Visualizing Gene Expression Data via Voronoi Treemaps

Conference Paper · June 2009

DOI: 10.1109/ISVD.2009.33 · Source: DBLP

Jörg Bernhardt
University of Greifswald
168 PUBLICATIONS **2,820** CITATIONS
SEE PROFILE

Juliane Siebourg
ETH Zurich
17 PUBLICATIONS **116** CITATIONS
SEE PROFILE

Visualizing Gene Expression Data via Voronoi Treemaps

Jörg Bernhardt*‡, Stefan Funke†, Michael Hecker* and Juliane Siebourg†
*Institute for Microbiology, Greifswald University, Friedrich-Ludwig-Jahn-Strasse 15a, 17487 Greifswald, Germany
†Institute for Mathematics and C.S., Greifswald University, Robert-Blum-Strasse 2, 17487 Greifswald, Germany
‡DECODON GmbH, Biotechnikum, Walther-Rathenau-Strasse 49a, 17489 Greifswald, Germany

***Abstract*—We investigate the use of Voronoi Treemaps to visualize complex biological data from gene expression analysis in conjunction with hierarchical classifications like *Gene Ontology* or *KEGG Orthology*.**

I. Introduction

With modern molecular biological techniques which use up to several million probes as well as high throughput protein detection, it has become quite easy to gather massive amounts of genomic and proteomic data, which need to be evaluated. In the field of gene expression and proteome analysis biologists try to discover how *gene expression* works. *Gene expression* is the process by which inheritable information of a gene, such as the DNA sequence, is turned into a functional gene product, such as a RNA or a protein. Questions like under which physiological conditions which genes have to be active and which proteins are needed or which genes lead to which proteins are tried to be solved. Furthermore the function of proteins is investigated and complex regulatory networks between genes and proteins respectively are examined.

To monitor gene activity biologists use techniques like DNA microarrays or proteome studies (functional genomics studies). These make it possible to examine gene expression in particular under defined stimuli (e.g. treated and untreated; healthy and unhealthy; optimal, stressed and starved). The analysis usually results in lists of genes/proteins with similar expression behavior. Often a comparison of similarly expressed proteins with known functional classifications leads to a first hypothesis revealing what occurs in the biological system under analysis.

A typical representation of gene expression data is a so-called *heat-map*, see Figure 2 for a partial representation of such a heat-map. Here, different genes are represented as *columns*, different stimuli/experiments as *rows* of the heat-map. Each gene is assigned to a color ranging from black (maximum gene expression) via gray (average gene expression) to white (minimum gene expression). Commonly color shading is applied to standardized expression data (mean centering and subsequent standard deviation scaling within a column of expression data).

A clear indication of correlations of similarly expressed genes supports a global understanding of the gene expression program at the condition under analysis. However, a clear visualization of the functional relatedness of genes by using heat-maps is difficult. Because of the space consuming representation of heat-maps the reader will probably find it difficult to draw any conclusion from them or interpret the results.

When looking for groups of genes that exhibit a strong correlation under different conditions, it might be helpful to incorporate additional known information about the gene function. For this purpose different classification schemes for genes exist that allow for a pre-grouping of the genes such that genes that are more likely to exhibit expression correlation are visualized 'close to each other'. Among COG, KEGG Orthology, FunCat and others the *gene ontology project* (GO) [4] for example may meet this need. It provides three (quasi-hierarchical) ontologies, one based on the molecular function of a gene, one based on the biological process in which a gene participates, and one based on the cellular compartment where a gene product is active. All available ontologies are polyhierarchies given by directed acyclic graphs, which almost form trees. Many of these ontologies have been complemented with tools, which should help with the analysis of gene expression according to these classifications, but a lot of them fail. Often this is because of poor usability, slow speed or inconvenient visualization of the results. The following is a natural way to make use of this knowledge to support the interpretation of gene expression data:

1) determine an ontology based layout/drawing of all genes/proteins under analysis
2) for each experimental condition, create a copy of the drawing and visualize the gene expressions (e.g. using colors)

By visual inspection of these drawings, groups of functionally related genes whose expression patterns are highly correlated across the experimental conditions can be identified. As an example, see Figure 1 where the genes of *Bacillus subtilis* are drawn as Voronoi Treemaps (explained later) according to KEGG orthology [6]. Each of them shows gene expression data at a different phase of a glucose starvation experiment with subsequent growth recovery caused by glucose resupply within a *Bacillus subtilis* liquid culture [7]. The expression values are color coded ranging from black (maximum gene expression) via gray (average

