Chapter 10

Minimization – Global Methods

The minimal solution is usually defined as the global one or one of them when there are multiple global minima. Finding a global minimum is non-trivial if the energy function contains many local minima. Whereas methods for local minimization are quite mature, with commercial software on the market, the study of global minimization is still young. There are no efficient algorithms that guarantee finding globally minimal solutions as there are for local minimization.

In analytical studies, local and global minima may be studied via convexity analysis. Let $E(f)$ be a real function defined on $\mathbb{F} = \mathbb{R}^n$. $E(f)$ is said to be convex in $\mathbb{F}$ if

$$E(\lambda x + (1-\lambda)y) \leq \lambda E(x) + (1-\lambda)E(y) \tag{10.1}$$

for any two points $x, y \in \mathbb{F}$ and any real number $\lambda \in [0, 1]$. It is strictly convex if strict inequality "$<$" holds in the above. When plotted, a convex $E(f)$ will always lie below the line segment connecting any two points on itself. The neighboring points of a minimum are always above the tangent at the minimum. A local minimum is also a global one if $E(f)$ is convex on $\mathbb{F}$. It is the unique global minimum if $E(f)$ is strictly convex. Therefore, when $E(f)$ is convex, we can find a global minimum by finding a local minimum using the gradient-descent method.

Global minimization requires (1) finding all (a finite number of) local minima and (2) proving that there are no more local minima. Without an efficient algorithm, this amounts to an exhaustive search. In reality, one is always facing one of two choices: (1) to find the exact global minimum with possibly intolerable expense or (2) to find some approximations to it with much less cost.

S.Z. Li, *Markov Random Field Modeling in Image Analysis*,
Advances in Pattern Recognition, DOI: 10.1007/978-1-84800-279-1_10,

Two methods may be used to deal with the local minimum problem: random search and annealing. In random search methods, a new configuration does not always make energy descend; occasional energy increases are allowed. The probability with which a configuration is generated is proportional to $e^{-E(f)/T}$, where T is a control parameter called the temperature. Therefore, a lower-energy configuration is generated with a larger probability.

Annealing is incorporated into a local search method to overcome the problem of local minima. It is performed by decreasing the temperature parameter in the Gibbs distribution from a high value to a low value during the iterative minimization. At one extreme (e.g., a high temperature), the energy landscape is convex and smooth and thus the unique minimum can be located easily; the minimum is tracked as the is gradually decreased to a sufficiently low value. This is the so-called continuation method (Wasserstrom 1973). Such a technique can significantly overcome the local minimum problem and improve the quality of the solution. A disadvantage is that they take more time since an annealing algorithm has to decrease the parameter gradually over a range of values and at each value some convergence has to be reached.

There are two types of annealing: deterministic and stochastic. In MRF vision work, the stochastic simulated annealing algorithm (Geman and Geman 1984) and the deterministic graduated nonconvexity (GNC) algorithm of Blake and Zisserman (1987) enjoy a popularity. Other deterministic algorithms include mean field annealing (Peterson and Soderberg 1989; Yuille 1990; Geiger and Girosi 1991) and the Hopfield network approach (Hopfield and Tank 1985; Koch et al. 1986; Yuille 1987).

While stochastic annealing such as simulated annealing is theoretically justified (Geman and Geman 1984), deterministic annealing remains heuristic. There is no guarantee that the minimum at high temperature can always be tracked to the minimum at low temperature.

10.1 Simulated Annealing

Simulated annealing (SA), introduced independently by Cerny (1982), Kirkpatrick, Gellatt, and Vecchi (1982),Cerny (1985), and Kirkpatrick, Gellatt, and Vecchi (1983), is a stochastic algorithm for combinatorial optimization. It simulates the physical annealing procedure in which a physical substance is melted and then slowly cooled down in search of a low energy configuration. Consider a system in which any f in the configuration space $\mathbb{F}$ has probability

$$P_T(f) = [P(f)]^{1/T} \tag{10.2}$$

where $T > 0$ is the temperature parameter. When $T \to \infty$, $P_T(f)$ is a uniform distribution on $\mathbb{F}$; for $T = 1$, $P_T(f) = P(f)$; and as $T \to 0$, $P_T(f)$ is concentrated on the peak(s) of $P(f)$. This gives intuition as to how the samples of f distribute in $\mathbb{F}$.

```
initialize T and f;
repeat
    randomly sample f from N(f) under T;
    decrease T;
until (T → 0);
return f;
```

Figure 10.1: The simulated annealing algorithm.

The SA algorithm is described in Fig. 10.1. SA applies a sampling algorithm, such as the Metropolis algorithm (Metropolis et al. 1953) or Gibbs sampler (Geman and Geman 1984) (see Section 7.1.6), successively at decreasing values of temperature T. Initially, T is set very high and f is set to a random configuration. At a fixed T, the sampling is according to the Gibbs distribution $P_T(f) = \mathrm{e}^{-E(f)/T} / \sum_f \mathrm{e}^{-E(f)/T}$. After the sampling converges to the equilibrium at current T, T is decreased according to a carefully chosen schedule. This continues until T is close to 0, at which point the system is "frozen" near the minimum of $E(f)$. The cooling schedule, specified by a decrement function and a final value, plays an important part; see below.

Two convergence theorems have been developed (Geman and Geman 1984). The first concerns the convergence of the Metropolis algorithm. It states that if every configuration is visited infinitely often, the distribution of generated configurations is guaranteed to converge to the Boltzmann (i.e., Gibbs) distribution. The second is about SA. It states that if the decreasing sequence of temperatures satisfy

$$\lim_{t \to \infty} T^{(t)} = 0 \tag{10.3}$$

and

$$T^{(t)} \geq \frac{m \times \Delta}{\ln(1+t)} \tag{10.4}$$

where $\Delta = \max_f E(f) - \min_f E(f)$, then the system converges to the global minimum regardless of the initial configuration $f^{(0)}$. Note that the conditions above are sufficient but not necessary for the convergence.

Unfortunately, the schedule (10.4) is too slow to be of practical use. In practice, heuristic, faster schedules have to be used instead. Geman and Geman (1984) adopt the following

$$T^{(t)} = \frac{C}{\ln(1+t)} \tag{10.5}$$

where the constant C is set to $C = 3.0$ or $C = 0.4$ for their problem. Kirkpatrick et al. (1983) choose

$$T^{(t)} = \kappa T^{(t-1)} \tag{10.6}$$

where κ typically takes a value between 0.8 and 0.99. The initial temperature is set high enough that essentially all configuration changes are accepted. At each temperature, enough configurations are tried that either there are $10m$ (10 times the number of sites) accepted configuration transitions or the number of tries exceeds $100m$. The system is frozen and annealing stops if the desired number of acceptances is not achieved at three successive temperatures. In (Kato et al. 1993a), a multi-temperature annealing scheme is proposed for annealing in multiresolution computation; there, the temperature in the Gibbs distribution is related not only to the time but also to the scale.

The two sampling procedures (i.e., the Metropolis algorithm and Gibbs sampler) are proven to be asymptotically equivalent for SA performed on lattices, but this is not generally true for non-lattice structures (Chiang and Chow 1992). The interested reader is referred to (Kirkpatrick et al. 1982; Geman and Geman 1984; Aarts 1989) for discussions devoted on SA.

A performance comparison between SA and GNC for edge-preserving surface reconstruction is given in (Blake 1989). Based on the results of the experiments under controlled conditions, Blake argues that GNC excels SA both in computational efficiency and problem-solving power. An experimental comparison between SA and deterministic RL algorithms for combinatorial optimization will be presented in Section 10.6.1.

10.2 Mean Field Annealing

Mean field annealing (Peterson and Soderberg 1989) provides another approach for solving combinatorial optimization problems using real computation. It can be considered as a special RL algorithm incorporating an annealing process. Let $f = \{f_i(I) \in \{0,1\} | i \in \mathcal{S}, I \in \mathcal{L}\}$ be an unambiguous labeling assignment in $\mathbb{P}^*$ defined in (9.24); each f_i takes the value of one of the vectors $(1,0,\ldots,0)$, $(0,1,\ldots,0)$, and $(0,0,\ldots,1)$. Still denote the energy with the RL representation by $E(f)$. The Gibbs distribution of the energy under temperature $T > 0$ is given as

$$P_T(f) = Z^{-1} \mathrm{e}^{-\frac{1}{T}E(f)} \tag{10.7}$$

where the partition function

$$Z = \sum_{f \in \mathbb{P}^*} \mathrm{e}^{-\frac{1}{T}E(f)} \tag{10.8}$$

sums over $\mathbb{P}^*$. The statistical mean of the distribution is defined as

$$\langle f \rangle_T = \sum_f f_i P_T(f) = \sum_f f_i Z^{-1} \mathrm{e}^{-\frac{1}{T}E(f)} \tag{10.9}$$

As the temperature approaches zero, the mean of the Gibbs distribution approaches its mode or the global energy minimum f^*

$$\lim_{T \to 0} \langle f \rangle_T = \lim_{T \to 0} \sum_f f P_T(f) = f^* \tag{10.10}$$

This suggests that instead of minimizing $E(f)$ directly, we could try to evaluate the mean field $\langle f \rangle_T$ at a sufficiently high temperature and then track it down using the continuation method (Wasserstrom 1973) as the temperature is lowered toward zero.

The analysis of the partition function is central in mean field theory because once it is calculated all statistical information about the system can be deduced from it. The mean field trick for calculating the partition function is to (1) rewrite the sum over a discrete space as an integral over a pair of continuous variables p and q, $p = \{p_i(I) \in \mathbb{R} | i \in \mathcal{S}, I \in \mathcal{L}\}$ and $q = \{q_i(I) \in \mathbb{R} | i \in \mathcal{S}, I \in \mathcal{L}\}$ having the same dimensionality as f, and then (2) evaluate its integrand at the saddle point (saddle-point approximation). As will be seen later, p and q correspond to a labeling assignment as defined in (9.20) and a gradient as defined in (9.29), respectively. In the following, f, p, and q are considered $m \times M$ matrices. To do step (1), we begin by rewriting any function $g(f)$ as the integral

$$g(f) = \int_{\mathbb{D}_R} g(p) \delta(f - p) \mathrm{d}p \tag{10.11}$$

where δ is the Dirac delta function and $\mathbb{D}_R$ is the $|\mathcal{S}| \times |\mathcal{L}|$ dimensional real space. The definition of the Dirac delta function

$$\delta(q) = C \int_{\mathbb{D}_I} \mathrm{e}^{p^{\mathrm{T}} q} \mathrm{d}p \mathrm{d}q \tag{10.12}$$

is used in the above, where T denotes transpose, $\mathbb{D}_I$ is the $|\mathcal{S}| \times |\mathcal{L}|$-dimensional imaginary space, and C is a normalizing constant. Therefore,

$$g(f) = C \int_{\mathbb{D}_R} \int_{\mathbb{D}_I} g(p) \mathrm{e}^{(f^{\mathrm{T}} q - p^{\mathrm{T}} q)} \mathrm{d}p \mathrm{d}q \tag{10.13}$$

Using the formula above, we can rewrite the partition function, which is the sum of the function $\mathrm{e}^{-\frac{1}{T}E(p)}$, as

$$\begin{aligned} Z &= \sum_{f \in \mathbb{P}^*} C \int_{\mathbb{D}_R} \int_{\mathbb{D}_I} \mathrm{e}^{-\frac{1}{T}E(p)} \mathrm{e}^{(f^{\mathrm{T}} q - p^{\mathrm{T}} q)} \mathrm{d}p \mathrm{d}q \qquad (10.14) \\ &= C \int_{\mathbb{D}_R} \int_{\mathbb{D}_I} \mathrm{e}^{-\frac{1}{T}E(p)} \mathrm{e}^{(-p^{\mathrm{T}} q)} \left[\sum_{f \in \mathbb{P}^*} \mathrm{e}^{(f^{\mathrm{T}} q)} \right] \mathrm{d}p \mathrm{d}q \end{aligned}$$

The summation inside the integrand can be written as

$$\sum_{f\in\mathbb{P}^*} \mathrm{e}^{(f^{\mathrm{T}}q)} = \prod_{i\in\mathcal{S}} \sum_{I\in\mathcal{L}} \mathrm{e}^{q_i(I)} \tag{10.15}$$

Therefore,

$$Z = C \int_{\mathbb{D}_R} \int_{\mathbb{D}_I} \mathrm{e}^{-\frac{1}{T}E_{eff}(p,q)} \mathrm{d}p\mathrm{d}q \tag{10.16}$$

where

$$E_{eff}(p,q) = E(p) + Tp^{\mathrm{T}}q - T\sum_{i\in\mathcal{S}} \ln \sum_{I\in\mathcal{L}} \left(\mathrm{e}^{q_i(I)}\right) \tag{10.17}$$

is called the *effective energy*.

