Research Article

Evolutionary Computation for Label Layout on Unused Space of Stacked Graphs

Alejandro Toledo,¹ Kingkarn Sookhanaphibarn,¹ Ruck Thawonmas,¹ and Frank Rinaldo²

¹DHCJAC, Ritsumeikan University, Kuatsu, Shiga 525-877, Japan

²ISE, Ritsumeikan University, Kuatsu, Shiga 525-877, Japan

Correspondence should be addressed to Alejandro Toledo, alex@ice.ci.ritsumei.ac.jp

Received 22 November 2011; Accepted 22 December 2011

Academic Editors: W. Lam and C. Nikou

Copyright © 2012 Alejandro Toledo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Placing numerous objects and their corresponding labels in the stacked graph visualization is a challenging problem. In the stacked graph, different combinations of initial parameters and filtering effects yield views with hidden information, illegible labels, and unused space. The result is a tool that does not take advantage on the unused space to reveal information to the user for further investigation. We present an automatic method for label layout on the unused space in a stacked graph. An evolutionary computation (EC) is used to optimize the best label position according to legibility requirements, as well as requirements for keeping semantic relationships between labels and their representative visual objects. A number of EC experiments, as well as a usability study on label legibility, show that our proposed solution looks promising, as compared to the traditional solutions.

1. Introduction

Designing effective information visualization systems with abundant associated labels is a difficult task, especially when the screen space is limited. Label layout is an essential task in the field of information visualization to identify the visualized data and different visual objects. The challenge is to find a global labeling solution without any overlapping of other labels. Since labels are used to explain the data, the solution is reduced not only to minimize the overlapping, but also to keep semantic relationships between labels and their corresponding visual objects. The label layout problem, which is proved to be NP-hard, has been extensively studied. In cartographic design, for instance, label legibility is the most important requirement [1].

2. Related Work

In this paper, we use and extend the stacked graph visualization as proposed in [2, 3], where a collection of job occupations is visualized using stripes representing occupation's time series (Figure 1). Since the data values are normalized, the heights of individual stripes add up to the height of

the overall graph; thus, there can be no spaces between the stripes because this would distort their sum. In the case of stacked graphs with many occupations, this may cause the individual stripes to be harder to read. The legibility of the data is, indeed, an unsolved design consideration of the stacked graph [4].

Colors have been used to distinguish stripes without using heavy borders [2]. Another proposed solution in [2] is to use a threshold for label visibility (TLV), which is the minimum width, in pixels, for a stripe to receive a label. However, in a previous experiment [5], we found that the TLV makes no significant improvement on data legibility. Moreover, as for the obtained results, the lack of labels on some of the stripes was the main reason they found on the stacked graph illegibility. In this work, we use the TLV to compare our proposed system with two versions of the traditional stacked graph.

Maximizing the display of data content in a comprehensible way is a problem that has been addressed by many researchers. Li et al. [6] implemented a set of techniques in LifeLines, a visualization environment as applied to medical patient records. The authors presented algorithms and metrics based on map-oriented techniques.

FIGURE 1: Stacked graph view (left) showing 40 job occupation names starting with the prefix “A”, and TLV = 0 pixels. A number of labels are illegible due to either overlapping or small size. The solid line encloses the visualization area, and the dotted line the unused space. At the right, an increased section of the same view (top) with TLV = 12 pixels (some labels are hidden). The same section (bottom) can be seen with TLV = 0 (some labels are overlapped) and with their minimum font size increased three times.

Text labels either overlay visual objects or placed outside (internal versus external labels). Works supporting the idea to use “call-outs” or “leaders” connecting external labels to visual objects can be found on hand-made illustrations, as in [7, 8], or for labeling server motherboards, as in [9].

Works on aesthetic layouts to support the legibility of stacked graphs can be found in [4, 10]. Here, the author’s proposal is on the use of different shapes or stripes, as well as coloring. Hartman et al. [11] developed a similar idea with the use of external labels.

