An energy-based model to optimize cluster visualization

Taoufiq Dkaki
Institut de Recherche en Informatique de Toulouse,
UMR5505, CNRS
Université de Toulouse
Toulouse, France
Taoufiq.dkaki@irit.fr

Josiane Mothe
Institut de Recherche en Informatique de Toulouse,
UMR5505, CNRS
Université de Toulouse
Toulouse, France
Josiane.mothe@irit.fr

Abstract— **Graphs are mathematical structures that provide natural means for complex-data representation. Graphs capture the structure and thus help modeling a wide range of complex real-life data in various domains. Moreover graphs are especially suitable for information visualization. Indeed the intuitive visual-abstraction (dots and lines) they provide is intimately associated with graphs. Visualization paves the way to interactive exploratory data-analysis and to important goals such as identifying groups and subgroups among data and helping to understand how these groups interact with each other. In this paper, we present a graph drawing approach that helps to better appreciate the cluster structure in data and the interactions that may exist between clusters. In this work, we assume that the clusters are already extracted and focus rather on the visualization aspects. We propose an energy-based model for graph drawing that produces an esthetic drawing that ensures each cluster will occupy a separate zone within the visualization layout. This method emphasizes the inter-groups interactions and still shows the inter-nodes interactions. The drawing areas assigned to the clusters can be user-specified (prefixed areas) or automatically crafted (free areas). The approach we suggest also enables handling geographically-based clustering. In the case of free areas, we illustrate the use of our drawing method through an example. In the case of prefixed areas, we first use an example from citation networks and then use another example to compare the results of our method to those of the divide and conquer approach. In the latter case, we show that while the two methods successfully point out the cluster structure our method better visualize the global structure.**

Keywords—graphs; clusterd graphs; graph drawing; visualization;

I. Introduction

Automatic graph drawing has been an active research area over the last decades. It has been boosted by the increasing interest for graph structures which provide natural and intuitive ways for complex-data representation. Indeed, Graphs have proven their effectiveness in many fields, such as social network [1], software engineering [2], electronic circuit design [3], database design [4] etc... More generally, graphs effortlessly and perfectly model data that can be seen as sets of objects sharing some relationships. The major benefit of such abstract representation is that it easily transforms into a meaningful drawing which can be the core of some information visualization approach and the beginning of some data exploratory analysis process. Visualization paves the way to interactive exploratory data-analysis and to one of its targeted goals: identifying groups and subgroups among data and allowing users to better understand how these groups interact with each other.

Many strategies have been used to draw graphs. Most popular algorithms are based on force-directed methods [5, 6]. They are rather simple but give satisfactory results for graph with few hundreds of nodes. Some methods are also scalable and demonstrate their ability to handle larger graphs [7]. These methods are effective to show the cluster structure of a graph when the clusters stem from the core graph structure; that is to say when it can be derived from the edges. However existing methods fail to represent the cluster structure of a graph when this structure is derived from extra-graph properties; for example from nodes attributes.

Finding effective visualization for clustered graphs depends on the application domains and the nature of the clustering. We can distinguish the cases when a prefixed visualization area is associated with each cluster and the cases when the visualization areas are computed. Indeed, in various cases, the spatial constraints are part of the model. This is for example the case of geographic graphs or electronic circuit boards that usually comprise a large number of components which have to fit geographic constraints. This is also the case when the user is asked to make this association.

In this paper, we propose an energy-based model for graph drawing that produces an esthetic drawing that ensures each cluster will occupy a separate zone within the visualization layout. In this work, we assume that the clusters are already calculated and known. Moreover, we consider that clusters can have been obtained independently from the core graph-structure (that is to say edges and nodes), for example from attribute data in an attributed graph or colored graph. We also hypothesize that the drawing areas assigned to the clusters can be user-specified (prefixed areas) or automatically crafted (free areas). Our method seeks good layout for clustered graphs for both prefixed and free areas. Good layouts are generally associated to esthetics and ergonomics [8]; our method aims at providing drawings that allow users to investigate and explore the data by properly apprehending the clusters and the inter-clusters interactions while still esthetically showing the inter-nodes connections. We propose a new energy-based method to deal with this problem. Our method modifies the force system proposed in Fruchterman and Reingold [5] and provides an approach that handles user-selected cluster areas.

The remaining of this paper is organized as follows. In section II we give an overview of graph drawing methods. In section III, we present the structure of clustered graphs, describe our main contributions and extend it to the case of prefixed cluster-areas. Section IV presents three use case examples. In the case of prefixed areas, we first use an example from citation networks [9] and then use an example from [10] to compare the results of our method to those of the divide and conquer approach. In the latter case, we show that while the two methods successfully point out the cluster structure our method better visualize the global structure.

II. RELATED WORKS

Graph drawing is a long-standing issue in mathematics and computer science. Indeed, this flourishing domain can be traced back to the sixties and the barycenter method of Tutte [11] although it is in the early eighties that computerized graph drawing really emerged [12][13]. Different strategies have been proposed to answer the challenge of drawing graphs. Some of these strategies are suitable for general graph structures e.g. orthogonal layout methods [14], circular layout methods [15] while others are only suitable for specific structures such as trees or hierarchy [16].

Eades [17] popularized one of the most used general drawing strategies –namely energy-based layout. Energy-based methods view the graph as a mechanical system in which each vertex is seen as a steel ring and each edge as a spring connecting the vertices –rings- at its endpoints. The induced attractive and repulsive forces drive the graph to a minimal energy state. This heuristic paved the way for one of most popular force-directed algorithms [6][5]. Kamada and Kawai [6] use adaptable spring forces. The force of each spring is proportional to the graph theoretic distance (shortest-path length) between its endpoints. Fruchterman and Reingold adds electrical charges to each vertex and ends with electrical repulsion forces between vertices [5]. The authors also add the notion of “cooling temperature” that lowers the amounts of vertices’ displacement as the layout becomes finer. Further propositions such as multi-level or multi-scale approaches followed mainly to address the scalability problem. For example [18] proposes a multi-scale method based on a vertex filtration, [19] quickly draw a graph in a very high dimensional space plunge it into a 2D or 3D space using principal components analysis. Moreover, [20] induces a coarser graph by recursively clustering the original graph and draws the obtained graphs in inverse order of the previous clustering process.

