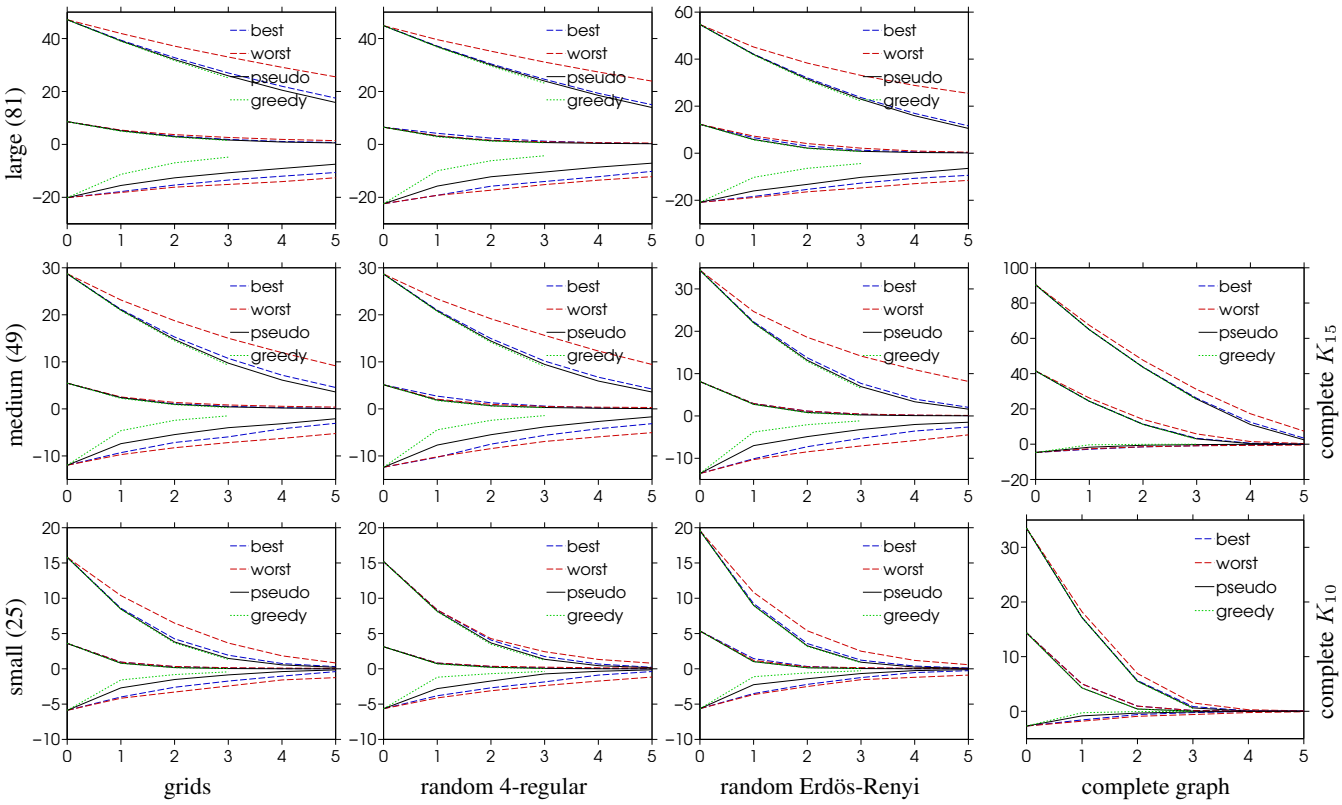

Figure 7: Mixed $[-12, 12]$

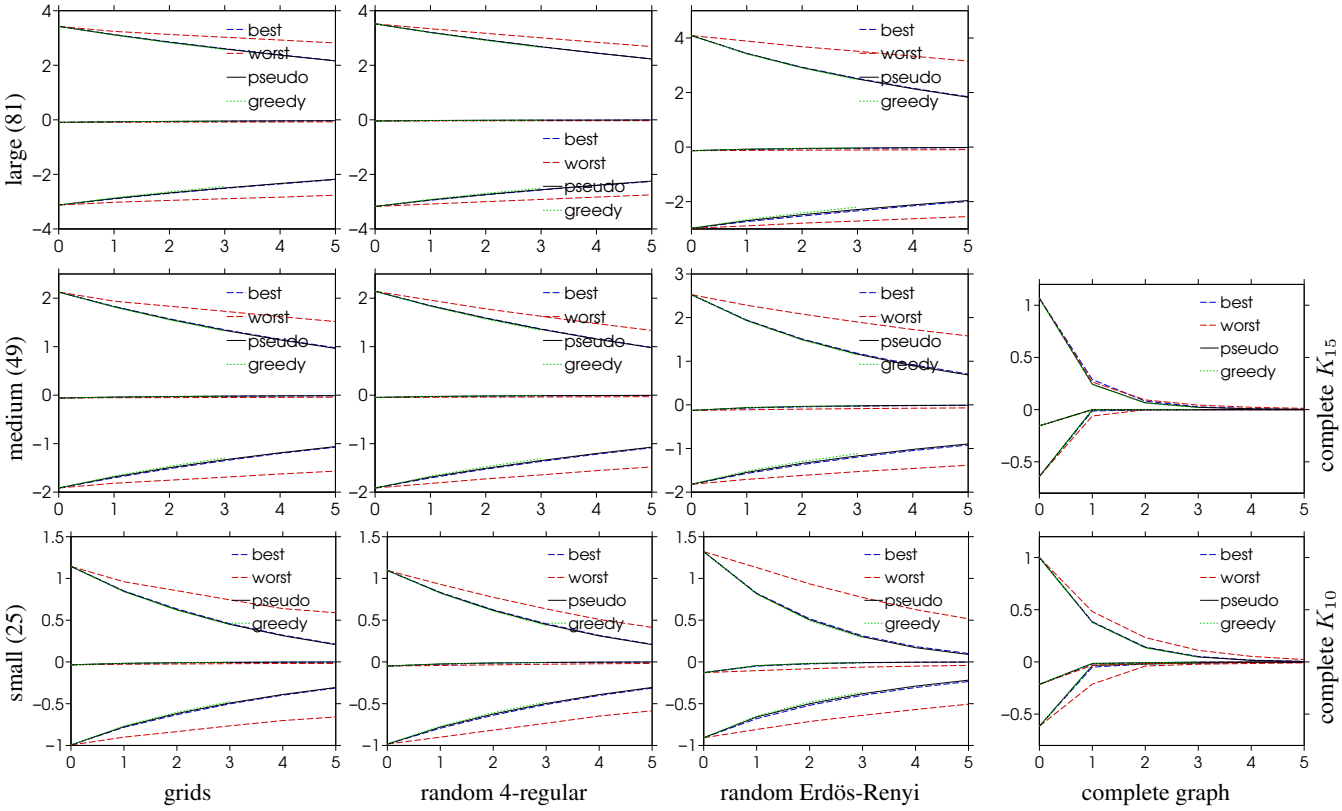

Figure 8: Attractive $[0, 2]$

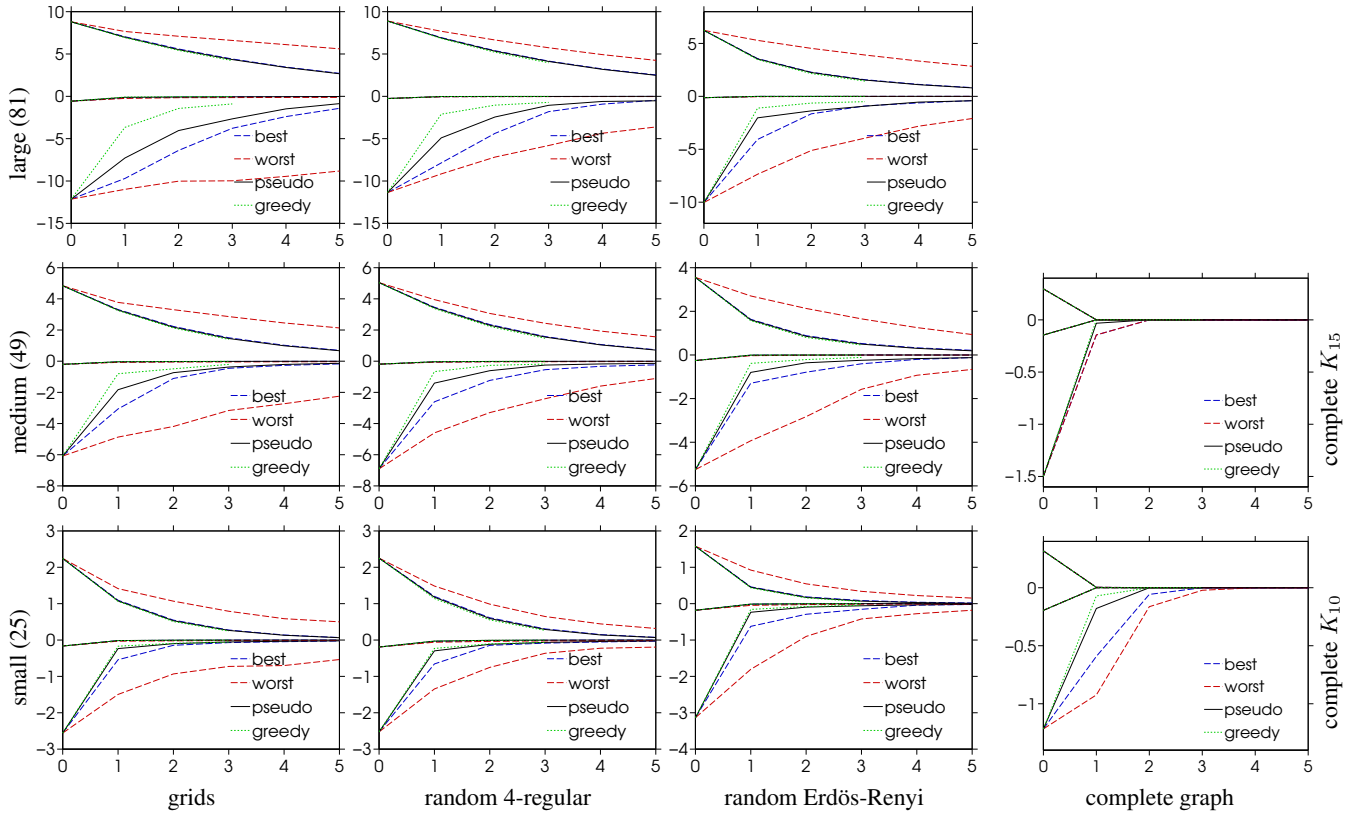

