5.1 Stochastic methods: simulated annealing

Let a posteriori Markov random field model involve only pairwise interactions, i.e. the posterior Gibbs probability distribution is

$$P(y|x) = \frac{1}{Z} \exp \left( \frac{1}{2T} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} V(y_i, y_j) \right)$$

with symmetric potentials, $V(q,s) = V(s,q)$; $s,q \in \mathbb{Y}$, having a common inverse factor, $T$. The normalizing factor $Z$ is called the partition function. Due to the very large number $|\mathbb{Y}|^n$ of possible samples (configurations) $y \in \mathbb{Y}^n$, the partition function cannot be computed, excepting a few simple cases. By physical analogies, the negative exponent,

$$E(y) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} V(y_i, y_j)$$

and the factor $T$ are called energy and temperature, respectively.

The MAP solution $y^* = \arg \max P(y|x)$ can be found in principle by simulated annealing (SA), i.e. by generating a Markov chain of samples with stochastic relaxation (a so-called Markov chain Monte Carlo, or MCMC) and simultaneously modifying the model.
in such a way as to eventually come to and stay in a close vicinity of the global maximum. The MCMC generation changes a single component $y_i$ each time in accord with how the new value effects the energy.

**Metropolis MCMC sampler:** Let $\Delta_i = E(y_1, \ldots, \hat{y}_i, \ldots, y_n) - E(y_1, \ldots, y_i, \ldots, y_n)$ be the difference between the energies for the potential new value $\hat{y}_i$, randomly and equiprobably sampled from $\mathbb{Y}$, and for the current old value $y_i$, provided that all other values are just the same. Then if $\Delta_i \geq 0$, the new value is accepted, but otherwise ($\Delta_i < 0$) it is accepted with a probability equal to $\exp(\Delta_i / T)$. The latter occasional acceptance of less energetically favorable states allows for jumping out of local energy minima.

To compute the energy difference, only components $j$ that interact with $i$, i.e., are its neighbors $j \in \mathcal{N}_i$, should be taken into account, because all other components contribute the same total amount to the full energy in both the cases.

**Gibbs MCMC sampler:** The new value $\hat{y}_i \in \mathbb{Y}$ is accepted in accord with the conditional probabilities of these values given the same values of all other components:

$$
p(\hat{y}_i = q | y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n) = \frac{\exp(E(y_1, \ldots, \hat{y}_i = q, \ldots, y_n))}{\sum_{s \in \mathbb{Y}} \exp(E(y_1, \ldots, \hat{y}_i = s, \ldots, y_n))} \equiv \frac{\exp(E(\hat{y}_i = q, \{y_j : j = \mathcal{N}_i\}))}{\sum_{s \in \mathbb{Y}} \exp(E(\hat{y}_i = s, \{y_j : j = \mathcal{N}_i\}))}; \quad q \in \mathbb{Y}
$$

**Stochastic and deterministic SA** The components $y_i$ are visited in a random sequence (thus the algorithm is sometimes called sequential SA). The basic idea of SA is that the temperature factor is gradually changing during the process, so that each iteration $k$ has its own temperature $T(k)$ during visiting every component $y_i$; $i = 1, \ldots, n$. The function $t(k)$ of the iteration index $k$ is called the cooling schedule, or the annealing schedule. At the very beginning, $T(1)$, the temperature is to be sufficiently high, so that all configurations $y \in \mathbb{Y}^n$ are roughly equiprobable. Then the temperature decreases gradually and slow enough to move to any part of the sample space $\mathbb{Y}^n$ before being trapped in a local minimum.

Unfortunately, theoretically justified annealing schedules are too slow, and known estimates of the number of iterations to reach the global minimum of energy with high
probability are of order of full exhaustion, \( \mathbb{Y}^n \). In practice in many cases it is more efficient than exhaustive search but is still too slow. A greedy deterministic approximation of the SA with Gibbs sampler, proposed by J. Besag and called the Iterated Conditional Mode (ICM) algorithm, accepts at each step the conditionally most probable new value 
\[
\hat{y}_i = \arg\max_{q \in \mathbb{Y}} \{ p(y_i = q | y_j : j \in \mathcal{N}_i) \}
\]
But this approach can reach only a local minimum closest to the starting configuration.

