Leaning Graphical Model Structures using L1-Regularization Paths (addendum)

Mark Schmidt and Kevin Murphy

Computer Science Dept.
University of British Columbia
{schmidtm,murphyk}@cs.ubc.ca

1 Introduction

This document contains additional information and results for the paper 'Learning Graphical Model Structure using L1-Regularization Paths' (AAAI 2007), that we were not able to fit into the required page limit. In particular, this document discusses:

- The LARS-MLE algorithm, an efficient algorithm that returns the unpenalized Maximum Likelihood Estimates (MLEs) for all non-zero subsets of variables encountered along the LARS regularization path.
- The Two-Metric Projection algorithm used for L1-regularized Logistic Regression.
- The L1PC algorithm, a relaxed form of the L1MB algorithm that allows scaling to much larger graphs.
- Extensions the algorithms to interventional (experimental) data.
- Extended experimental results.

2 LARS-MLE

Using X to denote the n instance by d feature design matrix, y to denote the n instance regression target, and θ as the regression parameters, the constrained variant of the LASSO regression and feature selection problem is defined as follows (Tib96):

$$\begin{aligned} \min_\theta \frac{1}{2}||X\theta - y||_2^2 \\ s.t.||\theta||_1 \le t \end{aligned} \tag{1}$$

The non-negative parameter t controls the scale of the restriction on the regression parameters. Many approaches have been proposed to solve this particular problem. However, in many scenarios we will typically want to solve the problem for more than a single value of the parameter t. The Least Angle Regression (LARS) procedure provides an efficient algorithm for computing the optimal solution for all values of t (EJHT04). The LARS algorithm computes all of the discontinuities along the regularization path (ie. values of t where variables become exactly 0, or move away from exactly 0), while all intermediate values can be computed based on interpolation (since the regularization path is linear between discontinuities). The asymptotic runtime of the LARS algorithm is $O(nd^2)$ (for $n > d$), the same cost as finding the unpenalized Maximum Likelihood Estimate (MLE), which is itself obtained as the last discontinuity on the regularization path.

Efficient calculation of the entire regularization is appealing when we are interested in the restricted MLE. However, model selection criteria such as the Bayesian Information Criteria (BIC) and Akaike Information Criteria (AIC) require the unrestricted MLEs for the non-zero set of variables. The 'LARS-MLE' algorithm is a simple extension of the LARS algorithm that also returns the MLE for each non-zero subset of variables encountered along the regularization path.

Computing the MLE for a non-zero subset of variables can be done at a cost of $O(nd^2)$ using standard matrix factorization methods (ie. Cholesky factorization). Thus, a naive implementation could compute the MLEs for the $O(d)$ subsets encountered along the regularization path at a cost of $O(nd^3)$. However, note that the non-zero subset of variables changes by only a single variable at each discontinuity. Thus, given the matrix factorization used for finding the previous MLE along the regularization path, the current matrix factorization (with one variable added or removed) can be obtained by computing the corresponding row of X^TX at a cost of $O(nd)$ (in the case of additions), and performing a low-rank matrix update at a cost of $O(d^2)$ (ie. using a rank-1 Cholesky factorization update). The updated MLE values can be obtained with this updated matrix at a cost of $O(d^2)$ by performing two triangular back-substitions. Given that the initial matrix factorization costs $O(n)$, the entire set of MLE values can be computed using this updating scheme at a cost of $O(nd^2)$, which is again the same cost as computing the MLE with all variables active (and again the MLE is the final value found by the algorithm when $n > d$).

One gains further efficiency in the LARS-MLE procedure by noting that the LARS algorithm already incorporates updating of a matrix factorization of X^TX. Thus, the only cost incurred by additionally computing all MLE values along the regularization path is the cost of the back-substitutions. Algorithm 1 gives the LARS-MLE algorithm. It is identical to the LARS algorithm, except the additional line that computes θ_i^{MLE}. For simplicity, we have outlined the Least Angle Regression variant of LARS, the LASSO variant of LARS is obtained as a simple modification that removes variables from the Active Set that change sign (truncating γ when this happens).