Figure 9: Attractive $[0, 6]$

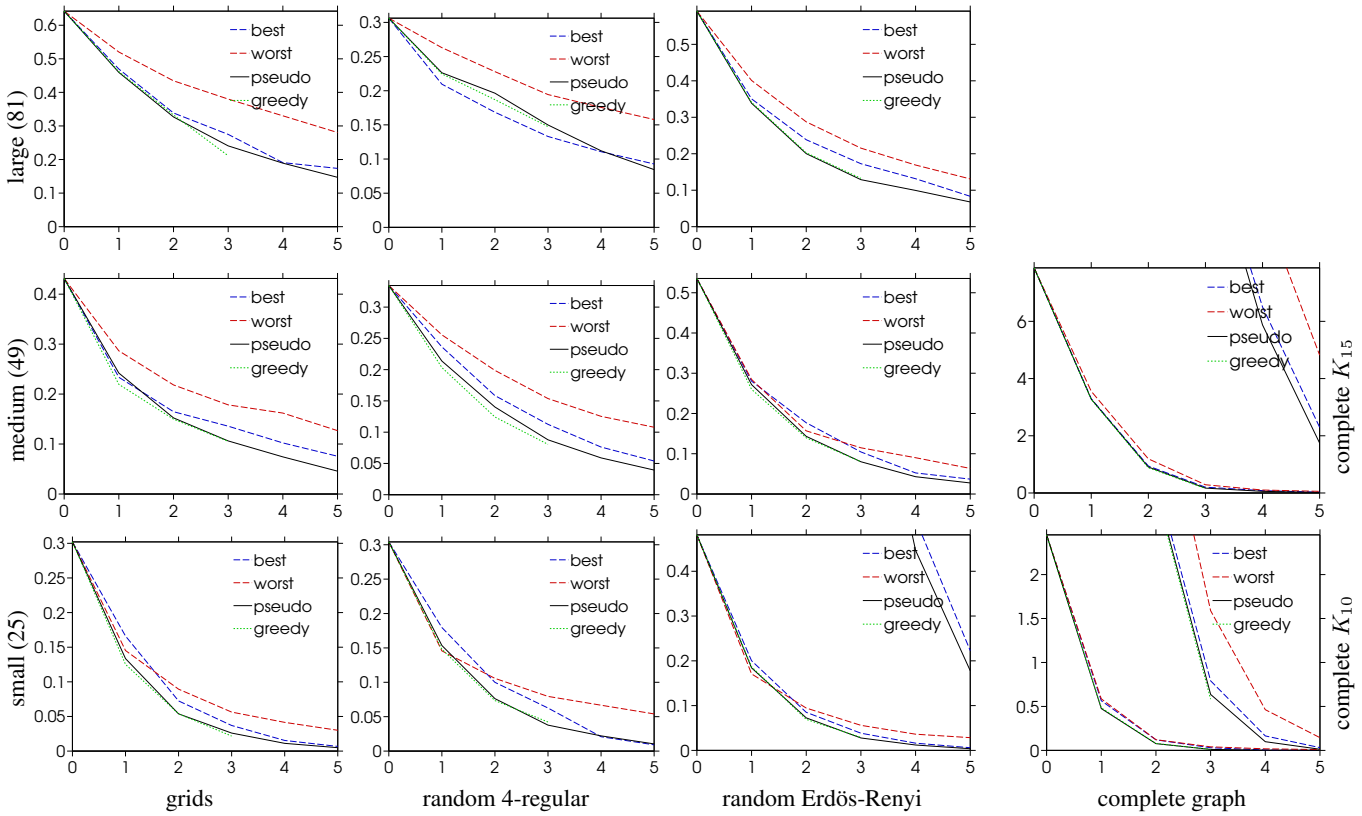

Figure 10: Mixed, zoomed around Bethe $[-6, 6]$

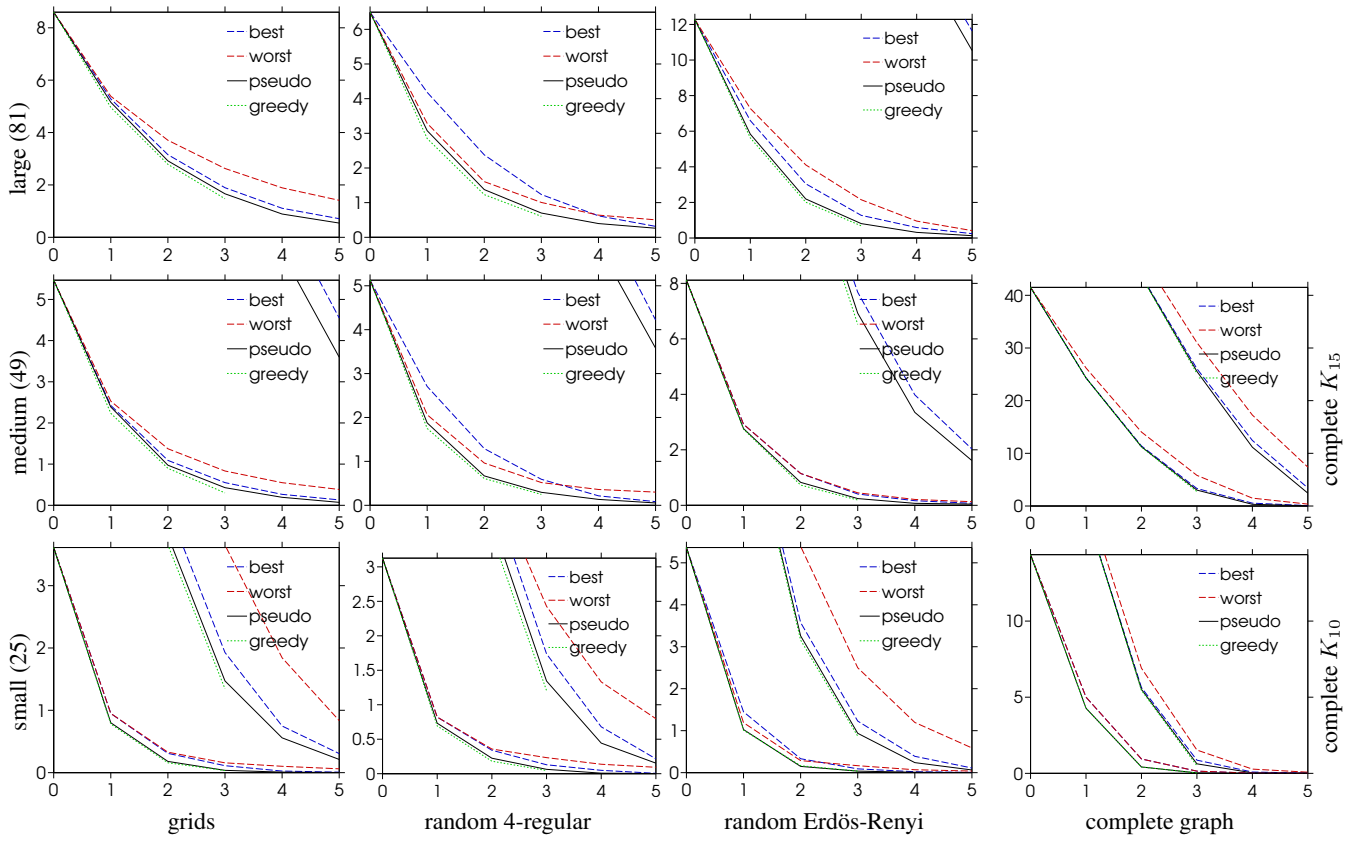

