CARL VON OSSIETZKY universität OLDENBURG

BERICHTE

AUS DEM DEPARTMENT FÜR INFORMATIK
der Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften

Herausgeber: Die Professorinnen und Professoren des Departments für Informatik

Playing with Knuth's words.dat

Günther Stiege

Bericht

Nummer 1/12 - Mai 2012
ISSN 0946-2910

CARL VON OSSIETZKY universität OLDENBURG

BERICHTE

aus dem

DEPARTMENT FÜR INFORMATIK

der Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften

Herausgeber: Die Professorinnen und Professoren
des Departments für Informatik

Playing with Knuth's words.dat

Günther Stiege

Graphen und Netzwerke

Bericht

Nummer 1/12 - Mai 2012
ISSN 0946-2910

Author's address:

Prof. em. Dr. Günther Stiege
Universität Oldenburg
Graphen und Netzwerke
D-26111 Oldenburg, Germany
www-bvs.informatik.uni-oldenburg.de

This report is available from `www-bvs.informatik.uni-oldenburg.de`

Bibliographische Angaben:
Stiege, Günther:
Playing with Knuth's words.dat / Stiege, Günther – Oldenburg: Universität Oldenburg, 2012
(Berichte aus dem Department für Informatik Nr. 1/12)

ISSN 0946-2910

Playing with Knuth's words.dat

Günther Stiege
Universität Oldenburg
(May 2012)

Abstract. Knuth's file `words.dat` is a large collection of five-letter English words provided with a structure as undirected graph [Knut1993]. For this graph a complete biblock decomposition is presented and analyzed. A complete decomposition into cliques is also presented. Since `words.dat` has a strongly transitive edge partition, finding all cliques is not a hard task. First results of finding higher connected components conclude the report.

Kurzfassung. Knuth's `words.dat` ist eine umfangreiche Sammlung von Wörtern der englischen Sprache, die fünf Buchstaben lang sind. Die Sammlung wird mit einer ungerichteten Graphenstruktur versehen [Knut1993]. Für diesen Graphen wird eine vollständige Biblockzerlegung angegeben und untersucht. Außerdem werden alle Cliquen gefunden. Da `words.dat` eine stark transitive Kantenpartition aufweist, ist das nicht allzu schwierig. Erste Ergebnisse zum Auffinden höherer Zusammenhangskomponenten beschließen den Bericht.

Categories and Subject Descriptors: E.1 [**Data Structures**] (Graphs and Networks).

General Terms: Algorithms

Additional Keywords and Phrases: Strongly transitive partition, words.dat.

Contents

1 Introduction

A first draft of this report was written in 2001. For various reasons publication was delayed until now.

The Stanford GraphBase is a collection of graphs representing files, programs, and a book [Knut1993], collected, compiled, and written by Donald E. Knuth and presented to the research community in 1993 as "A Platform for Combinatorial Computing".

The Stanford GraphBase is ideally suited for teaching purposes. However, it seems to find its way into the textbook literature only slowly. A random selection of recent[1] textbooks and handbooks – Diestel [Dies1996], Cormen/Leiserson/Rivest [CormLR1990], Bollobás [Boll1998], Aho/Ullman [AhoU1995], Volkmann [Volk1996], Balakrishnan [Bala1997], Ottmann/Widmayer [OttmW1996], Atallah [Atal1999], Turau [Tura1996] – show only one short citation, namely in [Tura1996].

The most interesting graph in the Stanford GraphBase is `words.dat`. It is a collection of 5757 English five-letter words structured as a simple undirected graph. Two words are neighbors if they differ in exactly one character position. The graph is far from being trivial, but still small enough to be processed rather quickly by modern PCs and workstations, assuming algorithms with acceptable time complexity. I have chosen `words.dat` as a running example for my research in graph algorithms, and it is fun to use it[2]. Nevertheless, since `words.dat` has a rather strong clique structure as explained in section 3, it cannot be considered typical in all aspects.

`words.dat` and the program `LADDER`, also provided by Knuth, allow for ladders like

```
 0  words
 1  cords
 2  corps
 3  coops
 4  crops
 5  drops
 6  drips
 7  grips
 8  gripe
 9  grape
10  graph
```

See [Knut1993] for this and many more examples.

To deal with `words.dat` and to support research in graph algorithms in general, a program system called *Graph Handling System (GHS)* algorithms has been developed. For details on GHS see [Stie2009a]. GHS includes a variaty of functions, among them those necessary to obtain the results of this report. GHS uses incidence lists to represent graphs and requires vertex records as well as edge records to have unique names. Naturally, the vertices of `words.dat` are taken as their names. The name of an edge consists of the names of its end points, in alphabetical order, separated by an asterisk. In addition, all

[1] Selection made in 2001

[2] I have to confess, however, that I am not a native English speaker and that I probably do not know the meaning of more than 20% of the words. In the meantime a became tired of consulting the dictionary.

edge names in GHS must start with the character underscore. Vertices `edges` and `edger` are joined by edge `_edger*edges`. In GHS, graphs also must have names. Observing Knuth's requirement not to alter the original GraphBase files, `words.dat` has been given the name `GHSwords`.

This report is organized as follows:
Connected components decompose a simple graph in a natural way. Further decomposition leads to biconnected blocks (biblocks). A rather exhaustive analysis of the biblock decomposition of `words.dat` is presented in section 2.

Sometimes special properties of a graph allow interesting decompositions which are not possible or extremely time-consuming in the general case. From the definition of neighborhood in `words.dat` follows immediately that this graph has a strongly transitive edge partition. Using it, all cliques of that graph are found very easily. Section 3 deals with these questions.

Decomposing simple undirected graphs into connected components of higher order is an interesting but not so simple problem. Concerning `words.dat` some results can be obtained without much effort. Section 4 is dedicated to them.

2 The Biblock Decomposition of `words.dat`

2.1 General remarks on the biblock decomposition

Usually, undirected graphs are decomposed into connected components. They are also decomposed into blocks, where a block is a maximal biconnected subgraph or a subgraph generated from a bridge or an isolated vertex. In [Stie2007a] I introduced a similar graph decomposition, termed *biblock decomposition*, which I consider better suited for theoretic and graph algorithmic problems. Table 1 shows the building elements of the biblock decomposition. Improper connected components are isolated vertices, acyclic connected

Level 0	graph
Level 1	improper connected component
Level 1	proper connected component
Level 2	acyclic component
Level 2	cyclic component
Level 3	peripheral tree
Level 3	stopfree kernel
Level 4	internal tree
Level 4	subcomponent
Level 5	biblock

Table 1: *Building Elements of the Biblock Decomposition*

components are trees, and cyclic connected components contain a circuit. In the latter, three equivalence relations between lines are defined:

1. Two lines are stropfree related if there is a stopfree path[3] passing through both.
2. Two lines are line-simple related if there is a closed line-simple path through both.
3. Two lines are circuit related if there is a circuit through both.

The subgraphs generated from these equivalence classes are *stopfree kernel*, *subcomponents*, and *biblocks*. The stopfree kernel of a cyclic connected component is unique. It consists of one ore more subcomponents. The lines not in the stopfree kernel give rise to the peripheral trees. Each subcomponent consists of one or more biblocks. The lines of a subcomponent not in a biblock give rise to the internal trees. Figure 1 shows the elements of the biblock decomposition. The figure shows also the vertices common to different building elements, the attachment points. Border points are common to peripheral trees and the stopfree kernel. Check points are common to an internal tree and a subcomponent, Hinge points are common points of different biblocks of a subcomponent. A vertex may have all three properties simultaneously. The building elements have both, an internal name – e.g `WCOMP145` – and an alphanumeric name – e.g. `_abaca*abaci`. The latter is the smallest of its line names and a such unique.

The biblock decomposition of a graph leads to a derived bipartite graph, the *biblock graph*. The building elements, namely improper connected components, acyclic components, peripheral trees, internal trees, and biblocks on the one hand and the attachment points on the other hand are the vertices. An attachment point is joined by an edge to every elementary building block it is part of. There are no other edges. The biblock graph is always cyclefree. It is connected if the original graph is. In this case it is called *biblock tree*.

[3] A closed path where no line is followed by itself and the first line differs from the last.

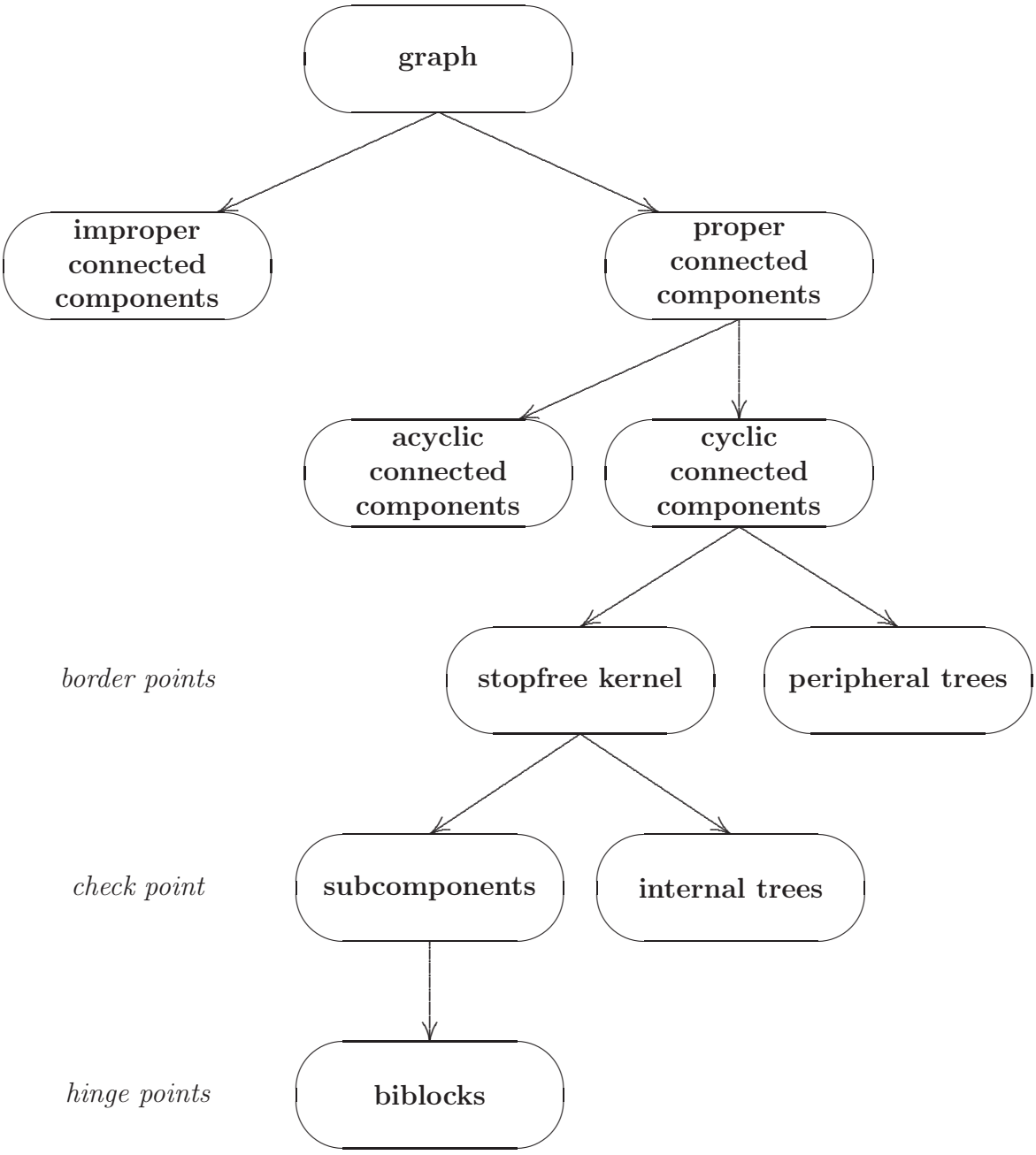

Figure 1: *Biblock Decomposition of General Graphs*

2.2 The Small Connected Components

When exploring an unknown graph it is always a good idea to start with simple degree statistics. Knuth presents them in section 1.1 of his book [Knut1993] and they are repeated here in table 2 (in GHS format). To find the biblock decomposition of `words.dat` the

```
Statistics of graph GHSwords    Type = UGSLF
  #Vertices =  5757
  #Edges    = 14135  #Mult.edges =   0  #Loops(u) =   0  #Mult.loops(u) =   0
  #Arcs     =     0  #Mult.arcs  =   0  #Loops(d) =   0  #Mult.loops(d) =   0

Degree summary (undirected loops are counted twice!):
      Mean total degree                  4.91
      Standard deviation of total degree 4.43
    671 Vertices of total degree     0
    774 Vertices of total degree     1
    727 Vertices of total degree     2
    638 Vertices of total degree     3
    523 Vertices of total degree     4
    428 Vertices of total degree     5
    329 Vertices of total degree     6
    280 Vertices of total degree     7
    249 Vertices of total degree     8
    213 Vertices of total degree     9
    188 Vertices of total degree    10
    162 Vertices of total degree    11
    120 Vertices of total degree    12
    116 Vertices of total degree    13
    102 Vertices of total degree    14
     75 Vertices of total degree    15
     53 Vertices of total degree    16
     32 Vertices of total degree    17
     32 Vertices of total degree    18
     20 Vertices of total degree    19
      8 Vertices of total degree    20
      6 Vertices of total degree    21
      4 Vertices of total degree    22
      2 Vertices of total degree    23
      3 Vertices of total degree    24
      2 Vertices of total degree    25
```

Table 2: *General Statistics of* `words.dat`

function `astd` is applied. Since a rather large amount of decomposition items are found by that function it is convenient to start with a condensed summary of the results. This is shown in table 3. The table starts with general information: Name and type of the graph, numbers of vertices, edges and arcs. After that, more information is returned than we asked for. The numbers of all five types of weak connected components are provided.

```
BEGIN CONDENSED STRUCTURE OF GENERAL DECOMPOSITION
$GRAPH GHSwords
$TYPE  UGSLF
$No_VERTICES                          5757
$No_EDGES                            14135
$No_ARCS                                 0
$No_ISOLATED_VERTICES                  671
$No_WEAK_COMPONENTS_TYPE_1               0  (0 f-trees)
$No_WEAK_COMPONENTS_TYPE_2             145  (145 f-trees)
$No_WEAK_COMPONENTS_TYPE_3               0  (0 rooted)
$No_WEAK_COMPONENTS_TYPE_4               0  (0 rooted)
$No_WEAK_COMPONENTS_TYPE_5              37  (37 rooted)
$No_WEAK_ATTACHMENT_POINTS               0
$No_STRONG_COMPONENTS_(f-ACYCLIC)      145
$No_STRONG_COMPONENTS_(f-CYCLIC)        37
$No_STOPFREE_KERNELS                    37
$No_PERIPHERAL_TREES                   429
$No_SUBCOMPONENTS                       68
$No_BIBLOCKS                            93
$No_INTERNAL_TREES                      31
$No_HINGE_POINTS                        25
$No_CHECK_POINTS                        62
END CONDENSED STRUCTURE OF GENERAL DECOMPOSITION
```

Table 3: *Condensed Statistics of the General Decomposition of* `words.dat`

It so happens that only two of the five types can occur in an undirected graph. So we have 145 components which are trees and and 37 with a stopfree kernel needing a biblock decomposition. The remaining entries of the table summarize the decomposition of the cyclic components. Table 4 shows a statistic grouping the weak components according to their numbers of vertices and edges. Tables 5 and 6 show complete lists.

As can be seen in table 5, the trees are very simple: 103 of them consist of 2 vertices, 30 have 3 vertices. Only three vertices – `auger` in tree `_angel*anger`, `macho` in tree `_macho*macro` and `unban` in tree `_unban*unbar` – have degree 3, all others have degree 1 or 2. The largest tree – `_bogie*bowie` – has 7 vertices. See remark 3.1, page 23, why larger trees are unlikely to exist and why no vertex may incide with more than 5 tree edges.

Table 6 lists the 37 cyclic components. The component `_abaca*abaci` has 4493 vertices and 13619 edges and is much larger than any of the others. It is called the *giant component* by Knuth and will be dealt with in detail in subsection 2.3. The remaining 36 cyclic components are rather small. 25 of them have the simplest possible stopfree kernel, namely a biblock of 3 vertices and 3 edges. 12 of these have no other edges, 13 show 1 or more simple peripheral trees.

