BRIDES User Guide

April 2016 / version 1.0

Table of contents

BRIDES is a computer program that finds the number of Breakthroughs, Roadblocks, Impasses, Detours, Equals and Shortcuts (BRIDES) characterizing the evolution of an original network X into an augmented network Y (new nodes and edges are added to X in order to create Y). It is distributed either as an R source file making use of the *igraph* and *SDDE* packages (this R version is suitable for networks with less than 1000 nodes) or as a standalone C++ program suitable for large evolving networks. The *BRIDES* program and its source code are freely available at: https://github.com/etiennelord/BRIDES under the GPL3 license. Our program and the corresponding algorithms were designed and developed at Université de Montréal (Monteal, Canada) and Université du Québec à Montréal (Montreal, Canada).

1. Pre-requisites

1.1 R version

This version requires the R version 3.2 or higher. It has been tested on the MacOSX, Windows and Linux platforms. It also requires the installation of the two following R packages (including their dependencies): *SDDE* (https://cran.r-project.org/web/packages/SDDE), and *igraph* (https://cran.r-project.org/web/packages/igraph).

1.2 C++ version

The C++ version is distributed as a C++ source code with different MakeFile(s) for different operating systems. The compilation of the source code has been tested on the MacOSX 10.10, Windows and Linux platforms using the gcc compiler suite. Our program uses the OpenMP[1] directives for parallel computing when available.

1.3 Objectives

Both the R and C++ functions can be used to estimate the number of different pathways, including Breakthroughs, Roadblocks, Impasses, Detours, Equals and Shortcuts (BRIDES), characterizing the evolution of an original network X into an augmented network Y (new nodes and edges have are to X in order to create Y). Note that all nodes of X should be also present in Y, but some edges of X may be absent in Y. These pathways, found in the network Y, must contain at least one node that was absent in the original network X.

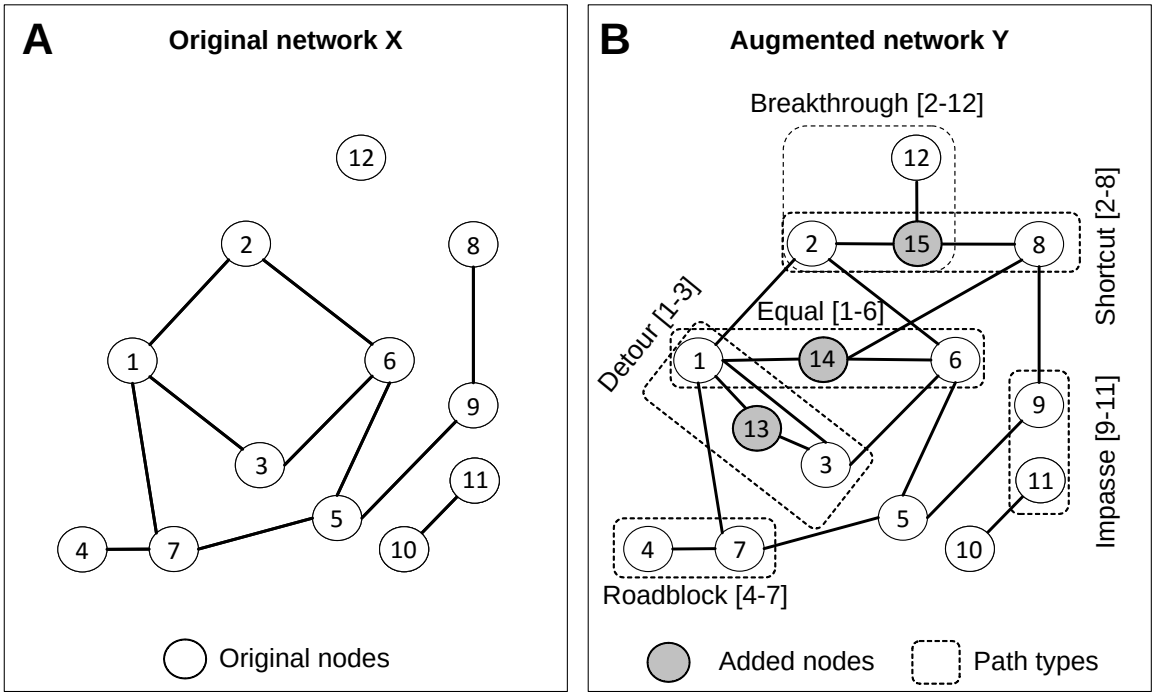
We can define the six following types of pathways, related to the existence of added nodes in Y, which can be used to characterize complex relationships in evolving networks (see Figure 1 and Table 1):