Studies on labeling and label legibility have been conducted to support information visualizations. Talbot et al. [12] extended Wilkinsons optimization approach to produce labelings on axis in a two-dimensional graph. Luboschik et al. [13] present a labeling method using fast point-feature, to support information visualization systems. An interesting experiment on label readability can also be found in [14], where the authors investigated the readability of medication labels. The contributions of this paper are (i) a new component, based on evolutionary computation, for augmenting the readability of labels in the stacked graph visualization, (ii) an adjustment to the PMX crossover operator for permutation of label positions, (iii) an adjustment to the SWAP mutation operator for replacement of label positions, and (iv) a usability study on label legibility proving the effectiveness of our method.

3. Evolutionary Computation Design

Using a grid layout, the visualization area is divided into a collection of equal partitions. The center point of each partition is the actual absolute position. Each partition has a unique identifier, and those not intersecting with the stacked area—in the unused space—are considered as the candidate partitions. Figure 2 shows the partitions obtained from the stacked graph of Figure 1. A 5×8 grid layout (5 rows and 8 columns) was used; leading to 16 candidate partitions (circles colored blue). To support a more efficient solution encoding, candidate partitions are reindexed, for example, in Figure 2 from 0 to 15.

FIGURE 2: Partition of the visualization area of the stacked graph of Figure 1. Partitions in blue color correspond to the candidate partitions for label layout.

TABLE 1: Chromosome encoding.

Label 1	...	Label $ L $
$\Gamma_{ P }$ bits	...	$\Gamma_{ P }$ bits

3.1. Chromosome Encoding. We encode a label layout design in one chromosome, using variable-length, fixed-order binary encoding. Given a set of illegible labels L , and a set of candidate partitions P , a chromosome (Table 1) is produced using $|L|$ genes. Under this representation, each gene consists of $\Gamma_{|P|}$ bits indicating the candidate partition for positioning the corresponding label. We define Γ_n using the following recurrence:

$$\Gamma_0 = 0;$$

$$\Gamma_n = \gamma_n + \Gamma_{n-2^{\gamma_n}}. \quad (1)$$

Here, $\gamma_n = \lfloor \log_2 n \rfloor$ denotes a locus encoding a partial gene evaluation.

The evaluation of the i th gene for the candidate label $l_i \in L$ will then yield a partition, upon which positioning l_i on the unused space. Therefore, all partitions derived from the complete chromosome characterize a label layout for all selected illegible labels.

As an example, consider two layouts represented by the following two chromosomes, where six labels are to be placed on the unused space of the stacked graph of Figure 1, using the partitions of Figure 2:

Chromosome A: 0000 0110 0010 1000 0111 1001
Chromosome B: 1001 0101 0110 0011 0001 0010

Our approach is similar to several proposed GAs for reordering or permutation problems. The most famous is the traveling salesman problem (TSP) in which a salesman wants to visit a number of cities while traveling the least possible distance. In the TSP, each city must be represented once and only once; and the fitness of the chromosome is related to the tour length, which in turn depends on the ordering of the cities. Our solution is comparable to the TSP in the encoding method, that is, we prefer partitions to be represented once and only once in the chromosome. This is because their repetitions would lead to overlapping, and thus infeasible solutions. Our label layout problem and TSP differ, however, in the number of encoded elements. In the TSP, all cities are represented in a chromosome, while our solution encodes only a subset of the candidate partitions. This will require an adjustment of the crossover and mutation operators designed for the TSP, which we describe later in the following sections.

3.2. Fitness Function. We decided that the quality of label layout depends on the fitness of candidate partitions for the illegible labels. We enforce appropriate label arrangements on the unused space using the following criteria:

F_1 : Minimize the overlapping between labels,

F_2 : Maximize the correlation between the label order and the stripe order.

The fitness value for F_1 for individual i is inversely proportional to the number of labels with overlapping:

$$F_1(i) = 1 - \text{overlapping}(i),$$

$$\text{overlapping}(i) = \frac{\Phi_i}{|L|}, \quad (2)$$

where Φ_i is the number of labels, of individual i , with overlapping. Here, two labels are overlapped if the intersection between their bounding boxes is greater than zero.

