CGD – A new Algorithm to Optimize Space Occupation in Ellimaps

Benoît Otjacques[1], Maël Cornil[1], Monique Noirhomme[2] and Fernand Feltz[1]

[1] Public Research Center – Gabriel Lippmann
Department ISC – Informatics, Systems and Collaboration
41, Rue du Brill
L-4422 Belvaux, Luxembourg
{otjacque@lippmann.lu, cornil@lippmann.lu, feltz@lippmann.lu}

[2] University of Namur (FUNDP)
Computer Science Institute
21, Rue Grangagnage
B-5000 Namur, Belgium
monique.noirhomme@info.fundp.ac.be

Abstract How to visualize datasets hierarchically structured is a basic issue in information visualization. Compared to the common diagrams based on the nodes-links paradigm (e.g. trees), the enclosure-based methods have shown high potential to represent simultaneously the structure of the hierarchy and the weight of nodes. In addition, these methods often support scalability up to sizes where trees become very complicated to understand. Several approaches belong to this class of visualization methods such as treemaps, ellimaps, circular treemaps or Voronoi treemaps. This paper focuses on the specific case of ellimaps in which the nodes are represented by ellipses nested one into each other. A controlled experiment has previously shown that the initial version of the ellimaps was efficient to support the perception of the dataset structure and was reasonably acceptable for the perception of the node weights. However it suffers from a major drawback in terms of display space occupation. We have tackled this issue and the paper proposes a new algorithm to draw ellimaps. It is based on successive distortions and relocations of the ellipses in order to occupy a larger proportion of the display space than the initial algorithm. A Monte-Carlo simulation has been used to evaluate the filling ratio of the display space in this new approach. The results show a significant improvement of this factor.

Keywords: Information Visualization, Ellimaps, Hierarchies Visualization

1 Introduction

Generic data structures are often taken as a basis to classify visualization techniques and hierarchies are considered as one of these reference structures. They are encountered in various domains like botany, management or computer science. In

some cases we can observe the importance to visualize not only the data structure itself but also some attributes of the nodes. For instance, the organization chart of a company can also show the number of employees in each division and service. As another example, the administrators of a large data storage system can be equally interested by the structure of the repository, the size of each folder and the last access date of each file.

This paper explores how to visualize weighted hierarchies and tries to find a reasonable balance between competing constraints: a good perception of the data structure, a visual representation of the node weights and an efficient occupation of the display space. It is structured as follows. After the introduction, the state-of-art is discussed and some weaknesses of current propositions are pointed out. Next a new algorithm aiming to tackle a pending issue (i.e. low space occupation) is described. Then we discuss the results of an empirical evaluation of this new approach. Finally some conclusions are drawn and some paths for further research are highlighted.

2 State of art

Grouping is acknowledged as a central issue in perception since the seminal work of Gestaltist psychologists in the 1920's. More recent works have added new principles of grouping to the initial list. Among those principles, two are especially relevant in our study. *Connectedness* expresses the fact that drawing lines among the items of a set is a powerful mean to show that some relationships exist among them [11]. *Common region* is the tendency to group together the elements that lie within the same bounded area [10]. This principle explains, for instance, the success of Venn and Euler diagrams to represent a set of elements (cf. [17] pp. 194-196).

It appears that most of the techniques to visualize hierarchies rely on the *common region* and / or the *connectedness* principles.

In this specific context, *connectedness* means representing nodes by punctual visual objects (e.g. points, icons) and the hierarchical relationship by lines (e.g. straights, curves). The various versions of trees are typical examples of this approach. This class of techniques highlights well the structure of the dataset and makes it very easy to understand. Unfortunately, they do not support very well scalability (cf. [2] p. 149). If the number of nodes grows their layout rapidly becomes very difficult to understand and the navigation tasks become challenging. However, considering their power to visualize the dataset structure, they cannot be neglected and numerous researchers have proposed some improvement to the basic diagrams, such as using polar coordinates [3], hyperbolic geometry [4] or patterns based on circles [6].

For visualizing hierarchies, the *common region* principle materializes by nesting shapes into each other. The nodes are represented by rectangles [13], ellipses [8], complex shapes [1] or even distorted shapes [16]. The hierarchical relationship is represented by the successive inclusion of these shapes. The treemaps [13] are a well-known example of this paradigm and have been by far the most investigated (see [14] for an historical review). They support very well scalability and they have proven to be able to display thousands of nodes in a single screen. They also show the relative weight of the nodes, which allows to rapidly identify the preeminent ones.

Unfortunately, treemaps must also acknowledge some limitations. Lee et al. [5] explain that '*treemaps are appropriate when showing the attribute value distributions is more important than showing the graph structure*'. In order to tackle the issues associated to the initial treemap algorithm, several researchers have proposed new strategies to dimension and position the nested rectangles (e.g. [15] for improving the perception of the data structure, [18] for the ordering consistency).

