Convex Onion Peeling Genetic Algorithm: An Efficient Solution to Map Labeling of Point-Feature

Wan D. Bae
University of Wisconsin-Stout
baew@uwstout.edu

Shayma Alkobaisi
United Arab Emirates University
shayma.alkobaisi@uaeu.ac.ae

Petr Vojtěchovský
University of Denver
petr@math.du.edu

Sada Narayanappa
University of Denver
snarayan@cs.du.edu

Kye Bae
kyebae01@noa.nintendo.com
Nintendo of America, Inc.

ABSTRACT

Map labeling of point-feature is the problem of placing text labels to corresponding point features on a map in a way that minimizes overlaps while satisfying basic rules for the quality. This problem is a critical problem in the applications of cartography and Geographical Information Systems (GIS). In this paper we study the fundamental issues related to map labeling of point-feature and develop a new genetic algorithm to solve this problem. We adopt a data structure called convex onion peeling and utilize it in our proposed Convex Onion Peeling Genetic Algorithm ($COPGA$) to efficiently manage point features. We evaluated the performance of the proposed algorithm through extensive experiments on both synthetic and real datasets. The experimental results show that our genetic algorithm based on the convex onion peeling structure is an efficient, robust and extensible algorithm for automated map labeling of point-feature.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Cartography and GIS

General Terms

Design, Performance

Keywords

Cartography, GIS, automated map labeling, onion peeling, genetic algorithm

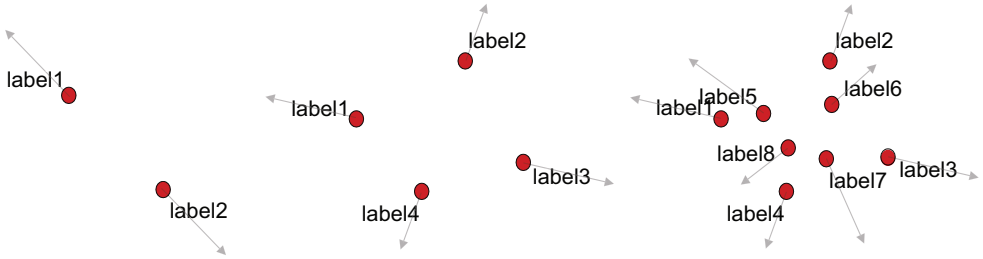
1. INTRODUCTION

Placing text labels to corresponding features is a critical task and makes up a large proportion of map production. Labels are essential components in a map for delivering information to users and hence map labeling needs to avoid label-label and label-point overlaps for clarity. Although many solutions have been proposed for automated map labeling, placing labels is still performed manually in many applications, which is time consuming and expensive. Hence developing efficient methods for automated map labeling is a central problem in many applications of cartography and GIS. Solutions to this problem can be also beneficial for other application domains in the areas of graph diagrams, architectural drawings, and medical image analysis.

Although map features to be labeled could be points (e.g., cities), lines (e.g., rivers, streets), or polygons (e.g., lakes, countries), this combinatorial aspect of map labeling is independent of the nature of the features being labeled. This paper therefore concentrates on point features which are also referred to as sites. Due to the high degree of freedom to place each label, map labeling of point-feature is a complex combinatorial problem where a search space and a cost function need to be defined. It has been proven that finding the optimal solution to automated map labeling is NP-hard [9, 13]. It is therefore reasonable to resort to approximate solutions based on heuristics. Several studies based on the genetic algorithm [10, 15] have discussed map labeling of point-feature and have shown promising results. However, little research has been done on developing good data structures to optimize the solution. In this paper, we focus on developing an efficient data structure and methods that can be utilized in the genetic algorithm.

Our approach is based on a simple observation in which labels tend to be placed away from each other in order to avoid conflicts. Suppose that we have two labels, $label1$ and $label2$, which need to be placed to the corresponding points (sites). In Figure 1 (a), we place $label1$ at a left-upper position of its corresponding point, then a right-lower position of the other point might be one of the possible positions for $label2$. Similarly, Figure 1 (b) might be a solution to map labeling placement for four points. The idea is to group the points into layers and start placing labels to corresponding points towards the outer direction of each layer while keeping the labels away from the other labels and points. If we have few number of points, achieving this goal seems simple. However, this is not the case in more complicated real world cases. Figure 1 (c) shows an example of eight points where additional techniques need to be considered for placing labels. In this paper we define the search space and a cost function for map labeling of point-feature. To minimize this cost function, we consider the following important aspects of

SACÍ0, March 22-26, 2010, Sierre, Switzerland.



(a) Two points (b) Four points (c) Eight points

Figure 1: A heuristic approach to map labeling

the genetic algorithm: 1) Initialization: generating a good initial population (labels' positions) can result in a better solution, 2) Selection: constraints vs. non-constraints in search space of the positions of a label during the evolutionary process, 3) Global optimization: any genetic algorithm possibly suffers from the local optimal trap (no improvement after a certain number of evolutions).