Breakthrough: pathway that is impossible in network *X* but possible in network *Y* (e.g. path between nodes 2 and 12, passing by added node 15 in Figure 1B);
Roadblock: pathway that is possible in network *X* but impossible in network *Y* (e.g. a simple path between nodes 4 and 7 that passes by an added node in *Y* is impossible, see Figure 1B);
Impasse: pathway that is impossible in both networks, *X* and *Y* (e.g. there is no path between nodes 9 and 11 in Figures 1A and 1B);
Detour: pathway that is shorter in network *X* than in network *Y* (e.g. path between nodes 1 and 3 in Figures 1A and 1B);
Equal: pathway that has the same length in networks *X* and *Y* (e.g. path between nodes 1 and 6, assuming that all edge lengths in *X* and *Y* are equal, Figures 1A and 1B);

[1] http://openmp.org/wp/

Shortcut: pathway that is longer in network *X* than in network *Y* (e.g. path between nodes 2 and 8 in Figures 1A and 1B).

Figure 1. Examples of the BRIDES (Breakthrough, Roadblock, Impasse, Detour, Equal and Shortcut) paths in evolving networks. Panel (A) presents an original network X with 12 nodes. Panel (B) presents an augmented network Y with 15 nodes (including 12 original and 3 added nodes, which are colored in grey).



These networks correspond to the files sampleX.txt (A) and sampleY.txt (B) which can be found in the sample directory.

Table 1. Possible BRIDES outcomes depending on the length of simple path.

Simple path between i to j in X	***Simple path between i to j in Y***	***BRIDES statistic***
Impossible	Possible	Breakthrough
Possible	Impossible	Roadblock
Impossible	Impossible	Impasse
Shorter distance	Longer distance	Detour
Equal distance	Equal distance	Equal
Longer distance	Shorter distance	Shortcut

2. Quickstart

2.1 Installation of the R version

Download or clone the git distribution found at: https://github.com/etiennelord/BRIDES.

```
git clone https://github.com/etiennelord/BRIDES.git
```

The BRIDES.R version requires the installation of the *SDDE* and *igraph* packages.

In the R terminal, in the R_src directory of the BRIDES repository, execute the following commands to install both packages:

```
install.packages("igraph")
install.packages("SDDE")
```

Then, execute the following command from the directory containing the BRIDES.R archive (R_src):

```
source("BRIDES.R")
```

2.2 Installation of the C++ version

Download the C++ source file (`main.cpp`) and the corresponding `Makefile` from Github. Then using the gcc compiler (version 4.5 or higher), compile the source code using the command make. Different Makefile(s) are provided.

```
make –f Makefile.Linux

//or

g++ -O3 main.cpp –o brides -fopenmp
```

Note, that under MacOSX, you will need to install the gcc compiler with an OpenMP support, using Homebrew (http://brew.sh/). For installation details, see the documentation available at: http://openmp.org/wp/openmp-compilers/ for more information.

```
brew install gcc --without-multilib
```

Otherwise, you can use the Makefile without OpenMP (Makefile.MacOSX_wo_Openmp).

This will either generate the brides.exe (under Windows) or brides (under Mac OSX and Linux) executables.

2.3 Input files

For both R and C++ versions, the edge-list network representation is required in the program input. Additional annotation files containing nodes identification can also be supplied. The program requires as input either an original network X and an augmented network Y, or a network X and an annotation file identifying each the “color” or attribute of each node.

