Eurographics Conference on Visualization (EuroVis) 2019
M. Gleicher, H. Leitte, and I. Viola
(Guest Editors)

Volume 38 (2019), Number 3

Optimizing Stepwise Animation in Dynamic Set Diagrams (Supplementary Material)

Kazuyo Mizuno[1], Hsiang-Yun Wu[2], Shigeo Takahashi[3], and Takeo Igarashi[4]

[1]Yahoo Japan Corporation, Japan
[2]TU Wien, Austria
[3]The University of Aizu, Japan
[4]The University of Tokyo, Japan

Appendix A: Formulation of Action Grouping

To find an optimal collection of subsets of actions, we solve a variant of the *set cover problem* that finds an optimal collection of subsets that covers all elements while minimizing a predefined cost function. In the proposed approach, we duplicate all actions except for translation type actions as two different groups; that is, actions $\{a^p_1, a^p_2, \ldots, a^p_k\}$ that are performed prior to the translation of the corresponding elements, and actions $\{a^q_1, a^q_2, \ldots, a^q_k\}$ that are performed after the translation. Recall that k is the number of actions. We then exhaustively enumerate a collection of all possible subsets of compatible actions in such a way that every pair of actions in each subset satisfies the conditions. Our set cover problem can be solved as an integer programming problem by minimizing:

$$\sum_{j=1}^{r} w_j u_j, \tag{S.1}$$

while satisfying:

$$\sum_{j=1}^{r} (b^p_{ij} + b^q_{ij}) u_j = 1 \quad \forall i, \tag{S.2}$$

where r indicates the total number of such possible subsets of compatible actions. Here, the binary value b^p_{ij} or b^q_{ij} is 1 only if a^p_i or a^q_i, respectively, is contained in the j-th subset. Note that the duplicated actions a^p_i and a^q_i are incompatible with each other, and thus only one of the two actions is activated in the entire animation. Furthermore, w_j is a weight value for the j-th subset and is computed as the weighted sum of the average distance between all pairs of elements in that subset and the average distance between all pairs of sets along the depth axis in that subset. In this formulation, u_j is 1 if the corresponding j-th subset is selected as a cluster of actions; thus, by referring to the solution, we can group actions into several clusters to reduce the total number of animation steps.

Note that in our experimental results, the maximum number of actions that coexist in a single animation step is five, following the evaluation of dynamic graph animation conducted by Archambault et al. [AP13].

Appendix B: Optimization of the Action-Cluster Order

Weights of Cost Function

Note that α, β, γ, and δ are the weight values of the cost functions for ordering action clusters (in Section 5.3), respectively, and, unless stated otherwise, are set as $\alpha = \beta = \gamma = 1.0$ and $\delta = 10000$. For adjusting the weight values, our guideline suggests that we should increase α if the dynamic set diagram has a large amount of translation for its elements, β if the diagram has a large number of exclusions and inclusions, and γ if it intrinsically contains frequent change in the depth order of sets. The last weight δ is set to be relatively large since the corresponding function F_4 penalizes the unnecessary splitting of sets in the optimization.

Computational Cost

Our optimization seeks the best permutation of action clusters that optimizes the total cost function and thus can be formulated as a variant of the traveling salesman problem. With respect to algorithm complexity, the worst computational complexity of the cost functions at one permutation is $\mathcal{O}(k^2 + nmk')$, where k and k' are the number of primitive actions and action clusters, respectively, and n and m are the number of elements and sets, respectively. This means that the computational complexity of all possible permutations in the proposed method is $\mathcal{O}(k'!(k^2 + nmk'))$. We employed a simple *genetic algorithm* (GA) [Gol89] to find the best possible solution within a reasonable period of time. This was accomplished by encoding the sequence of action cluster IDs as a value-encoding chromosome and performing genetic-based optimization by evolving the collection of elite chromosomes.

Appendix C: Implementation Details of Rendering Set Diagrams

Contour Generation

To draw a smooth boundary for each set, we extract the *contours* of the 2D scalar field as *implicit curves* [WWI07] to properly associate spatial regions with the constructed skeleton graph.

Given a set of data samples $m_1, m_2, \ldots, m_n$, a blending function at a point m is commonly defined as follows:

$$f_h(m) = \frac{1}{n} \sum_{i=1}^{n} K_h(m - m_i), \tag{S.3}$$

where K is a kernel function and h is its bandwidth parameter. To expand the skeleton graph to a contour region, we first calculate the entire scalar field as the sum of individual kernel functions associated with its member elements. Then, we assign a line kernel function $L(e)$ to each edge e in the skeleton graph and incorporate the expanding effect into the scalar field to reinforce the scalar field within the set region. Note that we employ the previously proposed line kernel function [DH11] as follows:

$$L_h(e) = \int_0^1 K_h(m - ((1-\phi)m_s + \phi m_e))d\phi, \tag{S.4}$$

where m_s and m_e are the start and end positions of an edge. Specifically, we use a Gaussian window function as the kernel function. After accumulating all kernel functions, we can compute the entire scalar field by following the work of Daae Lampe and Hauser [DH11]. Finally, we extract a contour line for each set by sectioning the scalar field at a specific value.

In the proposed approach, we place a set that is smaller in terms of its size over larger sets to avoid unwanted occlusions. For this purpose, we increase the bandwidth parameter by a constant value as the depth of the corresponding set becomes more.