Acyclic components		Cyclic components		
No	Vertices	No	Vertices	Edges
103	2	12	3	3
30	3	6	4	4
7	4	4	5	5
2	5	1	6	6
2	6	1	6	9
1	7	4	7	7
-	-	1	7	12
-	-	1	8	9
-	-	1	15	15
-	-	1	15	16
-	-	1	15	19
-	-	1	17	42
-	-	1	19	22
-	-	1	24	50
-	-	1	4493	13619

Table 4: *Numbers of weak components grouped according to size*

The biblock structure of the remaining 13 of the 36 small cyclic components is somewhat more interesting. The components `_refit*refix, _avers*avert`, `_unsee*unset`, and `_dicot*didot` contain 2 subcomponents joined by an internal tree. Each of the components `_allot*allow` and `_unsee*unset` contain a subcomponent consisting of two biblocks connected by a hinge point. The only biblocks of more than 10 vertices are `_bound*found` and `_biffs*biffy` in the components having the same names. The former contains the K_8 {`bound, found, hound, mound, pound, round, sound, wound`}. See figure 2 in Knuth's book [Knut1993]. As a short inspection of that figure reveals, it consists of a biblock and two single-line peripheral trees. The latter shows an interesting mixed connection and we will return to it in example 3.1, page 23. in section 3,

```
wacsno = 145
wcompno = 0   wcompname = _abets*abuts     (2V, 1E, 0A)
wcompno = 1   wcompname = _admit*admix     (2V, 1E, 0A)
  ....
  ....
  ....
wcompno = 143   wcompname = _valor*vapor     (2V, 1E, 0A)
wcompno = 144   wcompname = _wassa*watsa     (2V, 1E, 0A)
-------------------------------------------------------
wcompno = 5   wcompname = _ahoys*bhoys     (3V, 2E, 0A)
wcompno = 9   wcompname = _alike*alive     (3V, 2E, 0A)
  ....
  ....
  ....
wcompno = 133   wcompname = _tonal*total     (3V, 2E, 0A)
-------------------------------------------------------
wcompno = 2   wcompname = _aerie*eerie     (4V, 3E, 0A)
wcompno = 15   wcompname = _amped*umped     (4V, 3E, 0A)
wcompno = 46   wcompname = _dance*dunce     (4V, 3E, 0A)
wcompno = 64   wcompname = _felon*melon     (4V, 3E, 0A)
wcompno = 81   wcompname = _inapt*inept     (4V, 3E, 0A)
wcompno = 87   wcompname = _kazoo*wazoo     (4V, 3E, 0A)
wcompno = 88   wcompname = _kabob*kebob     (4V, 3E, 0A)
-------------------------------------------------------
wcompno = 16   wcompname = _angel*anger     (5V, 4E, 0A)
wcompno = 97   wcompname = _macho*macro     (5V, 4E, 0A)
-------------------------------------------------------
wcompno = 95   wcompname = _logic*login     (6V, 5E, 0A)
wcompno = 137   wcompname = _unban*unbar     (6V, 5E, 0A)
-------------------------------------------------------
wcompno = 36   wcompname = _bogie*bowie     (7V, 6E, 0A)
```

Table 5: *Acyclic weak components of GHSwords*

```
wccsfno = 37
wcompno = 161   wcompname = _deign*feign     (3V, 3E, 0A)
wcompno = 153   wcompname = _arbor*ardor     (3V, 3E, 0A)
wcompno = 169   wcompname = _heigh*neigh     (3V, 3E, 0A)
wcompno = 181   wcompname = _whelk*whelm     (3V, 3E, 0A)
wcompno = 147   wcompname = _adapt*adept     (3V, 3E, 0A)
wcompno = 179   wcompname = _unarc*unarm     (3V, 3E, 0A)
wcompno = 167   wcompname = _getup*letup     (3V, 3E, 0A)
wcompno = 159   wcompname = _condo*mondo     (3V, 3E, 0A)
wcompno = 175   wcompname = _paean*pagan     (3V, 3E, 0A)
wcompno = 162   wcompname = _demur*femur     (3V, 3E, 0A)
wcompno = 174   wcompname = _opals*orals     (3V, 3E, 0A)
wcompno = 180   wcompname = _urged*urger     (3V, 3E, 0A)
wcompno = 148   wcompname = _adman*admen     (4V, 4E, 0A)
wcompno = 172   wcompname = _humor*rumor     (4V, 4E, 0A)
wcompno = 152   wcompname = _anion*onion     (4V, 4E, 0A)
wcompno = 168   wcompname = _gumbo*jumbo     (4V, 4E, 0A)
wcompno = 166   wcompname = _field*fiend     (4V, 4E, 0A)
wcompno = 177   wcompname = _rigor*vigor     (4V, 4E, 0A)
wcompno = 165   wcompname = _enact*epact     (5V, 5E, 0A)
wcompno = 157   wcompname = _buzzy*fuzzy     (5V, 5E, 0A)
wcompno = 176   wcompname = _patio*ratio     (5V, 5E, 0A)
wcompno = 164   wcompname = _eject*elect     (5V, 5E, 0A)
wcompno = 163   wcompname = _edits*emits     (6V, 6E, 0A)
wcompno = 178   wcompname = _squab*squad     (6V, 9E, 0A)
wcompno = 150   wcompname = _amber*ember     (7V, 7E, 0A)
wcompno = 171   wcompname = _infix*unfix     (7V, 7E, 0A)
wcompno = 151   wcompname = _amble*ample     (7V, 7E, 0A)
wcompno = 173   wcompname = _knead*kneed     (7V, 7E, 0A)
wcompno = 158   wcompname = _cable*fable     (7V, 12E, 0A)
wcompno = 149   wcompname = _agley*alley     (8V, 9E, 0A)
wcompno = 154   wcompname = _bebug*debug     (15V, 15E, 0A)
wcompno = 146   wcompname = _abend*amend     (15V, 16E, 0A)
wcompno = 170   wcompname = _idled*idler     (15V, 19E, 0A)
wcompno = 156   wcompname = _bound*found     (17V, 42E, 0A)
wcompno = 160   wcompname = _dados*didos     (19V, 22E, 0A)
wcompno = 155   wcompname = _biffs*biffy     (24V, 50E, 0A)
wcompno = 145   wcompname = _abaca*abaci     (4493V, 13619E, 0A)
```

Table 6: *Cyclic weak components of GHSwords*

2.3 The Giant Component

Of course, the most interesting component is the giant component `_ached*aches`. Its structure is listed in tables 28 to 42, in the appendix. Table 7 shows the general statistics of the giant component. As expected, the mean degree has risen, namely from 4.91 in the entire graph to 6.06 in the giant component. Table 8 shows the condensed statistics of its biblock decomposition. There are a rather large number of peripheral trees and 29 subcomponents divided into 52 biblocks. Table 41, page 57, shows that the giant component contains a giant subcomponent of 3756 vertices and 12792 edges, whose name is also `_ached*aches`. This subcomponent in turn contains a giant biblock of 3708 vertices and 12711 edges. Its name is again `_ached*aches`.

For further study of the giant component, we will use its biblock tree instead of the tables. Figures 2, 3, 4, and 5 show the biblock tree of the giant component. Circles stand for attachment points, rectangles for peripheral or internal trees, and ovals for biblocks. Naturally, the giant biblock `_ached*aches` has been chosen as root of the tree and covers all 4 figures. Each of the subtrees starting at the root is uniquely identified by an attachment point. The names of these have been put into the root's representation and represent the attachment point as well as the corresponding subtree. We see that the structure of all subtrees is rather simple. The deepest is `arise`. It is of depth 10, of breadth 4, and is shown with more details in figure 6. The part of the original `words.dat` corresponding to subtree `arise` is shown in figure 7. Subtrees `capes` and `booby` are of depth 8 and the subtrees from `scarf` to `enter` are of depth 6. Among them, subtree `begat` is of special interest in spite of its simple appearance (see figure 3). It comprises biblock `_bight*eight` which is the K_9 {`bight, eight, fight, light, might, night, right, sight, tight`}. 19 subtrees of depth 4 follow – `swash` to `quite`. All other subtrees have the minimal depth 2. Only 2 of them – `brave` and `curie` – have breadth 2, all others are of breadth 1. The subtrees from `women` to `vires` all have at least 4 vertices, the largest being `women` with 9 vertices. The subtrees `bevel` to `tense` are the simplest possible biblocks, K_3s. From `aimed` to `yours` 73 peripheral trees consisting of 3 vertices each follow. The remaining 239 peripheral trees from `agone` to `zings` are 'pins', i.e. have 2 vertices only.

```
Statistics of graph  _ached*aches   Type = UGSLF
  #Vertices =  4493
  #Edges    = 13619  #Mult.edges =   0  #Loops(u) =   0  #Mult.loops(u) =   0
  #Arcs     =     0  #Mult.arcs  =   0  #Loops(d) =   0  #Mult.loops(d) =   0

Degree summary (undirected loops are counted twice!):
      Mean total degree                  6.06
      Standard deviation of total degree 4.32
    435 Vertices of total degree     1
    564 Vertices of total degree     2
    584 Vertices of total degree     3
    505 Vertices of total degree     4
    426 Vertices of total degree     5
    326 Vertices of total degree     6
    270 Vertices of total degree     7
    247 Vertices of total degree     8
    212 Vertices of total degree     9
    187 Vertices of total degree    10
    162 Vertices of total degree    11
    120 Vertices of total degree    12
    116 Vertices of total degree    13
    102 Vertices of total degree    14
     75 Vertices of total degree    15
     53 Vertices of total degree    16
     32 Vertices of total degree    17
     32 Vertices of total degree    18
     20 Vertices of total degree    19
      8 Vertices of total degree    20
      6 Vertices of total degree    21
      4 Vertices of total degree    22
      2 Vertices of total degree    23
      3 Vertices of total degree    24
      2 Vertices of total degree    25
```

Table 7: *General Statistics of the Giant Component*

```
BEGIN CONDENSED STRUCTURE
$GRAPH _ached*aches
$TYPE  UGSLF
$No_VERTICES              4493
$No_EDGES                13619
$No_ISOLATED_VERTICES        0
$No_ACYCLIC_COMPONENTS       0     (0V, 0E)
$No_CYCLIC_COMPONENTS        1     (4493V, 13619E)
$No_PERIPHERAL_TREES       393
$No_STOPFREE_KERNELS         1
$No_SUBCOMPONENTS           29
$No_BIBLOCKS                52
$No_INTERNAL_TREES          28
$No_ATTACHMENT_POINTS      461
$No_BORDER_POINTS          393
$No_CHECK_POINTS            56
$No_HINGE_POINTS            23
END CONDENSED STRUCTURE
```

Table 8: *Condensed Statistics of the Biblock Decomposition of the Giant Component*