3 Two-Metric Projection for L1-Regularized Logistic Regression

Designing efficient methods to solve L1-regularization problems has been an active area of recent research. While the LARS proved efficient enough for our purposes, the choice of an efficient algorithm for L1-Regularized Logistic Regression

Algorithm 1 LARS-MLE

```
$\theta_0 = \mathbf{0}$                                  {Initial Point on Regularization Path}
$\mu = \mathbf{0}$                                       {Initial value of Xw}
$A = \{\}$                                               {Active Variables (initially empty)}
$I = \{1, 2, ..., d\}$                                   {Inactive Variables (initially full)}
$R = []$                                                 {Initial Cholesky Factorization}
$i = 1$
while $I \neq \{\}$ do
  $c = X^T(y - \mu)$                                     {Correlations (ie. Gradient)}
  $C = \max_{j \in I} |c(j)|$
  $j = \arg\max_{j \in I} |c(j)|$                        {Inactive Variable with Max Correlation}
  $R = cholupdate(R, X_A, X_j)$                          {Add $j$ to Cholesky of $X_A^T X_A$}
  $A = A \bigcup j$                                      {Add j to Active Set}
  $I = I \backslash j$                                   {Remove j from Inactive Set}
  $s = sign(c_A)$
  $G = R \triangleright (R^T \triangleright s)$          {Backsubstitute to compute $(X_A^T X_A)^{-1} s$}
  $z = \frac{1}{\sqrt{s^T G}}$
  $w = s \bullet (Gz)$                                   {$\bullet$: element-wise product}
  $u = X_A(w \bullet s)$                                 {Equiangular Vector}
  $a = X^T u$
  if $|A| < d$ then
    $\gamma = \min^+_{j \in I}\{\frac{C - c(j)}{z - a(j)}, \frac{C + c(j)}{z + a(j)}\}$   {Minimum over positive values}
  else
    $\gamma = C/z$                                       {Stop at MLE when all variables active}
  end if
  $\theta_i = \mathbf{0}$
  $\theta_i(A) = \theta_{i-1}(A) + \gamma(w \bullet s)$  {Compute next discontinuity $\theta$ along path}
  $\theta_i^{MLE}(A) = R \triangleright (R^T \triangleright X_A^T y)$   {MLE of Active Set}
  $i = i + 1$
end while
```

was less clear. We initially used the 'IRLS-LARS' algorithm of (LLAN06), which was shown to be superior to several competitive alternatives. However, we found that L1-Regularized Logistic Regression problems could be solved much more efficiently by re-formulating the problem as a bound-constrained optimization problem and using a 'Two-Metric Projection' strategy (Ber99). This algorithm has similar convergence properties to the IRLS-LARS procedure, but has a substantially smaller iteration cost. As opposed to LARS, this algorithm only solves the problem for a single value of the regularization parameter. However, an optimal solution for a 'close' regularization parameter can be used to 'warm-start' the optimization for a new paramter (this can substantially reduce the number of iterations needed).

Algorithm 2 outlines the Projection algorithm. This algorithm splits θ into non-negative components $\theta^+ = max(0, \theta)$ and $\theta^- = -min(0, \theta)$, such that $\theta = \theta^+ - \theta^-$. It also uses the 'scaled norm' variant of L1-regularization rather than the 'constrained norm' variant. This gives a problem with twice the number of variables, but turns the non-differentiable problem into a differentiable problem with bound constraints on the variables:

$$\min_{\theta^+,\theta^-} L(\theta^+,\theta^-) = f(\theta^+ - \theta^-) + \lambda \sum_i [\theta_i^+ + \theta_i^-] \quad s.t. \forall_i \ \ \theta_i^+ \geq 0, \theta_i^- \geq 0$$

In the above, $f(\theta)$ represents the Logistic Regression negative log-likelihood for parameters θ. Using $\sigma = \frac{1}{1+\exp(-y \bullet X\theta)}$, $f(\theta) = -\sum \log \sigma$, $\nabla f(\theta) = -X^T(y \bullet (1-\sigma))$, and $\nabla^2 f(\theta) = X^T diag(\sigma \bullet (1-\sigma))X$. In Algorithm 2, we use θ^* to denote the concatenation of θ^+ and θ^- into one vector. The derivatives of $L(\theta)$ with respect to this change of variables are: $\nabla L(\theta^*) = [\nabla L(\theta); -\nabla L(\theta] + \lambda$, and $\nabla^2 L(\theta^*) = [\nabla^2 L(\theta) \quad -\nabla^2 L(\theta); -\nabla^2 L(\theta) \quad \nabla^2 L(\theta)]$.

