Best-Choice Edge Grafting for Efficient Structure Learning of Markov Random Fields

Walid Chaabene
Department of Computer Science
Virginia Tech
Blacksburg, USA
walidch@vt.edu

Bert Huang
Department of Computer Science
Virginia Tech
Blacksburg, USA
bhuang@vt.edu

***Abstract*—Incremental methods for structure learning of pairwise Markov random fields (MRFs), such as *grafting*, improve scalability by avoiding inference over the entire feature space in each optimization step. Instead, inference is performed over an incrementally grown active set of features. In this paper, we address key computational bottlenecks that current incremental techniques still suffer by introducing *best-choice edge grafting*, an incremental, structured method that activates edges as groups of features in a streaming setting. The method uses a reservoir of edges that satisfy an activation condition, approximating the search for the optimal edge to activate. It also reorganizes the search space using search-history and structure heuristics. Experiments show a significant speedup for structure learning and a controllable trade-off between the speed and quality of learning.**

I. Introduction

A powerful family of approaches for learning the structure of Markov random fields (MRFs) is based on minimizing ℓ_1-regularized scores such as the negative log likelihood [2] of a fully connected MRF. The ℓ_1 regularization reduces the parameters of irrelevant edges to zero. The main challenge when using these methods is that, for large MRFs, the feature space becomes extremely large and causes an overwhelming computational cost. Active-set methods, such as grafting [10], [15], [18], were introduced to promote more scalability. Despite the benefits of active-set learning, grafting retains significant computational costs: As a mandatory pre-learning step, grafting computes sufficient statistics of all possible variable pairs to enable greedy activation tests on the entire search space, and each iteration of grafting requires a search over the large combinatorial space of all possible edges.

This paper introduces *best-choice edge grafting*, an active-set method that activates edges in a streaming fashion as groups of parameters. The method is agnostic to the underlying inference method. We derive an edge-activation test using a structured group-ℓ_1 learning objective. Structure learning is performed in a priority order: Edges are assigned different search priorities. We use a combination of random sampling and a min-heap priority queue [16] to efficiently explore the prioritized search space. The method performs "online" edge activation using a control sample of candidate edges stored in a limited-memory reservoir. Search priorities are updated based on the search history and structural information derived from the partially constructed MRF structure learned so far. These strategies allow on-demand computation of sufficient statistics for edges that are likely to be activated, which eliminates the heavy computation of grafting's pre-learning phase. Best-choice edge grafting can start—and often finish—learning well before grafting is able to begin learning. We also introduce a trade-off parameter to balance the speed of learning and the quality of learned MRFs. Our experiments show that best-choice edge grafting scales better to large datasets than grafting. Synthetic experiments show that the proposed method performs well at recovering the true structure, while real data experiments show that we learn high-quality MRFs from large and diverse datasets.

II. Background and Preliminaries

Throughout this paper, we consider the case of log-linear, pairwise Markov random fields (MRFs). Based on a given MRF structure, the probability of a set of variables $x = \{x_1,..,x_n\}$ is $p_w(x) = \frac{1}{Z(w)} \prod_{c \in C} \phi_c(x; w)$, where, w is a vector containing the model's parameters, $Z(w)$ is a normalizing partition function $Z(w) = \sum_x \prod_{c \in C} \phi_c(x; w)$, and ϕ is a clique potential function $\phi_c(x; w) = \exp\left(\sum_{k \in c} w_k^\top f_k(x)\right)$. The set C contains sets of indices representing cliques and variables, and $f_k(x)$ are feature functions often defined as indicator functions. In the case of pairwise Markov random fields, a clique can refer to either a node or an edge. A pairwise MRF is associated with an undirected graph $G(V, E)$, where V is the set of n nodes corresponding to variables, and E is the set of edges corresponding to pairwise cliques. The factorization for a pairwise MRF is $p_w(x) = \frac{1}{Z(w)} \prod_{i \in V} \phi_i(x; w) \prod_{(i,j) \in E} \phi_{ij}(x; w)$.

A. Parameter Learning Through ℓ_1-Regularized Likelihood

Given a set of data $X = \{x^{(m)}\}_{m=1...N}$, the likelihood is expressed as $l(w) = \prod_{m=1}^{N} p_w(x^{(m)})$. We formulate learning as a minimization of a scaled negative log likelihood $L(w)$, where $L(w) = -\frac{1}{N} \sum_{m=1}^{N} \log p_w(x^{(m)}) = -\frac{1}{N} \sum_{m=1}^{N} \left(w^\top f(x^{(m)})\right) + \log Z(w)$, and w and f correspond to the vectors of w_k and f_k, respectively. Minimizing $L(w)$ is often done using gradients. The gradient of $\log Z(W)$ with respect to the k^{th} feature is the model's expectation of the k^{th}

feature function [8]: $\frac{\partial \log Z}{\partial w_k} = E_w[f_k(x)] = \sum_x p_w(x) f_k(x)$. Hence, the gradient of L with respect to the k^{th} feature is

$$\begin{aligned} \frac{\partial L}{\partial w_k} &= -\frac{1}{N} \sum_{m=1}^{N} f_k(x^{(m)}) + E_w[f_k(x)] \\ &= E_w[f_k(x)] - E_D[f_k(x)] := \delta_k L. \end{aligned} \tag{1}$$

In other words, the feature-wise gradient $\delta_k L$ is the error between the data expectation E_D and the model expectation E_w of $f_k(x)$. The goal of learning can be seen as minimizing that error. To avoid over-fitting and to promote sparsity of the learned weights and of the learned Markov network, classical methods add an ℓ_1 regularization and solve $\min_w \ L(w) + \lambda||w||_1$.