The problem of visualizing clustered graphs can be divided into two problems depending on the nature of the clustering. Indeed, algorithms dealing with clustered graphs must differ one from another depending if the graphical areas associated to each cluster are pre-defined (user-defined) or if they are defined by the visualization algorithm it-self.

Many works consider self-organizing cluster areas. The most significant of them [21][22][23][24][25][26] either modify the energy function or add 'dummy' nodes and related edges to lay out a “nice” drawing that enhances the visual identification of clusters by ensuring that the nodes belonging to a given cluster are always placed close to each other. Some energy based approaches use more sophisticated strategies to achieve the separation of clusters. For instance, [25] use a constraint optimization technique to keep nodes strictly inside non-overlapping rectangular boundaries. IPSep-Cola [23] allows separation constraints, which enforce a minimum horizontal or vertical separation between nodes -or clusters. However the underlying linear constraints present some limitations due to the mandatory rectangular shape that the clusters must fit which may end in space waste. Another limitation is related to the decoupling of intra and inter-clusters visualizations which obviously favors the local drawings and prevents appreciating the influence of inter-clusters edges. [24] sets up an approach that clearly displays different clusters of graph by using a new energy model called LinLog which use a cut ratio as a measure for the coupling of two disjoint sets of vertices. Noack proves that a graph having the minimum energy according to LinLog has its clusters properly displayed: each cluster is separated from the remaining graph vertices. The distance of each cluster from each of the remaining graph vertices is inversely proportional to the "coupling". [27] proposes a method based on the “level of detail visualization” to draw a clustered graph where the clusters are created in the hierarchical way. The approach allows users to change their view from a very abstracted to a very detailed visualization. It only modifies the visual representation of the graph without altering its structure. Alternatively, rather than been self-organized, the cluster areas can be geographically constrained as in [28]

What distinguishes the energy-based approach we propose in this paper from the works represented above is how clusters are defined -or uncovered. Many approaches consider clusters as sets of vertices with many internal edges and few edges to outside vertices: the clusters are graph relationship dependent. Under this assumption, clusters are often dense and strongly-connected graphs components. In other words, the density of clusters is generally greater than the one of the entire graph. This is a quite restricting definition since graph vertices can group together and form clusters not only according to the graph binary relationship but also according to some exogenous or attribute based information . For example, in a collaboration or citation graph, the clusters may be built in relation to gender, age, geography or other similarity relationship among vertices that has nothing to do with the citation or the collaboration relationship. The approaches from the literature do not apply well to this kind of clustered graphs.

Our motivation is to propose a method for drawing clustered graph where clusters can be any set of vertices. Under these assumptions, a cluster can be any set of graph vertices. It is even possible to consider clusters containing only pairwise disconnected nodes.

III. Contribution

Before describing our method, we provide some definitions we will be using in the paper.

A. *Definitions and notations*

1) *Graphs*

A graph consists of a set of nodes and a binary relationship between the nodes that induces a set of edges. A graph G is then a couple (V, E), where V is a finite set of vertices and $E \subseteq V \times V$ a finite set of edges.

2) *Clustered graphs*

A clustered graph G is a triplet $G(V, E, P)$ where V is a finite set of vertices, $E \subseteq V \times V$ a finite set of edges and P is a partition over V.

$P \subset \mathscr{P}(V)$ the power set of V, $\bigcup_P C_i = V$ and $\forall\, C_i, C_j$ two elements of V, P $C_i \cap C_j = \emptyset$

The number of elements in P corresponds to the number of clusters in G.

The Fig. 1 illustrates the structure of clustered graphs –the graph in this figure contains three clusters.

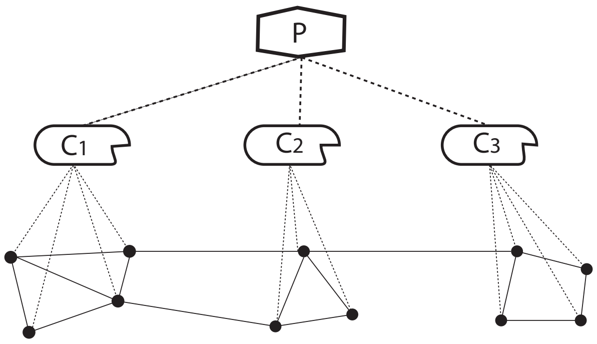

Fig. 1. Illustration of the structure of a clustered graph

3) *Further notations*

If X is a set, $|X|$ represents the number of elements of X.

If X is a vector $||X||$ represents the Euclidian norm of X.

B. *Defining our drawing models*

When building a force-directed or energy-based model for graph drawing, one must define its three major components: the initial nodes positioning, the force system that controls how the graph vertices move to meet their final positions and, since the drawing is an iterative process, the stop condition that determines when the system reaches stability).

As said before, the problem deals with finding the best drawing for a clustered graph. This mainly includes that all vertices belonging to the same cluster must be placed in a restricted and exclusive area. The areas associated to the clusters can either be automatically determined or be chosen by the user. The resulting drawing must meet these placement constraints while still optimizing the graph global layout as well as the clusters' local-layouts.

Whatever the types of areas (pre-defined and self-defined areas), the methods we propose for drawing clustered graphs are based on the force directed placement model proposed by Fruchterman and Reingold [5]. For this reason, we briefly present this algorithm.

1) *Fruchterman and Reingold ' algorithm*

The force directed placement model proposed in [5] is one of the most popular methods for graph drawing.

Fruchterman and Reingold ' approach begins with a random initial positioning of the graph vertices. Then, it iterates two steps that consist in 1) computing the attraction and repulsion forces exerted on each vertex and 2) positioning adjustment of vertices that follows this computing. Each couple of vertices v_i and v_j, exerts on each other two opposite repulsive forces of the same intensity. $f_r = \frac{K^2}{\|\overrightarrow{v_1 v_j}\|} \frac{\overrightarrow{v_1 v_j}}{\|\overrightarrow{v_1 v_j}\|}$ is the repulsive force that v_i exerts on v_j. K is a constant that depends on the visualization-space size and on the number of displayed vertices $K \leftarrow \sqrt{\frac{\text{surface}}{|V|}}$. K is the optimal distance connected nodes.

Only related vertices (for example v_i and v_j) exert attractive and opposite forces of the same intensity on each other (see Fig. 2).