The integral expression of Z above is exact but too complicated for precise calculation. The calculation may be approximated based on the following heuristic argument: The double integral is dominated by saddle points in the integration intervals, and therefore the integral is approximated by

$$Z \approx \mathrm{e}^{-\frac{1}{T}E_{eff}(p^*,q^*)} \tag{10.18}$$

where (p^*, q^*) is a saddle point of E_{eff}. The saddle points are among the roots of the equations

$$\nabla_p E_{eff}(p,q) = 0 \quad \text{and} \quad \nabla_q E_{eff}(p,q) = 0 \tag{10.19}$$

where ∇_p is the gradient w.r.t. p. This yields the mean field theory equations

$$q_i(I) = -\frac{1}{T}\frac{\partial E(p)}{\partial p_i(I)} \quad \text{and} \quad p_i(I) = \frac{\mathrm{e}^{q_i(I)}}{\sum_J \mathrm{e}^{q_i(J)}} \tag{10.20}$$

The p matrix represents a labeling assignment in the feasibility space defined by (9.19). The mean field theory equations for combinatorial minimization have been derived in many places, for example, in (Yuille 1990; Simic 1990; Elfadel 1993; Haykin 1994; Yang and Kittler 1994).

Note that in terms of the RL representation, the energy can be written as $E(f) = Const - G(f)$, and so q can be computed as

$$q_i(I) \quad = \quad \tfrac{1}{T}\left[r_i(I) + 2\sum_{i'\in\mathcal{S}}\sum_{I'\in\mathcal{L}} r_{i,i'}(I,I')\ p_{i'}(I')\right] \tag{10.21}$$

(see 9.29). Now we obtain the fixed-point iteration

$$p_i^{(t+1)}(I) \leftarrow \frac{\mathrm{e}^{q_i^{(t)}(I)}}{\sum_J \mathrm{e}^{q_i^{(t)}(J)}} \tag{10.22}$$

In iterative computation, the temperature T in q is decreased toward 0^+ as the iteration preceeds. The unambiguity of p is achieved when $T \to 0^+$.

in (Peterson and Soderberg 1989), the convergence is judged by a quantity defined as

$$S = \frac{1}{m} \sum_{I} p_i^2(I) \tag{10.23}$$

also called the "saturation" of f. The iteration (10.22) is considered converged if $S > 0.99$. Analog network implementation schemes for mean field algorithms are investigated in (Elfadel 1993).

It is worth pointing out that the idea of mean field annealing for global optimization was proposed earlier by Pinkus (1968) as the integral limit method. There, the global minimum is expressed in closed form as a limit of an integral. Assume that $\mathbb{F}$ is the closure of a bounded domain in $\mathbb{R}^m$. Suppose that $E(f)$ is continuous on $\mathbb{F}$ and has a unique global minimum $f^* = \{f_1^*, \ldots, f_m^*\} \in \mathbb{F}$. Pinkus shows that the integral

$$\bar{f}_i = \frac{\int_{\mathbb{F}} f_i \mathrm{e}^{-\frac{1}{T}E(f)} df}{\int_{\mathbb{F}} \mathrm{e}^{-\frac{1}{T}E(f)} df} \qquad i = 1, \ldots, m \tag{10.24}$$

approaches the global minimum f^* as $T \to 0^+$. Obviously, the integral above is the continuous version of the mean field defined in (10.9), and the idea of the integral limit is similar to that of mean field annealing. Theoretical analysis shows that this approximation has rather good asymptotic properties, but its realization may need some heuristic trials. Computationally, the values f_i^* can be obtained analytically only in exceptional cases, and a numerical method has to be used in general.

10.3 Graduated Nonconvexity

Graduated nonconvexity (GNC) (Blake and Zisserman 1987) is a deterministic annealing method for approximating a global solution for nonconvex minimization of unconstrained, continuous problems such as (9.5). It finds good solutions with much less cost than stochastic simulated annealing. Energies of the form

$$E(f) = \sum_{i=1}^{m} \chi_i (f_i - d_i)^2 + \lambda \sum_{i=1}^{m} \sum_{i' \in \mathcal{N}_i} g_\gamma (f_i - f_{i'}) \tag{10.25}$$

will be considered in the subsequent discussion of GNC. When g is a nonconvex function, a gradient-based method such as (9.7) finds only a local minimum.

10.3.1 GNC Algorithm

The idea of GNC is the following. Initially, γ is set to a sufficiently large value $\gamma^{(0)}$ such that $E(f|\gamma^{(0)})$ is strictly convex. It is easy to find the unique

minimum of $E(f|\gamma^{(0)})$ regardless of the initial f by using the gradient-descent method. The minimum $f^*_{\gamma^{(0)}}$ found under $\gamma^{(0)}$ is then used as the initial value for the next phase of minimization under a lower value $\gamma^{(1)} < \gamma^{(0)}$ to obtain the next minimum $f^*_{\gamma^{(1)}}$. As γ is lowered, $E(f|\gamma)$ becomes nonconvex and local minima appear. However, if we track the sequence of minima as γ decreases from the high value $\gamma^{(0)}$ to the target value γ, we may approximate the global minimum f^*_γ under the target value γ (i.e., the global minimum f^* of the target energy $E(f)$).

The first term on the right-hand side of (10.25) is quadratic and hence strictly convex w.r.t. f. The second term may or may not be convex, depending on f and g. Its nonconvexity could be partly compensated by the convexity of the closeness term. If $g(\eta) = g_q(\eta) = \eta^2$, then the second term is strictly convex and so is $E(f)$. If $g(\eta) = g_\alpha(\eta) = \min\{\eta^2, \alpha\}$ (see (5.10)), the second term is nonconvex and so is $E(f)$. However, if the function g and the parameters involved are chosen to satisfy

$$g''(f_i - f_{i-1}) \geq -c^* \qquad \forall i \tag{10.26}$$

where $c^* > 0$ is some constant, then the convexity of $E(f)$ is guaranteed. The value of c^* depends on the type of model and whether the data d are complete. Provided d_i are available for all i, the value is $c^* = 1/2$ for the string, $1/4$ for the membrane, $1/8$ for the rod, and $1/32$ for the plate (Blake and Zisserman 1987). If d_i are missing altogether, it requires $g''(f_i - f_{i-1}) > 0$, so that the second term is convex by itself, to ensure the convexity of $E(f)$. The practical situation is generally better than this worst case because the data cannot be missing altogether.

Therefore, to construct an initially convex g_γ, we can choose an initial parameter $\gamma = \gamma^{(0)}$ for which $g''_{\gamma^{(0)}}(f_i^{(0)} - f_{i-1}^{(0)}) \leq 0$ for all i. This is equivalent to choosing a $\gamma^{(0)}$ such that $f_i^{(0)} - f_{i-1}^{(0)} \in B_{\gamma^{(0)}}$ where B_γ is the band. For APFs, 1, 2 and 3, the $\gamma^{(0)}$ must be larger than $2v$, $3v$ and v, respectively, where $v = \max_i[f_i^{(0)} - f_{i-1}^{(0)}]^2$.

The GNC algorithm is outlined in Fig. 10.2. Given d, λ, and a target value γ_{target} for γ, the algorithm aims to construct a sequence $\{\gamma^{(t)}\}$ ($\gamma^{(\infty)} \to \gamma_{target}$) and thus $\{f^{(t)}_{\gamma^{(t)}}\}$ to approach the global minimum $f^* = \lim_{t\to\infty} f^{(t)}_{\gamma^{(t)}}$ for which $E(f^*) = \min$. In the algorithm, ϵ is a constant for judging the convergence, and κ is a factor for decreasing $\gamma^{(t)}$ toward γ_{target}, $0 < \kappa < 1$. The choice of κ controls the balance between the quality of the solution and the computational time. In principle, $\gamma^{(t)}$ should vary continuously to keep good track of the global minimum. In discrete computation, we choose $0.9 \leq \kappa \leq 0.99$. A more rapid decrease of $\gamma^{(t)}$ (with smaller κ) is likely to lead the system to an unfavorable local minimum. Witkin, Terzopoulos, and Kass (1987) present a more sophisticated scheme for decreasing γ by relating the step to the energy change: $\gamma^{(t+1)} = \gamma^{(t)} - c_1 e^{-c_2|\nabla E|}$, where c_1 and c_2 are constants. This seems reasonable.

Choose a convex $\gamma^{(0)}$; Set $f^{(0)} \leftarrow d$ and $t = 0$;
Do 1) Update $f^{(t)}$ using (9.7);
2) Set $t \leftarrow t + 1$;
3) If $(f^{(t)} - f^{(t-1)} < \epsilon)$
set $\gamma^{(t)} \leftarrow \max\{\gamma_{target}, \kappa\gamma^{(t-1)}\}$;
Until $(f^{(t)} - f^{(t-1)} < \epsilon)$ and $(\gamma^{(t)} = \gamma_{target})$;
Set $f^* \leftarrow f^{(t)}$

Figure 10.2: A GNC algorithm for finding the DA solution.

To approximate the truncated quadratic function $\lambda g_\alpha(\eta) = \lambda \min\{\eta^2, \alpha\}$, Blake and Zisserman (1987) construct the function

$$\lambda g^{(p)}_{\alpha,\lambda}(\eta) = \begin{cases} \lambda\eta^2 & |\eta| < q \\ \alpha - c(|\eta| - r)^2/2 & q \le |\eta| < r \\ \alpha & \text{otherwise} \end{cases} \tag{10.27}$$

where $p \in [0, 1]$ is a continuation parameter, $c = c^*/p$, $r^2 = \alpha(2/c + 1/\lambda)$, and $q = \alpha/(\lambda r)$. The corresponding interaction function is

$$\lambda h^{(p)}_\alpha(\eta) = \begin{cases} \lambda\eta & |\eta| < q \\ -c(|\eta| - r)\text{sign}(\eta) & q \le |\eta| < r \\ 0 & \text{otherwise} \end{cases} \tag{10.28}$$

The continuation parameter p is decreased from 1 to 0. When $p = 1$, the corresponding $E^{(p=1)}_\alpha$ is convex, where E_α is the energy with g_α. When $p = 0$, $E^{(p=0)}_\alpha = E_\alpha$ equals the original energy. A performance comparison between GNC and SA for reconstruction is given in (Blake 1989).

However, the complexity of the convexity treatment above can be avoided. Instead of modifying the definition of g_α, we can simply modify the parameter α in it. To achieve graduated nonconvexity, we can use α as the control parameter and decrease it from a sufficiently high value, $\alpha^{(0)} > (\max|d_i - d_{i-1}|)^2$, toward the target value. Let f be initialized so that $\max|f^{(0)}_i - f^{(0)}_{i-1}| \le \max|d_i - d_{i-1}|$. Then the energy is convex at such $\alpha^{(0)}$. An assumption is that if so initialized, $\max|f^{(t)}_i - f^{(t)}_{i-1}| < \max|d_i - d_{i-1}|$ always holds at any t during the iteration. Parameter values for the convexity in this way are related to the band B defined in (5.29) for the DA (discontinuity-adaptive) model. In a similar way, parameter values for the convexity can be derived for the LP (line process) approximation models given by Koch, Marroquin, and Yuille (1986), Yuille (1987), and Geiger and Girosi (1991).

The computation can be performed using an analog network. Let f_i be the potential of neural cell i. Let $C_i = C = 1/2\mu$ be the membrane capacitance

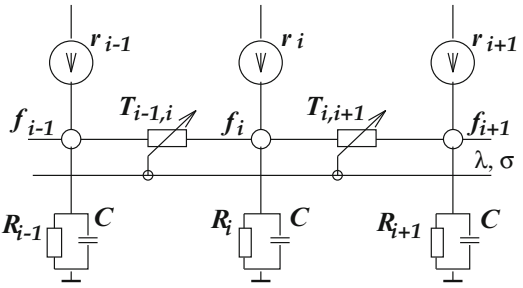


Figure 10.3: Schematic diagram of the analog network circuit.

and $R_i = 1/\chi_i$ the membrane resistance. Let

$$T_{i,i'} = T_{i',i} = \lambda h_\gamma(f_i - f_{i'}) \tag{10.29}$$

be the conductance or synaptic efficacy between neurons i and i', where $|i - i'| = 1$ and $h_\gamma \in \mathbb{H}_\gamma$. If the exponential $g_{1\gamma}$ is used, then

$$T_{i,i'} = T_{i',i} = \lambda \exp\{-[f_i - f_{i'}]^2/\gamma\} \tag{10.30}$$

Let d_i be the external current input to i, with $d_i = 0$ when $\chi_i = 0$. Now (9.7) can be written as

$$C\frac{\partial f_i}{\partial t} = -\frac{1}{R_i} f_i + d_i + T_{i-1,i}[f_{i-1} - f_i] + T_{i+1,i}[f_{i+1} - f_i] \tag{10.31}$$

The above is the dynamic equation at neuron i of the network. The diagram of the network circuit is shown in Fig. 10.3. The synaptic current from i to i' is

$$I_{i,i'} = \lambda g'_\gamma(f_i - f_{i'}) = \lambda[f_i - f_{i'}]h_\gamma(f_i - f_{i'}) \tag{10.32}$$

If the exponential $g_{1\gamma}$ is used, then

$$I_{i,i'} = \lambda[f_i - f_{i'}] \exp\{-[f_i - f_{i'}]^2/\gamma\} \tag{10.33}$$

A plot of current $I_{i,i'}$ versus potential difference $f_i - f_{i'}$ was shown at the bottom of Fig. 5.2. The voltage-controlled nonlinear synaptic conductance $T_{i,i'}$, characterized by the h function in (5.27), realizes the adaptive continuity control; the corresponding nonlinear current $I_{i,i'}$ realizes the adaptive smoothing. The current $I_{i,i'}$ diminishes asymptotically to zero as the potential difference between neurons i and i' reaches far beyond the band B_γ.