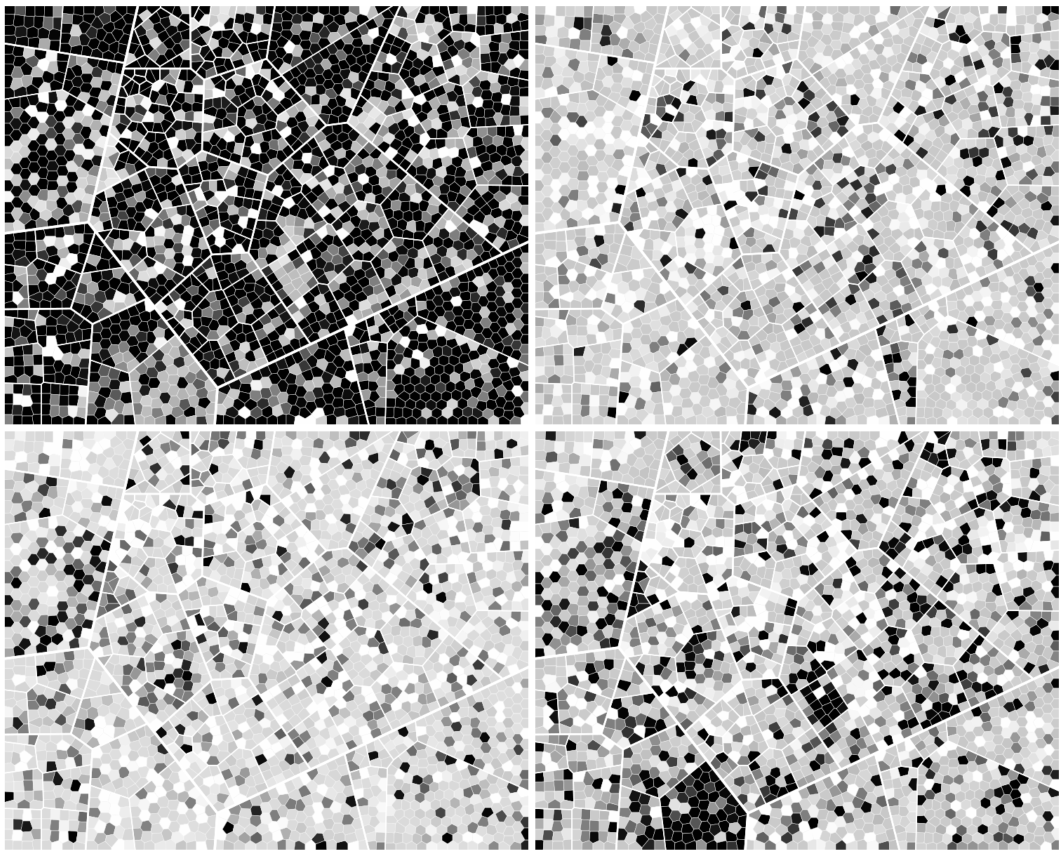

Figure 1. Voronoi Treemaps Visualization of four phases of a glucose starvation experiment with Bacillus subtilis based on the KEGG orthology: upper left: control, upper right: transient phase, lower left: late stationary phase and lower right: growth recovery.

gene expression) to white (minimum gene expression). On the upper left we see the expression levels in the control phase, on the lower right in the growth recovery phase. In both Treemaps there is a group of genes near the left bottom corner (*ribosomal proteins*) which apparently exhibit a strong correlation under these environmental conditions. Identifying the same correlation from a heat-map like Figure 2 is a lot more difficult.

On an abstract level, one could view the chosen ontology as a metric on the genes according to which an embedding of the genes in $\mathbb{R}^2$ should be computed. A typical goal is to embed with *low distortion*, that is, if two genes are 'close' in the ontology-induced metric, they should also be drawn close to each other in the diagram, while if they are far away from each other in the metric they should be geometrically distant in the diagram. Many methods are known to compute such embeddings, some of them are based purely on the metric, others take the metric-inducing graph (in our case 'almost-trees') into account (see [2] for more references).

One particular type of embedding, which we focus on in this paper are so-called *Treemaps* [5]. Originally meant to visualize hard-drive usage, they can be used to visualize any tree-like structure. The idea is quite straightforward. We start with an arbitrary shape (e.g. a square) which we identify with the root of the tree. In a next step we partition the shape into as many cells as the number of children of the root. The size of the cells is typically chosen proportional to the number of leaves of each of the respective subtrees, but can be chosen arbitrarily depending on the application semantics. Recursively, these cells are then again subdivided based on the number of the respective subtree leaves.

One of the first incarnations of a Treemap was given by the so-called "Slice&Dice" algorithm in [5]. Figure 3 gives an example of a hierarchy and its corresponding Treemap according to the "Slice&Dice" algorithm. The subdivision into cells at the root level is created solely by *vertical* cuts, in the next level, *horizontal* cuts are used to subdivide the cells. This approach suffers from one major disadvantage: if some of the subcells have a very small desired size, the "Slice&Dice" algorithm yields extremely skinny cells with very large *aspect ratio* (see the formal definition below). Figure 4 (left) gives an example.