The fitness value for F_2 is proportional to the correlation between the label order and stripe order. For each individual i , we calculate two tuples. A tuple of partition positions (PP) is obtained from the absolute y -coordinates of the encoded partitions, and a tuple of stripe positions (SP) is the order of the L illegible labels under consideration in the current stacked graph. The two tuples are used to reward a

correlation ratio between the label order and the stripe order using the following equation:

$$F_2(i) = \frac{\text{pc}(\text{PP}, \text{SP}) + 1}{2}, \quad (3)$$

where pc denotes the Pearson product moment correlation coefficient. Note that $|\text{PP}| = |\text{SP}| = |L|$. Finally, the total fitness value for individual i is obtained as

$$\text{fitness}_i = w_1 F_1(i) + w_2 F_2(i), \quad (4)$$

where w_x is a weighting ratio for criterion F_x .

3.3. Genetic Algorithm and Genetic Operators. We used the following genetic algorithm.

- (1) Create an initial population of C chromosomes (generation 0)
- (2) Evaluate the fitness of each chromosome
- (3) Select the fittest chromosome
- (4) Excluding the chromosome selected in step 3, select $C - 1$ parents from the current population via roulette wheel selector
- (5) Choose at random a pair of parents for mating. Apply crossover operator
- (6) Process each offspring by the mutation operator, and insert the new offspring into the new population
- (7) Repeat steps 5 and 6 until $C - 1$ offsprings are created
- (8) Replace the old population of chromosomes with the new population
- (9) Add the chromosome selected in step 3 into the new population
- (10) Evaluate the fitness of each chromosome in the new population
- (11) Increase the generation number and go back to step 3 if this number is less than some upper bound.

The above algorithm introduces concepts such as crossover and mutation operator. In this work we used the following four operators.

(1) *Single Point Crossover (C1).* The single point crossover operator is aimed at exchanging bit strings between two parent chromosomes. A random position between 1 and $|L| - 1$ is chosen along the two parent chromosomes, where L is the chromosome's length. Then, the chromosomes are cut at the selected position, and their end parts are exchanged to create two offspring. For instance, the two chromosomes of the previous section are selected to produce two offspring as follows.

parent A: 06|2879 offspring A': 066312
parent B: 95|6312 offspring B': 952879

It should be noted that single-point crossover cannot guarantee representations with no repeated values.

(2) *Partially Matched Crossover, PMX, (C2)*. PMX [15] arose in considering ways to tackle issues associated with the classic traveling salesman problem (TSP). Given two parents A and B , PMX randomly picks two crossover points. The offspring is then constructed in the following way. Starting with a copy of A , the positions between the crossover points are, one by one, set to the values of B in these positions. To keep the string a valid chromosome, the cities in this positions are not just overwritten. To set position p to city c , the city in position p and city c swap positions. Consider the example below of this encoding for the two tours A : 6, 2, 3, 4, 1, 7, 5 and B : 5, 2, 4, 1, 3, 7, 6.

swap 4 and 1	swap 3 and 4	no change	first offspring
623 417 5	623 1475	6241375	$A' : 6241375$
↑	↑	↑	
524 137 6	5241376	5241376	
623 417 5	6234175	6234175	
↓	↓	↓	
524 137 6	5214376	5234176	$B' : 5234176$
swap 4 and 1	swap 1 and 3	no change	second offspring

Since TSP chromosomes encode all cities, the city in position p and city c can always be exchanged, for example, cities 1, 3, 7 and 4, 1, 7 could be found in the copies of A and B , respectively. In our domain, however, a partition in position p cannot always find its pair within the same parent. So, we propose to solve this by replacing such partition with the one in position p in the other parent. Consider the example below of this encoding for the two chromosomes A : 0, 6, 2, 8, 7, 9 and B : 9, 5, 6, 3, 1, 2.

swap 6 and 2	replace 8 with 3	first offspring
06 28 79	026879	026379
↑	↑	
95 63 12	956312	
06 28 79	062879	
↓	↓	
95 63 12	952316	952816
swap 6 and 2	replace 3 with 8	second offspring

(3) *Simple Mutation (M1)*. The bits of the two offspring generated by the crossover operator are then processed by the mutation operator. This operator is applied to each bit with a small probability (e.g., 0.001). When it is applied to a given position, the new bit value switches from 0 to 1 or from 1 to 0. Here, it should also be noted that the simple mutation operator cannot guarantee representations with no repeated values.