Partially the slow convergence of stochastic SA is due to the discrete search through the space of all configurations \( \mathbb{Y}^n \) along each single edge. Such a search misses gradient information which might be provided by analog state values in the continuous "interior" of this space.

Deterministic SA is a faster algorithm that allows each discrete component \( y_i \) to take analog values during search; at the end of the search the values are forced to be back in \( \mathbb{Y} \). The analog value is typically the conditional expected value of the discrete \( y_i \) at temperature \( T \):

\[
\hat{y}_i = \frac{\sum_{q \in \mathbb{Y}} q \exp(E(y_i = q, \{ y_j : j \in \mathcal{N}_i \}))}{\sum_{q \in \mathbb{Y}} \exp(E(y_i = q, \{ y_j : j \in \mathcal{N}_i \}))}
\]

This search method is called mean-field annealing. It is deterministic because in principle the simultaneous equations governing the \( y_i \) as the temperature is lowered can be solved deterministically.

In practice, deterministic and stochastic annealing give similar solutions, but in the large problems deterministic annealing is faster by two-three orders of magnitude.

### 5.2 Exact min cut / max flow optimization

In most of computer vision applications, the Bayesian MAP classification or estimation can be formulated as combinatorial optimization on graphs specifying binary relationships restricting the solution. Typically, the graph structure specifies pixel neighborhoods, i.e. subsets of pixels where signals are mutually dependent. Weights of nodes and edges characterize possible signal values and/or specific labels. Desired classification maps, estimated images, or other 2D arrays of estimated pixel-wise properties are probabilistically described in terms of (gibbs) energies accumulating particular weights of nodes / edges. Generally, under dense 2D neighborhoods, an optimal minimum-energy solution is an
NP-complete problem. But it needs to be at least approximately solved or reduced to an exactly solvable particular case.

### 5.2.1 Maximum flow problem

Let \( G = [N; E] \) denote a directed (linear) graph with a collection of nodes (vertices, points) \( N = \{x, y, \} \) and a subset \( E \) of ordered pairs (edges, arcs) \( (x, y) \) of elements from \( N \). A chain is a sequence of nodes \( x_1, x_2, \ldots \) such that \( (x_i, x_{i+1}) \in N \). A path is a sequence \( x_1, x_2, \ldots \) such that \( (x_i, x_{i+1}) \in N \) or \( (x_{i+1}, x_i) \in N \). Let \( A(x) = \{y | y \in N; (x, y) \in E \} \) and \( B(x) = \{y | y \in N; (y, x) \in E \} \) be subsets of the nodes after \( x \) and before \( x \), respectively.

A network is a graph with two special nodes \( s \) and \( t \) called source and sink, respectively, and with a non-negative capacity \( c(x, y) \geq 0 \) assigned to every edge \( (x, y) \in E \). The function \( c : E \to \mathbb{R}^{\geq 0} \) is called the capacity function.

A static flow of value \( v \) from \( s \) to \( t \) in a network \( G = [N; E] \) with a capacity function \( c \) is a function \( f : E \to \mathbb{R}^{\geq 0} \) that satisfies the linear constraints:

1. The flow through every edge does not exceed the edge capacity:
   \[
   \forall_{(x,y) \in E} \quad f(x, y) \leq c(x, y) \tag{5.1}
   \]

2. Every node other than the source and the sink has equal total out- and in-flows:
   \[
   \sum_{y \in A(x)} f(x, y) - \sum_{y \in B(x)} f(x, y) = \begin{cases} 
   v & \text{if } x = s \\
   0 & \text{if } x \in N \setminus \{s, t\} \\
   -v & \text{if } x = t 
   \end{cases} \tag{5.2}
   \]

A simple network flow example is shown in Fig. 5.1. The static maximum flow problem is to maximize the variable \( v \) subject to the above flow constraints.