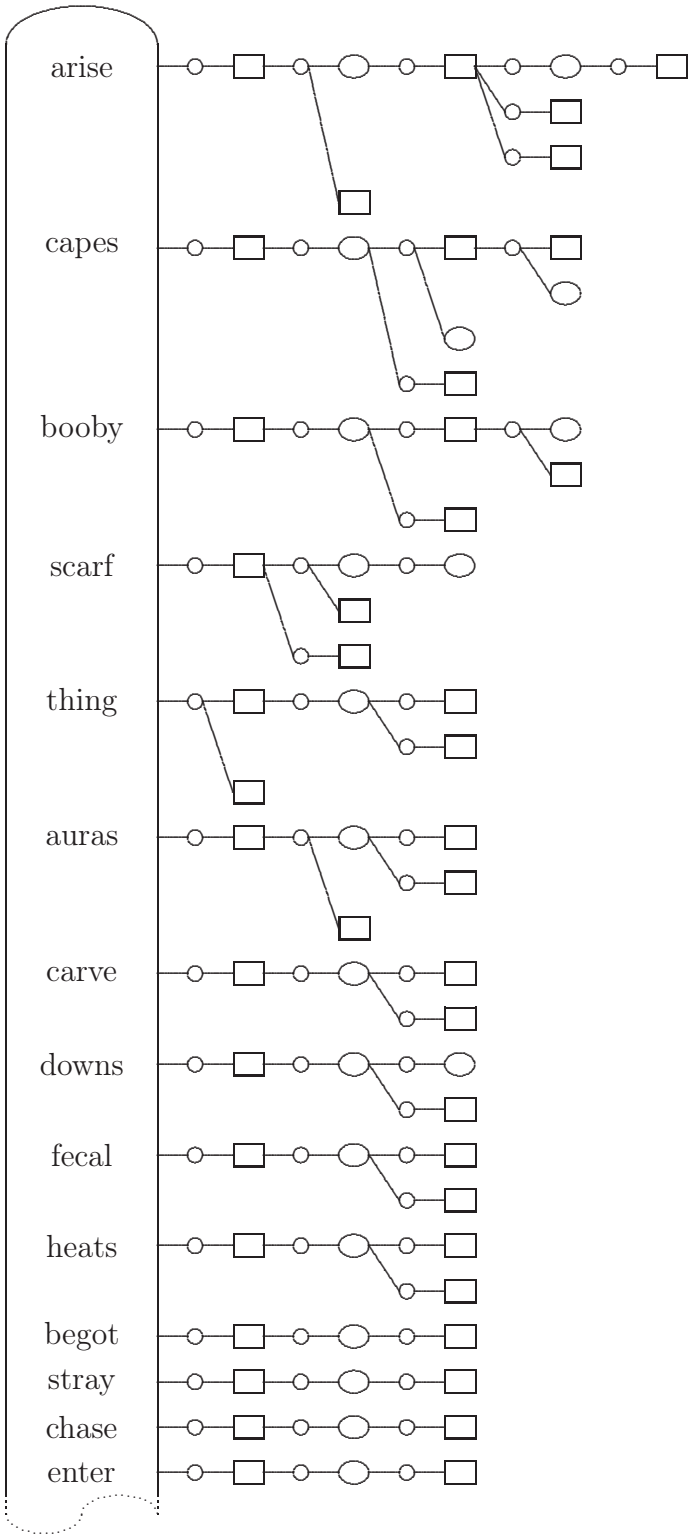

Figure 2: *Biblock Tree of the Giant Component (Part 1)*

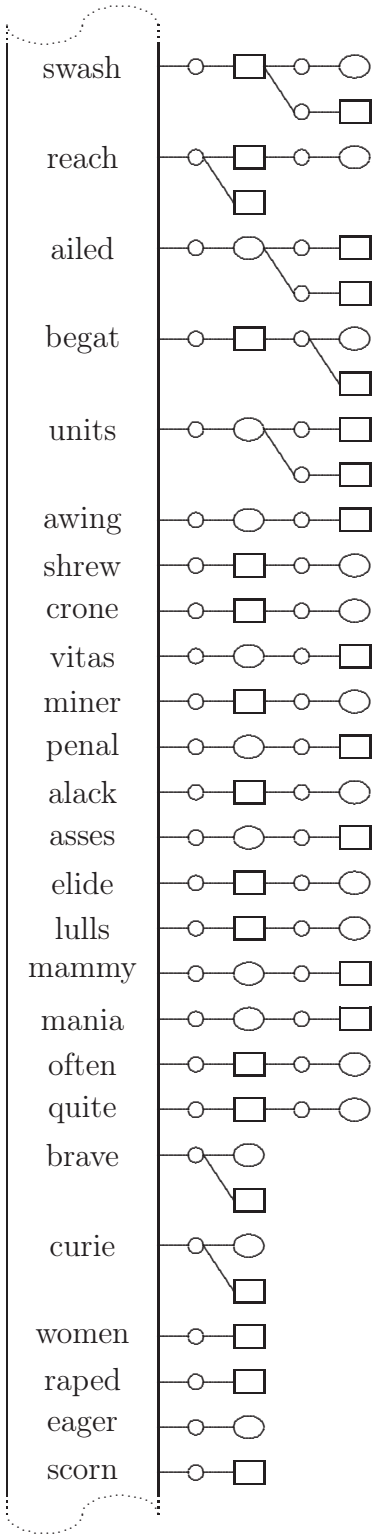

Figure 3: *Biblock Tree of the Giant Component (Part 2)*

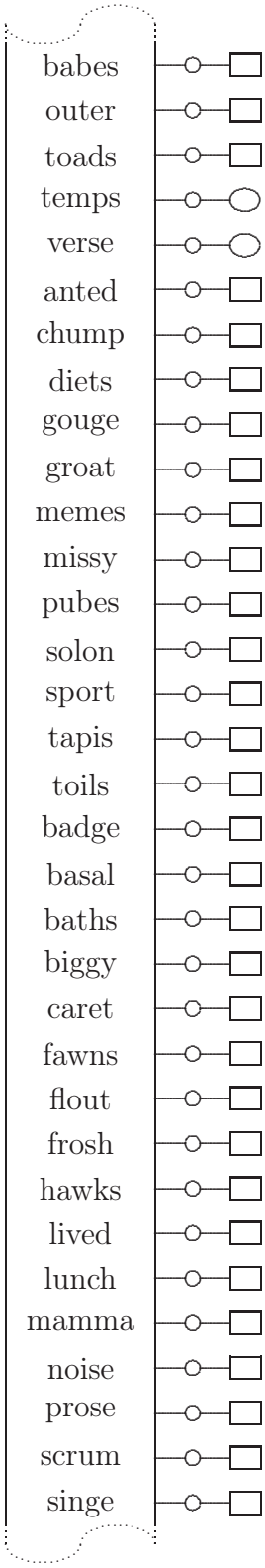

Figure 4: *Biblock Tree of the Giant Component (Part 3)*

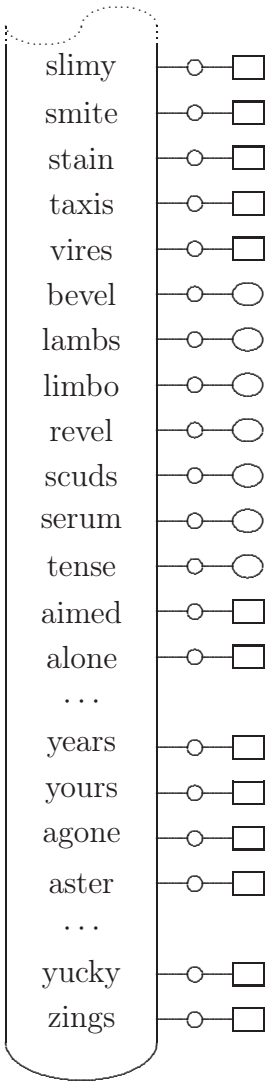

Figure 5: *Biblock Tree of the Giant Component (Part 4)*

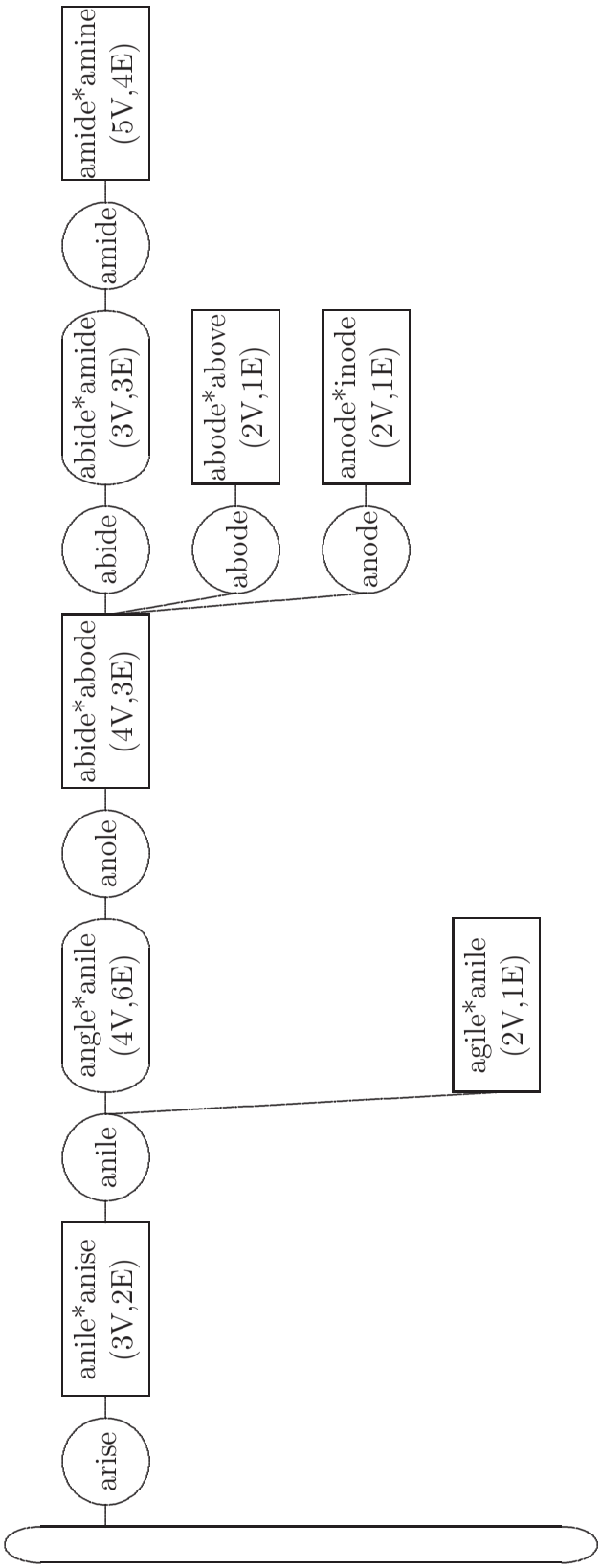

Figure 6: *Subtree* `arise` *of the Biblock Tree of the Giant Component*

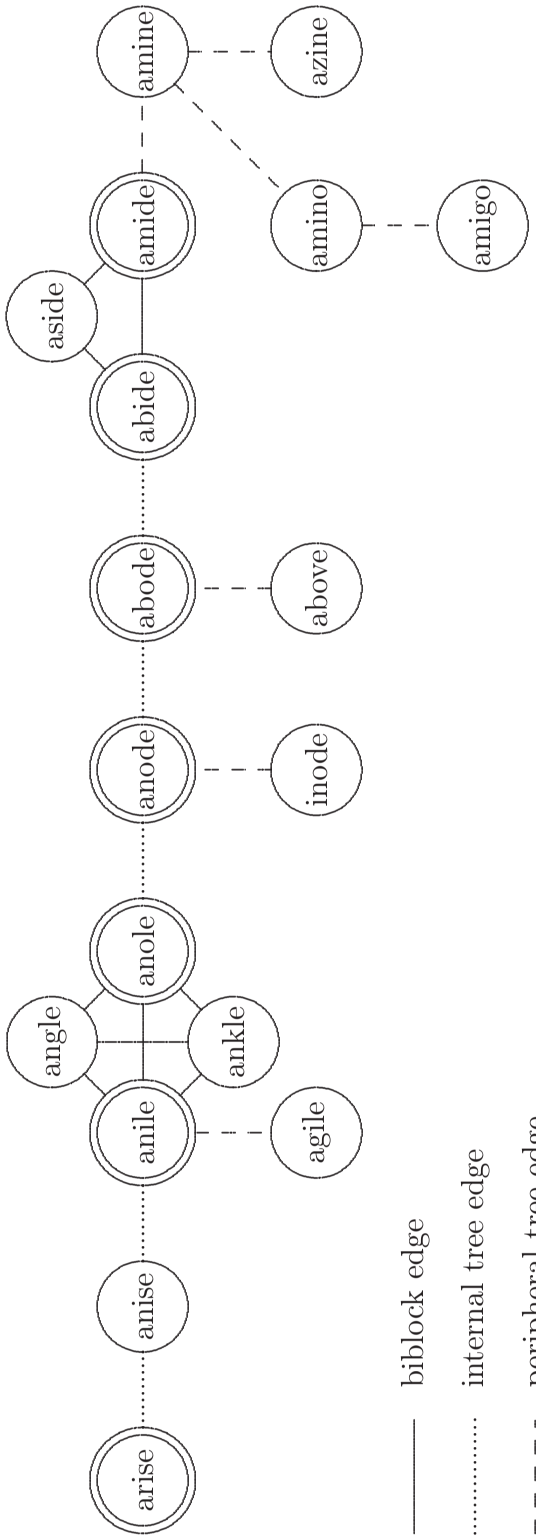

Figure 7: *Original Graph of Subtree* `arise`

3 The Clique Decomposition of `words.dat`

In `words.dat` two vertices are joined by an edge if their characters differ in exactly one position and the position is the same in both words. Playing with `words.dat` one gets rather quickly the impression, that edges of the same position are closer related than edges of different positions. Grouping the edges according to the position number results in an edge partition. We may paint each edge with a color i ($i = 0, \ldots, 4$) if the edge's end points differ in character position i. We speak of an edge coating.[4] In this sense, every edge coating is an edge partition and vice-versa. As such it is transitive, of course. However, the position dependent edge partition we started with, has additional and far reaching properties.

Lemma 3.1 *For the edge partition generated by the the position numbers of the edges of* `words.dat` *holds:*

1. *If there are edges of color i from vertex u to vertex v and from vertex v to w, then there exists an edge of color i from u to w.*

2. *K_3 subgraphs of* `words.dat` *have edges of only one color.*

Proof: Property 1. follows easily from the definition of edges in `wordsdat`.
Property 2: Assume the following three words

$$\begin{array}{lllllll} u & = & u_0 & u_1 & u_2 & u_3 & u_4 \\ v & = & v_0 & v_1 & v_2 & v_3 & v_4 \\ w & = & w_0 & w_1 & w_2 & w_3 & w_4 \end{array}$$

Let an edge, say of color 0, join u an v. Then $u_0 \neq v_0$ and $u_i = v_i$ for $i = 1, 2, 3, 4$. If v and w are joined by an edge of another color, say color 3, then $v_0 = w_0$, $v_1 = w_1$, $v_2 = w_2$, $v_3 \neq w_3$ and $v_4 = w_4$. It follows that $w_0 = v_0 \neq u_0$ and $w_3 \neq v_3 = u_3$. w differs from u in more than one position. No edge joining w and u is possible. □

The lemma motivates a definition.

Definition 3.1 *An edge coating is called* strongly transitive *if the following holds*

1. *If vertices u and v are joined by an edge of of color i and vertices v and w are joined by an edge of the same color, then there is an edge of color i joining u an w.*

2. *The edges of triangles are of the same color.*

From 1. follows that a vertex v together with all its i-neighbors form a maximal clique of color i. From 2. follows that there are no other cliques: All cliques are monochromatic. Two maximal cliques must not have edges in common. The may have at most one vertex in common. If the do, they must be of different colors. Therefore, in any strongly transitive edge relation maximal cliques are an edge partition which is a refinement of that relation. The classes of this refinement are the connected components of the subgraphs generated from the edges of the same color. These can very easily be found by a color-restricted

[4] The term *edge coloring* has a well established different meaning in graph theory.

depth-first search. Both partitions have the same set of attachment points. A vertex is an attachment point if and only if it is incident with edges of different colors.

So, we have a very comfortable way of finding all clique in `words.dat`. With the help of GHS this has been done. The results follow. We start with the distribution of colors among the set of edges. It is shown in table 9. As is to be expected, the colors are not

Color	`words.dat`		*Giant Component*		*Giant Biblock*	
0	5510	38.98%	5314	39.02%	5077	39.94%
1	1679	11.88%	1608	11.81%	1476	11.61%
2	2855	20.20%	2775	20.38%	2610	20.53%
3	2133	15.09%	2093	15.36%	1955	15.38%
4	1958	13.85%	1829	13.43%	1593	12.54%
	14135	100.00%	13619	100.00%	12711	100.00%

Table 9: *Edge Color Distribution in* `words.dat`, *in the giant component and in the giant biblock*

uniformly distributed. Table 9 shows the color distribution in `words.dat`, in its giant component, and in its giant biblock. They show almost the same color distributions.

Table 10 presents the distribution of clique sizes within the same three graphs. Again they

Clique Type	`words.dat`			*Giant Component*			*Giant Biblock*		
	#Cli.	#Edg.	%	#Cli.	#Edg.	%	#Cli.	#Edg.	%
2 (1 edge)	3196	3196	22.61 %	2871	2871	21.08 %	2210	2210	17.40 %
3 (3 edges)	910	2730	19.31 %	872	2616	19,21 %	832	2496	19.64 %
4 (6 edges)	360	2160	15.28 %	357	2142	15.73 %	346	2076	16.33 %
5 (10 edges)	166	1660	11.74 %	165	1650	12.12 %	164	1640	12.90 %
6 (15 edges)	77	1155	8.17 %	77	1155	8.48 %	76	1140	8.97 %
7 (21 edges)	53	1113	7.87 %	52	1092	8.02 %	52	1092	8.59 %
8 (28 edges)	28	784	5.29 %	27	756	5.55 %	27	756	5.95 %
9 (36 edges)	12	432	3.06 %	12	432	3.17 %	11	396	3.11 %
10 (45 edges)	13	585	4.14 %	13	585	4.30 %	13	585	4.60 %
11 (55 edges)	2	110	0.78 %	2	110	0.81 %	2	110	0.87 %
12 (66 edges)	2	132	0.93 %	2	132	0.97 %	2	132	1.04 %
13 (78 edges)	1	78	0.55 %	1	78	0.57 %	1	78	0.61 %
Σ		14135	100.00%		13619	100.00%		12711	100.00%

Table 10: *Clique Distribution in* `words.dat`, *in the giant component and in the giant biblock*

show very similar distributions. A slight increase of the percentages of larger cliques can be observed. The large cliques – from 9 vertices on – are shown in table 11. The largest clique not in the the giant biblock is the K_9 {`bight, eight, fight, light, might, night,`

right, sight, tight} (see page 12). The largest clique not in the giant component is the K_8 {bound, found, hound, mound, pound, round, sound, wound} (see page 9). Tables 12 and 13 show the combined statistics of sizes and colors of cliques. Again words.dat, the giant component, and the giant biblock have similar distributions. However, the distribution of colors over clique sizes is more or less uniform only for cliques of 3 and 4 vertices. For larger cliques color 0 – the first position in a word – dominates. The largest cliques of color 4 are of size 5. There are 10 of them, an example is {sward, sware, swarf, swarm, swart}. Four cliques of size 6 are the largest of color 1, an example is {scats, seats, slats, spats, stats, swats}. The clique of size 8 {flabs, flags, flaks, flams, flaps, flats, flaws, flays} is the largest of color 3. The three largest cliques of color 2 are of size 9. All other cliques with 9 or more vertices are of color 0 as shown in table 11.

Remark 3.1 Assume graph G has a strongly transitive edge relation. Then in no tree of the biblock decomposition (acyclic weak component, peripheral tree, internal tree) any edge may be followed by an edge of the same color. Trees are color-alternating.
It also holds that no vertex may incide with more tree edges than the number of colors of the edge relation.
Therefore, in words.dat no vertex of a tree incides with more than five tree edges. Any vertex with degree at least 6 incides with clique edges. □

Example 3.1 In this example we shall analyze the connected component _biffs*biffy. It is shown in figure 8. To avoid overloading the picture, the lines inside the figure representing K7 or K4 are not drawn. The biblock decomposition of _biffs*biffy is rather simple. It consists of one single biblock and a simple peripheral tree. Its decomposition in cliques is of more interest. To this end the edges have been colored according to position number. "Red" corresponds to position 0, "green" to position 1, and blue to position 4. Edge positions 2 and 3 do not occur. There are 5 cliques with more than 2 vertices: 3 of them are red (K7, K4, K3), 1 is green (K3), and 1 is blue (K3). The only vertex common to more than one of these cliques is buffs. □

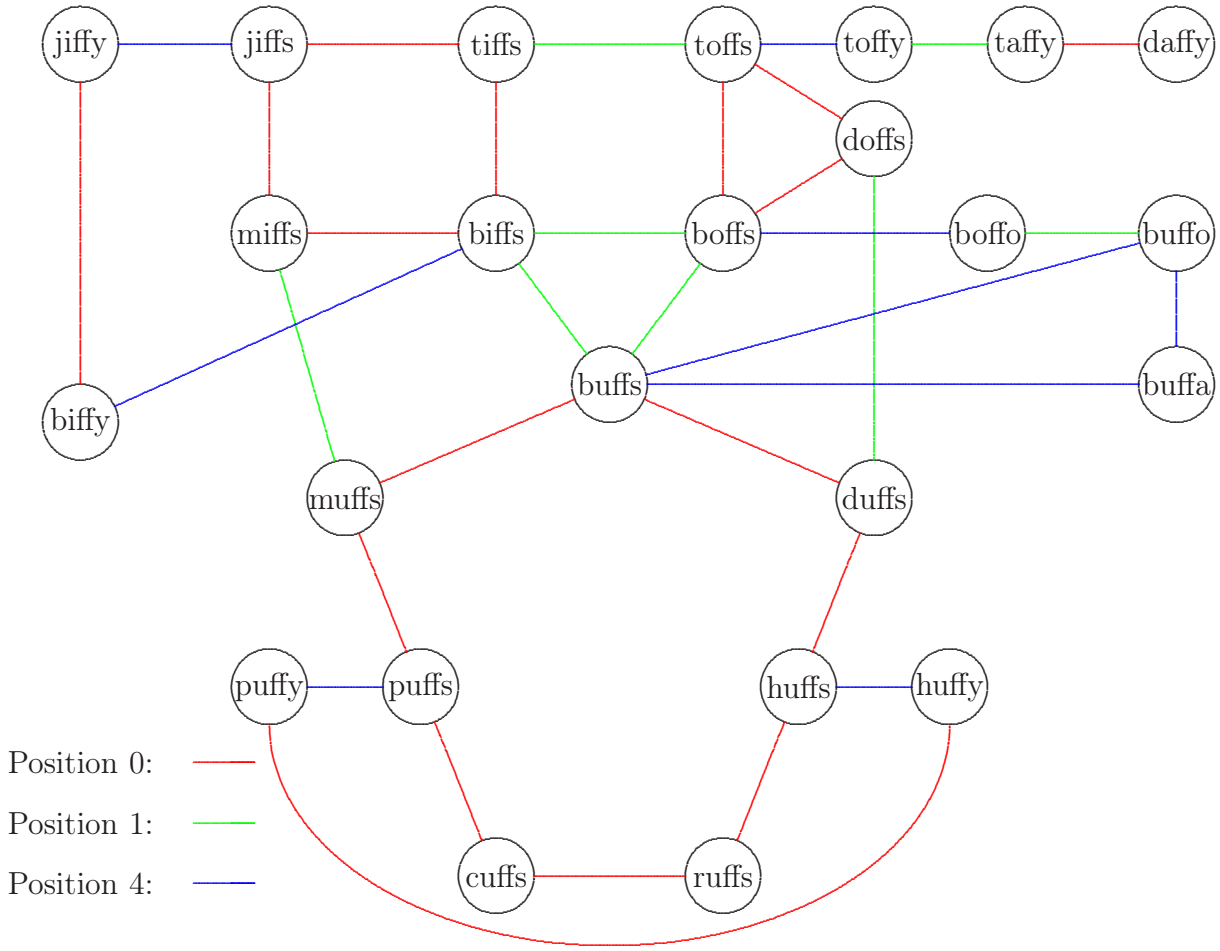

Figure 8: *Structur of the connected component* `_biffs*biffy`

Clique Type	*Color*	*Clique Vertices*
9 (36 edges)	0	bests, jests, nests, pests, rests, tests, vests, wests, zests
	0	bight, eight, fight, light, might, night, right, sight, tight
	0	bikes, dikes, hikes, kikes, likes, mikes, pikes, tikes, yikes
	0	books, cooks, gooks, hooks, kooks, looks, nooks, rooks, zooks
	0	boots, coots, foots, hoots, loots, moots, roots, soots, toots
	0	bores, cores, fores, gores, lores, mores, pores, sores, yores
	0	bowed, cowed, lowed, mowed, rowed, sowed, towed, vowed, wowed
	0	bumps, dumps, humps, jumps, lumps, mumps, pumps, rumps, sumps
	0	caves, eaves, haves, laves, naves, paves, raves, saves, waves
	2	codes, cokes, comes, cones, copes, cores, cotes, coves, coxes
	2	lacer, lager, laker, lamer, laser, later, laver, laxer, layer
	2	pacer, pager, paler, paper, parer, pater, paver, pawer, payer
10 (45 edges)	0	bakes, cakes, fakes, jakes, lakes, makes, rakes, sakes, takes, wakes
	0	bares, cares, dares, fares, hares, mares, nares, pares, tares, wares
	0	bates, dates, fates, gates, hates, mates, nates, pates, rates, sates
	0	bells, celld, dells, fells, hells, jells, sells, tells, wells, yells
	0	bends, fends, lends, mends, pends, rends, sends, tends, vends, wends
	0	bocks, cocks, docks, hocks, jocks, locks, mocks, pocks, rocks, socks
	0	bower, cower, dower, lower, mower, power, rower, sower, tower, vower
	0	bulls, culls, dulls, fulls, gulls, hulls, lulls, mulls, nulls, pulls
	0	cater, dater, eater, hater, later, mater, pater, rater, tater, water
	0	deals, heals, meals, peals, reals, seals, teals, veals, weals, zeals
	0	dicks, hicks, kicks, licks, micks, nicks, picks, sicks, ticks, wicks
	0	dines, fines, lines, mines, nines, pines, sines, tines, vines, wines
	0	dings, jings, kings, lings, pings, rings, sings, tings, wings, zings
11 (55 edges)	0	bales, dales, gales, hales, kales, males, pales, sales, tales, vales, wales
	0	dinks, finks, jinks, kinks, links, minks, oinks, pinks, rinks, sinks, winks
12 (66 edges)	0	bails, fails, hails, jails, mails, nails, pails, rails, sails, tails, vails, wails
	0	bears, dears, fears, gears, hears, nears, pears, rears, sears, tears, wears, years
13 (78 edges)	0	bills, cills, dills, fills, gills, hills, kills, mills, pills, rills, sills, tills, wills