An input network file should be given as an edge-list in a tab-separated (\t or tab) format:

Source	Destination	Edge

node id	node id	weight

**Note* that that each node can either be identified using a character string or a number. Single nodes are allowed in the edge-list. Rows beginning with the characters # or ! are ignored.

An input annotation file must be in the following, tab-separated, format:

Node id	color or attributes	…

Example of a network file:

```
#src dest distance
x1   x2   1
x1   x3   2
x12
```

Example of an annotation file:

```
#ID    Attributes
x1     2
x2     1
x3     1
x12    3
```

2.3.1 Input under the R version

The BRIDES program in R uses the *igraph* network manipulation functions and objects. For example, to load edge-list encoded networks, one can use the `load_network` function as follows (using the t0.txt and t1.txt sample files from the examples directory, see section 3.1):

```
source("BRIDES.R")
networkX<-load_network("sample/t0.txt")
networkY<-load_network("sample/t1.txt")
results<-BRIDES(networkX,networkY)
print(results)
```

2.3.2 Input under the C++ version

Specify on the command line both desired networks (see section 3.2) as follows:

```
brides.exe -X=sample/t0.txt -Y=sample/t1.txt
```

2.4 Output files

For the R version, an output file can be specified using the "outfile" parameter (see section 3.1). For the C++ version, and output file can be specified using the "-outfile=file" parameter (see section 3.2). Note that the detailed path information is only available in this "outfile".

2.4.1 R output file

For the *R version*, the output file is a tab-separated list (tsv):

Source node (src)	Destination node (dest)	Original path length (dist_x)	Augmented path length (dist_y)	BRIDES	Path Visited (path)	Path of visited taxa (taxa)

```
#src   dest   Dist_X Dist_Y BRIDES       path                taxa
x2     x1     1      1      Roadblock
x4     x1     2      4      Detour       x4,x3,y6,x2,x1       0,0,1,0,0
x6     x1     Inf    Inf    Impasse
x5     x1     3      5      Detour       x5,x4,x3,y6,x2,x1    0,0,0,1,0,0
x3     x1     3      3      Equal        x3,y6,x2,x1          0,1,0,0
x4     x2     1      3      Detour       x4,x3,y6,x2          0,0,1,0
...
```

2.4.2 C++ output file

For the *C++ version*, the output file results follow the following format:

(Run parameters)

```
...
============================ PARTIAL RESULTS ================================
src    dest   Dist_X Dist_Y BRIDES CPU time (ms)path               taxa
x2     x1     1      1      R      0
x4     x1     2      4      D      0            x4,x3,y6,x2,x1      0,0,1,0,0
x5     x1     3      5      D      0            x5,x4,x3,y6,x2,x1   0,0,0,1,0,0
x3     x1     3      3      E      0            x3,y6,x2,x1         0,1,0,0
x6     x1     Inf    Inf    I      0
x4     x2     1      3      D      0            x4,x3,y6,x2         0,0,1,0
...
```

An output file section contains statistics for individual nodes, either the original node (NK) or the added nodes (K). The column *inside* refers to the number of times that the related node was part of the other path(s). For example, node `x2` was part of 5 other simple paths, while node `y6` was part of 8 other simple paths in the augmented network. The last column, *attribute*, refers to the node information (if any).

Node	B	R	I	D	E	S	Inside	Atribute

```
============================== STATISTICS ====================================
NK     B      R      I      D      E      S      Inside  Attribute
x1     0      1      1      2      1      0      0       0
x2     0      1      1      2      1      0      5       0
x4     0      1      1      3      0      0      3       0
x5     0      1      1      3      0      0      0       0
x3     0      0      1      2      2      0      4       0
x6     0      0      5      0      0      0      0       0

K      Inside Attribute
y6     8      1
============================== RESULTS ===================================
       B      R      I      D      E      S      Total  Time (s)
       0      2      5      6      2      0      15     0
```