B. Learning Challenges

Learning with the gradient in Eq. (1) is prohibitively expensive for three reasons: (i) The expectation requires computing sufficient statistics for every unary and pairwise feature, which is especially expensive for large datasets. (ii) The classical ℓ_1 formulation ignores the structure of MRFs and treats parameters independently. This treatment dismisses the importance of local consistencies and the power of Markov independence encoded in structure. Finally, (iii) inference of the model expectations $E_w[f_k(x)]$ is generally #P-complete [8]. Existing techniques focus on minimizing the cost of the inference subroutine (iii). Active-set methods allow the learning optimization to compute inference over simpler MRFs. And various approximate inference methods efficiently approximate the expectations, such as loopy belief propagation, its variants, and pseudolikelihood-based methods [3], [6], [9], [11], [12], [14], [17]. Group sparsity methods [5] can be straightforwardly adapted to address the structural coherence issue (ii). Our approach extends these methods of handling issues (ii) and (iii) while also addressing the critical bottleneck of sufficient statistic computation (i) and additional bottlenecks in active-set algorithms.

C. Grafting

Grafting and its variants [10], [15], [18] are active-set learning approaches that alternate between two primary operations: (1) They learn parameters for an active set S of features (e.g., using a sub-gradient method); and (2) they expand the active set by activating one feature using a gradient test based on the Karush-Kuhn-Tucker (KKT) conditions of the ℓ_1-regularized objective. Grafting converges when the activation step does not find any new features to activate or when a predefined maximum number of features $f_{\max}$ is reached. Grafting has a startup bottleneck with an $O(n^2 N s_{\max}^2)$ computational cost to compute the sufficient statistics for each feature, for a system of n variables and a maximum number of states $s_{\max}$. Grafting also involves an exhaustive $O(n^2 s_{\max}^2)$-time search to activate each feature. These bottlenecks translate to poor scalability for large systems and datasets.

Although grafting is fairly suitable for learning MRFs, it does not consider structural information or different clique-memberships of features. Furthermore, while grafting avoids expensive inference, it still requires various quadratic-cost operations: one that scales with the typically large amount of data and one that must be repeated each iteration of the main learning loop.

D. Group Sparsity and Edge Grafting

Using ℓ_1 regularization promotes sparsity uniformly across all parameters. Different regularizers can instead enforce structured sparsity [5]. A similar approach leads to a natural extension of grafting for MRF structure learning. We refer to this variation as *edge grafting*, as it activates edges instead of features. Accordingly, we define the search set F and the active set S respectively as sets of inactive and active edges. To include structural information in the likelihood function, we use group-ℓ_1 regularization. This regularization prefers parameters of each node and each edge to be homogeneous and whole edges to be sparse. We define the group-ℓ_1 negative log likelihood as follows:

$$\mathbb{L}(w) = L(w) + \sum_{g \in G} \lambda d_g ||w_g||_2 + \lambda_2 ||w||_2^2 \ . \tag{2}$$

Each group g contains the weights for either a node or an edge. Consequently, w_g refers to the sub-vector containing all weights related to features of group g. We define d_g as a group scaling parameter and set it to the number of states per clique. As in elastic-net methods [19], we add the ℓ_2-norm to avoid some shortcomings of group-ℓ_1 regularization, such as parameter imbalance and aggressive group selection. We derive a KKT optimality condition for the optimization problem $\min_w \ \mathbb{L}(w)$ as follows:

$$\begin{cases} \frac{||\delta_g L||_2}{d_g} + \lambda_2 ||w_g||_2^2 = 0 & \text{if} \quad ||w_g||_2 \neq 0 \\ \frac{||\delta_g L||_2}{d_g} \leq \lambda & \text{if} \quad ||w_g||_2 = 0 \ , \end{cases} \tag{3}$$

where $\delta_g L$ is the sub-vector constructed using the entries of the gradient vector L corresponding to group g. The above condition requires the probability errors of inactive edges to be smaller than λ (see eq. 5). Hence, we derive an edge-activation test for a given edge $e \in F$,

$$C_2 : s_e > \lambda \ , \tag{4}$$

where s_e is the activation score representing the error between the model $\hat{p}_w(e)$ and the data $p_D(e)$:

$$s_e = \tfrac{1}{d_e} ||\delta_e L||_2 = \tfrac{1}{d_e} ||\hat{p}_w(e) - p_D(e)||_2 \ . \tag{5}$$

By using the group-ℓ_1 regularizer and maintaining an active set of edges, edge grafting runs analogously to grafting but activates parameters for entire edge potentials. Edge-based group-ℓ_1 regularization encourages structural sparsity consistent with Markov notions of variable independence.

III. Best-Choice Edge Grafting

Edge grafting requires computing sufficient statistics for all possible edges and searching over all possible edges at each iteration. Each of these operations costs $O(n^2)$ time. We propose *best-choice edge grafting*, a method that grafts edges

in a streaming fashion using a variation of reservoir sampling [13]. This method activates edges without considering the entire search space. Best-choice edge grafting computes statistics on-demand within the learning loop, and reorganizes the search space by assigning search priorities to edges based on search history and the structure of the partially constructed MRF graph. This reorganization helps the method test edges more likely to be relevant.

We include pseudocode for the discussed algorithms in the appendix.

A. Method Overview

Best-choice edge grafting activates edges without exhaustively computing all sufficient statistics or performing all activation tests. Instead, the approach starts activating edges by computing a small fraction of the edge sufficient statistics. A naive strategy would be to activate the first encountered edge that satisfies C_2, i.e., a "first-hit" approach. However, this approach can introduce many spurious edges. We propose an adaptive approach inspired by *best-choice problems* that uses a control sample of edges satisfying C_2 stored in a limited-memory reservoir R. By maintaining a reservoir of potential edges to activate, we increase the probability of the algorithm activating relevant edges. (See the discussion in our complexity analysis in Sec. III-B.) Moreover, by introducing a reservoir strategy, we directly generalize both the first-hit approach and exhaustive edge grafting, which can be equivalently viewed as using size-one reservoirs and unlimited reservoirs, respectively.