Ellimaps [8] are more recent and have been significantly less studied (e.g. [9] for an example of use for monitoring web-based platforms). Basically they are founded on layout algorithms similar to the ones used in treemaps. Nevertheless, they use nested ellipses instead of nested rectangles and this offers some new perspectives as well as new challenges. A previous study [8] has shown that (under the conditions summarized hereafter) the ellimaps provide a better support to the perception of the hierarchical structure than squarified treemaps. In addition, they received better score for the subjective perception of the test users. These findings are based on a controlled experiment with 34 subjects who were asked to answer questions like identifying the sibling nodes of a given node N, comparing the weight of several nodes or finding the most weighted node among the children of a node N (cf. *objective measures: time to complete, success rate*) and to fill in a satisfaction questionnaire (cf. *subjective measures: rating of items on a Likert-scale*). The experiment was carried out with sample hierarchies having around 500 nodes and 8 levels of depth.

The authors of the ellimaps acknowledge, however, that the rather poor occupation of the display space is a major drawback of their technique. This limitation raises some issues concerning the scalability of the initial version of this technique. We have therefore decided to tackle this issue in order to reach a better balance between the perception of the dataset structure and the occupation of the display space.

At the end of our literature review we must also point out that beside the techniques that clearly rely on one of the above-mentioned Gestalt principles, we can also find some hybrid approaches. The Space-Optimized Tree [7] uses a rule based on *common region* to divide the display space but uses nodes and links to show the items and their relationships. Another original approach is proposed by Schultz et al. [12] who obtain a space-filling effect with a point-based rendering technique.

3 New algorithm: Combined Geometrical Distortions

3.1 Formalization of the problem

Our research problem consists in finding how to draw the *n* ellipses E_i corresponding to the *n* children N_i into the ellipse E* representing their parent node. It can be formalized by three conditions.

- Including the *n* ellipses E_i into the ellipse E* $(1 \leq i \leq n)$;
- Keeping the ratio of the areas of the ellipses E_i equal to the ratio of their corresponding node weights N_i;
- Occupying a larger proportion of the display space than the initial ellimap algorithm.

Considering the third condition we first need to compute the occupation space of the initial algorithm. It is easy to show that the aspect ratio of the inserted rectangle R that maximizes Area(R) is equal to the ratio (*length of semi major axis / length of semi minor axis*) of the parent ellipse E*. Our subsequent developments are based on this configuration.

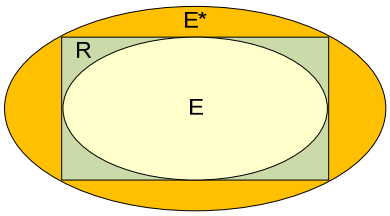

Fig. 1. Computation of display space occupation in initial ellimap algorithm

We need to compute the ratio ρ = Area(E) / Area(E*) which can be expressed as $\rho = \rho_1 . \rho_2$ with ρ_1 = Area(E) / Area(R) and ρ_2 = Area(R) / Area(E*)

This ratio ρ does not depend on the number of ellipses E_i (i.e. children nodes) that are included into E* (i.e. parent node). Indeed, in case of several children the rectangle R is divided into smaller rectangles R_i having a cumulated area equal to Area(R). Each ellipse E_i is inserted into the corresponding R_i. For every ellipse E_i the ratio Area(E_i) / Area(R_i) is identical and consequently is also equal to ρ_1. It is trivial to show that $\rho_1 = \pi/4 \approx 0.7854$. We can also easily compute that $\rho_2 = 2/\pi$. Finally we obtain $\rho = 0.5$. In other words, the initial ellimap algorithm uses only 50% of the available display space of the parent node (cf. ellipse E*) to display its children (cf. ellipses E_i). This obviously appears to be insufficient and it must be increased.

3.2 New Algorithm

Our new algorithm is called Combined Geometrical Distortions (CGD) because it successively applies translations and distortions of the nested ellipses in order to increase their areas while conserving their surface ratio at each hierarchical level.

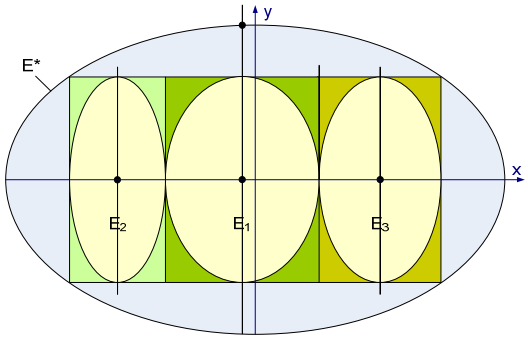

Fig. 2. Initialization step by size-depending ellimap algorithm

The process is initialized (cf. **Fig. 2**) with the former ellimap algorithm with size-depending division (i.e. a rectangle R is inserted into the parent ellipse E*, then R is divided along its largest dimension in *n* smaller rectangles R_i in which the ellipses E_i corresponding to the children nodes are inserted).