$f_a = \frac{\|\overrightarrow{v_1 v_j}\|^2}{K} \frac{\overrightarrow{v_1 v_j}}{\|\overrightarrow{v_1 v_j}\|}$ is the repulsive force that v_i exerts on v_j.

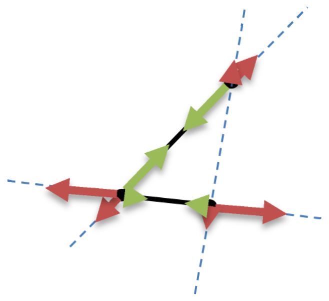

Fig. 2. The sets of attractive and repulsive forces exercised within a graph of three vertices and two edges. Attractive forces are in green and repulsive ones are in red

A temperature model is used and a cooling function ensures that the displacement of each vertex is majored by a maximum displacement value that decreases over time. This traduces the idea that, as the layout becomes better the amount and need for adjustment becomes smaller.

The stopping condition is based on energy minimization but rather than computing the energy and finding its minimum or a local minimum, most algorithms perform a fixed number of iterations.

Fig. 3 shows a pseudo code of Fruchterman and Reingold ' algorithm.

```
surface ← W * L;   # W and L are the width and length of the frame
G ← (V,E);         # The vertices are assigned random initial positions
k ← $\sqrt{\frac{surface}{|V|}}$;
function $f_a(x)$:
    return $\left(\frac{x^2}{k}\right)$
function $f_r(x)$:
    return $\left(\frac{k^2}{x}\right)$
$t = t_0$;                   #initialise temperature t
for i ← 0 to nbIterations:
# Compute the repulsive forces on vertices
  for v in V:
              # Each vertex is associated two 2D vectors:
              # position (pos) and displacement (disp)
    v.disp ← (0,0);
    for u in V:
      if (u ≠ v: )
            # δ is the difference vector
        δ ← v.pos - u.pos;
        v.disp ← v.disp + $\frac{\delta}{\|\delta\|}$ * $f_r(\|\delta\|)$;
 # Compute the attractive forces on vertices
  for e in E:
     # u an v are the endpoint vertices of e
     δ ← e.v.pos - e.u.pos;
    e.v.disp ← e.v.disp - $\frac{\delta}{\|\delta\|}$ * $f_a(\|\delta\|)$;
    e.u.disp ← e.u.disp + $\frac{\delta}{\|\delta\|}$ * $f_a(\|\delta\|)$;
 # Limit the max displacement  using tenperature
 # and keep them inside the frame
  for v in V:
    v.pos ← v.pos+ $\frac{v.disp}{\|v.disp\|}$ * min(v.disp, t);
  v.pos.x ← min(W/2, max(-W/2, v.pos.x));
  v.pos.y ← min(L/2, max(-L/2, v.pos.y));
           # Reduce the temperature
   t = cool(t);
```

Fig. 3. Algorithm 1 : Fruchterman and Reingold ' algorithm.

2) Clustered-graph drawing :the case of self-defined cluster areas.

In this section, we focus on the case when areas associated to clusters are defined by the drawing algorithm it-self. When dealing with clustered-graph drawing, the main goal is to find a way to draw vertices from a given cluster near to each other; while vertices from different clusters must be drawn as far as possible from each other. When the clustering is performed directly using the graph relationships (edges), the clusters are generally sets of vertices that form dense sub-graphs [24] relatively to the entire graph. In force-based model such as in [5], these vertices exert naturally attractive forces on each other making them close in the final graph drawing. The edges relating vertices from different clusters are infrequent which tends to separate the clusters in the final drawing. Since we are also interested in clusters that do not stems from the graph relationship, the force system has to be revisited to create attractive forces between the vertices that compose a given clusters in order to make them closer in the final drawing. Unlike the methods proposed in [EH00, FT04] that create a 'dummy' node for each cluster, we rather add "invisible" edges -that lead to additional attractive forces- between non-connected nodes of the same cluster. This is intended to keep the nodes of a given cluster close together in the visualization space. We consider that the layout of a cluster is good if it resembles to some extend the layout of that cluster when it is considered as a separated graph. The forces generated by the additional edges must be lowered to prevent them from over affecting the internal layout of clusters. In other words, we want these extra edges to help creating homogenous and separated clusters without over-modifying the relative placement of the cluster nodes. To do so, we adjust the intensity of the additional attractive forces relatively to the one of real edges.

The main differences with the approach used in [5] are:

1) we distinguish two types of spring forces and add attractive forces between non-connected nodes of a same cluster (some kind of invisible edges) and

2) we increase the repulsive forces between vertices from different clusters.

Indeed, our force model is built as a combination of three types of springs and an electrical force (in the next algorithm we will add another electrical force):

- Real springs that link adjacent nodes from a same given cluster,
- Invisible springs that link non adjacent nodes from a same cluster (they correspond to invisible links that represents the other criteria making the nodes belonging to the same cluster). These springs are weakened, using an attenuation function *reduce_in*, in order to produce attenuated forces and store less energy,
- Real springs that link adjacent nodes from different clusters. These springs link are also weakened, in a different manner using an attenuation function *reduce_out*, to produce a different force comparing to real springs that link adjacent nodes from a same cluster,
- The repulsive forces that concern vertices from two different clusters are amplified using the function *enhance*.

Fig. 4 illustrates the force system we propose. The colors help distinguishing the different kinds of force.

It is Attractive and repulsive forces depend on whether v_i and v_j belong or not to the same cluster

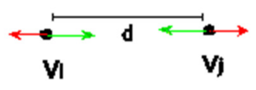

Fig. 4. Attractive and repulsive forces depend on whether vi and vj belong or not to the same cluster

The remaining of Fruchterman and Reingold algorithm components such as temperature and optimal distance are leaved unchanged in our algorithm.