Subsequently, other Treemap algorithms have been developed which aimed at maintaining good aspect ratios. In

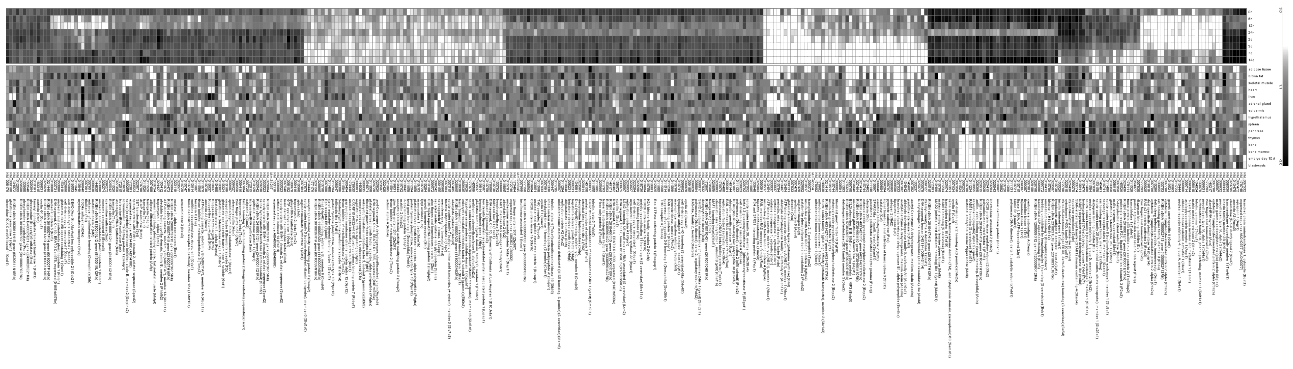
Figure 2. Heat-Map (MeV-Team), 2008.

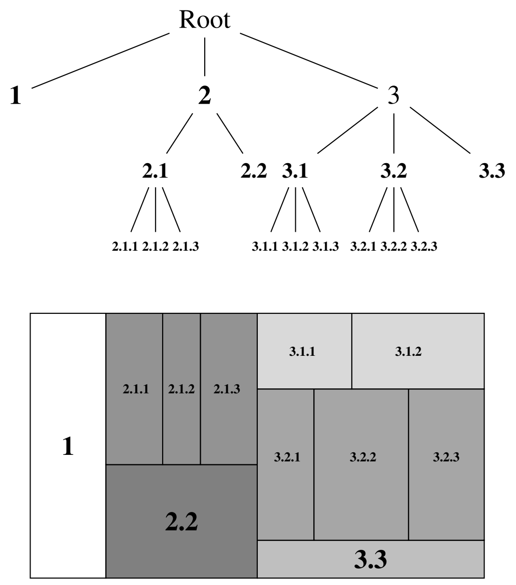

Figure 3. A given hierarchy and its respective Treemap according to a "Slice&Dice" algorithm.

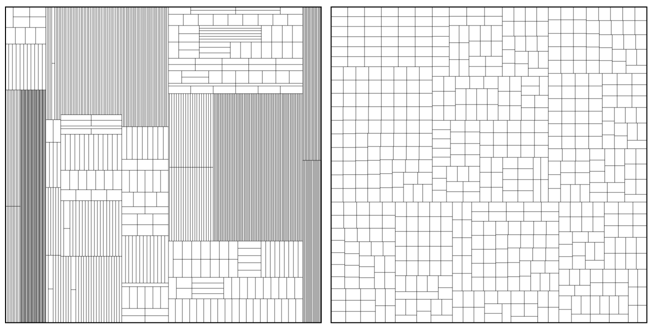
Figure 4. Problems with "Slice&Dice" (left) and "Squarified Treemaps" (from [1]).

particular squarified Treemaps [3] which employ a more sophisticated subdivision technique allowing both horizontal and vertical subdivision lines when creating subcells, have been widely used in practice. Main drawback of this solution is the fact that the hierarchical structure is not as well represented as for example with the "Slice & Dice" method, see Figure 4 (right).

Balzer and Deussen in [1] proposed *Voronoi Treemaps* which in their experiments exhibit both good aspect ratios as well as a clear visualization of the hierarchical structure. Cells are associated with nodes of the tree. In the recursion step, the goal is to place as many generators in the cell as the number of children of the node such that the cells of the Voronoi diagram (or a variant thereof) of these generators have the desired size. Balzer/Deussen achieve this through a reweighting and repositioning strategy which is revisited in Section II.

Problematic with this approach is the fact that the computation of the cell subdivision is a rather time consuming process which does not allow for real-time visualization of dynamically changing datasets. Still, for our application domain – the visualization of gene expression data – this method seems most promising.

Our Contribution

The main contribution of this paper is the application of Voronoi Treemaps on gene expression datasets. Therefore we are using (a variant of) the Balzer/Deussen algorithm. We exhibit certain problems of the algorithm which were only briefly mentioned in the Balzer/Deussen paper and also devise certain optimization strategies some of which proved to be useful for our application.

We developed a software that uses hierarchically organized gene functional classification data to calculate Voronoi Treemaps. This leads to the formation of convex clusters of functionally related proteins. Proteins are represented by cells of constant size. Expression data is mapped into these cells by using color shades. If similarly colored proteins share the same cellular function, the corresponding cluster will be dominated by this color. This clearly highlights the whole functional cluster as being involved in the physiological phenomenon under analysis.