In the first step (cf. **Fig. 3**) of CGD we identify the ellipse E_1 of which the center is the closest to the center of E* along the division axis (in the example: x axis). If two ellipses E_i are equally distant from this point we randomly choose one of them. Then we stretch E_1 along the other axis (i.e y axis) to make it tangent to E*. The distortion factor is called D. Next the ellipse E_1 is stretched with a factor k/D along the x axis ($\sqrt{2}$ seems to be a good value for k and it is used in the evaluation, cf. section 4).

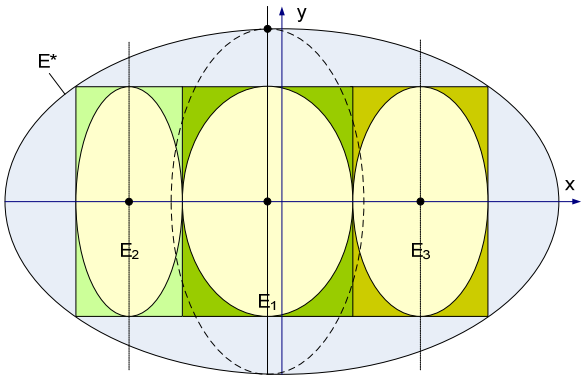

Fig. 3. CGD algorithm: step 1

In the second step the ellipses E_i ($i \neq 1$) are translated along the x axis in order to be tangent again (cf. **Fig. 4**).

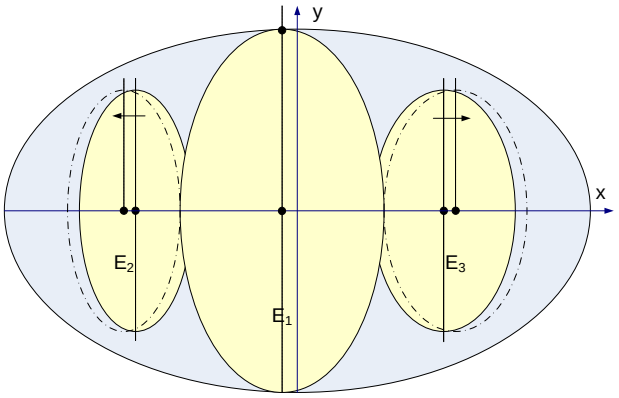

Fig. 4. CGD algorithm: step 2

In the third step the ellipses E_i ($i \neq 1$) are distorted similarly as E_1 has been in the first step: stretched along the y axis to be tangent to E* and stretched along the x axis.

Note that these steps increase the area of the ellipses E_i by a factor of k. However, their ratio is kept constant.

Then (step 4) the ellipses E_i ($i \neq 1$) that have just been distorted are translated along the x axis to be tangent again.

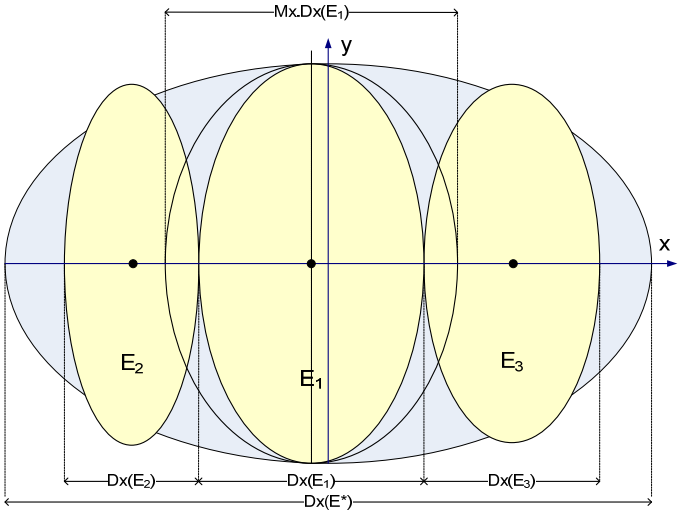

Fig. 5. CGD algorithm: step 5

We stretch then (step 5) the E_1 ellipse along the x axis with a magnifying factor Mx that is computed in order to use the remaining space along the x axis:

