Graph Methods for Social Network Analysis

Quoc Dinh Truong[1(✉)], Quoc Bao Truong[2], and Taoufiq Dkaki[3]

[1] College of Information and Communication Technology, Can Tho University, Campus 2, Can Tho University, 3/2 street, Ninh Kieu district, Can Tho City, Vietnam
tqdinh@ctu.edu.vn

[2] College of Engineering Technology, Can Tho University, Campus 2, Can Tho University, 3/2 street, Ninh Kieu district, Can Tho City, Vietnam
tqbao@ctu.edu.vn

[3] Institut de Recherche en Informatique de Toulouse, Université de Toulouse, Toulouse, France
dkaki@irit.fr

Abstract. Social network is a structure in which nodes are a set of social actors that are connected together by different types of relationships. Because of the complexity of the actors and the relationships between actors, social networks are always represented by weighted, labeled and directed graph. Social network analysis (NSA) is a set of techniques for determining and measuring the magnitude of the pressure. Social network analysis is focused also on visualization techniques for exploring the networks structure. It has gained a significant following in many fields of applications. It has been used to examine how the problems have been solved, how organizations interact with others, to understand the role of an individual in an organization… In this paper, we focus on two methods: 1- graphs visualization; 2- network analysis based on graph vertices comparison.

Keywords: Social network · Graph · Graph drawing · Graph comparison

1 Introduction

In reality, social network is a common word but there are many definitions of it. Today, people think that social networks are Facebook, Twitter, Google+ or any website with the keyword "social network" in its description such as: www.ResearchGate.net (social networking site for scientists and researchers), Zing Me (Vietnamese social networking and entertainment)… In general, social network is known as a social structure in which actors (individuals or organizations), called nodes, are connected by a number of types of relationships such as: friendship, common interest, relationships of beliefs, knowledge or prestige [1]. The main characteristics of a social network are:

- There exist a collection of entities that participate in the network. Typically, these entities are people.
- There exist at least one type of relationships between entities. The relationship can be a type of "all-or-nothing" or has a degree.

P.C. Vinh et al. (Eds.): ICTCC 2016, LNICST 168, pp. 276–286, 2016.
DOI: 10.1007/978-3-319-46909-6_25

- There exist an assumption of nonrandomness. It means that if A has relationships with both B and C, then there is a high probability that B and C are related.

Social network analysis is a field of study that attempt to understand relationships between entities in a network based on the assumption about the "importance" of relationship between entities. There are many definitions of social network analysis and the most used one is: "*Social network analysis has emerged as a set of methods for the analysis of social structures, methods which are specifically geared towards an investigation of the relational aspects of these structures. The use of these methods, therefore, depends on the availability of relational rather than attribute data* [2]". Social network analysis is focused also on visualization techniques for exploring the networks structure. Graph structure is often used to represent social networks and their size becomes increasingly large as the progression of the means for data gathering and storage steadily strengthens. This call for new methods in graph visualization and analysis for dealing with the problem of large graphs. In this paper, we present two methods: 1- method for clustered graphs drawing; 2- graph vertices comparison method for network analysis. The rest of paper is organized as follow. Section 2 introduces notion of clustered graph and presents a drawing method for this type of graphs. In Sect. 3, we present our proposed method for graph vertices comparison. Section 4 discuss the performance of our proposed methods through several application examples. In final section we derive conclusions and gives some suggestions.

2 Clustered Graph Drawing

2.1 Clustered Graph Structure

A graph is a couple (V, E), where V is an arbitrary set of vertices and E is a finite set of edges, that is, E consists of some couples (x, y) where x, y $\in$ V. A clustered graph is a triple (V, E, P) where V is a finite set of vertices, E $\subseteq$ V x V a finite set of edges and P is a partition over V. The Number of elements in P corresponds to the number of clusters in graph. The following figure illustrates structure of clustered graphs –the graph in the figure contains three clusters (Fig. 1).