**Simplifying notation**  Let \( (X, Y); X \subset N, Y \subset N \) denote a set of all edges from nodes \( x \in X \) to nodes \( y \in X \). For any function \( g : E \to \mathbb{R} \), let \( g(X, Y) \) denote the sum of values of the function over the set of edges:

\[
\sum_{(x, y) \in (X, Y)} g(x, y)
\]
Figure 5.1: An example of the flow of value 3 assuming \( c(x,y) \geq f(x,y) \) for all the edges.

It is easily shown that

\[
\begin{align*}
  g(X,Y \cup Z) &= g(X,Y) + g(X,Z) - g(X,Y \cap Z) \\
  g(Y \cup Z,X) &= g(Y,X) + g(Z,X) - g(Y \cap Z,X)
\end{align*}
\]

so that for disjoint sets \( Y \) and \( Z \)

\[
g(X, Y \cup Z) = g(X,Y) + g(X,Z) \quad \text{and} \quad g(Y \cup Z,X) = g(Y,X) + g(Z,X)
\]

In particular, \((B(x),x) = (N,x)\) and \((x,A(x)) = (x,N)\) so that

\[
g(N,X) = \sum_{x \in X} g(N,x) = \sum_{x \in X} g(B,x) \quad \text{and} \quad g(X,N) = \sum_{x \in X} g(x,N) = \sum_{x \in X} g(x,A)
\]

Cuts \quad A cut \( C \) in \( G = [N;E] \) separating the source \( s \) and the sink \( t \) is a set of edges \((X,N\setminus X)\) such that \( s \in X \) and \( t \in N\setminus X \). The capacity of a cut \((X,N\setminus X)\) is \( c(X,N\setminus X) \). A simple example of the cut is given in Fig. 5.2.

Lemma 5.1 (Ford,Fulkerson;1956 (see also [8])). Let \( f \) be a flow from \( s \) to \( t \) in a network \([N;E]\), and let \( f \) have value \( v \). If \((X,N\setminus X)\) is a cut, separating \( s \) and \( t \), then

\[
f(X,N\setminus X) - f(N\setminus X,X) \leq c(X,N\setminus X)
\]

Proof. Since \( f \) is a flow, then \( f(s,N) - f(N,s) = v \); \( f(x,N) - f(N,x) = 0 \) for all \( x \in N\setminus\{s,t\} \), and \( f(t,N) - f(N,t) = -v \). Let us sum these equations over \( x \in X \). Since
**Figure 5.2:** The set of edges $\mathcal{C} = \{(s, y), (x, y), (x, t)\}$ with $X = \{s, x\}$ and $N \setminus X = \{y, t\}$ is the cut separating $s$ and $t$. Its capacity is $c(s, y) + c(x, y) + c(x, t) = 3 + 1 + 3 = 7$.

$s \in X$ and $t \in N \setminus X$, the result is

$$v = \sum_{x \in X} (f(x, N) - f(N, x)) = f(X, N) - f(N, X)$$

Because of the obvious relationship $N = X \cup (N \setminus X)$ the above equality yields

$$v = f(X, X \cup (N \setminus X)) - f(X \cup (N \setminus X), X)$$
$$= f(X, X) + f(X, N \setminus X) - f(X, X) - f(N \setminus X, X)$$

thus verifying the equality in Eq. (5.3). Since $f(N \setminus X, X) \geq 0$ and $f(X, N \setminus X) \leq c(f(X, N \setminus X)$ due to Eq. (5.1), the inequality in Eq. (5.3) follows immediately. $\square$

**Maximal flow** The equality in Eq. (5.3) states that the value of a flow from $s$ to $t$ is equal to the net flow across any cut separating $s$ and $t$. The fundamental result concerning maximal network flow is given by

**Theorem 5.1** (Max-flow min-cut theorem: Ford, Fulkerson 1956 (see also [8])). For any network the maximum flow value from $s$ to $t$ is equal to the minimal cut capacity of all cuts separating $s$ and $t$. The proof is given in [8].