2. RELATED WORK

A set of general cartographic principles for map label placement have been defined in [11]; 1) Legibility: labels must have legible font sizes and be positioned in a way that is easily read, 2) Unambiguity: each label must clearly identify a single feature and not interfere with any other label or feature. Label positions to the right of a point feature are preferred to those on the left. Label above a point feature are preferred to those below, 3) Overlap avoidance: a label should not overlap any other label or feature, 4) Aesthetics: labels should not be overly clustered or distract from map features.

Popular solutions to map labeling problem using a rule-based approach were proposed in [2, 1, 7]. This approach was to encode the knowledge needed for the objective function. The authors in [17] represented the map labeling problem in a computational geometric approach. Their definition simply disallows any overlaps between labels. This version of the problem was shown to be NP-hard in [9] and an $O(nlogn)$ running time approximation algorithm was proposed. Their proposed algorithm finds a valid labeling of at least half of the optimal size. Another type of map labeling called boundary labeling was presented in [3]. This problem is a combination of label-placement and graph drawing problems. The authors provided different versions of their algorithms and compared the running time of each.

In [5] a heuristic method to the optimization of the map labeling was proposed based on Simulated Annealing [12]. The quality of the proposed labeling method was quantified using a metric that calculates a labeling scoring function based on the number of conflicts of labels with sites and labels with labels. The main advantage shown of simulated annealing approach over others are shown to be the simplicity to implement and the generality of the method that can be applied to any feature (point, line and area).

Evolutionary algorithms are strategies for function optimization that are useful when solving problems that cannot be feasibly solved on large problem sets [8]. A genetic algorithm is an iterative global search procedure to achieve the optimization of the cost function. The algorithm works in parallel on a population of candidate solutions from the search space [8]. This approach can be adopted in the map labeling problem. There are many algorithms for cost reduction. However most of them suffer from the same difficulty called local minimum trap [6]. A genetic algorithm for labeling point features was proposed in [15]. The authors compared their solution to Simulated Annealing approach and have shown that their genetic algorithm provided slightly better solutions than Simulated Annealing, however, it required more CPU time. Another genetic algorithm for automated map labeling of point feature was proposed in [10]. The authors compared their approach to previously proposed algorithms based on hill climbing, simulated annealing and random algorithms and showed by experimental results that their proposed algorithm outperformed the other approaches in terms of number of conflicts.

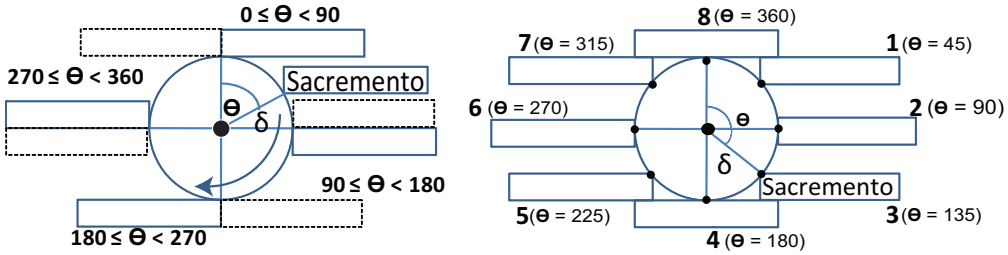
A comprehensive survey of map labeling algorithms can be found in [6]. The paper discussed the NP-hard nature of the map labeling problem and the exponential time complexity of heuristic solutions. It also proposed two algorithms; one based on discrete gradient descent and the other is based on Simulated Annealing that were empirically evaluated along with other previously proposed solutions. Extensive experimental results showed that none of the map labeling algorithms outperforms the others in all cases.

3. PROBLEM DEFINITION

Map labeling of point-feature consists of a set S of n sites in the plane $S = \{s_1, s_2, ..., s_n\}$ and a set L of label candidates (possible position) for each site, $L = \{l_1, l_2, ..., l_n\}$. Our objective is to place each l_i to the corresponding s_i within a search space while minimizing a cost function.

We first discuss basic notations and structures in placing a label to a site that are used in our paper. The Minimum Bounding Box [6] of label l_i is the smallest box that encloses text label l_i, which is denoted by MBB_i. MBB_i consists of four corner points, P_{ll} (lower-left corner), P_{ul} (upper-left corner), P_{ur} (upper-right corner), and P_{lr} (lower-right corner). Let δ be the distance between the center of site s_i and the closest point on MBB_i of the label l_i. A label is placed to its associated site with a distance of δ for overall map readability. In this paper, we use a fixed value of δ for all labels. Figure 3 illustrates an example of placing a label to its corresponding site. The upright vector v_h from the site and the vector to the closest point on the MBB of the label $v_c^{'}$ creates an angle Θ.