(4) *Swap Mutation (M2)*. Mutation operators for the TSP are aimed at randomly generating new permutation of cities. As opposed to the simple mutation operator, which introduces small perturbations into the chromosome, the permutation operators for the TSP often greatly modifies the original tour. One example is the swap mutation operator, where two cities are selected and swapped (i.e., their positions are exchanged). Since TSP chromosomes encode all cities, the selection of two of them is always possible. In our domain, however, there

may be cases where this is not possible. Here, as part of our second adjustment, we propose to perform this selection by choosing one partition from the chromosome, and the other from the set of candidate partitions P . The latter is randomly selected until a chromosome with no repeated values can be formed. Then, this partition replaces the one in the chromosome.

4. Experiments

Aiming at solving the overlapping problem among labels ($w_1 = 1, w_2 = 0$), our goal here was to compare the performance of the four combinations C1M1, C1M2, C2M1, and C2M2, all in terms of the time required to find a solution and the quality of the solution obtained. In order to be fair to the four operators, as well as to be thorough in our experiment, we compared the results among combinations using scaled problems through the use of different grid layout resolutions.

4.1. *Experimental Setup*. We selected 25 illegible labels describing job occupation names from the view of Figure 1. The illegibility criterion was based on font size, where the smaller the font size, the most illegible the label. These labels have different lengths and, thus, different bounding boxes. To improve their legibility, we increased by 40% the base minimum size of each label. We also used five different grid layouts for partitioning the visualization area, 10×10 , 12×12 , 14×14 , 16×16 , and 18×18 ; leading to 34, 49, 71, 97, and 134 candidate partitions, respectively. As for crossover operators, we used single point (C1) and the adjusted PMX crossover (C2), both using a crossover rate of 0.6. As for mutation operators, we used simple (M1) and the adjusted swap mutation (M2), both using a mutation rate of $1/|L|$. Finally, we used the following five different values for C (population size): 25, 50, 75, 100, 125.

All candidate partitions, genetic operators (C1, C2, M1, M2), and population sizes were combined to produce 100 types of parameter selections for running the genetic algorithm. We call each type a “configuration.” Thus, for this experiment, we have the configuration tuples (conf1, 34, C1, M1, 25), ..., (conf4, 34, C2, M2, 25), ..., and (conf100, 134, C2, M2, 125). To produce a fair comparison among configurations, we reuse populations and create new ones at given intervals. Since populations can be reused only for tuples with the same number of candidate partitions, a new random initial population is created when the configuration number is a multiple of 4, otherwise, it is reused. Finally, each configuration was run using 500 generations, and all reported values are the results of 10 independent runs.

4.2. *Results*. Two metrics were used to measure the results.

(1) Quality of Solutions (QS_n)

QS_1 : Fitness value obtained at generation 500.

QS_2 : Difference between QS_1 and the initial fitness value.

(2) Convergence Speed (CS_n)

CS_1 : Best-fitness average of all generations [16]

$$\frac{1}{T} \sum_{j=1}^T f_j. \quad (5)$$

CS_2 : Difference between CS_1 and the initial fitness value.

Since we have different initial fitness values (25 in total), QS_2 and CS_2 will let us know the quality and convergence speed, respectively, of the solutions with respect to the evolution of populations.

In terms of QS_1 , the best configuration was obtained from a grid layout of 14×14 (71 candidate partitions), population size C of 50 individuals, adjusted PMX crossover (C2), and adjusted SWAP mutation (M2); with a fitness average of $QS_1 = 0.944$, as shown in Table 2.