**5.2.2 Ford-Fulkerson max-flow algorithm**

Let a flow augmenting path with respect to a flow $f$ be defined as a path from $s$ to $t$ such that $f < c$ on forward edges of the path and $f > 0$ on reverse edges of the path. The following corollary is of fundamental importance in searching for maximal network flows:
Corollary 5.1. A flow \( f \) is maximal if and only if there is no flow augmenting path with respect to \( f \).

The proof is given in [8].

This corollary states that in order to increase the value of a flow, it is sufficient to search for improvements of a very restricted kind. Let an edge \((x, y)\) be called saturated with respect to a flow \( f \) if \( f(x, y) = c(x, y) \) and called flowless with respect to \( f \) if \( f(x, y) = 0 \). An edge being both saturated and flowless has zero capacity. A minimal cut is characterized in terms of these notions by

Corollary 5.2. A cut \( X, N \setminus X \) is minimal if and only if every maximal flow saturates all edges of \((X, N \setminus X)\) whereas all edges of \((N \setminus X, X)\) are flowless with respect to \( f \).

The proof is given in [8].

Ford-Fulkerson labeling algorithm
Under mild restrictions on the capacity function, the proof of the max-flow min-cut theorem 5.1 provides an algorithm for constructing a maximal flow and minimal cut in a network. To ensure termination, all the capacities should have integer values. Initialization is with the zero flow. Then a sequence of "labelings" is performed (Routine A below). Each labeling either results in a flow of higher value (Routine B below) or terminates with the conclusion that the present flow is maximal.

Given an integral flow \( f \), labels are assigned to nodes of the network according to Routine A in Fig. 5.3. A node can be in one of three states: unlabeled (UL), labeled and scanned (LS), or labeled and unscanned (LUS). Each label has one of the forms \((x^+, \epsilon)\) or \((x^-, \epsilon)\), where \( x \in N \) and \( \epsilon \) is a positive integer or \( \infty \). Initially all nodes are UL.

The labeling process searches systematically for a flow augmenting path from \( s \) to \( t \) (see Corollary 5.1). Information about the paths is carried along in the labels, so that if the sink is labeled, the resulting flow change along the path is readily made. On the other hand, if Routine A ends and the sink is not labeled, the flow is maximal and the set of edges leading from labeled (LUS, LS) nodes to unlabeled (UL) nodes is a minimal cut [8].

The labeling process is computationally efficient because once a node is labeled and scanned, it is ignored for the remainder of the process. Labeling a node \( x \) locates a path from \( s \) to \( x \) that can be the initial segment of a flow augmenting path. There may be many such paths from \( s \) to \( x \), but it is sufficient to find one.

Computational complexity
Both the initial Ford-Fulkerson max-flow min-cut algorithm and subsequent solutions of the same problem have polynomial time complexity.
**Routine A**: labeling

**Initial step**: Make the source LUS with the label $(-, e(s) = \infty)$; other nodes remain UL.

**Iteration**: Repeat the general step until

1: either the sink $t$ is LUS $-$ then go to **Routine B** or
2: no more labels can be assigned and the sink is UL $-$ then terminate

**General step**: Select any LUS node $x$ with a label $(z^x, e(x))$.

1: Convert all UL nodes $y$ such that $f(x, y) < c(x, y)$ into the LU nodes with the labels $(x^+, e(y) = \min\{e(x), c(x, y) - f(x, y)\})$.

2: Convert now all UL nodes $y$ such that $f(y, x) > 0$ into LUS nodes with the labels $(x^-, e(y) = \min\{e(x), f(y, x)\})$

3: The node $x$ is now LS.

