A Constructive Genetic Approach to Point-Feature Cartographic Label Placement

Missae Yamamoto* Luiz Antonio Nogueira Lorena[†]

INPE – Instituto Nacional de Pesquisas Espaciais
12.245-970 São José dos campos – SP, Brazil
*missae@dpi.linpe.br [†]lorena@lac.inpe.br

1 Introduction

Cartographic label placement refers to the text insertion process in maps and is one of the most challenging problems in geoprocessing and automated cartography [8]. Positioning the texts requires that overlap among texts be avoided, that cartographic conventions and preference be obeyed, that unambiguous association be achieved between each text and its corresponding feature and that a high level of harmony and quality be achieved. In this article we are concerned with the placement of labels for point features, approaching the problem from a combinatorial optimization viewpoint.

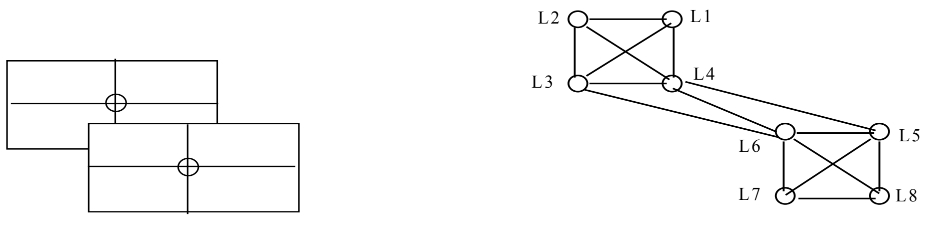

Figure 1: Two points – potential label positions and corresponding conflict graph

The Point-Feature Cartographic Label Placement (PFCLP) problem considers potential label positions for each point feature, which will be considered as candidates. Figure 1 shows two points with a set of four potential label positions and the corresponding conflict graph. The PFCLP considers the *placement of all labels* searching for the *large subset of labels with no* *conflict*. A related but different problem appears when label selection is permitted, and in that case the problem searches the maximum vertex independent set on the conflict graph [6].

Several heuristics and metaheuristics have been used to solve the PFCLP problem, such as greedy algorithms, discrete gradient descent, Hirsch's algorithm [2], Lagrangean relaxation [10], Simulated Annealing and others. They are reviewed by [1]. A Genetic Algorithm (GA) with mask is described in [7] and [9] presented a Tabu Search algorithm presenting better results in label placement quality than other methods.

This paper describes the application of a Constructive Genetic Algorithm (CGA) to PFCLP. The CGA was recently proposed by Lorena and Furtado [3] and applied to Gate Matrix Layout Problems [4, 5]. Basically it differs from other GAs for evaluating schemata directly. It also has a number of new features compared to a traditional GA. These include a population of dynamic size composed of schemata and structures, and the possibility of using heuristics in structure representation and in the fitness function definitions.

The paper is organized as follows. Section 2 presents a pseudo-code for the CGA and starts to describe aspects of modeling for schema and structure representations and the consideration of the PFCLP as a bi-objective optimization problem. Section 3 describes the some CGA operators, namely, selection and recombination. Section 4 shows computational results using instances formed by standard sets of randomly generated points suggested in the literature.

2 CGA Modeling for PFCLP

The CGA is usually modeled as a bi-objective optimization problem indirectly solved by an evolutionary process.

Let X be the space of all schemata and structures s_k = (label or #, label or #, …, label or #), that can be created using the alphabet {label, #}, where label can be any corresponding point label (in Figure 1: L1, …, L8), and # is an indetermination typical of schemata. The PFCLP is modeled using two fitness functions. Function f returns the number of conflict free labels in s_k and g returns the corresponding number after a local search in s_k. The variation ($g(s_k) - f(s_k)$) reflects the local search improvement and schemata have lower absolute values than structures (schemata instantiations).