Table 11: *Cliques of Size 9 and Larger in* `words.dat`

Clique Type	*Color*	*No. of Cliques in* words.dat		*No. of Cliques in the Giant Component*		*No. of Cliques in the Giant Biblock*	
2 (1 edge)	0	526	16.46%	440	15.33%	305	13.80%
	1	658	20.59%	608	21.18%	516	23.35%
	2	675	21.12%	616	21.45%	496	22.44%
	3	535	16.74%	498	17.35%	369	16.70%
	4	802	25.09%	709	24.69%	524	23.71%
	Σ	3196	100.00%	2871	100.00%	2210	100.00%
3 (3 edges)	0	210	23.08%	195	22.36%	184	22.12%
	1	167	18.35%	160	18.35%	154	18.51%
	2	158	17.36%	151	17.32%	140	16.82%
	3	147	16.15%	146	16.74%	143	17.19%
	4	228	25.06%	220	25.23%	211	25.36%
	Σ	910	100.00%	872	100.00%	832	100.00%
4 (6 edges)	0	115	31.94%	114	31.93%	111	32.08%
	1	50	13.89%	50	14.01%	48	13.87%
	2	72	20.00%	72	20.17%	70	20.23%
	3	61	16.95%	61	17.09%	61	17.63%
	4	62	17.22%	60	16.80%	56	16.18%
	Σ	360	100.00%	357	100.00%	346	100.00%
5 (10 edges)	0	68	40.96%	67	40.61%	67	40.85%
	1	16	9.64%	16	9.70%	15	9.15%
	2	29	17.47%	29	17.58%	29	17.68%
	3	43	25.91%	43	26.06%	43	26.22%
	4	10	6.02%	10	6.06%	10	6.10%
	Σ	166	100.00%	165	100.00%	164	100.00%
6 (15 edges)	0	47	61.04%	47	61.04%	46	60.53%
	1	4	5.19%	4	5.19%	4	5.26%
	2	15	19.48%	15	19.48%	15	19.74%
	3	11	14.29%	11	14.29%	11	14.47%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	77	100.00%	77	100.00%	76	100.00%
7 (21 edges)	0	30	56.61%	29	55.77%	29	55.77%
	1	0	0.00%	0	0.00%	0	0.00%
	2	15	28.30%	15	28.85%	15	28.85%
	3	8	15.09%	8	15.38%	8	15.38%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	53	100.00%	52	100.00%	52	100.00%
8 (28 edges)	0	15	53.57%	14	51.85%	14	51.85%
	1	0	0.00%	0	0.00%	0	0.00%
	2	12	42.86%	12	44.44%	12	44.44%
	3	1	3.57%	1	3.70%	1	3.70%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	28	100.00%	27	100.00%	27	100.00%

Table 12: *Cliques Distribution in* words.dat, *in the giant componentent and in the giant biblock (Part 1)*

Clique Type	*Color*	*No. of Cliques in* `words.dat`		*No. of Cliques in the Giant Component*		*No. of Cliques in the Giant Biblock*	
9 (36 edges)	0	9	75.00%	9	75.00%	8	72.73%
	1	0	0.00%	0	0.00%	0	0.00%
	2	3	25.00%	3	25.00%	3	27.27%
	3	0	0.00%	0	0.00%	0	0.00%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	12	100.00%	12	100.00%	11	100.00%
10 (45 edges)	0	13	100.00%	13	100.00%	13	100.00%
	1	0	0.00%	0	0.00%	0	0.00%
	2	0	0.00%	0	0.00%	0	0.00%
	3	0	0.00%	0	0.00%	0	0.00%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	13	100.00%	13	100.00%	13	100.00%
11 (55 edges)	0	2	100.00%	2	100.00%	2	100.00%
	1	0	0.00%	0	0.00%	0	0.00%
	2	0	0.00%	0	0.00%	0	0.00%
	3	0	0.00%	0	0.00%	0	0.00%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	2	100.00%	2	100.00%	2	100.00%
12 (66 edges)	0	2	100.00%	2	100.00%	2	100.00%
	1	0	0.00%	0	0.00%	0	0.00%
	2	0	0.00%	0	0.00%	0	0.00%
	3	0	0.00%	0	0.00%	0	0.00%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	2	100.00%	2	100.00%	2	100.00%
13 (78 edges)	0	1	100.00%	1	100.00%	1	100.00%
	1	0	0.00%	0	0.00%	0	0.00%
	2	0	0.00%	0	0.00%	0	0.00%
	3	0	0.00%	0	0.00%	0	0.00%
	4	0	0.00%	0	0.00%	0	0.00%
	Σ	1	100.00%	1	100.00%	1	100.00%

Table 13: *Cliques Distribution in* `words.dat`, *in the giant componentent and in the giant biblock (Part 2)*

4 Remarks on Higher Components of words.dat

We consider simple graphs. Vertices u and v ($u \neq v$) of such a graph are *mutually k-reachable* if there are k paths from u to v which have no internal vertices in common, i.e. which are internally disjoint. A simple graph is *k-connected* if every two distinct vertices are mutually k-reachable. A maximal k-connected subgraph of a simple graph is called a *k-component*. k-components are *uniquely determined* in the following sense: If one starts with a k-connected subgraph and adds to it vertices and lines of the graph as long as this is possible one finally reaches the same k-component independently from the order of the additions. However, distinct k-components *are not disjoint*. The may have up to $k-1$ vertices and all edges joining these in common. 1-components are connected components, 2-components are biblocks.

Each k-component ($k \geq 2$) is subgraph of a $(k-1)$-component. A $(k-1)$-component may comprise zero, one or more k-components. If there is no k-component, the decomposition into higher components stops with that $(k-1)$-component. Otherwise there may be vertices and edges in it that do not belong to any of the k-components. These generate the *$(k-1)$-cocomponent* of the $(k-1)$ component. Finding the complete higher decomposition of a graph, i.e finding the k-components for all k is a hard task. Maybe, it is not feasible in polynomial time. Deciding, whether a candidate subgraph is k-connected can be done in polynomial time with a Menger algorithm. Therefore, the complete higher decomposition can be done in polynomial time if and only if the number of k-components is polynomially bounded by the number of vertices of the graph.

Finding 1-components and 2-components is an not so difficult task. Triconnected blocks can be found with an efficient algorithm due to Hopcroft and Tarjan [HopcT1973a]. An heuristically motivated algorithm to find the complete higher decomposition of a graph was proposed by Stiege [Stie2007a]. Since this algorithm is not implemented in GHS as yet, we cannot present the complete decomposition of `words.dat`. We confine ourselves to some results which are easy to obtain.

k-components must consist of at least $(k+1)$ vertices and each vertex must be of degree at least k in the component. So a first step may be to find the subgraphs where all vertices have degree at least k starting with the highest possible k. Function `degreedelete` of GHS can be used for this purpose. There are two vertices in `ghswords` having the highest degree 25, namely `cores` and `bares`. Since at least $k+1$ vertices are needed to build a k-component, there cannot exist a 25-component in `wordsdat`. As can be seen in table 2 the largest k for which a k-component may be possible is $k = 19$. Applying `degreedelete` to eliminate recursively all vertices with degree 18 or less, an empty set of vertices remains. Going down with minimum degree required, we see that the first time a nonempty set of vertices remains is for degree 11 to be deleted. A set of 13 vertices, all of degree 12 remain. With help of table 11 it is not difficult to interpret this fact. The remaining vertices constitute clique `bills/cills`. In fact we have found the one and only 12-component of `ghswords`. Proceeding in analogous manner, we see that cliques `malls/falls` and `bears/dears` are the only final 11-components and cliques `bales/dales` and `dinks/finks` are the only final 10-components in `wordsdat`. All these cliques being red, they have no vertices in common. From degree 9 on proceeding this

way does not work anymore. We have to wait until the general high component finding algorithms are available in GHS or may try to use the clique decomposition in a more subtile way.

A Listing of the Biblock Decomposition of words.dat

```
BEGIN REDUCED STRUCTURE
$GRAPH GHSwords
$TYPE  UGSLF
$No_VERTICES              5757
$No_EDGES                14135
$No_ISOLATED_VERTICES      671
$No_ACYCLIC_COMPONENTS     145     (353V, 208E)
   $ACYCLIC_COMPONENT _abets*abuts    (2V, 1E)
   $ACYCLIC_COMPONENT _admit*admix    (2V, 1E)
   $ACYCLIC_COMPONENT _agent*anent    (2V, 1E)
   $ACYCLIC_COMPONENT _ahhhh*ohhhh    (2V, 1E)
   $ACYCLIC_COMPONENT _aisle*lisle    (2V, 1E)
   $ACYCLIC_COMPONENT _algae*algal    (2V, 1E)
   $ACYCLIC_COMPONENT _alien*align    (2V, 1E)
   $ACYCLIC_COMPONENT _alkyd*alkyl    (2V, 1E)
   $ACYCLIC_COMPONENT _aloha*alpha    (2V, 1E)
   $ACYCLIC_COMPONENT _altos*autos    (2V, 1E)
   $ACYCLIC_COMPONENT _amity*arity    (2V, 1E)
   $ACYCLIC_COMPONENT _annul*annum    (2V, 1E)
   $ACYCLIC_COMPONENT _antsy*artsy    (2V, 1E)
   $ACYCLIC_COMPONENT _aphid*aphis    (2V, 1E)
   $ACYCLIC_COMPONENT _apian*avian    (2V, 1E)
   $ACYCLIC_COMPONENT _aquae*aquas    (2V, 1E)
   $ACYCLIC_COMPONENT _audio*audit    (2V, 1E)
   $ACYCLIC_COMPONENT _aught*ought    (2V, 1E)
   $ACYCLIC_COMPONENT _avant*avast    (2V, 1E)
   $ACYCLIC_COMPONENT _avoid*ovoid    (2V, 1E)
   $ACYCLIC_COMPONENT _bairn*cairn    (2V, 1E)
   $ACYCLIC_COMPONENT _baize*maize    (2V, 1E)
   $ACYCLIC_COMPONENT _balsa*salsa    (2V, 1E)
   $ACYCLIC_COMPONENT _befog*defog    (2V, 1E)
   $ACYCLIC_COMPONENT _bhang*whang    (2V, 1E)
   $ACYCLIC_COMPONENT _blitz*glitz    (2V, 1E)
```

Table 14: *Biblock Decomposition of* words.dat *(Part 1)*

```
$ACYCLIC_COMPONENT _boule*joule    (2V, 1E)
$ACYCLIC_COMPONENT _bronc*bronx    (2V, 1E)
$ACYCLIC_COMPONENT _buena*bueno    (2V, 1E)
$ACYCLIC_COMPONENT _calix*calyx    (2V, 1E)
$ACYCLIC_COMPONENT _churl*churn    (2V, 1E)
$ACYCLIC_COMPONENT _civvy*divvy    (2V, 1E)
$ACYCLIC_COMPONENT _cobra*copra    (2V, 1E)
$ACYCLIC_COMPONENT _debar*rebar    (2V, 1E)
$ACYCLIC_COMPONENT _deice*deuce    (2V, 1E)
$ACYCLIC_COMPONENT _dicta*dictu    (2V, 1E)
$ACYCLIC_COMPONENT _donna*gonna    (2V, 1E)
$ACYCLIC_COMPONENT _droid*druid    (2V, 1E)
$ACYCLIC_COMPONENT _echos*ethos    (2V, 1E)
$ACYCLIC_COMPONENT _edict*evict    (2V, 1E)
$ACYCLIC_COMPONENT _emery*every    (2V, 1E)
$ACYCLIC_COMPONENT _empty*umpty    (2V, 1E)
$ACYCLIC_COMPONENT _excel*expel    (2V, 1E)
$ACYCLIC_COMPONENT _fiord*fjord    (2V, 1E)
$ACYCLIC_COMPONENT _fungi*fungo    (2V, 1E)
$ACYCLIC_COMPONENT _furor*juror    (2V, 1E)
$ACYCLIC_COMPONENT _gaffe*gaffs    (2V, 1E)
$ACYCLIC_COMPONENT _gamic*gamin    (2V, 1E)
$ACYCLIC_COMPONENT _gonad*monad    (2V, 1E)
$ACYCLIC_COMPONENT _hydra*hydro    (2V, 1E)
$ACYCLIC_COMPONENT _ideal*ideas    (2V, 1E)
$ACYCLIC_COMPONENT _idiom*idiot    (2V, 1E)
$ACYCLIC_COMPONENT _image*imago    (2V, 1E)
$ACYCLIC_COMPONENT _ivied*ivies    (2V, 1E)
$ACYCLIC_COMPONENT _judos*kudos    (2V, 1E)
$ACYCLIC_COMPONENT _juice*juicy    (2V, 1E)
$ACYCLIC_COMPONENT _kaiak*kayak    (2V, 1E)
$ACYCLIC_COMPONENT _keyed*keyer    (2V, 1E)
$ACYCLIC_COMPONENT _kluge*klugy    (2V, 1E)
$ACYCLIC_COMPONENT _libra*lubra    (2V, 1E)
$ACYCLIC_COMPONENT _licit*limit    (2V, 1E)
$ACYCLIC_COMPONENT _lymph*nymph    (2V, 1E)
$ACYCLIC_COMPONENT _mecum*tecum    (2V, 1E)
```

Table 15: *Biblock Decomposition of* `words.dat` *(Part 2)*

```
$ACYCLIC_COMPONENT _media*medic    (2V, 1E)
$ACYCLIC_COMPONENT _motor*rotor    (2V, 1E)
$ACYCLIC_COMPONENT _naive*waive    (2V, 1E)
$ACYCLIC_COMPONENT _newly*newsy    (2V, 1E)
$ACYCLIC_COMPONENT _niece*piece    (2V, 1E)
$ACYCLIC_COMPONENT _nitro*vitro    (2V, 1E)
$ACYCLIC_COMPONENT _novae*novas    (2V, 1E)
$ACYCLIC_COMPONENT _nylon*pylon    (2V, 1E)
$ACYCLIC_COMPONENT _odium*opium    (2V, 1E)
$ACYCLIC_COMPONENT _oleos*olios    (2V, 1E)
$ACYCLIC_COMPONENT _ousel*ouzel    (2V, 1E)
$ACYCLIC_COMPONENT _outdo*outgo    (2V, 1E)
$ACYCLIC_COMPONENT _owned*owner    (2V, 1E)
$ACYCLIC_COMPONENT _peeve*reeve    (2V, 1E)
$ACYCLIC_COMPONENT _petit*pewit    (2V, 1E)
$ACYCLIC_COMPONENT _pieta*piety    (2V, 1E)
$ACYCLIC_COMPONENT _pilaf*pilau    (2V, 1E)
$ACYCLIC_COMPONENT _quaff*quiff    (2V, 1E)
$ACYCLIC_COMPONENT _quash*quasi    (2V, 1E)
$ACYCLIC_COMPONENT _queen*queer    (2V, 1E)
$ACYCLIC_COMPONENT _quoin*quoit    (2V, 1E)
$ACYCLIC_COMPONENT _radon*rayon    (2V, 1E)
$ACYCLIC_COMPONENT _rajah*rajas    (2V, 1E)
$ACYCLIC_COMPONENT _recta*recto    (2V, 1E)
$ACYCLIC_COMPONENT _ridge*ridgy    (2V, 1E)
$ACYCLIC_COMPONENT _sirup*syrup    (2V, 1E)
$ACYCLIC_COMPONENT _spume*spumy    (2V, 1E)
$ACYCLIC_COMPONENT _suede*swede    (2V, 1E)
$ACYCLIC_COMPONENT _supra*sutra    (2V, 1E)
$ACYCLIC_COMPONENT _techs*techy    (2V, 1E)
$ACYCLIC_COMPONENT _torsi*torso    (2V, 1E)
$ACYCLIC_COMPONENT _twist*twixt    (2V, 1E)
$ACYCLIC_COMPONENT _ulnar*ulnas    (2V, 1E)
$ACYCLIC_COMPONENT _unfed*unwed    (2V, 1E)
$ACYCLIC_COMPONENT _untie*until    (2V, 1E)
$ACYCLIC_COMPONENT _usurp*usury    (2V, 1E)
$ACYCLIC_COMPONENT _uteri*utero    (2V, 1E)
```