Algorithm 2 L1-Regularized Logistic Regression with Two-Metric Projection

```
$\theta^* = \mathbf{0}$        {Initialize to 0 (or alternately to a nearby solution if available)}
while 1 do
  $L_{old} = L(\theta^*)$
  $A = \{i | \theta_i^* \geq \epsilon, \nabla L(\theta^*) < 0\}$        {Active Set of Variables}
  $d = \mathbf{0}$
  $d_A = \nabla_A^2 L(\theta^*)^{-1} \nabla_A L(\theta^*)$        {Newton direction for Active sub-matrix/vector}
  $t = 1$
  while $L([\theta^* + td]^+) > L(\theta^*) + ct\nabla L(\theta^*)^T d$ do
    $t = \{i | i \in (0,t)\}$        {t selected by cubic interpolation}
  end while
  $\theta_A^* = [\theta_A^* + td]^+$        {Take step, project into non-negative orthant}
  if $||L(\theta^*) - L_{old}|| < \epsilon$ then
    return        {Terminate when insufficient progress}
  end if
end while
```

4 L1PC

Our proposed L1MB algorithm requires $O(nd^2)$ time per node in the Gaussian case, and $O(nd^3)$ time per node (assuming a fixed number of Newton iterations) in the binary case. For graphs where we consider a very large number of nodes, these runtimes make the algorithm prohibitive. However, note that for the purpose of pruning a DAG-Search algorithm, we are only interested in a subset of the node's Markov Blanket (ie. its children and parents, but not co-parents). This leads us to propose the more efficient 'L1-Regularized selection of Parents and Children' (L1PC) algorithm (by analogy with the MMPC algorithm).

Assuming that L1-Penalization is an ideal variable selector, L1MB regresses each node on all other nodes, and subsequently returns all of the node's parents, children, and co-parents. If instead we regressed separately on disjoint subsets of the nodes, we would still select the parents and children, but we might include additional nodes (ie. ancestors who are not directly connected but are not d-separated by the subset) or not include co-parents (ie. if the common child is not included in the subset). For the purposes of identifying a node's Markov Blanket, excluding co-parents is problematic, but it is reasonable if we are only interested in identifying direct links. The L1PC algorithm is a modification of the L1MB algorithm that builds the set of potential edges incrementally. Rather than regressing on all other nodes, it first regresses on several disjoint subsets of nodes (where the subset sizes are chosen small enough to be computationally efficient). Subsequently, in L1PC the inactive variables from these small selection problems are discarded and the active variables are used to form the subsets in next iteration. The number of iterations used will be dependent on the number of nodes. For our experiments, we used initial subsets of size 10 (allowing very efficient estimation of L1-penalized Linear/Logistic Regression regularization paths), and in the second iteration we regressed on the remaining active variables. This strategy allowed the approach to be applied efficiently to graphs with hundreds of nodes (assuming sparsity), since (in the discrete case) it replaces the d^3 term with $\frac{d}{10}10^3$ in the first iteration, and a^3 to estimate the final edge set (where a is the number of variables selected from the first step, typically much smaller than d if the graph is sparse).

The L1PC strategy does not produce the same set of edges as the L1MB algorithm, and is not a Markov Blanket estimation procedure (although both will return the parents and children). In the extreme case where the first iteration uses subsets of size 1, the incremental L1PC algorithm will (assuming perfect variable selection) select all nodes that are ancestors, descendants, and nodes that share a common ancestor with the node of interest. As opposed to the L1PC algorithm, the incremental L1PC algorithm can hence remove coparents that do not share a common ancestor. Although undesirable in the undirected case, this leads to a reduction of the search space in the directed case. Unfortunately, this reduction needs to be balanced against the increased number of edges that may be found by not conditioning on a pruned co-parent (ie. through explaining away). In our experiments, we found that L1MB and L1PC achieved a similar level of pruning, while L1PC was substantially faster for graphs with hundreds of nodes.

5 Inverventional Data

Using only observational data, it is not possible to distinguish between Markov Equivalent graphs, and hence it may not possible to determine edge directionality. However, we can determine directionality with interventional data. A perfect intervention is a scenario where a node's value is clamped to a specific value (Pearl00). Since parents should not be affected by interventions on their children, this causes an asymmetry that allows the identification of the causal directionality, thus allowing us to distinguish between Markov Equivalent graphs. The methods presented here are easily extended to allow interventional data as follows: (i) when performing an L1-penalized regression (or computing an MLE) with target node i, do not include cases where i was set by intervention, and (ii) ignore the contribution to the BIC score of nodes that were set by intervention.

