Simulated Annealing and Genetic Algorithms for Partial Shape Matching

Ender Ozcan   Chilukuri K. Mohan*
2-120 CST, Dept. of Electrical Eng. and Computer Science
Syracuse University, Syracuse, NY 13244-4100, U.S.A.
eozcan/mohan@top.cis.syr.edu

February 10, 1997

Abstract

Partial shape matching may be viewed as an optimization problem, to be solved using methods such as simulated annealing (SA) and genetic algorithms (GAs). We apply and compare both these methods for matching input shapes with model shapes described in terms of features such as line segments and angles. The quality of matching is gauged using a measure derived from attributed shape grammars [10, 11]. Current results show that both SA and GA succeed in the shape matching task; the GA is faster and yields the global optimum more often than the versions of SA implemented.
Keywords: *shape matching, pattern recognition, simulated annealing, genetic algorithms, attributed strings, evolutionary computing*

1 Introduction

Shape recognition is a challenging task studied by numerous researchers. Given a set of *model* shapes and an *input* shape, the problem consists of identifying the model shapes whose instances are present in the input shape. For example, most robotics applications for part inspection and VLSI design involve locating and identifying objects. For shapes of objects that are occluded, or touch or overlap with other objects, flexible shape recognition algorithms are needed to identify overlapping shapes, making use of incomplete information.

We have formulated shape matching as the problem of optimizing a cost function, successfully solved using a genetic algorithm (GA) [6] and an attributed string representation [10, 11]. Preliminary results, for about 10 model shapes, are described in [7]. In this paper, we describe and compare a GA for shape recognition (using a new mutation operator) with variants of the simulated annealing (SA) algorithm using a larger library of forty model shapes. Both algorithms are successful in shape matching, although the GA is faster and

*Author for correspondence

reached global optima more often than the versions of SA implemented. This is consistent with the theoretical comparison of Hart [5]. Di Ianni [3] has also applied GAs and simulated annealing for matching shapes, but the results obtained were not encouraging, possibly because of using raw pixel arrays rather than shape features. Bala and Wechsler [1] use GAs to develop morphological operators that can discriminate among classes containing different shapes, not directly for shape matching.

Section 2 outlines the representation underlying our algorithms. Section 3 describes measures used to evaluate a candidate solution for the shape matching problem. Sections 4 and 5 present the versions of SA and GA implemented. Results are described in Section 6, followed by concluding remarks.

2 Attributed string representation

Attributed strings [10, 11] are used for the representation of polygonal shapes, consisting of line segments. A string of features $(x_1, x_2, \ldots, x_i, \ldots, x_n)$ is used to represent each shape. Each feature $x_i = (l_i, \theta_i)$ is formed of two attributes: the length l_i of the corresponding line segment, and the relative angle θ_i it forms with the preceding line segment x_{i-1}.

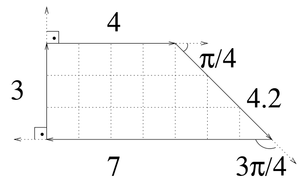

Figure 1: Attributed string representation of a shape.

We assume the use of existing feature extraction algorithms. All lengths are normalized to provide size invariance. However, the total observed length of a partially obscured shape is not a reliable measure. Therefore, normalization is performed using the immediately preceding feature's length: $l'_i = l_i/l_{i-1}$. For example, Figure 1 depicts a quadrilateral whose normalized representation is $((1.33, \frac{\pi}{2}),\ (1.05, \frac{\pi}{4}),\ (1.67, \frac{3\pi}{4}),\ (0.43, \frac{\pi}{2}))$. This representation is invariant under translation, scale and rotation transformations. The shape recognition problem now reduces to multiple substring matching.

We use the following notation:

- For each feature, the normalized lengths and angles are obtained by $l(.)$ and $\theta(.)$.
- Input shape $I = (I_1, I_2, \ldots, I_n)$, where I_p is the pth feature, consisting of attributes $(l(I_p), \theta(I_p))$. Size (length) of input shape $|I| = n$, the number of features in I.
- The model shapes are $M_1, M_2, \ldots, M_S$, where $M_j = (M_{j,1}, M_{j,2}, \ldots, M_{j,m_j})$, $M_{j,r}$ is the rth feature of the jth model shape, consisting of attributes $(l(M_{j,r}), \theta(M_{j,r}))$. Size of the jth model shape $|M_j| = m_j$.