Therefore the visualization of data from global gene expression analysis using Voronoi Treemaps is a very effective tool for molecular biologists due to the discovery of relationships of co-expressed genes and their functions.

II. Voronoi Treemap Visualization

The basic idea of Voronoi Treemaps is very straightforward. For the remainder of the paper we aim at partitioning a cell corresponding to a node always proportionally to the number of leaves of the respective subtrees (since this is desired for our particular application domain). Furthermore aiming at polygonal subdivisions into convex cells, we restrict our attention to the *additively weighted power Voronoi diagram* which is induced by the following distance function (measuring the distance of a point p to a generator g_i with weight w_i):

$$dist_{g_i,w_i}(p) = ||g_i - p||^2 - w_i$$

Clearly, with $w_i = 0$, this distance function induces the 'regular' Voronoi diagram. With weights $w_i > 0$ we still retain the property that bisectors between generators are straightline segments.

So given a tree and a visualization area P (e.g. a polygon or simple square), the following recursive procedure invoked with the root r of the tree and P computes a Voronoi Treemap of the tree:

```
PartitionAndRecurse(r, P)
  1. let c_1, c_2, ... c_k   be the children
     of r, s_1, s_2, ... s_k the number of
     leaves in the respective subtrees
  2. place generators g_1, g_2, ... g_k
     inside P and potentially choose weights
     for the generators such that in the
     resulting (weighted) Voronoi diagram
     within P the respective cells
     C_1, C_2, ... C_k have
      sizes according to s_1, s_2, ... s_k
  3. for each c_i that is not a leaf,
      invoke PartitionAndRecurse(c_i, C_i)
```

The second step – placement of the generators and choosing weights for the weighted Voronoi diagram – is the most crucial one in this procedure to create visually pleasing Treemaps.

Note that if there are no additional constraints on the shape of the Voronoi cells, generators can be easily placed in a convex cell P such that the respective cells have the desired size: construct a line passing through both one point with minimum as well as one maximum x-coordinate within P. Then $g_1, \dots g_k$ can trivially be distributed on the intersection of this line with P such that the respective (unweighted) Voronoi cells have exactly the desired weight. The disadvantage of this solution is that it might lead to exactly those skinny cells with bad aspect ratio as the "Slice&Dice" method.

Definition 1: The *aspect ratio* α_P of a polygon P is defined as $\alpha_P = \frac{r_{encl}}{r_{inscr}}$ where r_{encl} is the radius of the smallest enclosing circle of P and r_{inscr} is the radius of the largest inscribed circle in P.
We call a polygon (or cell) P *α^*-fat* if for some given α^*, $\alpha_P \leq \alpha^*$.

We note that there are cases where all Voronoi partitions with cells of the desired sizes have one cell with arbitrarily bad aspect ratio. In fact all valid convex partitions can be forced to contain a cell with arbitrarily bad aspect ratio:

Lemma 1: For any $\alpha > 0, \alpha^* > 0$ there exists a convex α^*-fat polygon P and weights s_1, s_2 such that any partition of P into two convex pieces contains a cell with aspect ratio $> \alpha$.

Proof: We only sketch the (very simple) proof here. Consider as P a regular n-gon. The internal angle between two adjacent edges can be made $> \pi - \gamma$ for any $\gamma > 0$ by choosing n large enough. Equally, for sufficiently large n, P becomes α^*-fat since the radii of the minimum enclosing and maximum inscribed circle get arbitrarily close to each other. We fix $s_1 = \epsilon, s_2 = 1 - \epsilon$ with ϵ very small. First, observe that the 'partition' of P has to be a simple, straight-line cut; otherwise, one of the cells would not be convex. By choosing ϵ very small, this cut can be forced arbitrarily close to the boundary of the n-gon, creating a cell which can become arbitrarily skinny (as given by α). Also note that P is α^*-fat, so the skinniness of cell C_1 is not induced by the skinniness of P. ■

In practice very often it is possible to find positions and weights for the generators such that the resulting cells have a 'nice', that is 'fat', shape.

A. A variant of the Balzer/Deussen algorithm

Balzer and Deussen in [1] propose Voronoi Treemaps as a visually pleasing variant of the Treemap idea and also provide an algorithm to compute such a Treemap. We use a variant of their algorithm in the following.

Let $a(.)$ denote the area of a cell, $\frac{s_i}{\sum s_j}$ the desired fraction of P's area that should be assigned to cell C_i. For a given partition of P into cells $C_1, \dots C_k$, we call $\rho_i = \frac{a(P)s_i}{a(C_i)\sum s_j}$ the area ratio of cell C_i. Clearly we are aiming for area ratios $\rho_i = 1$; if $\rho_i > 1$ the cell C_i is too small, if $\rho_i < 1$, C_i is actually too big. To compute a Voronoi partition of a cell P into subcells $C_1, \dots C_k$ such that $\rho_i \approx 1$, we proceed as follows:

