USER GUIDE FOR VSEP1 PROGRAM

1. **Introduction**

VSEP1 program implements ***VSEP*** algorithms that finds the ***generators, orbits and order of the graph automorphism group*** Aut(G,PI) (shortly GOOGAG(G,PI)), PI is the input partition – usually PI is the unit partition.. Aut(G,PI) is also called partition- wise stabilizer. The graphs are simple - undirected without loops and parallel edges. ***Vsep1*** is a Fortran program that can be compiled and built by any Compaq Fortran compiler. Preliminary version of Vsep algorithm is described in [1] – only the version VSEP-SCH is not presented in [1] (see below). The program VSEP1 is ***tested on all benchmark graphs in [2].***

Three new algorithms, named Vsep, are implemented. One of them, ***Vsep-е*** is exact and others two, ***Vsep-orb***, ***Vsep-sch*** are heuristic. All are for determining the generators, orbits and order of an undirected graph automorphism group. Vsep-orb firstly finds heuristically the generators and orbits and then uses the exact one on the orbital partition for determining the order of the group. Vsep-sch differs from Vsep-orb in using the Schreier-Sims algorithm for determining the order of the group using as input the preliminary found generators. There is a version of each of the three programs that finds GOOGAG(G,PI)), ***when PI is defined by the user***.

1. **Idea of the algorithms**

A basic tool of these algorithms is the ***adjacency refinement procedure*** that gives finer output partition on a given input partition of graph vertices. The refinement procedure is a simple iterative algorithm based on the criterion of relative degree of a vertex toward a basic cell in the partition. A ***search tree*** is used in the algorithms - each node of the tree is a partition. Each terminal node of the search tree is a discrete partition called a ***numeration***. All numerations derivative of the selected vertices called a “***bouque***t” are stored in a coded form in a hash table in order to reduce the necessary storage – this is a ***main difference*** of Vsep-e with the known graph automorphism group algorithms. A new strategy is used in the exact algorithm: if during its execution some of the searched or intermediate variables obtain a wrong value (cases CS3,CS4 [1]) then the algorithm continues from a new start point losing some of the results determined so far. The new start point is such that the correct results can be obtained. The proposed algorithms has been tested on the nauy&Traces [2] benchmark graphs and compared with Traces, and the results show that “Vsep-e” has graph families that are best cases for it and worst cases for Traces and vice versa. The heuristic versions of Vsep are based on determining some number of discreet partitions derivative of a representative of the current orbits in the selected cell of the initial partition and comparing them for an automorphism, i.e. their search trees are reduced. The heuristic algorithms are almost exact and are many times faster than the exact one even for the very difficult graph families. The experimental tests exhibit that the worst-cases running time of the exact algorithm is exponential. Several cell selectors are used in Vsep, some of them are known and some are new. We also use a chooser of cell selector for choosing the optimal cell selector for the manipulated graph. The experiments show that the running time of Vsep algorithms does not depend on the vertex labeling.

1. **Some basic notions and procedures**

:a) Two ***cells*** are called ***adjacent*** if there is at least one edge between their vertices;b) Two ***cells*** of a partition are called ***non-trivially joined*** (have non-trivial join, non-uniformly joined) if the number of edges between them is greater than 0 and less than maximum possible; c) A **channel of a cell C**, denoted as *Ch(C))*, *is the number of the edges adjacent to all vertices of the cell*. A **channel of two cells C1,C2** , denoted as *Ch(C1,C2))*, *is the number of all edges between all vertices of the two cells; d)* *A* ***channel graph of a partition П*** *(ChG) is an weighted graph with loops: each vertex of ChG corresponds uniquely to a cell of Π and its weight is the channel of the cell; each edge of ChG corresponds to a channel of the corresponding cells of Π and the weight of this edge is equal to the weight of this channel; e)* *The* **relative degree ρ(x,Ci) of a vertex х∈Сi toward a cell Сi** *is equal to the number of vertices of cell С*i *adjacent to a vertex x. We denote by ν(x, П) a* ***cell-degree vector*** *defined as ν(x, П)=(ρ(x,Ci), i=1,…,p) – it is a vector whose components are the relative degrees of x to each cell in П; f)* ***LP*** *- last level to which a back move is made,****LMIN****-the minmal level to which a back move is made;****POSL****- array of the indeces of the base points of the target cells of the parent partitions of a given partition;*