Figure 11: Mixed, zoomed around Bethe $[-12, 12]$

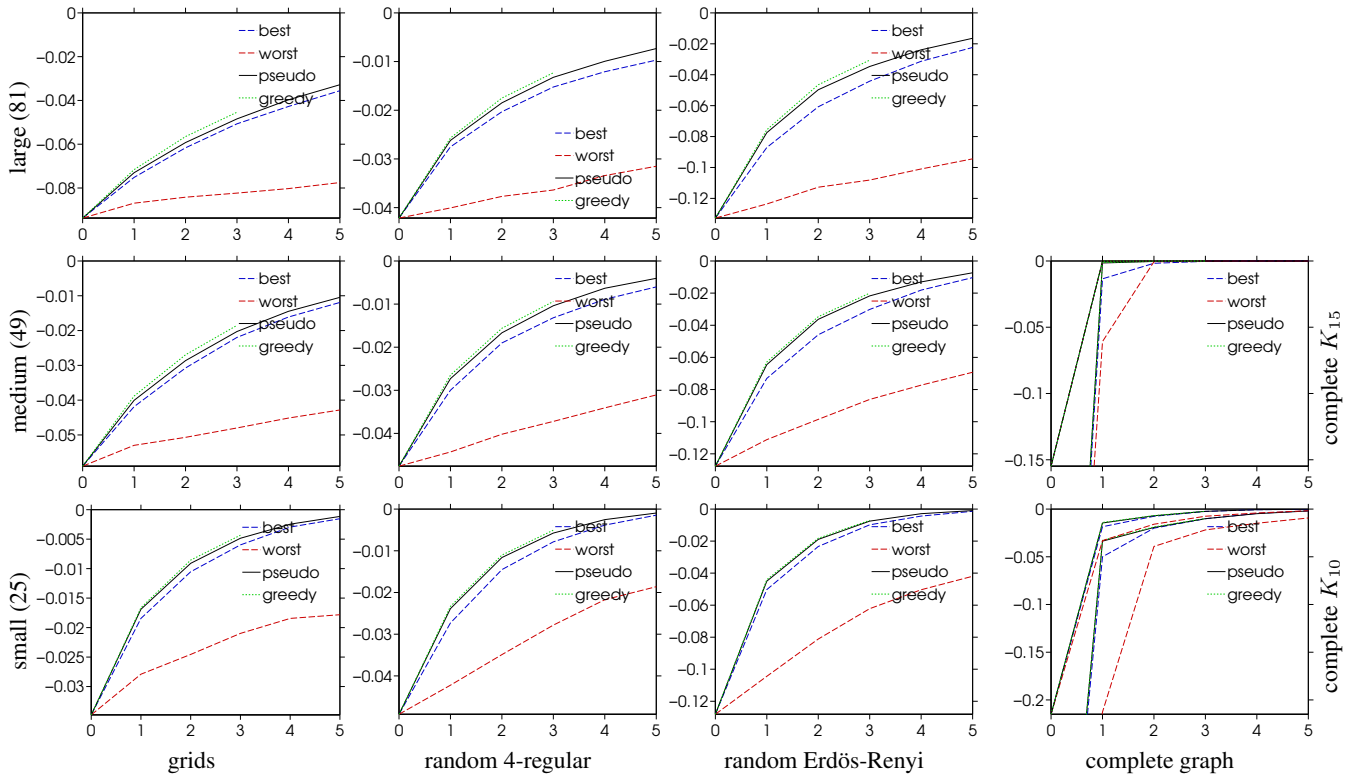

Figure 12: Attractive, zoomed around Bethe $[0, 2]$

