Genetic algorithm parameter sets for line labelling

Richard Myers, Edwin R. Hancock *

Department of Computer Science, University of York, York Y01 5DD, United Kingdom

Abstract

This paper concerns the use of genetic algorithms for line labelling. We are interested in finding an optimal set of algorithm control parameters for this problem. We give results from using a simple genetic algorithm to solve several line labelling problems and discuss the effects of crossover type, population size, crossover rate, mutation rate and iteration limit on algorithm performance. We conclude that the algorithm is very sensitive to mutation rate, and that there is a threshold population size beyond which success rates are very high but that this threshold increases rapidly with the problem size. We recommend that a mutation rate of 0.02 be used in conjunction with a crossover rate of between 0.6 and 0.9. Iteration limit should initially be high, and should only be lowered when the other parameters have been tuned. © 1997 Elsevier Science B.V.

Keywords: Genetic algorithm; Hill climbing; Line labelling; Ambiguity; Consistency

1. Introduction

Computer simulations of evolution were first suggested in the late 1950s and early 1960s (Fraser, 1957; Toombs, 1967). Holland’s genetic algorithm (Holland, 1975) has become the most commonly used technique for simulated evolutionary optimisation. Genetic algorithms have been successfully used for a wide variety of problems (Ambati et al., 1990; Jefferson et al., 1991; Koza, 1992). Despite this there has never been an adequate theoretical framework which allows researchers to set optimal control parameters for the algorithm, although some qualitative aspects of its behaviour are known (Rudolph, 1994; Grefenstette, 1986; Qi and Palmieri, 1994a,b). Quantitative results have been obtained on a suite of test functions, and may be estimated from a variety of theoretical models. However, experience has shown that setting optimal control parameters is not easy (Grefenstette, 1986; DeJong, 1990), and that the performance of the algorithm is problem-specific; the “standard” suite of test functions (DeJong, 1975) is criticised in (Graves, 1995); and the available models make simplifying assumptions such as infinite population size (Qi and Palmieri, 1994a,b). To make matters worse, the algorithm itself is a grossly oversimplified model of the real evolutionary process: there is no speciation, no complex gene interactions such as dominance and variable penetration, no species interaction, and the “environment” does not change.

In this paper we aim to investigate the use of genetic algorithms for line-labelling. The interpretation of line drawings has been an important topic in machine vision since the seminal work of Huffman (1991), Clowes (1971) and Waltz (1975), and has obvious applications in document analysis, processing architects’ sketches, engineering drawings and so...
on. Viewed as a constraint satisfaction problem, line-labelling is one in which there are many “related” ambiguous solutions. In this paper we cast line labelling in an optimisation framework and consider the effectiveness of a genetic algorithm for locating ambiguous solutions. In particular, we focus on the choice of genetic algorithm control parameters, their interactions, and their problem-specificity.

2. Genetic algorithms

A genetic algorithm manipulates a population of candidate solutions to a problem. The candidate solutions are typically binary strings, but any representation may be used. At every generation, some of the candidate solutions are paired and parts of each individual are mixed to form two new solutions; this is crossover: uniform crossover exchanges individual bits whereas multi-point crossover exchanges whole substrings. Additionally, every individual is subject to random change – mutation. The next generation is produced by selecting individuals from the current one on the basis of their fitness, which is a measure of how good each candidate solution is. Eventually, the population should become saturated with individuals of very high fitness.

Consider a population of \( N \) binary strings (individuals) of length \( n \). The strings correspond to chromosomes; atomic regions in the strings correspond to genes. There may be several variants of each gene: a particular version is called an allele. We define the fitness of the \( i \)th individual, \( F(i) \), as an arbitrary function mapping the space \( \{0,1\}^n \) onto \((0,1]\). The population is a panmictic unit, i.e. every individual has an equal probability of mating with any other. This probability is the crossover rate. All individuals are also subject to background mutation at a relatively low rate. At the end of each generation, all parents and offspring are subject to roulette selection with replacement, allowing more fit individuals to increase their representation in the population. The probability of survival is the relative fitness, \( p_i \).

\[
p_i = \frac{F(i)}{\sum_{j=1}^{N} F(j)}. \tag{1}
\]

This quantity satisfies the axioms of probability, since \( 0 < p_i \leq 1 \) and \( \sum p_i = 1 \). It is common practice to guarantee survival of at least one copy of the fittest individual; this elitist selection heuristic is a necessary and sufficient condition for eventual convergence of the algorithm (Rudolph, 1994).