Algorithm 2 presents the algorithm we propose.

```
surface ← W * L;  # W and L are the width and length of the frame
G ←(V,E,P);  # The vertices are assigned random initial positions
k ← $\sqrt{\frac{surface}{|V|}}$;
function $f_a(x)$:
    return $\left(\frac{x^2}{k}\right)$
function $f_r(x)$:
    return $\left(\frac{k^2}{x}\right)$
$t = t_0$;                    #initialise temperature t
for i ← 0 to nbIterations:
# Compute the repulsive forces on vertices
  for v in V:
                  # Each vertex is associated two 2D vectors:
                  # position (pos) and displacement (disp)
    v.disp ←(0,0);
    for u in V:
      δ←v.pos - u.pos;     # δ is the difference vector
      if(u ≠ v):
        if(u in cluster(v)):
          v.disp ← v.disp + $\frac{\delta}{\|\delta\|}$ *$f_r(\|\delta\|)$ ;
        else:
          v.disp ← v.disp + $\frac{\delta}{\|\delta\|}$ *enhance($f_r(\|\delta\|)$) ;
 # Compute the attractive forces on vertices
  for e in E:
     # u an v are the endpoint vertices of e
     δ←e.v.pos - e.u.pos;
     if(u in cluster(v)):
        e.v.disp ←e.v.disp - $\frac{\delta}{\|\delta\|}$ *$f_a(\|\delta\|)$;
        e.u.disp ←e.u.disp + $\frac{\delta}{\|\delta\|}$ *$f_a(\|\delta\|)$;
     else:
        e.v.disp ←e.v.disp - $\frac{\delta}{\|\delta\|}$ *reduce_out($f_a(\|\delta\|)$);
        e.u.disp ←e.u.disp + $\frac{\delta}{\|\delta\|}$ *reduce_out($f_a(\|\delta\|)$);
  for v in V:
       # case of non-related vertices of the same cluster
     for u in cluster(v) and (u,v) not in E:
        δ←e.v.pos - e.u.pos;
          e.v.disp ←e.v.disp - $\frac{\delta}{2*\|\delta\|}$ *reduce_in($f_a(\|\delta\|)$);
          e.u.disp ←e.u.disp + $\frac{\delta}{2*\|\delta\|}$ *reduce_in($f_a(\|\delta\|)$);
 # Limit the max displacement using tenperature
 # and keep them inside the frame
  for v in V:
    v.pos ←v.pos+$\frac{v.disp}{\|v.disp\|}$ *min(v.disp, t);
    v.pos.x ←min(W/2, max(-W/2, v.pos.x));
    v.pos.y ←min(L/2, max(-L/2, v.pos.y));
         # Reduce the temperature
  t = cool(t);
```

Fig. 5. Algorithm 2 : our algorithm for self-defined areas, based on [5].

3) Clustered-graph drawing : the case of predefined cluster areas

In this section, we examine the case of user pre-defined areas: each area is associated to a graph cluster. We believe that this is an important issue and to the best of our knowledge, it has not been addressed as a force placement problem in previous works. In our model, the visualization area associated with each cluster is a convex polygon. Nodes of clusters will be placed according to inter-cluster forces and extra-cluster forces. Two types of forces –attractive and repulsive- are defined the same way as in the previous case and the concept of optimal distance is personalized to each cluster. In our model, we distinguish two types of distance: one between vertices belonging to two different clusters –optDist. The other is relevant to vertices of a same cluster -optDistCluster. We hypothesize that the optimal distance between vertices of a given cluster must be smaller than the global optimal distance. We argue that the size the area occupied by a cluster is obviously smaller than the one of the total drawing area. In addition, the density of a cluster is generally higher than the whole graph density. For the reason, we define the ratio between optDist and optDistCluster as larger than 1.

The idea behind the use of two different optimal distances is to preserve a "satisfactory" display inside the area related to one given cluster. In fact, we consider that the layout of a cluster is "satisfactory" if it almost resembles the layout of that cluster when it is considered separately from the rest of the graph nodes while it still shows the influence of those nodes. When having optDistCluster < optDist, the influence of internal-nodes are considered as more important than the one of external-nodes. Note that we could have proposed a more personalized optimal distance computation so that each cluster C_i would have its own optimal distance $K_i = \sqrt{\frac{Surface\ C_i}{|C_i|}}$. However, this latter idea is not optimal as the distances between nodes in different clusters cannot be easily compared. Instead, we use only two different values one for the global graph and one for the clusters. We use

$$optDist = K_G = \sqrt{\frac{Surface}{|V|}}$$

and

$$optDistCluster = K_C = \frac{1}{|P|}\sum_{P}\sqrt{\frac{Surface\ C_i}{|C_i|}}$$

K_C is obtained averaging K_i

We also offer another way to differentiate the influence of intra and extra-cluster vertices by enhancing intra-cluster interactions and by reducing extra-cluster ones. We achieve this goal using the functions: $reduce\_in()$, $reduce\_out()$ and $enhance()$.

Moreover, to actively keep nodes of clusters inside their cluster area, we define repulsive forces between nodes and the border of the cluster area they belong to. We call these forces *frontier forces* (f_f). The strength of such repulsive forces is formulated according to the distance between the nodes and the border of the clusters they belong to. It must be sufficiently strong when the distance to the border is small. In order to achieve this goal, we consider that the nodes are charged particles and that the borders of a visualization area are also electrically charged. The induced repulsive forces follow the electrical force principles. There is no need that this kind of force operates beyond a considered strip along the cluster border. In our model, the maximum displacement possible for a vertex is governed by the temperature and the cooling function. The width of the strip we consider is then two times this maximum displacement, that is to say $(2 * t)$.

Still, because of the discrete placement of nodes, in some special cases, the strength of cumulative forces exercised on a given node may push it outside the cluster frontier. We use the same strategy as in Fruchterman and Reingold model to return the vertex to its associated area.

In this context of fixed and none porous areas, it is no more necessary to add the "invisible" edges to stick together the vertices of a given cluster. However, we maintained them in the proposed solution to better handle the incidence of extra-cluster influences.

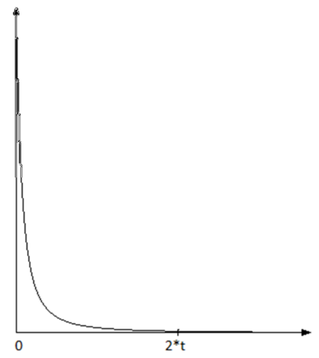

Fig. 6. The strength of frontier forces over the distance to the frontier. This strength rapidly decrease as the distance separating a vertex from its cluster-frontier reach 2*t

We also consider that this force has to be rapidly lowered to reach and stick to zero as the vertex reaches a distance of 2 times the value of the threshold.

Let f_f be the frontier force and d the distance between a node and the border of its cluster. f_f is defined as:

$$f_f(d) = \begin{cases} 0 \; If \; d > 2\,th \\ \dfrac{k_C}{d^2} \end{cases}$$

where $th = cool(t)$ is the maximum possible displacement for any vertex,

k_c is the optimal distance in cluster context.