6 Extended Results

In this section, we give extended experimental results, where we vary several parameters. This section contains the following results:

- Results with different data set sizes.
- Results with different random network parameters.
- Results when using interventional Data.
- Results with the L1PC algorithm

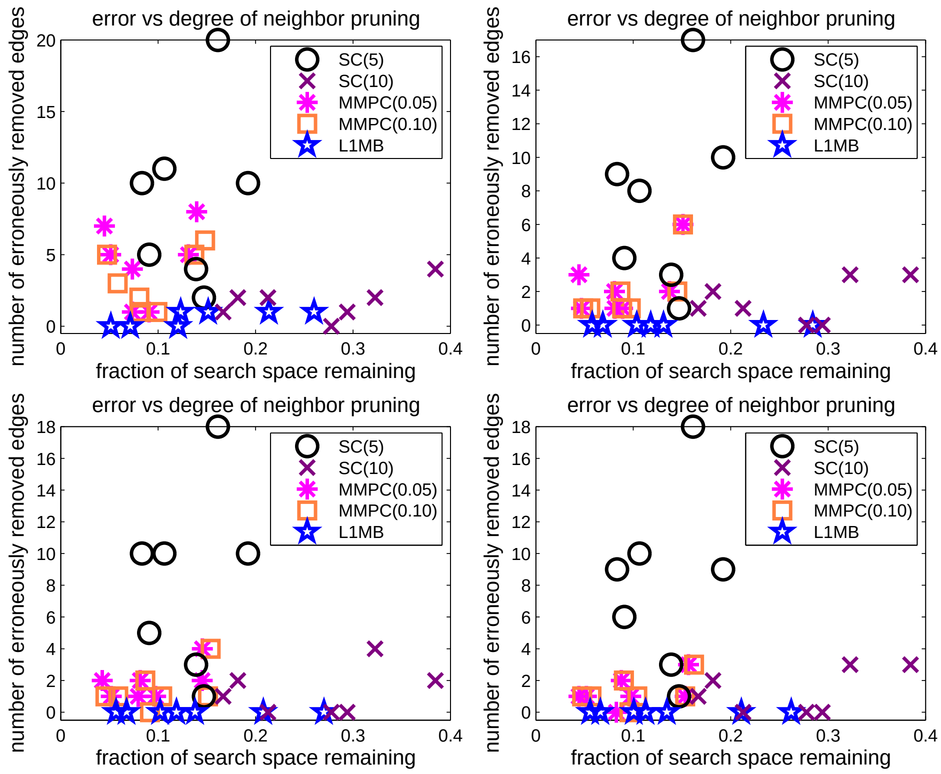

Fig. 1. False negatives plotted against the ratio of edges removed by different pruning strategies for different data set sizes. Top left: 1000 samples. Top right: 5000 samples. Bottom left: 10000 samples. Bottom right: 15000 samples.