3. Line labelling

It was Waltz who first showed how a dictionary of consistent junction labellings could be used in an efficient search for consistent interpretations of polyhedral objects – this led to his seminal discrete relaxation algorithm. Such dictionaries are derived from the geometric constraints on the projection of 3D scenes onto 2D planes (Waltz, 1975; Sugihara, 1978). Hancock and Kittler have built on the work of Faugeras and Berthod (1981) and Hummel and Zucker (1983) by developing a Bayesian framework for measuring consistency in discrete relaxation (Hancock, 1990). More recently, Hancock (1994) has applied this framework to labelling polyhedral scenes. It is this formulation of line labelling as an optimisation process that we adopt in this paper.

To commence our formal development, suppose that a polyhedral scene under consideration consists of lines drawn from a set \( U = \{ u_1,\ldots,u_n \} \). Each junction in the scene can be characterised by the set of indices \( J_k \) of the lines from which it is constructed. We can form a set \( J = \{ J_1,\ldots,J_K \} \) whose elements are the tuples of line indices making up each junction. Each of the ELL, TEE, FORK or ARROW junction types has a distinct dictionary which is a compilation of the permitted label configurations. Suppose that \( \lambda_k \) denotes the dictionary for the \( k \)th junction. If the label-set applying to the scene interpretation task is \( \Lambda = \{ +, -, \rightarrow, \leftarrow \} \), then the cardinality of the junction dictionary \( |\lambda_k| \) is usually much smaller than the number of possible configurations \( |\Lambda^{|J_k|}| \). For example, there are only five consistent labellings for a FORK junction (Fig. 1), whereas \( 4^5 = 64 \) combinatorial possibilities exist.

A candidate solution to this labelling problem is a list of labels, \( L = \langle \lambda_1,\ldots,\lambda_n \rangle \), where \( \lambda_i \in \Lambda \). According to Hancock (1994) the probabilities of the individual junction labellings are computed using a model of the label corruption mechanism. This
label-error process assumes that the label on each line is subject to the action of memoryless corruption which occurs with probability \( p \). The consequence of this model is that the consistency of the junction labellings is gauged by an exponential function of their Hamming distances to the various dictionary items. Suppose that \( H_{k,l} \) denotes the Hamming distance between the current labelling \( L_k \subseteq L \) of the junction \( J_k \in \mathcal{J} \) and the dictionary item \( l \in A_k \). The Bayesian model leads to the following expression for the junction probability, \( \Gamma \):

\[
\Gamma(L_k) = \frac{(1-p)^{|J_k|}}{|A_k|} \sum_{l \in A_k} \left( \frac{p}{1-p} \right)^{H_{k,l}}.
\]  

The parameter of this criterion is the probability of memoryless label errors, \( p \). We can re-write the above expression to make the exponential role of Hamming distance explicit,

\[
\Gamma(L_k) = \frac{(1-p)^{|J_k|}}{|A_k|} \sum_{l \in A_k} \exp \left( -H_{k,l} \ln \frac{1-p}{p} \right).
\]  

As the error probability, \( p \), decreases towards zero, labellings lying outside the dictionary make smaller contributions. Under these conditions the sum of exponential terms appearing in Eq. (3) becomes dominated by the term involving the smallest Hamming distance. Thus we can maximise the consistency of a labelling by minimising its cost,

\[
C(L) = \sum_{k=1}^{|J|} \min_{l \in A_k} H_{k,l}.
\]  

The associated fitness is computed by exponentiating the cost function

\[
F(i) = \exp(-\beta C_i(L)),
\]  

where \( C_i(L) \) is the cost of the labelling represented by individual \( i \). The scaling parameter, \( \beta \geq 0 \), which defaults to 1. Increasing \( \beta \) makes the fitness function less tolerant of errors in labellings.