Comparing the DA and LP models, the former is more suitable for VLSI implementation than the latter. This is because the continuous shape of the adaptive conductance $T_{i,i'} = \lambda h_\gamma(f_i - f_{i'})$ in the DA is easier to implement using analog circuits than the piecewise λh_α in the LP. This advantage of the DA model is also reflected in the work by Harris, Koch, Staats, and

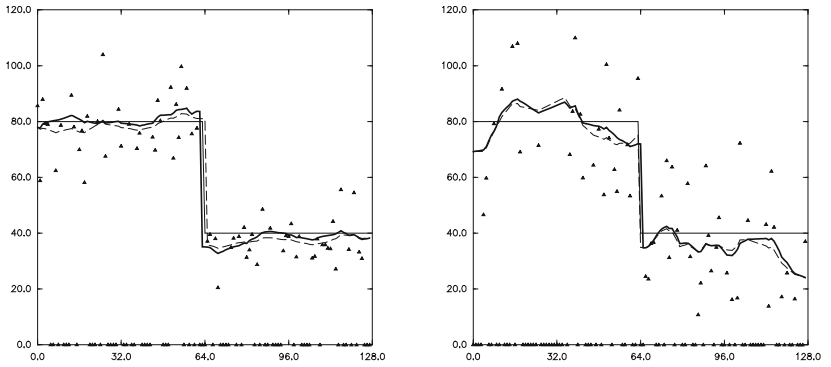


Figure 10.4: Stability of the DA solution under disturbances in parameters.

Lou (1990) and Lumsdaine, Waytt, and Elfadel (1990). There, the piecewise current-voltage characteristic $I = \Delta V h_\alpha(\Delta V)$ is implemented by a resistive fuse. Interestingly, the resistive fuse is of *finite gain* and thus does not give rise to a sharp switch-off as described by the piecewise h_α. The actual current-voltage characteristic of the resistive fuse is more like $I = \Delta V h_\gamma(\Delta V)$ than $I = \Delta V h_\alpha(\Delta V)$. It seems to satisfy all the requirements in (5.27) except for the C^1 continuity. The definition of $\mathbb{H}_\gamma$ offers guidelines for the DA circuit design.

Figure 10.4 shows the behavior of the analog DA network under component defects such as manufacturing inadequacy, quality changes, etc.The defects are simulated by adding $\pm 25\%$ evenly distributed random noise into R, C, and T in (10.31). The data d are shown in triangles with a 50% missing rate; the locations of the missing data, for which $\chi_i = 0$, are indicated by triangles at the bottom. The noise in the data is white Gaussian with standard deviation $\sigma = 10$ (left) and $\sigma = 20$ (right). The interaction function is chosen to be $h_{2\gamma}(\eta) = \frac{1}{[1+\eta^2/\gamma]^2}$. Solutions obtained with simulated component defects are shown as dashed lines in comparison with those obtained without such noise, shown as thicker solid lines. The ideal signal is shown as thinner solid lines. As can be seen, there is only a little difference between the solutions obtained with and without such noise. This demonstrates not only the stability of the network circuit but also the error-tolerance property of the DA model.

10.3.2 Annealing Labeling for MAP-MRF Matching

Here we describe a special GNC algorithm, called annealing labeling (Li et al. 1994), which may be used to improve the optimization of the posterior energy (see 4.18) formulated for the MAP matching.

We first do an energy-gain conversion (Section 9.3.2). Substituting (4.11), (4.12), (4.16), and (4.17) into (4.18), we have

$$\begin{aligned} U(f|d) = \quad & \{v_{10}N_1 + v_{20}N_2\} + \textstyle\sum_{i\in\mathcal{S}:f_i\neq 0} V_1(d_1(i)|f_i) \\ & + \textstyle\sum_{i\in\mathcal{S}:f_i\neq 0} \sum_{i'\in\mathcal{S}-\{i\}:f_{i'}\neq 0} V_2(d_2(i,i')|f_i,f_{i'}) \end{aligned} \tag{10.34}$$

where $N_1 = \#\{f_i = 0 | i \in \mathcal{S}\}$ is the number of NULL labels in f and $N_2 = \#\{f_i = 0 \text{ or } f_{i'} = 0 | i \in \mathcal{S}, i' \in \mathcal{N}_i\}$ is the number of label pairs at least one of which is NULL . The constants v_{10} and v_{20} control the number of sites assigned the NULL label. The smaller they are, the more sites will be assigned the NULL . The corresponding gain function is

$$\begin{aligned} G(f) = \quad & \{g_{10}N_1 + g_{20}N_2\} + \\ & \textstyle\sum_{i\in\mathcal{S}:f_i\neq 0} r_i(f_i) + \sum_{i\in\mathcal{S}:f_i\neq 0} \sum_{i'\in\mathcal{S}-\{i\}:f_{i'}\neq 0} r_{i,i'}(f_i,f_{i'}) \end{aligned} \tag{10.35}$$

In the above, $g_{10} = Const_1 - v_{10}$, $g_{20} = Const_2 - v_{20}$, and $r_i(f_i)$ and $r_{i,i'}(f_i, f_{i'})$ are related to $V_1(d_1(i)|f_i)$ and $V_2(d_2(i,i')|f_i,f_{i'})$ by (9.26) and (9.27). These determine the compatibilities in RL. The gain is to be maximized.

The parameters g_{10} and g_{20} affect not only the NULL labels but also the local behavior of the maximization algorithm. When both are zero, there will be no NULL labels. The larger their values, the more sites will be assigned the NULL and the more serious the problem of local maxima becomes. In other words, we found that a labeling f in which $f_i = 0$ for some i, is likely a local optimum and that the labeling f in which $f_i = 0$ for all i, is a deep local optimum. This has motivated us to incorporate a heuristic annealing procedure into the labeling process, in a way similar to the graduated nonconvexity (GNC) algorithm (Blake and Zisserman 1987), to overcome the local optimum problem.

Introduce a temperature parameter into the gain

$$\begin{aligned} G(f|T) = \quad & \{g_{10}N_1 + g_{20}N_2\}/T + \textstyle\sum_{i\in\mathcal{S}:f_i\neq 0} r_i(f_i) \\ & + \textstyle\sum_{i\in\mathcal{S}:f_i\neq 0} \sum_{i'\in\mathcal{S}-\{i\}:f_{i'}\neq 0} r_{i,i'}(f_i,f_{i'}) \end{aligned} \tag{10.36}$$

with $G(f) = \lim_{T\to 1} G(f|T)$. T is initially set to a very high value $T^{(0)} \to \infty$ and gradually lowered toward $T^{(\infty)} = 1$. The maximum f^* obtained at $T^{(t-1)}$ is used as the initial value for the next new phase of maximization at $T^{(t)}$. In this way, the f^* are tracked from high T to low T. Because g_{10} and g_{20} are weighted by $\frac{1}{T}$, the optimum f^* obtained at high T is less affected by the local optimum problem; when it is tracked down, a better quality solution may be obtained than one found by a nonannealing algorithm. The improvement of

annealing labeling ICM, a procedure incorporating annealing labeling into ICM, over the simple ICM will be shown shortly.

Note that the T in the annealing labeling acts on only the prior part, whereas in SA, T acts on both the prior and likelihood parts. So the present annealing procedure is more like GNC (Blake and Zisserman 1987) than SA (Geman and Geman 1984).

10.4 Graph Cuts

Graph cuts is a class of algorithms that uses max-flow algorithms to solve discrete energy minimization problems. It was first proposed to obtain the global minimum of a two-label MRF model (Ising model) by Greig, Porteous, and Seheult (1989). It was then extended to solve convex multilabel problems (Roy and Cox 1998) and approximate the global solution for more general multilabel MRF problems (Boykov et al. 2001). Graph cuts has now been widely used in image analysis problems such as image segmentation, restoration, super-resolution, and stereo. This section introduces basic concepts of graph cuts. The reader is referred to the Website of Zabih () for more about the theories and applications.

10.4.1 Max-Flow

Max-flow (or min-cut or s-t cut) algorithms play a key role in graph cuts. There are two types of max-flow algorithms: augmenting paths algorithms (Ford-Fulkerson style) and push-relabel algorithms (Goldberg-Tarjan style). A comparison of these algorithms for energy minimization in image analysis can be found in (Boykov and Kolmogorov 2004).

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a digraph (directed graph) with nonnegative weight on each edge, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ the set of edges. In max-flow, $\mathcal{V}$ contains two special vertices, the source s and sink t. A cut $\mathcal{E}_c$ of $\mathcal{G}$ is a subset of $\mathcal{E}$, satisfying: (1) the subgraph excluding $\mathcal{E}_c$, $\tilde{\mathcal{G}} = (\mathcal{V}, \mathcal{E} - \mathcal{E}_c)$, is disconnected; (2) adding to this subgraph any edge in $\mathcal{E}_c$, giving $\tilde{\mathcal{G}} = (\mathcal{V}, (\mathcal{E} - \mathcal{E}_c) \cup \{e\})$ $(\forall e \in \mathcal{E}_c)$, the subgraph is connected. The cost of a cut $\mathcal{E}_c$ is the sum of its edge weights. The min-cut problem is to find the cut that has the minimum cost. According to the Ford-Fulkerson theorem, this problem is equivalent to find the maximum flowing from s to t.

Fig. 10.5 shows an example of max-flow. On each edge there is a pair of number (x, y), where x is the maximum flux permitted by this edge and y is the maximum flux in practice. The edges are cut by the dashed line and this forms the min-cut. The flux of this cut is equivalent to the maximum flux from the source s to the terminal t, which in this digraph is 11. The dashed line divides the nodes of the digraph into two parts, S and T, with all the nodes in S connected to the source s, and all the nodes in T connected to the terminal t. In a two-label MRF problem, this can be explained as all the

nodes in S having the same label ℓ_1 and all the nodes in T having the label ℓ_2.

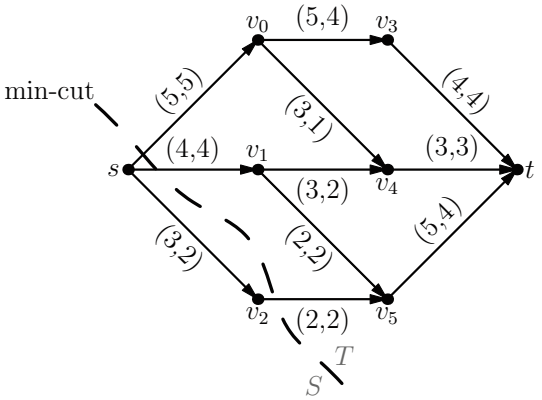


Figure 10.5: An example of max-flow.

10.4.2 Two-Label Graph Cuts

The simplest case of a labeling problem is binary labeling, where $\mathcal{L} = \{\ell_1, \ell_2\}$. If the prior of an MRF is defined as an Ising model, the exact optimum solution can be obtained by finding the max-flow on a special defined graph constructed according to the energy function; in other words, the minimum cut or the maximum flow of this graph is the same as the global minimum of the energy function (Greig et al. 1989).

A related concept for the max-flow is the submodularity. Let $\mathcal{S}$ be a finite set and $g : 2^{\mathcal{S}} \rightarrow \mathbb{R}$ be a real-valued function defined on the set of all subsets of $\mathcal{S}$. U is called submodular if for arbitrary $X, Y \subset \mathcal{S}$

$$g(X) + g(Y) \geq g(X \cup Y) + g(X \cap Y) \tag{10.37}$$

When the energy function is submodular, the following algorithm produces a graph that represents the energy function.

Introduce two auxiliary vertices s and t (source and sink), and let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the digraph to be constructed, with $\mathcal{V} = \mathcal{S} \cup \{s, t\}$. The following describes how to construct the edge set $\mathcal{E}$. This can be done in two parts.

For the closeness terms $V_1(i, f_i), \forall i \in \mathcal{S}$, according to the definition of submodularity, all functions with only one variable are submodular. Therefore, an energy function with the V_1 term only is submodular. If $V_1(i, 1) > V_1(i, 0)$, then add an edge $< s, i >$ with the weight $w_{s,i} = V_1(i, 1) - V_1(i, 0)$; otherwise, add an edge $< i, t >$ with the weight $w_{i,t} = V_1(i, 0) - V_1(i, 1)$. This can be

done using a procedure $edge(i, \ell)$: if $\ell > 0$, then add an edge $< s, i >$ with the weight ℓ. Otherwise, add an edge $< i, t >$ with the weight $-\ell$; see Fig. 10.6.