The label candidates for a site need to be predefined. A



(a) Continuous search space (b) Discrete search space

Figure 2: Search space for label positions

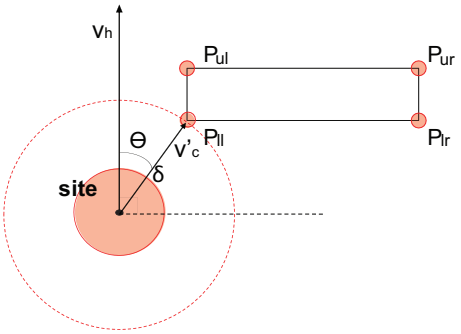


Figure 3: Notations for labels

label candidate is one of many possibilities to place a label for a certain site. We define two different models of the search space: 1) continuous space: use an infinite number of candidates, 2) discrete space: use a discrete number of label candidates. For the discrete number of label positions we use eight positions in this paper to compare to previously proposed algorithms. Figure 2 (a) and (b) show examples of a continuous and a discrete search space, respectively.

Both models are refined and used in our proposed algorithm: the first is used for the initialization and the second is used for the selection in the mutation process.

The task of the evaluation step for labeling is to detect all such conflicts and rate the label candidates accordingly. The evaluation process has to detect the following categories: 1) conflicts of a label candidate with other labels (label-label), 2) conflicts of a label candidate with sites other than the associated site (label-site), and 3) aesthetic preference and tradition. The outcome from this classification can be used in a cost function to assign a particular value to each label candidate describing its overall suitability.

Let C be a set of n label costs, $C = \{C_1, C_2, ..., C_n\}$, where C_i is the cost of l_i. The parameters for a label cost are defined as follows:

- c_i^l: number of conflicts of l_i with $\forall l_j \in L \setminus l_i$,
- c_i^s: number of conflicts of l_i with $\forall s_j \in S \setminus s_i$,
- p_i: a penalty for preferred location. In this paper, positions in the 1^{st} quadrant ($0 \leq \Theta \leq 90$) are considered as preferred positions and the penalty values of these positions are set to 0 and all the rest of the positions have penalty equal to 1.

C_i is then calculated as follows: $C_i = a_1 \cdot c_i^l + a_2 \cdot c_i^s + a_3 \cdot p_i$, where a_1, a_2, and a_3 are constant factors. Label-label and label-site conflicts result in the same cost in this paper (1.0, 1.0 and 0.1 are used for the values of a_1, a_2 and a_3 in our experiments). The cost function for the map labeling is $\sum_{i=1}^{n} C_i$. Hence, we define a cost function $F(S)$ for a given set of sites S as follows:

$$F(S) = \sum_{i=1}^{n} (a_1 \cdot c_i^l + a_2 \cdot c_i^s + a_3 \cdot p_i)$$

The optimal solution should find a set of label locations that minimizes the number of overlaps among labels as well as the number of overlaps between labels and sites while we minimize the number of evolutions in the genetic algorithm (Section 4.2).

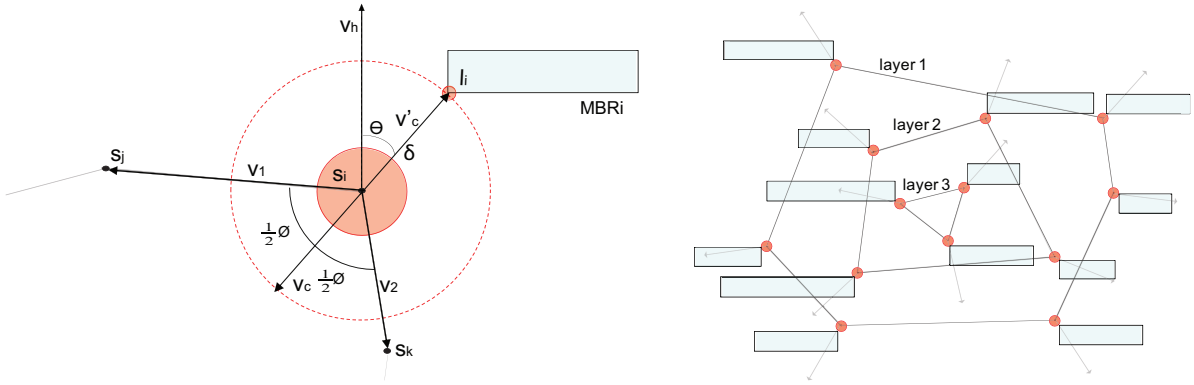
4. COPGA

Convex Onion Peeling Genetic Algorithm ($COPGA$) adopts the genetic algorithm with the search space and cost function defined in Section 3. $COPGA$ consists of three main steps:

1. Convex Onion Peeling (COP): construct the convex onion peeling data structure of the sites in a given map.

2. Initialization: generate a population of individual solution, where each individual is a vector of labeling position. Let $P(t)$ be the population of individuals at generation t, where $t = 0, 1, 2, ...$ Then the initial population is $P(0)$. The initial population is evaluated using the cost function and the cost C_{P_t} is calculated.