Table 16: *Biblock Decomposition of* `words.dat` *(Part 3)*

```
$ACYCLIC_COMPONENT _vacua*vacuo    (2V, 1E)
$ACYCLIC_COMPONENT _valor*vapor    (2V, 1E)
$ACYCLIC_COMPONENT _wassa*watsa    (2V, 1E)
$ACYCLIC_COMPONENT _ahoys*bhoys    (3V, 2E)
$ACYCLIC_COMPONENT _alike*alive    (3V, 2E)
$ACYCLIC_COMPONENT _amahs*amass    (3V, 2E)
$ACYCLIC_COMPONENT _argon*argot    (3V, 2E)
$ACYCLIC_COMPONENT _assai*assay    (3V, 2E)
$ACYCLIC_COMPONENT _bassi*basso    (3V, 2E)
$ACYCLIC_COMPONENT _bocce*bocci    (3V, 2E)
$ACYCLIC_COMPONENT _cavil*civil    (3V, 2E)
$ACYCLIC_COMPONENT _chain*chair    (3V, 2E)
$ACYCLIC_COMPONENT _deify*deity    (3V, 2E)
$ACYCLIC_COMPONENT _digit*dixit    (3V, 2E)
$ACYCLIC_COMPONENT _duple*tuple    (3V, 2E)
$ACYCLIC_COMPONENT _edema*enema    (3V, 2E)
$ACYCLIC_COMPONENT _endue*ensue    (3V, 2E)
$ACYCLIC_COMPONENT _enjoy*envoy    (3V, 2E)
$ACYCLIC_COMPONENT _exeat*exert    (3V, 2E)
$ACYCLIC_COMPONENT _femme*lemme    (3V, 2E)
$ACYCLIC_COMPONENT _fiche*fichu    (3V, 2E)
$ACYCLIC_COMPONENT _genie*genii    (3V, 2E)
$ACYCLIC_COMPONENT _gismo*gizmo    (3V, 2E)
$ACYCLIC_COMPONENT _idols*idyls    (3V, 2E)
$ACYCLIC_COMPONENT _ileum*ileus    (3V, 2E)
$ACYCLIC_COMPONENT _infra*intra    (3V, 2E)
$ACYCLIC_COMPONENT _labor*tabor    (3V, 2E)
$ACYCLIC_COMPONENT _lepta*septa    (3V, 2E)
$ACYCLIC_COMPONENT _nixie*pixie    (3V, 2E)
$ACYCLIC_COMPONENT _rebox*redox    (3V, 2E)
$ACYCLIC_COMPONENT _resin*rosin    (3V, 2E)
$ACYCLIC_COMPONENT _tenon*tenor    (3V, 2E)
$ACYCLIC_COMPONENT _tonal*total    (3V, 2E)
$ACYCLIC_COMPONENT _aerie*eerie    (4V, 3E)
$ACYCLIC_COMPONENT _amped*umped    (4V, 3E)
$ACYCLIC_COMPONENT _dance*dunce    (4V, 3E)
$ACYCLIC_COMPONENT _felon*melon    (4V, 3E)
```

Table 17: *Biblock Decomposition of* `words.dat` *(Part 4)*

```
   $ACYCLIC_COMPONENT _inapt*inept    (4V, 3E)
   $ACYCLIC_COMPONENT _kabob*kebob    (4V, 3E)
   $ACYCLIC_COMPONENT _kazoo*wazoo    (4V, 3E)
   $ACYCLIC_COMPONENT _angel*anger    (5V, 4E)
   $ACYCLIC_COMPONENT _macho*macro    (5V, 4E)
   $ACYCLIC_COMPONENT _logic*login    (6V, 5E)
   $ACYCLIC_COMPONENT _unban*unbar    (6V, 5E)
   $ACYCLIC_COMPONENT _bogie*bowie    (7V, 6E)
$No_CYCLIC_COMPONENTS       37     (4733V, 13927E)
   $CYCLIC_COMPONENT _adapt*adept     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
   $No_PERIPHERAL_TREES 0
        $STOPFREE_KERNEL _adapt*adept     (3V, 3E)
        $No_SUBCOMPONENTS 1
            $SUBCOMPONENT _adapt*adept        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
            $No_BIBLOCKS 1
                $BIBLOCK _adapt*adept             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
        $No_INTERNAL_TREES 0
   $CYCLIC_COMPONENT _arbor*ardor     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
   $No_PERIPHERAL_TREES 0
        $STOPFREE_KERNEL _arbor*ardor     (3V, 3E)
        $No_SUBCOMPONENTS 1
            $SUBCOMPONENT _arbor*ardor        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
            $No_BIBLOCKS 1
                $BIBLOCK _arbor*ardor             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
        $No_INTERNAL_TREES 0
   $CYCLIC_COMPONENT _condo*mondo     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
   $No_PERIPHERAL_TREES 0
        $STOPFREE_KERNEL _condo*mondo     (3V, 3E)
        $No_SUBCOMPONENTS 1
            $SUBCOMPONENT _condo*mondo        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
            $No_BIBLOCKS 1
                $BIBLOCK _condo*mondo             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
        $No_INTERNAL_TREES 0
```

Table 18: *Biblockd Decomposition of* `words.dat` *(Part 5)*

```
$CYCLIC_COMPONENT _deign*feign     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _deign*feign     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _deign*feign        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _deign*feign             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _demur*femur     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _demur*femur     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _demur*femur        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _demur*femur             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _getup*letup     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _getup*letup     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _getup*letup        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _getup*letup             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _heigh*neigh     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _heigh*neigh     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _heigh*neigh        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _heigh*neigh             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
```

Table 19: *Biblockd Decomposition of* `words.dat` *(Part 6)*

```
$CYCLIC_COMPONENT _opals*orals     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _opals*orals     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _opals*orals        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _opals*orals             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _paean*pagan     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _paean*pagan     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _paean*pagan        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _paean*pagan             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _unarc*unarm     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _unarc*unarm     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _unarc*unarm        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _unarc*unarm             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _urged*urger     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _urged*urger     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _urged*urger        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _urged*urger             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
```

Table 20: *Biblockd Decomposition of* `words.dat` *(Part 7)*

```
$CYCLIC_COMPONENT _whelk*whelm     (3V, 3E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _whelk*whelm     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _whelk*whelm        (3V, 3E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _whelk*whelm             (3V, 3E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _anion*onion     (4V, 4E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _union*unwon     (2V, 1E)
     $STOPFREE_KERNEL _anion*onion     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _anion*onion        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _anion*onion             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _field*wield     (4V, 4E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _field*fiend     (2V, 1E)
     $STOPFREE_KERNEL _field*wield     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _field*wield        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _field*wield             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _gumbo*jumbo     (4V, 4E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _mambo*mumbo     (2V, 1E)
     $STOPFREE_KERNEL _gumbo*jumbo     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _gumbo*jumbo        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _gumbo*jumbo             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
```

Table 21: *Biblockd Decomposition of* `words.dat` *(Part 8)*

```
$CYCLIC_COMPONENT _humor*rumor     (4V, 4E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _tumor*tutor     (2V, 1E)
     $STOPFREE_KERNEL _humor*rumor     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _humor*rumor        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _humor*rumor             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _vigor*visor     (4V, 4E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _rigor*vigor     (2V, 1E)
     $STOPFREE_KERNEL _vigor*visor     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _vigor*visor        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _vigor*visor             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _adman*admen     (4V, 4E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _adman*admen     (4V, 4E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _adman*admen        (4V, 4E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _adman*admen             (4V, 4E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _dizzy*fizzy     (5V, 5E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _buzzy*fuzzy     (3V, 2E)
     $STOPFREE_KERNEL _dizzy*fizzy     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _dizzy*fizzy        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _dizzy*fizzy             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
```

Table 22: *Biblockd Decomposition of* `words.dat` *(Part 9)*

```
$CYCLIC_COMPONENT _eject*elect     (5V, 5E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _erect*eruct     (3V, 2E)
     $STOPFREE_KERNEL _eject*elect     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _eject*elect        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _eject*elect             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _enact*epact     (5V, 5E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _exact*exalt     (3V, 2E)
     $STOPFREE_KERNEL _enact*epact     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _enact*epact        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _enact*epact             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _radii*radio     (5V, 5E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _patio*ratio     (3V, 2E)
     $STOPFREE_KERNEL _radii*radio     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _radii*radio        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _radii*radio             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _edits*emits     (6V, 6E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _emirs*emits     (4V, 3E)
     $STOPFREE_KERNEL _edits*emits     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _edits*emits        (3V, 3E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _edits*emits             (3V, 3E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
```

Table 23: *Biblockd Decomposition of* `words.dat` *(Part 10)*

```
$CYCLIC_COMPONENT _squab*squad     (6V, 9E, 0AP, 0BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 0
     $STOPFREE_KERNEL _squab*squad     (6V, 9E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _squab*squad        (6V, 9E, 0AP, 0BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _squab*squad             (6V, 9E, 0AP, 0BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _amber*ember     (7V, 7E, 2AP, 2BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 2
     $PERIPHERAL_TREE _umbel*umber     (2V, 1E)
     $PERIPHERAL_TREE _ebbed*embed     (4V, 3E)
     $STOPFREE_KERNEL _amber*ember     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _amber*ember        (3V, 3E, 2AP, 2BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _amber*ember             (3V, 3E, 2AP, 2BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _kneed*kneel     (7V, 7E, 2AP, 2BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 2
     $PERIPHERAL_TREE _knead*kneed     (2V, 1E)
     $PERIPHERAL_TREE _kneel*knell     (4V, 3E)
     $STOPFREE_KERNEL _kneed*kneel     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _kneed*kneel        (3V, 3E, 2AP, 2BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _kneed*kneel             (3V, 3E, 2AP, 2BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
```

Table 24: *Biblockd Decomposition of* `words.dat` *(Part 11)*

```
$CYCLIC_COMPONENT _unfit*unhit     (7V, 7E, 2AP, 2BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 2
     $PERIPHERAL_TREE _infix*unfix     (3V, 2E)
     $PERIPHERAL_TREE _unhip*unhit     (3V, 2E)
     $STOPFREE_KERNEL _unfit*unhit     (3V, 3E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _unfit*unhit        (3V, 3E, 2AP, 2BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _unfit*unhit             (3V, 3E, 2AP, 2BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _ample*amply     (7V, 7E, 3AP, 3BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 3
     $PERIPHERAL_TREE _amble*ample     (2V, 1E)
     $PERIPHERAL_TREE _amply*imply     (2V, 1E)
     $PERIPHERAL_TREE _apply*aptly     (2V, 1E)
     $STOPFREE_KERNEL _ample*amply     (4V, 4E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _ample*amply        (4V, 4E, 3AP, 3BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _ample*amply             (4V, 4E, 3AP, 3BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _cable*fable     (7V, 12E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _sable*sabre     (3V, 2E)
     $STOPFREE_KERNEL _cable*fable     (5V, 10E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _cable*fable        (5V, 10E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _cable*fable             (5V, 10E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
```

Table 25: *Biblockd Decomposition of* `words.dat` *(Part 12)*

```
$CYCLIC_COMPONENT _allot*allow     (8V, 9E, 4AP, 3BP, 0CP, 1HP)
$No_PERIPHERAL_TREES 3
     $PERIPHERAL_TREE _agley*alley     (2V, 1E)
     $PERIPHERAL_TREE _aglow*allow     (2V, 1E)
     $PERIPHERAL_TREE _allay*alway     (2V, 1E)
     $STOPFREE_KERNEL _allot*allow     (5V, 6E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _allot*allow        (5V, 6E, 4AP, 3BP, 0CP, 1HP)
         $No_BIBLOCKS 2
             $BIBLOCK _allay*alley             (3V, 3E, 3AP, 2BP, 0CP, 1HP)
             $BIBLOCK _allot*allow             (3V, 3E, 2AP, 1BP, 0CP, 1HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _refit*refix     (15V, 15E, 2AP, 2BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 2
     $PERIPHERAL_TREE _befit*refit     (2V, 1E)
     $PERIPHERAL_TREE _bebug*debug     (11V, 10E)
     $STOPFREE_KERNEL _refit*refix     (4V, 4E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _refit*refix        (4V, 4E, 2AP, 2BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _refit*refix             (4V, 4E, 2AP, 2BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _avers*avert     (15V, 16E, 5AP, 4BP, 2CP, 0HP)
$No_PERIPHERAL_TREES 4
     $PERIPHERAL_TREE _alert*avert     (2V, 1E)
     $PERIPHERAL_TREE _apers*avers     (2V, 1E)
     $PERIPHERAL_TREE _evens*event     (3V, 2E)
     $PERIPHERAL_TREE _abend*amend     (5V, 4E)
     $STOPFREE_KERNEL _avers*avert     (7V, 8E)
     $No_SUBCOMPONENTS 2
         $SUBCOMPONENT _omens*opens        (3V, 3E, 2AP, 2BP, 1CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _omens*opens             (3V, 3E, 2AP, 2BP, 1CP, 0HP)
         $SUBCOMPONENT _avers*avert        (4V, 4E, 3AP, 2BP, 1CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _avers*avert             (4V, 4E, 3AP, 2BP, 1CP, 0HP)
     $No_INTERNAL_TREES 1
         $INTERNAL_TREE _ovens*overs       (2V, 1E, 2AP, 1BP, 2CP, 0HP)
```

Table 26: *Biblockd Decomposition of* `words.dat` *(Part 13)*

```
$CYCLIC_COMPONENT _unsee*unset     (15V, 19E, 4AP, 2BP, 2CP, 1HP)
$No_PERIPHERAL_TREES 2
     $PERIPHERAL_TREE _oncet*onset     (2V, 1E)
     $PERIPHERAL_TREE _unmet*unset     (3V, 2E)
     $STOPFREE_KERNEL _unsee*unset     (12V, 16E)
     $No_SUBCOMPONENTS 2
         $SUBCOMPONENT _idled*idler        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _idled*idler             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
         $SUBCOMPONENT _unsee*unset        (6V, 9E, 3AP, 2BP, 1CP, 1HP)
         $No_BIBLOCKS 2
             $BIBLOCK _inset*onset             (3V, 3E, 3AP, 2BP, 1CP, 1HP)
             $BIBLOCK _unsee*unset             (4V, 6E, 1AP, 1BP, 0CP, 1HP)
     $No_INTERNAL_TREES 1
         $INTERNAL_TREE _idles*isles       (5V, 4E, 2AP, 0BP, 2CP, 0HP)
$CYCLIC_COMPONENT _bound*found     (17V, 42E, 2AP, 2BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 2
     $PERIPHERAL_TREE _count*court     (2V, 1E)
     $PERIPHERAL_TREE _world*would     (2V, 1E)
     $STOPFREE_KERNEL _bound*found     (15V, 40E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _bound*found        (15V, 40E, 2AP, 2BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _bound*found             (15V, 40E, 2AP, 2BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _dicot*didot     (19V, 22E, 3AP, 2BP, 2CP, 0HP)
$No_PERIPHERAL_TREES 2
     $PERIPHERAL_TREE _dicot*dicut     (2V, 1E)
     $PERIPHERAL_TREE _didot*didst     (10V, 9E)
     $STOPFREE_KERNEL _dicot*didot     (9V, 12E)
     $No_SUBCOMPONENTS 2
         $SUBCOMPONENT _dados*didos        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _dados*didos             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
         $SUBCOMPONENT _dicot*didot        (6V, 8E, 2AP, 2BP, 1CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _dicot*didot             (6V, 8E, 2AP, 2BP, 1CP, 0HP)
     $No_INTERNAL_TREES 1
         $INTERNAL_TREE _didos*didot       (2V, 1E, 2AP, 1BP, 2CP, 0HP)
```

Table 27: *Biblockd Decomposition of* `words.dat` *(Part 14)*

```
$CYCLIC_COMPONENT _biffs*biffy     (24V, 50E, 1AP, 1BP, 0CP, 0HP)
$No_PERIPHERAL_TREES 1
     $PERIPHERAL_TREE _daffy*taffy     (4V, 3E)
     $STOPFREE_KERNEL _biffs*biffy     (21V, 47E)
     $No_SUBCOMPONENTS 1
         $SUBCOMPONENT _biffs*biffy        (21V, 47E, 1AP, 1BP, 0CP, 0HP)
         $No_BIBLOCKS 1
             $BIBLOCK _biffs*biffy             (21V, 47E, 1AP, 1BP, 0CP, 0HP)
     $No_INTERNAL_TREES 0
$CYCLIC_COMPONENT _ached*aches     (4493V, 13619E, 461AP, 393BP, 56CP, 23HP)
$No_PERIPHERAL_TREES 393
     $PERIPHERAL_TREE _abled*abler     (2V, 1E)
     $PERIPHERAL_TREE _abode*above     (2V, 1E)
     $PERIPHERAL_TREE _acing*icing     (2V, 1E)
     $PERIPHERAL_TREE _agile*anile     (2V, 1E)
     $PERIPHERAL_TREE _agone*agony     (2V, 1E)
     $PERIPHERAL_TREE _alias*arias     (2V, 1E)
     $PERIPHERAL_TREE _anode*inode     (2V, 1E)
     $PERIPHERAL_TREE _apace*space     (2V, 1E)
     $PERIPHERAL_TREE _apses*apsos     (2V, 1E)
     $PERIPHERAL_TREE _arras*array     (2V, 1E)
     $PERIPHERAL_TREE _aster*astir     (2V, 1E)
     $PERIPHERAL_TREE _atilt*stilt     (2V, 1E)
     $PERIPHERAL_TREE _aways*sways     (2V, 1E)
     $PERIPHERAL_TREE _axled*axles     (2V, 1E)
     $PERIPHERAL_TREE _baggy*jaggy     (2V, 1E)
     $PERIPHERAL_TREE _bahts*baits     (2V, 1E)
     $PERIPHERAL_TREE _barbs*garbs     (2V, 1E)
     $PERIPHERAL_TREE _beano*beans     (2V, 1E)
     $PERIPHERAL_TREE _began*vegan     (2V, 1E)
     $PERIPHERAL_TREE _being*bring     (2V, 1E)
     $PERIPHERAL_TREE _belie*belle     (2V, 1E)
     $PERIPHERAL_TREE _berth*birth     (2V, 1E)
     $PERIPHERAL_TREE _biped*bipod     (2V, 1E)
     $PERIPHERAL_TREE _bitsy*bitty     (2V, 1E)
     $PERIPHERAL_TREE _bogus*bonus     (2V, 1E)
     $PERIPHERAL_TREE _bolos*bozos     (2V, 1E)
```