4. Parameter choice

It is well known that choosing suitable parameter values for genetic algorithms is very difficult (Grefenstette, 1986; DeJong, 1990). Here we adopt an empirical approach in which several different parameter sets will be tried on a set of labelling problems. We would expect the genetic algorithm to proceed by combining locally consistent labellings until a solution is found. The choice of operator parameters (crossover and mutation rates) is essentially a tradeoff between conservatism and exploration: high mutation rates with uniform crossover will tend to cover the search space well but will disrupt partial solutions. Conversely, low mutation rates and 2-point crossover will tend to preserve solutions found so far but may not adequately explore the search space. Relatively unconstrained line labelling problems such as ours should have many consistent solutions and it should not be necessary to undertake sweeping exploration of the search space. We therefore expect that more conservative operator settings will be better for line labelling.

It is also important to stress that we expect the different parameter settings to interact with one-another. For instance, there is clearly a relationship between population size and number of iterations in that both allocate trials: the population allocates trials horizontally and the number of iterations allocates trials vertically. A problem such as line labelling, in which there are many alleles, which have little significant interaction, should benefit from a large population size rather than a large number of iterations, since the selection process tends to reduce the diversity of the population. On the other hand, a problem with few alleles per locus but complex interactions between them would benefit more from
Table 1
A 9×9 graeco-latin square

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Aα</td>
<td>Bβ</td>
<td>Cγ</td>
<td>Dδ</td>
<td>Eε</td>
<td>Fζ</td>
<td>Gη</td>
<td>Hθ</td>
<td>Iι</td>
</tr>
<tr>
<td>II</td>
<td>Bγ</td>
<td>Cδ</td>
<td>Dε</td>
<td>Eζ</td>
<td>Fη</td>
<td>Gθ</td>
<td>Hη</td>
<td>Iλ</td>
<td>Aα</td>
</tr>
<tr>
<td>III</td>
<td>Cε</td>
<td>Dζ</td>
<td>Eη</td>
<td>Fθ</td>
<td>Gι</td>
<td>Hα</td>
<td>Iβ</td>
<td>Aγ</td>
<td>Bδ</td>
</tr>
<tr>
<td>IV</td>
<td>Dη</td>
<td>Eε</td>
<td>Fζ</td>
<td>Gθ</td>
<td>Hη</td>
<td>Iγ</td>
<td>Aδ</td>
<td>Bε</td>
<td>Cζ</td>
</tr>
<tr>
<td>V</td>
<td>Eτ</td>
<td>Fα</td>
<td>Gβ</td>
<td>Hγ</td>
<td>Iδ</td>
<td>Aε</td>
<td>Bζ</td>
<td>Cη</td>
<td>Dθ</td>
</tr>
<tr>
<td>VI</td>
<td>Fδ</td>
<td>Gη</td>
<td>Hδ</td>
<td>Iε</td>
<td>Aζ</td>
<td>Bη</td>
<td>Cθ</td>
<td>Dσ</td>
<td>Eη</td>
</tr>
<tr>
<td>VII</td>
<td>Gδ</td>
<td>Hε</td>
<td>Iζ</td>
<td>Aη</td>
<td>Bθ</td>
<td>Cι</td>
<td>Dσ</td>
<td>Eβ</td>
<td>Fη</td>
</tr>
<tr>
<td>IX</td>
<td>Hη</td>
<td>Iθ</td>
<td>Aι</td>
<td>Bα</td>
<td>Cβ</td>
<td>Dη</td>
<td>Eδ</td>
<td>Fε</td>
<td>Gζ</td>
</tr>
</tbody>
</table>

Table 2
Parameter values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Coding</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>1–9</td>
<td>10 20 30 40 50 60 70 80 90</td>
</tr>
<tr>
<td>Iteration limit</td>
<td>1–IX</td>
<td>10 20 100 200 300 400 500</td>
</tr>
<tr>
<td>Crossover rate</td>
<td>A–I</td>
<td>0.10 0.20 0.35 0.24 0.60 0.65 0.75 0.85 0.90</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>α–ι</td>
<td>0.001 0.005 0.010 0.020 0.050 0.075 0.100 0.200 0.300</td>
</tr>
</tbody>
</table>

a large number of iterations because it is unlikely that any correct solutions would arise as the result of a few crossovers and mutations.