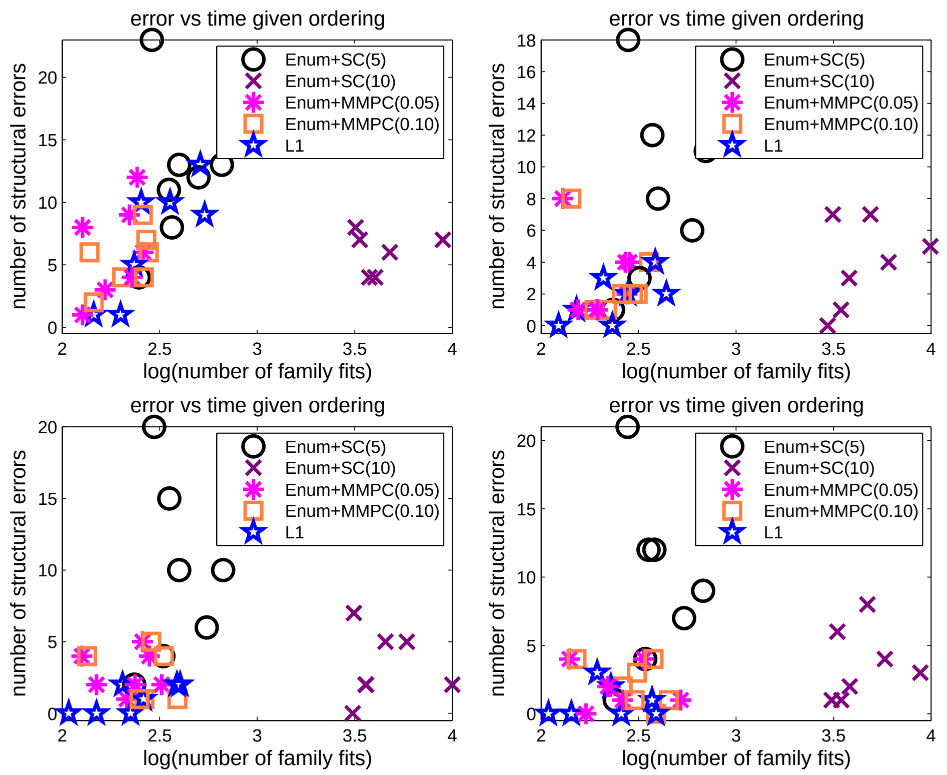

Fig. 2. Number of Structural Errors after Variable Selection plotted against the Number of Familty Fits required under a topological ordering for different data sets sizes. Top left: 1000 samples. Top right: 5000 samples. Bottom left: 10000 samples. Bottom right: 15000 samples.

6.1 Different Data Set Sizes

We first examine the effect of changing the data set size on the results of variable selection. Using the same parameters as the AAAI paper, we repeated the 'Edge Pruning' and 'Parent Selection given an ordering' experiments with data set sizes of 1000, 5000, 10000, and 15000 data samples (Figures 1 and 2).

From these results, we see that pruning based on pairwise statistics (SC) produces poor results, irrespective of the data set. In contrast, MMPC and L1MB are more effective as the data set size increases. For edge pruning, L1MB erroneously removes 1 edge from 4 of the data sets when only 1000 samples are used, but does not erroneously remove any edges for the larger sample sizes. In contrast, MMPC still erroneously removes several edges even in the largest sample size. Under a topological ordering, L1 variable selection recovers the true structure on 0 data sets for 1000 samples, 2 data sets for 5000 samples (insurance and mildew), 3 data sets for 10000 samples (insurance, water, and mildew), and

4 data sets for 15000 samples (insurance, water, barley, and carpo). In contrast, MMPC recovers the true structure in 1 data set for 15000 samples (alarm), and does not recover the true structure in any other data sets or for any smaller sample sizes (while SC(10) recovers the true structure in the mildew data set for each of the 5000 sample and 10000 sample experiments, but at a substantially higher cost).

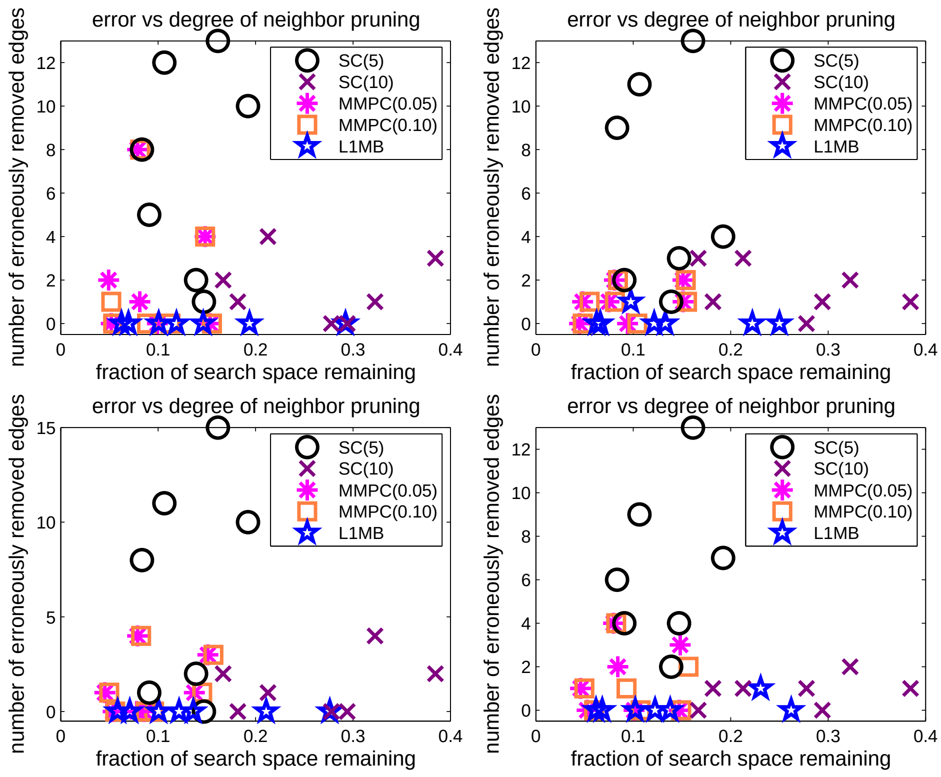

Fig. 3. False negatives plotted against the ratio of edges removed by different pruning strategies for different parameterizations.