For simplification reasons, the frontier force exercised on a given node is determined by its nearest frontier point.

The Fig. 7 shows the principle of our model. The distance between two vertices belonging to two different clusters is split in three parts: d_1, d_2 and d_3. d_2 which is the distance between the two clusters is handled differently from the two other distances. We use a function to alter this distance (lower it) in order to manage the attractive force that depends on. In some sort, we curve the space between clusters zones so that we can handle the strength of inter-clusters forces. This is suitable when the clusters are too far from each other for example in geographic visualization. Note that this strength reduction is also the reason of using a multilevel optimal distance.

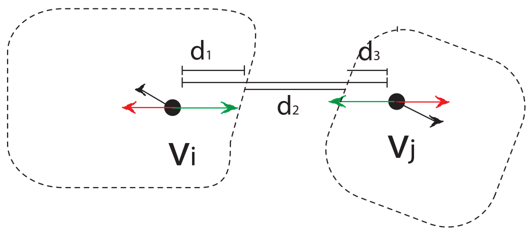

Fig. 7. The sets of exercised attractive (green) and repulsive (red) and frontier (black)

Fig. 8 presents the algorithm we propose.

```
surface ← W * L;  # W and L are the width and length of the frame
G ← (V,E,P);   # The vertices are assigned random initial positions
                    # inside the area they belong to
                    # The boundaries of eache element of are
                    # convex polygones

$K_G = \sqrt{\frac{Surface}{|V|}}$

$K_C = \frac{1}{|P|}\sum_{P}\sqrt{\frac{Surface\ C_i}{|C_i|}}$

function $f_a(x,k)$:
    return $\left(\frac{x^2}{k}\right)$

function $f_r(x,k)$:
    return $\left(\frac{k^2}{x}\right)$

function $f_{front}(x)$:
    if $(x < 2 * t)$:
        return $\left(\frac{K_C{}^2}{x^2}\right)$
    else:
        return(0)

$t = t_0$;                #initialise temperature t

function $\Delta(u, v)$:
    if(u in cluster(v)):
        $\delta$ ← v.pos - u.pos;      # $\delta$ is the difference vector
    else:
        $\delta$ ← v.pos - front(v).pos +
            curve(front(v).pos-front(u).pos) +
            front(u).pos)-u.pos;
    return($\delta$);

function to_front(v):
        front(v) ← nesrest_point(v,cluster(v).boundary)
        $\delta$ ← v.pos - v.front.pos;      # $\delta$ is the difference vector
    return($\delta$);

for i ← 0 to nbIterations:
# Compute the repulsive forces on vertices
  for v in V:
```

```
            # Each vertex is associated two 2D vectors:
            # position (pos) and displacement (disp)
    v.disp ←(0,0);
    $\delta \leftarrow to\_front(v)$:
    v.disp ← v.disp + $\frac{\delta}{|\delta|}$ *$f_{front}(|\delta|)$ ;
    for u in V:
        if($u \neq v$):
        $\delta$←Δ(u,v)     # $\delta$ is the difference vector
        if(u in cluster(v)):
            v.disp ← v.disp + $\frac{\delta}{|\delta|}$ *$f_r(|\delta|, K_C)$ ;
        else:
            v.disp ← v.disp + $\frac{\delta}{|\delta|}$ *enhance($f_r(|\delta|, K_G)$) ;
# Compute the attractive forces on vertices
for e in E:
    # u an v are the endpoint vertices of e
    δ←Δ(e.u,e.v)
    if(u in cluster(v)):
        e.v.disp ←e.v.disp - $\frac{\delta}{|\delta|}$ *$f_a(|\delta|, K_C)$;
        e.u.disp ←e.u.disp + $\frac{\delta}{|\delta|}$ *$f_a(\delta, K_C)$;
    else:
        e.v.disp ←e.v.disp - $\frac{\delta}{|\delta|}$ *reduce_out($f_a(|\delta|, K_G)$);
        e.u.disp ←e.u.disp + $\frac{\delta}{|\delta|}$ *reduce_out($f_a(|\delta|, K_G)$);
for v in V and (u,v) not in E: :
        # case of non-related vertices of the same cluster
    for u in cluster(v):
        δ←Δ(e.u,e.v)
        e.v.disp ←e.v.disp -
            $\frac{\delta}{2*|\delta|}$ * reduce_in($f_a(|\delta|, K_C)$);
        e.u.disp ←e.u.disp +
            $\frac{\delta}{2*|\delta|}$ *reduce_in($f_a(|\delta|, K_C)$);
# Limit the max displacement using tenperature
# and keep them inside the frame
for v in V:
    w←$cluster(v).w$;
    l←$cluster(v).l$;
    n←$cluster(v).n$;
    $X \leftarrow cluster(v).pos\_x$
    $Y \leftarrow cluster(v).pos\_x$
    v.pos ←v.pos+$\frac{v.disp}{|v.disp|}$ * min(v.disp , t);
    v.pos.x ←X+min(w/2, max(-w/2, v.pos.x-X));
    v.pos.y ←Y+min(l/2, max(-l/2, v.pos.y-Y));
        # Reduce the temperature
t = cool(t);
```

Fig. 8. Algorithm 3: Our algorithm in the case of pre-defined areas based on [5]

IV. Evaluation and Results

In this section, we provide and discuss three examples to show the practicability of our models.

A. *First example: a clustered collaboration network*

This first example presents a collaboration network where the vertices represent researchers that have been associated to eight different clusters. How these clusters have been obtained is out of the scope. First we perform a drawing of the related graph using Fruchterman and Reingold algorithm (Fig. 3). As we can see, the resulting drawing is quite pleasant but it does not clearly show the clustered structure of the graph in eight clusters. The two next figures (Fig. 10 and Fig. 11) present the resulting drawing for the same graph when using our methods. Fig. 10 shows the results when the self-defined area approach is applied (Algorithm 2 in Fig 5) while Fig. 11 is related to predefined-area approach (Algorithm 3 Fig. 8). As we can see the cluster structure is clearly presented and the global aesthetic of the two graphs is still quite pleasant.