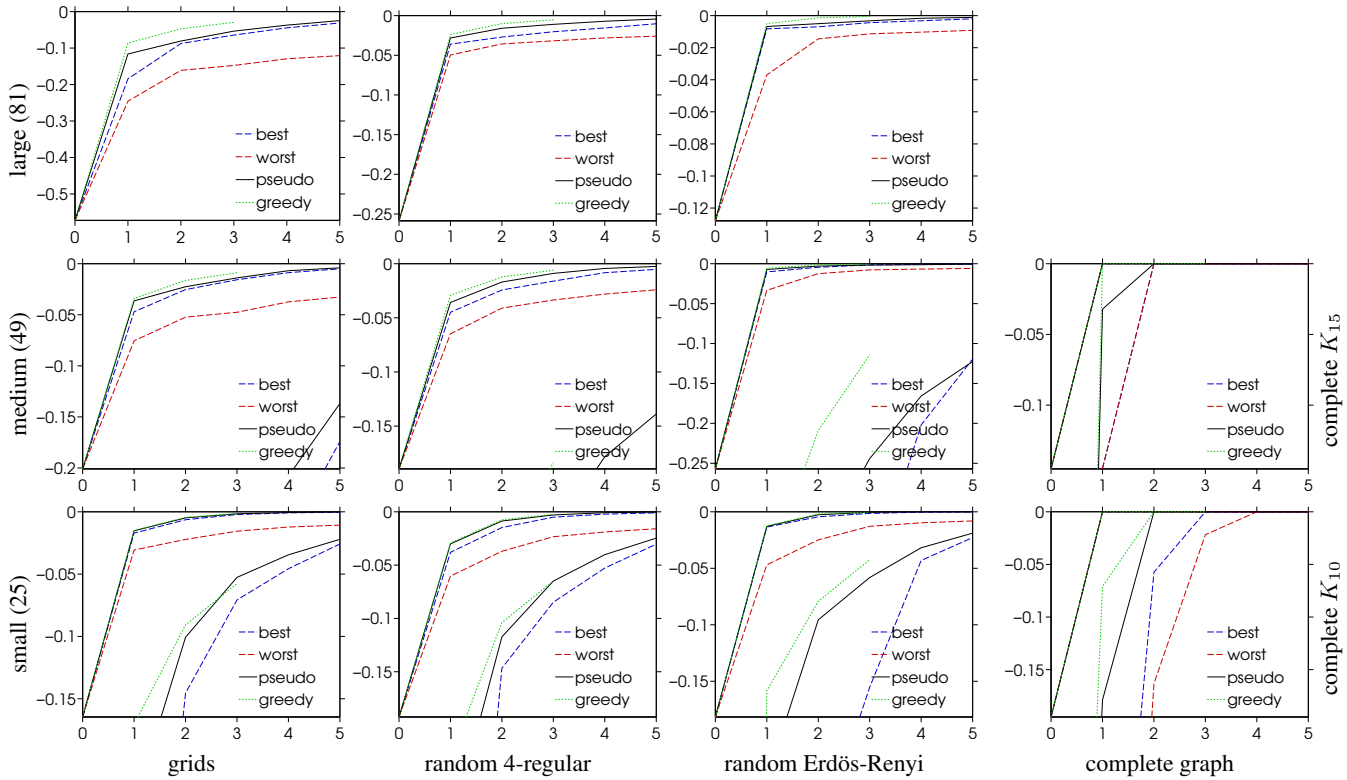

Figure 13: Attractive, zoomed around Bethe $[0, 6]$

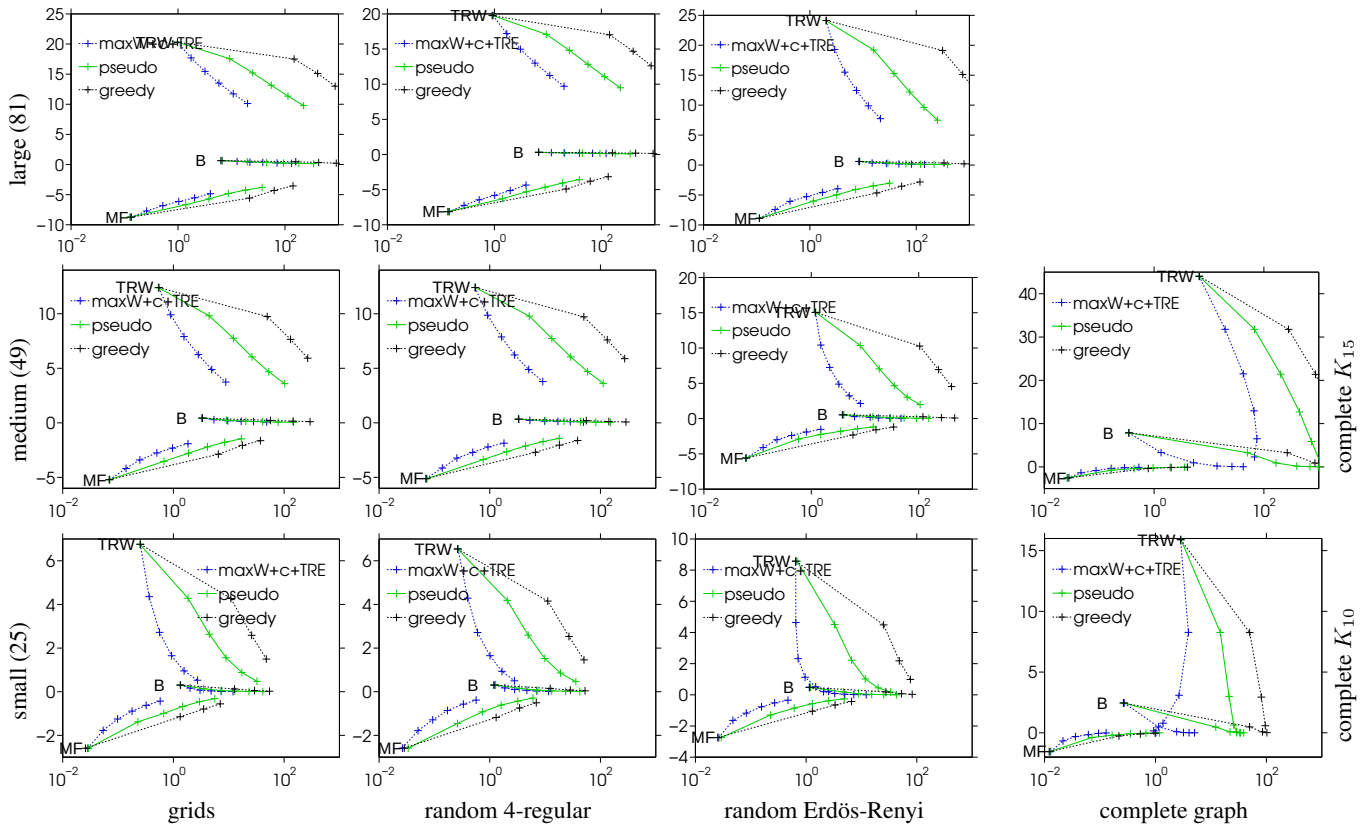

Figure 14: Mixed $[-6, 6]$ timings (in secs, log scale, these give an overall sense but may be sensitive to implementation details and convergence thresholds)

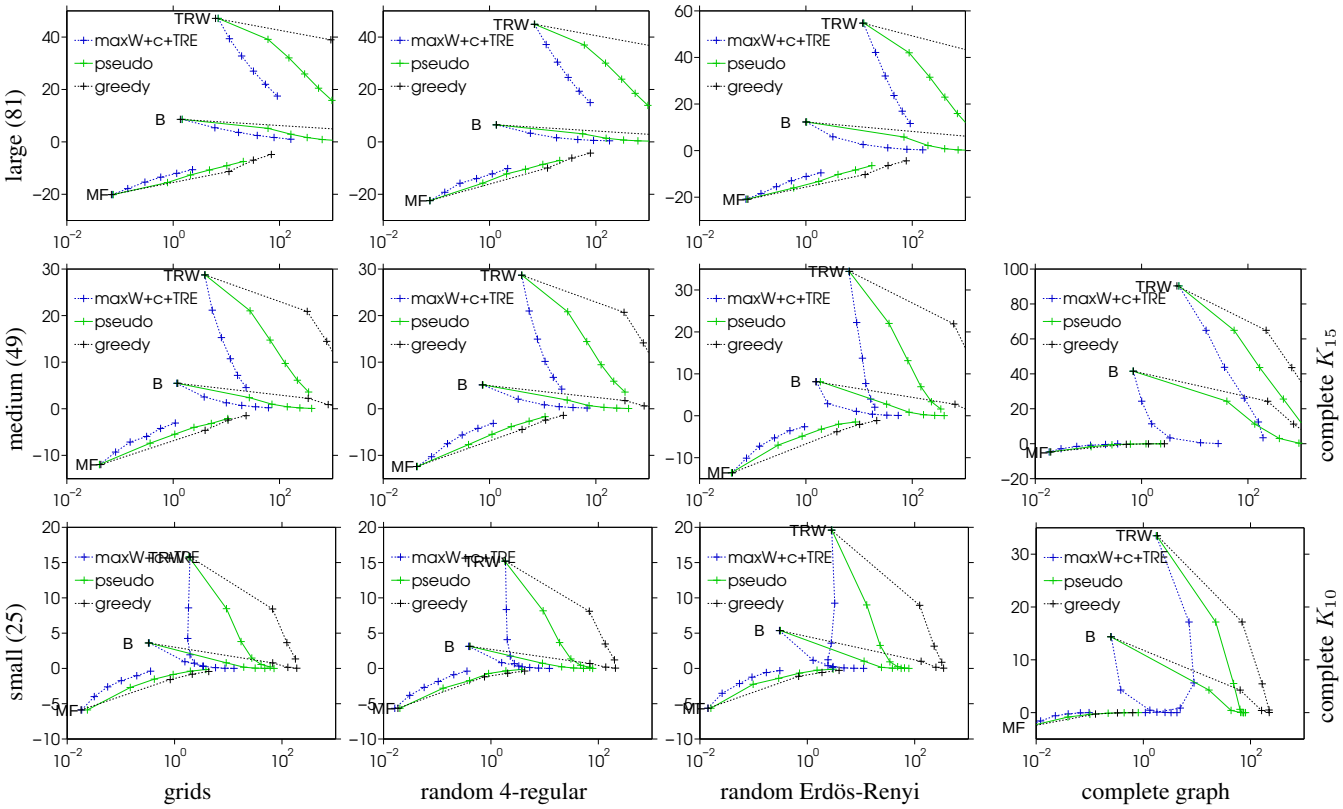

Figure 15: Mixed $[-12, 12]$ timings (in secs, log scale, these give an overall sense but may be sensitive to implementation details and convergence thresholds)