Table 28: *Biblockd Decomposition of* `words.dat` *(Part 15)*

```
$PERIPHERAL_TREE _bombe*bombs     (2V, 1E)
$PERIPHERAL_TREE _booze*boozy     (2V, 1E)
$PERIPHERAL_TREE _brave*breve     (2V, 1E)
$PERIPHERAL_TREE _brief*grief     (2V, 1E)
$PERIPHERAL_TREE _brine*urine     (2V, 1E)
$PERIPHERAL_TREE _brisk*frisk     (2V, 1E)
$PERIPHERAL_TREE _bromo*promo     (2V, 1E)
$PERIPHERAL_TREE _bruit*fruit     (2V, 1E)
$PERIPHERAL_TREE _bunco*junco     (2V, 1E)
$PERIPHERAL_TREE _butte*butts     (2V, 1E)
$PERIPHERAL_TREE _calla*calls     (2V, 1E)
$PERIPHERAL_TREE _canoe*canon     (2V, 1E)
$PERIPHERAL_TREE _cedar*ceder     (2V, 1E)
$PERIPHERAL_TREE _cento*lento     (2V, 1E)
$PERIPHERAL_TREE _chant*chart     (2V, 1E)
$PERIPHERAL_TREE _chews*chewy     (2V, 1E)
$PERIPHERAL_TREE _chimp*chirp     (2V, 1E)
$PERIPHERAL_TREE _chino*rhino     (2V, 1E)
$PERIPHERAL_TREE _chute*shute     (2V, 1E)
$PERIPHERAL_TREE _climb*clime     (2V, 1E)
$PERIPHERAL_TREE _coati*coats     (2V, 1E)
$PERIPHERAL_TREE _codon*colon     (2V, 1E)
$PERIPHERAL_TREE _combo*combs     (2V, 1E)
$PERIPHERAL_TREE _comma*momma     (2V, 1E)
$PERIPHERAL_TREE _conic*cynic     (2V, 1E)
$PERIPHERAL_TREE _coral*moral     (2V, 1E)
$PERIPHERAL_TREE _coupe*coups     (2V, 1E)
$PERIPHERAL_TREE _cowry*dowry     (2V, 1E)
$PERIPHERAL_TREE _craze*crazy     (2V, 1E)
$PERIPHERAL_TREE _crept*crypt     (2V, 1E)
$PERIPHERAL_TREE _crimp*crisp     (2V, 1E)
$PERIPHERAL_TREE _crude*cruds     (2V, 1E)
$PERIPHERAL_TREE _cruel*gruel     (2V, 1E)
$PERIPHERAL_TREE _crumb*crump     (2V, 1E)
$PERIPHERAL_TREE _curie*cutie     (2V, 1E)
$PERIPHERAL_TREE _cusps*cuspy     (2V, 1E)
$PERIPHERAL_TREE _daubs*drubs     (2V, 1E)
```

Table 29: *Biblockd Decomposition of* `words.dat` *(Part 16)*

```
$PERIPHERAL_TREE _deads*dyads     (2V, 1E)
$PERIPHERAL_TREE _deals*dealt     (2V, 1E)
$PERIPHERAL_TREE _deary*diary     (2V, 1E)
$PERIPHERAL_TREE _death*depth     (2V, 1E)
$PERIPHERAL_TREE _debts*dents     (2V, 1E)
$PERIPHERAL_TREE _decor*decoy     (2V, 1E)
$PERIPHERAL_TREE _deism*deist     (2V, 1E)
$PERIPHERAL_TREE _delis*dells     (2V, 1E)
$PERIPHERAL_TREE _dewed*dewey     (2V, 1E)
$PERIPHERAL_TREE _dilly*dimly     (2V, 1E)
$PERIPHERAL_TREE _dinar*diner     (2V, 1E)
$PERIPHERAL_TREE _ditto*ditty     (2V, 1E)
$PERIPHERAL_TREE _divan*divas     (2V, 1E)
$PERIPHERAL_TREE _doggo*doggy     (2V, 1E)
$PERIPHERAL_TREE _douse*dowse     (2V, 1E)
$PERIPHERAL_TREE _doyen*dozen     (2V, 1E)
$PERIPHERAL_TREE _drama*drams     (2V, 1E)
$PERIPHERAL_TREE _dread*dryad     (2V, 1E)
$PERIPHERAL_TREE _dreck*wreck     (2V, 1E)
$PERIPHERAL_TREE _drive*drove     (2V, 1E)
$PERIPHERAL_TREE _duchy*ducky     (2V, 1E)
$PERIPHERAL_TREE _dwarf*swarf     (2V, 1E)
$PERIPHERAL_TREE _eared*erred     (2V, 1E)
$PERIPHERAL_TREE _earls*early     (2V, 1E)
$PERIPHERAL_TREE _eaves*elves     (2V, 1E)
$PERIPHERAL_TREE _eking*eying     (2V, 1E)
$PERIPHERAL_TREE _elope*slope     (2V, 1E)
$PERIPHERAL_TREE _emote*smote     (2V, 1E)
$PERIPHERAL_TREE _epics*spics     (2V, 1E)
$PERIPHERAL_TREE _facto*facts     (2V, 1E)
$PERIPHERAL_TREE _faker*fakir     (2V, 1E)
$PERIPHERAL_TREE _fancy*fanny     (2V, 1E)
$PERIPHERAL_TREE _farad*fared     (2V, 1E)
$PERIPHERAL_TREE _farce*force     (2V, 1E)
$PERIPHERAL_TREE _fetes*fetus     (2V, 1E)
$PERIPHERAL_TREE _finif*finis     (2V, 1E)
$PERIPHERAL_TREE _flame*frame     (2V, 1E)
```

Table 30: *Biblockd Decomposition of* `words.dat` *(Part 17)*

```
$PERIPHERAL_TREE _forge*forgo     (2V, 1E)
$PERIPHERAL_TREE _franc*frank     (2V, 1E)
$PERIPHERAL_TREE _galls*gaols     (2V, 1E)
$PERIPHERAL_TREE _giant*grant     (2V, 1E)
$PERIPHERAL_TREE _girls*girly     (2V, 1E)
$PERIPHERAL_TREE _glebe*grebe     (2V, 1E)
$PERIPHERAL_TREE _godly*golly     (2V, 1E)
$PERIPHERAL_TREE _gofer*goner     (2V, 1E)
$PERIPHERAL_TREE _goody*grody     (2V, 1E)
$PERIPHERAL_TREE _gouts*gouty     (2V, 1E)
$PERIPHERAL_TREE _grasp*grass     (2V, 1E)
$PERIPHERAL_TREE _grata*grate     (2V, 1E)
$PERIPHERAL_TREE _green*preen     (2V, 1E)
$PERIPHERAL_TREE _grime*grimy     (2V, 1E)
$PERIPHERAL_TREE _gronk*grook     (2V, 1E)
$PERIPHERAL_TREE _gutta*outta     (2V, 1E)
$PERIPHERAL_TREE _hafta*hafts     (2V, 1E)
$PERIPHERAL_TREE _hanky*lanky     (2V, 1E)
$PERIPHERAL_TREE _haply*happy     (2V, 1E)
$PERIPHERAL_TREE _harem*harum     (2V, 1E)
$PERIPHERAL_TREE _heard*heerd     (2V, 1E)
$PERIPHERAL_TREE _hexad*hexed     (2V, 1E)
$PERIPHERAL_TREE _hippo*hippy     (2V, 1E)
$PERIPHERAL_TREE _hoagy*hoary     (2V, 1E)
$PERIPHERAL_TREE _holly*hotly     (2V, 1E)
$PERIPHERAL_TREE _horde*horse     (2V, 1E)
$PERIPHERAL_TREE _houri*hours     (2V, 1E)
$PERIPHERAL_TREE _humps*humus     (2V, 1E)
$PERIPHERAL_TREE _hurts*yurts     (2V, 1E)
$PERIPHERAL_TREE _iodic*ionic     (2V, 1E)
$PERIPHERAL_TREE _items*stems     (2V, 1E)
$PERIPHERAL_TREE _jammy*jimmy     (2V, 1E)
$PERIPHERAL_TREE _jetty*petty     (2V, 1E)
$PERIPHERAL_TREE _johns*joins     (2V, 1E)
$PERIPHERAL_TREE _kinda*kinds     (2V, 1E)
$PERIPHERAL_TREE _knout*snout     (2V, 1E)
$PERIPHERAL_TREE _known*knows     (2V, 1E)
```

Table 31: *Biblockd Decomposition of* `words.dat` *(Part 18)*

```
$PERIPHERAL_TREE _krona*krone     (2V, 1E)
$PERIPHERAL_TREE _laird*lairs     (2V, 1E)
$PERIPHERAL_TREE _lamas*mamas     (2V, 1E)
$PERIPHERAL_TREE _later*latex     (2V, 1E)
$PERIPHERAL_TREE _laude*lauds     (2V, 1E)
$PERIPHERAL_TREE _lawny*lawzy     (2V, 1E)
$PERIPHERAL_TREE _legos*logos     (2V, 1E)
$PERIPHERAL_TREE _liars*liers     (2V, 1E)
$PERIPHERAL_TREE _liens*miens     (2V, 1E)
$PERIPHERAL_TREE _lithe*litho     (2V, 1E)
$PERIPHERAL_TREE _lotsa*lotta     (2V, 1E)
$PERIPHERAL_TREE _louis*louts     (2V, 1E)
$PERIPHERAL_TREE _manly*wanly     (2V, 1E)
$PERIPHERAL_TREE _maria*varia     (2V, 1E)
$PERIPHERAL_TREE _mercy*merry     (2V, 1E)
$PERIPHERAL_TREE _modal*nodal     (2V, 1E)
$PERIPHERAL_TREE _modes*modus     (2V, 1E)
$PERIPHERAL_TREE _molar*mylar     (2V, 1E)
$PERIPHERAL_TREE _moons*muons     (2V, 1E)
$PERIPHERAL_TREE _mucks*mucus     (2V, 1E)
$PERIPHERAL_TREE _mules*muley     (2V, 1E)
$PERIPHERAL_TREE _narco*narcs     (2V, 1E)
$PERIPHERAL_TREE _neath*neato     (2V, 1E)
$PERIPHERAL_TREE _noons*nouns     (2V, 1E)
$PERIPHERAL_TREE _nudge*nudie     (2V, 1E)
$PERIPHERAL_TREE _ogles*ogres     (2V, 1E)
$PERIPHERAL_TREE _olden*older     (2V, 1E)
$PERIPHERAL_TREE _opine*spine     (2V, 1E)
$PERIPHERAL_TREE _orate*ovate     (2V, 1E)
$PERIPHERAL_TREE _parka*parks     (2V, 1E)
$PERIPHERAL_TREE _pasha*pasta     (2V, 1E)
$PERIPHERAL_TREE _pearl*pears     (2V, 1E)
$PERIPHERAL_TREE _phage*phase     (2V, 1E)
$PERIPHERAL_TREE _pions*pious     (2V, 1E)
$PERIPHERAL_TREE _piths*pithy     (2V, 1E)
$PERIPHERAL_TREE _plain*plein     (2V, 1E)
$PERIPHERAL_TREE _plash*plasm     (2V, 1E)
```

Table 32: *Biblockd Decomposition of* `words.dat` *(Part 19)*

```
$PERIPHERAL_TREE _poems*poets     (2V, 1E)
$PERIPHERAL_TREE _poset*posit     (2V, 1E)
$PERIPHERAL_TREE _prier*prior     (2V, 1E)
$PERIPHERAL_TREE _prise*prism     (2V, 1E)
$PERIPHERAL_TREE _pupal*pupil     (2V, 1E)
$PERIPHERAL_TREE _quail*quais     (2V, 1E)
$PERIPHERAL_TREE _qualm*quals     (2V, 1E)
$PERIPHERAL_TREE _quips*quipu     (2V, 1E)
$PERIPHERAL_TREE _rains*rainy     (2V, 1E)
$PERIPHERAL_TREE _rally*rawly     (2V, 1E)
$PERIPHERAL_TREE _razer*razor     (2V, 1E)
$PERIPHERAL_TREE _reach*react     (2V, 1E)
$PERIPHERAL_TREE _relax*relay     (2V, 1E)
$PERIPHERAL_TREE _renew*resew     (2V, 1E)
$PERIPHERAL_TREE _rente*rents     (2V, 1E)
$PERIPHERAL_TREE _repro*retro     (2V, 1E)
$PERIPHERAL_TREE _reuse*rouse     (2V, 1E)
$PERIPHERAL_TREE _rifle*rille     (2V, 1E)
$PERIPHERAL_TREE _risks*risky     (2V, 1E)
$PERIPHERAL_TREE _roids*voids     (2V, 1E)
$PERIPHERAL_TREE _sagas*sages     (2V, 1E)
$PERIPHERAL_TREE _scald*scold     (2V, 1E)
$PERIPHERAL_TREE _scoff*scuff     (2V, 1E)
$PERIPHERAL_TREE _scour*scout     (2V, 1E)
$PERIPHERAL_TREE _scowl*scows     (2V, 1E)
$PERIPHERAL_TREE _scull*skull     (2V, 1E)
$PERIPHERAL_TREE _scurf*smurf     (2V, 1E)
$PERIPHERAL_TREE _scuse*souse     (2V, 1E)
$PERIPHERAL_TREE _seats*sects     (2V, 1E)
$PERIPHERAL_TREE _seine*seize     (2V, 1E)
$PERIPHERAL_TREE _serve*servo     (2V, 1E)
$PERIPHERAL_TREE _shard*sherd     (2V, 1E)
$PERIPHERAL_TREE _sheep*shlep     (2V, 1E)
$PERIPHERAL_TREE _shift*swift     (2V, 1E)
$PERIPHERAL_TREE _shirk*smirk     (2V, 1E)
$PERIPHERAL_TREE _shorn*thorn     (2V, 1E)
$PERIPHERAL_TREE _sines*sinew     (2V, 1E)
```

Table 33: *Biblockd Decomposition of* `words.dat` *(Part 20)*

```
$PERIPHERAL_TREE _skied*skyed     (2V, 1E)
$PERIPHERAL_TREE _skimp*skims     (2V, 1E)
$PERIPHERAL_TREE _slogs*smogs     (2V, 1E)
$PERIPHERAL_TREE _smoke*smoky     (2V, 1E)
$PERIPHERAL_TREE _solar*sonar     (2V, 1E)
$PERIPHERAL_TREE _soled*solid     (2V, 1E)
$PERIPHERAL_TREE _sonly*sonny     (2V, 1E)
$PERIPHERAL_TREE _spend*upend     (2V, 1E)
$PERIPHERAL_TREE _spoil*spool     (2V, 1E)
$PERIPHERAL_TREE _staph*stash     (2V, 1E)
$PERIPHERAL_TREE _stela*stele     (2V, 1E)
$PERIPHERAL_TREE _sties*styes     (2V, 1E)
$PERIPHERAL_TREE _stoma*stomp     (2V, 1E)
$PERIPHERAL_TREE _studs*study     (2V, 1E)
$PERIPHERAL_TREE _style*styli     (2V, 1E)
$PERIPHERAL_TREE _suets*suety     (2V, 1E)
$PERIPHERAL_TREE _swami*swamp     (2V, 1E)
$PERIPHERAL_TREE _tabus*talus     (2V, 1E)
$PERIPHERAL_TREE _taels*tails     (2V, 1E)
$PERIPHERAL_TREE _taros*tarot     (2V, 1E)
$PERIPHERAL_TREE _terce*terse     (2V, 1E)
$PERIPHERAL_TREE _texas*texts     (2V, 1E)
$PERIPHERAL_TREE _there*therm     (2V, 1E)
$PERIPHERAL_TREE _thing*thong     (2V, 1E)
$PERIPHERAL_TREE _tikes*tikis     (2V, 1E)
$PERIPHERAL_TREE _timed*timid     (2V, 1E)
$PERIPHERAL_TREE _topoi*topos     (2V, 1E)
$PERIPHERAL_TREE _torah*torch     (2V, 1E)
$PERIPHERAL_TREE _torts*torus     (2V, 1E)
$PERIPHERAL_TREE _toxic*toxin     (2V, 1E)
$PERIPHERAL_TREE _troth*truth     (2V, 1E)
$PERIPHERAL_TREE _trust*tryst     (2V, 1E)
$PERIPHERAL_TREE _tubal*tubas     (2V, 1E)
$PERIPHERAL_TREE _unate*unite     (2V, 1E)
$PERIPHERAL_TREE _unify*unity     (2V, 1E)
$PERIPHERAL_TREE _vales*valet     (2V, 1E)
$PERIPHERAL_TREE _veins*veiny     (2V, 1E)
```

Table 34: *Biblockd Decomposition of* `words.dat` *(Part 21)*

```
$PERIPHERAL_TREE _velds*veldt     (2V, 1E)
$PERIPHERAL_TREE _viers*views     (2V, 1E)
$PERIPHERAL_TREE _viral*vital     (2V, 1E)
$PERIPHERAL_TREE _vocab*vocal     (2V, 1E)
$PERIPHERAL_TREE _vowel*voxel     (2V, 1E)
$PERIPHERAL_TREE _wades*wadis     (2V, 1E)
$PERIPHERAL_TREE _weald*weals     (2V, 1E)
$PERIPHERAL_TREE _weird*weirs     (2V, 1E)
$PERIPHERAL_TREE _welch*welsh     (2V, 1E)
$PERIPHERAL_TREE _whens*wrens     (2V, 1E)
$PERIPHERAL_TREE _which*whish     (2V, 1E)
$PERIPHERAL_TREE _whoas*whops     (2V, 1E)
$PERIPHERAL_TREE _wifey*winey     (2V, 1E)
$PERIPHERAL_TREE _wooed*wooer     (2V, 1E)
$PERIPHERAL_TREE _wowed*wowee     (2V, 1E)
$PERIPHERAL_TREE _wrath*wroth     (2V, 1E)
$PERIPHERAL_TREE _yucky*yukky     (2V, 1E)
$PERIPHERAL_TREE _zincs*zings     (2V, 1E)
$PERIPHERAL_TREE _abase*abuse     (3V, 2E)
$PERIPHERAL_TREE _aimed*armed     (3V, 2E)
$PERIPHERAL_TREE _alone*along     (3V, 2E)
$PERIPHERAL_TREE _altar*alter     (3V, 2E)
$PERIPHERAL_TREE _aorta*sorta     (3V, 2E)
$PERIPHERAL_TREE _areal*areas     (3V, 2E)
$PERIPHERAL_TREE _ashen*ashes     (3V, 2E)
$PERIPHERAL_TREE _awake*aware     (3V, 2E)
$PERIPHERAL_TREE _baulk*caulk     (3V, 2E)
$PERIPHERAL_TREE _bloke*broke     (3V, 2E)
$PERIPHERAL_TREE _bonne*bonny     (3V, 2E)
$PERIPHERAL_TREE _bubba*hubba     (3V, 2E)
$PERIPHERAL_TREE _camel*cameo     (3V, 2E)
$PERIPHERAL_TREE _cargo*largo     (3V, 2E)
$PERIPHERAL_TREE _clean*glean     (3V, 2E)
$PERIPHERAL_TREE _cocoa*cocos     (3V, 2E)
$PERIPHERAL_TREE _culpa*cuppa     (3V, 2E)
$PERIPHERAL_TREE _curbs*nurbs     (3V, 2E)
$PERIPHERAL_TREE _daily*gaily     (3V, 2E)
```