3. Evolutionary process: this step starts if the evaluation result of $P(0)$ does not satisfy the terminal condition In the evolutionary process, an offspring population is generated by means of selection and search operators. The main search operators are recombination and mutation. The following is an outline of the evolutionary process and the details will be discussed in Section 4.2:
   3.1 Generate an offspring population from the parent population using recombination and mutation operators. There are intermediate populations generated during evolutionary process: P^1, P^m, P^2 and P^3. If no improvement is made for some number of evolutions, we inverse the search space for mutation to get out of the local optimal trap.
   3.2 Evaluate the offspring population using the cost function. A comparison of each solution's cost is made



(a) Calculate a label position

(b) Generate initial population

Figure 4: Label positions using convex onion peeling

using the algorithm's cost function.
3.3 A number of offsprings or parents survive this natural selection, and the rest are discarded. The surviving solutions become the new parents for the next generation. This selection process is based on the evaluation of the cost function and the cost C_{P_t} is calculated.
3.4 Repeat these steps until the convergence criteria (termination condition) has been satisfied.

COPGA provides solution to the basic elements of the genetic algorithm by taking advantages of utilizing the convex onion peeling structure. We summarize *COPGA* in Algorithm 1. The details of each function in the algorithm are omitted due to space limitation.

Algorithm 1 COPGA(S, L); a set of sites, a set of labels

```
1: N ← 100; t ← 0 {population size; # of evolutions}
2: COP ← constructCOP(S)
3: P(0) ← initializeSites(COP, L, N)
4: P(t) ← P(0)
5: C_{P_t} ← evaluate P(t)
6: while termination condition not satisfied do
7:    P^1 ← selectForRecombination(P_t)
8:    P^m ← selectForMating(P^1)
9:    P^2 ← crossover(P^m)
10:   if local optimal trap condition then
11:      P^3 ← mutate(P^2) with inversion
12:   else
13:      P^3 ← mutate(P^2) without inversion
14:   end if
15:   P(t+1) ← selectReplacement(P^3, P(t))
16:   P(t) ← P(t+1)
17:   C_{P_t} ← evaluate P(t)
18:   t ← t+1
19: end while
```

4.1 Convex Onion Peeling and Initialization

The convex onion peeling technique [4] has been widely used in many application domains, i.e., image processing, pattern recognition, photo image analysis, and study of Earth atmosphere. *COPGA* constructs a sequence of nested convex hulls for the sites in a map. We refer to this structure as convex onion peeling (*COP*) of the sites. Sites in a map are divided into several groups (nested layers) using the convex onion peeling technique (Figure 4 (b)). Two well-known initialization techniques are commonly used in many of the proposed algorithms for the map labeling problem with point feature: preferred position (upper-right position) and random position. In this paper we propose a new initialization technique that utilizes the convex onion peeling structure. *COPGA* calculates the initial position of each site's label based on the convex onion peeling. Each site is associated with three vectors, two (v_1, v_2) are related to the neighbor sites in the layer and one (v_c) is the half of the inner angle ϕ (see Figure 4 (a)). Then the reverse vector v_c' and the vertical vector of the site v_h determines the angle Θ. A label's initial position is determined based on the value of Θ; an example of the initialization result is shown in Figure 4 (b).

COP provides an efficient geometric data structure for placing labels to reduce the possibility of conflicts by placing labels outside of each layer. It also allows to solve the problem in a divide and conquer manner; it places the labels to the sites of each layer and combines all label positions for an initial solution to the whole map labeling problem.

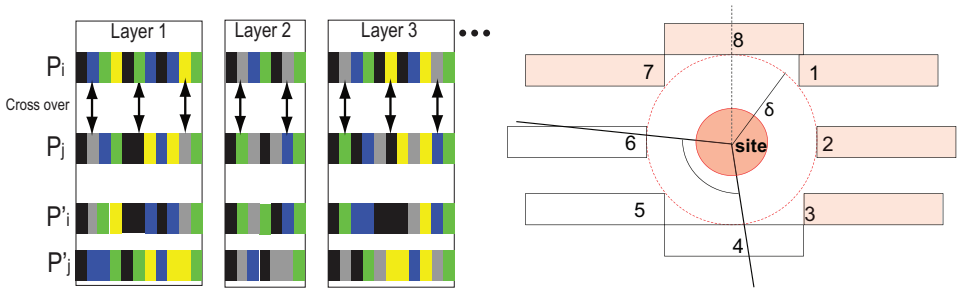
4.2 Evolutionary Process

4.2.1 *Recombination*

The recombination operator (cross over) [8, 14] is used to create new individuals by combining the genetic information of two parents. The individuals of population $P(t)$ are selected for recombination according to our cost function. Let P^1 be an intermediate population of these selected individuals. Individuals from P^1 enter the mating set P^m with a given probability $p_c = 0.5$. The individuals from P^m are mated using the cross over operator. Cross over on the pair for mating is conducted based on each layer of the convex onion peeling. Our algorithm does not allow cross over across the layers. Then a new intermediate population P^2 is obtained. Figure 5 shows an example of the cross over in our algorithm.