The smoothness term V_2 is submodular if it satisfies the following condition:

$$V_2(i, i', 0, 1) + V_2(i, i', 1, 0) \geq V_2(i, i', 0, 0) + V_2(i, i', 1, 1) \quad \forall i, i' \in \mathcal{S} \tag{10.38}$$

The digraph for representing a smoothness term $V_2(i, i', f_i, f_{i'})$ can be constructed as follows (see also Fig. 10.6):

- $edge(i, V_2(i, i', 1, 0) - V_2(i, i', 0, 0))$;
- $edge(i', V_2(i, i', 1, 1) - V_2(i, i', 1, 0))$;
- add an edge $< i, i' >$ with the weight $w_{i,i'} = V_2(i, i', 0, 1) + V_2(i, i', 1, 0) - V_2(i, i', 0, 0) - V_2(i, i', 1, 1)$.

A digraph is thus constructed for a submodular energy function. A max-flow algorithm can then be applied to this graph to find the optimum. The necessary condition for finding the exact optimum of binary labeling in polynomial time is that the energy function be submodular (Kolmogorov and Zabih 2004). If the energy function can be expressed as $V_{\mathcal{C}_1}(f) + V_{\mathcal{C}_2}(f) + V_{\mathcal{C}_3}(f)$, this is also the sufficient condition (Kolmogorov and Zabih 2004). Methods exist for constructing digraphs for nonsubmodular energy functions; see (Kolmogorov and Rother 2007).

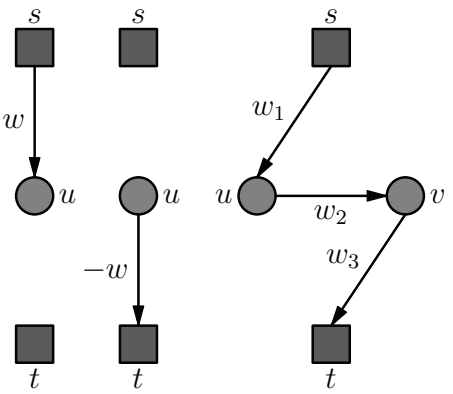


Figure 10.6: Construction of edges of a digraph for the closeness term (left and middle) and the smoothness term (right) of a submodular energy function.

10.4.3 Multilabel Graph Cuts

Minimization of a convex energy function with multiple discrete labels can be solved exactly. Assuming that the discrete label set is ordered (such as

$1 < 2 < \ldots < M$ as the depths for stereo vision (Roy and Cox 1998)) and the smoothness term is convex, then the original multilabel problem can be transformed into a two-label one for a graph with an augmented set of nodes (sites) and solved using a two-label graph cut method (Roy and Cox 1998). This finds a global solution.

Minimization of a more general energy function with multiple discrete labels is an NP-hard problem. Numerous algorithms have been proposed to approximate the global solution with a lower complexity. Such an approximation could be done using graph cuts by transforming the original problem to an equivalent problem of multiway cuts of a special graph constructed according to the primal minimization problem (Boykov et al. 1998). The two algorithms presented in (Boykov et al. 2001), α–β swap and α-expansion, use methods of iteratively solving multiple two-label graph cuts and achieve better computational performance. It is shown that the α-expansion algorithm has a linear time complexity w.r.t. the number of labels ($O(M)$), whereas α–β has an $O(M^2)$ complexity. The α-expansion can be further improved to achieve a logarithmical time complexity (Lempitsky et al. 2007). The following introduces the α-expansion and α–β swap algorithms.

α-expansion

The α-expansion algorithm (Boykov et al. 2001) assumes that the smoothness prior term V_2 is a metric, that is

- $V_2(f_i, f_{i'}) \geq 0, V_2(f_i, f_{i'}) = 0 \Leftrightarrow f_i = f_{i'}$,
- $V_2(f_i, f_{i'}) = V_2(f_{i'}, f_i)$,
- $V_2(f_i, f_{i'}) \leq V_2(f_i, f_{i''}) + V_2(f_{i''}, f_{i'})$.

such that the submodularity condition (10.37), which is weaker than the aforementioned convexity condition, is satisfied. Each iteration considers an $\alpha \in \mathcal{L}$ value and the two-label set of $\{\alpha, \text{non-}\alpha\}$, and solves the two-label graph cut problem. The algorithm decides whether or not a site $i \in \mathcal{S}, f_i \neq \alpha$ needs to change from non-α to α. The iteration continues until the energy does not decrease any more; see Fig. 10.7. The energy of the approximate solution is upper-bounded by cU^*, where U^* is the global minimum of the original energy function and $c \geq 1$ is a constant that depends on the original energy function.

α–β Swap

The α–β Swap (Boykov et al. 2001) is an iterative graph cut algorithm applicable to situations when the smoothness term V_2 is semimetric, i.e.,

- $V_2(f_i, f_{i'}) \geq 0, V_2(f_i, f_{i'}) = 0 \Leftrightarrow f_i = f_{i'}$;
- $V_2(f_i, f_{i'}) = V_2(f_{i'}, f_i)$;

Begin
1. Initialize f with an arbitrary labeling;
2. Flag: $=$ *false*;
3. For each label $\alpha \in \mathcal{L}$;
   3.1 Find $\hat{f} = \arg\min E(f')$ among f' within one α expansion of f;
   3.2 If $E(\hat{f}) < E(f)$, set $f := \hat{f}$ and Flag: $=$ *true*;
4. If Flag $=$ *true* goto step 2
5. Return f;

End

Figure 10.7: α-expansion Algorithm

It chooses two labels $\alpha, \beta \in \mathcal{L}$ in each iteration. Let $\mathcal{S}_\alpha = \{i \in \mathcal{S} \mid f_i = \alpha\}$ and $\mathcal{S}_\beta = \{i \in \mathcal{S} \mid f_i = \beta\}$. A two-label graph cut algorithm is used to determine two subsets, $\hat{\mathcal{S}}_\alpha \subset \mathcal{S}_\alpha$ and $\hat{\mathcal{S}}_\beta \subset \mathcal{S}_\beta$, that need to swap the labels. The iteration continues until the energy function stops decreasing; see Fig. 10.8.

Begin
1. Initialize f with an arbitrary labeling;
2. Flag: $=$ *false*;
3. For each pair $\{\alpha, \beta\} \in \mathcal{L}$;
   3.1 Find $\hat{f} = \arg\min E(f')$ among f' within one $\alpha - \beta$ swap of f;
   3.2 If $E(\hat{f}) < E(f)$, set $f := \hat{f}$ and Flag: $=$ *true*;
4. If Flag $=$ *true* goto step 2
5. Return f;

End

Figure 10.8: α–β swap algorithm

10.5 Genetic Algorithms

Genetic algorithms (GAs) (Holland 1975; Goldberg 1989) are another class of heuristic procedures for global optimization. Inspired by the principle of natural evolution in the biological world, the procedures simulate the evolutionary process in which a population of individuals who have the highest goodness-of-fit values survives.

generate initial population $P = \{f^1, \ldots, f^n\}$;
compute $E(f)$ $\forall f \in P$;
repeat
    select two individuals from P;
    recombine the individuals to produce two offsprings;
    compute $E(f)$ of the offsprings;
    update P with the offsprings;
until (converged);
return $f = \arg\min_{f \in P} E(f)$;

Figure 10.9: A standard genetic algorithm.

10.5.1 Standard GA

Figure 10.9 illustrates a standard genetic algorithm for unconstrained, combinatorial optimization.[1] Initially, a number of (say, $n = 50$) individuals are *generated*, yielding a population P. An individual is determined by his chromosome f, which is a string of m genes (labels) f_i $(i = 1, \ldots, m)$.

At a point of the evolution, two individuals are randomly *selected* for mating. The selection is done according to a scheme that favors the fitter individuals (with lower E value); the lower the value of $E(f)$, the more likely f will be selected. For example, f may be selected with a probability proportional to $1 - \frac{E(f)}{\sum_{f \in P} E(f)}$ or a monotonically increasing function of it, assuming $E(f)$ is nonnegative. There are numerous selection schemes (Goldberg 1989).

Recombination takes place between the two individuals selected. The mechanisms of *crossover* and *mutation* are typically used for this. Figure 10.10 illustrates two such basic operations in which a gene can take any alphanumeric value. Crossover takes two individuals, cuts their chromosome strings at some randomly chosen position, and recombines the opposite segments to create two offsprings. Crossover is not always invoked; it is applied with a probability p_c typically between 0.6 and 1. If it is not applied, the offsprings are simply the duplicates of the selected individuals. Mutation is applied to each offspring after the crossover. It randomly alters a randomly chosen gene with a small probability p_m, typically 0.001. As is usual in GA practice, there are many crossover and mutation operators.

The offsprings are then added to P, and the two least fit individuals (i.e., those with the highest $E(f)$ values) are removed from the population P. As the evolution continues in this way, the fitness of the best individual as

[1] Special coding of solutions into chromosomes needs to be done when GA is used for (piecewise) continuous optimization problems.

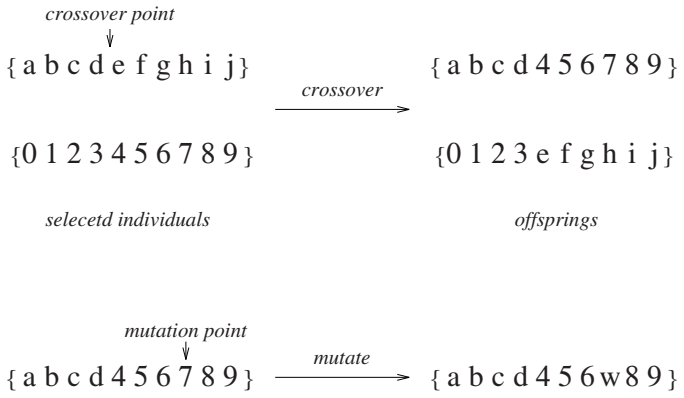


Figure 10.10: The two basic operations in recombination: crossover and mutation.

well as the average fitness increases toward the global optimum. *Convergence* is judged by uniformity: A gene is said to have converged when most (say 95%) of the population share the same value. The population is said to have converged when all the genes have converged. The convergence of GAs is obtained, but no theoretical proof is given.

Impressive empirical results for solving real and combinatorial, unconstrained and constrained optimization problems, such as the traveling salesman problem, and neural network optimization and scheduling are reported (see (Goldberg 1989) for a review). Applications in computer vision are also seen (Bhanu et al. 1989; Ankenbrandt et al. 1990; Hill and Taylor 1992). Currently, there is no theory that explains why GAs work. However, some hypotheses exist. Among these are the schema theorem (Holland 1975) and building block hypothesis (Goldberg 1989).

10.5.2 Hybrid GA: Comb Algorithm

Combining a local search with a GA yields a hybrid GA, also called a memetic algorithm (Moscato 1989; Radcliffe and Surry 1994). Here, a new random search method, called the Comb method, is described for combinatorial optimization. Assume that an energy function has been given that is formulated based on the MRF theory for image restoration and segmentation. The Comb method maintains a number of best local minima found so far, as a population based method. It uses the common structure of the local minima to infer the structure of the global minimum. In every iteration, it derives one or two new initial configurations based on the Common structure (common

labels) of the Best local minima (hence "Comb"): If two local minima have the same label (pixel value) in a pixel location, the label is copied to the corresponding location in the new configuration; otherwise, a label randomly chosen from either local minimum is set to it. The configuration thus derived contains about the same percentage of common labels as the two local minima (assuming the two have about the same percentage of common labels). But the configuration derived is no longer a local minimum, and thus further improvement is possible. The new local minimum then updates the best existing ones. This process is repeated until some termination conditions are satisfied.

The resulting Comb algorithm is equivalent to a GA hybridized with steepest descent, in which the Comb initialization therein works like a uniform crossover operator. There have been various interpretations for the crossover operation. The idea of encouraging common structures in the Comb initialization provides a new perspective for interpreting the crossover operation in GA.

Experimental results in both image restoration and segmentation are provided to compare the Comb method with the ICM, HCF (Chou et al. 1993) and SA. The results show that the Comb yields solutions of better quality than the ICM and HCF and comparable to SA.

The Comb method maintains a number N of best local minima found so far, denoted $F = \{f^{[1]}, \ldots, f^{[N]}\}$, as a population based method. In every iteration, it derives a new initial configuration from F and performs steepest descent using the initial configurations derived. If the local minimum found is better than an existing one in F, it replaces it.

Ideally, we desire that all local minima in F converge towards the global minimum $f^{[global]}$, in which case, there must be

$$f_i^{[n]} = f_i^{[global]} \qquad 1 \le n \le N \tag{10.39}$$

for all $i \in \mathcal{S}$. We call $f_i^{[global]}$ the *minimal label* at i. To achieve (10.39), all the labels at i, $\{f_i^{[n]} | \forall n\}$, should finally converge to the minimal label $f_i^{[global]}$. The Comb is performed with this objective.

The following heuristic is the basis for deriving new initial configurations. Although $f^{[1]}, \ldots, f^{[N]}$ are local minima, they share some structure with the global minimum $f^{[global]}$. More specifically, some local minima $f^{[n]}$ have the minimal label $f_i^{[n]} = f_i^{[global]}$ for some $i \in \mathcal{S}$. Figure 10.11 shows the (approximate) global minimum for an MAP-MRF restoration problem and some local minima found by using the multi-start method with initially random configurations. A statistic over a number of $N = 10$ local minima is made to see how many minimal labels they have. Table 10.1 shows the statistic in terms of the percentile of the sites (pixels) $i \in \mathcal{S}$ at which at least k local minima $f^{[n]}$ have the minimal label $f_i^{[n]} = f_i^{[global]}$.