A key subroutine for best-choice edge grafting selects the candidate edge with the highest *priority*. We define a mechanism that combines random sampling with a min-heap priority queue [16] to allow fast priority-based selection that requires no quadratic-time operations under mild assumptions. At all times, the edges are grouped into *prioritized edges* and *unseen edges*. A prioritized edge is any edge whose priority is adjusted by the algorithm, and all prioritized edges are stored in the priority queue. Unseen edges are implicitly assigned a default priority score ρ_0, but they are not explicitly stored. To select the edge with maximum priority, the algorithm first examines the maximum prioritized edge from the priority queue and compares it to ρ_0. If the edge extracted from the priority queue has lower priority than ρ_0 or if the priority queue is empty, we repeatedly sample random edges until we sample one that is not already prioritized. As we show in Sec. III-B, the probability of sampling an already prioritized edge approaches zero asymptotically as the number of variables grows.

The learning loop selects edges in order of priority score and places edges that pass the activation test into the reservoir, which retains the edges that most violate the optimality condition. Once enough edges have been seen, the algorithm selects edges to activate from the reservoir and then performs inference to update the model expectations. The priority queue and reservoir are then updated based on the newly estimated MRF structure (see Secs. III-A1 and III-A2), and the next iteration begins.

There are no quadratic-time operations within the main loop. Though there may in the worst case be $O(n^2)$ elements in the priority queue, each insertion, removal, or update operation costs time logarithmic in the number of stored elements, which is $O\left(\log\left(n^2\right)\right) = O(2\log(n)) = O(\log(n))$. This efficiency enables prioritization and management of the large search space.

1) Reservoir management: Best-choice edge grafting strategically manages the reservoir to retain edges likely to be relevant. Let $A \subseteq F$ be the unknown set of all edges that currently satisfy C_2. We construct a control sample $R \subseteq A$ from which we activate edges. Edges in the search set F are initially assigned equal, default priorities ρ_0. These priorities will be adjusted based on informed heuristics. The method iterates through F by extracting the highest priority edge, generating a prioritized stream of edges to test. If a tested edge satisfies C_2, it is added to R. Otherwise, it is ignored.

When a maximum number of edge tests $t_{\max}$ is reached, we start activating edges. To maintain a high-quality reservoir, edge activation only starts after we fill R to capacity in the first edge-activation iteration. Furthermore, if R reaches its capacity before $t_{\max}$ is reached, we replace the minimum-scoring edge in R with the newly tested edge whenever it has a higher score. We use a simple strategy to activate edges in the reservoir with high activation scores where we consider edges that have at least an above-average score. We compute the average activation score $\mu = \frac{1}{|R|}\sum_{e\in R} s_e$, and then we define a confidence interval for choosing edges that are likely to be relevant as $I_\alpha = \left[\mu + \alpha(\max_{e\in R} s_e - \mu), \max_{e\in R} s_e\right]$, where $\alpha \in [0,1]$. The algorithm considers activating edges with activation scores no less than $\tau_\alpha = (1-\alpha)\mu + \alpha \max_{e\in R} s_e$. After deriving τ_α, edges are activated in a decreasing order with respect to their scores. To avoid redundant edges and to promote scale-free structure, we only activate edges that are not adjacent. Note that when $\alpha = 1$, we only select the maximum-scoring edge. Reducing α increases the number of edges to be activated at a certain step but can also result in adding spurious edges. See the pseudocode in the appendix.

After each optimization step, edge gradients change. Therefore, scores of reservoir edges are updated, and edges that no longer satisfy C_2 are dropped from the reservoir.

2) Search space reorganization: Best-choice edge grafting performs search space reorganization by assigning and updating the search priority of edges in the priority queue. The aim of this reorganization is to increase the quality of the received stream of edges and the reservoir. We leverage search history and structural information.

Search history Each activation iteration, an edge with a small activation score is unlikely to satisfy C_2 in the future and is placed further towards the tail of the priority queue. We define an edge-violation offset $v_e = 1 - \frac{s_e}{\lambda}$. When the activation tests fail or edges are dropped from R, the low-score edges are not immediately returned to the priority queue but instead are "frozen" and placed—along with their violation offsets—in a separate container L. When the search priority queue is emptied, we refill it by re-injecting frozen edges from

TABLE I: Time complexity of different methods. The second column measures how much computation is necessary at startup. Edge grafting requires the sufficient statistics of all possible edges to do any activation, while best-choice edge grafting only computes statistics as needed. The activation-step cost is the cost to decide which edge to activate next. For reference, we include the approximate cost to approximate the expectations and gradient, which is typically linear in the size of the current active set (e.g., belief propagation or pseudolikelihood).

Algorithm	Suff. stats. at j^{th} edge activation	Activation step	Inference
Edge grafting	$O\big(n^2 N s_{\max}^2\big)$	$O\big(n^2 s_{\max}^2\big)$	$\sim O(j+n)$
Best-choice edge grafting	$O\big((n+jt_{\max})N s_{\max}^2\big)$	$O\big(t_{\max} s_{\max}^2\big)$	$\sim O(j+n)$

L with their respective violation offsets as their new priorities.