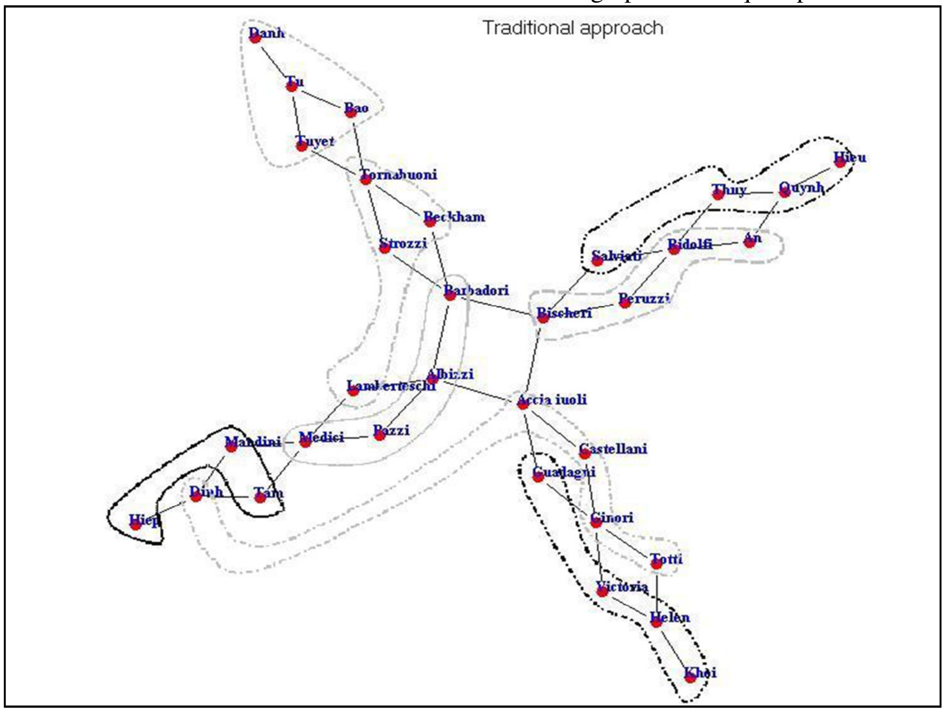

Fig. 9. Graph drawing based on algorithm 1

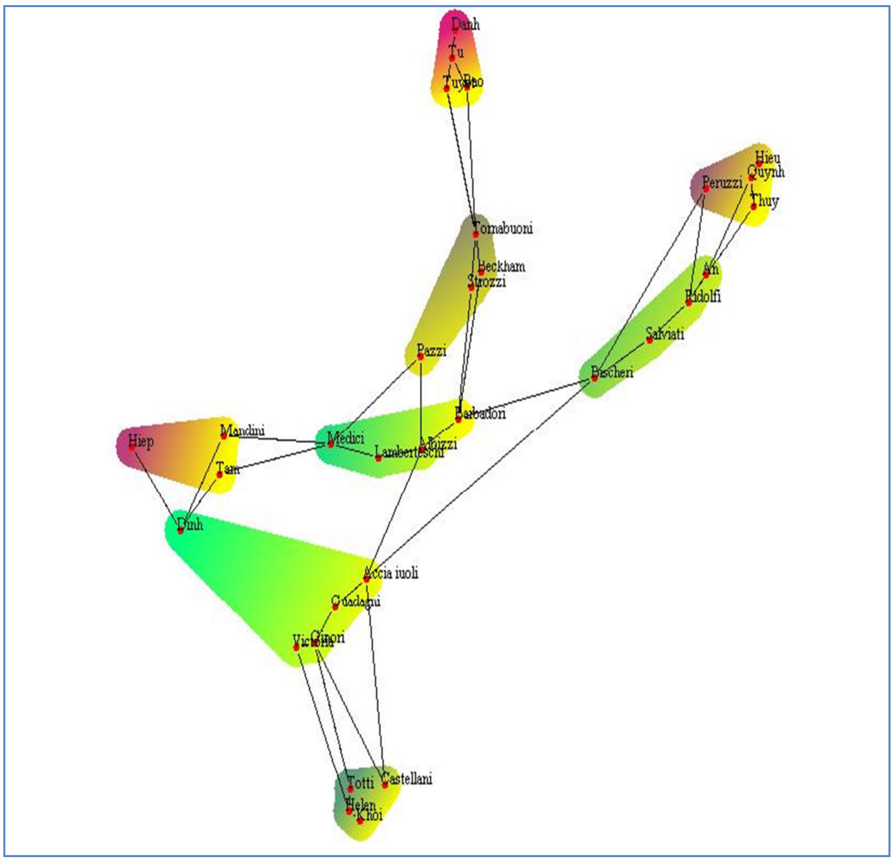

Fig. 10. Graph drawing based on algorithm 2 – Self-defined drawing areas

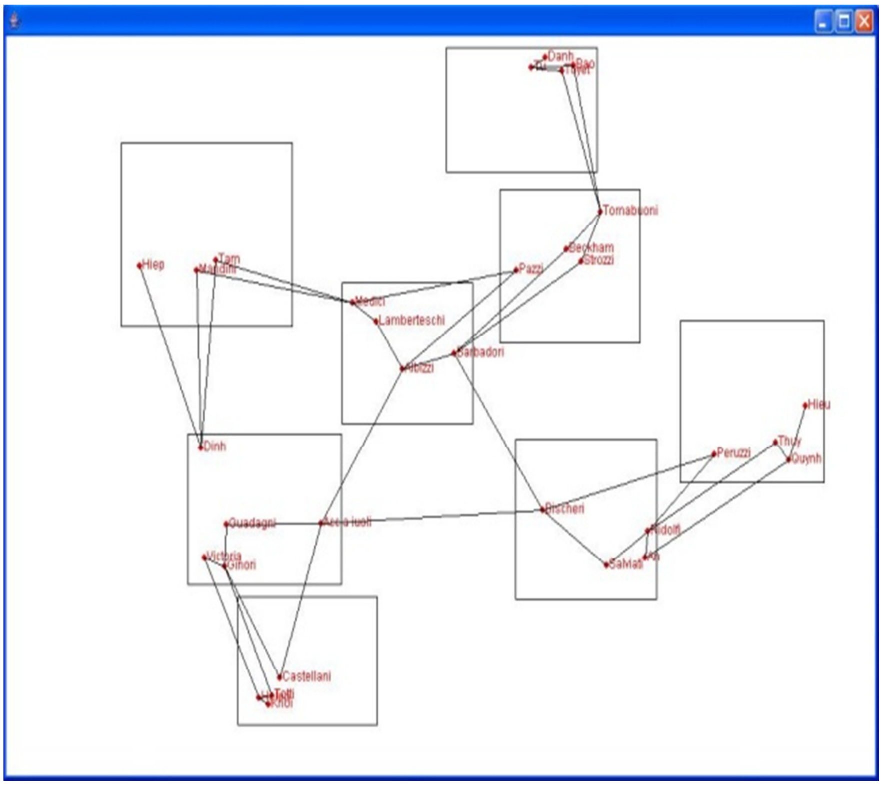

Fig. 11. Graph drawing based on algorithm 3 – Pre-defined drawing areas

B. Second example: a citation network

This second example is intended to show that the predefined area drawing we suggest can be used to help appreciating the correlation between the clustering and the binary relationships inherent to the graph structure. The graph we present in this section represents the citations network in the context of graph drawing [9]. The graph consists in 311 vertices and 647 edges. Each vertex represents a paper published in the *International Symposium on Graph Drawing* from 1994 to 2000.

We perform an authority analysis [29][30][31] and obtain an authority score for each vertex. Table 1 shows the top authority scores. This authority score is used to generate eight clusters. Each cluster is composed of papers that have similar authority scores. Actually, the clustering results from the fragmentation of the authority scores into eight segments of the same amplitude. Each cluster is then associated to a prefixed area where the areas are organized as horizontal strips or layers of the same width and positioned on the top of each other's. Clusters on the top of the drawing correspond to nodes that have best authority scores.

TABLE I. AUTHORITY SCORES OF GD PAPERS 1994-2000

Paper	***Paper title***
GD 96, 139 Eades, ...	Two Algorithms for Three Dimensional Orthogonal Graph Drawing.
GD 94, 1 Cohen, ...	Three-Dimensional Graph Drawing
GD 95, 254 Foessmeier, ...	Drawing High Degree Graphs with Low Bend Numbers
GD 94, 286 Garg, ...	On the Compuational Complexity of Upward and Rectilinear Planarity Testing
GD 95, 419 Papakostas, ...	Issues in Interactive Orthogonal Graph Drawing
GD 95, 99 Bruss, ...	Fast Interactive 3-D Graph Visualization
GD 94, 388 Frick, ...	A Fast Adaptive Layout Algorithm for Undirected Graphs