The population is controlled by an evolution parameter (time) α, receiving small increments from generation to generation. The corresponding population is denoted by P_α, and have a number of schemata and structures, ranked by the following expression $\delta(s_k) = \dfrac{d \cdot g_{\max} - [g(s_k) - f(s_k)]}{d[g_{\max} - g(s_k)]}$,

$g_{max} \neq g(s_k)$, where g_{max} is the total number of points and $0 < d \leq 0.2$. It is dynamic in size controlled by α , and can be emptied during the process. At the time they are created, structures and/or schemata receive their corresponding rank value $\delta(s_k)$, that are compared with the current evolution parameter α ($\alpha > \delta(s_k)$?). Better schemata or structures have high ranks, reflected by small variations $(g(s_k) – f(s_k))$ and/or greater $g(s_k)$ values. In this sense better individuals are those trained by the local search heuristic and have large number of conflict free labels.

The algorithm implemented in this work is showed as follows.

```
CGA { Constructive Genetic Algorithm }
        Given $g_{max}$ , d and $\alpha$;
        Initialize $P_\alpha$ ;
        for all $s_k \in P_\alpha$ compute $g(s_k)$, $f(s_k)$, $\delta(s_k)$;
        while (not stop condition) do
                while (number of recombination) do
                        Select $s_{base}$ and $s_{guide}$ from $P_\alpha$ ;
                        Recombine $s_{base}$ and $s_{guide}$;
                        Evaluate Offspring;
                        Update Offspring in $P_{\alpha+1}$;
                end_while
                for all $s_k \in P_\alpha$ satisfying $\alpha > \delta(s_k)$ do
                        Eliminate $s_k$ from $P_\alpha$ ;
                end_for
                if $s_{base}$ is a schema then
                        Change the #s to labels giving $s_{base-struct}$;
                        Evaluate $s_{base-struct}$;
                end_if
                update $\alpha$ ;
        end_while
```

The best $g(s_k)$ is kept in the process. The initial population, selection, transforming a schema base in structure ($s_{base-filled}$), recombination, α updating and the local search heuristic are detailed in the following.

3 CGA operators

The *initial population* is composed exclusively of schemata, considering that for each schema, a proportion of random positions receive a unique label number. The remaining positions receive labels #. Along the generations, the population increases by addition of new offspring generated out of the combination of two schemata.

For *selection*, the structures and schemata in population P_α are maintained in descending order of ranks. Two structures and/or schemata are selected for recombination. The first is called the base (s_{base}) and is randomly selected out of the first positions in P_α, and in general it is a good structure or a good schema. The second structure or schema is called the guide (s_{guide}) and is randomly selected out of the total population. The objective of the s_{guide} selection is the conduction of a guided modification on s_{base}.

In the *recombination* operation, the current labels in corresponding positions are compared. Let s_{new} be the new structure or schema (offspring) after recombination. Structure or schema s_{new} is obtained by applying the following operations:

```
{ Recombination } - Recombination with mask (based in Verner et al. (1997):
        $M_{base}$   =   mask of $s_{base}$ (0 = conflict, 1 = without conflict, 2 = #)
        $M_{guide}$  =   mask of $s_{guide}$ (0 = conflict, 1 = without conflict, 2 = #)
        $U$          =   0 or 1 (randomly generated)
        Repeat for each position (j) in structure or schemata representation:
        If $M_{base}(j) = 1$ then $s_{new}(j) \leftarrow s_{base}(j)$
        If $M_{base}(j) \neq 1$ and $M_{guide}(j) = 1$ then $s_{new}(j) \leftarrow s_{guide}(j)$
        If $M_{base}(j) \neq 1$ and $M_{guide}(j) \neq 1$ and $U = 0$ then $s_{new}(j) \leftarrow s_{guide}(j)$
        If $M_{base}(j) \neq 1$ and $M_{guide}(j) \neq 1$ and $U = 1$ then $s_{new}(j) \leftarrow s_{base}(j)$
```

If s_{base} is a schema, it is *transformed in structure* changing #s by labels. The labels are positioned searching small number of extra conflicts.