*h) The* ***tree4*** *procedure finds heuristically the generators and orbits of Aut(G,* *Π). It calls a procedure* ***table1*** *with different parameters mbrsyvp,wth1,dpth1, br1a,br2a (see appendix,d) untill the number of found orbits remains unchanged mbrsyvp+1 times.*

*i)* *), The* ***table1*** *procedure [1] finds heuristically the generators and orbits of Aut(G, Π) with concrete parameters wth1,dpth1, br1a,br2a. It has two parts:a* ***fork*** *and a* ***forest of******regular selection trees*** *(****FRST****).* The***fork*** *procedure generates one numeration for each vertex in target cell by series of forward move and compares it for an automorphism with the previously found numerations.The* ***FRST*** *procedure generates a forest of regular selection trees with defined parameters. The nodes of FRST are numerations that are compared for automorphism .*

*I) The* ***BUCKETCLASS***  *procedure partitions the vertices x in each cell of a partition PI according to S code [1] of the derived stable partition PI(X)*

***Vsep1*** program implements the following three versions of ***VSEP*** algorithm:

1. ***VSEP-E*** (exact version) - finds the generators, orbits and order of the graph automorphism group ***Aut(G,PI),*** shortly ***GOOGAG(G,PI)***. There are two main parts of ***VSEP-E***: ***PART1*** – finds ***GOOGAG(G,PI(x1)),*** x1-the first selected vertex in PI (level L=1) and the ***bouquet*** B(x1): ***PART2*** – finds GOOGAG(G,PI) by generating one numeration for each representative of current orbits and comparing it with B(x1) bouquet.
2. ***VSEP-ORB*** – heuristic algorithm that firstly finds the generators and orbits of Aut(G) by an heuristic algorithm called ***tree4*** and then call the exact one to determine the order |Aut(G)|.
3. ***VSEP-SCH*** – heuristic algorithm that firstly finds the generators and orbits of Aut(G) by an heuristic algorithm called tree4 and then call the Schreier-Sims algorithm to determine the order |Aut(G)|.
4. Each above version of Vsep-1 algorithm can find the generators, orbits and order of the automorphism group Aut(G, π) on unit partition (πu) or on arbitrary input partition π (partition stabilizer) that is entered by the user;
5. **Before running Vsep1**

Create a folder into which you should write: a) Vsep-1 program file; b) a folder with the description of graphs in dre format (**http://pallini.di.uniroma1.it/**) and c) a text file with the paths to graphs. Then build the program and run it!

**Example:**

File **blmalki1**.txt (text file with the paths to graphs):

malki/gira.dre

malki/faradj1.dre

malki/prizma.dre

and a folder **malki1** with the description of graphs in dre format:



1. **Vsep1 input**

To run Vsep1 program clique the run tab and then:

* 1. enter the text file with the paths to graphs (for example, blmalki1.txt)
  2. if there are no options clique E+enter and the program start running with default parameters.

Default values are (see below the meaning of the notations) :UPR=3 - automatic selection between VSEP-orb and VSEP-sch; R=0 - no relabeling;MTH2=.false.- no Mathon doubling; mck=.false.- no McKay extension; IZB1,IZB2=1,1; MBRSYVP, WTH1, DPTH1, BR1A, BR2A=4,5,2,2,1; PARTPROIZ=.false.- unit input partition.

The user has options - then he should enter the following **capital letters**:

* U - for a selection of the algorithm version, then enter: 0 for schsims, 1 for orbital,2 for exact algorithm, 3 for automatic selection from orbital and schsims
* R - for random relabeling of graph vertices (the default is no relabeling)
* M - for Mathon doubling vertices (the default is no Mathon doubling), see Appendix b
* K – for McKay extension (the default is no McKay extension), see Appendix a
* I – for entering new values of IZB1,IZB2 (the default is Izb1=izb2=1), see Appendix c
* S – for stop running
* T – for entering new values of MBRSYVP,WTH1,DPTH1,BR1A,BR2A
* P – for entering a user defined partition, see Appendix d
* L – for selecting bliss format for the input graphs

Always, ***the last entered letter should be E***.