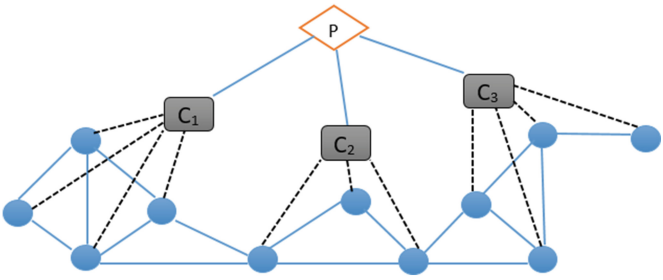

Fig. 1. Structure of a clustered graph associated to a cluster dendogram

2.2 Model

The objective of automatic graph drawing is the development of algorithms for creating nice and readable layout of graphs. In the two last decades, there are enormous works that have been undertaken in the field of graph drawing in order to generate intelligible graph layouts. In the case of basic graph structure, several representative approaches are described in [3–5].

When considering the case of clustered graph drawing, only a very few works can be cited. Among these works we can list some major works such as [6–8]. Ho [6] introduced a method for drawing clustered graph in space of three dimensions by using "divide and conquer" approach. He proposed to draw each cluster in a 2D plane and the 3D drawing of clustered graph is produced by combining these flat drawings. Chuang [7] considers the problem of clustered graph drawing as a problem of drawing graphs with nonuniform nodes represented as charged polygons/polyhedron exposed to both spring forces and magnetic forces induced by the potential fields generated by the other nodes in the graph. Eades [8] propose adding an extra vertex, called super node, to each cluster and using different types of spring force. In this work, the drawing zones of clusters are predefined by the users and can be modified by the proposed model.

We believe that clustered graph drawing rules differ to that of basic graph drawing. These heuristic rules that abstracts our approach for clustered graphs drawing. We present some visualization constraints for drawing clustered graph in priority order:

- The drawing zone of each cluster must be convex.
- The drawing zones associated to the clusters must not overlap.
- Edge crossings are reduced as much as possible within a cluster.

We distinguish also two problems of drawing clustered graph: self-generated cluster areas and predefined cluster areas. By considering the set of heuristic rules we list above, the two first rules are considered such as preconditions for the cluster areas in the case of predefined cluster areas. Our method is based on force directed placement method proposed in [4]. Like this method, our proposed method iterates two steps: computations of the forces among vertices on the one hand and the positioning adjustment of each vertex and the computation of its new coordinates following the temperature model on the other hand. The displacement of a vertex at every iteration is limited to the maximum displacement value decreasing over time. This means that, as the layout becomes better the amount of adjustment becomes smaller.

The difference with the approach proposed in [4] is that we differ two types of spring forces and add additional attractive forces (internal-spring type) between non-connected nodes within the same cluster (invisible edges).

- Internal-spring: A spring force (repulsive and/or attractive) between a pair of nodes which are in the same clusters.
- External-spring: A spring force (repulsive and/or attractive) between a pair of nodes which are in different clusters.

The forces are defined based on the concept of an optimal distance between vertices that are not connected. Generally, this distance can be computed as follow:

$$optDist = \sqrt{\frac{frame\_size}{number\ of\ vertices}} \tag{1}$$

f_a, f_r are attractive and repulsive forces respectively which can be defined as follow:

$$f_a(d) = d^2/\text{optDist} \tag{2}$$

$$f_r(d) = -\text{optDist}^2/d \tag{3}$$

where d is the distance between the two vertices.

Since in our model, we distinguish two types of spring forces so two optimal distances are needed: one between vertices belonging to two different clusters –optDist- and the other relevant to vertices of a same cluster -optDistCluster. We believe that the optimal distance between vertices of a same cluster must be less than the global optimal distance because the size of a cluster occupied area must obviously be smaller than the size of the total drawing area (optDist = a *optDistCluster where a > 1).

Self-generated cluster areas. In this case, we propose adding invisible edges (additional attractive forces) between non-connected nodes within same cluster to keep them close together in the drawing. The forces generated by the invisible edges must then be lowered to prevent them from over affecting the internal layout of clusters. In other word, extra edges will be help to create separated cluster with relatively close nodes without over modifying the relative placement -of the cluster's nodes. To do that, the strength of additional attractive forces must be according to the proportion between the number of invisible edges and the number of real edges. The following figure shows our spring force model for this application case (Fig. 2).