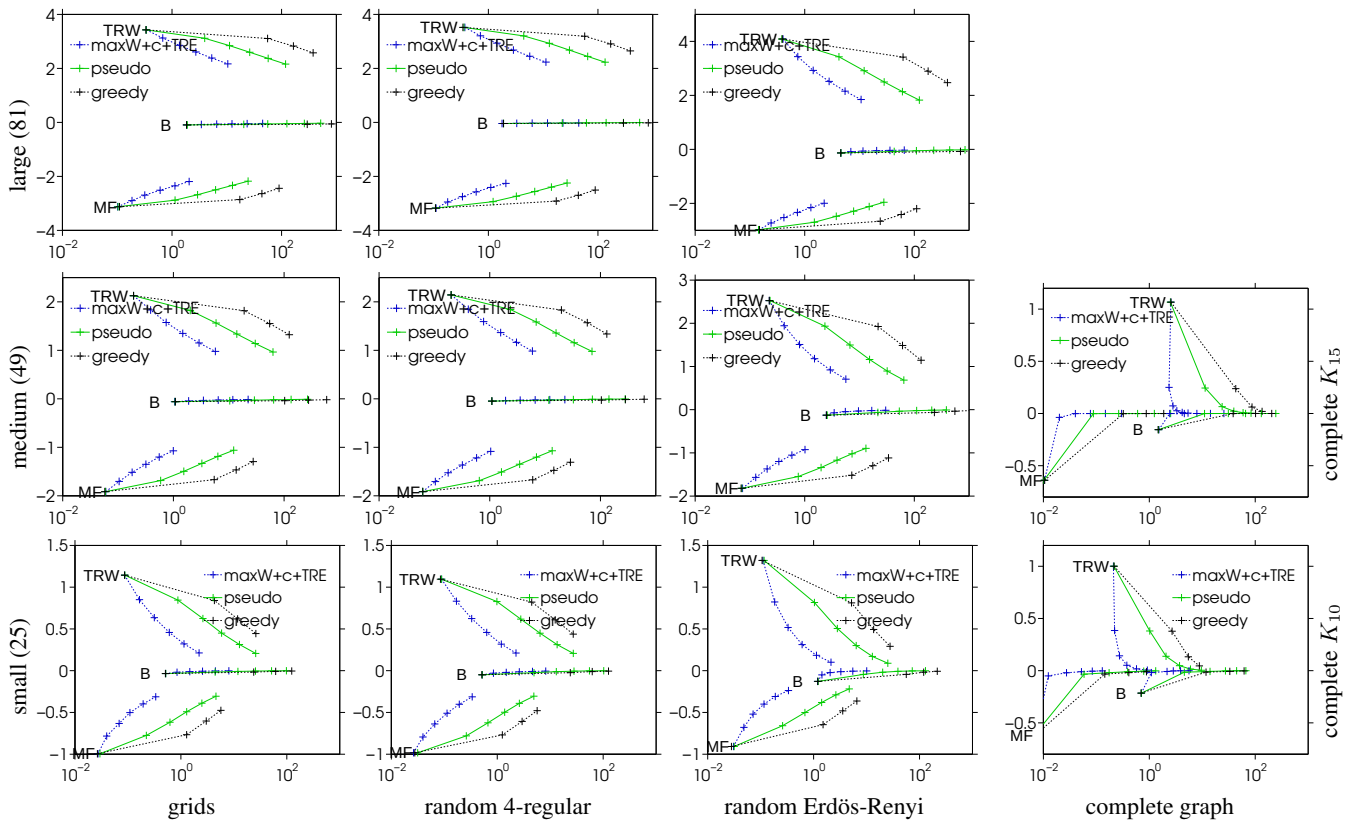

Figure 16: Attractive $[0, 2]$ timings (in secs, log scale, these give an overall sense but may be sensitive to implementation details and convergence thresholds)