Partial structure information As the active set grows, so does the underlying MRF graph G. The resulting partial structure contains rich information about dependencies between variables. We rely on the hypothesis that graphs of real networks have a scale-free structure [1]. We promote such structure in the learned MRF graph by encouraging testing of edges incident to central nodes. We start by measuring node centrality on the partially constructed MRF graph G to detect hub nodes. A degree-based node centrality c_i for a node i is the fraction of all possible neighbors N_i it is connected to: $c_i = |N_i|/(|V|-1)$. We then use a centrality threshold $\hat{c}$ to identify the set of hubs $H = \{i \in V \text{ such that } c_i > \hat{c}\}$. Finally, we prioritize all edges incident to nodes in H by decrementing their priorities by 1. The total cost of updating the min-heap structure is $O(|H| n \log(n))$, where $O(\log(n))$ is the cost of updating the priority of an edge. Reorganizing the priority queue pushes edges more likely to be relevant to the front of the queue. This induces a higher-quality reservoir and promotes the activation of higher-quality edges at each activation iteration.

B. Complexity Analysis

The selection of a candidate edge, either from the priority queue or from random sampling, is at most an $O(\log n)$ cost under the assumption that we only observe $O(n)$ candidate edges through learning. For each edge selection, we must examine the highest priority entry in the priority queue, which costs $O(\log n)$. Then if that priority is higher than the default priority, the selection task is complete. If not, we must randomly sample an unseen edge. Randomly sampling any edge costs $O(1)$ time, and checking that the edge has not yet been seen requires another $O(1)$ set-membership check. If the edge has been seen, then we need additional samples until we sample an unseen edge. Fortunately, if the number of seen edges is less than some constant factor of n, i.e., βn, then the probability of sampling a previously seen edge is $\beta n / \binom{n}{2}$, or $2\beta/(n-1)$, which asymptotically approaches zero.

If it takes the algorithm r edge tests to fill the reservoir in one pass, then best-choice edge grafting performs $\tilde{O}(r N s_{\max}^2)$ operations to compute the sufficient statistics necessary to fill the reservoir and activate the first edges. (We use $\tilde{O}$ notation, omitting the logarithmic costs of using the priority queue.) To activate the j^{th} edge, the algorithm needs to perform at most $\tilde{O}\big((r+jt_{\max})N s_{\max}^2\big)$ operations, where $t_{\max}$ is the allowed number of edge tests between two activation steps.

In most cases, we can assume that it takes the algorithm $r = O(|R|)$ tests to fill the reservoir R. Furthermore, to construct a relevant reservoir, we set $|R| = O(n)$. We also set $t_{\max} \ll n$, so to activate the j^{th} edge, the algorithm needs to compute at most $O\big((n+jt_{\max})N s_{\max}^2\big)$ sufficient statistics tables. In the case of edge grafting, the algorithm needs to compute $O(n^2 N s_{\max}^2)$ sufficient statistics to start grafting the first edge. Since $(n+jt_{\max}) \ll n^2$, our method provides a drastic speedup, as we circumvent the quadratic term in n and replace it with a linear term. Finally, between each activation step, we must compute approximate expectations to obtain the gradient. Approximate inference techniques typically take time linear in the number of active edges. Table I summarizes the different time complexity for the discussed algorithms.

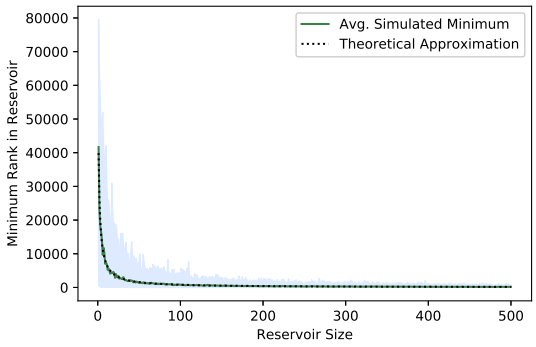


Fig. 1: Simulated edge ranks using the reservoir. The curves show that the average and expected ranks are nearly identical, and the shaded region indicates the full range of ranks obtained during 100 random trials.

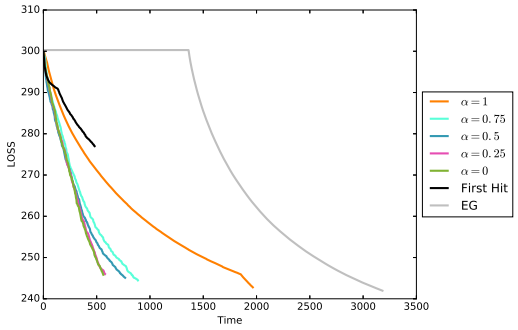
IV. Experimental Evaluation

Our experiments include simulation of reservoir-based search and evaluation of the speed and quality of learning from synthetic and real data. We use datasets of different variable dimensions and dataset sizes. In the synthetic setting, we simulate MRFs and generate data using a Gibbs sampler. In the real setting, we use two datasets. We use exhaustive edge grafting (labeled "EG" in figures) as our primary baseline. To show the benefits of the reservoir, we also include best-choice edge grafting with no reservoir ("first hit"), which activates the first edge that passes the edge-activation condition.