1) start with an initial placement of generators $g_1, \dots g_k \in P$, set all weights $w_i = 0$
2) compute the weighted Voronoi diagram of $g_1, \dots, g_k$ with weights $w_1, \dots, w_k$ restricted to P
3) if there exists a cell C_i in the resulting diagram with $|\rho_i - 1| > \epsilon$

a) update the weights of those cells C_i with $|\rho_i - 1| > \epsilon$
b) relocate the generators towards the center of mass of their respective cell

otherwise the partition is completed

Note that we use for the stopping criterion a *relative* bound ($\frac{a(P)s_i}{a(C_i)\sum s_j} = |\rho_i - 1| \leq \epsilon$), whereas Balzer/Deussen employ an absolute one ($|\frac{a(C_i)}{a(P)} - \frac{s_i}{\sum s_j}a(P)| = |\frac{a(C_i)}{a(P)} - \rho_i a(C_i)| \leq \epsilon$). If the weights are relatively uniform, there is no big difference between the two measures; in our application scenario we frequently encounter subtree sizes which are extremely different, though; that is, the desired cell sizes might differ by order of magnitudes (e.g. in a partition of P into 20 subcells, some of the subcells should occupy only about $1/1000$th of P's area). In this case, an absolute error bound would have to be extremely small to guarantee that each leaf of the tree occupies about the same area in the Treemap – a property that was considered quite important in our application domain (each gene should occupy about the same area).

1) Initial Placement of Generators: We experimented with different strategies for an initial placement of the generators, since at the beginning we did not implement the relocation strategy (we wanted to achieve ρ_i values close to 1 solely by adjusting their weights). The goal was to place the generators such that the normal Voronoi diagram would already give cells 'close' to the desired sizes. To this end we employed simple packing heuristics where we packed circles $K_1, \dots K_k$ disjointly into P and used their centers as positions for the generators $g_1, \dots, g_k$. The packing was performed in a greedy fashion, where circles were placed one after the other in some prescribed geometric pattern inside P, see Figure 5.

At the beginning we choose circle K_i to have exactly the desired area of cell C_i, that is, $a(C_i) = a(P)\frac{s_i}{\sum s_j}$. It is then impossible to pack those circles disjointly into P. We then scale down all circles by a large factor σ, e.g., $\sigma = 10$. Again we try to pack the scaled-down circles into P (which always succeeded due to the high scaling factor). Then using binary search, we determine the smallest scaling factor σ such that our packing heuristic could still pack the circles disjointly into P.

Of course, we would like to be able to prove a strong approximation guarantee stating for example that in the resulting first Voronoi partition after the circle packing step, for any cell C_i we have $\rho_{lower<}\rho_i < \rho_{upper}$. Unfortunately, it seems quite hopeless to achieve such general upper and lower bounds, in particular if the cell P can be of arbitrary shape. Still, we can show the following simple lower bound:

Lemma 2: If the circle packing algorithm succeeds in packing the circles after scaling with some factor $\sigma > 1$, we have

$$\forall i : \rho_i > 1/(4\sigma^2)$$

Proof: Assume the unscaled circle around g_i has area a_i, then the circle scaled down by 2σ has area $a_i/(4\sigma^2)$. This circle is certainly part of g_i's (unweighted) Voronoi cell, since no g_j, $j \neq i$ can be closer than g_i. The bound follows. ■

That means, no cell is much too small. For example for $\sigma = 1.5$, each Voronoi cell has at least one nineth of the desired area. Upper bounding ρ_i is difficult since a very small circle adjacent to a very large circle typically gets a too large Voronoi cell. Without any reweighting, the maximum deviation for the cell areas (the value $\max_i |\rho_i - 1|$) for the examples in Figure 5 were 0.85, 0.55, 3.08 (from left to right), so the initial placement in the center turned out to be the best choice.

We later combined this initial placement with a variant of the reweighting and relocation scheme by Balzer/Deussen.

2) Reweighting and Relocation: In our first attempt we hoped to achieve the desired cell sizes solely by reweighting the generators. We used the following rule to update the weight of generator g_i (in case $|\rho_i - 1| > \epsilon$):

$$w_i' = w_i(1 + \alpha_i \frac{a_i - v_i}{a_i})$$

where a_i is the desired, v_i the current area of cell C_i. That is, if $a_i > v_i$ the weight for generator g_i is increased. We also introduced an attenuation factor α_i which we decreased over time to improve convergence behaviour; at the beginning $\alpha_i = 1$.

For test instances of not too small size, individually decreasing the α_i (depending on whether the actual area v_i oscillated around the desired value a_i) turned out to be the best choice compared to uniformly decreasing all α_i over time, which in turn was superior to a constant attenuation factor, as can be seen in Figure 6. We stopped our algorithm after 15000 iterations even if no satisfying solution has been found. This happened frequently when no decreasing attenuation factor was used.

This picture changed completely, when we incorporated the relocation strategy by Balzer/Deussen, where each generator is relocated to the center of mass of its cell after every iteration. In this case, playing with the attenuation factor only worsened the number of required iterations. Relocating towards the center of mass also has the nice side-effect of improving the fatness of the resulting Voronoi cells. See Figure 7.