**Routine B**: flow change: the sink $t$ has been labeled $(y^t, e(t))$

**Start**: If $t$ is labeled $(y^+, e(t))$, replace $f(y, t)$ by $f(y, t) + e(t)$, otherwise if $t$ is labeled $(y^-, e(t))$, replace $f(t, y)$ by $f(t, y) - e(t)$, and in either case go on to node $y$

**Repeat**: In general, if $y$ is labeled $(x^+, e(y))$, replace $f(x, y)$ by $f(x, y) + \epsilon(t)$, otherwise if $y$ is labeled $(x^-, e(y))$, replace $f(y, x)$ by $f(y, x) - \epsilon(t)$, and in either case go on to node $x$

**Stop** the flow change when the source $s$ is reached, discard the old labels, and go back to **Routine A**

---

**Figure 5.3**: Ford-Fulkerson max-flow/min-cut algorithm.

(below, $n = |N|$ is the size, or cardinality of the set of nodes $N$ and $m = |E|$ is the size of the set of edges $E$):
### Algorithm | Principle | Complexity
--- | --- | ---
Ford–Fulkerson, 1956 | Finding flow augmenting paths | $O(nm^2)$
Dinic, 1970 | Shortest augmenting paths in 1 step in a dense graph: | $O(n^2 m)$
in a sparse graph: | $O(n^3)$
Goldberg–Tarjan, 1985 | Pushing a pre-flow | $O(nm \log n)$

The time complexity of the Ford-Fulkerson algorithm depends on how the flow augmenting paths are determined. The $O(nm^2)$ Edmonds-Karp algorithm implements the Ford-Fulkerson scheme by using the **breadth-first search** for the augmenting path being a shortest path from $s$ to $t$ in the residual graph (network) where each edge has unit length, or weight. For a given network $G = [N; E]$ and a flow $f$, the residual network $G_f$ induced by $f$ consists of edges, $E_f$, that allow for increasing the flow. The additional flow permitted by the edge capacity is called the residual capacity of the edge, $r_f(u, v) = c(u, v) - f(u, v)$, $(u, v) \in E$. An edge $(x, y)$ with the non-zero residual capacity is called a residual edge if $r_f(x, y) > 0$. The residual graph $G_f = [N; E_f]$ for a pre-flow $f$ is the graph with the same set of nodes $N$ but with the set of only residual edges: $E_f = \{(x, y) : (x, y) \in E; r_f(x, y) > 0\}$. The maximum increment of the flow along each augmenting path is called the residual capacity of the path and is equal to $r_f(\text{path } p) = \min_{(u, v) \in p} \{r_f(u, v)\}$.

#### 5.2.3 Goldberg-Tarjan max-flow algorithm

Search strategies based on flow augmenting paths keep a flow at each step although the maximal flow is needed only at the very end. This is why these strategies turned to be less efficient than the subsequent ones based on a pre-flow notion coined by A. V. Karzanov in 1974. The Karzanov’s pre-flow violates the conditions of Eq. (5.2): the in-flow and out-flow nodes may not be equal, the difference at node $x$ being called the excess at $x$. If $f$ is a pre-flow in the graph $G = [N; E]$, then the excess $e(x)$ at $x$ is $e(x) = f(B(x), x) - f(x, A, x)$.

The "push–label" search process pushes as much as possible into the graph, then trims the excess to zero. Until the very end (when the excess at each node goes eventually to zero), no flow exists in the graph. To describe the process, the residual capacity $r_f(x, y)$ of an edge $x, y$ is defined as $r_f(x, y) = c(x, y) - f(x, y) + f(y, x)$. The distance $d_C(x, y)$ from $x$ to $y$ in $G$ is defined as the minimum number of edges on a path from $x$ to $y$ in $G$. If there is no such path, $d_C(x, y) = \infty$. To estimate the distance from a node to $s$ or $t$, a
valid labeling $d$ is defined \cite{13} as a function $d : \mathbb{N} \rightarrow \mathbb{R}^{>0}$, such that $d(s) = n$, $d(t) = 0$, and $d(x) \leq d(y) + 1$ for every residual edge $(x, y)$. If $d(x) < n$, then $d(x)$ is a lower bound of the actual distance from $x$ to $t$ in the residual graph $G_f$, and if $d(x) \geq n$, then $d(x) - n$ is a lower bound on the actual distance to $s$ in the residual graph (it can be proven that in the latter case $t$ is not reachable from $x$ in $G_f$). A node $x$ is called active if $x \in \mathbb{N} \setminus \{s, t\}$, $d(x) < \infty$, and $e(x) > 0$ (the source and sink are never active). The algorithm in Fig. 5.4 iteratively selects active nodes, tries to push as much as possible from them, and relabel them in order to convert a pre-flow into a maximal flow.