3. Program parameters

3.1 R function parameters

```
BRIDES(X=networkX, Y=networkY)
```

Parameters	Default value	Description
X	NA	Original network X (an *igraph* network object)
Y	NA	Augmented network Y (an *igraph* network object)
A	"default"	Attribute which can be specified for an added node (K) "tax" attribute: e.g. V(Y)$tax
src	"default"	Name or vertex number (source)
dest	"default"	Name or vertex number (destination)
random	0	If specified, will sample XX random pathways, default 100
maxdistance	100	Maximum distance to an added node, default 100
maxnode	100	Maximum number of added nodes (K) to consider, default 100
maxcores	1	Maximum number of threads to use
outfile	None	If specified, will print the characteristics of each path into outfile
size	1000	Group size
first	1	First group computed
last	1	Last group computed

3.1.1 Example of a BRIDES network analysis in R

```
source("BRIDES.R")
networkX<-load_network("sample/t0.txt")
networkY<-load_network("sample/t1.txt")
results<-BRIDES(networkX,networkY)
print(results)
    B      R      I      D      E      S (utime stime)
 0.00   2.00   5.00   6.00   2.00   0.00   0.07   0.95
```

3.1.2 Searching for a specific path between the source and destination nodes

Using the R version of BRIDES, the user can specify the source and destination nodes. For example, using the networks from the files t0.txt and t1.txt one can examine the augmented path from node x1 to node x3:

```
BRIDES(networkX,networkY,src="x1",dest="x3")
...
$from
[1] "x1"
$to
[1] "x3"
$path_type
[1] "Equal"
$path_type0
[1] 5
$original_path_length
[1] 3
$augmented_path_length
[1] 3
$path
[1] "x1" "x2" "y6" "x3"
$path_visited_taxa
[1] 1 1 2 1
```

The results are returned as a list containing:

Identification	Description
`from`	Source node
`to`	Destination node
`path_type`	BRIDES statistics (Breakthrough, Roadblock, Impasse, Detour, Equal or Shortcut)
`path_type0`	BRIDES statistics as a number (1=B, 2=R, 3=I, 4=D, 5=E, 6=S)
`original_path_length`	The path length in original network X
`augmented_path_length`	The path length containing an added node in augmented network Y
`path`	The visited path
`path_visited_taxa`	The attributes (V(g)$tax) associated with each visited nodes

3.1.3 Sampling 100 random paths in a Erdős–Rényi random network

The user can specify the number of paths to investigate in a network. For example, he/she can examine at most 100 paths each time:

```
#Random network with 20 original nodes and 5 added nodes
g=random_network(20,5,model="erdos")
BRIDES(g$g1,g$g2,random=100)
```

* Note: the random_network function is part of the *SDDE* package.

3.1.4 Limit the distance to added nodes in network Y using the maxdistance parameter

The user can limit the distance to added nodes (this distance equals 2 in the example below):

```
BRIDES(g$g1,g$g2, maxdistance=2)
```

3.1.5 Running BRIDES on groups 1 to 4 with a group size of 10 pathways

If the workload needs to be distributed on a cluster, the group size (*size*, referring to a number of path evaluated), the starting group (*first*) and the ending group (*last*) can be specified. For example, one can use a total of 10 paths and iterate from group 1 to 4 (40 paths evaluated).

```
BRIDES(g$g1,g$g2, size=10, first=1, last=4)
```

3.1.6 Default analysis with an attribute file and directed network

```
source("BRIDES.R")
# Specify an attributes files and that the network is directed
dX<-load_network("sample/U0.txt","sample/U0.attr.txt",directed=T)
# Specify which nodes are the added nodes (K).
results<-BRIDES(dX,A="2")
```