4.2.2 *Mutation*

The mutation operator [8, 14] generates new individuals by variations (labels' position changes) of a single individual with a probability equal to the probability of mutation



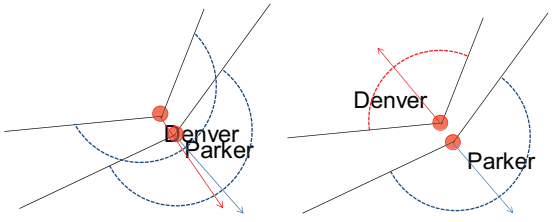
(a) Cross over (b) Search space for mutation

Figure 5: Cross over and mutation

$p_m = 0.1$. Let P^3 be the population obtained by applying mutation on P^2. The selected label's position is replaced with a new position. Considering all eight possible locations shown in Figure 2 (b), this change could cause more conflicts because most labels tend to be outside of the convex. Hence we modify the search space by removing the angle ϕ resulting in only positions outside of the convex. No significant effect on the freedom of the search space is found since the convex hull always creates outer angles greater than 180°. Hence there are always at least four possible label positions available. Figure **??** shows an example for the search space by mutation. We also investigate another approach, a greedy approach, for the selection of mutation. All conflicted labels are retrieved and random selection for mutation is conducted on these conflicted labels.

4.2.3 Inversion

Convex onion peeling does not work very well when the sites in the adjacent layers are too close which results in the labels' tending to be shifted in similar directions. The methods for initialization, recombination, and mutation are all based on convex onion peeling structure. Hence it may not be easy to resolve this problem. We relax this restriction when the algorithm reaches the possible local minimum trap, no better solution after a certain number of evolutions (0.3∗ number of initial conflicts used in our experiments). In that case, we configure the label positions so that the search space for the conflicting labels is changed to the inner space. Figure 6 shows an example of the unsolved problem and our solution by using the inverse of the search space.



(a) Unsolved conflicts (b) Reverse search space

Figure 6: Unsolved local conflict and reverse space

5. PERFORMANCE EVALUATION

We first show the results of our initialization using convex onion peeling on three well-known existing algorithms and then present the results of *COPGA* compared to a previously proposed genetic algorithm (GA) [10].

In our experiments, we considered both synthetically generated maps and real maps. The number of sites in the synthetic datasets was varied between 40 and 160, and the sites' locations were distributed uniformly and independently. For each size of synthetic datasets, we randomly generated 100 maps and we conducted 100 trials for each real dataset. Then the average values were reported. Our real datasets were obtained and extracted from USGS [16]. The number of sites in the real datasets are 73 (Englewood, CO), 116 (New Jersey, NY) and 161. Each dataset was converted into a given boundary of the map with a size of 650 x 650 pixels. Although our algorithm supports different font styles and sizes of map labels, we only present the results with the following setup for sites and labels due to space considerations: 1) sites were represented by circles with $radius = 3$ pixels, 2) the font style was set to "courier", the font size was set to 10, and a random length between 6 and 15 characters were used for labels, 3) $\delta = 3$ pixels was used for the distance between the center of a site and the closest point on the corresponding label's MBB. Similar qualitative and quantitative trends were observed in all other experiments.

The common performance metrics for the map labeling problem are the values of the cost function $F(S)$, the CPU time and the number of evolutions. However, different algorithms use various implementation characteristics and parameters. Both CPU time and reduction rate were used for the termination condition of HC. For SA, we set initial temperature to 2.5 and temperature drop rate was 0.2. Normal temperature drop and sudden temperature drop were set to 5∗ number of sites and 1∗ number of sites, respectively. Table 1 summarizes the parameters for GA and $COPGA$.

5.1 Results of Initialization

First, we present the results of our proposed map initialization method using convex onion peeling compared with two other initialization methods, preferred position (default) and random position. Figure 7 shows the average costs after initialization on synthetic datasets computed using $F(S)$ in equation (1). On average onion peeling initialization resulted in 26.04% and 15.22% of reduction over default and random initialization, respectively. In addition, the initialization using onion peeling produced less conflicts than the default and random initialization methods in all cases.