In terms of QS_2 , the best configuration was obtained from a grid layout of 16×16 (97 candidate partitions), population size C of 25 individuals, adjusted PMX crossover (C2), and adjusted SWAP mutation (M2); with an improvement of $QS_2 = 0.448$.

In terms of both CS_1 and CS_2 , the best configuration was obtained from a grid layout of 14×14 (71 candidate partitions), population size C of 125 individuals, adjusted PMX crossover (C2), and adjusted SWAP mutation (M2); with a convergence speed $CS_1 = 0.849$ and $CS_2 = 0.329$.

Our results show that our proposed operators, in this domain, outperform the standard ones. Also, as from our observations of the second and the third best results, the combination C2M2 outperforms, 99% of the time, the other two operators. Another observation is that our best results of C2M2 were concentrated on configurations sharing the grid layout 14×14 (71 candidate partitions). This suggests that our proposed method has no significant improvement by increasing the number of candidate partitions. Since we were also interested in the time required to obtain the best solution, we also observed the convergence speeds of our proposed method. Here, the convergence speed of the best QS_1 , $CS_1 = 0.841$, is the third best convergence speed. Moreover, it has no significant difference with the best convergence speed $CS_1 = 0.849$. This number is associated to the group of 71 candidate partitions (14×14) and was also obtained from the combination C2M2. In general, this can be explained by the variations in the number of individuals $50 \leq C \leq 125$, which confirms previous results, such as those of De Jong test functions [16].

Figure 3 shows the performance comparison among C1M1, C2M2, C2M2, and C2M2. Best fitness value versus generation number is plotted for all configurations. The experiment has been repeated 10 times to plot average values, using different maximum generation numbers. Because of space limitations, the plots for only the best QS_1 , QS_2 , CS_1 , and CS_2 are given.

Figure 4 shows contour plots, where the best results for the three variables (i) populations size, (ii) candidate

partitions, and (iii) fitness value, are shown. It can be observed that where M2 is used, better fitness values are obtained, and also lower numbers of individuals are required, which suggests better computational performance. Moreover, where the combination C2M2 was used, the best fitness values were obtained.

Figure 5 shows stacked graph views produced by our genetic algorithm. Figure 5(a) show the result of experiment 3 d), as applied to Figure 1, where the weights $w_1 = w_2 = 0.5$ was used. Figure 5(b) and Figure 5(c) show the results of views with prefixes “C” and “F”, respectively.

5. Usability Study

We conducted a usability study with 30 subjects to investigate the readability of labels using 9 views extracted from 3 different versions of the stacked graph visualization. Our goal here was to compare the performance of the three versions, all in terms of subject reading speeds, as well as subject accuracy.

5.1. Subjects. Participants in this study were computer science students taking courses at our university and consisting of 23 undergraduate students and 7 graduate students with the average age of 22.17, ranging from 20 to 26 years old. All subjects were selected from a test on stacked graph usage using a questionnaire for visual tasks [17]. The scores obtained from the task were used to divide the subjects into three groups; where each group contains 4 subjects with high score, 3 with score average, and 3 with low score. To avoid bias from ordering effects, each subject group was tested using different version orderings.

5.2. Material. The materials used were as follows.

- (I) Three versions of the stacked graph visualization. One version (V1) with TLV = 12 (threshold for label visibility), another version (V2) with TLV = 0, and our proposed system (V3).
- (II) Nine views obtained from the three aforementioned versions; three views from each version, from which the first group, view_a, corresponds to the views with prefix “A” Figure 5(a), the second group, view_c, to the prefix “C” Figure 5(b), and the third group, view_f, to the prefix “F” Figure 5(c).
- (III) Twenty five illegible labels for each prefix, with variations in their font size.

5.3. Design. The study was a 3 (stacked graph version) $\times 3$ (version orderings) $\times 3$ (view prefix) mixed-model design. The between-subject factor was version ordering (V1V2V3, V2V3V1, V3V1V2), and the within-subject factor was view prefix (view_a, view_c, view_f).

