FINDING CONNECTED COMPONENTS OF A GRAPH USING PERTURBATIONS OF ADJACENCY MATRIX

A.V. Prolubnikov

Omsk State University

a.v.prolubnikov@mail.ru

16.03.23

CONNECTIVITY PROBLEMS ON GRAPHS

GRAPH CONNECTIVITY

$G = \langle V(G), E(G) \rangle$ is undirected graph.
– G is *connected* if $\forall i, j \in V(G)$ there is a *chain* in G that connect i and j.
– *connected component* is a connected subgraph that is not part of any larger connected subgraph.

APPLICATIONS:

- Infrastructure reliability:
  - road network,
  - routing in networks (Internet),
  - development of large integrated circuits etc.
- applications for which we need to obtain solutions of *large* connectivity problems on graphs.

The large graphs are sparse: $|E(G)| = O(|V(G)|)$.

BREADTH-FIRST SEARCH

— the natural approach to test connectivity of a graph:
construct a rooted reachability tree for some vertex from $V(G)$

If *diameter* of G is ℓ then *it takes ℓ iterations to construct the tree.*

The computational complexity: $O(n+m)$.

- K. Zuse. Der Plankalkul, pp. 96–105 (2.47–2.56). Konrad Zuse Internet Archive, 1972. URL: http: //zuse.zib.de/item/gHI1cNsUuQweHB6.
- E.F. Moore. *The shortest path through a maze* // Proceedings of the International Symposium on the Theory of Switching. Harvard University Press, 1959, pp. 285–292.
- C.Y. Lee. *An Algorithm for Path Connections and Its Applications* // IRE Transactions on Electronic Computers. 1961.

ALGEBRAIC BFS

— *a graph traversal implemented as computaions of the form:*

$$x^{(0)} = e_i, \quad x^{(k+1)} = Ax^{(k)}, \quad A \text{ is an } \textit{adjacency matrix}$$

— GraphBLAS et al. — *fast* realizations of algebraic BFS for **sparse** graphs.

The computational complexity: $O(n) - O(mn)$.

- H.M. Bucker, C. Sohr. *Reformulating a breadth-fist search algorithm on an undirected graph in the language of linear algebra* // Intern. Conf. on Math. and Compu. in Sci. and in Industry, 2014, pp. 33–35.
- M. Besta, F. Marending, E. Solomonik, T. Hoefler. *SlimSell: a vectorizable graph representation for breadth-first seach* // 2017 IEEE Intern. Paral. and Distr. Proc. Symp., 2017, pp. 32–41.
- P. Burkhardt. *Optimal algebraic Breadth-First Search for sparse graphs* // arXiv:1906.03113v4 [cs.DS]. 30 Apr 2021.

PERTURBATION OF AN ADJACENCY MATRIX

$A = A(G)$ is invertible modified adjacency matrix of G.

$$A' = A + \varepsilon E_i,$$

$(E_i)_{ii} = 1$, $(E_i)_{jk} = 0$, $j \neq i$, $k \neq i$: $\qquad a'_{ii} = a_{ii} + \varepsilon$, $\varepsilon > 0$

$$\Rightarrow \; G \to G + \varepsilon(i, i)$$

- perform *perturbation* of a diagonal element of A;
- find connected component of a graph, considering changing of A^{-1} entries.

PERTURBATION PROPAGATE WITHIN CONNECTED COMPONENT:

$$(A^{-1})_{ij} = (-1)^{i+j} \cdot \frac{\det A_{ij}}{\det A} = (-1)^{i+j} \cdot \frac{\det A_{1,ij} \det A_2}{\det A_1 \det A_2} = (-1)^{i+j} \cdot \frac{\det A_{1,ij}}{\det A_1},$$

where A_1, A_2 are matrices of connected components.

MODIFICATION OF ADJACENCY MATRIX

Non-oriented graph: $(i,j) \in E(G) \Leftrightarrow (j,i) \in E(G)$

Modification of adjacency matrix

$A_0(G)$ is adjacency matrix of G.

$$A(G) = A_0(G) + d\mathcal{I},$$

where $\mathcal{I}$ is identity matrix,

$$d > \max_{i \in V(G)} d_i,$$

d_i is degree of vertex $i \in V(G)$.

$A(G)$ is a positive definite matrix
with strong diagonal predominance.