- Each individual $P = (P_1, P_2, \ldots, P_k, \ldots, P_n)$ corresponds to a mapping μ_P from input shape features to model shape features such that $P_k = \mu_P(I_k) = M_{j,i}$, where $1 \leq k \leq n$, $1 \leq j \leq S$, and $1 \leq i \leq m_j$.

Example: Let $[(2,3),(2,4),(2,5),(3,4)\ldots]$ be a configuration used for SA. The first feature of the input shape (I_1) is mapped by this configuration to (2,3), the third feature of the second model shape ($M_{2,3}$).

3 Shape Matching as Optimization

Shape matching can be viewed as the task of optimizing a $Cost$ function that evaluates the quality of how well each feature of the input image is matched by features of the model shapes to which a candidate solution (configuration) maps them. Cost also depends on considerations such as requiring that adjacent features in the input image should be mapped to adjacent features in the same model shape. The $Cost$ function, described below, also discourages fragmented mappings: a candidate solution is of poor quality if input shape features are mapped to fragments from too many different model shapes. For a given candidate solution,

$$Cost = \text{ Number of model shapes } + \text{ Number of unmatched features of input shape.}$$

For two features $(I_k, \mu_P(I_k))$ to be considered to have matched, their "distance" $d(I_k, \mu_P(I_k))$ must exceed a threshold value:

$$Matched(I_k, \mu_P(I_k)) = \begin{cases} 1 & \text{if } d(I_k, \mu_P(I_k)) < threshold \\ 0 & otherwise \end{cases}$$

The matching $threshold$ depends on feature lengths, e.g., 0.2 when $\max(l(I_k), l(\mu_P(I_k))) >$ 0.5, and 0.9 when this quantity is less than 0.005. The distance (dissimilarity) between input shape feature I_k and model feature $f_k = \mu_P(I_k)$ is measured by means of a distance function $d(I_k, \mu_P(I_k))$, defined as follows:

$$d(I_k, \mu_P(I_k)) = \begin{cases} d_\theta(I_k, \mu_P(I_k)) + d_l(I_k, \mu_P(I_k)) & \text{if } \mu_P(I_{k-2}) = M_{i,j-2},\ \mu_P(I_{k-1}) = M_{i,j-1} \text{ and} \\ & \mu_P(I_k) = M_{i,j} \text{ for some } i, j, k \\ d_\theta(I_k, \mu_P(I_k)) & \text{if } \mu_P(I_{k-1}) = M_{i,j-1} \text{ and } \mu_P(I_k) = M_{i,j} \\ & \text{ for some } i, j, k \\ \infty & otherwise. \end{cases}$$

This measure has angle and length components. Note that if the current feature and the preceding feature are not mapped to successive features of the same model shape, then neither the angle nor the length component is reliable, and these features cannot be considered to have matched. The first component, from angle measurements, is defined as follows:

$$d_\theta(I_k, \mu_P(I_k)) = c_\theta \ \text{abs}(\theta(I_k) - \theta(\mu_P(I_k)))$$

The constant c_θ is chosen in our experiments so that differences up to $\pi/18$ are considered negligible. For angle information (d_θ) to be useful, it is necessary for two successive input shape features to be mapped to two successive features of the same model shape.

The length component of the distance measure compares the normalized feature lengths as follows:

$$d_l(I_k, \mu_P(I_k)) = \frac{\mathrm{abs}(l(I_k) - l(\mu_P(I_k)))}{\max(l(I_k), l(\mu_P(I_k)))}$$

This measure is invoked only if three successive input shape features to be mapped to three successive features of the same model shape. This is because normalized length information for the kth feature is reliable only if the $(k-1)$th feature's length is known, and the latter information is unreliable if the $(k-2)$th input feature is not matched to the corresponding feature of the same model.