$Mx = Dx(E^*) / [Dx(E_1) + Dx(E_2) + Dx(E_3)]$

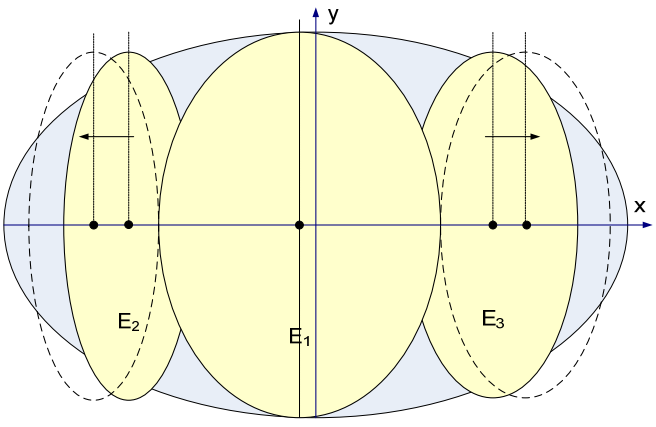

Fig. 6. CGD algorithm: step 6

In the sixth step the ellipses E_i ($i \neq 1$) are translated to keep the tangency property. Then (step 7) they are stretched along the x axis with the magnifying factor Mx.

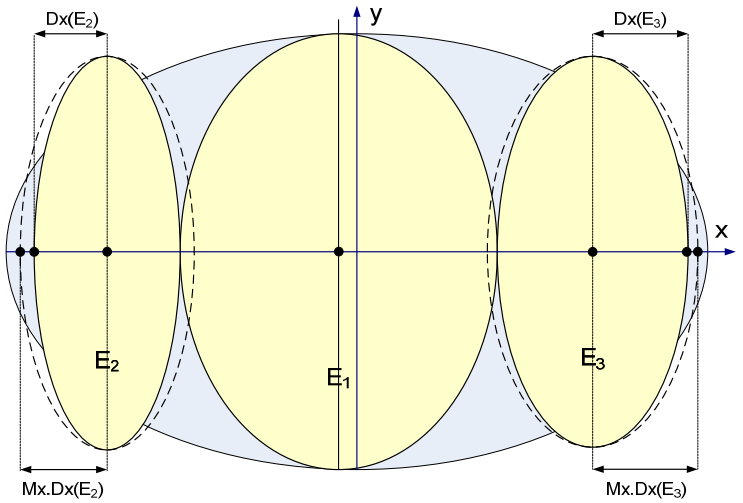

Fig. 7. CGD algorithm: step 7

The ellipse E_i ($i \neq 1$) are then translated again along the x axis to keep the tangency property (step 8, cf. **Fig. 8**).

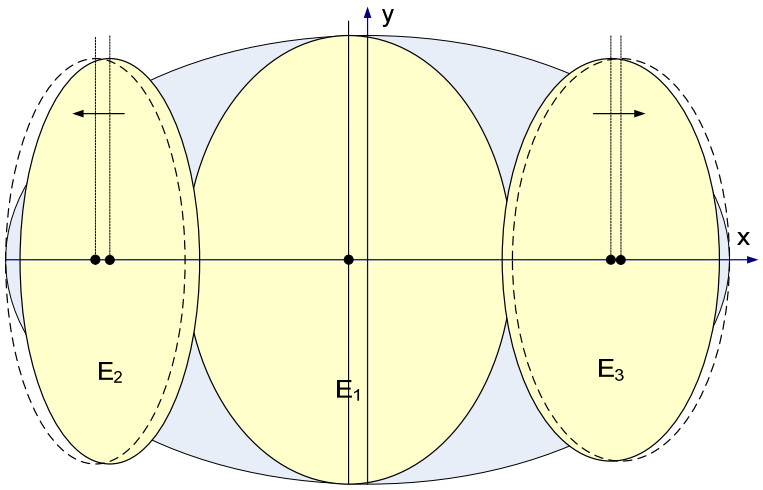

Fig. 8. CGD algorithm: step 8

In the ninth step, an empirical approach is used to decrease the size of the ellipses E_i (cf. **Fig. 9**). This iterative strategy (steps 9 to 11) was found to be the most effective to reach a well balanced result between display space occupation and computation time. For each ellipse E_i we compute the distance $Sy(E_i)$ defined as the difference between $Dy(E_i)$ (i.e. semi-axis of E_i along the y axis) and the y coordinate of the intersection point I_i between the ellipse E* and the vertical symmetry axis of E_i.

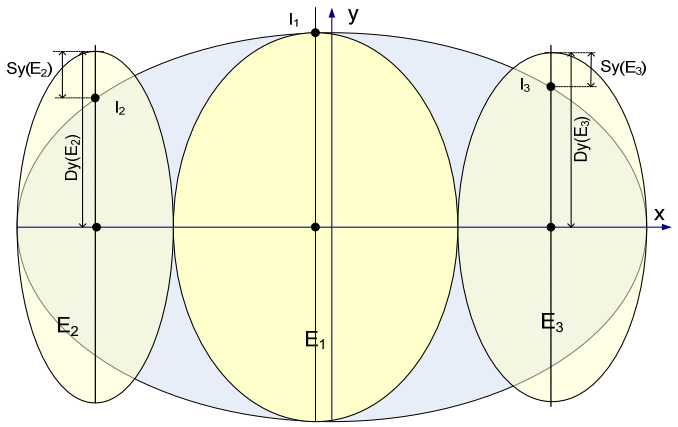

Fig. 9. CGD algorithm: step 9

Next we compute $Syr(E_i) = Dy(E_i) / [Dy(E_i) – Sy(E_i)]$ and
Syr-max = max { $Syr(E_i)$} with i = 1 to n.
If Syr-max > 1, we must go to the tenth step to decrease the size of the ellipses E_i .
If Syr-max ≤ 1, we go to the eleventh step.