GD 95, 8 Alt, ...	Universal 3-Dimensional Visibility Representations for Graphs
GD 97, 52 Papakostas, ...	Incremental Orthogonal Graph Drawing in Three Dimensions
GD 95, 234 Fekete, ...	New Results on a Visibility Representation of Graphs in 3D

Top authority papers from the International Symposium on Graph Drawing from 1994 to 2000

The resulting drawing (Fig. 12) confirms that citations mainly go from nodes in lower layers to nodes in higher layers. It also shows that papers that obtain high authority scores do not cite each other's.

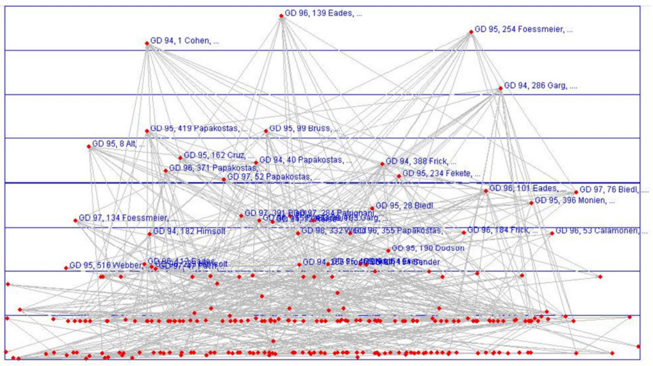

Fig. 12. Drawing of the citation network where vertices are grouped according to their authority score and placed into corresponding layers

C. third example: a short comparison

This third example is intended to compare the results of our method to those of a divide-and-conquer approach.

[10] presents a divide-and-conquer approach to generate visually structured layouts for clustered graphs. The proposed algorithm partitions a graph into sub graphs and composes them to form the resulting layout. To position each vertex, the divide-and-conquer approach uses a composite force that includes intra- forces, progressively increasing inter-cluster forces and gradually decreasing meta forces. The authors propose an example and draw it as follows:

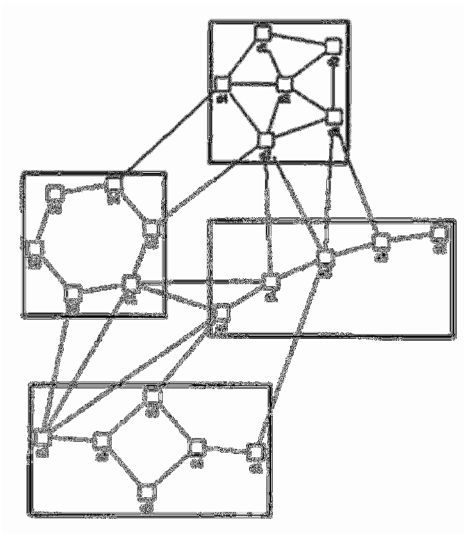

Fig. 13. The grah example as it have been drawn by the divide-and-conquer methode in [10]

Fig. 14 shows the drawing our method outputs.

The two drawings presented in Fig. 13 and Fig.14 show that while the two methods successfully point out the cluster structure, our method better acknowledges the global structure of the graph.

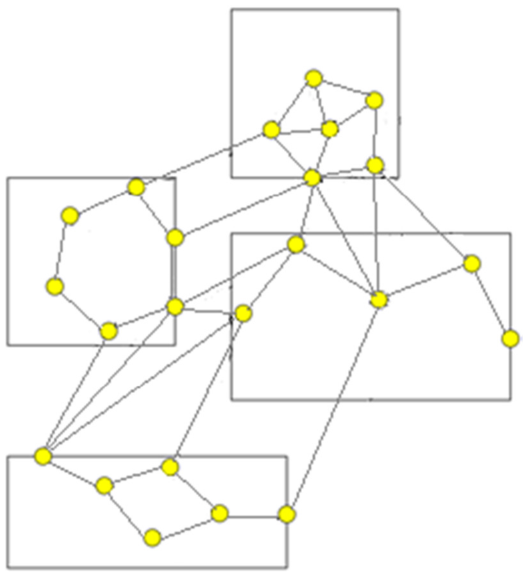

Fig. 14. The grah example given in [10] as it have been drawn our prefixed areas related method

V. CONCLUSION

In this paper, we have introduced a new energy-based model for clustered graph drawing. Two aspects of the drawing corresponding to self-defined and pre-defined cluster related drawing areas have been addressed; a distinct solution has been proposed for each one of them.