6.2 Different Parameterizations

To test the robustness of our results, we re-ran all experiments using 4 additional different random parameterizations of the graphs (for the 3 real data sets, we used different random samples for the training data). The pruning and results under a correct ordering are plotted in Figures 3 and 4.

The edge pruning results show that the L1MB consistently prunes large areas of the search space without introducing false negatives. However, under different

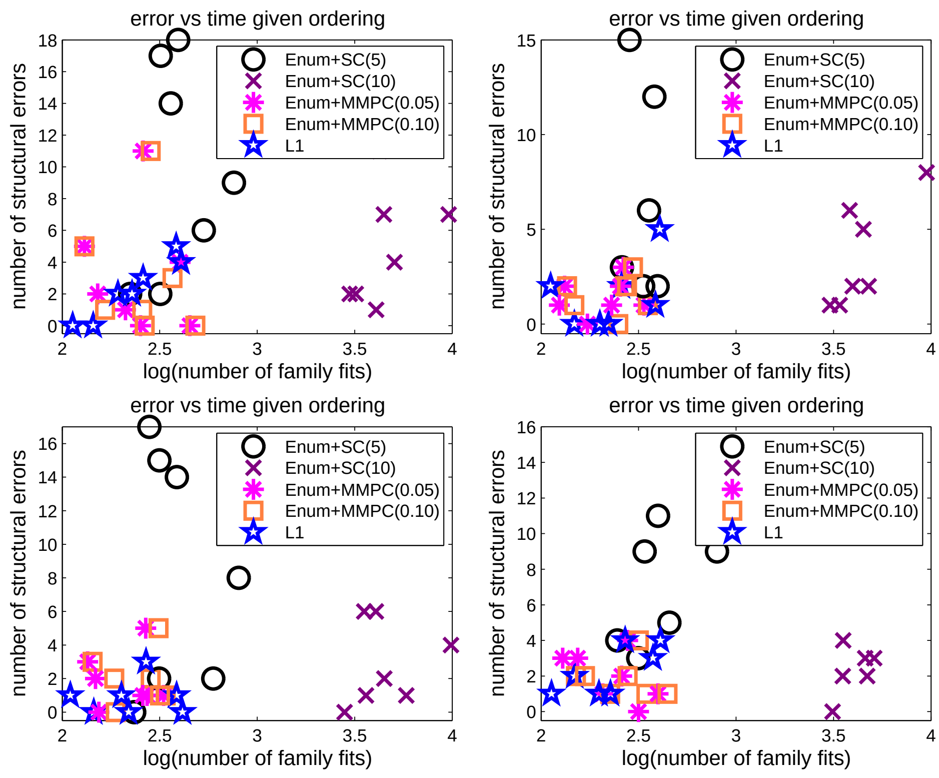

Fig. 4. Number of Structural Errors after Variable Selection plotted against the Number of Familty Fits required under a topological ordering for different parameterizations.

parameterizations false negatives were introduced for 2 of the data sets (among a possible 35, when including those in the main paper). In contrast, the other strategies included at least 1 false negative on most data sets. Regarding accuracy under a correct ordering, L1 variable selection is the method that most often recovers the true structure, but in some cases has up to 5 structural errors. However, note that in the MMPC method, we prespecified that subsets larger than 5 would not be examined, while the L1MB method made no such assumption.

Figure 5 plots the search results under the 4 samplings. These show similar trends across the experiments. DAG+L1MB is consistently among the best for the synthetic data sets, while the OrderL1 often produced better results on the real data sets (8-10). DAG+MMPC tends to have among the lowest structural errors (even though BIC and test set likelihood are lower).