Table 35: *Biblockd Decomposition of* `words.dat` *(Part 22)*

```
$PERIPHERAL_TREE _decal*ducal     (3V, 2E)
$PERIPHERAL_TREE _derby*herby     (3V, 2E)
$PERIPHERAL_TREE _disco*discs     (3V, 2E)
$PERIPHERAL_TREE _doers*dyers     (3V, 2E)
$PERIPHERAL_TREE _doing*dying     (3V, 2E)
$PERIPHERAL_TREE _drily*dryly     (3V, 2E)
$PERIPHERAL_TREE _dulse*pulse     (3V, 2E)
$PERIPHERAL_TREE _dwell*dwelt     (3V, 2E)
$PERIPHERAL_TREE _facet*tacet     (3V, 2E)
$PERIPHERAL_TREE _faery*fairy     (3V, 2E)
$PERIPHERAL_TREE _fault*vault     (3V, 2E)
$PERIPHERAL_TREE _favor*savor     (3V, 2E)
$PERIPHERAL_TREE _feted*fetid     (3V, 2E)
$PERIPHERAL_TREE _filar*filer     (3V, 2E)
$PERIPHERAL_TREE _flail*flair     (3V, 2E)
$PERIPHERAL_TREE _flint*flirt     (3V, 2E)
$PERIPHERAL_TREE _genes*genet     (3V, 2E)
$PERIPHERAL_TREE _gives*gyves     (3V, 2E)
$PERIPHERAL_TREE _glare*glary     (3V, 2E)
$PERIPHERAL_TREE _gnarl*snarl     (3V, 2E)
$PERIPHERAL_TREE _gnats*gnaws     (3V, 2E)
$PERIPHERAL_TREE _goths*moths     (3V, 2E)
$PERIPHERAL_TREE _gulch*mulch     (3V, 2E)
$PERIPHERAL_TREE _harsh*marsh     (3V, 2E)
$PERIPHERAL_TREE _heron*heros     (3V, 2E)
$PERIPHERAL_TREE _hobos*homos     (3V, 2E)
$PERIPHERAL_TREE _inked*inker     (3V, 2E)
$PERIPHERAL_TREE _jewel*newel     (3V, 2E)
$PERIPHERAL_TREE _knack*knock     (3V, 2E)
$PERIPHERAL_TREE _learn*yearn     (3V, 2E)
$PERIPHERAL_TREE _legal*regal     (3V, 2E)
$PERIPHERAL_TREE _leggo*leggy     (3V, 2E)
$PERIPHERAL_TREE _limen*lumen     (3V, 2E)
$PERIPHERAL_TREE _litre*livre     (3V, 2E)
$PERIPHERAL_TREE _local*loyal     (3V, 2E)
$PERIPHERAL_TREE _lucid*lurid     (3V, 2E)
$PERIPHERAL_TREE _magic*manic     (3V, 2E)
```

Table 36: *Biblockd Decomposition of* `words.dat` *(Part 23)*

```
$PERIPHERAL_TREE _midis*minis     (3V, 2E)
$PERIPHERAL_TREE _minas*mynas     (3V, 2E)
$PERIPHERAL_TREE _monte*month     (3V, 2E)
$PERIPHERAL_TREE _mourn*yourn     (3V, 2E)
$PERIPHERAL_TREE _panda*panga     (3V, 2E)
$PERIPHERAL_TREE _pesos*pests     (3V, 2E)
$PERIPHERAL_TREE _playa*plays     (3V, 2E)
$PERIPHERAL_TREE _quell*quill     (3V, 2E)
$PERIPHERAL_TREE _rhumb*thumb     (3V, 2E)
$PERIPHERAL_TREE _rhyme*thyme     (3V, 2E)
$PERIPHERAL_TREE _runic*tunic     (3V, 2E)
$PERIPHERAL_TREE _salve*salvo     (3V, 2E)
$PERIPHERAL_TREE _sauce*saucy     (3V, 2E)
$PERIPHERAL_TREE _seest*weest     (3V, 2E)
$PERIPHERAL_TREE _seeth*teeth     (3V, 2E)
$PERIPHERAL_TREE _semen*seven     (3V, 2E)
$PERIPHERAL_TREE _shall*shawl     (3V, 2E)
$PERIPHERAL_TREE _shoal*shoat     (3V, 2E)
$PERIPHERAL_TREE _silty*sixty     (3V, 2E)
$PERIPHERAL_TREE _siree*spree     (3V, 2E)
$PERIPHERAL_TREE _sprig*sprit     (3V, 2E)
$PERIPHERAL_TREE _tacky*wacky     (3V, 2E)
$PERIPHERAL_TREE _terra*terry     (3V, 2E)
$PERIPHERAL_TREE _tufas*tufts     (3V, 2E)
$PERIPHERAL_TREE _twerp*twirp     (3V, 2E)
$PERIPHERAL_TREE _widen*wider     (3V, 2E)
$PERIPHERAL_TREE _aloud*cloud     (4V, 3E)
$PERIPHERAL_TREE _badge*cadge     (4V, 3E)
$PERIPHERAL_TREE _banal*basal     (4V, 3E)
$PERIPHERAL_TREE _baths*beths     (4V, 3E)
$PERIPHERAL_TREE _biggy*piggy     (4V, 3E)
$PERIPHERAL_TREE _cadet*caret     (4V, 3E)
$PERIPHERAL_TREE _faith*saith     (4V, 3E)
$PERIPHERAL_TREE _fauna*fauns     (4V, 3E)
$PERIPHERAL_TREE _frond*front     (4V, 3E)
$PERIPHERAL_TREE _gamba*gamma     (4V, 3E)
$PERIPHERAL_TREE _gawks*gawky     (4V, 3E)
```

Table 37: *Biblockd Decomposition of* `words.dat` *(Part 24)*

```
$PERIPHERAL_TREE _highs*sighs     (4V, 3E)
$PERIPHERAL_TREE _liege*siege     (4V, 3E)
$PERIPHERAL_TREE _lipid*livid     (4V, 3E)
$PERIPHERAL_TREE _lunch*lynch     (4V, 3E)
$PERIPHERAL_TREE _mavis*maxis     (4V, 3E)
$PERIPHERAL_TREE _moire*noire     (4V, 3E)
$PERIPHERAL_TREE _prexy*proxy     (4V, 3E)
$PERIPHERAL_TREE _scrub*scrum     (4V, 3E)
$PERIPHERAL_TREE _shyly*slyly     (4V, 3E)
$PERIPHERAL_TREE _skein*stein     (4V, 3E)
$PERIPHERAL_TREE _video*vireo     (4V, 3E)
$PERIPHERAL_TREE _abort*about     (5V, 4E)
$PERIPHERAL_TREE _amide*amine     (5V, 4E)
$PERIPHERAL_TREE _anted*antes     (5V, 4E)
$PERIPHERAL_TREE _chief*chiff     (5V, 4E)
$PERIPHERAL_TREE _chugs*chums     (5V, 4E)
$PERIPHERAL_TREE _croup*group     (5V, 4E)
$PERIPHERAL_TREE _cubic*cubit     (5V, 4E)
$PERIPHERAL_TREE _demon*demos     (5V, 4E)
$PERIPHERAL_TREE _dieth*diets     (5V, 4E)
$PERIPHERAL_TREE _gassy*sassy     (5V, 4E)
$PERIPHERAL_TREE _gaudy*gauzy     (5V, 4E)
$PERIPHERAL_TREE _lapin*lapis     (5V, 4E)
$PERIPHERAL_TREE _nobby*nobly     (5V, 4E)
$PERIPHERAL_TREE _salon*solon     (5V, 4E)
$PERIPHERAL_TREE _toile*toils     (5V, 4E)
$PERIPHERAL_TREE _babel*babes     (6V, 5E)
$PERIPHERAL_TREE _ether*other     (6V, 5E)
$PERIPHERAL_TREE _noddy*toddy     (6V, 5E)
$PERIPHERAL_TREE _acorn*adorn     (7V, 6E)
$PERIPHERAL_TREE _baron*boron     (7V, 6E)
$PERIPHERAL_TREE _magus*vagus     (7V, 6E)
$PERIPHERAL_TREE _rabid*rapid     (8V, 7E)
$PERIPHERAL_TREE _recap*remap     (9V, 8E)
```

Table 38: *Biblockd Decomposition of* `words.dat` *(Part 25)*

```
$STOPFREE_KERNEL _ached*aches     (3889V, 13015E)
$No_SUBCOMPONENTS 29
    $SUBCOMPONENT _abaca*abaci        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _abaca*abaci             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $SUBCOMPONENT _abide*amide        (3V, 3E, 2AP, 1BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _abide*amide             (3V, 3E, 2AP, 1BP, 1CP, 0HP)
    $SUBCOMPONENT _agape*agate        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _agape*agate             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $SUBCOMPONENT _areas*arias        (3V, 3E, 3AP, 3BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _areas*arias             (3V, 3E, 3AP, 3BP, 1CP, 0HP)
    $SUBCOMPONENT _bacon*baron        (3V, 3E, 1AP, 1BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _bacon*baron             (3V, 3E, 1AP, 1BP, 1CP, 0HP)
    $SUBCOMPONENT _began*begin        (3V, 3E, 1AP, 1BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _began*begin             (3V, 3E, 1AP, 1BP, 1CP, 0HP)
    $SUBCOMPONENT _chaff*chiff        (3V, 3E, 2AP, 1BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _chaff*chiff             (3V, 3E, 2AP, 1BP, 1CP, 0HP)
    $SUBCOMPONENT _death*heath        (3V, 3E, 3AP, 2BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _death*heath             (3V, 3E, 3AP, 2BP, 1CP, 0HP)
    $SUBCOMPONENT _elude*etude        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _elude*etude             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $SUBCOMPONENT _focal*local        (3V, 3E, 3AP, 2BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _focal*local             (3V, 3E, 3AP, 2BP, 1CP, 0HP)
    $SUBCOMPONENT _icons*ikons        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _icons*ikons             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $SUBCOMPONENT _luaus*lulus        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
    $No_BIBLOCKS 1
        $BIBLOCK _luaus*lulus             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
```

Table 39: *Biblockd Decomposition of* `words.dat` *(Part 26)*

```
$SUBCOMPONENT _major*manor        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _major*manor             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
$SUBCOMPONENT _offed*offen        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _offed*offen             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
$SUBCOMPONENT _quota*quote        (3V, 3E, 1AP, 0BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _quota*quote             (3V, 3E, 1AP, 0BP, 1CP, 0HP)
$SUBCOMPONENT _angle*anile        (4V, 6E, 2AP, 1BP, 2CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _angle*anile             (4V, 6E, 2AP, 1BP, 2CP, 0HP)
$SUBCOMPONENT _bobby*hobby        (4V, 6E, 3AP, 1BP, 2CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _bobby*hobby             (4V, 6E, 3AP, 1BP, 2CP, 0HP)
$SUBCOMPONENT _calve*halve        (4V, 6E, 3AP, 2BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _calve*halve             (4V, 6E, 3AP, 2BP, 1CP, 0HP)
$SUBCOMPONENT _fetch*ketch        (4V, 6E, 1AP, 0BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _fetch*ketch             (4V, 6E, 1AP, 0BP, 1CP, 0HP)
$SUBCOMPONENT _infer*inker        (4V, 6E, 2AP, 1BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _infer*inker             (4V, 6E, 2AP, 1BP, 1CP, 0HP)
$SUBCOMPONENT _three*threw        (5V, 6E, 1AP, 0BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _three*threw             (5V, 6E, 1AP, 0BP, 1CP, 0HP)
$SUBCOMPONENT _dowdy*howdy        (5V, 6E, 3AP, 1BP, 1CP, 1HP)
$No_BIBLOCKS 2
    $BIBLOCK _dowdy*downy             (3V, 3E, 3AP, 1BP, 1CP, 1HP)
    $BIBLOCK _dowdy*howdy             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
$SUBCOMPONENT _splat*splay        (6V, 7E, 2AP, 1BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _splat*splay             (6V, 7E, 2AP, 1BP, 1CP, 0HP)
$SUBCOMPONENT _cabby*cubby        (6V, 8E, 1AP, 1BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _cabby*cubby             (6V, 8E, 1AP, 1BP, 1CP, 0HP)
```

Table 40: *Biblockd Decomposition of* `words.dat` *(Part 27)*

```
$SUBCOMPONENT _carob*carol        (6V, 9E, 3AP, 1BP, 2CP, 1HP)
$No_BIBLOCKS 2
    $BIBLOCK _canon*capon             (3V, 3E, 3AP, 1BP, 2CP, 1HP)
    $BIBLOCK _carob*carol             (4V, 6E, 1AP, 0BP, 1CP, 1HP)
$SUBCOMPONENT _dying*eying        (6V, 15E, 3AP, 2BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _dying*eying             (6V, 15E, 3AP, 2BP, 1CP, 0HP)
$SUBCOMPONENT _bight*eight        (9V, 36E, 2AP, 1BP, 1CP, 0HP)
$No_BIBLOCKS 1
    $BIBLOCK _bight*eight             (9V, 36E, 2AP, 1BP, 1CP, 0HP)
$SUBCOMPONENT _scuff*sluff        (11V, 19E, 2AP, 1BP, 1CP, 1HP)
$No_BIBLOCKS 2
    $BIBLOCK _bluff*fluff             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _scuff*sluff             (9V, 16E, 2AP, 1BP, 1CP, 1HP)
$SUBCOMPONENT _ached*aches        (3756V, 12792E, 406AP, 365BP,
                                                        25CP, 20HP)
$No_BIBLOCKS 21
    $BIBLOCK _abled*ailed             (3V, 3E, 3AP, 2BP, 0CP, 1HP)
    $BIBLOCK _apses*arses             (3V, 3E, 2AP, 1BP, 0CP, 1HP)
    $BIBLOCK _betel*bevel             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _bimbo*himbo             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _brava*brave             (3V, 3E, 1AP, 1BP, 0CP, 1HP)
    $BIBLOCK _curia*curie             (3V, 3E, 1AP, 1BP, 0CP, 1HP)
    $BIBLOCK _dense*sense             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _hammy*jammy             (3V, 3E, 2AP, 1BP, 0CP, 1HP)
    $BIBLOCK _iambs*jambs             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _mafia*mania             (3V, 3E, 2AP, 1BP, 0CP, 1HP)
    $BIBLOCK _penal*renal             (3V, 3E, 2AP, 1BP, 0CP, 1HP)
    $BIBLOCK _rebel*repel             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _scudi*scudo             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _sebum*sedum             (3V, 3E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _unite*units             (3V, 3E, 3AP, 2BP, 0CP, 1HP)
    $BIBLOCK _tempi*tempo             (4V, 6E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _versa*verse             (4V, 6E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _vitae*vital             (4V, 6E, 2AP, 1BP, 0CP, 1HP)
    $BIBLOCK _acing*aging             (5V, 10E, 2AP, 1BP, 0CP, 1HP)
    $BIBLOCK _eager*edger             (6V, 8E, 1AP, 0BP, 0CP, 1HP)
    $BIBLOCK _ached*aches             (3708V, 12711E, 396AP, 355BP,
                                                        25CP, 20HP)
```

Table 41: *Biblockd Decomposition of* `words.dat` *(Part 28)*

```
        $No_INTERNAL_TREES 28
            $INTERNAL_TREE _aback*alack       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _arras*auras       (2V, 1E, 2AP, 1BP, 2CP, 0HP)
            $INTERNAL_TREE _baron*caron       (2V, 1E, 2AP, 1BP, 2CP, 1HP)
            $INTERNAL_TREE _began*begat       (2V, 1E, 2AP, 1BP, 2CP, 0HP)
            $INTERNAL_TREE _bobby*booby       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _calve*carve       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _downs*downy       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _elide*elude       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _enter*inter       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _fecal*focal       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _heath*heats       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _hobby*hubby       (2V, 1E, 2AP, 1BP, 2CP, 0HP)
            $INTERNAL_TREE _lulls*lulus       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _offen*often       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _quite*quote       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _reach*retch       (2V, 1E, 2AP, 1BP, 2CP, 0HP)
            $INTERNAL_TREE _shrew*threw       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _spray*stray       (2V, 1E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _thing*tying       (2V, 1E, 2AP, 1BP, 2CP, 0HP)
            $INTERNAL_TREE _anile*anise       (3V, 2E, 2AP, 1BP, 2CP, 0HP)
            $INTERNAL_TREE _begot*bigot       (3V, 2E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _capes*capos       (3V, 2E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _chafe*chaff       (3V, 2E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _manor*minor       (3V, 2E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _scarf*scurf       (3V, 2E, 3AP, 2BP, 2CP, 0HP)
            $INTERNAL_TREE _abide*abode       (4V, 3E, 4AP, 2BP, 2CP, 0HP)
            $INTERNAL_TREE _crone*crony       (4V, 3E, 2AP, 0BP, 2CP, 0HP)
            $INTERNAL_TREE _abase*abash       (6V, 5E, 3AP, 1BP, 2CP, 0HP)
END REDUCED STRUCTURE
```

Table 42: *Biblockd Decomposition of* `words.dat` *(Part 29)*

References

Pages where cited are in parentheses.