| Initialization: Set $d(s) = n$; $d(t) = 0$; $\forall x \in \mathbb{N} \setminus \{s, t\} d(x) = 1$. For every edge $s, x) \in E$, set $f(s, x) = c(s, x)$
| Push–label process: While there are active nodes, select an active node, $x$, and try to push more pre-flow towards the sink
| Pushing is possible in the two situations:
| Condition 1: $x, y) \in E$, $d(x) = d(y) + 1$, and $r_f(x, y) > 0$, or
| Condition 2: $y, x) \in E$, $d(x) = d(y) + 1$, and $r_f(x, y) > 1$
| In either situation, there is capacity to push $\Delta = \min\{e(x), r_f(x, y)\}$ units of flow\textsuperscript{a} from $x$ to $y$ and update the excesses and pre-flow values accordingly:
| $f(x, y) \leftarrow f(x, y) + \Delta$; $f(y, x) \leftarrow f(y, x) - \Delta$
| $e(x) \leftarrow e(x) - \Delta$; $e(y) \leftarrow e(y) + \Delta$
| Relabeling of $x$ if nothing can be pushed more from it: $d(x) \leftarrow \min\{d(y) + 1 : (x, y) \in A(x); r(x, y) > 0\}$
| Termination: Once the push–label process is finished, the pre-flow $f$ is a maximum flow

\textsuperscript{a}That is, as much as the excess at $x$ and the residual capacity of the edge $(x, y)$ allow.

\textbf{Figure 5.4: Goldberg-Tarjan push-label max-flow algorithm.}
5.3 Energy minimization via graph cuts: a binary case

Greig et al. [14] were the first who have shown how Bayesian MAP estimation for denoising (restoration) of binary images can be reformulated as a minimum cut problem in a certain network and solved exactly with the max-flow/min-cut technique such as the Ford-Fulkerson algorithm.

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a noisy image such that the measurements $y_i; \ i = 1, \ldots, n$, are conditionally independent given a noiseless image $\mathbf{x}$. Each measured signal $y_i$ has a known conditional density function $p(y_i|x_i)$, dependent on $\mathbf{x}$ only through $x_i$. Therefore,

$$p(y|x) = \prod_{i=1}^{n} p(y_i|x_i) = \prod_{i=1}^{n} [p(y_i|1)]^{x_i} [p(y_i|0)]^{1-x_i} = \prod_{i=1}^{n} \left( \frac{p(y_i|1)}{p(y_i|0)} \right)^{x_i} p(y_i|0)$$

The hidden noiseless images are modeled as samples of a Markov random field with pairwise interactions having the Gibbs probability distribution

$$p(x) \propto \exp \left[ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} (x_i x_j + (1-x_i)(1-x_j)) \right]$$

where $\beta_{ii} = 0$ and $\beta_{ij} = \beta_{ji} \geq 0$. The inequality $\beta_{ij} = \beta_{ji} > 0$ holds for neighbors in the lattice, e.g. for the nearest 4- or 8-neighbors.

Apart from an additive constant, the posterior log-likelihood $\ell(x|y)$ is as follows:

$$\ell(x|y) = \ln \left( \frac{p(x)p(y|x)}{p(y)} \right) = \sum_{i=1}^{n} \lambda_i x_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} (x_i x_j + (1-x_i)(1-x_j))$$

where $\lambda_i = \ln (p(y_i|1)/p(y_i|0))$ is a log-likelihood ratio at pixel $i$. The Bayesian MAP estimate of the image maximizes the posterior likelihood:

$$\hat{x} = \arg \max_{x \in \{0,1\}^n} \ell(x|y) = \arg \min_{x \in \{0,1\}^n} [-\ell(x|y)]$$

**Capacitated network model** contains $n+2$ nodes, being a source $s$, a sink $t$, and the $n$ pixels. If the log-likelihood ratio is positive $\lambda_i > 0$, there is a directed edge $(s, i)$ from $s$ to pixel $i$ with capacity $c(s, i) = \lambda_i$; otherwise, there is a directed edge $(i, t)$ from $i$ to $t$.
with capacity $c(i, t) = -\lambda_i$. There is an undirected edge $(i, j)$ between two pixels $i$ and $j$ with capacity $c(i, j) = \beta_{ij}$ if these pixels are neighbors. Figure 5.5 exemplifies such a network model.