1. **Vsep1 output**

The generators are written into the file ***GEN.DAT*** in form of permutation ***P*** that is an image of the trivial permutation 1,2,…,n to ***P*** (i maps to ***P(i)***). On the first line is the path to the graph representation, on the second line - ***N***-the number of graph vertices and on the third – the base points.

***Vsep*** output depends on the program version (if the procedures ***bucketclass*** and ***tee4*** are called on defined conditions). Usually the output contains (in file ***resvsep1.dat*** and on the screen) the following sequence of text messages and values (the text in quotes below is printed without quotes) :

1. Program name,
2. Date and time;
3. list of graph names;
4. For each graph the output is:
5. Graph name
6. Serial number of the graph in the list
7. N-number of the graph vertices and KL2 – twice the graph edges+1
8. MINVAL,MAXVAL - minimal and maximal vertex degree, respectively.
9. ‘END OF IZBILP1 LKMIN=’ value1 ‘IZB=’ value2 ‘POSL(1)=’vlulue3 –after the first cell selector chooser choosed values are: value1 is for LKMIN,value2 is for izb(serial number of the cell selector),value3 is for POSL(1)-the index of the selected vertex at level=1
10. IT1, IT2-values of the first and last index of the target cell of PI(1) and REL= maxval/minval
11. ‘***bucketclass’*** appears if the bucketclass subroutine is called
12. The output of this section depends on the selected algorithm version:
13. If there is a call to ***tree4*** and then a call to part1 (***version vsep-orb***) then the output is:
14. ‘TREE4-START’ – means call to tree4;
15. Values of ***mbrsyvp, wth1, dpth1, br1a,br2a***
16. ‘END FORK’ and value of ***brorb*** (number of current orbits)
17. Current values of ***wth, dpth***
18. ‘AFTER TABLE1’ and the value of ***brorb1***, the last two lines are repeated until ***brorb1=1*** or ***brorb1*** becomes unchanged ***mbrsyvp+1*** times.
19. Time after tree4
20. After tree4 brorb – the orbit number after the run of tree4.
21. If the **schsims** procedure is called after ***tree4*** call (***vsep-sch version***) then the following lines to the output are added:
22. ‘CALL SCHSIMS’
23. ‘SCHSIMS IOPR’=value of the number of base points at the call to schsims
24. ‘NGENschsims=’value1 ‘ KOREDS=’value2 – value1 is the number of generators, value2-group order

If there is no call to tree4 (i.e. vsep-e is run) this section is empty

1. The sizes of orbits (orbsize) in increasing order of their lengths
2. The orbits in 2 forms: first – each orbit in brackets, second – to each vertex is given the representative of its orbit
3. THE ORDER OF THE AUTOMORPHISM GROUP
4. Time of operation (runtime) in seconds
5. TTOTAL – total runtime in seconds for all graphs until the current graph (including it)
6. BRNOM – size of the bouquet
7. NGEN – number of generators
8. BROB, BRNOB –number of generators that unites orbits and that not unites orbits, respectively (only if TREE4 has been called)
9. BRORB – number of orbits
10. If CS3 case [1] occurs in PART1 then there is a message in a form: CS3, values of LMIN,LP and POSL
11. If CS4 case [1] occurs in PART1 then there is a message in a form: CS4, values of LMIN, L, LP,

stabilizer order |A(LMIN-1)| and POSL

1. Finally, for the graph family is presented a table each line of which contains: graph name, graph order (N), number of orbits, running time, number of digits of |Aut(G)| and |Aut(G)|
2. TTOTAL – total runtime in seconds for whole graph family.