Note that (Syr-max < 1) does not mean that every ellipse E_i is fully located inside the ellipse E*. However, it is almost always true and if it is false our algorithm is still valid because the eleventh step will correct any inaccuracy from this view point.

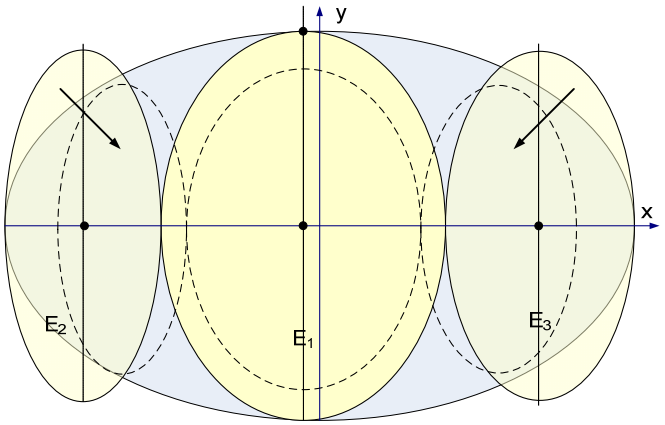

Fig. 10. CGD algorithm: step 10

The tenth step aims to decrease the size of every ellipse E_i with an identical factor R along the x and y axis. Having explored several strategies to compute R, it has appeared that a good option was to choose a constant function (e.g. R = 0.95). After the reduction step, some translations along the x axis are applied to the ellipses E_i in order to keep them tangent. Then we go back to step 9 to evaluate whether all the ellipses E_i are (approximately) located inside E*.

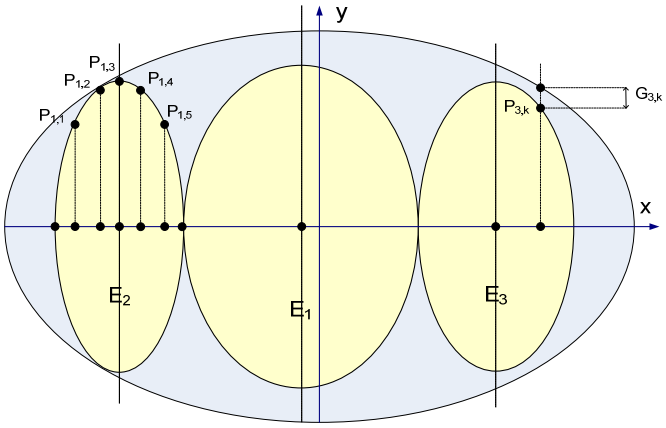

Fig. 11. CGD algorithm: step 11

Our purpose at the end of the tenth step is to have the best layout possible because it will be used in the eleventh step as the initial point of a process that is more intensive in computation than the previously described operations.

The eleventh step checks whether the ellipses E_i are located inside E* by computing the coordinates of *m* points $P_{i,j}$ of their perimeter. If at least one of the points $P_{i,j}$ for one of the ellipses E_i is located outside the ellipse E*, we apply again the reduction factor R to every ellipse E_i. When all points $P_{i,j}$ are inside the ellipse E*, we iteratively increase the size of all ellipses E_i with a magnifying factor M_k (e.g. 1.1) until one of the points $P_{i,j}$ is either tangent to the perimeter of E* (tangency for $P_{i,k}$ is defined by the expression $G_{i,k} < \varepsilon$, see **Fig. 11**) (case A) or outside E (case B). This moment of the iterative process is called the critical step. In the case A, the iterative process is stopped and the ellipses E_i are the final ones. In the case B, a new factor M_{k+1} (with $M_{k+1}<M_k$) is chosen (e.g. 1.01). The iterative process is then resumed at the step just before the critical one and the ellipses are iteratively magnified with the new ratio M_{k+1}.