The Comb initialization is aimed at deriving configurations having a substantial number of minimal labels so as to improve F toward the objective

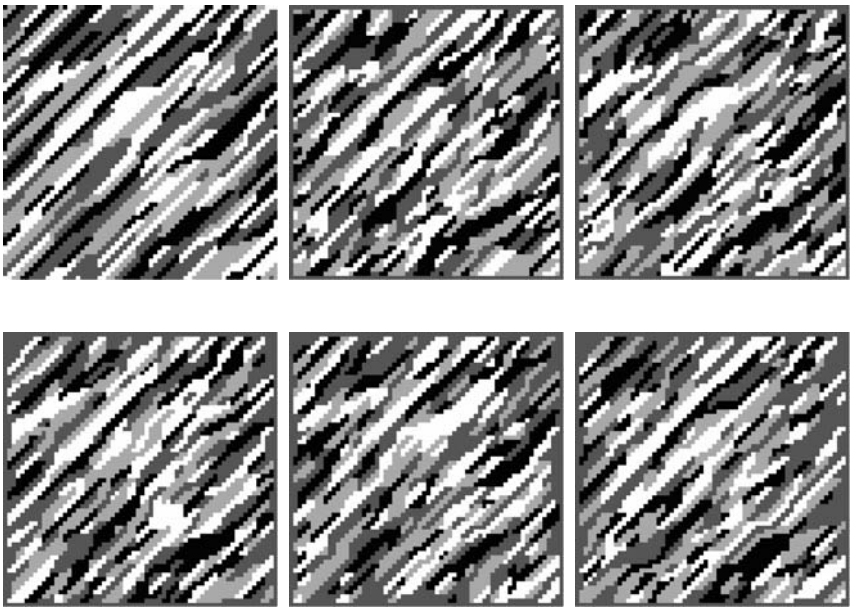


Figure 10.11: The global minimum (upper left) and five local minima. The local minima share some structure with the global minimum.

of (10.39). Although a configuration with a larger number of minimal labels does not necessarily have a lower-energy value, we hope that it provides a good basis to start with (i.e., it can serve as a good initial configuration).

Table 10.1: Percentile (rounded up to integers) of the sites (pixels) $i \in \mathcal{S}$ at which *at least* k local minima $f^{[n]}$ have the same label $f_i^{[n]} = f_i^{[global]}$ as the global minimum $f^{[global]}$.

k	0	1	2	3	4	5	6	7	8	9	10
%	100	98	95	87	75	60	43	28	16	7	2

The Comb algorithm is described in Fig. 10.12. The initialization at the beginning of the Comb algorithm is done according to a uniform distribution like the multi-start method. This is followed by iterations of four steps. First, two local minima in F, $f^{[a]}$ and $f^{[b]}$, $(a \neq b)$, are randomly selected according to a uniform distribution. Second, a new initial configuration $f^{[0]}$ is derived from $f^{[a]}$ and $f^{[b]}$ using the standard Comb initialization, which will be

comb_initialization($f^{[a]}$, $f^{[b]}$, F)
begin
for (each $i \in \mathcal{S}$) do:
if ($f_i^{[a]} == f_i^{[b]}$ && $rand[0,1] < 1 - \tau$)
then $f_i^{[0]} = f_i^{[a]}$;
else
$f_i^{[0]} = rand(\mathcal{L})$;
end

Comb_Algorithm
begin
initialize the set $F = \{f^{[1]}, \ldots, f^{[N]}\}$;
do {
random_selection($f^{[a]}$, $f^{[b]}$, F);
comb_initialization($f^{[0]}$, $f^{[a]}$, $f^{[b]}$);
steepest_descent(f^*, $f^{[0]}$);
update(F, f^*);
} until (termination condition satisfied);
return(arg $\min_{f \in F} E(f)$);
end

Figure 10.12: The Comb algorithm.

explained shortly. Then, steepest descent is applied to $f^{[0]}$ to produce a local minimum f^*. Finally, the set F is updated by f^*: If $E(f^*) < \max_{f \in F} E(f)$, then the configuration arg max$\{f | f \in F\}$, which has the highest energy value, higher than $E(f^*)$, is replaced by f^*. The termination condition may be that all configurations in F are the same or that a certain number of local minima have been performed. The algorithm returns the best local minimum in F (i.e., the one having the lowest energy).

The central part of the Comb method is the derivation of new initial configurations. The Comb is aimed at deriving $f^{[0]}$ in such a way that $f^{[0]}$ contains as many minimal labels as possible. Because the minimal labels are not known a priori, the Comb attempts to use common structure, or *common labels*, of $f^{[a]}$ and $f^{[b]}$ to infer the minimal labels. We say that f_i^{comm} is a common label of $f^{[a]}$ and $f^{[b]}$ if $f_i^{comm} = f_i^{[a]} = f_i^{[b]}$. The Comb makes a hypothesis that f_i^{comm} is a minimal label if $f_i^{[a]} = f_i^{[b]}$. The Comb initialization schemes are illustrated as follows:

1. The *basic Comb* initialization. For each $i \in \mathcal{S}$, if $f_i^{[a]}$ and $f_i^{[b]}$ are identical, then set $f_i^{[0]} = f_i^{[a]}$; otherwise set $f_i^{[0]} = rand(\mathcal{L})$, which is a label randomly drawn from $\mathcal{L}$. The use of the common label in the initial configuration encourages the enlargement of common structure in the local minimum to be found subsequently.

2. The *standard Comb* initialization. The basic Comb initialization is accepted with a probability $1 - \tau$, where $0 < \tau < 1$. The probabilistic acceptance of common labels diversifies the search and prevents F from converging to a local minimum too soon. The standard Comb initialization is shown in the upper part of Fig. 10.12, where $rand[0, 1]$ stands for an evenly distributed random number in $[0, 1]$.

Then, how many minimal labels are there in $f^{[0]}$ as the result of copying common labels(i.e., as the result of inferring minimal labels using common labels)? In supervised tests where the (near) global minimum is known, we find that the percentage of minimal labels in $f^{[0]}$ is usually only slightly (about 1.0–2.0%) lower than those in $f^{[a]}$ and $f^{[b]}$. That is, the number of minimal labels retained in $f^{[0]}$ is about the same as those in $f^{[a]}$ and $f^{[b]}$. Given this and that $f^{[0]}$ is no longer a local minimum like $f^{[a]}$ and $f^{[b]}$, there is room to improve $f^{[0]}$ using a subsequent local minimization. This makes it possible to yield a better local minimum from $f^{[0]}$.

There are two parameters in the Comb algorithm, N and τ. The solution quality increases (i.e., the minimized energy value decreases) as the size of F, N, increases from 2 to 10, but remains about the same (probabilistically) for greater N values; and a larger N leads to more computational load. Therefore, we choose $N = 10$. Empirically, when $\tau = 0$, the algorithm converges sooner or later to a unique configuration, and choosing a smaller N makes such a convergence quicker. But $\tau = 0$ often gives a premature solution. The value of $\tau = 0.01$ is empirically a good choice.

The Comb algorithm corresponds to a hybrid GA as described in Fig. 10.13. The standard Comb initialization is effectively the same as a crossover operation followed by a mutation operation, the major and minor operations in genetic algorithms (GA) (Goldberg 1989). More exactly,

- the basic Comb corresponds to uniform crossover and
- the probability acceptance in the standard Comb corresponds to mutation.

In GA, two offspring, $f_i^{[01]}$ and $f_i^{[02]}$, are produced as the result of crossover. In the uniform crossover, either of the following two settings are accepted with equal probability:

(i) $f^{[01]} = f_i^{[a]}$ and $f^{[02]} = f_i^{[b]}$,

(ii) $f^{[01]} = f_i^{[b]}$ and $f^{[02]} = f_i^{[a]}$.

So, if $f_i^{[a]} = f_i^{[b]}$, there must be $f_i^{[01]} = f_i^{[02]} = f_i^{[a]}$, just as in the Comb initialization. This encourages common labels because common labels are copied to the new initial configurations; in contrast, noncommon labels are subject to swap. The discussion above is about the uniform crossover. It should be noted that even the simplest one-point crossover also works in a way that encourages common labels.

The above suggests that the essence of both the Comb and GA is captured by using common structures of local minima. This is supported by the fact that the original Comb and the GA-like Comb yield comparable results: In the GA-like Comb algorithm (Fig. 10.13), when $f_i^{[a]} \neq f_i^{[b]}$, $f_i^{[01]}$ and $f_i^{[02]}$ inherit the values of $f_i^{[a]}$ and $f_i^{[b]}$, as does a crossover operator. However, setting $f_i^{[01]}$ and $f_i^{[02]}$ to a random label $rand(\mathcal{L})$ (i.e., not necessarily inheriting $f_i^{[a]}$ and $f_i^{[b]}$) leads to comparable results as long as the common labels are copied to $f_i^{[0]}$ when $f_i^{[a]} = f_i^{[b]}$. Moreover, whether to derive one initial configuration $f^{[0]}$ or two initial configurations $f^{[01]}$ and $f^{[02]}$ does not matter; both schemes yield comparable results. In summary, the Comb and the GA-like Comb produce comparable results, and this suggests that retaining common labels is important and provides an interpretation for the crossover operation in GA.

The Comb is better than the multi-start method. Running a steepest descent algorithm a number of times using the Comb initialization gets a better solution than running it the same number of times using the independent initialization of multi-start. The Comb has a much higher efficiency in descending to good local minima because it makes use of the best local minima.

To summarize, the Comb attempts to derive good initial configurations from the best local minima found so far in order to achieve better solutions. To do so, it uses the common structure of the local minima to infer label values in the global minimum. An initial configuration thus derived has about the same number of minimal labels as the two local minima from which it is derived. However, it is no longer a local minimum and thus its quality can be improved by a subsequent local minimization. This makes it possible to yield a better local minimum and thus increments the solution quality step by step. The comparison shows that the Comb produces better results than the ICM and HCF though at higher computational cost, and results comparable to the SA at lower cost (Section 10.6.3). This suggests that the Comb can provide a good alternative to the well-known global minimizer SA. Further, the Comb algorithm is applicable, in principle, to many optimization problems of vision and pattern recognition.

```
GA_comb_initialization(f^[a], f^[b], F)
begin
        for each i ∈ S
            /* uniform crossover */
            if (f_i^[a] == f_i^[b])
                then f_i^[01] = f_i^[02] = f_i^[a];
            else if (rand[0,1] < 0.5)
                f_i^[01] = f_i^[a] and f_i^[02] = f_i^[b];
            else
                f_i^[01] = f_i^[b] and f_i^[02] = f_i^[a];

            /* mutation */
            if (rand[0,1] < τ)
                f_i^[01] = rand(L);
            if (rand[0,1] < τ)
                f_i^[02] = rand(L);
end
```

```
GA_Comb_Algorithm
begin
        initialize F = {f^[1], ..., f^[N]};
        do {
            random_selection(f^[a], f^[b], F);
            GA_comb_initialization(f^[01], f^[02], f^[a], f^[b]);
            steepest_descent(f^*1, f^[01]);
            steepest_descent(f^*2, f^[02]);
            update(F, f^*1, f^*2);
        } until (termination condition satisfied);
        return(arg min_{f∈F} E(f));
end
```

Figure 10.13: A GA-like Comb algorithm.

10.6 Experimental Comparisons

10.6.1 Comparing Various Relaxation Labeling Algorithms

In the following experiments, we compare several algorithms for combinatorial minimization and for constrained minimization:

1. ICM of Besag (1986),
2. RL of Hummel and Zucker (1983),
3. RL of Rosenfeld, Hummel, and Zucker (1976),
4. MF annealing of Peterson and Soderberg (1989),
5. SA of Geman and Geman (1984),
6. Annealing ICM (Section 10.3.2).

The comparison is in terms of (1) the solution quality measured by the maximized gain $G(f^*)$, (2) the computational cost measured by the number of iterations required and (3) the need for heuristics to tune the algorithm.

The schedules for the annealing algorithms are as follows. There are two annealing schedules for MFA. In the first schedule, which is given in (Peterson and Soderberg 1989), T is decreased according to $T^{(t+1)} \leftarrow 0.99T^{(t)}$; we set the initial temperature as $T^{(0)} = 10^9 \times M \times m$. The second is also in the form $T^{(t+1)} \leftarrow \kappa^{(t+1)}T^{(t)}$, but $\kappa^{(t+1)}$ is chosen on an ad hoc basis of trial and error to get the best result, which is

$$\kappa^{(t+1)} = \begin{cases} 0.9 & \text{if } S < 0.9 \; rmand \; \kappa^{(t)} \leq 0.9 \\ 0.95 & \text{if } S < 0.95 \; rmand \; \kappa^{(t)} \leq 0.95 \\ 0.99 & \text{otherwise} \end{cases} \tag{10.40}$$

where S is the saturation defined in (10.23). We refer to these two schedules as MFA-1 and MFA-2, respectively. For the SA, it is $T^{(t+1)} \leftarrow 0.99T^{(t)}$ but with the initial temperature set to $10^9 \times M \times m$; annealing stops if the desired number of acceptances is not achieved at 100 successive temperatures (Kirkpatrick et al. 1983). These are purposely tuned for the best result possible. The schedule for the annealing ICM is quite tolerant; it is chosen as $T^{(t+1)} \leftarrow 0.9T^{(t)}$ with $T^{(0)} = 10$.