**EXAMPLES OF VSEP-1 OUTPUT**

**Example** 1:

File blmalk1.txt:

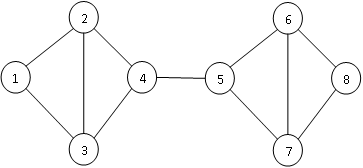
malki/gira.dre

malki/faradj1.dre

malki/prizma.dre

**RESULTS** for graphs:

a) malki/gira.dre



$=1; n=8 g

1: 2 3

2: 3 4

3: 4

4: 5

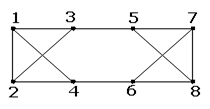
5: 6 7

6: 7 8

7: 8.

$$

b)



malki/faradj1.dre

$=1; n=8 g

1: 2 3 4

2: 3 4

3: 5

4: 6

5: 7 8

6: 7 8

7: 8.

$$

c)

malki/prizma.dre (regular triangle prizma)

$=1; n=6 g

1: 2 3 4

2: 3 5

3: 6

4: 5 6

5: 6.

$$

PROGRAM\_NAME=VSEP4P\_PUB3

YEAR,MONTH,DAY,DIFFERENCE,HOUR,MINUTES,SEONDS,MILLISECONDS=

2016 4 18 180 13 37 10 317

malki/gira.dre malki/faradj1.dre malki/prizma.dre

malki/gira.dre

GRAPH # 1

N= 8 KL2= 23

MINVAL= 2 MAXVAL= 3

END OF IZBILP1 LKMIN= 3 IZB= 1 POSL(1)= 5

IT1= 5 IT2= 8 REL= 1.500000

the number of base points for the exact version = 2

PART1 START

PART2 START

STARTING VERTEX VZ1= 2 DORBVZ1= 4

ORBSIZE= 1- 2 2- 2 3- 4

ORBITS=

5 4 0 8 1 0 2 3 7 6 0

ARRAY NORB=

1( 8) 2( 2) 3( 2) 4( 5) 5( 5) 6( 2) 7( 2)

8( 8)

THE ORDER OF THE AUTOMORPHISM GROUP IS

8(2\*\*48)(\*\* 0)

Time of operation= 0.0000000E+00 seconds

TTOTAL= 0.000000000000000E+000

BRNOM= 1

NGEN= 3

BROB,BRNOB= 0 0

BRORB= 3

malki/faradj1.dre

GRAPH # 2

N= 8 KL2= 25

MINVAL= 3 MAXVAL= 3

END OF IZBILP1 LKMIN= 4 IZB= 1 POSL(1)= 1

IT1= 1 IT2= 8 REL= 1.000000

BUCKETCLASS

BRCL1,BRCL2,BRCL3= 1 2 2

END OF IZBILP LKMIN= 4 IZB= 1 POSL(1)= 1

the number of base points for the exact version = 3

PART1 START

PART2 START

STARTING VERTEX VZ1= 8 DORBVZ1= 4

ORBSIZE= 1- 4 2- 4

ORBITS=

3 4 6 5 0 8 7 1 2 0

ARRAY NORB=

1( 8) 2( 8) 3( 3) 4( 3) 5( 3) 6( 3) 7( 8)

8( 8)

THE ORDER OF THE AUTOMORPHISM GROUP IS

16(2\*\*48)(\*\* 0)

Time of operation= 0.0000000E+00 seconds

TTOTAL= 0.000000000000000E+000

BRNOM= 1

NGEN= 4

BROB,BRNOB= 0 0

BRORB= 2

malki/prizma.dre

GRAPH # 3

N= 6 KL2= 19

MINVAL= 3 MAXVAL= 3

END OF IZBILP1 LKMIN= 3 IZB= 1 POSL(1)= 1

IT1= 1 IT2= 6 REL= 1.000000

BUCKETCLASS

END OF IZBILP LKMIN= 3 IZB= 1 POSL(1)= 1

TREE4-START

mbrsyvp= 4 WTH1= 5 DPTH1= 2 br1A= 1

br2A= 1

WTH, DPTH= 5 2

AFTER TABLE L1= 1 BRORB1= 1

Time AFTER TREE4 0.0000000E+00 seconds

AFTER TREE4 BRORB1= 1

END OF IZBILP LKMIN= 3 IZB= 1 POSL(1)= 1

the number of base points for the exact version = 2

PART1 START

PART2 START

STARTING VERTEX VZ1= 1 DORBVZ1= 6

ORBSIZE= 1- 6

ORBITS=

1 3 2 5 6 4 0

ARRAY NORB=

1( 1) 2( 1) 3( 1) 4( 1) 5( 1) 6( 1)