GA	population	100
	cross over	uniform cross over w/ $P_c = 0.5$
	mutation	random replacement w/ $P_m = 0.1$ on conflicted labels
	termination	cost = 0 or 1000 evolutions
$COPGA$	population	100
	cross over	onion peeling cross over w/ $P_c = 0.5$
	mutation	onion peeling random replacement w/ $P_m = 0.1$ on conflicted labels
	inversion	# evolutions ($0.3 \cdot$ # initial conflicts) without finding a new best solution
	termination	cost = 0 or 1000 evolutions

Table 1: Implementation characteristics and parameter

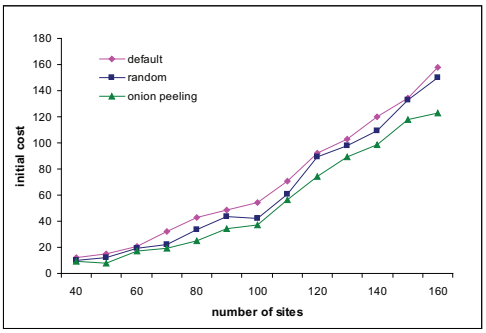


Figure 7: Map initialization

To compare the effect of initialization, we implemented three existing map labeling algorithms, hill climbing (HC), simulated annealing (SA) and a genetic algorithm (GA) in [10] by applying the three initialization methods. Figure 8 illustrates the performance of default, random, and onion peeling initializations using the synthetic datasets. Figure 8 (a) and (b) show the comparisons of HC using the three initialization methods. In Figure 8 (a), we plotted the average conflict costs of HC after 10 seconds of CPU time along with the initial costs. For example, when the number of sites is 110, the final costs with default, random, and onion peeling are 6.5, 5 and 2, respectively. The overall cost reduction rates using default, random and onion peeling initializations were 75%, 81% and 84%, respectively. Figure 8 (b) shows the CPU time to reach 90% conflict reduction rate of HC with different initializations. On average, HC using onion peeling required 25.72% and 15.96% less CPU time to obtain 90% cost reduction compared to HC using default and random positions, respectively. The results show that the larger the datasets is, the more advantage of using onion peeling over the other methods is.

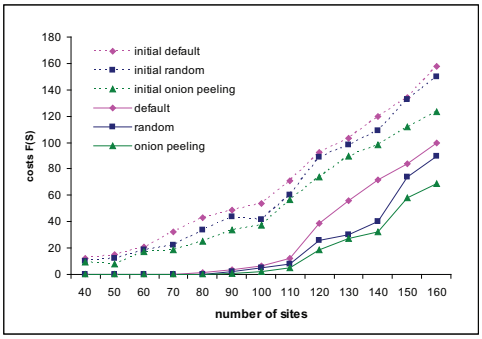
Figure 8 (c) and (d) show the average final conflict costs and the CPU time of SA using the three initializations. SA using onion peeling resulted in 39.31% less conflict cost than SA using default position and 27.71% less final cost than SA using random position on average. The CPU times required for termination were also compared. When the number of sites is 120, the required CPU time was 25, 23 and 198 seconds for default, random and onion peeling, respectively. The results show that SA with onion peeling required 13.51% and 11.03% less CPU time than SA with default and random initializations. Similar qualitative and quantitative trends were observed in the results of GA using these three initializations. In Figure 8 (e) and (f), the average final conflict costs and CPU times of GA with different initializations were plotted. Figure 8 (f) shows that GA with onion peeling required 28.20% and 19.70% less CPU time than GA with default and random initializations, respectively.

Table 2 show the comparisons of the three initialization methods using the real datasets. For HC, we showed the cost after 30 seconds and the CPU time to achieve a 80% conflict reduction rate using the three initialization. For SA and GA, the final costs and CPU times are reported. Overall, the results of the real datasets are similar to those of the synthetic datasets despite the skewed distribution of the real datasets. All the results show that better map initialization can improve the performance of map labeling algorithms. The results illustrate that the performance improvement with GA using onion peeling was the best among the three algorithms. This motivated us to implement $COPGA$ that utilizes convex onion peeing not only in the initialization step, but also in the evolutionary process.

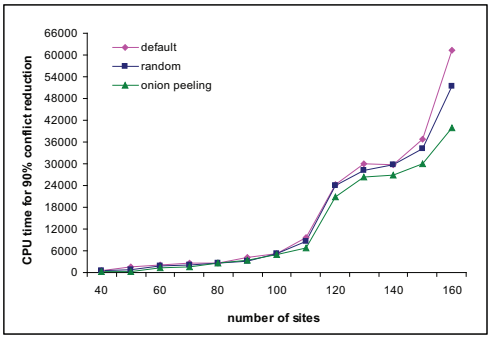
5.2 Results of Evolutionary Process

This section presents the results of our proposed algorithm $COPGA$ compared with GA. In Figure 9 (a), we plotted the average final conflict costs of the algorithms along with their initial costs, $COPGA$ uses onion peeling position and GA uses random position for initializing the maps. The average initial costs of $COPGA$ was 15.56% less than that of GA. For the maps with the size of sites between 40 and 80, both $COPGA$ and GA resulted in no conflicts in the final solution. However, $COPGA$ outperformed GA for the maps with the sizes of sites between 90 and 160, resulting in 61.29% less values of the conflict costs on average. Figure 9 (b) shows the CPU time required for termination. The performance improvement of $COPGA$ over GA was between 52.67% and 93.85%. On average, $COPGA$ required 64.77% less CPU time than GA. In addition, the number of evolutions of $COPGA$ to terminate was 50.38% less than that of GA on average.