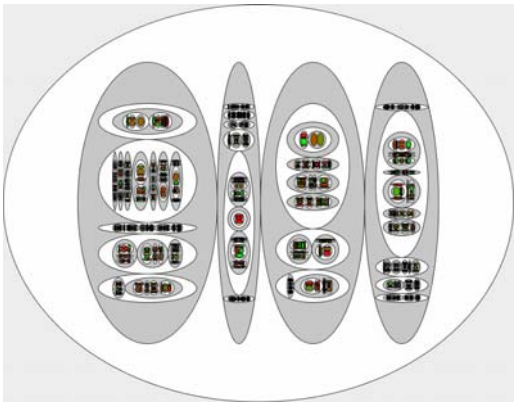

Fig. 12. Initial size-depending algorithm for a simulated hierarchy with 4477 nodes

The final result is illustrated below. The CGD algorithm (cf. **Fig. 13**) can advantageously be compared to the initial size-depending one (cf. **Fig. 12**).

We can also mention at this point that our Java implementation of CGD running on a classic PC (Windows XP, Intel Core 2 CPU at 2.40 GHz, 1 GB RAM) produces graphics allowing real time interaction up to several thousands of nodes (tested with datasets up to 10,000 nodes).

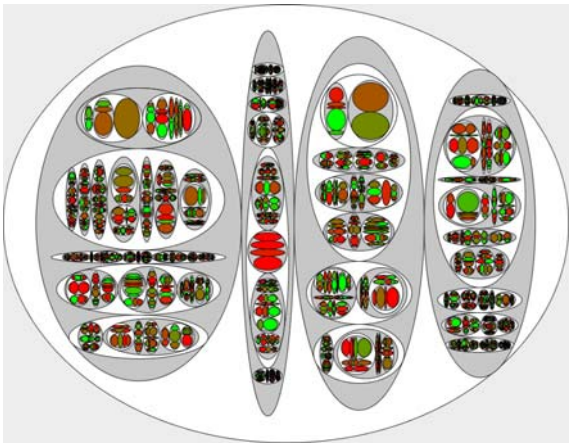

Fig. 13. CGD algorithm with the same hierarchy as Fig. 12

It is important to point out at this stage that any percent of gain in display space occupation can have a significant influence on the final result because it applies to every level of the hierarchy.

In fact, the occupation of the display space decreases at an exponential rate with the level of depth in the hierarchy.

With the hypothesis of a mean occupation rate γ at each level and considering that the ratio between the area of the root ellipse and the rectangular display space is known ($= \pi/4$), we can approximate the cumulated area of all ellipses E_i of level *n* with (Eq1):

$$\text{occupied space(level n)} = \gamma^n \,.\, (0.25 \,.\, \pi \,.\, \text{display space}) \text{ with } \gamma \in]0, 1] \qquad \text{(Eq.1)}$$

4 Evaluation

4.1 Mean rate of display space occupation

In order to evaluate the gain in space usage of the CGD algorithm we have identified two parameters having a potential influence on the occupation of the display space: the mean number of children by node and the ratio of the node weights. This is due to the fact that for each node having some children a new instance of CGD is executed

to define the most appropriate ellipses to be drawn. Therefore neither the total number of nodes in the hierarchy nor the number of depth levels influence the space occupation for each instance of CGD. Nevertheless, it is obvious that those parameters influence the global picture. In this first sub-section we evaluate the γ factor of the equation (Eq.1) and in the second one we study its influence on the global rate of display space occupation.

Our evaluation is based on a Monte Carlo approach. A large number of hierarchical datasets H_k were generated. They are hypothesized to be a significant sample of the datasets encountered in reality. Considering the preceding discussion, we only need to generate datasets having one level of depth with varying values for the number of children nodes of the root and for the distribution of the weights of those children nodes.

For each hierarchy H_k we compute for the root N_k the value of ODS_k (Occupation of the Display Space) as the ratio of the cumulated area of the *n* ellipses representing its children $C_{i,k}$ and the ellipse representing N_k : $ODS_k = [\Sigma_{i=1..n} \text{Area}(C_{i,k})] / \text{Area}(N_k)$.

Next we compute the mean value μ_{ODS} and the standard deviation σ_{ODS} of the ODS parameter for the complete sample of hierarchies $\{H_k\}$.

Our random datasets generation module permits to set the value of the mean and standard deviation of the number of children by node (according to a lognormal law because negative values does not make sense) as well as the mean and the standard deviation of the node weight (according to a lognormal law for the same reason as above). The complete set of hierarchies $\{H_k\}$ is composed by aggregating some subsets $\{H_{k,s}\}$. Each subset $\{H_{k,s}\}$ is randomly generated by setting a value for the four parameters mentioned just above. In our evaluation the values set are given in Table 1. A random set $\{H_{k,s}\}$ was generated for each combination of the boxes, which means that 25 subsets $\{H_{k,s}\}$ were produced. Each subset $\{H_{k,s}\}$ contains 100 hierarchies.