Predefined cluster areas. In this case, our proposed model takes user predefined area of each cluster as parameter. So the process of adding "invisible" edges is not necessary. Two types of forces are defined in the same way as in the previous case. And

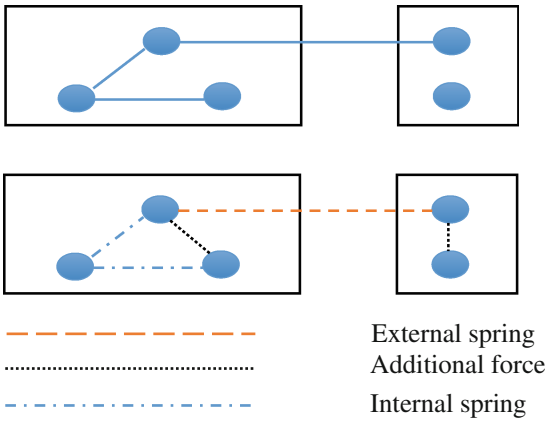

Fig. 2. Spring forces model for self-generated cluster areas

then, to keep nodes inside their cluster area, we add repulsive forces between nodes and the border of visualization area of cluster (we call them as frontier forces). If f_f is the frontier force, d the distance between node and the border and δ predefined constant equal to the maximum possible displacement of node then it can be defined as following

$$f_f(d) = \begin{cases} \infty \ \textit{if} \ d \leq \delta \\ \frac{optDistCluster}{d^3} \end{cases} \quad (4)$$

The following figure shows the spring force model in this case (Fig. 3).

Algorithm
Input: Clustered graph CG (V, E, P); Visualization area of each cluster (optional)
Output: Drawing of clustered graph CG

```
→ self-generated cluster area
random placement of nodes
→ predefined cluster area
placement of nodes according to their clusters areas
while temperature > 1 begin
    for v ∈ V do begin
    → predefined cluster area
         calculate frontier force on v
         for u ≠ v in V do begin
           calculate forces on v and u
         end
    end
    for (u, v) ∈ E do begin
         calculate force on u and v
    end
    → self-generated cluster area
    for (u, v) ∉ E and u, v in the same cluster do
    begin
         calculate additional attractive force on u, v
    end
    for v ∈ V do begin
         accumulate forces on v
         displacement of v in function of the
         temperature
    end
    decrease the temperature
end
```

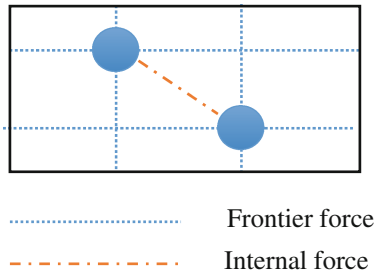

Fig. 3. Spring forces model for predefined cluster areas

3 Graph Vertices Comparison

We proposed in [9] a method for graph vertices comparison. The starting point of our method is the one proposed in [10]. This method is applied on two graphs: one to be analyzed (the target graph) and the other serving as the model (the source graph). By using this method we can identify which vertices in the source graph are the most similar with the given vertex in the target graph (Fig. 4).

The main idea of our method is that the similarity between two vertices i and j respectively from target graph and source graph is computed by examining the similarity scores between their related vertices (vertices pointing to i or j and vertices pointed by i or j). In other word, the similarity score S_{ij} between vertex i of target graph and vertex j of source graph can be updated by following equation:

$$S_{ij} = \sum_{r:(r,i)\in E_B, t:(t,j)\in E_A} S_{rt} + \sum_{r:(i,r)\in E_B, t:(j,t)\in E_A} S_{rt} \tag{5}$$