GRAPH ISOMORPHISM PROBLEM

$G = \langle V(G), E(G) \rangle$, $H = \langle V(H), E(H) \rangle$ are simple graphs.
$A(G)$, $B(G)$ are matrices of the graphs.

Isomorphism:

$$G \simeq H \Leftrightarrow$$
$$\Leftrightarrow \ \exists \varphi : V(G) \to V(H) \ : \ \Big((i,j) \in E(G) \Leftrightarrow (\varphi(i), \varphi(j)) \in E(H) \Big)$$

Theorem

$G \simeq H \Leftrightarrow$ the *consisted perturbations* may be implemented:

$$\det A^{(i)} = \det B^{(j_i)}, \ i = \overline{1, n},$$

$$A^{(0)} = A, \ B^{(0)} = B, \ A^{(i)} = A^{(i-1)} + \varepsilon_i E_i, \ B^{(i)} = B^{(i-1)} + \varepsilon_{j_i} E_{j_i}.$$

MODIFIED CHRACTERISTIC POLYNOMIAL

Characteristic polynomial:

$$\chi_G(x) = \det(A(G) - x\mathcal{I}).$$

Modified characteristic polynomial:

$$\eta_G(x_1, \dots, x_n) = \det(A(G) + X),\ X = \mathrm{diag}(x_1, \dots, x_n).$$

Theorem

$$G \simeq H \text{ and } \varphi\colon V(G) \to V(H) \Leftrightarrow \eta_G(x_1, \dots, x_n) \equiv \eta_H(x_{\varphi(1)}, \dots, x_{\varphi(n)}).$$

Comparison of A^{-1} elements equivalent to the comparison of the modified characteristic polynomials values at points $\varepsilon^{(i)} \in \mathbb{R}^n$:

$$\varepsilon^{(1)} = (\varepsilon_1, 0, 0, \dots, 0),\ \varepsilon^{(2)} = (\varepsilon_1, \varepsilon_2, 0, \dots, 0),\ \varepsilon^{(3)} = (\varepsilon_1, \varepsilon_2, \varepsilon_3, \dots, 0), \dots :$$

$$\eta_G(\varepsilon^{(i)}) = \eta_H(\varepsilon_{\varphi}^{(i-1)} + \varepsilon_i e_j), \quad i = \overline{1, n}.$$

MODIFIED CHRACTERISTIC POLYNOMIAL

The modified characteristic polynomials for graphs on $n = 1, 2, 3$ vertices:

- $n = 1$:

  $\eta_1 = x_1$;

- $n = 2$:

  $\eta_1 = x_1 x_2$,
  $\eta_2 = x_1 x_2 - 1$;

- $n = 3$:

  $\eta_1 = x_1 x_2 x_3$,
  $\eta_2 = x_1 x_2 x_3 - x_1$,
  $\eta_3 = x_1 x_2 x_3 - x_1 - x_3$,
  $\eta_4 = x_1 x_2 x_3 - x_1 - x_2 - x_3 + 2$.

THE APPROACH TO THE GRAPH ISOMORPHISM PR.

- R.T. Faizullin, A.V. Prolubnikov *An algorithm of the spectral splitting for the double permutation cipher* // Pattern Recognition and Image Analysis. 2002. Vol. 12, No. 4. P. 365–375.
- A.V. Prolubnikov. *Reduction of the graph isomorphism problem to equality checking of n-variables polynomials* //Trudy Instituta Matematiki i Mekhaniki UrO. RAN Proceedings of Krasovskii Institute of Mathematics and Mechanics UB RAS. 2016. T. 22, №1. C. 235–240. *(in Russian)*
- A.V. Prolubnikov. *Precision and complexity of computations that is needed to solve the graph isomorphism problem using its reduction to equality checking of n-varaibles polynomials* // Computational Technologies. 2016. T. 21, № 6. C. 71–88. *(in Russian)*
- A.V. Prolubnikov. Reduction of the graph isomorphism problem to equality checking of n-variables polynomials and the algorithms that use the reduction // arXiv.org, 2016.

  http://arxiv.org/pdf/1512.03139.pdf

RESPONSE TO A PERTURBATION

Implementing perturbations of graph matrix A we analyze the response on it in A^{-1} entries

Finding rows (columns) of A^{-1}:

Solve SLAE $Ax = e_i$. x is the i-th column of A^{-1}

How to check that the vertices belong to the same connected component:

1). Solve SLAE

$$Ax = e_i. \quad (1)$$

2). Implement the perturbation: $A' = A + \varepsilon E_i$.