Considering that the well-adapted individuals need to be preserved for recombination, the *evolution parameter α* is started from the lowest δ value (taken from the bottom of the ranked population), and then increases with step proportional to actual population size $|P_\alpha|$, $\alpha = \alpha + k \cdot |P_\alpha| \cdot \frac{\delta_{top} - \delta_{bot}}{Gr} + l$, where k is a proportionality constant, l is the minimum increment allowed, Gr is the remaining number of generations, and (δ_{top}-δ_{bot}) is the actual range of values of δ. One can observe that the adaptive increment of α is affected by the own environment (population size, best and worst δ's, etc). Thus, once the CGA achieves very good regions and does not get to improve the best rank, the parameter α goes eliminating the individuals until the population is emptied.

The *local search heuristic* is used to calculate $g(s_k)$ and drives the evolution process to a trained population. The heuristic used in this paper is very simple (other sophisticated heuristics can be tried). It takes the subset of labeled points in s_k (points without #s) and their original potential label conflicts, i. e., a subgraph of the conflict graph. The vertices of the subgraph are then considered in non-decreasing order of degrees. The corresponding point receives a label with no conflict and the subgraph is updated eliminating this label (vertex) and their incident vertices and edges. The process is then repeated until no more labels to place.

4 Computational results

Christensen et al. [1], Verner et al. [7] and Yamamoto et al. [9] compared several algorithms using standard sets of randomly generated points: grid size of 792 by 612 units, fixed size label of 30 by 7 units and page size of 11 by 8.5 inch. In order to compare the CGA algorithm with previous work, we used the standard sets of randomly generated points and simulated the same conditions as described by [1] and followed the same assumptions as [7] (the instances used in this paper are available at www.lac.inpe.br/~lorena/instancias):

- Number of the points: N = 100, 250, 500, 750, 1000;
- For each problem size, we generated 25 different configurations with random placement of point feature using different seeds;
- For each problem size, we calculated the average percentage of labels placed without conflict of the 25 trials;
- No penalty was attributed for labels that extended beyond the boundary of the region;
- 4 potential label positions were considered;
- Cartographic preferences were not taken into account;
- No point selection was allowed (i.e., no points are removed even if avoiding superposition is inevitable);
- The parameters used for CGA are presented in Table 1.

Regarding the optimization algorithms of the literature, the CGA showed superior results in quality of label placement. Table 2 shows the percentage of labels placed without conflict for 100, 250, 500, 750 and 1000 points, considering different algorithms of the literature. The CGA$_{\text{-average}}$ reports the average result for 6 trials and CGA$_{\text{-best}}$ is the best result. The lines show the percentage of labels placed without conflict by the optimization algorithms tested on [1] (greedy-depth first, gradient descent, 2-opt gradient descent, 3-opt gradient descent, Hirsch, Zoraster and simulated annealing), [7] (GA without masking and GA with masking), on Yamamoto [9] (Tabu search) and the CGA.

Table 3 compares the Tabu search of [9], CGA$_{\text{-best}}$ and CGA$_{\text{-average}}$ running times to obtain the best solutions (using a Pentium III computer). The Tabu search approach is faster than CGA.

5 Conclusion

This work has proposed and evaluated a CGA applied to the PFCLP problem. By using a standard set of randomly generated points and the same conditions described by [1], [7] and [9], we have shown that CGA has better results in label placement quality than other methods published in the literature.

Acknowledgments: The second author acknowledges Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (process 300837/89-5) for partial financial research support.