4 Simulated Annealing

Simulated annealing (SA) was proposed by Kirkpatrick *et al.* [8] and applied to many problems, such as partitioning, wiring, and component replacement in VLSI design. This technique is derived from the simulation of the evolution of a solid to thermal equilibrium which was introduced by Metropolis *et al* [9].

SA starts the search with an initial *configuration* at an initial *temperature*. A *cooling schedule* is used to reduce the temperature of the system at each step. At each temperature $T > 0$, the algorithm repeatedly proposes changes to the current configuration. If the cost decreases, the change is accepted unconditionally, otherwise it is accepted with probability $\max(1, e^{-\Delta Cost/T})$.

In the SA for shape matching, each configuration is an attributed string representation of a shape. An initial configuration C_0 is obtained as follows. First, an input shape feature I_k is randomly chosen, and mapped to a randomly chosen feature $M_{j,r}$ in a randomly chosen model shape M_j. Features $I_{k+1}, I_{k+2}, \ldots$ are then mapped to $M_{j,r+1}, M_{j,r+2}$, etc., with the features of M_j being traversed in circular manner. After assigning all possible features of M_j, if any input shape features remain unassigned, then the next unassigned input shape feature I_{k+m_j} is mapped to a new randomly chosen model shape feature, $M_{j',r'}$, followed by mapping as many as possible of the remaining input shape features $I_{k+m_j+1}, I_{k+m_j+2}, \ldots$ to $M_{j',r'+1}, M_{j',r'+2}, \ldots$. This process is repeated until all input shape features have been assigned, tracing through input shape features in circular sequence, i.e., I_1 is the feature presumed to follow I_n. Initial temperature is set be the attributed string length of the input shape: $T_0 = n$.

A new configuration is obtained by replacing a sub-sequence of the current configuration by a randomly chosen model shape sub-sequence (of equal length). We have experimented with three versions of SA, referred to as $SA1, SA2$ and $SA3$, that differ as follows. In $SA1$, sub-sequence length (≥ 3) to be replaced is chosen randomly. In $SA2$ and $SA3$, the sub-sequence length to be replaced is fixed at 3 and a *Similarity List* is used to bias the replacement; an input shape feature and a model shape feature are considered to be *similar* when the error corresponding to each of the next two successive angles is less than $\pi/18$.

$$Similar(I_k, \mu_P(I_k)) = \begin{cases} 1 & \text{if } \max\left(|d_\theta(I_{k+1}, \mu_P(I_{k+1}))|,\ |d_\theta(I_{k+2}, \mu_P(I_{k+2}))|\right) < \pi/18 \\ 0 & otherwise \end{cases}$$

In $SA2$, each feature of a configuration is replaced with probability $1/n$. In $SA3$, it is assured that at least one sub-sequence is replaced, and the expected number of sub-sequence replacements is $n/10$.

Each simulated annealing trial is executed for a fixed number of steps (temperature values). To limit the amount of computation, we use a cooling schedule in which the temperature is reduced at each step by a fixed factor α, chosen to be 0.95 in our experiments. The average change in cost caused by an initial modification is $O(n)$. The initial temperature T_0 is chosen as $O(n)$, high enough to guarantee that most cost changes are accepted in the early steps of the algorithm. To bring the final temperature down to $O(1)$, when most cost-worsening moves are rejected, the number of steps is chosen to be a sufficiently large multiple of $\log(n)$.

5 Genetic Algorithm

Genetic Algorithms (GAs) were introduced by J. Holland [6], and have been used to solve many difficult problems [4]. Search is conducted using a "population" of individuals, where each individual is a candidate solution to the problem at hand. Operators such as crossover and mutation are applied to individuals in the current population, yielding offspring individuals, often with a greater number of offspring allocated to individuals of high "fitness" (quality). The new generation is selected from among the offspring and members of the old generation. This process is iterated until either the population converges to a relatively unchanging state, or until computational limitations are exceeded.

The negative of the $Cost$ function described was used as the fitness function in the GA. A linear ranking strategy was used during reproduction, with the best individual being allocated roughly five times more offspring than the worst individual. An elitist survival selection mechanism was also used: the best third (67%) of all individuals in a generation are allowed to survive into the next generation. These choices were made arbitrarily.