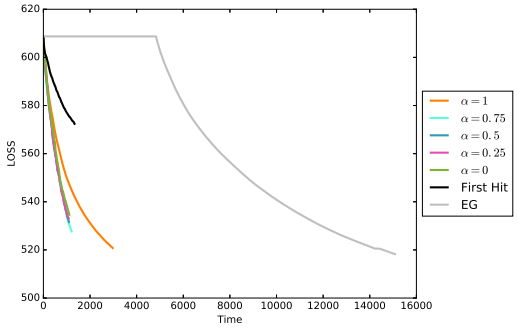
A. Benefits of the Reservoir

We analyze the increase in edge quality that the reservoir provides. The reservoir management protocol mimics a reservoir of randomly selected entries from the full population. Asymptotically, the relative percentile of any randomly chosen edge is equivalent to a uniform random variable in the range $[0, 1]$. The rank of the best edge in a size-$|R|$ reservoir is thus

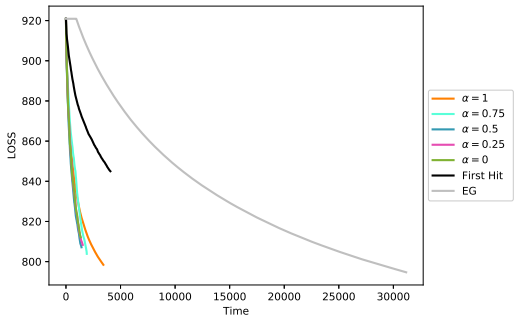
analogous to the minimum of $|R|$ uniform draws. It is well known that the expected value of this minimum is $1/(|R|+1)$. We simulate the behavior in finite settings, sampling $|R|$ ranks from the list of all possible numbers from 1 to $\binom{n}{2}$ and taking the minimum. We then plot the average minimum rank over 100 trials, using $n = 400$ with values of $|R|$ from 1 to 500. Fig. 1 plots the average ranks over the trials. The results suggest that a small reservoir provides significant gains over using first hit ($|R| = 1$). Using an unlimited reservoir—i.e., grafting—only provides negligible gains over a small reservoir. Since grafting is equivalent to a reservoir of size $\binom{400}{2}$, or 79,800, the benefits of using a reservoir are evident.



(a) 200 nodes and 600 edges



(b) 400 nodes and 1,200 edges



(c) 600 nodes and 1,800 edges

Fig. 2: Loss vs. time (seconds) for varying MRF sizes with $O(n)$ edges.

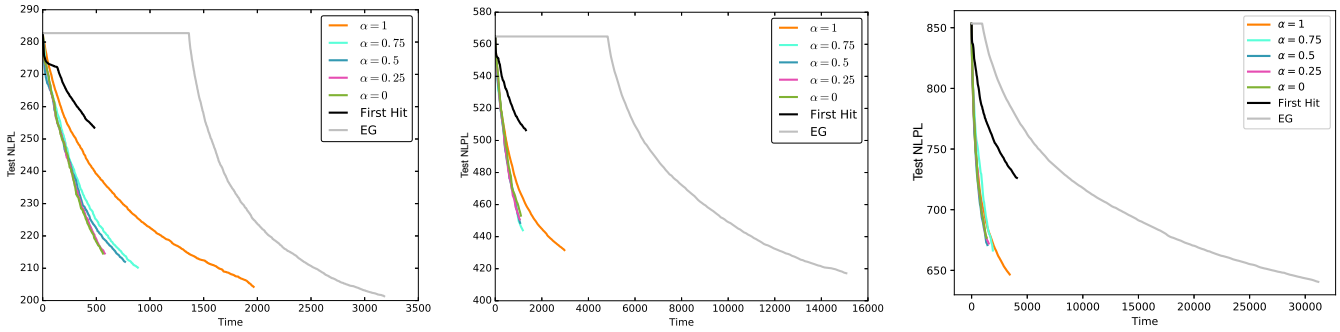
B. Synthetic Data

We construct random, scale-free structured MRFs with variables having five states each. We generate preferential-attachment graphs [1] of size 200, 400, and 600, with 498,500, 1,997,000, and 4,495,500 parameters, respectively. We then use Gibbs sampling to generate a set of 20,000 data points, which we randomly split into 19,000 training and 1,000 held-out test data points. We use grid search to tune the learning parameters. We fix $|R| = n$, and $t_{\max} \ll n$. See the appendix for more details on the experimental setup.

Faster convergence We first investigate the behavior of different methods when executed until no violating edge remains in the search space. We measure the different methods' convergence speeds and confirm that they lead to similar solutions. Figure 5 shows that all methods reach a similar solution, but best-choice edge grafting has a faster convergence rate. The first-hit baseline starts with a fast descent rate, but its activation of lower-violation edges eventually causes it to converge slower than edge grafting. Using a reservoir maintains a steep descent until convergence. These experiments also confirm that edge grafting suffers a major cost of computing sufficient statistics, causing it to start optimization after best-choice edge grafting has nearly converged.

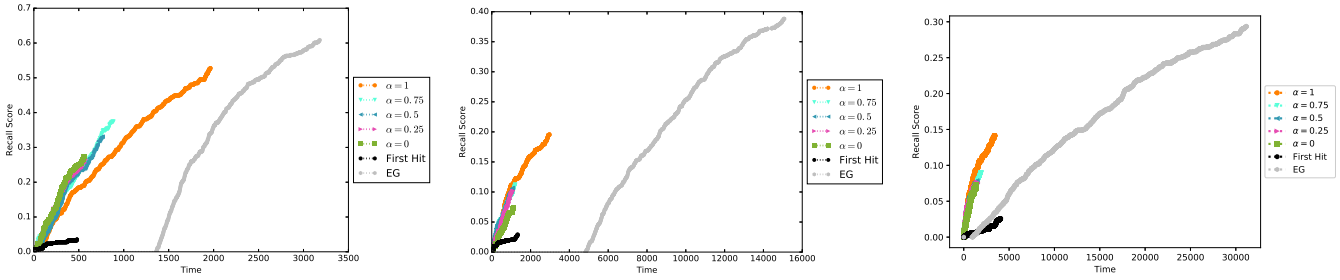
Controllable tradeoff between learning speed and quality In subsequent experiments, we fix the maximum number of activated edges to $3n$, stopping early when each algorithm exhausts this limit. We first measure the objective value during learning and plot it over running time in Fig. 2. We observe similar trends: Best-choice edge grafting provides significant speedups over exhaustive and first-hit edge grafting. Higher α values enable better quality but result in a slower optimization. Experiments on held-out testing data (Fig. 3) show that there is a positive correlation between the learning objective and the test negative log pseudo-likelihood (NLPL), which confirms that the learned models do not over-fit even with small values of α. The first-hit baseline reaches the edge limit faster, but its lower-quality edges cause slower convergence (see Fig. 5) and a lower quality model (Figs. 2 and 4).