5. Experiments

5.1. Method

5.1.1. Experimental design
We wish to examine the effects of varying population size, iteration limit, crossover rate, and mutation rate for a variety of problems and two crossover types. The space of possible parameter sets is large even if we restrict ourselves to a few values for each parameter: if 9 values of each parameter are to be tested, $9^4 = 6561$ possible combinations are available. To reduce the dimensionality of this space we adopt a graeco-latin square design: if we encode the four parameters as 1–9, I–IX, A–I and α–ι, the design in Table 1 results. Each combination occurs exactly once, but each value of every parameter is tested in conjunction with all the values of the other three. This allows us to reduce the number of experiments needed to $9^2 = 81$. The total number of experiments per value for each parameter is 9. Table 2 gives the values for each of the parameters.

5.1.2. Test problems
The problems used were simple line drawings containing 7, 8, 9, 28, 40 and 42 lines, as shown in Fig. 2. The sizes of the search spaces for these drawings are $16 \times 384$, $65 \times 536$, $262 \times 144$, $7.2 \times 10^{16}$, $1.2 \times 10^{24}$ and $1.9 \times 10^{25}$, respectively.

5.1.3. Analysis
The result of each experiment is the proportion of program runs that yield a consistently labelled drawing. The standard method for analysing proportions is logistic regression (Dobson, 1983). We adopt the approach given in (Green et al., 1993): check for 2-factor interactions, then assess each factor individually.

![Fig. 2. Test drawings. (a) 7 and 8 line figures, (b) 9 and 28 (''house'') line figures, (c) 40 (''groove2'') and 42 (''wedding cake'') line figures.](image-url)
Table 3
Main results: total numbers of solutions found in all experiments \((N = 20)\). U-tests were conducted with \(H_0: \mu_{\text{uniform}} = \mu_{\text{2-point}}\) and \(H_A: \mu_{\text{uniform}} < \mu_{\text{2-point}}\), except for \(^{\dagger}\), where \(H_A: \mu_{\text{uniform}} > \mu_{\text{2-point}}\).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Crossover type</th>
<th>Observations</th>
<th>U-test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2-Point</td>
<td>Uniform</td>
<td></td>
</tr>
<tr>
<td>7-line figure</td>
<td>1168</td>
<td>1023</td>
<td>1620</td>
</tr>
<tr>
<td>8-line figure</td>
<td>1245</td>
<td>1204</td>
<td>1620</td>
</tr>
<tr>
<td>9-line figure</td>
<td>1224</td>
<td>1220</td>
<td>1620</td>
</tr>
<tr>
<td>House (28)</td>
<td>369</td>
<td>423</td>
<td>1620</td>
</tr>
<tr>
<td>Groove-2 (40)</td>
<td>74</td>
<td>3</td>
<td>1620</td>
</tr>
<tr>
<td>Wedding cake</td>
<td>3</td>
<td>5</td>
<td>1620</td>
</tr>
</tbody>
</table>

Table 4
Analysis of deviance for 7-line figure with uniform crossover \((N = 20)\).

| Model          | Deviance | d.f. |         |         |         |         |         |
|----------------|----------|------|---------|---------|---------|---------|
| 0) All 2-factor          | 5.60     | 53   |         |         |         |         |
| 1–0) – interactions      | 5.02     | 17   |         |         |         |         |
| 1) Main effects          | 9.17     | 62   |         |         |         |         |
| 2–1) – population        | 4.06     | 8    | \(p = 0.8514\) |         |         |         |
| 3–1) – iteration         | 1.45     | 8    | \(p = 0.9936\) |         |         |         |
| 4–1) – crossover         | 0.88     | 1    | \(p = 0.3482\) |         |         |         |
| 5–1) – mutation          | 4.44     | 1    | \(p = 0.0351\) |         |         |         |

5.2. Results

The main results are summarised in Table 3. The probabilities in the right-most column are the results of a one-tailed U-test with the null hypothesis that there is no difference between number of successes for 2-point and uniform crossovers, against the alternative hypothesis that the success rate is greater for 2-point. The \(p\)-value given is the probability that the observed differences arose by chance.

We examine the effects of individual parameters with an analysis of deviance table (Green et al., 1993), an example of which for the 7-line figure with uniform crossover is given in Table 4. Initially a model is fitted with all parameters and 2-way parameter interactions present. The interactions are then removed; if there is no significant change in the deviance, factor interactions can be ignored. Subsequently, each factor is removed from the model. As in Table 3, \(p\)-values are the probabilities that these

Table 5
Significance of the parameters \((N = 20)\).