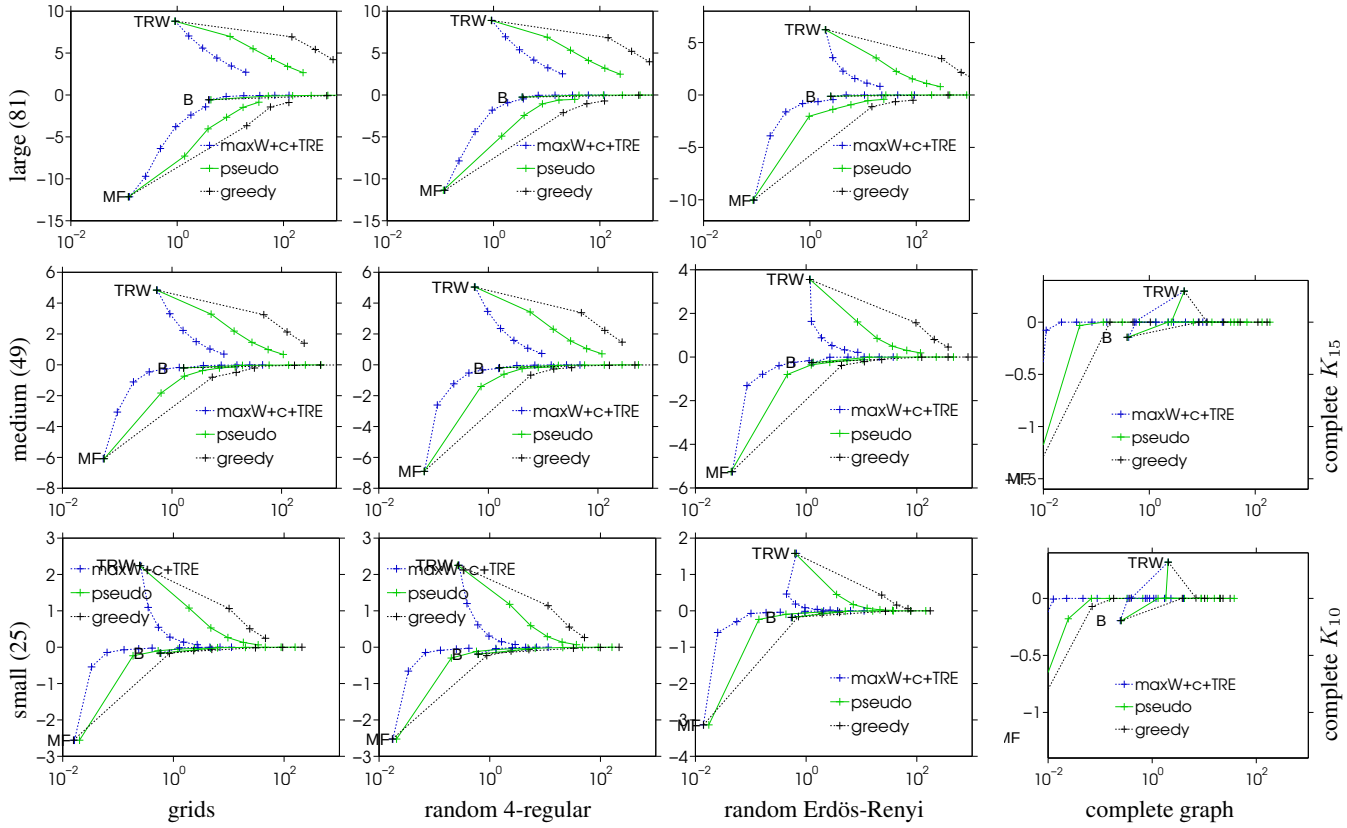

Figure 17: Attractive $[0, 6]$ timings (in secs, log scale, these give an overall sense but may be sensitive to implementation details and convergence thresholds)

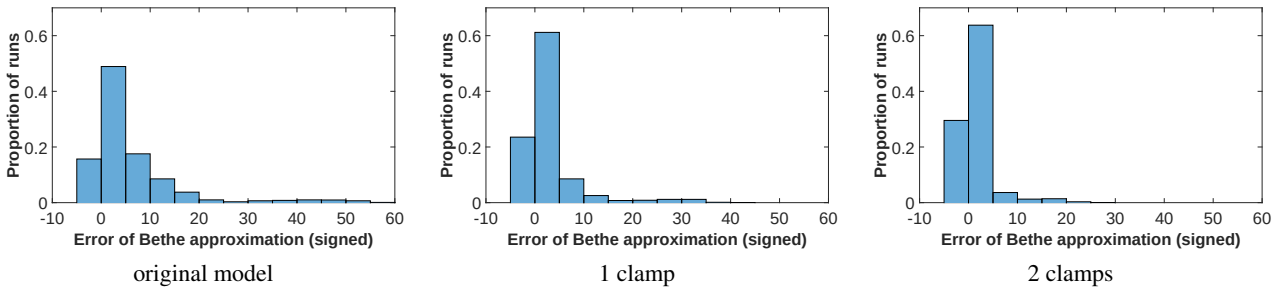

Figure 18: Histograms of occurrences of *signed* error bins of $\tilde{A}_B(\theta) - A(\theta)$ for Bethe, across all runs of *mixed* models. This shows that the error is dominated by being too *high*, particularly before clamping, as would be expected from the reasoning in §5.2.

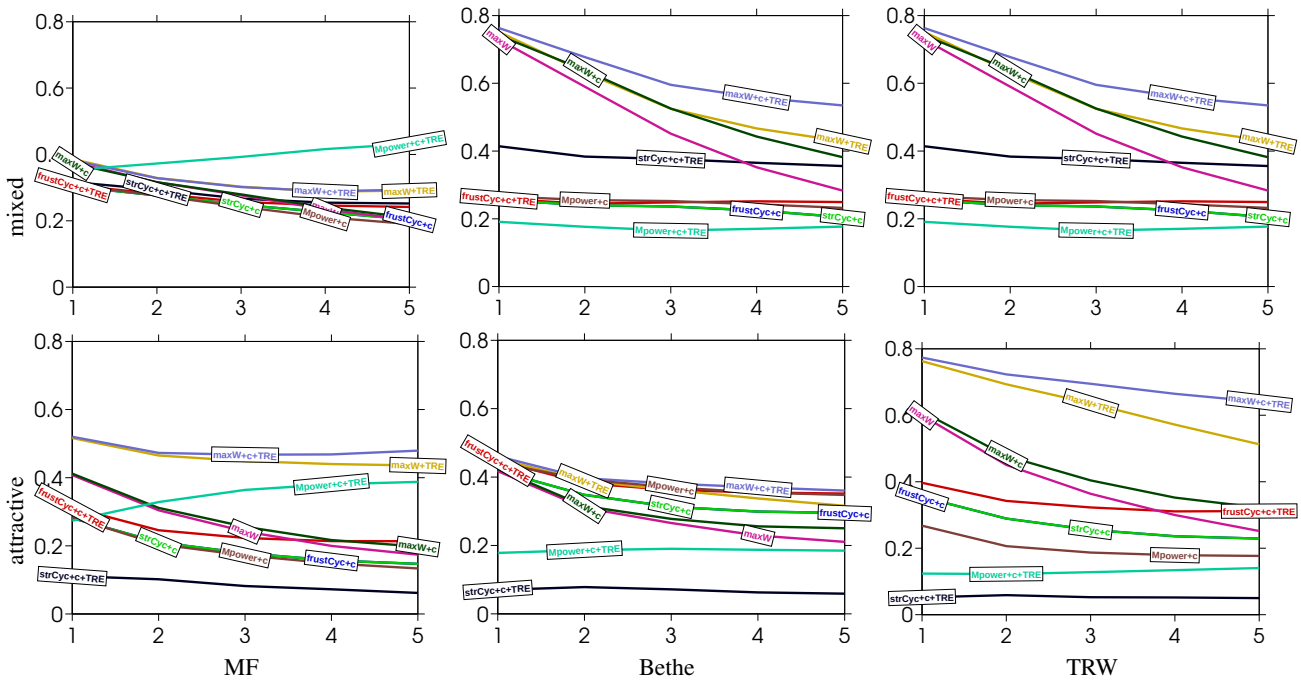

Figure 19: Fraction of the time each heuristic picks the same variable to clamp as pseudo-greedy at that specific clamp step. Note that for mixed models, by our choice Bethe mimics TRW (empirically the best option: Bethe is not a bound in this case).