Faster edge activation For increasing amounts of variables, the learning gap between best-choice edge grafting and exhaustive edge grafting increases, which demonstrates the better scalability of the best-choice algorithm. In the smaller graphs, exhaustive edge grafting has the advantage that its greedy search for the worst-violating edge enables large improvements in the objective (and pseudo-likelihood). However, in the larger graphs, even though edge grafting precomputes sufficient statistics, the remaining $O(n^2)$ cost of the greedy search causes its objective to descend at a slower rate than the best choice variants, which avoid this exhaustive search. This high cost is especially evident in the 600-variable problems (e.g., Fig. 2c), where it takes nearly ten times as long for edge grafting to add the desired number of edges as the reservoir-based best-choice approaches.



(a) 200 nodes and 600 edges (b) 400 nodes and 1,200 edges (c) 600 nodes and 1,800 edges

Fig. 3: Negative log pseudo-likelihood (NLPL) vs. time (seconds) for varying MRFs sizes with $O(n)$ edges.



(a) 200 nodes and 600 edges (b) 400 nodes and 1,200 edges (c) 600 nodes and 1,800 edges

Fig. 4: Recall vs. time (seconds) for varying MRFs sizes with $O(n)$ edges.

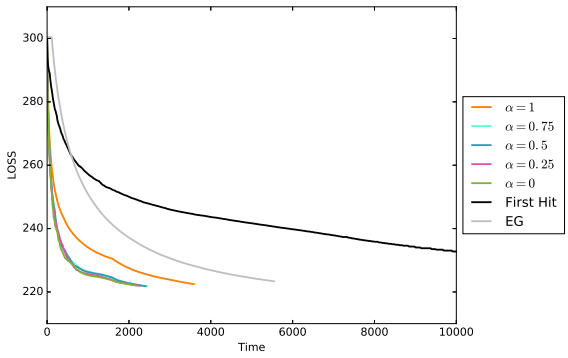


Fig. 5: Objective values vs. seconds during full convergence of all methods (200 nodes). The first-hit method takes around 30,000 seconds to converge, so we truncate the horizontal axis for a better view of the other methods.

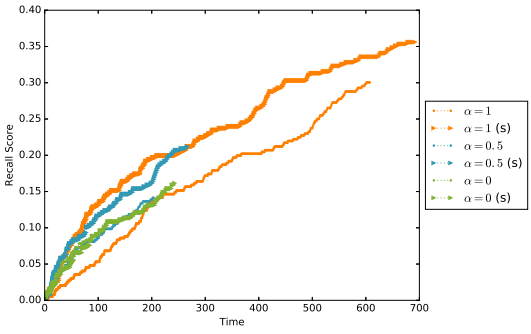


Fig. 6: Role of structure heuristics in improving the quality of the learned MRF over training time (seconds). The proposed heuristics (labeled "s") produce higher recall for an MRF of 200 nodes and 600 edges.

Structure learning quality Fig. 4 plots the recall of true edges over time until the maximum number of added edges is met. Increasing values of α lead to better recall for different MRF sizes. However, this comes at the cost of a more expensive learning optimization. In fact, for smaller values of α, best-choice edge grafting tends to activate more edges but with lower quality, which introduces a greater number of false-positive edges. This early stopping setting forces the number of correctly activated edges to be lower for lower values of α. However, if the goal of learning the MRF structure is to produce a good generative model, then lower α values speed up learning with only a small loss in generative quality.

Effectiveness of structure heuristics To evaluate the effectiveness of structure heuristics, Fig. 6 plots the edge recall—for the same limited number of activated edges—with and without the structure-based priority queue reorganization. The significant gap between the recall curves shows that the heuristics produce a higher-quality edge stream from the priority queue, resulting in a higher-quality reservoir and more relevant edges to activate.

C. Real Data

We use the Jester joke ratings [4] and Yummly recipes [7] datasets. Jester is a joke ratings dataset containing 100 jokes and 73,421 user ratings. We use jokes as variables and user ratings as instances. We map the ratings interval from a continuous interval in $[0, 20]$ to a discrete interval from 1 to 5, leading to 124,250 parameters. We sample 10,000 recipes

from the Yummly data, using the 489 most common ingredients (which occur in at least 150 recipes) as binary variables and each recipe as a data instance, leading to $478{,}242$ parameters. We randomly sample 10% of the initial data and use it as held-out testing data. Since true edges for the underlying model of real data are not available, we measure learning quality using negative log pseudo-likelihood.

Figure 7 shows a positive correlation between the minimization of the learning objective and the testing negative log pseudo-likelihood. The results are similar to those from the synthetic experiments, where best-choice edge grafting converges faster than edge grafting, and smaller values of α result in a faster convergence. The different datasets illustrate the higher scalability of best-choice edge grafting, as its advantage in convergence time over edge grafting increases with dimensionality.