3). Solve SLAE

$$A'x' = e_i. \quad (2)$$

4). Comparison: $x_j \overset{?}{\neq} x'_j$ — **yes** $\Rightarrow$ i and j belong to the same connected component, **no** $\Rightarrow$ they are not.

THE COMPONENTS OF THE SLAE'S SOLUTION

$$x_j = x_j(0) = \frac{A_{ij}}{\det A(0)} = (-1)^{i+j} \cdot \frac{\eta_{G_{ij}}(0, \ldots, 0)}{\eta_G(0, \ldots, 0)},$$

$$x_j' = x_j(\varepsilon) = \frac{A_{ij}}{\det A(\varepsilon)} = (-1)^{i+j} \cdot \frac{\eta_{G_{ij}}(0, \ldots, 0)}{\eta_G(\varepsilon e_i)}.$$

We have: $\eta_G(0, \ldots, 0) \neq \eta_G(\varepsilon e_i), \ \varepsilon > 0.$

Checking the inequality $x_j \neq x_j'$:

$$x_j \neq x_j' \ \Leftrightarrow \ x_j(0) \neq x_j(\varepsilon) \Leftrightarrow \eta_{G_{ij}}(0, \ldots, 0) \neq 0$$

$\eta_{G_{ij}}(0, \ldots, 0) \neq 0 \ \Leftrightarrow i, j$ belong to the same connected component

ALGORITHM 1

Algorithm 1 (G)

$V \leftarrow V(G)$;
$K \leftarrow 1$;
$V_K \leftarrow \varnothing$;
while $V \neq \varnothing$:
    choose $i \in V$;
    $V_K \leftarrow V_K \cup \{i\}$; $V \leftarrow V \setminus \{i\}$;
    solve SLAE (1), the solution is x;
    solve SLAE (2), the solution is x';
    for $\forall j \in V$:
        if $x'_j \neq x_j$
            $V_K \leftarrow V_K \cup \{j\}$; $V \leftarrow V \setminus \{j\}$;
    $K \leftarrow K + 1$;

Output: V_k, $k = \overline{1, K}$, are connected components of G.

JUSTIFICATION OF THE ALGORITHM 1

Proposition 1.

$\det A_{ij} = 0$ iff $i, j \in V(G)$ belong to different connected components of G.

If $i, j \in V(G)$ belong to the same connected component then

$$|x_j - x_j'| = \frac{\varepsilon\,|\det A_{ij}|\det A_{ii}}{\det A(\det A + \varepsilon \det A_{ii})} \geq \Delta > 0.$$

Proposition 2.

If $i, j \in V(G)$ belong to the same connected component then

$$\Delta > \frac{\varepsilon}{d^{n+1}} = \frac{10}{d^n},$$

if $\varepsilon = 10d$.

The worst case: G is a simple chain, $\ell(i,j) = n$.

ACCURACY AND COMPLEXITY OF COMPUTATIONS

For iterative methods (simple iteration, Gauss-Seidel):

$$\left|x_j^{(k+1)} - x_j^{(k)}\right| \leq \left\|x^{(k+1)} - x^{(k)}\right\| < \frac{\Delta_0}{\mu^k},$$

where Δ_0 is accuracy of the initial approximation, μ: $d = \mu d_{\max}$.

It takes N iterations of the methods to fix the inequality of exact values $x_j \neq x_j'$:

$$\frac{\delta_0}{\mu^N} < \frac{\Delta}{4}.$$

$$\Rightarrow \quad N > \log_\mu\left(\frac{4\delta_0}{\Delta}\right) = (n+1)\log_\mu d + \log_\mu(8\delta_0) \approx (n+1)\log_\mu d,$$

where $\log_\mu d = \log_\mu(\mu d_{\max}) = 1 + \log_\mu d_{\max} = 2$ при $\mu = d_{\max}$.

$N = O(n)$. Complexity of an iteration — $O(m)$.
$\Rightarrow$ overall complexity of the **Algorithm 1** — $O(nm)$.

THE COMPUTATIONAL COMPLEXITY OF VBFS

BFS implemented as a graph traversal:

the compuatational complexity — $O(m+n)$:

since during the traversal

- we bypass no more than m edges,
- we bypass n vertices.