However (5) does not take into account the notion of similarity inheritance. We modified (5) to comply with the similarity inheritance principle. In our work, we consider the similarity inheritance as a "flooding" similarity proposed in [11]. So the notion of graph closure is used to express the similarity inheritance. To do that, adjacency matrices of target and source graphs are formulated as follows:

$$A \leftarrow A + \sum_{n=2}^{\infty} f_A(n) \frac{A^n}{A^n} \qquad \mathrm{B} \leftarrow B + \sum_{n=2}^{\infty} f_B(n) \frac{B^n}{B^n} \tag{6}$$

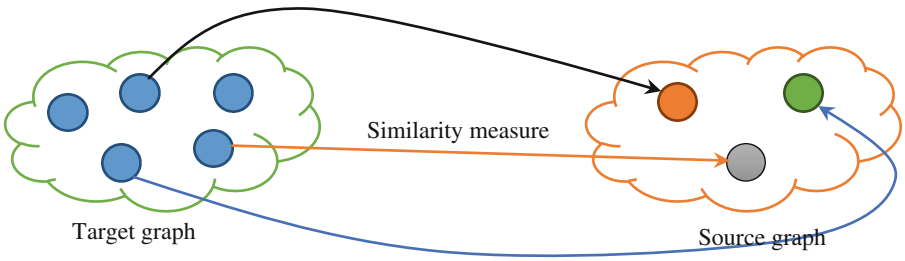

Fig. 4. Graph vertices comparison

where $f_A(n)$ and $f_B(n)$ are used to translate the idea that the influence of 'generations' exponentially decreases with path depth.

In the case of self-comparison where target and source graphs are the same, the following condition must be satisfied: $\forall(i, j)$, $s(i, i) = s(j, j) \geq s(i, j)$. So we normalized similarity matrix S_{AB} by dividing each value $S_{AB}(i, j)$ by the product of self-similarity $S_{AA}(i, i)$ of vertex i in graph A and $S_{BB}(j, j)$ of vertex j in graph B. The final proposed method for graph vertices comparison is described in the algorithm below.

$S_0 \leftarrow 1 \qquad k \leftarrow 0$

$$A \leftarrow A + \sum_{n=2}^{\infty} f_2(n) g_2\left(\frac{A^n}{\|A^n\|}\right) \quad ; \quad B \leftarrow B + \sum_{n=2}^{\infty} f_1(n) g_1\left(\frac{B^n}{\|B^n\|}\right)$$

Repeat

$$S_{AA_{k+1}} \leftarrow \frac{AS_{AA_k}A^T + A^T S_{AA_k}A}{\left\|AS_{AA_k}A^T + A^T S_{AA_k}A\right\|_F} \quad S_{BB_{k+1}} \leftarrow \frac{BS_{BB_k}B^T + B^T S_{BB_k}B}{\left\|BS_{BB_k}B^T + B^T S_{BB_k}B\right\|_F} \quad S_{AB_{k+1}} \leftarrow \frac{BS_{AB_k}A^T + B^T S_{AB_k}A}{\left\|BS_{AB_k}A^T + B^T S_{AB_k}A\right\|_F}$$

$k \leftarrow k + 1$

Until convergence is achieved for k even

$$S_{AB} \leftarrow \bullet \frac{S_{AB} \bullet * S_{AB}}{diag(S_{AA}) \bullet * diag(S_{BB})^T}$$

Output $S_k(kiseven)$ *as similarity matrix*

4 Results and Discussion

For verifying the performance of our proposed methods we use two social networks. This first network presents a collaboration network where vertices represent researchers that have been associated to eight clusters. First we perform a drawing of this network using the basic graph drawing method [4] (Fig. 5). As we can see, the resulting

Fig. 5. Graph drawing based on the algorithm described in [4]