Our earlier experiments, described in [7], indicated that the GA works best if all of the three operators (crossover, mutation and hill climbing) are used. Crossover and mutation generate new individuals, and hill climbing is used to improve the solutions obtained. Traditional one-point crossover (1PTX) is applied to the individuals, producing two children. More complex operators, such as 2PTX and uniform crossover, did not yield better results than 1PTX. In the work reported here, the mutation operation used is similar to the operation defined in $SA2$ to perturb a configuration; this operator resulted in better performance than the mutation operator described in [7].

Hill climbing is applied to improve the mappings obtained at the borders between feature sequences mapped to different model shapes. Each hill climbing step attempts to improve the fitness of an individual by shifting the "intersection point" (between feature sequences mapped to different model shapes) in either direction, replacing the relevant component by the most appropriate feature from the model to which neighboring shape features are mapped. For instance, if $\mu_P(I_k) = M_{j,i}$ and $\mu_P(I_{k+1}) = M_{j',i'}$, hill climbing changes either $\mu_P(I_k)$ to $M_{j',i'-1}$ or $\mu_P(I_{k+1})$ to $M_{j,i+1}$, in case such a change improves fitness.

6 Experimental Results

In our experiments, all shapes are polygons described as a sequence of adjacent features. A library of 40 model shapes (Figure 3) is used for the experiments. The total number of features of the model set is 1505. All of the input shapes (Figure 2) are obtained by overlapping two or three model shapes. Each test was repeated 100 times for all input shapes, on a Sun workstation.

In our GA experiments, we used a population size twice the number of features of the input shape. Each GA run was terminated when the correct solution was reached, or if the number of generations equals 1000. Each SA run was terminated if the correct solution, or the final temperature was reached. During the SA runs, temperature was decreased whenever the number of $attempts = 150n$, or $replacements = 15n$, was exceeded.

In the tables, prefixes "i" and "m" to refer to input and model shapes, respectively, e.g., "i1" is the first input shape. The input shape $i2$, containing two overlapping instances of $m22$, was used to choose the parameters of the best SA. As shown in Table 1, the best simulated annealing results were obtained with $SA1$.

Experimental results of SA and GA are compared in Table 2. The GA is faster than SA, requiring fewer moves in the search space. SA gets stuck at a local optimum in most of the input shapes ($i0, i5-7$), whereas the GA reached the correct results for all input shapes in all runs except $i2$. In matching image $i2$, the frequency of wrong matches was 2% for the GA, and 7% for the SA; these cases correspond to local optima of the cost function where part of the input shape is matched with some features of $m25$ instead of $m22$.

7 Conclusion

We have presented two approaches for shape recognition based on attributed string representations: genetic algorithms and simulated annealing. Outline features of shapes are represented using attributed strings. Each line segment is described in terms of length and angle attributes, normalized for size invariance.

The algorithms described are fast, and explore a relatively small number of elements of the search space. They are also space-efficient compared to neural network models, most of which have large memory requirements. They overcome the primary problem faced by greedy algorithms that tend to get stuck in locally optimal solutions; however, even the algorithms described here are occasionally susceptible to local optima when different model shapes have feature sub-sequences that are almost identical. Between GA and SA, the former yields superior performance, requiring less computation.

So far, only polygons without holes have been used as model shapes and input shapes. For shapes with curvilinear segments, a different representation would be more appropriate [2], but we expect that the GA and SA can still be used for matching. Current work involves testing the GA with more complex input shapes, and developing better variants of the algorithms.

References

[1] J. Bala, and H. Wechsler, *Shape Analysis Using Morphological Processing and Genetic Algorithms*, Int. Conf. on Tools for AI, pp.130-137, 1991.

[2] A. M. Bruckstein, N. Katzir, M. Lindenbaum, and M. Porat, *Similarity-Invariant Signatures for Partially Occluded Planar Shapes*, Int. J. of Computer Vision, 7:3, pp. 271-285, 1992.