III. Evaluation for Datasets from DNA microarray experiments

We have used a Java implementation of our algorithm to visualize the results of a series of DNA microarray experiments. We leave the biological discoveries to a companion paper (targeted more at a bio-audience) and restrict to the Treemaps produced by our implementation. Note, that the ontologies considered in the following are directed acyclic

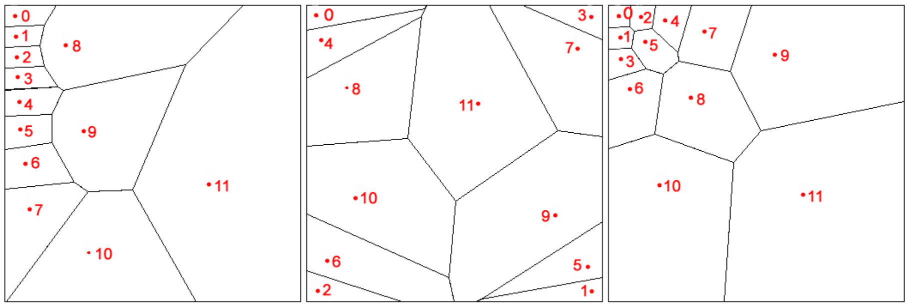

Figure 5. Different patterns when packing circles induce different initial Voronoi diagrams.

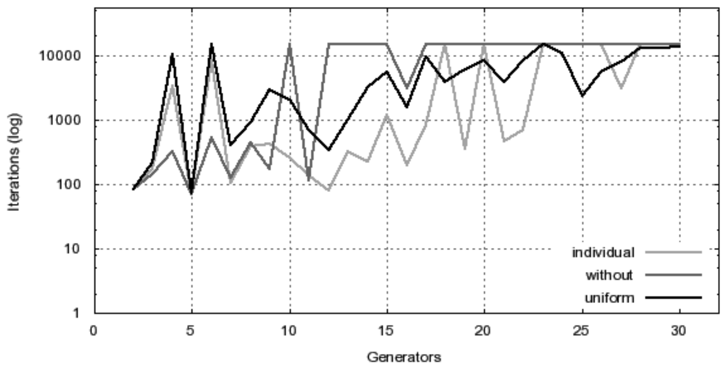

Figure 6. Number of iterations when varying the attenuation factors α_i; *without* relocation to the center of mass.

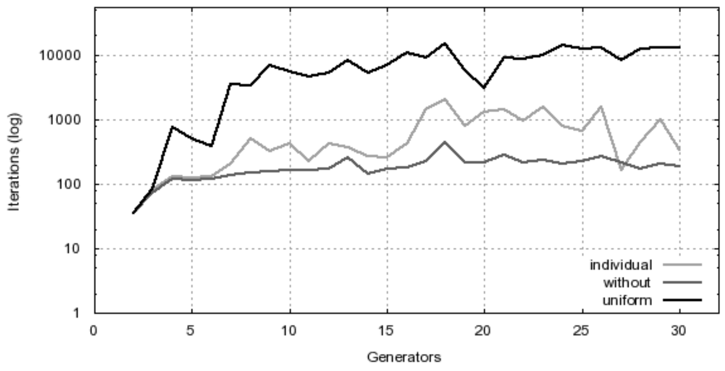

Figure 7. Number of iterations when varying the attenuation factors α_i; *with* relocation to the center of mass.

graphs which are only *almost* trees. Before applying the Voronoi Treemap algorithm we turn these DAGs into trees by subtree duplication, that is, if a node has x incoming edges, x copies of the node (together with its subtree) are created. This implies that some genes might be present several times in the drawing which is not a real problem in terms of interpretability, though.

A. Gene Ontology

As mentioned in the introduction, the gene ontology project provides essentially three hierarchies to group genes. We have run three versions of our algorithm with these hierarchies:

- **NoOptimization:** we only used the positions of the packing procedure and computed a normal Voronoi diagram (no reweighting); this variant, of course, incurs the largest error in terms of cell sizes
- **WithOptimization:** the weights were adjusted to control the cell sizes (with a maximum relative error of 5%)

- **WithCVT:** as before but including the relocation strategy towards the centers of mass (with a maximum relative error of 2%)

The results can be found in Figures 8, 9, 10. Here we used the *GO slim generic*, a cut down version of GO, only containing the top four levels. It can be observed that for the Treemaps on the left (NoOptimization), the sizes of the cells on the leaf level vary considerably (they are supposed to be all of the same size). In case of the 'cell localization' ontology, there is actually a maximum relative error of 200%. Using the reweighting strategy, the cell sizes on the leaf level are considerably more uniform, the cells often are rather skinny, though – an artefact of the initial placement strategy and the lack of any possibility to relocate the generators. As to be expected, the best results are produced by including the relocation strategy; the cells at leaf level are almost uniform in size and all cells exhibit nice aspect ratios.

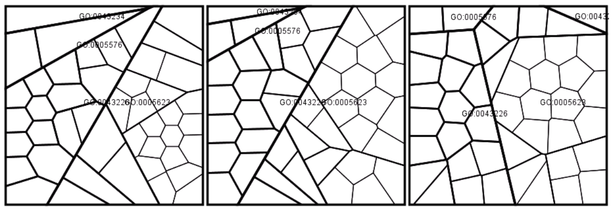

Figure 8. Treemaps of the GO slim generic (ontology: cell localization). Without optimization (left), with optimization (center), with CVT phase (right).