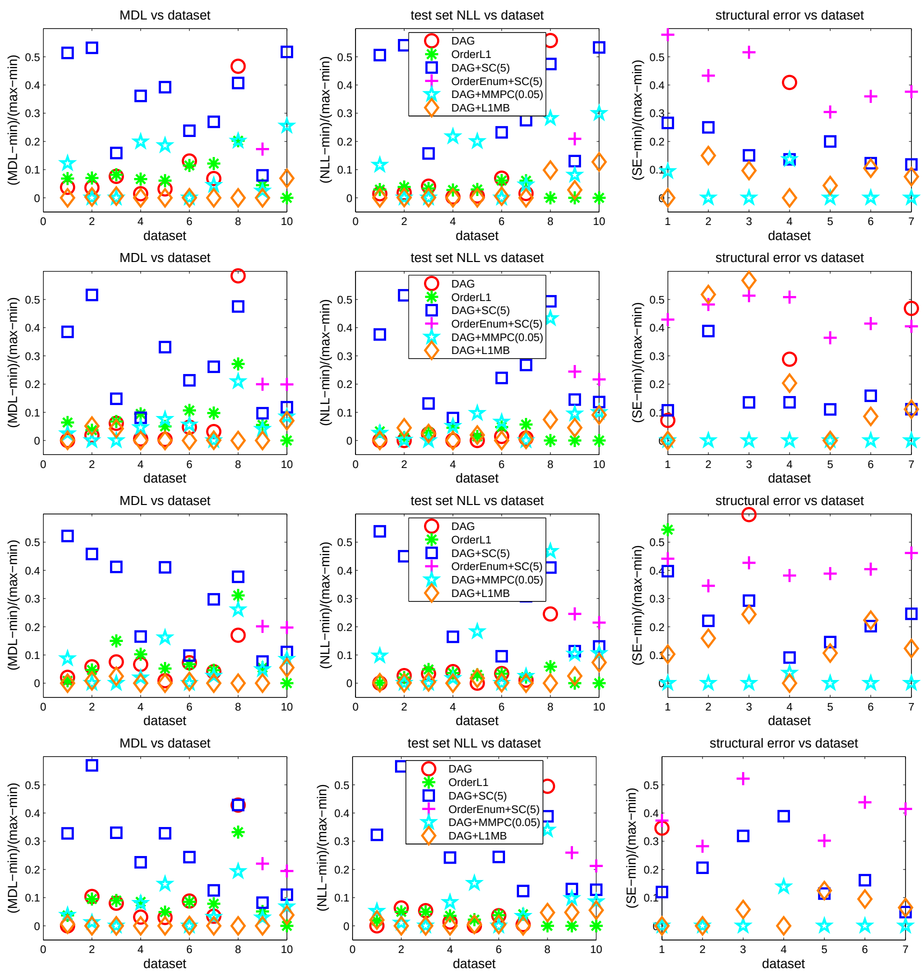

Fig. 5. Structure Search Results under different parameterizations and random samplings of the data.

6.3 Interventional Data

We modified our experimental set-up, such that for each sampled data instance, one of the nodes was set by a perfect intervention. We then re-ran the experiments, taking these interventions into account. The experimental set-up for these experiments was otherwise identical, except that we do not include results on the MMPC algorithm, since it does not support interventions in its current form. These results are shown for different parameterizations in Figure 6. In these experiments, the DAG+L1MB algorithm dominated, achieving close to the lowest score across all data sets and experiments in terms of MDL, test set negative log-likelihood, and structural errors. Note that for these experiments, we measured