Color Assignment of Sets

In the proposed system, we assign a unique color to each set and apply gradation according to the corresponding scalar field value of the set. This is implemented by first selecting the hue value of the HSV color space according to the ID of the corresponding set and then changing the saturation value according to the scalar field value within that set. Thus, the HSV color (i.e., $(\text{hue}, \text{saturation}, \text{value})$) at pixel p within the set s_i can be calculated as follows:

$$C(s_i, p) = (\frac{2\pi i}{l}, f(p), 1.0), \tag{S.5}$$

where l is the total number of sets, i is the ID of the current set, and $f(p)$ represents the normalized scalar field value at pixel p $(0 \leq f(p) \leq 1)$. To fully discriminate between different colors of adjacent sets, we set the color of the contours to gray.

Interpolation for Animated Transition

To synthesize the stepwise animation of the dynamic set diagrams, we must define how to smoothly interpolate individual nodes and edges in the skeleton graph according to their appearance/disappearance. Rather than morphing between contour lines of sets directly, we gradually change the bandwidth values of the kernel functions associated with the nodes and length of the edges according to the action types. In practice, we change three visual characteristics of the set diagrams before and after action a_i; that is, the contour lines associated with target element $m_{M(i)}$, edges incident to $m_{M(i)}$, and the depth ordering of set $s_{S(i)}$. Recall that $M(i)$ and $S(i)$ are the IDs of the elements and sets associated with the i-th action a_i, respectively. Note that in all cases, we introduce ease

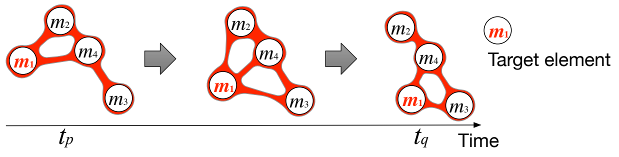

Figure S.1: *Example of a union graph.*

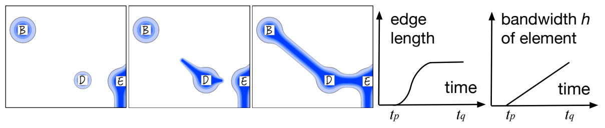

Figure S.2: *Animation steps associated with edge insertion.*

in/out effects [Las87] in the animation to augment the perceptual smoothness of the dynamic set diagrams.

When interpolating the contour lines around the target element, we linearly change the bandwidth value of the associated kernel functions (Eq. (S.3) and Eq. (S.4)) during the transition of a_i. Note that we set the bandwidth value h to 0 if the target element does not exist before or after the action.

When inserting/removing edges incident to the target element, we employ a union graph of the two skeleton graphs at t_p and t_q using the method developed by Diehl et al. [DGK01, DG02] to maximally preserve the mental map. In the middle of the animation step of the target action, we render the edges incident to the target element within the union graph, as shown in Figure S.1. We also smoothly control the length of the edges in proportion to the change in bandwidth, as shown in Figure S.2.

As described previously, in the proposed approach, the depths of all the sets are sorted according to size. In the actual implementation, each set implicitly has a real variable as its depth value, which allows us change the depth sequence of the sets by linearly interpolating the depth values, as shown in Figure S.3. When two sets have sufficiently close depth values, we also smoothly change the set diagram color by applying alpha blending in the overlapped region (Figure S.3).

Appendix D: Questionnaires for User Study

In the user study, the questions of each dataset were set as follows.

Flu

Q1) Find the countries that were infected with flu of Type A(H3) and Type B.
Q2) Find the types of viruses that infected the countries of Poland, the Czech Republic, and Austria (at least once).
Q3) Find countries that were always covered by the same types of virus.
Q4) Find the country that has the strongest impact on distributing virus to the other countries.

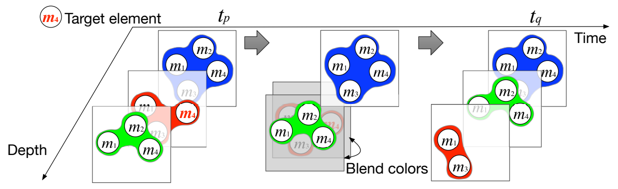

Figure S.3: *Smoothly changing the depth order and the color of sets.*

Authorship

Q1) Find the research topic that was investigated by the most researchers.
Q2) Find the researcher whose research topics were completely contained by other researchers.
Q3) Find a researcher who always concentrated on a specific set of research topics.
Q4) Find the researcher who increased her/his research topics the most.

References

[AP13] Archambault D., Purchase H. C.: The "Map" in the Mental Map: Experimental Results in Dynamic Graph Drawing. *International Journal of Human-Computer Studies 71*, 11 (2013), 1044–1055. doi:10.1016/j.ijhcs.2013.08.004. 1

[DG02] Diehl S., Görg C.: Graphs, They are Changing Dyamic Graph Drawing for a Sequence of Graphs. In *Proceedings of the 9th International Symposium on Graph Drawing (GD '02)* (2002), vol. 2528, Springer Lecture Notes in Computer Science, pp. 23–31. doi:10.1007/3-540-36151-0_3. 2

[DGK01] Diehl S., Görg C., Kerren A.: Preserving the Mental Map Using Foresighted Layout. In *Proceedings of the 3rd Joint Eurographics/IEEE VGTC Conference on Visualization (EuroVis '01)* (2001), vol. 1, pp. 175–184. doi:10.1007/978-3-7091-6215-6_19. 2

[DH11] Daae Lampe O., Hauser H.: Interactive Visualization of Streaming Data with Kernel Density Estimation. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis '11)* (2011), pp. 171–178. doi:10.1109/PACIFICVIS.2011.5742387. 2

[Gol89] Goldberg D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Professional, 1989. 1

[Las87] Lasseter J.: Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (1987), SIGGRAPH '87, pp. 35–44. doi:10.1145/37401.37407. 2

[WWI07] Watanabe N., Washida M., Igarashi T.: Bubble clusters: An Interface for Manipulating Spatial Aggregation of Graphical Objects. In *Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07)* (2007), pp. 173–182. doi:10.1145/1294211.1294241. 1