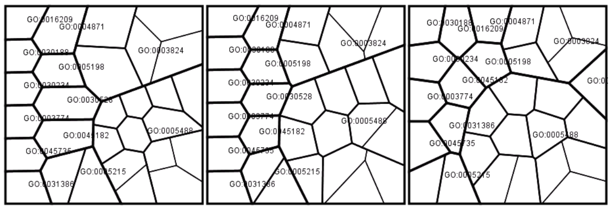

Figure 9. Treemaps of the GO slim generic (ontology: molecular function). Without optimization (left), with optimization (center), with CVT phase (right).

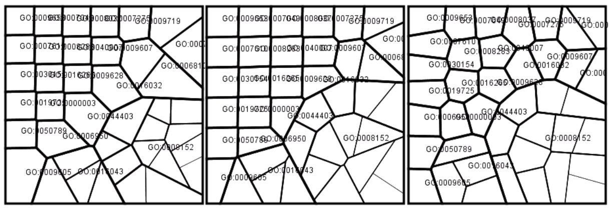

Figure 10. Treemaps of the GO slim generic (ontology: biological process). Without optimization (left), with optimization (center), with CVT phase (right).

Unfortunately at present there is no way to extract organism specific subhierarchies from GO. This made it impossible to create useful Treemaps for mapping expression data.

B. KEGG Orthology

In Figure 11 we present the Treemap used for Figure 1 in detail. It was created using all phases of our algorithm. The maximum relative deviation was set to 10%, the running time was around 10 minutes on a single core of a 1.6 GHz dual core processor.

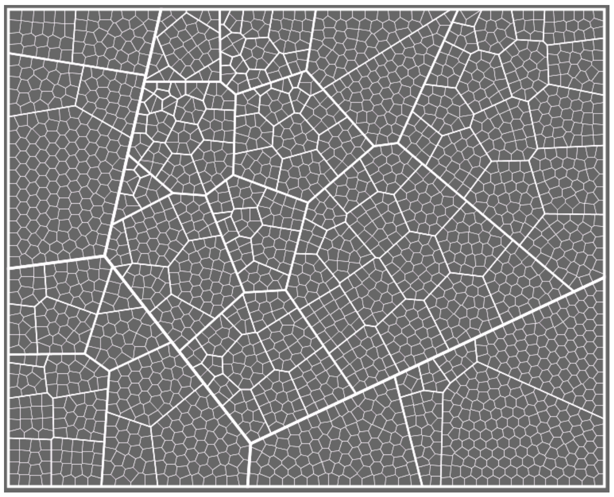

Figure 11. Treemap of KEGG orthology for *B.subtilis*.

KEGG is not as extensive as GO but provides organism specific hierarchies. It has four levels and for *Bacillus subtilis* the hierarchy consists of 2552 nodes containing about 2200 of its 4100 genes. Figure 12 shows Treemaps with different dissolutions of the KEGG hierarchy. As can be seen in the upper left Treemap, on first level the hierarchy is partitioned into the four main categories *Metabolism* (middle-gray), *Genetic Information Processing* (light-gray), *Environmental Information Processing* (dark-gray) and *Cellular Processes* (very lite-gray). On second level further partitions appear in the Treemap at the bottom left. Categories have numbers according to the following table:

- **Metabolism**
  1) Carbohydrate Metabolism
  2) Energy Metabolism
  3) Lipid Metabolism
  4) Nucleotide Metabolim
  5) Amino Acid metabolism
  6) Metabolism of Other Amino Acids
  7) Glycan Biosynthesis and Metabolism
  8) Biosynthesis of polyketides and nonribosomal Peptides