THE ORDER OF THE AUTOMORPHISM GROUP IS

12(2\*\*48)(\*\* 0)

Time of operation= 0.0000000E+00 seconds

TTOTAL= 0.000000000000000E+000

BRNOM= 1

NGEN= 3

BROB,BRNOB= 2 0

BRORB= 1

malki/gira.dre 8 3 0.000 1 8

malki/faradj1.dre 8 2 0.000 2 16

malki/prizma.dre 6 1 0.000 2 12

TTOTAL= 0.000000000000000E+000

**File GEN.DAT (generators):**

malki/gira.dre

8

1 2 3 4 5 7 6 8

1 3 2 4 5 6 7 8

8 6 7 5 4 2 3 1

malki/faradj1.dre

8

1 2 4 3 6 5 7 8

2 1 3 4 5 6 7 8

1 2 3 4 5 6 8 7

7 8 6 5 4 3 1 2

malki/prizma.dre

6

1 3 2 4 6 5

2 1 3 5 4 6

5 4 6 2 1 3

**EXAMPLE 2 (with call to schsims)**

latin/latin-8.dre

GRAPH # 6

N= 64 KL2= 1345

MINVAL= 21 MAXVAL= 21

END OF IZBILP1 LKMIN= 4 IZB= 1 POSL(1)= 1

IT1= 1 IT2= 64 REL= 1.000000

BUCKETCLASS

END OF IZBILP LKMIN= 4 IZB= 1 POSL(1)= 1

TREE4-START

mbrsyvp= 4 WTH1= 5 DPTH1= 2 br1A= 1

br2A= 1

WTH, DPTH= 5 2

END FORK brorb= 2

AFTER TABLE L1= 1 BRORB1= 1

Time AFTER TREE4 1.5602112E-02 seconds

AFTER TREE4 BRORB1= 1

CALL SCHSIMS

schsims iopr= 3

NGENsch= 9 koreds= 1536

ORBSIZE= 1- 64

ORBITS=

26 14 32 12 30 10 28 16 42 64 48 62 46 60 44 58

18 49 24 55 22 53 20 51 45 63 43 61 41 59 47 57

52 23 50 21 56 19 54 17 4 1 2 7 8 5 6 3

9 31 15 29 13 27 11 25 38 39 36 37 34 35 40 33

0

ARRAY NORB=

1( 26) 2( 26) 3( 26) 4( 26) 5( 26) 6( 26) 7( 26)

8( 26) 9( 26) 10( 26) 11( 26) 12( 26) 13( 26) 14( 26)

15( 26) 16( 26) 17( 26) 18( 26) 19( 26) 20( 26) 21( 26)

22( 26) 23( 26) 24( 26) 25( 26) 26( 26) 27( 26) 28( 26)

29( 26) 30( 26) 31( 26) 32( 26) 33( 26) 34( 26) 35( 26)

36( 26) 37( 26) 38( 26) 39( 26) 40( 26) 41( 26) 42( 26)

43( 26) 44( 26) 45( 26) 46( 26) 47( 26) 48( 26) 49( 26)

50( 26) 51( 26) 52( 26) 53( 26) 54( 26) 55( 26) 56( 26)

57( 26) 58( 26) 59( 26) 60( 26) 61( 26) 62( 26) 63( 26)

64( 26)

THE ORDER OF THE AUTOMORPHISM GROUP IS

1536(2\*\*48)(\*\* 0)

Time of operation= 1.5602112E-02 seconds

TTOTAL= 6.240081787109375E-002

BRNOM= 2

NGEN= 11

BROB,BRNOB= 3 8

BRORB= 1

**APPENIX**

1. **For NcKay extension**:

Take a connected graph with n vertices. Replace each vertex v by two vertices {v',v"} and each edge by four edges: if v--w was an edge before, now v'--w', v'--w", v"--w', v"--w" are edges. The new graph has 2n vertices. I believe that it is impossible to produce a discrete partition without fixing at least n vertices,

since any pair {v',v"} which is not fixed cannot be separated.