The results we obtained show the potential of our model in dealing with metric values such as authority scores. This work can also help dealing with geographic graph visualization when graphs vertices are grouped according to some of their geographic properties such as the authors' countries or locations in scientific citation networks. Such graph drawing approach will help to understand the implications –if any- of geographic characteristics over the vertices relationships.

REFERENCES

[1] P. J. Carrington, J. Scott, and S. Wasserman (Eds.), “Models and methods in social network analysis,” Cambridge university press, 2005.

[2] E. R. Gansner, and S. C. North, “An open graph visualization system and its applications to software engineering,” Software Practice and Experience, 30(11), pp. 1203-1233, 2000.

[3] R. F. i Cancho, C. Janssen, and R. V. Solé, “Topology of technology graphs: Small world patterns in electronic circuits,” Physical Review E, 64(4), p. 046119, 2001.

[4] S. B. Navathe, and M. Ra, “Vertical partitioning for database design: a graphical algorithm,” In ACM SIGMOD Record, Vol. 18, No. 2, pp. 440-450, 1989.

[5] T. M. J. Fruchterman, E. M. Reingold, “Graph drawing by force-directed placement,” Software. Pract. Exper. 21(11), pp. 1129–1164, 1991.

[6] T. Kamada, S. Kawai, “An algorithm for drawing general undirected graphs,” Information Processing Letters, 31(1), pp. 7-15, 1989.

[7] S. Hachul and M. Jünger, “An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs,” Pro. of the Graph Drawing 2005, vol. 3843, Lecture Notes in Computer Science, Springer-Verlag; pp. 235-250, Limerick, Ireland, 2006.

[8] U. Brandes, “Layout of Graph Visualizations,” PhD thesis, University of Konstanz, 1999. (http://www. ub. uni-konstanz/kops/volltexte/1999/255).

[9] T. Biedl and F. J. Brandenburg, “Graph-Drawing contest report,” Proc. of the Graph drawing 2001, Lecture Notes in Computer Science, Springer-Verlag, pp. 388-403, 2001.

[10] M.-A. Storey and H. Müller, “Graph layout adjustment strategies,” in Graph Drawing, vol. 1027, F. Brandenburg, Ed. Springer Berlin Heidelberg, 1996, pp. 487–499.

[11] W. T. Tutte, “How to draw a graph,” Proc. London Math. Society, 13(52):743–768, 1963.

[12] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, “Graph Drawing: Algorithms for the Visualization of Graphs,” Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998

[13] S. G. Kobourov, “Force-Directed Drawing Algorithms,” In Roberto Tamassia (editor), Handbook of Graph Drawing and Visualization, p. 383-408, CRC Press, 2013 (http://arxiv.org/pdf/1201.3011.pdf)

[14] C. Görg, P. Birke, M. Pohl, and S. Diehl, “Dynamic graph drawing of sequences of orthogonal and hierarchical graphs,” In Graph Drawing, pp. 228-238, 2005.

[15] U. Dogrusöz, B. Madden and P. Madden, “Circular Layout in the Graph Layout Toolkit, Proc. GD '96,” LNCS 1190, Springer-Verlag, pp. 92-100, 1997.

[16] G. Melancon and I. Herman, “Circular drawings of rooted trees,” Technical Report: INS-R9817, Centre for Mathematics and Computer Science.

[17] P. Eades, "A heuristic for graph drawing," Congressus Numerantium 42 (11): 149–160, 1984.

[18] P. Gajer and S. G. Kobourov, “GRIP: Graph Drawing with Intelligent Placement,” J. Graph Algorithms Appl., vol. 6, no. 3, pp. 203–224, 2002.

[19] D. Harel and Y. Koren, “A fast multi-scale method for drawing large graphs,” Journal of Graph Algorithms and Applications , 6(3):179–2002, 2002.

[20] C. Walshaw, “A Multilevel Algorithm for Force-Directed Graph Drawing,” in Graph Drawing, vol. 1984, J. Marks, Ed. Springer Berlin Heidelberg, 2001, pp. 171–182.

[21] J-H. Chuang, C-C. Lin, H-C. Yen, “Drawing Graphs with Nonuniform Nodes Using Potential Fields,” In Liotta, Giuseppe, Eds. Proceedings Graph Drawing, pages pp. 460- 465, Perugia, 2004

[22] Y. Frishman, A. Tal, “Dynamic Drawing of Clustered Graphs,” Proceedings of IEEE Symposium on Information Visualization 2004, pp. 191-198, 2004

[23] T. Dwyer, Y. Koren, and K. Marriott. IPSep-CoLa: An incremental procedure for separation constraint layout of graphs. Visualization and Computer Graphics, IEEE Transactions on, 12(5), pp. 821-828, 2006.

[24] A. Noack, “An Energy Model for Visual Graph Clustering,” In Liotta, Giuseppe, Eds. Proceedings Graph Drawing, pages pp. 425-436, Perugia, 2004.

[25] Q.D. Truong, T.Dkaki, and P.J. Charrel, “An energy model for the drawing of clustered graphs,” In 5th VSST, Marrakech, 2007

[26] P. Eades, M.L. Huang, “Navigating Clustered Graphs Using Force-Directed Methods,” J. Graph Algorithms Appl., 2000: 157-181, 2000

[27] M. Balzer and O. Deussen, “Level-of-detail visualization of clustered graph layouts,” in APVIS, 2007, pp. 133–140.

[28] P. Rodgers. Graph drawing techniques for geographic visualization. 2004.

[29] T. Dkaki, J. Mothe, Q.D. Truong, “Passage Retrieval Using Graph Vertices Comparison,” Third International IEEE '07Conference SITIS, Shanghai 2007.

[30] J. Kleinberg, "Authoritative sources in a hyperlinked environment," Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, PA, 1998, 668–677

[31] V. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren, “A measure of similarity between graph vertices,” cs/0407061, Jul. 2004.