Algorithm 20.1 Greedy score-based structure search algorithm for log-linear models

```
     Procedure Greedy-MN-Structure-Search (
       $\Omega$,   // All possible features
       $\mathcal{F}_0$,   // initial set of features
       $\mathrm{score}(\cdot : \mathcal{D})$,   // Score
     )
1      $\mathcal{F}' \leftarrow \mathcal{F}_0$   // New feature set
2      $\boldsymbol{\theta} \leftarrow \mathbf{0}$
3      do
4        $\mathcal{F} \leftarrow \mathcal{F}'$
5        $\boldsymbol{\theta} \leftarrow$ Parameter-Optimize($\mathcal{F}, \boldsymbol{\theta}, \mathrm{score}(\cdot : \mathcal{D})$)
6          // Find parameters that optimize the score objective, relative to
             current feature set, initializing from the current parameters
7        for each $f_k \in \mathcal{F}$ such that $\theta_k = 0$
8          $\mathcal{F} \leftarrow \mathcal{F} - f_k$
9            // Remove inactive features
10       for each operator $o$ applicable to $\mathcal{F}$
11         Let $\hat{\Delta}_o$ be the approximate improvement for $o$
12       Choose some subset $\mathcal{O}$ of operators based on $\hat{\Delta}$
13       $\mathcal{F}' \leftarrow \mathcal{O}(\mathcal{F})$   // Apply selected operators to $\mathcal{F}$
14       while termination condition not reached
15     return $(\mathcal{F}, \boldsymbol{\theta})$
```

Bayesian networks do not apply here. In the case of Bayesian networks, evaluating the score of a candidate structure is a very easy task, which can be executed in closed form, at very low computation cost. Moreover, the Bayesian network score satisfies an important property: it *decomposes* according to the structure of the network. As we discussed, this property has two major implications. First, a local modification to the structure involves changing only a single term in the score (proposition 18.5); second, the change in score incurred by a particular change (for example, adding an edge) remains unchanged after modifications to other parts of the network (proposition 18.6). These properties allowed us to design efficient search procedure that does not need to reevaluate all possible candidates after every step, and that can cache intermediate computations to evaluate candidates in the search space quickly.

score decomposability

Unfortunately, none of these properties hold for Markov networks. For concreteness, consider the likelihood score, which is comparable across both network classes. First, as we discussed, even computing the likelihood of a fully specified model — structure as well as parameters — requires that we run inference for every instance in our training set. Second, to score a structure, we need to estimate the parameters for it, a problem for which there is no closed-form solution. Finally, none of the decomposition properties hold in the case of undirected models. By adding a new feature (or a set of features, for example, a factor), we change the weight ($\sum_i \theta_i f_i(\xi)$) associated with different instances. This change can be decomposed, since it is a linear function of the different features. However, this change also affects the partition function, and, as we saw in the context of parameter estimation, the partition function couples the effects of changes in