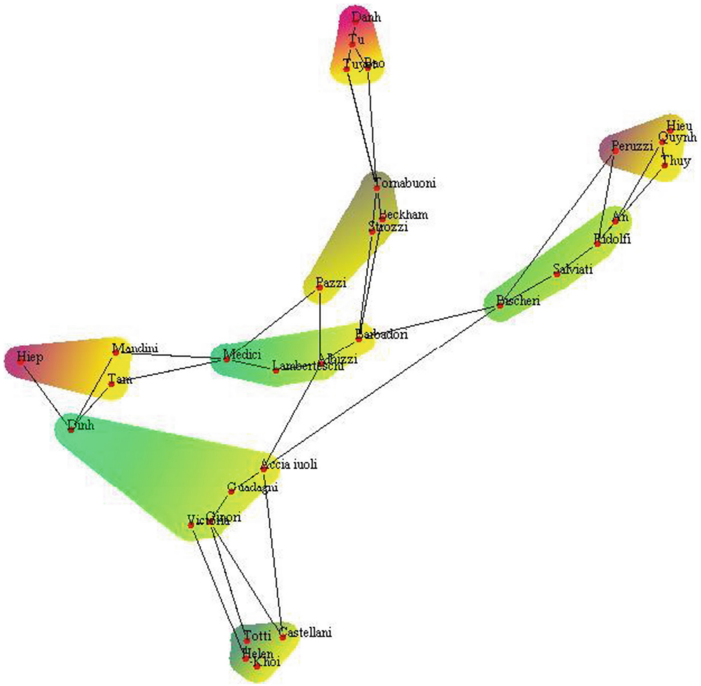

Fig. 6. Graph drawing based on our algorithm – the case of self-generated cluster area

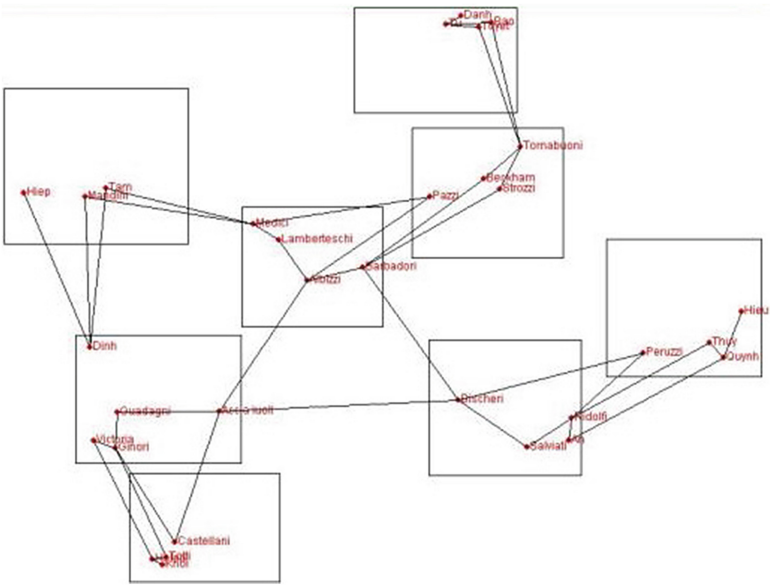

Fig. 7. Graph drawing based on our algorithm – the case of predefined cluster area

Table 1. Top authority papers from the international symposium on graph drawing (1994 to 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 Computational 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

drawing does not clearly show the cluster structure of the network. The two next figures (Figs. 6 and 7) present the resulting drawing for the same network by using our methods. As we can see the cluster structure is clearly presented.

The second example we present here is the citations network in the context of graph drawing [12]. The network consists of 311 vertices and 647 edges. Each vertex represents a paper published in the International Symposium on Graph Drawing from 1994 to 2000. We use our proposed method for graph vertices comparison to perform a