1. **For Mathon doubling** by subroutine MATHON2: R. Mathon, Sample graphs for isomorphism testing, in: Proc. 9th S.-E.conf. Comb. Graph Theory Comput. Boca - Raton, 1978: pp. 499–517 orR. Mathon, Sample graphs for isomorphism testing, Congressus Numerantium, 21(1978) 499-517. MATHON2 implements Mathon doubling. MATHON2 generates a new regular graph g2(v2,e2),|v2|=2n+2, on given graph g1(v1,e1), |v1|=n. g2 contains 2 copies g1'(v1',e1') and g1"(v1",e1") of g1 and two vertices with labels 1 and 2n+2. The labels of vertices in v1' are from 2 to n+1 and for vertces in v1" fron n+1 to 2n+2.Vertex 1 is connected to each vertex in v1' and vertex 2n+2 to each vertex in v1". Each vertex x inv1' with label lx (2:n+1) corresponds to a vertex y with label ly (n+1:2n+2). Each edge (v1,v2)in e1' that belongs to e1 remains in e1' with labels v1+1,v2+1 and each nonedge (v1,v2) is added to e2 asd (v1',v2'), v1'=v1+1 and v2'=v2+n. The vertex degree of g2 is n.
2. **Izb1,izb2** are the initial and final value of control variable of a loop that chooses a cell selector number in the interval izb1 ≤ izb2 according to some criterion, izb1,izb2=1,2,..,6**.** The serial number izb of cell selectors are (given a stable partition PI):

Izb=1, UK=mxnacs1, UK is the label of the cell of PI with maximal number (nbr) of non-trivially joined cells with length>1. If two cells have equal nbr then the cell with larger size is selected. This cell selector is mostly used since it is the most efficient for most of the graphs. It is the default cell selector;

Izb=2, UK= mxnacs2, UK is the label of the cell of PI with maximal number (nbr) of non-trivially joined cells with length>1. If two cells have equal nbr then the cell with smaller size is selected;

Izb=3, UK=mxvectchvl, UK is the cell of PI with maximal channel vector

Izb=4, UK=mxprchval, UK is the cell of PI with max channel;

Izb=5, UK=ICLMXBRCL, UK is the cell of PI (with length>1) with maximal number of cells (brcl) of the derived partition PI(x), x ∈PI cell with label UK. If two cells have equal brcl then the cell with lower label is selected.

IZB=6, UK=nink1a, UK is the cell of PI with minimal vector of relative degrees;

D) Enter the vertices of the start partition in the format I4,A1. Example:

If the partition is |1,2,3,4|5,6,...,25|26| then enter \*\*\*1-\*\*\*4|\*\*\*5-\*\*25|\*\*26#, where \* stands for a space. UNIT 2 is the input device (con-for console or file name). If the first read symbol is minus or # then the input partition is unit.

E) Entering new values of on ***mbrsyvp***, ***wth1, dpth1, br1a, br2a***, example: 3 2 2 2 1. These variables determines the width (***wth***) and depth (*dpth*) of a reduced regular search tree (procedure ***tree4***) that heuristically finds the generators and orbits of ***Aut(G,П).*** The *wth* variable takes values *wth1,wth1+1,…,wth1+br1a* (outer loop), and for each value of ***wth*** the variable ***dpth*** takes values ***dpth1, dpth1+1, …, dpth1+br2a*** (inner loop). For each couple (***wth, dpth***) the generators and orbits are found. The execution of these loops stops when the number of found orbits remains unchanged on ***mbrsyvp*** (usually=3) loop executions consecutively. The default value for ***(dpth1, br1a, br2a***) are (5,2,2,1) but for different graph families there are other values that give minimal running time: for example, (2,2,2,1) for ***pp16, pp25*** graphs and (5,1,4,1) for ***had*** graphs.