<table>
<thead>
<tr>
<th>Factor</th>
<th>Problem</th>
<th>7 uniform</th>
<th>7 2-point</th>
<th>8 uniform</th>
<th>8 2-point</th>
<th>9 uniform</th>
<th>9 2-point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
<td>0.9977</td>
<td>1.0</td>
<td>0.9950</td>
<td>0.9910</td>
<td>0.9090</td>
<td>0.9997</td>
<td></td>
</tr>
<tr>
<td>Population</td>
<td>0.8514</td>
<td>0.7658</td>
<td>0.8027</td>
<td>0.6267</td>
<td>0.4209</td>
<td>0.7434</td>
<td></td>
</tr>
<tr>
<td>Iteration</td>
<td>0.9936</td>
<td>0.9985</td>
<td>0.9980</td>
<td>0.9997</td>
<td>0.9960</td>
<td>0.9979</td>
<td></td>
</tr>
<tr>
<td>Crossover</td>
<td>0.3482</td>
<td>0.6631</td>
<td>0.5436</td>
<td>0.5315</td>
<td>0.5202</td>
<td>0.5998</td>
<td></td>
</tr>
<tr>
<td>Mutation</td>
<td>0.0351</td>
<td>0.0537</td>
<td>0.0070</td>
<td>0.0108</td>
<td>0.0273</td>
<td>0.0188</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>7 uniform</th>
<th>7 2-point</th>
<th>8 uniform</th>
<th>8 2-point</th>
<th>9 uniform</th>
<th>9 2-point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
<td>0.9998</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Population</td>
<td>0.1903</td>
<td>0.2423</td>
<td>1.0</td>
<td>0.9671</td>
<td>0.9996</td>
<td>1.0</td>
</tr>
<tr>
<td>Iteration</td>
<td>0.9505</td>
<td>0.8925</td>
<td>0.9999</td>
<td>0.9921</td>
<td>0.9994</td>
<td>0.9999</td>
</tr>
<tr>
<td>Crossover</td>
<td>0.7476</td>
<td>0.8049</td>
<td>0.9988</td>
<td>0.8605</td>
<td>0.6708</td>
<td>0.9972</td>
</tr>
<tr>
<td>Mutation</td>
<td>0.0023</td>
<td>0.0108</td>
<td>0.9993</td>
<td>0.1182</td>
<td>0.8208</td>
<td>0.9960</td>
</tr>
</tbody>
</table>
Fig. 3. Effect of varying the mutation rate. Note the marked improvement in success rate at lower mutation rates, especially around 0.02.

Fig. 4. Effect of varying the crossover rate. For each problem there is a peak in success rate for some crossover rate in the range [0.6,0.9]. There is also an upswing at lower crossover rates.
changes could have arisen by chance. Table 5 gives the $p$-values for all 12 line labelling experiments. As can be seen, the only factor which has a significant influence on the outcome is the mutation rate. The high $p$-values for the wedding cake and groove-2 problems may be due to the paucity of successes.

The analysis of deviance does not give a full picture of the effects of the various parameters. Combined plots of the effect of varying the four parameters are given in Fig. 3. These plots give a broad picture of the relationships between parameter values and success rate.

5.3. Discussion

Table 3 demonstrates that in general 2-point crossover is superior to uniform. The underlying assumption of the U-test is that the observations are continuous. We believe that this assumption is reasonable for proportions with sufficiently large denominators, and therefore make our claim about crossover type with some confidence. Moreover, this result is in line with our qualitative predictions given earlier. Table 3 also indicates that the parameter sets do not scale with the problem: the success rate varies inversely with the number of lines in the problem. This result is hardly surprising and suggests that the optimal parameter set will depend to some degree upon the size of the problem.