References

1. Chistensen, J.; Marks, J.; Shieber, S. *An empirical study of algorithms for point-feature label placement.* ACM Transactions on Graphics, v. 14, n. 3, p. 203-232, 1995.
2. Hirsch, S. A. *An algorithm for automatic name placement around point data*. American Cartographer, v. 9, n. 1, p. 5-17, 1982.
3. Lorena, L.A.N.; Furtado J.C. *Constructive genetic algorithm for clustering problems*. Evolutionary Computation. 9(3):309-327. 2001.
4. Oliveira A.C.M.; Lorena L.A.N. *A Constructive Genetic Algorithm for Gate Matrix Layout Problems*. IEEE Transactions on Computer-Aided Designed of Integrated Circuits and Systems. Vol. 21, No. 8, pp. 969-974, 2002.
5. Oliveira, A.C.M.; Lorena, L.A.N. *2-Opt Population Training for Minimization of Open Stack Problem.* Advances in Artificial Intelligence - XVI Brazilian Symposium on Artificial Intelligence. G. Bittencourt and G. L. Ramalho (Eds).LNAI 2507. Springer. pp.313-323. Porto de Galinhas/Recife. 2002.
6. Strijk, T.; Verweij, B.; Aardal, K. *Algorithms for Maximum Independent Set Applied to Map Labeling. September*, 2000. 42p. Available at ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2000/2000-22.ps.gz
7. Verner, O. V.; Wainwright, R. L.; Schoenefeld, D. A. *Placing text labels on maps and diagrams using genetic algorithms with masking*. INFORMS J. on Computing, v. 9, p. 266-275, 1997.
8. Wolff, A.; Strijk, T. *A Map Labeling Bibliography.* http://www.math-inf.uni-greifswald.de/map-labeling/bibliography/, 1996.
9. Yamamoto, M., Camara, G. and Lorena, L. A. N. *Tabu search heuristic for point-feature cartographic label placement.* GeoInformatica. Kluwer Academic Publisher,

Netherlands, 6:1, 77-90, 2002.

10. Zoraster, S. *The solution of large 0-1 integer programming problems encountered in automated cartography*. Operations Research, v. 38, n. 5, p. 752-759, Sept.-Oct. 1990.

N	**100**	**250**	**500**	**750**	**1000**
Initial population size	30	75	150	200	300
k	0.01	0.01	0.01	0.01	0.01
l	0.00001	0.00001	0.00001	0.00001	0.00001
Offspring	30	30	30	30	30
Number of generations	25	25	25	25	25
d	0.2	0.2	0.2	0.2	0.2
Number of # (initial population)	1	2	10	30	70
% of sub-population Base	15%	15%	15%	15%	15%

Table 1: CGA parameters.

Algorithm	**100**	**250**	**500**	**750**	**1000**
$\text{CGA}_{\text{-best}}$	100.00	100.00	99.6	97.1	90.7
$\text{CGA}_{\text{-average}}$	100.00	100.00	99.6	96.8	90.4
Tabu search [9]	100.00	100.00	99.2	96.8	90.00
GA with masking [7]	100.00	99.98	98.79	95.99	88.96
GA [7]	100.00	98.40	92.59	82.38	65.70
Simulated Annealing [1]	100.00	99.90	98.30	92.30	82.09
Zoraster [10]	100.00	99.79	96.21	79.78	53.06
Hirsch [2]	100.00	99.58	95.70	82.04	60.24
3-Opt Gradient Descent [1]	100.00	99.76	97.34	89.44	77.83
2-Opt Gradient Descent [1]	100.00	99.36	95.62	85.60	73.37
Gradient Descent [1]	98.64	95.47	86.46	72.40	58.29
Greedy [1]	95.12	88.82	75.15	58.57	43.41

Table 2: Computational results.

Algorithm	**100**	**250**	**500**	**750**	**1000**
$\text{CGA}_{\text{-best}}$	0	0.6	21.5	228.9	1227.2
$\text{CGA}_{\text{-average}}$	0	0.6	21.5	195.9	981.8
Tabu search [9]	0	0	1.3	76.0	352.9

Table 3 – Computational times to reach the best solutions