3.1.7 Output path details into file

```
BRIDES(dX,A="2", outfile="results.txt")
```

3.2 Command-line and options for the standalone C++ version

Base command-line

```
brides.exe –X=networkX.txt –Y=networkY.txt
```

Command-line options

```
Parameters:
-X=file            [filename for original network X]
-Y=file            [filename for augmented network Y]
-A=file            [filename for node attributes]
-usedist           [use edge distances found in network files]
-directed          [specifies that the networks are directed]
-K=B,C             [attributes to consider as added nodes, e.g. B,C]
-NK=A              [attributes to consider as original nodes, e.g. A]
-random=XX         [sample XX random pathways]
-first=1           [first group of paths to process]
-last=n            [last group of paths to process]
-size=1000         [group size, default: 1000]
-maxdistance=100   [maximum distance to an added node (K), default   : 100]
-maxnodes=100      [maximum number of added nodes to examine, default: 100]
-maxpathnumber=100 [maximum shortest-path returned by Dijkstra or Yen: 100]
-maxthread=XXX     [maximum number of OpenMP threads to use, default unlimited]
-outfile=file      [output path information into file: taxa, distance, etc.]
-seed=999          [set the random seed generator to a specific seed]
-strategy=1        [set the K nodes ordering strategy: (1) maxdist,(2) sum.]
-heuristic=1       [1-BRIDES, 2-BRIDES_Y, 3-BRIDES_YC, 4-BRIDES_EC, 5-DFS]
```

3.2.1 Default analysis

```
brides.exe –X=sample/sample_X.txt –Y=sample/sample_Y.txt
```

3.2.2 Randomly sample 10 paths with a specified seed

```
brides.exe –X=sample/t0.txt –Y=sample/t1.txt –random=10 –seed=1
```

3.2.3 Execute heuristic algorithm BRIDES_Y (2)

```
brides.exe –X=sample/t0.txt –Y=sample/t1.txt –heuristic=2
```

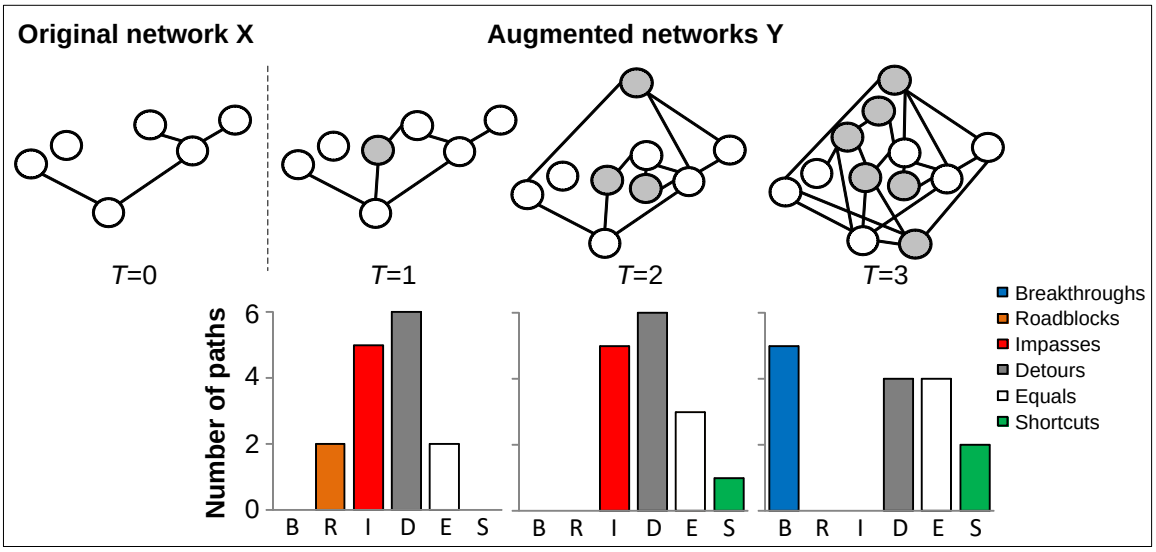
3.2.4 Output path details into a file

```
brides.exe –X=sample/t0.txt –Y=sample/t1.txt –outfile=results.txt
```

3.2.5 Use an attribute file and directed network

```
brides.exe –X=sample/U0.txt –A=sample/U0.attr.txt -K=2 -directed
```

4 Sample Network files (t0.txt, t1.txt,t2.txt,t3.txt)




```
##This analysis can be carried out using the following commands:

brides.exe -X=sample/t0.txt -Y=sample/t1.txt -outfile=result_t1.txt

brides.exe -X=sample/t0.txt -Y=sample/t2.txt -outfile=result_t2.txt

brides.exe -X=sample/t0.txt -Y=sample/t3.txt -outfile=result_t3.txt
```

* See the next page for an example of output file.