**EXPERIMENTAL RESULTS**

|  |  |  |  |
| --- | --- | --- | --- |
| Table for vsep4n,all graphs,18-Apr-16 ,VSEP4P\_PUB4 program,results in  D:\Fort77A\VSEP1\VSEP1\RESVSEP4N\RESVSEP4MOBSHT.dat | | | |
| Graph family | parameters | izb | Ttotal[secs] |
| blmalki | 4,5,2 ,1,1  (default=DEF) | 1 | 0.1716 |
| blmathon | DEF | 1 | 0.1248 |
| blmathondbl | DEF | 1 | 0.312 |
| bltnn | DEF | 1 | 3.97 |
| BLTNN | DEF | 4 | 3.31 |
| blcmz | DEF | 1 | 2.37 |
| blag | DEF | 1 | 0.71 |
| blchh | DEF | 1 | 2.19 |
| blcfi | DEF | 1 | 38.94 |
| blcfi | DEF | 4 | 36.77 |
| bllatin | DEF | 1 | 3.79 |
| bllattice | DEF | 1 | 0.99 |
| Blhypercubes3 | DEF | 1 | 0.25 |
| blpaley | DEF | 1 | 1.90 |
| blppsmall | DEF | 1 | 0.56 |
| Blpp16 | 3 3 2 4 1 | 1 | 7.28 |
| Blpp16 | 3 3 2 2 1 | 1 | 7.22 |
| Blpp16 | 3 3 2 1 1 | 1 | 7.19 |
| Blpp25 | 3 3 2 1 1 | 1 | 746.11 |
| Blpp27 | Default | 2 | 60.66 |
| Blpp27 | 3 5 2 1 1 | 2 | 57.60 |
| Blpp27 | 3 5 2 1 1 | 3 | 58.26 |
| Blpp27 | 3 4 3 4 2 | 1 | 68.53 |
| Blpp27 | 3 4 3 4 2 | 2 | 88.54 |
| Blpp27 | 2 4 3 4 2 | 2 | 70.76 |
| Blpp27 | 3 4 3 3 2 | 2 | 88.90 |
| Blpp27 | 3 4 3 4 1 | 2 | 60.31 |
| Blpp27 | 2 4 3 3 2 | 2 | 76.32 |
| Blpp27 | 2 4 3 3 1 | 2 | 68.62 |
| Blpp27 | 2 4 3 4 1 | 2 | 75.48 |
| Blpp27 | 2 4 3 2 1 | 2 | 69.28 |
| Blpp27 | 2 4 3 1 1 | 2 | 68.82 |
| Blpp49 | 3 5 2 2 1 | 1 | 505.00 |
| Blpp49 | 2 5 2 2 1 | 2 | 349.60 |
| Blpp49 | 2 5 2 2 1 | 1 | 365.10 |
| blhad | 4 5 2 1 1 | 1 | 293.28 |
| blhad | 4 5 2 1 1 | 3 | 278.40 |
| blhad | 3 6 2 1 1 | 3 | 281.36 |
| blsrg | 2 2 1 2 2 | 2 | 589.29 |
| BLREG | DEF | 1 | 18.47 |
| BLPG | DEF | 1 | 7.84 |
| BLPG | 3 4 2 1 1 | 1 | 7.80 |
| BLPG | 3 3 2 1 1 | 1 | 7.80 |
| BLMZ-AUG | DEF | 1 | 2.09 |
| BLMZ-AUG2 | DEF | 1 | 3.77 |
| TOTAL RUN TIME FOR ALL CASES IN THE TABLE | | | 4485.867 |

**REFERENCES**

[1] S.D. Stoichev, Vsep-New Heuristic and Exact Algorithms for Graph Automorphism Group Computation, (n.d.). arXiv:1007.1726v4 [cs.DS]

[2] B.D. McKay, A. Piperno, Nauty&Traces website, (n.d.). <http://pallini.di.uniroma1.it/>