The initial labeling is assigned as follows. First, we set $p_i^{(0)}(I) = 1 + 0.001 * rnd \; (\forall i, I)$ as the start point common to all the algorithms compared, where rnd is a random number evenly distributed between 0 and 1. For the continuous algorithms (2, 3, and 4), we normalized $p^{(0)}$ to satisfy (9.19). For the discrete algorithms (1, 5, and 6), i is initially labeled according to maximal selection, $f_i^{(0)} = I^* = \arg\max_I p_i^{(0)}(I)$. The convergence of the continuous algorithms is judged by checking whether the saturation, S, is larger than 0.99.

The test bed is the MRF matching of weighted graphs. Here, a node corresponds to a point in the $X - Y$ plane. Two random graphs, each containing $m = M$ nodes, are randomly generated as follows. First, a number of $\lfloor \frac{2}{3}m \rfloor$ nodes, where $\lfloor \cdot \rfloor$ is the "floor" operation, are generated using random numbers uniformly distributed within a box of size 100×100. They are for the first graph. Their counterparts in the other graph are generated by adding

Gaussian noise of $N(0, \sigma^2)$ to the x and y coordinates of each of the nodes in the first graph. This gives $\lfloor \frac{2}{3}m \rfloor$ deviated locations of the nodes. Then, the rest of the $\lceil \frac{1}{3}m \rceil$ nodes in each of the two graphs are generated independently by using random numbers uniformly distributed in the box, where $\lceil \cdot \rceil$ is the "ceiling" operation. This simulates outlier nodes.

The steps above generate two weighted graphs, $\mathcal{G} = (\mathcal{S}, d)$ and $\mathcal{G}' = (\mathcal{L}, D)$, where $\mathcal{S} = \{1, \ldots, m\}$ and $\mathcal{L} = \{1, \ldots, M\}$ index the sets of nodes and $d = [d(i, i')]_{i,i' \in \mathcal{S}}$ and $D = [D(I, I')]_{I,I' \in \mathcal{L}}$ are the distances between nodes in the first and second graphs, respectively. The weights in d and D, which reflect bilateral relations between nodes, are the basis for matching, whereas unary properties of nodes are not used in the test. We augment $\mathcal{L}$ by a special node, called the NULL node and indexed by 0, into $\mathcal{L}' = \{0, 1, \ldots, M\}$. The purpose is to cater for the matching of outlier nodes. Refer also to Section 4.1 for the graph representation. The matching is to assign a node from $\mathcal{L}'$ to each of the nodes in $\mathcal{S}$ so that some goodness (cost) of matching is maximized (minimized).

The posterior energy formulated in Section 4.2 is used to measure the cost of a matching $f = \{f_1, \ldots, f_m\}$. To be suitable for real, as opposed to combinatorial, computation, the problem is then reformulated in terms of relaxation labeling (see Section 9.3.2). From the two weighted graphs, the compatibility matrix $[r_{i,i'}(I, I')]$ is defined as

$$r_{i,i'}(I, I') = \begin{cases} Const_2 - [d(i, i') - D(I, I')]^2 & \text{if } I' \neq 0 \text{ and } I' \neq 0 \\ Const_2 - v_{20} & \text{otherwise} \end{cases} \tag{10.41}$$

where $v_{20} > 0$ is a constant. The gain function is then

$$G(f) = \sum_{i \in \mathcal{S}} \sum_{I \in \mathcal{L}'} \sum_{i' \in \mathcal{S}, i' \neq i} \sum_{I' \in \mathcal{L}'} r_{i,i'}(I, I')\ p_i(I)\ p_{i'}(I') \tag{10.42}$$

To reduce storage, we used only one byte (8 bits) to represent the compatibility coefficients $r_{i,i'}(I, I')$. Positive values are truncated to an integer between 0 and 255 while negative values are truncated to zero. In this case, we set $Const_2 = 255$. Although the precision of the compatibility coefficients is low, good results are still obtained; this demonstrates the error-tolerant aspect of the RL minimization approach.

The purpose of the MRF and RL formulations is to provide the compatibility matrix. Our ultimate objective here is more abstract: to compare the ability of the algorithms to solve the minimization problem expressed, given the posterior probability function or the corresponding compatibility matrix.

The following results are obtained after 200 runs. Figure 10.14 illustrates the solution quality in terms of the maximized gain as a function of the noise level σ. SA provides the best result when carefully tuned, followed by the Hummel-Zucker algorithm, whereas the quality of the ICM solution is the poorest. The annealing procedure (see Section 10.3.2) significantly improves the quality of the ICM solution.

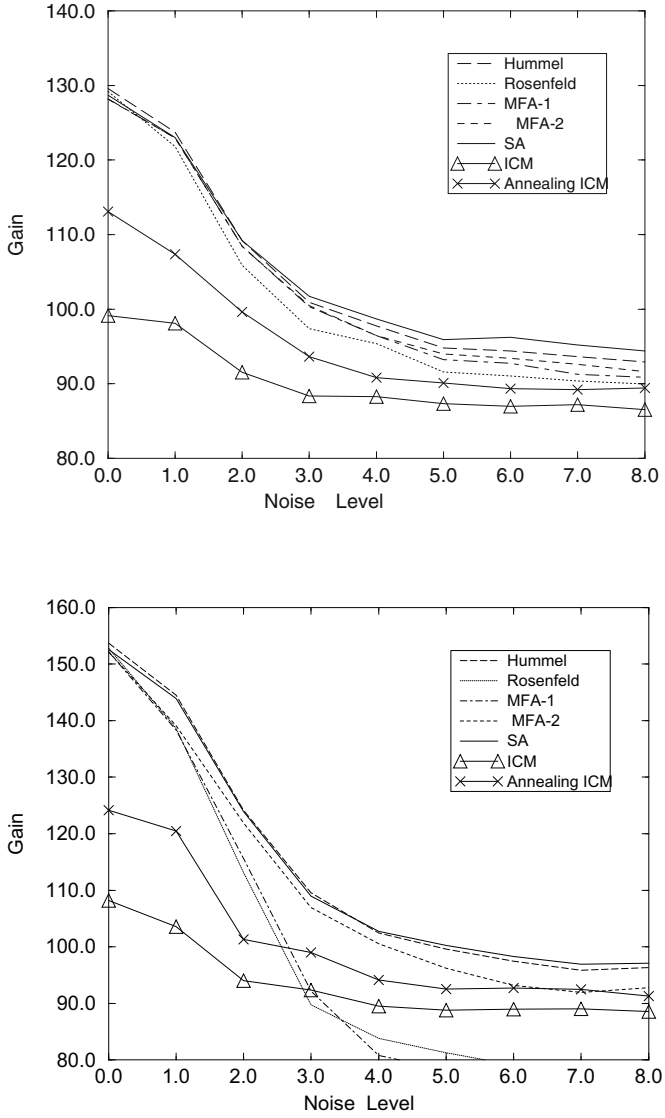


Figure 10.14: The maximized gain $G(f^*)$ (after dividing by m^2) for $m = 10$ (top) and $m = 20$ (bottom); the higher, the better.

The performance of MFT annealing is after the Hummel-Zucker algorithm when $m = M = 10$. However, for "harder" problems (e.g., when $m = M = 20$) the results produced by using the simple schedule MFA-1 deteriorate quickly as the noise level σ increases, dropping to even lower than that of ICM. For large m and M and high σ^2 values, the algorithm is often trapped in the all-NULL solution, which is a significant local optimum. We also found that a lower schedule (i.e., smaller κ in $T \leftarrow \kappa T$) does not necessarily yield better solutions.

Figure 10.15 demonstrates the cost in terms of the number of iterations as a function of the noise level σ. ICM converges very fast, after just a few iterations. In contrast, the number for SA with the specified heuristic schedule is two to four orders higher than all the deterministic algorithms. Figure 10.16 shows the efficiency measured by the maximized-gain/iteration-number ratio. The ordering by the efficiency is roughly consistent with the inverse ordering by the iteration number.

Besides the solution quality and the cost, an important factor that must be taken into account is the need for, and difficulties in, tuning the annealing schedule. It is well known that the schedule is critical to the success of SA, and it is an area of study (Aarts 1989). We add that the schedule is also critical to the mean field annealing. How to choose an optimal schedule depends not only on the type of the problem, but also on the size. Finding a good heuristic schedule based on trial and error can be tedious.

The comparison leads to a conclusion in favor of the Hummel-Zucker algorithm. It yields good-quality solutions quite comparable with the time-consuming simulated annealing; yet, it is much more efficient than SA, thus balancing well between quality and cost. Furthermore, it avoids the cumbersome needs for the heuristic tuning of annealing schedules in annealing algorithms. MFT annealing would also be a good choice if the rapid deterioration of solutions to "harder" problems could be remedied. An experimental result (Li 1995c) shows that the Lagrange-Hopfield algorithm described in Section 9.5 yields solutions of quality similar to those produced by the Hummel-Zucker algorithm.

An earlier experimental comparison of several RL algorithms in terms of the number of iterations (cost) and the number of errors (quality) was done by Price (1985). Price concluded that the algorithm of Faugeras and Price (1981) is the best, that of Hummel and Zucker (1983) is about as good, that by Peleg (1980) converges too fast to yield a good result, and that of Rosenfeld, Hummel, and Zucker (1976) performs only adequately. The goodness of interpretation in (Price 1985) relates not only to algorithms themselves but also to how the problem is formulated (e.g., the definition of compatibilities), which is a combination of the two issues, problem formulation and computational algorithm. Here, we measure the solution quality by the quantitative gain $G(f^*)$ and regard RL as just a mechanism of minimization rather than

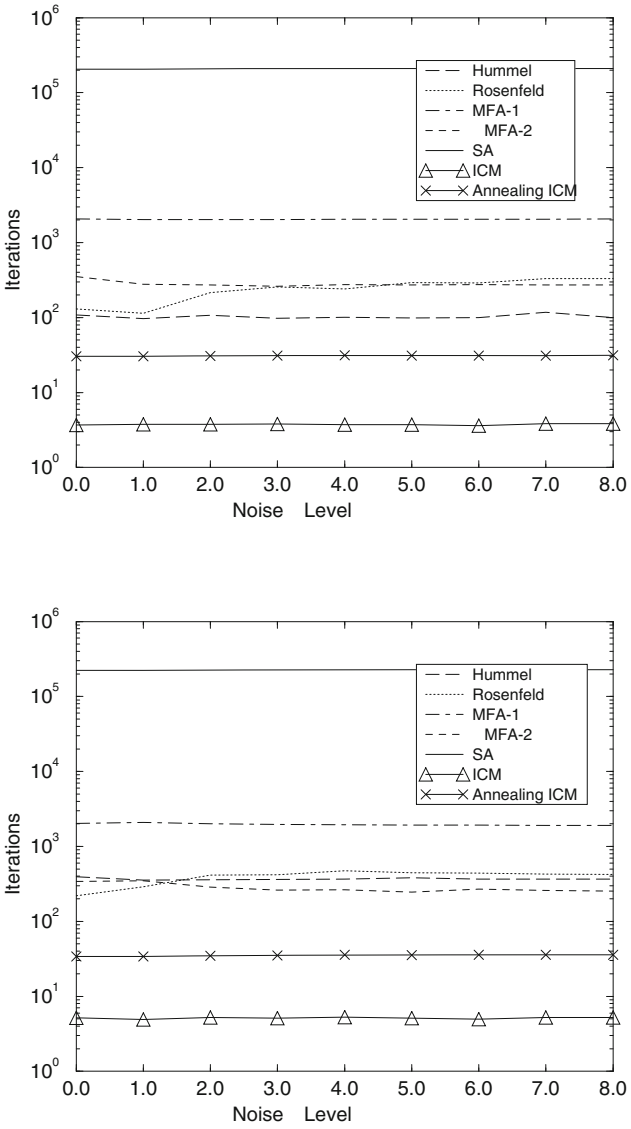


Figure 10.15: The number of iterations for $m = 10$ (top) and $m = 20$ (bottom); the lower, the better.