Table 1. Parameters settings for the random generation of hierarchies

Number of children by node	$\mu_{set} = 2$ $\sigma_{set} = 2$	$\mu_{set} = 4$ $\sigma_{set} = 4$	$\mu_{set} = 6$ $\sigma_{set} = 6$	$\mu_{set} = 8$ $\sigma_{set} = 8$	$\mu_{set} = 10$ $\sigma_{set} = 10$
Children weight	$\mu_{set} = 100$ $\sigma_{set} = 50$	$\mu_{set} = 100$ $\sigma_{set} = 75$	$\mu_{set} = 100$ $\sigma_{set} = 100$	$\mu_{set} = 100$ $\sigma_{set} = 125$	$\mu_{set} = 100$ $\sigma_{set} = 150$

At this stage it is important to mention that the random draw can generate some hierarchies where the root only has one child. In this case, the CGD algorithm makes the single child ellipse identical to the parent one. This behavior can be considered as a favorable bias that artificially increases the value of μ_{ODS}. Therefore, to strengthen the validity of our results, we have excluded the hierarchies corresponding to this case from the complete sample set $\{H_k\}$. We have also removed from $\{H_k\}$ the hierarchies H_k where the root has no children nodes. Finally the resulting set $\{H_k\}_{filtered}$ contains 2015 hierarchies. The global results of this simulation are displayed in Table 2. Note that the hierarchies H_k were not sorted (e.g. according to the node weights).

First we mention the distribution parameters of two main characteristics of $\{H_k\}_{filtered}$: the number of children of the root (NCh) and the ratio between the largest and smallest weight of these children (WR). For a node N having *p* children N_m, WR is defined by the expression: $WR = \max_{m=1..p}(weight(N_m)) / \min_{m=1..p}(weight(N_m))$.

Table 2. Global results

	Min	Max	Mean	Std. Dev.
NCh	2	119	7.06	6.78
WR	1.01	619.08	13.69	30
ODS - CGD	0.578	0.757	0.682	0.038
ODS – initial algorithm	0.5	0.5	0.5	0

We can observe that the mean rate of display space occupation ($\mu_{ODS} = 0.682$) is above the one of the initial ellimap algorithm (0.50). A t-test and a non-parametric Wilcoxon rank test showed that the observed difference between the mean ODS in both cases is highly significant ($p < 0.001$). Those figures confirm the impression given by comparing Fig. 12 and Fig. 13. Therefore it is clear that the CGD represents a step forward compared to the initial proposition. In our simulation the worst ODS for CGD (0.578) is still better than the ODS for the initial ellimap algorithm.

The low standard deviation of ODS is another notable result because it shows that the performance of CGD is stable for a relatively large range of configurations of the hierarchies to be displayed.

Finally, with the hypothesis of a mean occupation rate ($\gamma_{CGD} = 0.682$) at each level, the cumulated area of all ellipses E_i of level *n* can be approximated by (Eq. 2) for the CGD algorithm.

$$\text{occupied space(level n)} = (0.682^{n} \,.\, 0.25 \,.\, \pi \,.\, \text{display space}) \quad \text{(Eq.2)}$$

Table 3. Comparison of display space occupation for CGD and initial algorithms

		ODS		
	Level (*n*)	Initial algo.	CGD algo.	Ratio
Display space		100	100	1
Root	0	78.54	78.54	1
Level 1	1	39.27	53.56	1.36
Level 2	2	19.63	36.53	1.86
Level 3	3	9.82	24.91	2.54
Level 4	4	4.91	16.99	3.46
Level 5	5	2.45	11.59	4.72
…	…	…	…	…
Level 10	10	0.08	1.71	22.29

The Table 3 concretely illustrates the meaning of (Eq. 2) with numbers. At the second level of depth, the CGD algorithm uses almost the double of the display space

than the initial algorithm to draw the corresponding ellipses. At the fifth level, this ratio is nearly five and at the tenth level it reaches more than twenty.

While the absolute values of ODS may appear low for high levels of depth, it must be reminded that the remaining part of the display space is used to show the ellipses of the upper levels and therefore the hierarchical structure itself.

4.2 Exploration of the global range of display space occupation

While the γ factor is important to evaluate the performance of the algorithms, it may be useful to draw the attention on its statistical nature. Because it has been computed as a mean of observed values, the real value for a given instance of ellimap will most of the time be different. This phenomenon is illustrated in Table 4. In the most favorable cases ($\gamma = 0.75$ at every level), the ODS value is significantly higher and in the worst cases ($\gamma = 0.58$ at every level) it is much lower. Nevertheless, those extreme values have a very little probability to be observed. Therefore we have added in Table 4 the results of a real random draw (number of depth levels: 6; number of nodes: 15176; mean WR: 13.04; mean ODS = 0.70).