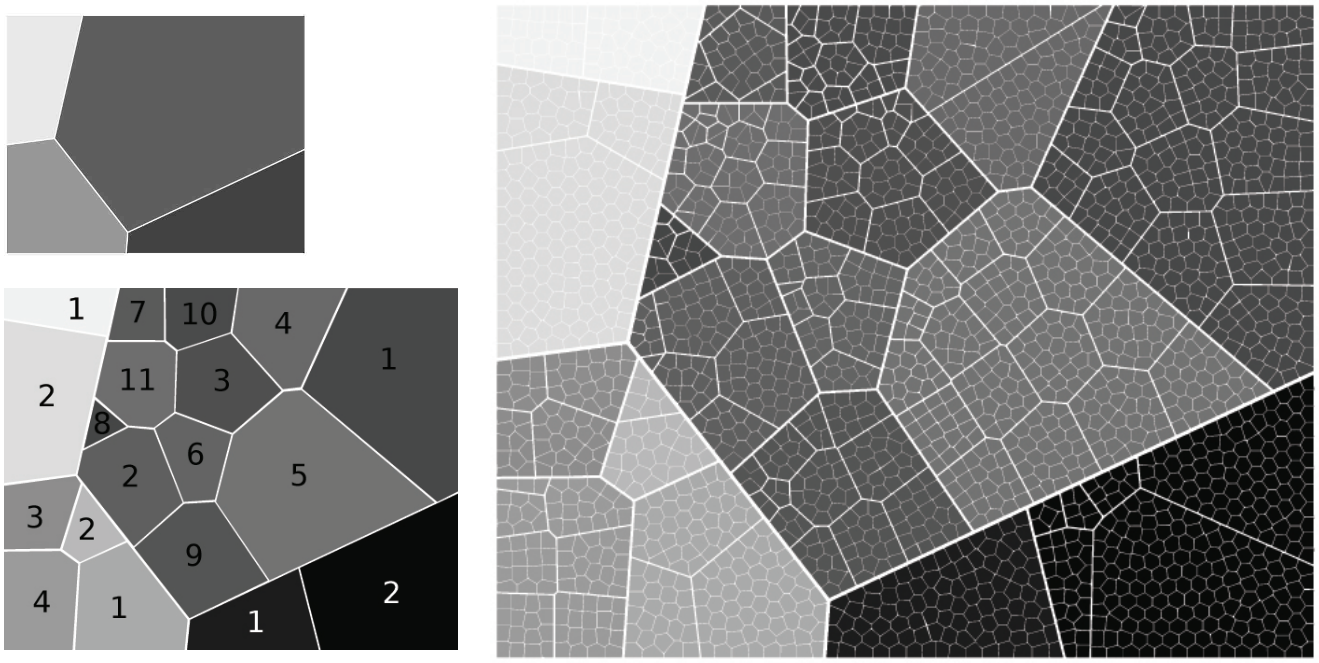

Figure 12. Treemaps of different levels of KEGG orthology for *B.subtilis*. Top left: first level only, bottom left: first and second level, right: all four levels.

9) Metabolism of Cofactors and Vitamins
10) Biosynthesis of Secondary Metabolites
11) Xenobiotics Biodegradation and Metabolism

- **Genetic Information Processing**
  1) Translation
  2) Transcription
  3) Folding, Sorting and Degradation
  4) Replication and Repair
- **Environmental Information Processing**
  1) Signal Transduction
  2) Membrane Transport
- **Cellular Processes**
  1) Cell Motility
  2) Cell Growth and Death

The Treemap on the right of figure 12 shows all levels of KEGG Ontology. The fourth level consists only of genes/ proteins where expression data can be mapped on.

C. General Remarks

Balzer/Deussen mention in [1], page 5, 'in most cases' their iterative algorithm converges to a good solution; sometimes a reinitialization with new random positions was necessary. In our experiments, problem instances with quite uniform desired cell sizes were handled very well. Extremely small desired cell sizes turned out to be difficult. Our stopping criterion based on the *relative* error often required an extremely high number of iterations, sometimes without finding a solution within the desired quality guarantee. Part of the problem might be due to round-off errors incurred by the floating-point evaluation of the geometric predicates in the Voronoi diagram algorithm. For extremely small cells rounding errors might falsify the computation results considerably. We are considering the use of exact arithmetic packages like LEDA [8] or GMP to exclude this source of error.

IV. Conclusions and Outlook

Overall, the use of Voronoi Treemaps has proven to be a very useful tool for biologists to get a better understanding of the correlation between the expression of genes and their functional categorization. Creating Treemaps of other ontologies could be interesting to see how they differ from each other. Our implementation based on the idea of Voronoi Treemaps due to Balzer/Deussen proved to be absolutely sufficient for the current demands. In the long run, it might be interesting to see whether the methodology can be improved to a degree such that dynamically changing data can be visualized in real-time. The current, iterative procedure for most data sets incurs a computation time of several minutes (which we want to emphasize is not a problem at the moment since the data is not dynamically changing). We hope to overcome some problems particularly with extremely small cells by switching to exact arithmetic packages for the underlying geometric predicates.

References

[1] M. Balzer and O. Deussen. Voronoi treemaps. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 49–56, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[2] Mohammad-Hossein Bateni, Mohammad-Taghi Hajiaghayi, Erik D. Demaine, and Mohammad Moharrami. Plane embeddings of planar graph metrics. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 197–206, New York, NY, USA, 2006. ACM.

[3] Mark Bruls, Kees Huizing, and Jarke J. Van Wijk. Squarified treemaps. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42. Press, 2000.

[4] The Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Res*, 11(8):1425–1433, August 2001.

[5] Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[6] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resource for deciphering the genome. *Nucleic Acids Res*, 32(Database issue), January 2004.

[7] T. Koburger, J. Weibezahn, J. Bernhardt, G. Homuth, and M. Hecker. Genome-wide mRNA profiling in glucose starved Bacillus subtilis cells. *Molecular Genetics and Genomics*, 274(1):1–12, 2005.

[8] Kurt Mehlhorn, Stefan Näher, and Christian Uhrig. The LEDA platform of combinatorial and geometric computing. In *ICALP '97: Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, pages 7–16, London, UK, 1997. Springer-Verlag.