![Network Model Diagram]

**Figure 5.5:** A network model for resoring a binary image having a Markov-Gibbs prior model with 4-neighborhood interactions (signals of a given noisy image $y_1, \ldots, y_n$ that result in positive, $\lambda_i > 0$, and non-positive, $\lambda_i \leq 0$, log-likelihood ratios are shown with black and white pixel nodes, respectively).

Any binary image $\mathbf{x} = (x_1, \ldots, x_n)$ produces a cut $(S, T)$ where $S = \{s\} \cup \{i : x_i = 1\}$ and $T = \{i : x_i = 0\} \cup \{t\}$ with the capacity

$$c(\mathbf{x}) = c(S, T) = \sum_{k \in S} \sum_{l \in T} c(k, l)$$

that can be represented as

$$c(\mathbf{x}) = \sum_{i=1}^{n} x_i \max(0, -\lambda_i) + \sum_{i=1}^{n} (1 - x_i) \max(0, \lambda_i) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} (x_i - x_j)^2$$

The latter differs from $-\ell(\mathbf{x}|\mathbf{y})$ by a term which does not depend on $\mathbf{x}$. Therefore, maximizing $\ell(\mathbf{x}|\mathbf{y})$ is equivalent to finding the minimum cut of the above network: in the
MAP estimate pixels \( x_i = 1 \) if they are on the source side \( S \) of the minimum cut and \( x_i = 0 \) otherwise.

Experiments in [14] have underlined the need of an adequate prior distribution because its global properties very rapidly can dominate the likelihood contribution to the posterior distribution. At the same time, the use of stochastic optimization such as simulated annealing algorithm applied with practicable schedules\(^1\), does not necessarily produce a good approximation to an MAP estimate.

**Accelerated solution:** A ten-fold acceleration of the basic Ford-Falkerson algorithm is obtained in [14] for this network model by partitioning the input image into \( 2^K \times 2^K \) connected sub-images. The MAP estimate is calculated for each sub-image separately, then the sub-images are amalgamated to form a set of \( 2^{K-1} \times 2^{K-1} \) larger sub-images. The MAP estimate is formed for each of them, and this process is continued until the MAP estimate of the complete image.

### 5.3.1 Energies minimized by graph cuts

**Theorem 5.2** (Friedman,Drineas; 2005\(^2\) [9]). Let \( x_i \in \{0, 1\}; i = 1, \ldots, n \), and let

\[
E(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} x_i x_j + L
\]

where \( L \) represents terms that are linear in the \( x_i \) plus any constants, i.e. \( L = \sum_{i=1}^{n} \lambda_i x_i + c \). Then \( E \) can be minimized via graph cut techniques if and only if \( \beta_{ij} \leq 0 \) for all \( i, j \).

**Proof.** To prove the “if” direction, the energy can be rewritten as

\[
E(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ij} x_i (1 - x_j) + L
\]

where \( \alpha_{ij} = -\beta_{ij} \) and the linear term \( L \) is altered. Minimum \( E(\ldots) \) over the binary variables \( x_i \) is given by a minimum cut in a complete graph with \( n \) nodes and edge

---

\(^1\)According to the “annealing theorem” of Geman and Geman [11], the inverse logarithmic decrease with time of the temperature term in a Gibbs distribution that controls the annealing process guarantees that the MAP solution will be eventually reached, although this may take an infinite time.