[AhoU1995] Aho, Alfred V. · Ullman, Jeffrey D. *Foundations of Computer Science – C Edition*. Principles of Computer Science Series. Computer Science Press, 1995. *(3)*

[Atal1999] Atallah, Mikhail J., editor. *Algorithms and Theory of Computation Handbook*. CRC Press, 1999. *(3)*

[Bala1997] Balakrishnan, V.K. *Graph Theory*. Schaum's Outlines. McGraw-Hill, 1997. *(3)*

[Boll1998] Bollobás, Béla. *Modern Graph Theory*. Graduate Texts in Mathematics. Springer-Verlag, 1998. *(3)*

[CormLR1990] Cormen, Thomas. H · Leiserson, Charles E. · Rivest, Ronald L. *Introduction to Algorithms*. The MIT Electrical and Computer Science Series. The MIT Press / McGraw-Hill Book Company, 1990. *(3)*

[Dies1996] Diestel, Reinhard. *Graphentheorie*. Springer-Lehrbuch. Springer, 1996. *(3)*

[HopcT1973a] Hopcroft, John E. · Tarjan, Robert Endre. Dividing a Graph into Triconnected Components. *SIAM Journal of Computing*, 2(3):135–158, 1973. *(28)*

[Knut1993] Knuth, Donald E. *The Stanford GraphBase*. Addison-Wesley Publishing Company, 1993. A Platform for Combinatorial Computing. *(1, 3, 7, 9)*

[OttmW1996] Ottmann, Thomas · Widmayer, Peter. *Algorithmen und Datenstrukturen*. Spektrum Lehrbuch. Spektrum Akademischer Verlag, 3 edition, 1996. *(3)*

[Stie2007a] Stiege, Günther. General Graphs. Berichte aus dem Department für Informatik 02/07, Universität Oldenburg, 2007. Available from *http://www-bvs.informatik.uni-oldenburg.de/Literatur/Berichte/oib07-02.html*. *(4, 28)*

[Stie2009a] Stiege, Günther. *GHS Graph Handling System User Manual – Version 5.0*. Universität Oldenburg – Fachbereich Informatik, D-26111 Oldenburg, Germany, 2009. Available from *http://www-bvs.informatik.uni-oldenburg.de/Literatur/handbuch.html*. *(3)*

[Tura1996] Turau, Volker. *Algorithmische Graphentheorie*. Addison-Wesley, 1996. *(3)*

[Volk1996] Volkmann, Lutz. *Fundamente der Graphentheorie*. Springer Lehrbuch Mathematik. Springer-Verlag, 1996. *(3)*

Technical Reports

Fakultät II, Department für Informatik, Universität Oldenburg,
Postfach 2503, 26111 Oldenburg, Germany

1/87 A. Viereck: *Klassifikationen, Konzepte und Modelle für den Mensch-Rechner-Dialog* (Dissertation)

2/87 A. Schwill: *Forbidden subgraphs and reduction systems: A comparison*

3/87 J. Kämper: *Non-uniform proof systems: A new framework to describe non-uniform and probabilistic complexity classes*

1/88 K. Ambos-Spies, H. Fleischhack, H. Huwig: *Diagonalizing over deterministic polynomial time*

2/88 A. Schwill: *Shortest edge-disjoint paths in geodetically connected graphs*

3/88 V. Claus, U. Lichtblau (Hrsg.): *1. Tagung zur Küsten-Informatik*

1/89 U. van der Valk: *Einige Entscheidbarkeits- und Unentscheidbarkeitsresultate für Klassen von S/T-Netzen unter Maximum Firing Strategie und unter Prioritätenstrategien*

2/89 J. Kämper: *Strukturelle Untersuchungen im Umfeld der Komplexitätsklassen P und NP unter besonderer Berücksichtigung nichtuniformer, probabilistischer und disjunktiv selbstreduzierender Algorithmen* (Dissertation)

3/89 J. Kämper: *Nondeterministic oracle Turing machines with maximal computation paths*

1/90 A. Schwill: *Shortest edge-disjoint paths in graphs* (Dissertation)

2/90 K.R. Apt, E.-R. Olderog: *Using transformations to verify parallel programs*

3/90 U. Lichtblau: *Flußgraphgrammatiken* (Dissertation)

4/90 K.R. Apt, E.-R. Olderog: *Introduction to program verification*

5/90 H. Jasper: *Datenbankunterstützung für Prolog-Programmierumgebungen* (Dissertation)

1/91 F. Korf: *Net-based efficient simulation of AADL specifications*

2/91 S.V. Krishnan, C. Pandu Rangan, A. Schwill, S. Seshadri: *Two disjoint paths in chordal graphs*

3/91 H. Eirund: *Modellierung und Manipulation multimedialer Dokumente* (Dissertation)

4/91 G. Schreiber: *Ein funktionaler Äquivalenzbegriff für den hierarchischen Entwurf von Netzen*

1/92 A. Viereck (Hrsg.): *Ergebnisse der 11. Arbeitstagung, Mensch-Maschine Kommunikation*

2/92 P. Gorny, U. Daldrup, H. Schwab: *Zwischenbilanz: Menschengerechte Gestaltung von Software*

3/92 E.-R. Olderog, St. Rössig, J. Sander, M. Schenke: *ProCoS at Oldenburg: The Interface between Specification Language and occam-like Programming*
Language

4/92 F. Korf: *Synthesis of VHDL Test Environments form Temporal Logic Specifications*

5/92 W. Kowalk: *Konstruktorentechnik: Neue Methoden zur Mengenrechnung,*
Logikrechnung und Intervallrechnung

1/93 Ch. Dietz, G. Schreiber: *Eine Termdarstellung für S/T-Netze*

2/93 J. Sauer: *Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken*

3/93 M. Sonnenschein, U. Lichtblau (Hrsg.): *6. Kolloquium der Arbeitsgruppe Informatik-Systeme*

4/93 H. Fleischhack, U. Lichtblau, M. Sonnenschein, R. Wieting: *Generische Definition hierarchischer zeitbeschrifteter höherer Petrinetze*

5/93 F. Köster, L. Twele, R. Wieting, W. Ziegler: *Fallbeispiele zur Modellierung mit THOR-Netzen*

1/94 R. Götze: *Dialogmodellierung für multimediale Benutzerschnittstellen*

2/94 B. Müller: *PPO – Eine objektorientierte Prolog-Erweiterung zur Entwicklung wissensbasierter Anwendungssysteme*

3/94 W. Damm/A. Mikschl: *Projekt Entwurf und Implementierung eines Multi-threaded RIS C-Prozessors*

4/94 S. Rössig: *A Transformational Approach to the Design of Communicating Systems* (Dissertation)

5/94 G. Schreiber: *Funktionale Äquivalenz von Petri-Netzen* (Dissertation)

1/95 A. Gronewold, H. Fleischhack: *Language Preserving Reductions of Safe Petri-Nets*

2/95 H. Reineke: *Struktur und Verhalten von verteilten endlichen Automaten* (Dissertation)

3/95 H. Behrends: *Beschreibung ereignisgesteuerter Aktivitäten in datenbankgestützten Informationssystemen* (Dissertation)

4/95 U. M. Levens: *Computerunterstütztes Modellieren von Musikstücken mit Petri-Netzen: Das Mailänder Konzept*

1/96 M. Burke: *FDDI und ATM in multimedialen Anwendungsumgebungen* (Dissertation)

2/96 I. Pitschke: *Interaktive Rekonstruktion geometrischer Modelle aus digi-*

1/98 S. Kleuker: *Inkrementelle Entwicklung von verifizierten Spezifikationen für verteilte Systeme* (Dissertation)

2/98 J. Bohn: *Mechanical Support and Validation of a Design Calculus for Communicating Systems by a Logic-Based Proof System* (Dissertation)

3/98 L. Köhler: *Fuzzy Geometrie und Anwendungen in der medizinischen Bildverarbeitung* (Dissertation)

4/98 J. Helbig: *Linking Visual Formalisms: A Compositional Proof System for*
Statecharts Based on Symbolic Timing Diagrams (Dissertation)

5/98 G. Stiege: *Edge Partitions in Undirected Graphs*

6/98 A. Gerns: *Entwicklung und Bewertung von Objektmigrationsstrategien für*
verteilte Umgebungen

7/98 M. Stadler: *Abstrakte Rechnernetzmodelle als Grundlage einer umfassenen Automatisierung des Netzmanagements – Konzepte und Sprachen zu ihrer Umsetzung* (Dissertation)

8/98 M. S. Steiner: *Lastverteilung in heterogenen Systemen*

9/98 Clemens Otte: *Fuzzy-Protopyp-Klassifikatoren und deren Anwendung zur automatischen Merkmalsselektion*

1/99 Juliane Vorndamme: *Die Auswirkungen rechtlicher Verpflichtungen auf die Softwareentwicklung*

2/99 Eike Best, Kerstin M. Richter: *Relational Semantics Revisited*

3/99 Jung Sun Lie: *Einsatz von Objektmigrationsstrategien zur Leistungssteigerung in verteilten Systemen*

4/99 Fachbereich Informatik: *Zwei-Jahresbericht des Fachbereichs Informatik (1.10.1996 – 30.9.1998)*

5/99 Ingo Stierand, Olaf Maibaum, Björn Briel und Günther Stiege: *Cassandra – Generierung, Analyse und Simulation von eingebetteten Multiprozessor-Echtzeitsystemen*

6/99 Gunnar Wittich: *Ein problemorientierte Ansatz zum Nachweis von Realzeiteigenschaften eingebetteter Systeme*

7/99 Annegret Habel, Jürgen Müller, Detlef Plump: *Double-Pushout Graph Transformation Revisited*

8/99 Ingo Stierand: *Eine Konfigurationssprache zur Erstellung von Ambrosia/MP-Systemen*

9/99 Igor V. Tarasyuk: *Equivalences for Concurrent and Distributed Systems*

10/99 Eike Best, Alexander Lavrov: *Generalised Composition Operations for High-Level Petri-Nets*

11/99 Alexander Lavrov: *Enhancing Mixed Nonlinear Optimisation: A Hybrid Approach*

12/99 Alexander Lavrov: *Hybrid Techniques in Discrete-Event System Modelling and Control: Some Examples*

13/99 Eike Best, Raymond Devillers, Maciej Koutny: *Recursion and Petri Nets*

14/99 Eike Best, Raymond Devillers, Maciej Koutny: *The Box Algebra = Petri Nets + Process Expressions*

15/99 Eike Best, Harro Wimmel: *Reducing k-safe Petri Nets to Pomset-Equivalent 1-safe Petri Nets*

16/99 Udo Brockmeyer: *Verifikation von STATEMATE Designs* (Dissertation)

1/00 Henning Dierks: *Specification and Verification of Polling Real-Time Systems* (Dissertation)

2/00 Clemens Fischer: *Combination and Implementation of Process and Data: From CSP-OZ to Java* (Dissertation)

3/00 Cheryl Kleuker: *Constraint Diagrams* (Dissertation)

4/00 Thomas Thielke: *Linear-algebraische Methoden zur Beschreibung, Verfeinerung und Analyse gefärbter Petrinetze* (Dissertation)

1/01 Günther Stiege: *Higher Decompositions in Undirected Graphs*

2/01 Ute Vogel: *Zwei-Jahres-Bericht*

3/01 Joseph Tapken: *Model-Checking in Duration Calculus Specifications* (Dissertation)

4/01 Björn Briel: *Analyse eingebetteter Systeme mittels verteilter Simulation* (Dissertation)

5/01 Günther Stiege: *Standard Decomposition and Periodicity of Digraphs*

6/01 Ingo Stierand: *Ambrosia/MP – Ein Echtzeitbetriebssystem für eingebettete Mehrprozessorsysteme*

1/02 Giorgio Busatto, Annegret Habel: *Improving the Quality of Hypertexts Using Graph Transformation*

2/02 Giorgio Busatto: *Modeling Hyperweb Dynamics through Hierarchical Graph Transformation*

3/02 Giorgio Busatto: *An Abstract Model of Hierarchical Graphs and Hierarchical Graph Transformation* (Dissertation)

4/02 Laila Kabous: *An Object Oriented Design Methotology for Hard Real-Time Systems: The OOHARTS Approach* (Dissertation)

1/03 Ute Vogel: *2-Jahres-Bericht*

2/03 Olaf Maibaum: *Bestimmung symbolischer Laufzeiten in eingebetteten Echtzeitsystemen* (Dissertation)

3/03 Günther Stiege, Ingo Stierand: *Connectedness-Based Hierarchical Decomposition of Undirected Graphs* (Bericht)

4/03 Willi Hasselbring, Susanne Petersen: *Standards f"ur medizinische Kommunikation und Dokumentation* (Bericht)

5/03 Andreas Möller: *Eine virtuelle Maschine f"ur Graphprogramme* (Bericht)

6/03 Tom Bienmüller: *Reducing Complexity for the Verification of Statemate Designs* (Bericht)

7/03 Sandra Steinert: *Graph Programs for Graph Algorithms* (Bericht)

8/03 Jochen Klose: *Live Sequence Charts: A Graphical Formalism for the Specification of Communication Behaviour* (Dissertation)

1/04 Jens Oehlerking: *Transformation of Edmonds' Maximum Matching Algorithm into a Graph Program* (Bericht)

2/04 Sergej Alekseev: *Dienste Intelligenter Netze / Graphentheoretische Methoden in der Kontrollflu"sanalyse* (Bericht)

3/04 Giorgio Busatto: *GraJ: A System for Executing Graph Programs in Java* (Bericht)

1/05 Sergej Alekseev, Johannes Wust: *Graph Theoretical Methods in the Control Flow Analysis of Object Oriented Real Time Software (Bericht)*

2/05 Ute Vogel: *2-Jahres-Bericht* (Bericht)

3/05 Igor Tarasyuk: *Discrete time stochastic Petri box calculus* (Bericht)

1/06 Henning Dierks: *Time, Abstraction and Heuristics* (Habilitation)

2/06 Li Sek Su: *Full-Output Siphons and Deadlock-Freeness for Free Choice Petri Nets* (Bericht)

3/06 Timo Warns: *Solving Concensus Using Structural Failure Models* (Bericht)

4/06 Sergej Alekseev: *Ablaufanalyse objektorientierter Echtzeitanwendungen mit graphentheoretischen Methoden* (Dissertation)

5/06 Li Sek Su: *Some Considerations on the Foundation of NP-Completeness Theory* (Bericht)

6/06 Li Sek Su: *Semitraps and Deadlock-Freeness for Reduced Asymmetric Choice Nets* (Bericht)

7/06 Li Sek Su: *Algorithms of computing the Deadlock Markings Sets for Petri Nets* (Bericht)

8/06 Annegret Habel, Karl-Heinz Pennemann, Arend Rensink: *Weakest Preconditions for High-Level Programs (Long Version)* (Bericht)

9/06 Jochen Hoenicke: *Combination of Processes, Data, and Time* (Dissertation)

10/06 Steffen Becker, Marco Boscovic, Abhishek Dhama, Simon Giesecke, Jens Happe, Wilhelm Hasselbring, Heiko Koziolek, Henrik Lipskoch, Roland Meyer, Margarethe Muhle, Alexandra Paul, Jan Ploski, Matthias Rohr, Mani Swaminathan, Timo Warns, Daniel Winteler: *Truthworthy Software Systems: A Discussion of Basic Concepts and Teminology* (Bericht)

11/06 Christian Zuckschwerdt: *Ein System zur Transformation von Konsistenz-und Anwendungsbedingungen* (Bericht)

01/07 Andreas Schäfer: *Specification and Verification of Mobile Real-Time Systems* (Dissertation)

02/07 Günther Stiege: *General Graphs* (Bericht)

03/07 Wolfgang Kowalk: *Integralrechnung* (Bericht)

04/07 Karl Azab, Karl-Heinz Pennemann: *Type Checking C++ Template Instantiation by Graph Programs* (Bericht)

01/08 Roland Meyer: *On depth and breath in the Pi-Calculus* (Bericht)

02/08 Ingo Brückner: *Slicing Integrated Formal Specifications for Verification* (Dissertation)

03/08 Ute Vogel: *2-Jahres-Bericht 2004 – 2006* (Bericht)

04/08 Günther Stiege: *Summierbare Familien* (Bericht)

05/08 Igor V. Tarasyuk: *Investigating equivalence realations in dtsPBC* (Bericht)

01/09 Elke Wilkeit: *2-Jahres-Bericht, 01.10.2006 – 3.09.2008* (Bericht)

02/09 Roland Meyer: *Structural Stationarity in the pi-Calculus* (Dissertation)

03/09 InformatikerInnen des Moduls Soft Skills: *E-Book Soft Skills* (Bericht)

04/09 Eike Best: *Separability in Persistent Petri Nets* (Bericht)

01/10 Igor V. Tarasyuk: *Equivalence relations for behaviour-perserving reduction and modular performance evaluation in dtsPBC* (Bericht)

02/10 Roman Dubtsov: *Time Trannsition Systems with Independence and Marked Sott Domains: an Adjunction* (Bericht)

01/11 Elena S. Oshevskaya: *Matching Equivalences on Higher Dimensional Automata Models* (Bericht)

02/11 Elke Wilkeit: *2-Jahres-Bericht, 01.10.2008 – 30.09.2010* (Bericht)

03/11 Johannes Faber: *Verification Architectures for Complex Real-Time Systems* (Dissertation)

04/11 Igor V. Tarasyuk: *Eqivalences for modular performance analysis in dtsPBC* (Bericht)

01/12 Günther Stiege: *Playing with Knuth's words.dat* (Bericht)