Commented Results file: `Result_t1.txt`

```
============================== PARAMETERS ===================================
-=[Multicores support]=-
Maximum threads  : 8
Number of cores  : 8

-=[Input files]=-
Networks         : undirected
NetworkX         : t0.txt     <- NetworkX
NetworkY         : t1.txt     <- NetworkY
Nodes in networkX: 6
Nodes in networkY: 7
Total added nodes: 1          <- One added node (K)
Total paths      : 15         <- 15 pathways to investigate

-=[Run parameters]=-
Running mode     : normal      <- Either normal or random
Edge distance    : unweighted  <- Either weighted or unweighted
Group size       : 1000        <- Group size
Number of groups : 1
First group      : 1
Last group       : 1
Maxdistance      : 100
Maxnode          : 100
Maxtime (s)      : 100
Maxpathnumber    : 100

-=[Miscellaneous]=-
Seed             : clock time   <- Either clock, or specific seed
K nodes ordering : strategy 1   <- added nodes (K) ordering strategy
Heuristic        : BRIDES (1)   <- Heuristic
Output           : t0.txt.output.txt <- Output to t0.txt.output.txt file

============================ PARTIAL RESULTS ================================
Node1   Node2   Dist_X  Dist_Y  Type    CPU time (ms)   Path    Taxa
x2      x1      1       1       R       0
x4      x1      2       4       D       0       x4,x3,y6,x2,x1  0,0,1,0,0
x5      x1      3       5       D       0       x5,x4,x3,y6,x2,x1 \
                                                              0,0,0,1,0,0
x3      x1      3       3       E       0       x3,y6,x2,x1     0,1,0,0
x6      x1      Inf     Inf     I       0
x4      x2      1       3       D       0       x4,x3,y6,x2     0,0,1,0
x5      x2      2       4       D       0       x5,x4,x3,y6,x2  0,0,0,1,0
x3      x2      2       2       E       0       x3,y6,x2        0,1,0
x6      x2      Inf     Inf     I       0
x5      x4      1       1       R       0
x3      x4      1       3       D       0       x3,y6,x2,x4     0,1,0,0
x6      x4      Inf     Inf     I       0
x3      x5      2       4       D       0       x3,y6,x2,x4,x5  0,1,0,0,0
x6      x5      Inf     Inf     I       0
x6      x3      Inf     Inf     I       0

================================ INFO =======================================
 (B) Breakthrough : pathway impossible in network X but possible in network Y
 (R) Roadblock    : pathway possible in network X but impossible in network Y
 (I) Impasse      : pathway impossible in both networks, X and Y
 (D) Detour       : pathway shorter in network X than in network Y
 (E) Equal        : pathway of same length in networks X and Y
 (S) Shortcut     : pathway longer in network X than in network Y
============================== STATISTICS ===================================
```

NK	B	R	I	D	E	S	Inside	Attribute
x1	0	1	1	2	1	0	0	0
x2	0	1	1	2	1	0	5	0
x4	0	1	1	3	0	0	3	0
x5	0	1	1	3	0	0	0	0
x3	0	0	1	2	2	0	4	0
x6	0	0	5	0	0	0	0	0

K	Inside	Attribute
y6	8	1

*Inside: number of times a node belongs to other paths; K: added nodes; NK: original nodes

========== RESULTS ==========

B	R	I	D	E	S	Total	Time (s)
0	2	5	6	2	0	15	0