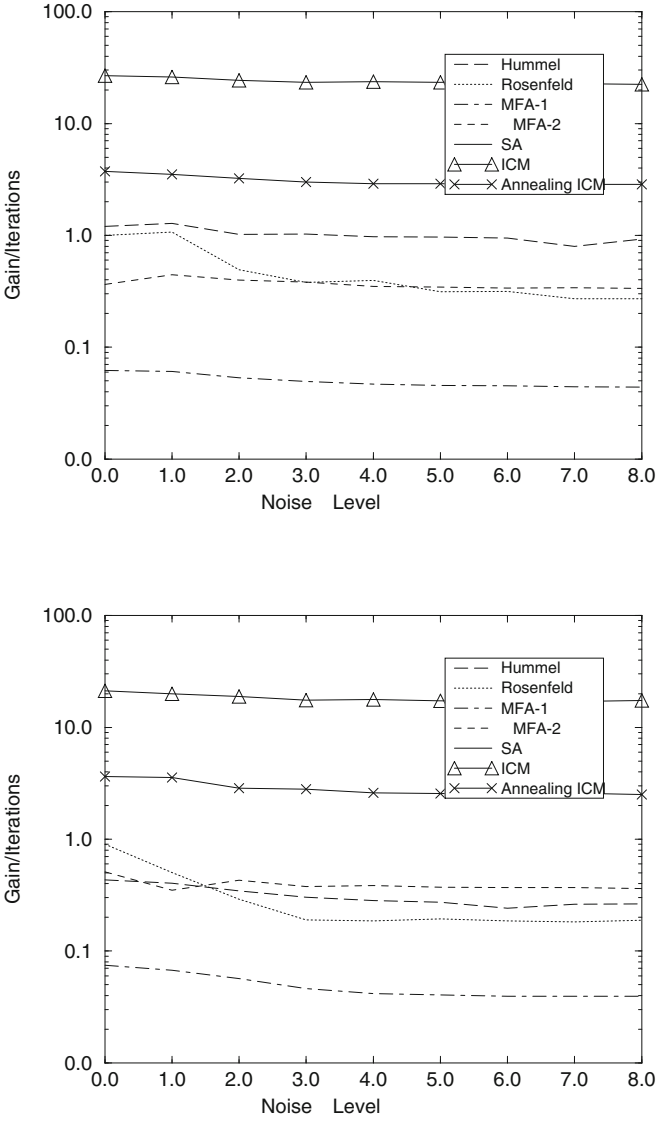


Figure 10.16: The efficiency measured by the maximized-gain/iteration-number ratio for $m = 10$ (top) and $m = 20$ (bottom); the higher, the better. The efficiency of SA is near zero.

one of interpretation. In our comparison, our interest is in how well an algorithm can minimize an energy regardless of the problem formulation.

10.6.2 Comparing the ALH Algorithm with Others

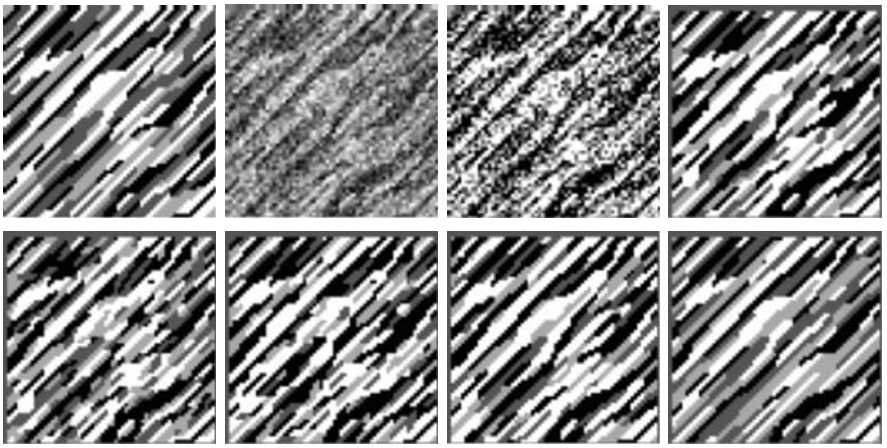
In the following, two experiments are presented, one for MAP-MRF image restoration and the other for segmentation, to compare the performance of the following algorithms: (1) ALH of this chapter, (2) ICM (Besag 1986), (3) HCF (Chou et al. 1993) (the parallel version), (4) MFA (Peterson and Soderberg 1989; Yuille and Kosowsky 1994) (see Section 10.2), and (5) SA with the Metropolis sampler (Kirkpatrick et al. 1983) implemented based on a procedure given in (Press et al. 1988). The comparison is in terms of (i) the solution quality measured by the minimized energy values and (ii) the convergence rate measured by the number of iterations. In calculating the energy, $E(f)$ of (9.94) is used where, for the continuous algorithms of ALH and MFA, f is obtained from p by a maximum selection (winner-take-all) operation.

For ALH, $\mu = 100$ and $T = 10^5$ are fixed, and β is increased from 1 to 100 according to $\beta \leftarrow 1.01\beta$. The convergence criterion is $\|u^{(t)} - u^{(t-1)}\|_\infty < 0.0001$. The schedule for SA is $T^{(t+1)} \leftarrow 0.999T^{(t)}$ (with $T^{(0)} = 10^4$) and for MFA is $T^{(t+1)} \leftarrow 0.99T^{(t)}$ (with $T^{(0)} = 10^3$).

The first set of experiments is for MAP-MRF restoration performed on three synthetic images of $M = 4$ gray levels, and Figs. 10.17–10.19 show the results. In each figure, (a) is the true image with $M = 4$ gray levels, the label set $\mathcal{L} = \{1, 2, 3, 4\}$, and the pixel gray values also in $\{1, 2, 3, 4\}$. The clique potential parameters α_I and $(\beta_1, \cdots, \beta_4)$ for generating the three images are shown in Table 10.2. (b) is the observed image in which every pixel takes a real value that is the true pixel value plus zero-mean i.i.d. Gaussian noise with standard deviation $\sigma = 1$. (c) is the maximum likelihood estimate that was used as the initial labeling. (d) to (h) are the solutions found by the algorithms compared.

Table 10.3 shows the minimized energy values, the error rates, and the iteration numbers required. It can be seen from the table that objectively ALH performs the best out of the three deterministic algorithms in terms of both the minimized energy values. Overall, the solution quality is ranked as "ICM < HCF < MFA < ALH < SA", which is in agreement with a subjective evaluation of the results. We also implemented a parallel ICM using codings but the results were not as good as for the serial ICM.

The second experiment compares the algorithms in performing MAP-MRF segmentation on the Lena image of size 256×240 into a tri-level segmentation map. The results are illustrated in Fig. 10.20. The input image (a) is the original Lena image corrupted by the i.i.d. Gaussian noise with standard deviation 10. The observation model is a Gaussian distribution of standard deviation 10 with mean values 40, 125, and 200 for the three-level segmentation. An isometric MRF prior is used, with the four β parameters



(a)	(b)	(c)	(d)
(e)	(f)	(g)	(h)

Figure 10.17: Restoration of image no. 1. (a) Original image. (b) Observed noisy image. (c) Maximum likelihood estimate. (d) ALH solution. (e) ICM solution. (f) HCF solution. (g) MFA solution. (h) SA solution.

Table 10.2: The MRF parameters (α and β) and noise parameter σ for generating the three images.

	σ	α_I	β_1	β_2	β_3	β_4
Image No. 1	1	0	-1	-1	-1	1
Image No. 2	1	0	-2	-2	1	1
Image No. 3	1	0	1	1	1	1

being $(-1, -1, -1, -1)$. (b) is the maximum likelihood estimate that was used as the initial segmentation. (c)–(f) are the segmentation results of ALH, ICM, MFA, and SA, respectively. Table 10.4 illustrates the minimized energy (i.e., maximized posterior probability) values and the iteration numbers numerically. The solution quality is ranked as "HCF < ICM < SA < ALH < MFA".

According to the iteration numbers, the ALH method takes much fewer iterations than SA, about 0.2%–4.4%, to converge. Although it takes more iterations than the other deterministic algorithms, it is not as slow as it might seem. That the ALH converged after thousands of iterations was due to the stringent convergence criterion $\|u^{(t)} - u^{(t-1)}\|_\infty < 0.0001$. However,

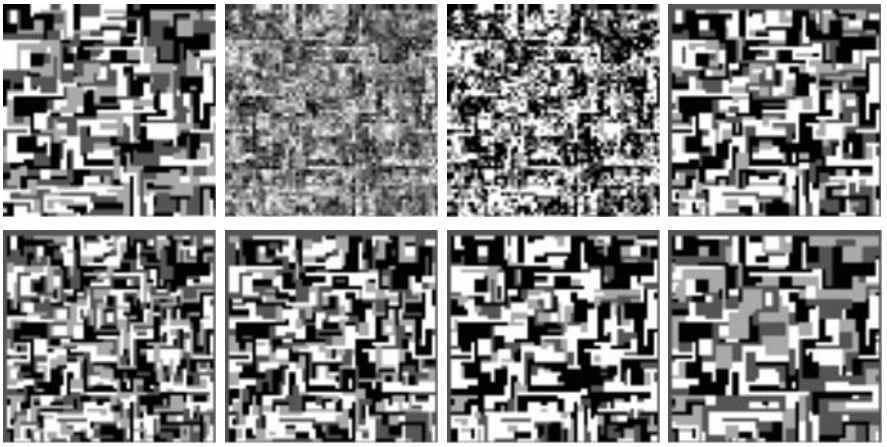


Figure 10.18: Restoration of image no.2 (refer to Fig. 10.17 for legend).

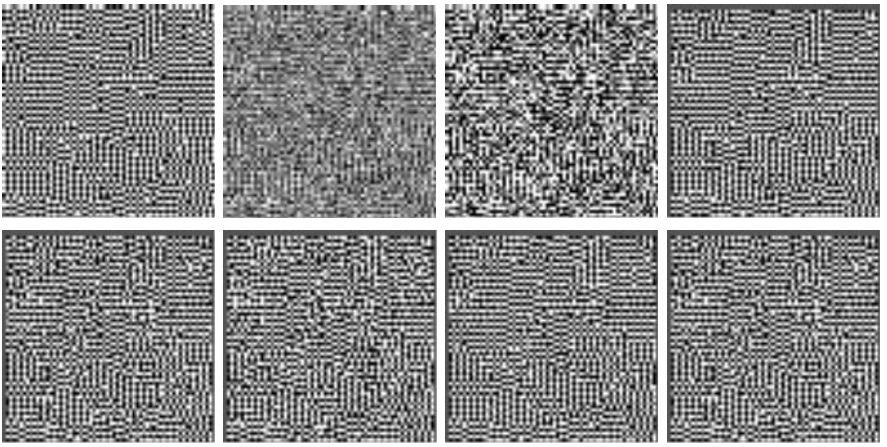


Figure 10.19: Restoration of image no.3 (refer to Fig. 10.17 for legend).

looking at the energy evolution curve in Fig. 10.21 (SA is not included there because it is far from the convergence within a thousand iterations), we see that the ALH reaches a solution better than ICM and HCF after only a dozen iterations and becomes nearly convergent after a hundred of iterations. The MFA needs five hundred iterations to converge.

10.6.3 Comparing the Comb Algorithm with Others

In the following, two experiments are presented, one for MAP-MRF image restoration and the other for segmentation, to compare the performance of

Table 10.3: The minimized energy values (top block), error rates (middle block), and iteration numbers (bottom block) for images 1–3.

	ALH	ICM	HCF	MFA	SA
Image No. 1	−11049	−10003	−10049	−10824	−11988
Image No. 2	−10191	−8675	−9650	−10073	−11396
Image No. 3	−26974	−25881	−26629	−26716	−27526
Image No. 1	0.268	0.349	0.367	0.291	0.164
Image No. 2	0.360	0.438	0.414	0.381	0.306
Image No. 3	0.212	0.327	0.273	0.181	0.125
Image No. 1	3654	7	31	553	83034
Image No. 2	1456	6	29	553	68804
Image No. 3	2789	7	36	560	92721

Table 10.4: Numerical comparison of the algorithms on the segmentation of the Lena image.

	ALH	ICM	HCF	MFA	SA
Min. Energy	−180333	−171806	−176167	−180617	−173301
Iterations	1255	7	38	545	593916

the following algorithms: (1) the Comb algorithm; (2) the ICM (Besag 1986); (3) the HCF (Chou et al. 1993) (the parallel version); and (4) the SA with the Metropolis sampler (Kirkpatrick et al. 1983). For the Comb, the parameters are $N = 10$ and $\tau = 0.01$. The implementation of SA is based on a procedure given in (Press et al. 1988). The schedules for SA are set to $T^{(t+1)} \leftarrow 0.999T^{(t)}$ with $T^{(0)} = 10^4$. The initial configurations for ICM, HCF, and SA are taken as the ML estimate, whereas those in F for the Comb are entirely random. The termination condition for Comb is that all configurations in F are the same or that 10000 new local minima have been generated.

The first set of experiments is for MAP-MRF restoration performed on three synthetic images of $M = 4$ gray levels, shown in Figs. 10.22–10.24. The original has the label set $\mathcal{L} = \{1, 2, 3, 4\}$ and the pixel gray values also in $\{1, 2, 3, 4\}$. Table 10.5 gives the clique potential parameters α_I and $\beta_1, \cdots, \beta_4$ for generating the three types of textures and the standard deviation σ of the Gaussian noise.

The second experiment compares the algorithms in performing MAP-MRF segmentation on the Lena image of size 256×240 into a tri-level segmentation map. The results are illustrated in Fig. 10.25. The input image is the original Lena image corrupted by the i.i.d. Gaussian noise with standard

Figure 10.20: Segmentation of the Lena image. Top raw: Input image, maximum likelihood segmentation, ALH solution. Bottom raw: ICM solution, MFA solution, SA solution.