V. Conclusion

We presented best-choice edge grafting, a method based on a group-ℓ_1 formulation and reservoir sampling. This incremental method activates edges instead of features and uses a reservoir to approximate greedy activation. Theoretical analysis of the iteration complexity shows that the method provides higher scalability than exhaustive edge grafting by avoiding its major bottlenecks. Experiments on synthetic and real data demonstrate that best-choice edge grafting yields faster convergence on multiple datasets, while also achieving structure recovery and predictive ability similar to the more costly exhaustive edge grafting.

References

[1] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

[2] G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40. ACM, 2007.

[3] J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 64(3):616–618, 1977.

[4] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[5] J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. *Journal of Machine Learning Research*, 12(Nov):3371–3412, 2011.

[6] A. T. Ihler, W. F. John III, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6(May):905–936, 2005.

[7] Kaggle. What's cooking? https://www.kaggle.com/c/whats-cooking.

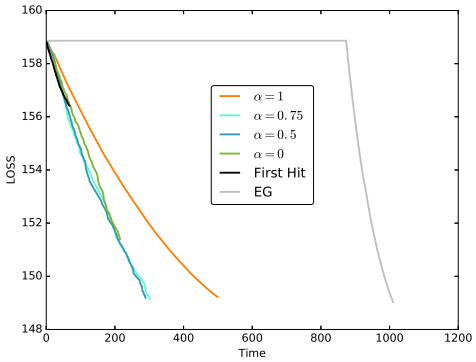
[8] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[9] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

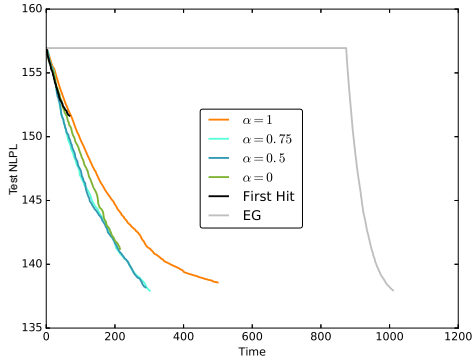
[10] S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using l1-regularization. In *Advances in Neural Information Processing Systems*, pages 817–824, 2006.

[11] T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms: a unifying view. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 393–401. AUAI Press, 2009.

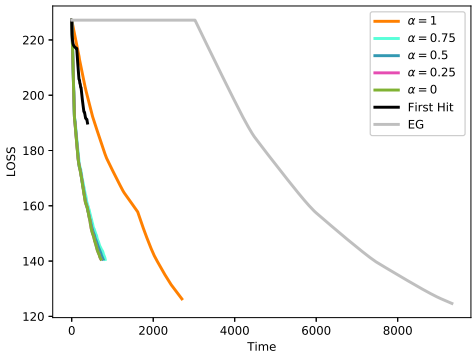
[12] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.



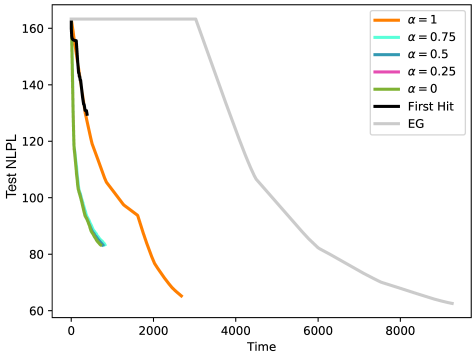
(a) Jester Objective



(b) Jester NLPL



(c) Yummly Objective



(d) Yummly NLPL

Fig. 7: Learning objective (a, c) and negative log pseudo-likelihood (b, d) vs. seconds for Jester and Yummly.

[13] F. Olken and D. Rotem. Random sampling from databases: a survey. *Statistics and Computing*, 5(1):25–42, 1995.
[14] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2014.
[15] S. Perkins, K. Lacker, and J. Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.
[16] P. van Emde Boas, R. Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. *Mathematical Systems Theory*, 10(1):99–127, 1976.
[17] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.
[18] J. Zhu, N. Lao, and E. P. Xing. Grafting-light: fast, incremental feature selection and structure learning of Markov random fields. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 303–312. ACM, 2010.
[19] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

APPENDIX

We summarize edge grafting, as discussed in Section 3.1, in Algorithm 1.

Figure 8 presents a high-level description of the best-choice edge-activation mechanism: From left to right, the first structure is the priority queue (pq) initialized with the set of all possible edges. The next diamond-shaped box represents the activation test C_2, after which an edge is either added to the reservoir (R) or to the frozen edge container L. The $t_{\max}$ box represents when the maximum number of edge tests is reached, after which edges are activated. The gray dashed line on the right ($R_{\min}$) indicates the injection of the minimum scoring edge in R into L, when R is full. The gray dashed line on the left (refill(L)) indicates refilling the priority queue with the frozen edges once it is emptied.

The priority-queue reorganization subroutine is summarized in Algorithm 2, and the activation mechanism is presented in Algorithm 3. These form the underlying components to construct the main best-choice edge grafting framework in Algorithm 4.

Our experiments show that there is only a small practical benefit of using random sampling for default-priority edges. For the data sizes in our experiments, the $O(n^2)$ cost to instead initialize the full priority queue was negligible, since it is a one-time operation. The quadratic cost only becomes a practical bottleneck when it is repeated or compounded by the data size N, as is the case for edge grafting. Thus, our experiments report running times where our best-choice edge-grafting methods take an extra half second or less for the one-time initialization of the full priority queue.

We generate MRFs with different sizes and variables cardinality five. In particular, we generate structures of size 200, 400, and 600, with 498,500, 1,997,000, and 4,495,500 parameters, respectively.