Table 4. Influence of the statistical nature of γ_{CGD}

	Level (n)	ODS		Real Example
		Very lucky draw $\gamma = 0.75$	Very unlucky draw $\gamma = 0.58$	
Display space		100	100	100
Root	0	78.54	78.54	78.54
Level 1	1	58.90	45.55	53.56
Level 2	2	44.18	26.42	39.29
…	…	…	…	…
Level 5	5	18.64	5.16	13.29

5 Conclusion

This paper proposes a new algorithm called CGD to increase the low occupation of the display space in the ellimap visualization technique, which has been identified as its major weakness. The evaluation of CGD shows that the mean occupation of the display space is significantly higher than what was observed with the initial proposition. It may be noted that, to the limit of our knowledge, this evaluation is the first one to quantitatively study the space occupation of a visualization technique of hierarchies. This new approach reaches an interesting balance between the representation of the data set structure, the weights of the nodes and the occupation of the display space. In the future, a user-based evaluation similar to Otjacques et al.'s one [5] is envisaged to assess the influence of this geometrical improvement in terms of data cognition.

References

1. Balzer M, Deussen O, Lewerentz C.: Voronoi Treemaps for the Visualization of Software Metrics. In: ACM Symposium on Software Visualization 2005, pp. 165-172. ACM Press, New York (2005)
2. Card S., Mackinlay J. and Shneiderman B.: Readings in Information Visualization, Using Vision to Think. Morgan Kaufmann, San Francisco, California (1999)
3. Chi E. H., Pitkow J., Mackinlay J., Pirolli P., Gosswiler R. and Card S.K.: Visualizing the Evolution of Web Ecologies. In: ACM Conference on Human Factors in Computing Systems 1998 (CHI'98), pp. 400-407. ACM Press, New York (2005) (1998)
4. Lamping L. and Rao R.: Laying out and Visualizing Large Trees Using a Hyperbolic Space. In: ACM Symposium on User Interface Software and Technology (UIST 1994), pp. 13-14. ACM Press, New York (1994)
5. Lee B., Parr C., Plaisant C., Bederson B., Veksler V., Gray W. and Kotfila, C. TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. In: IEEE Transactions on Visualization and Computer Graphics, 12(6), pp.1414-1426. IEEE Press, NewYork (2006).
6. Lin, C. and Yen, H.: On Balloon Drawing of Rooted Trees. In: Healy P., Nikolov N. (eds) GD'05. LNCS, vol. 3843, pp. 285-296. Springer, Heidelberg (2005)
7. Nguyen Q. and Huang M.: A Space-Optimized Tree Visualization. In: IEEE International Symposium on Information Visualization, pp. 85-92. IEEE Press, New York (2002)
8. Otjacques B., Collin P., Feltz F. and Noirhomme M.: Ellimaps: une technique basée sur la loi d'inclusion pour représenter des hiérarchies avec nœuds pondérés. Revue d'Intelligence Artificielle. 22(3-4): pp. 301-327 (2008)
9. Otjacques B., Noirhomme M., Gobert X., Collin P. and Feltz F.: Visualizing the activity of a web-based collaborative platform, In: International Conference on Information Visualization (IV'07), pp. 251-256. IEEE Computer Society Editions (2007).
10. Palmer S.E.: Common Region: A new principle of perceptual grouping. Cognitive Psychology. 24: pp. 436-447 (1992)
11. Palmer S.E. and Rock I.: Rethinking perceptual organization: The role of uniform connectedness. Psychonomic Bulletin and Review 1(1): pp. 29-55 (1994)
12. Schultz H.-J., Hadlak S. and Schumann H.: Point-Based Tree Representation: A new Approach for Large Hierarchies. In: IEEE Pacific Visualization Symposium (PacificVis'09), IEEE Press, NewYork (2009).
13. Shneiderman B: Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. In: ACM Transactions on Graphics. 11(1): pp. 92-99. ACM Press, New York (1992)
14. B. Shneiderman. Treemaps for space-constrained visualization of hierarchies, University of Maryland, Human – Computer Interaction Lab, Internet address: http://www.cs.umd.edu/hcil/treemap-history/, accessed 7 Apr. 2009.
15. van Wijk JJ, van de Wetering H.: Cushion TreeMaps, In: IEEE International Symposium on Information Visualization, pp. 73-78. IEEE Press, NewYork (1999)
16. Vliegen R, van Wijk J.J. and van der Linden E.: Visualizing Business Data with Generalized Treemaps. In: IEEE Transactions on Visualization and Computer Graphics, 12(5), pp. 789-796. IEEE Press, NewYork (2006)
17. Ware C.: Information Visualization, Perception for Design, 2nd Edition. Morgan Kaufmann, San Francisco, California (2004)
18. Wood J.: Spatially Ordered Treemaps. In: IEEE Transactions on Visualization and Computer Graphics. 14(6), pp. 1348-1355. IEEE Press, NewYork (2008)