\(^2\)Generally, it is considered to be part of combinatorial optimization folklore; see [17].
weights \( w_{ij} = \alpha_{ij} \). The cut splits the nodes with \( x_i = 0 \) from those with \( x_j = 1 \) because only the pair \( x_i = 1 \) and \( x_j = 0 \) adds the value \( \alpha_{ij} \) to the energy. As follows from the combinatorial optimization theory [8], polynomial time min cut is possible if and only if \( w_{ij} \geq 0 \), that implies \( \beta_{ij} \leq 0 \).

For altered linear terms \( \Lambda = \sum_{i=1}^{n} \gamma_i x_i + \sigma \), there is an edge \((s,i)\) with the weight \( w_{si} = \gamma_i \) if \( \gamma_i \geq 0 \) and an edge \((i,t)\) with \( w_{si} = |\gamma_i| \) if \( \gamma_i < 0 \), so that all weights are non-negative.

The proof of “only if” is slightly more complicated, see [17].

The MAP estimates based on the Markov-Gibbs posteriors with pairwise interactions result in the class of energy functions:

\[
E(x_1, \ldots, x_n) = \sum_{i=1}^{n} E_i(x_i) + \sum_{i=1}^{n} \sum_{j=1}^{n} E_{ij}(x_i, x_j); \quad (x_1, \ldots, x_n) \in \{0, 1\}^n
\]

Each pairwise energy term has only four values and can be equivalently rewritten in terms of these four values as

\[
E_{ij}(x_i, x_j) = E_{ij}^{00}(1 - x_i)(1 - x_j) + E_{ij}^{01}(1 - x_i)x_j + E_{ij}^{10}x_i(1 - x_j) + E_{ij}^{11}x_i x_j
\]

\[
= (E_{ij}^{00} + E_{ij}^{11} - E_{ij}^{01} - E_{ij}^{10})x_i x_j + \text{linear term in } x_i \text{ and } x_j
\]

Therefore, in accord with Theorem 5.2, such energies can be minimized with the graph min-cut techniques if and only if the regularity condition

\[
E_{ij}^{00} + E_{ij}^{11} - E_{ij}^{01} - E_{ij}^{10} \leq 0
\]

holds for all neighboring pixel pairs \( i, j \) forming edges of the graph model.

More sophisticated sufficient conditions for energy minimization using the min-cut techniques exist also for energy functions with \( k \)-wise interaction, \( k > 2 \). But natural generalizations of the minimum cut problem, such as a multiway cut and minimum \( k \)-cut, are NP-hard:

**Multiway cut** Given a set of terminals \( S = \{s_1, s_2, \ldots, s_k\} \subseteq N \), a multiway cut is a set of edges whose removal disconnect the terminals from each other. The multiway cut problem is to find such a set with the minimum weight.

**Minimum \( k \)-cut** A set of edges whose removal leaves \( k \) connected components is called a \( k \)-cut. The minimum \( k \)-cut problem asks for a minimum weight \( k \)-cut.
The problem of finding a minimum weight multiway cut is NP-hard for any fixed $k \geq 3$ (the case $k = 2$ is the minimum $s$-$t$ cut problem). The minimum $k$-cut problem has polynomial time complexity for fixed $k$ and is NP-hard if $k$ is one of input variables. Both the problems have approximation algorithms that guarantee the solution within factor $2 - 2/k$ from the exact optimum [22].

### 5.4 Exercises

**Exercise 5.1.** Find the maximal flow and a minimal cut for the network in Fig. 5.6 using the Ford-Fulkerson algorithm (in its Edvards-Karp implementation). Show steps of executing the algorithm including the residual networks and flow augmenting paths found.

![Diagram](image)

**Figure 5.6:** Numbers are the edge capacities.

**Exercise 5.2.** Show steps of executing the Goldberg-Tarjan algorithm for the network in Fig. 5.6.

**Exercise 5.3.** Show how the log-likelihood $\ell(x|y)$ in Section 5.3 for the MAP denoising of a noisy binary image can be transformed into the capacity $c(x)$ of a cut of the network representing this problem. Justify that the exact MAP denoising is equivalent to the minimum cut of the network.

**Exercise 5.4.** Apply the regularity condition of Eq. (5.4) to the network in Section 5.3.