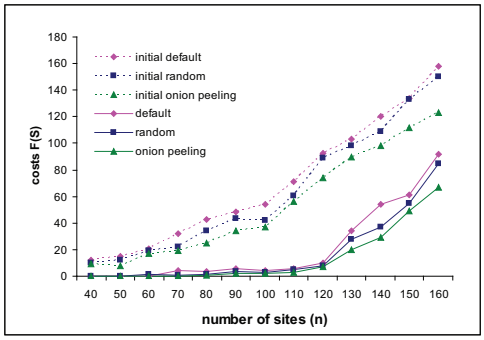
Table 3 illustrates the performance of $COPGA$ and GA using the real datasets. The initial conflict costs, final costs, and the reduction rates are presented. We also show the number of evolutions of $COPGA$ and GA for termination. $COPGA$ resulted in 20.00% less cost for the map with 116 sites and 21.55% less cost for the map with 161 sites. The numbers of evolutions for the termination of $COPGA$ were 47.76%, 55.00% and 68.16% less than those of GA for dataset 1, dataset 2 and dataset 3, respectively. The results show that the larger the dataset is, the better the performance of $COPGA$ over GA is.



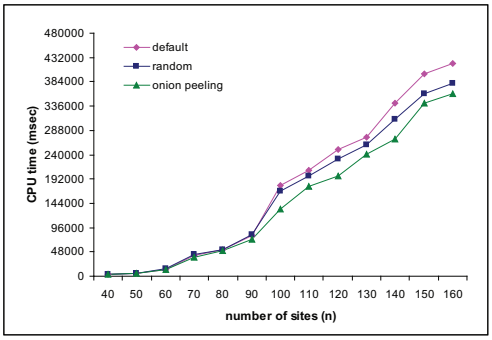
(a) HC: costs after 10000 msec



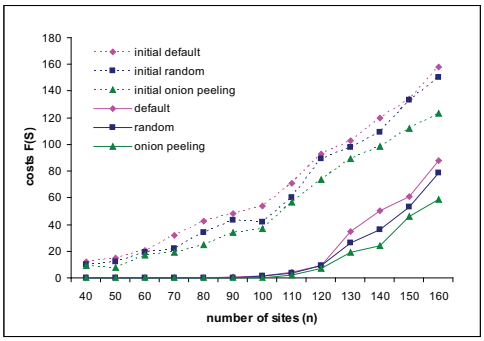
(b) HC: CPU time for 90% reduction



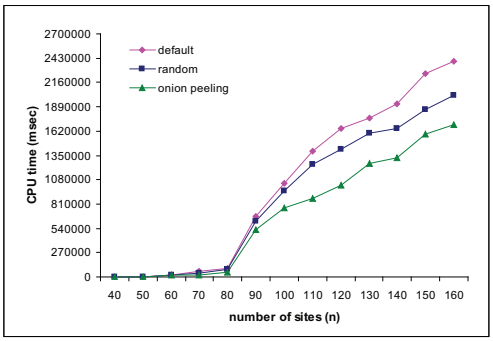
(c) SA: final costs for termination



(d) SA: CPU time for termination



(e) GA: final costs for termination

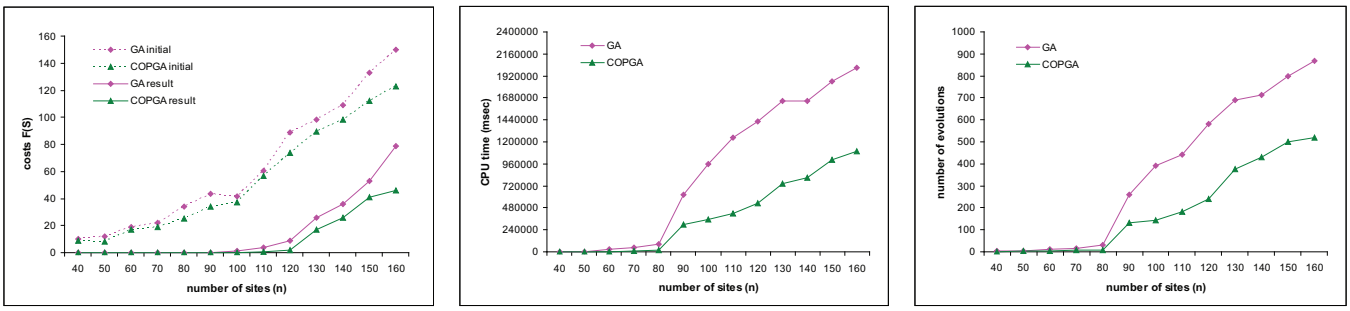


(f) GA: CPU time for termination

Figure 8: Synthetic data results of HC, SA and GA

real datasets	initialization method	initial cost	HC		SA	GA
			cost after 30000 msec	CPU for 80% reduction	final cost	final cost
dataset 1	default	47	0 (100%)	4212	2.5(95%)	0 (100%)
	random	43	0 (100%)	2371	2 (95%)	1 (98%)
	onion peeling	30	0 (100%)	2121	1 (97%)	0.5 (98%)
dataset 2	default	171	39 (77%)	15226	25 (85%)	21 (88%)
	random	157	33 (79%)	13195	21.5 (86%)	20 (87%)
	onion peeling	135	26 (81%)	12574	20 (85%)	17 (87%)
dataset 3	default	486	231 (52%)	95005	131 (73%)	125 (74%)
	random	402	201 (50%)	72431	128 (68%)	116 71%)
	onion peeling	346	153 (56%)	60710	121 (65%)	109 (68%)

Table 2: Real data result of HC, SA and GA: numbers in ( ) are reduction rate



(a) Costs (b) CPU time (c) Number of evolutions

Figure 9: Comparison of COPGA and GA w/synthetic datasets

real datasets (sites)	COPGA			GA		
	initial cost	result (reduction)	# evolutions	initial cost	result (reduction)	# evolutions
dataset 1	30	0 (100%)	22	43	1 (98%)	42
dataset 2	135	16 (88%)	243	157	20 (87%)	540
dataset 3	346	91 (74%)	312	402	116 (71%)	980

Table 3: Comparisons of COPGA and GA: cost, CPU time and number of evolutions w/real datasets

6. CONCLUSIONS