We first construct random, scale-free structured graphs. To do so, we use a preferential-attachment model where we grow a graph by attaching new nodes, each with two edges that are preferentially attached to existing nodes with high node centrality. The algorithm produces $(2n-4)$ edges. For each node and each created edge, we sample their corresponding parameters from a normal distribution with a mean equal to 100 and standard deviations $\sigma_v = 0.5$ (for nodes) and $\sigma_e = 1$ (for edges). This setting puts more emphasis on the edges (pairwise relationships between nodes) and helps avoid making the model overly dependent on only the unary potentials.

To generate data, we use a Gibbs sampler to generate a set of $20{,}000$ data points, which we randomly split as $19{,}000$ training data points and $1{,}000$ held-out testing data points.

Parameter tuning: We use a grid search to detect the best combination of λ and λ_2. We limit our search range to the set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ for λ and the set $\{0.5, 0.75, 1, 1.5\}$ for λ_2. For each case, we choose the pair (λ, λ_2) that produces the best recall and test NLPL for the baseline, i.e., edge grafting. It is worth noting that we did not notice a high sensitivity of the methods for different values of λ_2, whereas smaller values of λ produce spurious edges for different methods.

Algorithm 1 Edge Grafting

Define *EdgeNum* as the maximum allowed number of edges to graft
Initialize $\mathcal{E} = \emptyset$ and $\mathcal{F} = \{\text{set of all possible edges}\}$
Compute sufficient statistics of e $\forall e \in \mathcal{F}$ # cost: $O(n^2 N s_{\max}^2)$
AddedEges $= 0$
Continue $=$ True
while *EdgeNum* $>$ *AddedEges* and *Continue* **do**
  Compute score s_e $\forall e \in \mathcal{F}$ # cost: $O(n^2 s_{\max}^2)$
  $e^* = \arg\max_{s \in \mathcal{E}} s_e$. # cost: $O(n^2)$
  if $s_{e^*} > \lambda$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{e^*\}$; $\mathcal{F} \leftarrow \mathcal{F} \backslash \{e^*\}$
    AddedEges $\leftarrow$ *AddedEges* $+1$
    Optimization weights of edges in the active set $\mathcal{E}$
  else
    Continue $\leftarrow$ False
  end if
end while

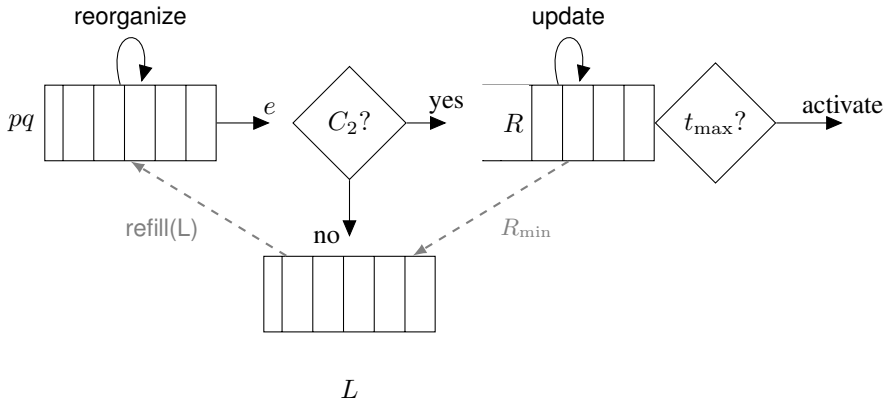


Fig. 8: High-level operational scheme of the edge activation mechanism.

Algorithm 2 Reorganize PQ

Compute centrality measures over partially constructed MRF graph G
Construct the hub set $\mathcal{H}$ # cost: $O(n)$
for $h \in \mathcal{H}$ **do**
  for $n \in \mathcal{V}$ **do**
    $pq[(h,n)] \leftarrow pq[(h,n)] - 1$ # total loop cost: $O(|\mathcal{H}| n \log(n))$
  end for
end for

Algorithm 3 Activation Test

Reorganize pq using Algorithm 2
$\mathcal{E}^* = \emptyset$
$t = 0$
repeat
  if pq is empty **then**
    if R is empty **then**
      Break
    end if
    pq = Refill(L)
  end if
  Get edge e with highest priority from pq (or by random sampling if pq is empty)
  if Sufficient statistics of e not already computed **then**
    Compute sufficient statistics of e # cost: $O(Ns_{\max}^2)$
  end if
  Compute score s_e. # cost: $O(s_{\max}^2)$
  if $s_e > \lambda$ and R not full **then**
    Add e to R # add e if capacity not reached
  else if $s_e > \lambda$ and (R is full and $R_{\min} < s_e$) **then**
    Replace $R_{\min}$ by e. # $R_{\min}$: minimum scroing edge in R
    Place $R_{\min}$ in L.
  else
    Place e in L
  end if
until $t = t_{\max}$
Compute τ
for $e \in R$ s.t $s_e \geq \tau$ **do**
  if e not adjacent to edges in $\mathcal{E}^*$ **then**
    $\mathcal{E} \leftarrow \mathcal{E}^* \cup \{e\}$
  end if
end for

Algorithm 4 Best-Choice Edge Grafting

Define *EdgeNum* as the maximum allowed number of edges to graft
Initialize $\mathcal{E} = \emptyset$ and $\mathcal{F} = \{$set of all possible edges$\}$
Initialize empty *pq*
AddedEges = 0
Fill R to capacity
while *EdgeNum* > *AddedEges* **do**
  Get set $\mathcal{E}^*$ of edges to activate using Algorithm 3
  if $\mathcal{E}^*$ is empty **then**
    Break
  end if
  for e^* in $\mathcal{E}^*$ **do**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{e^*\}$; $\mathcal{F} \leftarrow \mathcal{F} \backslash \{e^*\}$
    AddedEges $\leftarrow$ *AddedEges* $+1$
  end for
  Perform an optimization over active parameters.
end while