5.4. Procedure. We prepared a task involved in searching, using the features of the visualization system, for labels with different levels of legibility. Six labels, describing occupation

TABLE 2: Comparison of configurations C1M1, C1M2, C2M1, C2M2 on five different numbers of candidate partitions (left side) and five numbers of population sizes. The best result among candidate partitions is emphasized in boldface. For each candidate solution number, the first row corresponds to the QS_1 measurement, the second to QS_2 , the third to CS_1 , and the last one to CS_2 .

	25				50				75				100				125			
	C1M1	C1M2	C2M1	C2M2	C1M1	C1M2	C2M1	C2M2	C1M1	C1M2	C2M1	C2M2	C1M1	C1M2	C2M1	C2M2	C1M1	C1M2	C2M1	C2M2
34	0.360	0.656	0.420	0.640	0.404	0.632	0.472	0.648	0.424	0.632	0.484	0.672	0.476	0.688	0.504	0.660	0.484	0.648	0.504	0.676
	0.040	0.336	0.100	0.320	0.044	0.272	0.112	0.288	0.024	0.232	0.084	0.272	0.004	0.208	0.024	0.180	0.004	0.168	0.024	0.196
	0.359	0.567	0.418	0.564	0.401	0.528	0.469	0.575	0.424	0.556	0.482	0.586	0.475	0.605	0.502	0.595	0.483	0.571	0.501	0.604
	0.039	0.247	0.098	0.244	0.041	0.168	0.109	0.215	0.024	0.156	0.082	0.186	0.005	0.125	0.022	0.115	0.003	0.091	0.021	0.124
49	0.412	0.680	0.448	0.692	0.396	0.724	0.456	0.728	0.440	0.724	0.512	0.724	0.460	0.672	0.516	0.716	0.500	0.684	0.544	0.744
	0.052	0.320	0.088	0.332	0.036	0.364	0.096	0.368	0.040	0.324	0.112	0.324	0.020	0.232	0.076	0.276	0.100	0.284	0.144	0.344
	0.411	0.571	0.446	0.596	0.395	0.597	0.454	0.615	0.439	0.626	0.510	0.643	0.457	0.596	0.514	0.628	0.498	0.584	0.540	0.641
	0.051	0.211	0.086	0.236	0.035	0.237	0.094	0.255	0.039	0.226	0.110	0.243	0.017	0.156	0.074	0.188	0.098	0.184	0.140	0.241
71	0.600	0.864	0.656	0.876	0.576	0.896	0.632	0.944	0.616	0.908	0.668	0.928	0.652	0.876	0.712	0.900	0.620	0.892	0.752	0.932
	0.080	0.344	0.136	0.356	0.016	0.336	0.072	0.384	0.056	0.348	0.108	0.368	0.028	0.196	0.032	0.220	0.100	0.372	0.232	0.412
	0.598	0.768	0.654	0.767	0.575	0.801	0.631	0.841	0.615	0.798	0.666	0.845	0.648	0.799	0.708	0.824	0.618	0.807	0.746	0.849
	0.078	0.248	0.134	0.247	0.015	0.241	0.071	0.281	0.055	0.238	0.106	0.285	0.032	0.119	0.028	0.144	0.098	0.287	0.226	0.329
97	0.520	0.852	0.584	0.888	0.572	0.868	0.600	0.916	0.580	0.896	0.636	0.880	0.612	0.856	0.636	0.916	0.580	0.848	0.688	0.880
	0.080	0.412	0.144	0.448	0.052	0.348	0.080	0.396	0.020	0.336	0.076	0.320	0.132	0.376	0.156	0.436	0.020	0.288	0.128	0.320
	0.518	0.731	0.581	0.764	0.570	0.762	0.598	0.821	0.579	0.782	0.634	0.799	0.607	0.734	0.632	0.803	0.578	0.764	0.681	0.803
	0.078	0.291	0.141	0.324	0.050	0.242	0.078	0.301	0.019	0.222	0.074	0.239	0.127	0.274	0.152	0.323	0.018	0.204	0.121	0.243
134	0.448	0.744	0.480	0.748	0.496	0.828	0.568	0.840	0.516	0.832	0.548	0.844	0.500	0.772	0.584	0.804	0.568	0.812	0.600	0.832
	0.088	0.384	0.120	0.388	0.064	0.268	0.008	0.280	0.036	0.352	0.068	0.364	0.060	0.332	0.144	0.364	0.008	0.252	0.040	0.272
	0.447	0.643	0.478	0.662	0.496	0.735	0.567	0.720	0.514	0.720	0.546	0.727	0.497	0.662	0.580	0.706	0.565	0.703	0.595	0.724
	0.087	0.283	0.118	0.302	0.064	0.175	0.007	0.160	0.034	0.240	0.066	0.247	0.057	0.222	0.140	0.266	0.005	0.143	0.035	0.164