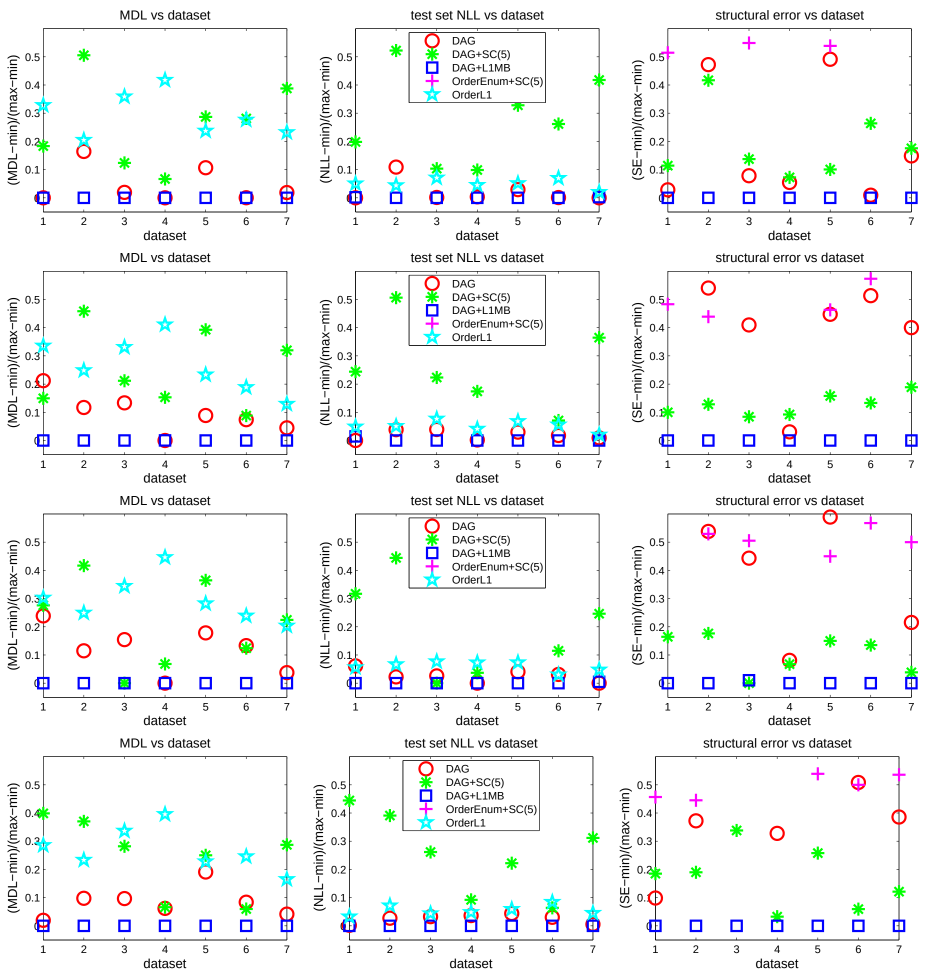

Fig. 6. Structure Search Results under different parameterizations and random samplings of the data, where 1 node in each sample was set by intervention.

the structural error in terms of hamming distance from the true DAG rather in terms of DPAG errors (the true DAG should be identifiable with experimental data).

6.4 L1 Parents and Children (L1PC)

Finally, we repeated the pruning experiment with the L1PC algorithm. This is plotted in Figure . From this Figure, we see that L1PC achieves a similar level of pruning and accuracy as L1MB, while the L1PC algorithm is much faster.

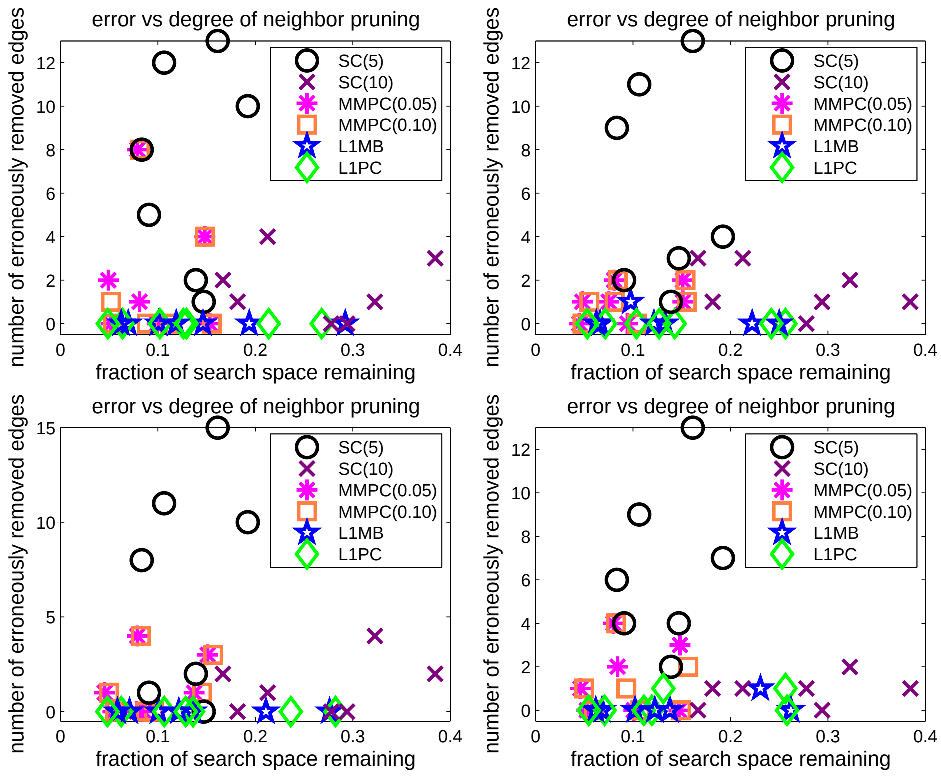

Fig. 7. False negatives plotted against the ratio of edges removed by different pruning strategies for different parameterizations.