Map labeling of point-feature is proven to be of interest in many applications. Due to the difficulty of the problem, different heuristic solutions have been proposed in the literature. We proposed the Convex Onion Peeling Genetic Algorithm (*COPGA*) that utilizes the convex onion peeling data structure in initialization and evolution processes. Experimental results using synthetic and real datasets showed that the convex onion peeling initialization results in less conflicts compared to the random and the preferred-position initializations regardless of the algorithm used. Compared to a previously proposed genetic algorithm, *COPGA* clearly outperformed the other algorithm with respect to the running CPU time and the number of evolutions to reach a terminal condition with reduced conflicts in all cases.

7. REFERENCES

[1] J. Ahn and H. Freeman. A comparison of simple mathematical approaches to the placement of spot symbols. *Cartographica*, 24(3).

[2] J. Ahn and H. Freeman. A program for automatic name placement. *Cartographica*, 21(2/3).

[3] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry*, Vol. 36(3):215–236, 2007.

[4] S. Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, Vol. 31(4):609–517, 1995.

[5] J. Christensen, J. Marks, and S. Shieber. *Placing Text Labels on Maps and Diagrams*. Graphics Gems $\mathcal{IV}$. Academic Press, Cambridge, Mass, 1994.

[6] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, Vol. 14(3):203–232, 1995.

[7] J. Doerschler and H. Freeman. A rule-based system for dense-map name placement. *Communications of ACM*, 35(1).

[8] D. Dumitrescu, B. Lazzerini, L. Jain, and A. Dumitrescu. *Evolutionary Computation*. CRC Press, 1996.

[9] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proceedings of the 7th Annual Symposium on Computational Geometry*, pages 281–288, 1991.

[10] F. Hong, L. Kaijun, and Z. Zuxun. An efficient and robust genetic algorithm approach for automatic map labeling. In *Proceedings of International Cartographic Conference (ICC'05*, 2005.

[11] E. Imhof. Positioning names on maps. *The American Cartographer*, Vol. 2(2):128–144, 1975.

[12] S. Kirkpatrick, C. D. G. Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, Vol. 220(4598):671–680, 1983.

[13] J. Marks and S. Shieber. The computational complexity of cartographic label placement. In *Technical Report TR-05-91, Harvard University*, 1991.

[14] Z. Michalewicz. *Gentic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.

[15] G. R. Raidl. A genetic algorithm for labeling point features. In *Proceedings of the Intl. Conference on Imaging Science*, pages 189–196, 1998.

[16] USGS. Citiesx020 - u.s. national atlas cities and towns, 2004. `http://coastalmap.marine.usgs.gov/GISdata/basemaps/usa/cities/citiesx020.htm`.

[17] F. Wagner and A. Wolff. A practical map labeling algorithm. *Computational Geometry: Theory and Applications*, 7:387–404, 1997.