FIGURE 3: Performance comparison among C1M1, C1M2, C2M1, and C2M2. Best fitness value versus generation number is plotted for three configurations.

names, were to be found using the mouse hovering feature of the stacked graph, where an information box was shown to the subjects while they were moving the mouse on the visualization area. For this experiment, subjects could see in the information box both the searched label and its corresponding population number; as shown in the views of Figure 5. In the beginning, each participant was given a questionnaire containing the question “*what is the population*

number for each of the following occupation names?” and six occupation names, sorted from easy to difficult in terms of legibility. Subjects had a limited time of one minute to write their answers.

5.5. *Results.* A 3 (stacked graph version) $\times$ 3 (version ordering) $\times$ 3 (view prefix) mixed model analysis of variance was performed on the reading speed data and indicated

FIGURE 4: Performance comparison among C1M1, C1M2, C2M1, and C2M2 for three configurations for the three variables (i) population size, (ii) candidate partitions, and (iii) fitness value.

significant main effects among stacked graph versions and view prefixes.

Speed tests indicate that V1 and V2 required significantly more time to read than V3 ($P < 0.05$). Similarly, subject error data indicated significant main effects among stacked graph versions. Accuracy tests indicated that more errors were committed with V1 and V2 than with V3 ($P < 0.05$).

A similar ANOVA of speed test indicated significant main effects among view prefixes when V2 is used ($P < 0.05$), that is, with this version some views will require lower time to read, but with some views, longer time. Here, we hypothesize that subjects, on some views, were able to find answers even if labels were overlapped. On the other hand, the same test indicated that there are no significant differences among view prefixes when V1 and V3 are used ($P < 0.05$). That is, if V1 is used, longer time will be required to find labels. If V3 is used, however, users will require significantly lower time to find them.

Interestingly, a similar ANOVA of subject accuracy showed an equivalent result as those in the previous para-

graph. Subject error data indicated significant main effects among view prefixes when V2 is used ($P < 0.05$), that is, with this version, the accuracy of users in their findings will be moderately high only on some views. On the other hand, the same test indicated that there are no significant differences among view prefixes when V1 and V3 are used ($P < 0.05$). That is, if V1 is used, the accuracy of users will always be low. If V3 is used, however, users will show significantly high accuracy. Figure 6 shows a comparison results of the conducted usability study.

6. Conclusions

In this paper, we intended to highlight the stacked graph as an interesting object of study. We have suggested an evolutionary computation for label layout on the unused space, where a method for maximizing the display of labels is proposed. An important purpose is to improve the legibility of the stacked graph by revealing to the users more information for further analysis. We developed a genetic

FIGURE 6: Comparison results of the conducted usability study with the three versions of the stacked graph visualization. Two different values for TLV (threshold for label visibility) are used in the three versions, as well as our proposed stacked graph with automatic label layout on unused space. The comparison among versions are shown for each view prefix.

algorithm which computes two fitness functions, one for solving the overlapping problem, and the other for keeping order correlations among labels and their corresponding visual objects. Our algorithm was inspired by previous solutions for ordering and permutation problems. Here, however, we adjusted two standard genetic operators in order to solve a similar problem in the stacked graph.