Almost every analysis of deviance in Table 5 indicates a marked sensitivity to mutation rate. This sensitivity is a striking feature in Fig. 3. Even for those experiments where success rates were very low and the mutation rate cannot be statistically said to contribute to the outcome, there is a peak at 0.02. It seems that the mutation rate should not be set much higher than 0.1 regardless of the size of the problem. The scaleability of mutation rate effects is unsurprising since mutation is genic (i.e. more genes, more mutations). The fact that low but non-zero values of the mutation rate are beneficial agrees with the general consensus that mutation is a necessary source of background noise, allowing the exploration of new regions of the search space, but not the primary cause of algorithm convergence.

The crossover rate does not appear to be as

![Effect of Population Size on Success Rate (N = 180)](image)

Fig. 5. Effect of varying the population size. There appears to be a point beyond which relatively little is gained by increasing population size.
important as one might have expected (Fig. 4). Higher crossover rates do seem to improve success rate up to some upper limit beyond which there is a falloff. There is also a slight upswing at very low crossover rates: this appears to confirm our model of a Metropolis-like genetic drift, which is unmasked at very low crossover rates. The effects are slight however and we can only say that the optimal crossover rate will probably occur in the range \([0.6,0.9]\) for each problem.

The analysis of deviance tables shows that population size has some effect on success rate, but that this effect is rather noisy. Nevertheless the plots in Fig. 5 suggest that a population size of at least 30 is necessary for optimal success rates with the smaller problems, and that in these cases there is little to be gained from increasing the population size above this threshold. This plateau is probably responsible for the high \(p\)-values in Table 5 and suggests that a linear model may not be appropriate here. Clearly, though, there is a sharp falloff in success rate as the population size decreases below its threshold, which indicates that finding the correct population size requires some experimentation for each problem. It is important in terms of both space and time requirements of the algorithm that the population size be set to a minimum.

6. Conclusion

We have demonstrated that for a range of line labelling problems, 2-point crossover often outperforms uniform crossover, and that mutation rates of the order of 0.02 with crossover rates in the range \([0.6,0.9]\) produce the best results, given a sufficiently large population size and a reasonable iteration limit. For small problems (about 10 lines), a population size of 30 is adequate, but larger problems (20 to 40 lines) require significantly larger populations. A rough calculation indicates that the threshold population size required for near-optimal success rates increases with problem size at least as rapidly as a polynomial of degree 3. This suggests that genetic algorithms may not be appropriate for solving very large line labelling problems. The iteration limit should initially be set high, and only lowered after the other parameters have been tuned.

Discussion (paper presented by Hancock)

Holz: Premature convergence is a known problem, for which there is a standard explanation, which is reproduction and a standard solution which is Baker’s ranking algorithm. Usually, when people don’t use ranking that is because either they can show that the problem domain does not exhibit premature convergence, so they are not concerned, or because the increase in computation time is of major concern. I am interested both in why you choose not to use ranking and also how you feel your results compared to ranking.

Hancock: Remember that our motivation in this study is looking at ambiguous labelling problems, so what we are concerned with is whether the algorithm has sufficient time to discover lots of different solutions.

Holz: Ranking is generally believed to address that issue.

Smyth: I have a question about labelling with real images. You tend to see it used on artificial data.

Hancock: Perhaps I ought to go back to the motivation again. We took this as a problem with known solutions and well studied. We don’t see this work necessarily as having a direct application to the labelling of real line drawings. We wanted to use it as a case study. Specifically we did it to understand, for instance, what happens about ambiguities and which is the best cross-over strategy for the consistent labelling problem. It is a well-controlled problem, whereas if you perform this study with real world graphs, it is more difficult to interpret the results.

Kappen: I would be interesting to know whether you also compared with competing algorithms, such as restarted iterative improvement, or simulated annealing.

Hancock: We certainly have a paper on that, but not for this problem. In the issue of “Pattern Recogni-
tion” this month (Pattern Recognition 30 (1997) 953–970) we compared with multiple random starts and with simulated annealing. There is also a comparison in the domain of graph matching for simulated annealing and multiple starts. The genetic algorithm seems to work better, but the genetic algorithm has to be very carefully tailored to the problem at hand.

Kappen: Better, meaning better in quality, in time or both?

Hancock: We were able to empirically show that the time to convergence was like $N^{2.8}$. But the main effective advantage was that we got the maximum of the cost function in a few iterations. This was much faster than simulated annealing. The quality of the final solution was also better than that obtained with gradient ascent, but not better than that obtained with simulated annealing.

References