Table 10.5: The MRF parameters (α and β) and noise parameter σ for generating the three images.

Image	σ	α_I	β_1	β_2	β_3	β_4
No. 1	1	0	-1	-1	-1	1
No. 2	1	0	-2	-2	1	1
No. 3	1	0	1	1	1	1

deviation 10. The observation model is assumed to be the Gaussian distribution superimposed on the mean values of 40, 125 and 200 for the three-level segmentation. An isometric MRF prior is used, with the four β parameters being $(-1, -1, -1, -1)$.

Table 10.6 compares the quality of restoration and segmentation solutions in terms of the minimized energy values. We can see that the Comb outperforms the ICM and the HCF and is comparable to SA. A subjective evaluation of the resulting images would also agree to the objective numerical comparison. The quality of the Comb solutions is generally also better than that produced by using a continuous augmented Lagrange method developed previously (Li 1998b).

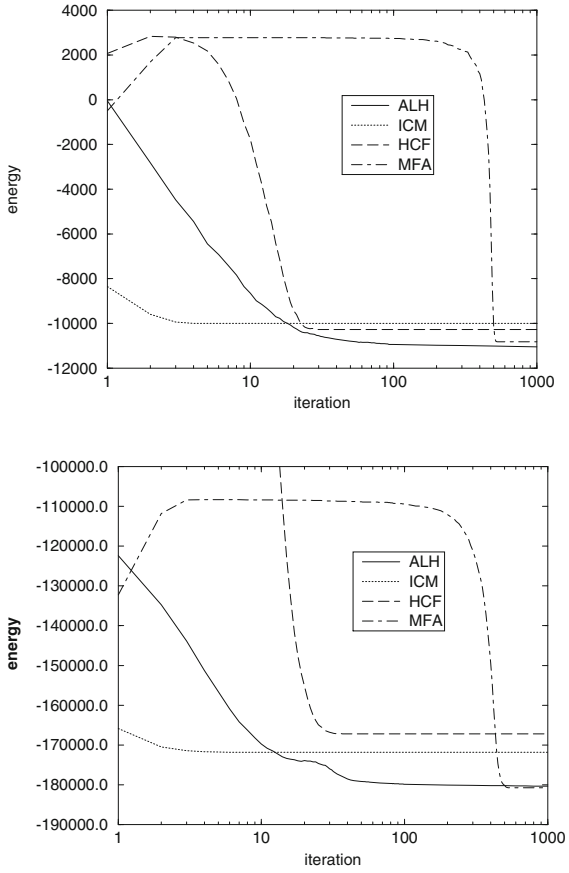
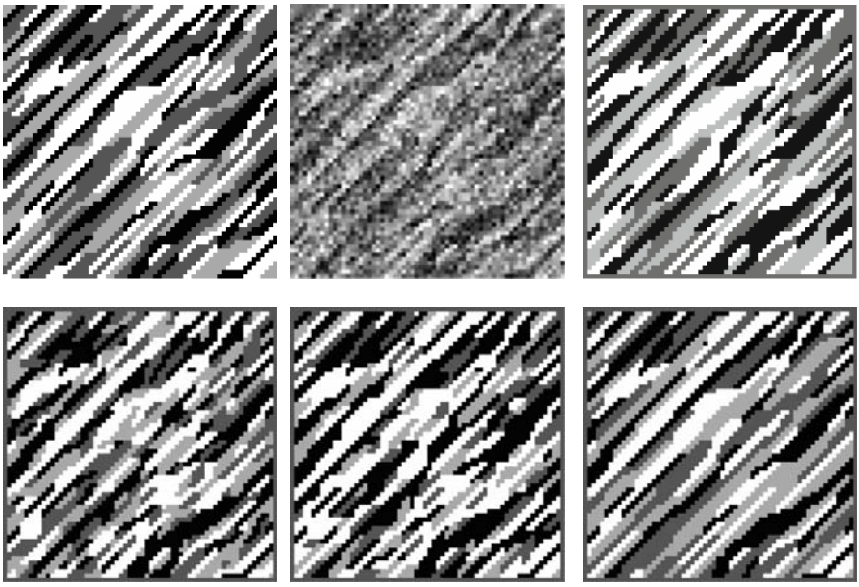


Figure 10.21: The solution evolution curves for image no.1 (top) and the Lena image (bottom).

The Comb as a random search method needs many iterations to converge, the number increasing as τ decreases. All the Comb solutions with $\tau = 0.01$ are obtained when the limit of generating $10,000$ local minima is reached. This is about 1000 times more than the fast-converging ICM and HCF. Nonetheless, the Comb takes about 1/20 of the computational effort needed by the SA.

The Comb algorithm does not rely on initial configurations at the beginning of the algorithm to achieve better solutions; the maximum likelihood estimate can lead to a better solution for algorithms that operate on a single



a	b	c
d	e	f

Figure 10.22: Restoration of image 1. (a) Original clean image. (b) Observed noisy image (input data). (c) Comb solution. (d) ICM solution. (e) HCF solution. (f) SA solution.

configuration, such as the ICM, HCF, and SA, but not necessarily for the Comb.

10.7 Accelerating Computation

The MRF configuration space for image and vision analysis is generally large, and the search for an energy minimum is computationally intensive. When the global solution is required, the computation is further increased by complications incurred by the problem of local minima and can well become intractable. Unrealistic computational demand has been criticism of the MAP-MRF framework. Efforts have been made to design efficient algorithms.

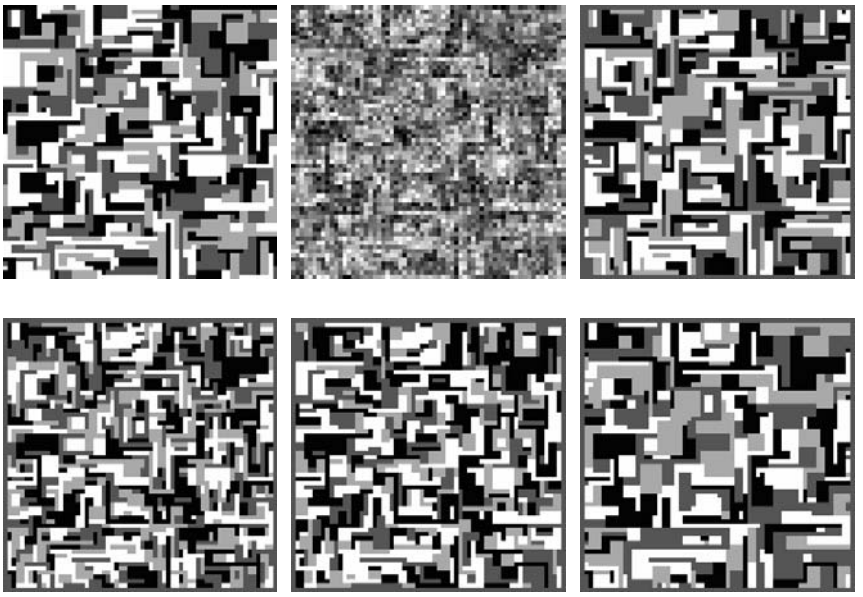


Figure 10.23: Restoration of image 2. Legends same as Fig. 10.22.

Table 10.6: The minimized energy values for the restoration of images 1–3 and the segmentation of the Lena image.

	Comb	ICM	HCF	SA
No. 1	−12057	−10003	−10269	−11988
No. 2	−10944	−8675	−9650	−11396
No. 3	−27511	−25881	−26629	−27526
Lena	−175647	−171806	−167167	−173301

10.7.1 Multiresolution Methods

Multiresolution methods provide a means for improving the convergence of iterative relaxation procedures (Hackbusch 1985). It was shown by Terzopoulos (1986a) that multiresolution relaxation can be used to efficiently solve a number of low-Level vision problems. This class of techniques has been used for MRF computation by many authors (Konrad and Dubois 1988b; Barnard 1989; Bouman and Liu 1991; Kato et al. 1993b; Bouman and Shapiro 1994). Gidas (1989) proposed a method that uses renormalization group theory, MRF's, and the Metropolis algorithm for global optimization. How to

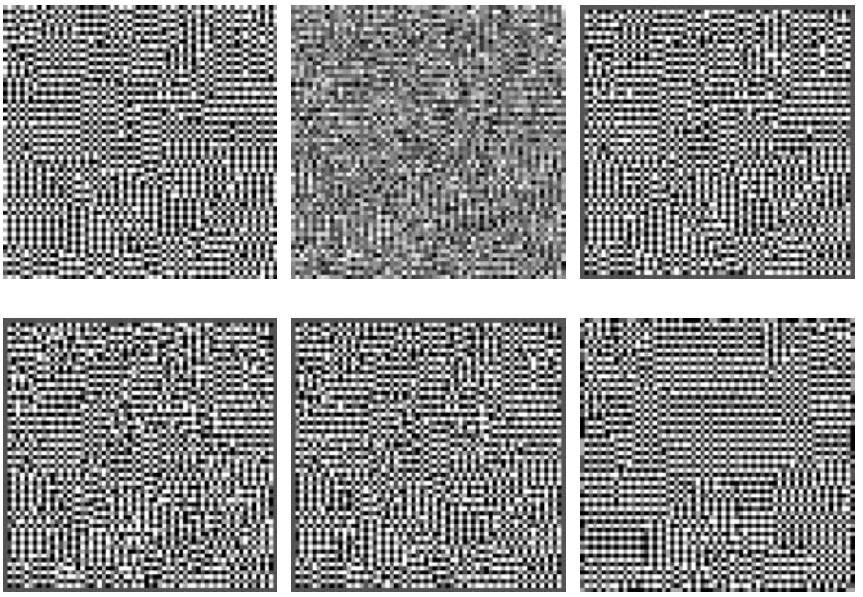


Figure 10.24: Restoration of image 3. Legends same as Fig. 10.22.

preserve the inter-relationships between objects in the scene using the renormalization group transformation has been studied in (Geiger and Kogler Jr. 1993; Hurn and Gennison 1993; Petrou 1993).

An important issue in multiresolution computation of MRF's is how to preserve the Markovianity and define consistent model descriptions at different resolutions. In general, the local Markovian property is not preserved at the coarse levels after a sub-sampling. In (Jeng 1992), two theorems are given for a periodic sub-sampling of MRF's. One gives necessary and sufficient conditions for preserving the Markovianity and the other states that there is at least one sub-sampling scheme by which the Markovianity is preserved. A multiresolution treatment is presented by Lakshmanan and Derin (1993) in which (possibly) non-Markov Gaussian fields are approximated by linear Gaussian MRF's. In (Heitz and Bouthemy 1994), a consistent set of parameters are determined for objective functions at different resolutions; a nonlinear multiresolution relaxation algorithm that has fast convergence towards quasi-optimal solutions, is developed. A general transformation model is considered in (Perez and Heitz 1994) as the "restriction" of an MRF, defined on a finite arbitrary non-directed graph, to a subset of its original sites; several results are derived for the preservation of the Markovianity which may be useful for designing consistent and tractable multiresolution relaxation algorithms.

a	b	c
d	e	f

Figure 10.25: Segmentation of the Lena image. (a) The noisy Lena image. (b) ML configuration. (c) Comb solution. (d) ICM solution. (e) HCF solution. (f) SA solution.

10.7.2 Use of Heuristics

Apart from theory-supported methods, heuristics are often combined into the search. Solution candidates are quickly located using some efficient means. They are then evaluated by the derived energy function. Because the number of such candidates is usually small, the energy values can be compared exhaustively to give the minimal solution.

Not to miss the true global solution is crucial to successfully applying heuristics. One has to balance between efficiency and the danger of missing the true solution. More restrictive heuristics reduce the number of hypotheses generated and increase the efficiency, but they also cause a greater chance of missing the true minimum.

The *bounded noise model* (Baird 1985; Breuel 1992) is a simple heuristic for approximating noise distributions to quickly reduce the search space. By checking whether an error is within the allowed bounds, numerical constraints are converted into symbolic ones so that a yes-or-no decision can be made to prune the search space. This strategy has been used in many methods where numerical constraints have to be converted into symbolic ones. For example,

in maximal cliques (Ambler et al. 1973), dynamic programming (Fischler and Elschlager 1973), and constraint search (Faugeras and Hebert 1986; Grimson and Lozano-Prez 1987), symbolic compatibilities are determined; in Hough transform (Hough 1962; Duda and Hart 1972) and geometric hashing (Lamdan and Wolfson 1988), numerical values are quantized to vote accumulators.

Hypothesis-verification is another approach for efficient search. Hypotheses are generated, which may correspond to peaks in Hough transform space or geometric hashing space, leaves of an interpretation tree (Grimson and Lozano-Prez 1987), random samples in random sampling (Fischler and Bolles 1981; Roth and Levine 1993), or result from minimal sets of image-model correspondences in the alignment method (Huttenlocher and Ullman 1987). Because the number of hypothesesgenerated is much smaller than the number of points in the original solution space, costs incurred by hypothesis evaluation is much reduced. In (Lowe 1992), matching and measurement errors are used to determine the probability of correctness for individual labels. Techniques presented therein may be used to speed verification. All these have potential applications in MRF's to tackle the problem of low computational efficiency.