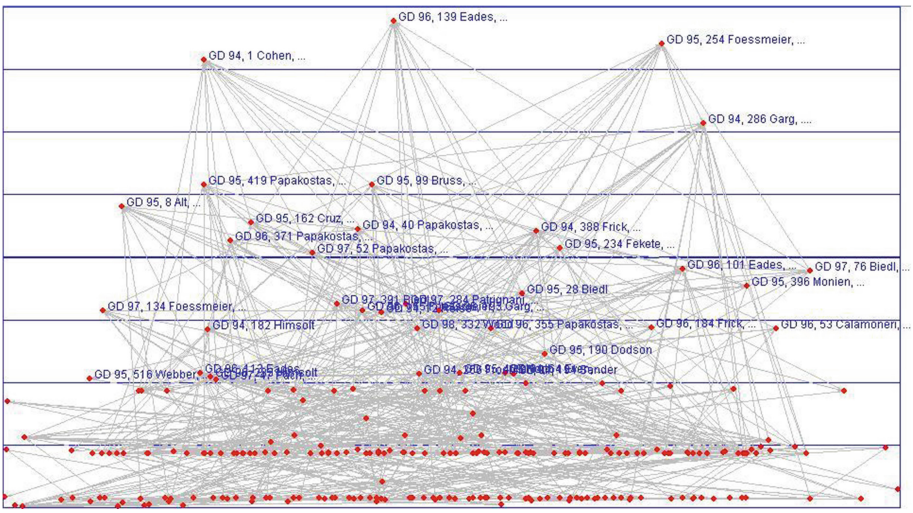

Fig. 8. Drawing of citation network

hub-authority analysis. In this example, we only care the authority of each vertex (Table 1). The authority scores are used to generate eight clusters. Each cluster is composed of papers that have similar authority scores. And then each cluster is associated to a prefixed rectangle area. Clusters on the top of the drawing correspond to papers that have best authority scores (Fig. 8).

5 Conclusion

In this paper we have proposed couple of methods for social network analysis: one for drawing clustered graph and another for graph vertices comparison. Our model for drawing clustered graph differs from previous works from two aspects: the forces in action and the nature of the clusters that can be handled. Indeed, the authors of these works define clusters as sets of vertices with many internal edges and few outside edges while clusters in our work are "freely" defined. We also presented and discussed a novel model for computing similarity scores between vertices of two graphs by extending methods previously submitted in [10, 11]. This method can be applied in the field of social network analysis in the way that we can identify the similar actors of two networks.

In the future work, we will extend the model for clustered-graph drawing to deal with geographic constraints when graphs vertices are clustered according to some of their geographic properties such as the authors' countries or locations in scientific citation networks.

References

1. Wasserman, S., Faust, K.: Social network analysis in the social and behavioral sciences. In: Social Network Analysis: Methods and Applications, pp. 1–27. Cambridge University Press (1994). ISBN 9780521387071
2. Scott, J.: Social Network Analysis. Sage, Newbury Park (1992)
3. Gajer, P., Kobourov, S.G.: GRIP: Graph dRawing with Intelligent Placement. In Marks, J. (ed.) Proceedings of the Graph Drawing, pp. 222–228. Colonial Wiliamsburg (2001)
4. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Softw. Pract. Exp. **21**(11), 1129–1164 (1991)
5. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Inf. Process. Lett. **31**(1), 7–15 (1989)
6. Ho, J., Hong, S.-H., Gronemann, M., Jünger, M.: Drawing Clustered Graphs as Topographic Maps. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS, vol. 7704, pp. 426–438. Springer, Heidelberg (2013)
7. Chuang, J.-H., Lin, C.-C., Yen, H.-C.: Drawing graphs with nonuniform nodes using potential fields. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 460–465. Springer, Heidelberg (2004)
8. Eades, P., Huang, M.L.: Navigating clustered graphs using force-directed methods. J. Graph Algorithms Appl. **4**, 157–181 (2000)

9. Truong, Q.D., Dkaki, T., Mothe, J., Charrel, P-J.: GVC: a graph-based information retrieval model. In: Conférence francophone en Recherche d'Information et Applications (CORIA 2008), Trégastel (France), 12–14 Mar 2008, pp. 337–351. CNRS (2008)
10. Blondel, V.D., Gajardo, A., Heymans, M., Senellart, P., Van Dooren, P.: A measure of similarity between graph vertices: applications to synonym extraction and web searching. SIAM Rev. **46**(4), 647–666 (2004)
11. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to scheme matching. In: Proceedings of the 18th ICDE Conference (2002)
12. Biedl, T.C., Brandenburg, F.J.: Graph-Drawing Contest Report. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 513–521. Springer, Heidelberg (2002)