As for our usability study, our method proved to be successful. Our statistical analysis results, as well as our observations, showed that our stacked graph version outperformed the traditional ones. It is clear that solving the overlapping problem contributes to improve the legibility of the visualization, however, more investigation is needed to know how effective the correlation among labels and their visual objects are, as a semantic relationship, and useful for visual analysis tasks. For this, we believe that a different user experiment should be designed.

Acknowledgment

This work is supported by the MEXT Global COE Program Digital Humanities Center for Japanese Arts and Cultures, Ritsumeikan University.

References

- [1] J. Marks and S. Shieber, "The computational complexity of cartographic label placement," Tech. Rep. TR-05-91, Center for Research in Computing Technology, Harvard University, 1991.
- [2] J. Heer, F. B. Viégas, and M. Wattenberg, "Voyagers and voyeurs: supporting asynchronous collaborative visualization," *Communications of the ACM*, vol. 52, no. 1, pp. 87–97, 2009.
- [3] M. Wattenberg and J. Kriss, "Designing for social data analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 549–557, 2006.
- [4] L. Byron and M. Wattenberg, "Stacked graphs—Geometry & aesthetics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1245–1252, 2008.
- [5] A. Toledo, K. Sookhanaphibarn, R. Thawonmas, and F. Rinaldo, "Personalized recommendation in interactive visual analysis of stacked graphs," *Journal of ISRN Artificial Intelligence*. In press.
- [6] J. Li, C. Plaisant, and B. Shneiderman, "Data object and label placement for information abundant visualizations," in *Proceedings of the 7th ACM International Conference on Information and knowledge Management—Workshop on New Paradigms in Information Visualization (NPIV '98)*, pp. 41–48, Bethesda, Md, USA, November 1998.
- [7] K. Ali, K. Hartmann, and T. Strothotte, "Label layout for interactive 3d illustrations," *Journal of the WSCG*, vol. 13, no. 1, pp. 1–8, 2005.
- [8] I. Vollick, D. Vogel, M. Agrawala, and A. Hertzmann, "Specifying label layout style by example," in *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*, pp. 221–230, Newport, RI, USA, October 2007.
- [9] C. C. Lin, "Crossing-free many-to-one boundary labeling with hyperleaders," in *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '10)*, pp. 185–192, March 2010.
- [10] S. Havre, E. Hetzler, P. Whitney, and L. Nowell, "ThemeRiver: visualizing thematic changes in large document collections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 9–20, 2002.
- [11] K. Hartmann, T. Gtzelmann, K. Ali, and T. Strothotte, "Metrics for functional and aesthetic label layouts," in *Smart Graphics*, A. Butz, B. Fisher, A. Krger, and P. Olivier, Eds., vol. 3638 of *Lecture Notes in Computer Science*, p. 924, Springer, Berlin, Germany, 2005.
- [12] J. Talbot, S. Lin, and P. Hanrahan, "An extension of Wilkinson's algorithm for positioning tick labels on axes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1036–1043, 2010.
- [13] M. Luboschik, H. Schumann, and H. Cords, "Particle-based labeling: fast point-feature labeling without obscuring other visual features," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1237–1244, 2008.
- [14] J. A. A. Smither and C. C. Braun, "Readability of prescription drug labels by older and younger adults," *Journal of Clinical Psychology in Medical Settings*, vol. 1, no. 2, pp. 149–159, 1994.
- [15] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
- [16] K. De Jong, *Analysis of the behavior of a class of genetic adaptive systems*, Ph.D. dissertation, The University of Michigan, 1975.
- [17] A. Toledo, K. Sookhanaphibarn, R. Thawonmas, and F. Rinaldo, "Experimental evaluation of a stacked graph visualization," in *Proceedings of the Eurographics/IEEE Symposium on Visualization (EuroVis '10)*, Bordeaux, France, June 2010.