[3] M. Di Ianni, *Simulated Annealing and Genetic Algorithms for Shape Detection*, Polish J. Control and Cybernetics, to appear.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading (MA), 1989.

[5] W. E. Hart, *A Theoretical Comparison of Evolutionary Algorithms and Simulated Annealing*, Sandia National Laboratories Technical Report, 1995

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. Mich. Press, 1975.

[7] E. Ozcan, and C. K. Mohan, *Shape Recognition Using Genetic Algorithms*, IEEE Proc. of Intl. Conf. on Evolutionary Computation, pp.411-417, May 1996.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing*, Science, vol. 220, pp. 671-680, 1983.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, *Equation of State Calculations by Fast Computing Machines*, J. of Chem. Physics, vol. 21, pp. 1087-1092, 1953.

[10] W. -H. Tsai and K. S. Fu, *Attributed grammar - a tool for combining syntactic and statistical approaches to pattern recognition*, IEEE Tran. on SMC, vol. 10, Dec. 1980.

[11] W. -H. Tsai and S.-S. Yu, *Attributed string matching with merging for shape recognition*, IEEE Tran. on PAMI, vol. 7, pp 453-462, July 1985.

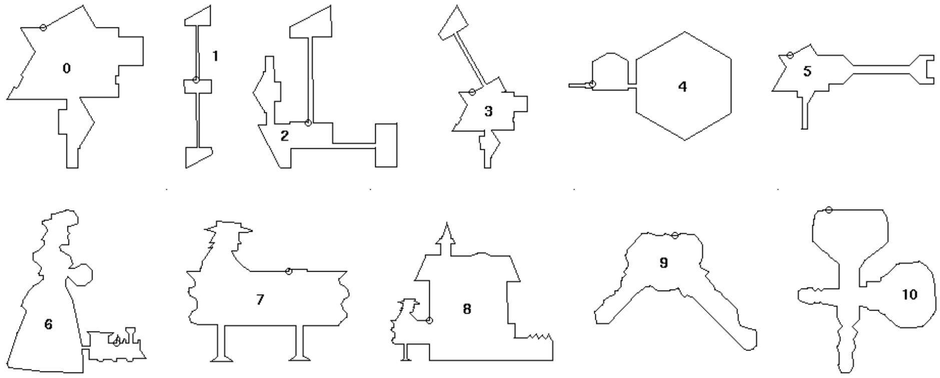

Figure 2: Normalized input shapes used for the experiments.

Tbale 1: Comparison of different instances of SA on input shape $i3$, in over 100 runs; $fr.$ = frequency with which the correct result was obtained.

SA	$fr.$	Attempts		Replacements	
		μ	σ	μ	σ
SA1	0.93	156,450	67,093	31,002	385
SA2	0.00	99,675	898	57,720	0
SA3	0.00	323,845	1,269	36,418	411

Table 2: Comparison of test results for GA and SA, in over 100 runs.

Shape label	No. of features	Constituent shapes	Freq. of Match		No. of States visited		Time (sec.)	
			GA	SA	GA	SA	GA	SA
$i0$	32	$m20, m21$	1.00	0.99	2,277	115,207	0.73	18.32
$i1$	20	$m22, m22$	1.00	1.00	772	35,633	0.12	3.68
$i2$	37	$m21, m22, m22$	0.98	0.93	28,341	156,450	9.91	27.96
$i3$	41	$m20, m21, m22$	1.00	1.00	4,320	189,572	1.52	37.00
$i4$	24	$m26, m27$	1.00	1.00	1,215	73,938	0.39	9.03
$i5$	29	$m28, m29$	1.00	0.77	1,068	154,299	0.23	22.31
$i6$	113	$m35, m37$	1.00	0.96	13,132	726,322	4.21	388.80
$i7$	71	$m31, m36$	1.00	0.76	17,136	477,104	5.25	164.29
$i8$	86	$m31, m34, m36$	1.00	0.44	8,815	819,652	1.78	337.85
$i9$	67	$m9$,$m19$	1.00	0.91	4,278	399,620	1.45	125.92
$i10$	91	$m4$,$m11$	1.00	0.49	12,294	864,839	7.17	372.13

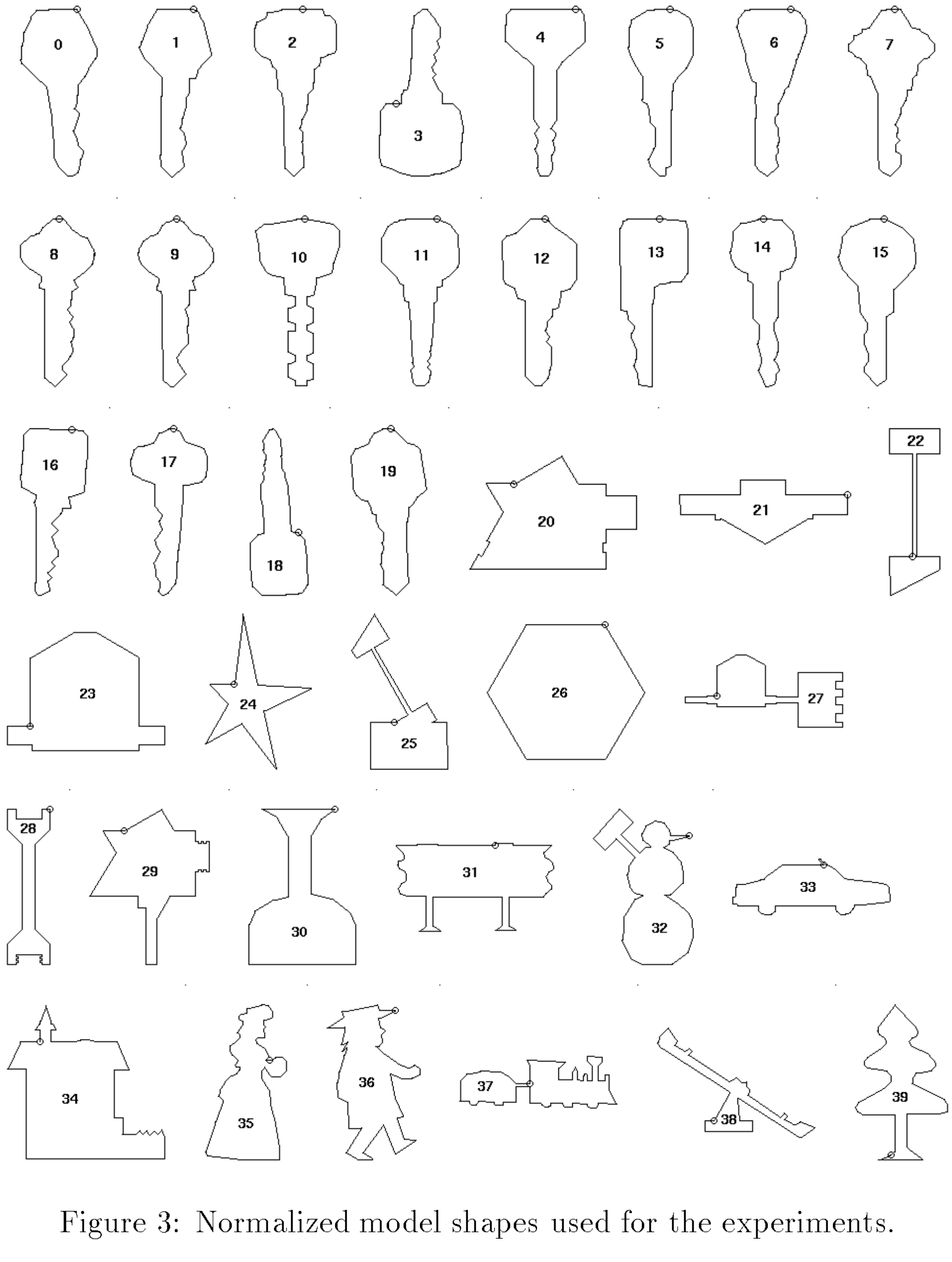

Figure 3: Normalized model shapes used for the experiments.