We may handle only one level of reachability tree to avoid cycles.

$\Rightarrow$ **there is no BFS implementation with the computational complexity is less than $O(m+n)$.**

Inspite the computional complexity of implementations of algebraic BFS is $O(mn)$, it may be faster then traversal-implemented BFS with complexity $O(m+n)$ due to parallelization of computations at one level of reachability tree.

USING IMPLEMENTATIONS OF NUMERICAL METHODS FOR SLAE TO SOLVE CONNECTIVITY PROBLEMS:

- parallelization of BFS is difficult.

While

- there are effective parallel implementations of numerical methods for SLAE,

  including methods for sparse SLAE.

ITERATIVE NUMERICAL METHODS AND GRAPH TRAVERSALS

$x^{(0)} = e_i = (0, \ldots, 1, \ldots, 0)$.

1. Simple iteration method:

$$\text{BFS: } x^{(k+1)} = Ax^{(k)} \;\rightarrow\; x^{(k+1)} = b - D^{-1}Ax^{(k)}.$$

$$x_j^{(k+1)} = \frac{1}{a_{jj}}\left(b_j - \sum_{l \neq j} a_{jl}x_j^{(k)}\right), \quad b_j \in \{0, 1\}.$$

2. Gauss-Seidel Method:

$$(L + D)x^{(k+1)} = -Ux^{(k)} + b.$$

$$x_j^{(k+1)} = \frac{1}{a_{jj}}\left(b_j - \sum_{l=1}^{j-1} a_{jl}x_l^{(k+1)} - \sum_{l=j+1}^{n} a_{jl}x_l^{(k)}\right), \quad b_j \in \{0, 1\}.$$

CONVERGENCE TO THE EXACT SOLUTION

USING ITERATIVE METHODS FOR SLAE:

If we search for the exact solution, then it is not essential which method we use to solve SLAE in **Algorithm 1.**

But if the number of iterations is relatively small then the iterations may considered as graph traversals.

The traversal that defines by Gauss-Seidel method is differnet than the one of BFS.

Traversal: traverse from visted vertex to not visted yet vertex j is equiavalent to:

$$x_j^{(k)} = 0, \text{ but } x_j^{(k+1)} \neq 0.$$

ALGORITHM 2 (GSS)

Algorithm 2 $(G; i \in V(G)) : C;$

$x^{(0)} = e_i;\ C \leftarrow \{i\};$

$x^{(1)} = 0 \in \mathbb{R}^n;$

$k \leftarrow 1;$

while $\exists j \in V(G) : \left(x_j^{(k)} = 0 \text{ и } x_j^{(k+1)} \neq 0\right)$

  for $\forall j \in V(G)$:

$$x_j^{(k+1)} = a_{jj} \cdot \left(b_j - \sum_{l=1}^{j-1} a_{jl} x_l^{(k+1)} - \sum_{l=j+1}^{n} a_{jl} x_l^{(k)}\right),\ b_j \in \{0, 1\}.$$

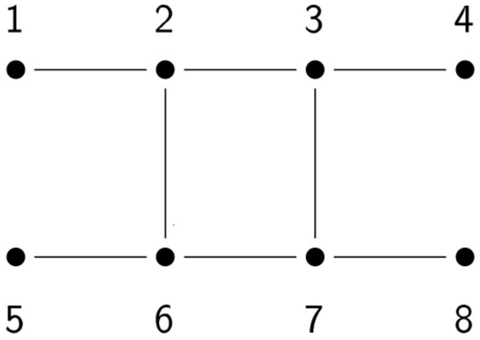
  $k \leftarrow k + 1;$

$C \leftarrow C \cup \{j : x_j^{(k)} \neq 0\}.$

Output: C is the set of vertices of the connected component, $i \in C$.

ITERATION OF BFS (SIS) FOR THE GRAPH:

The graph G:



$i = 1$

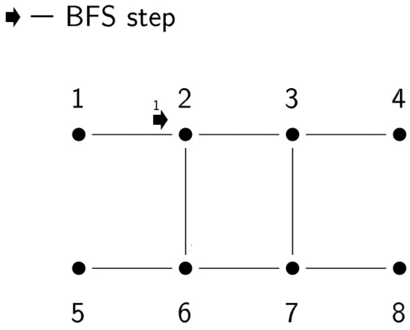
ITERATION OF BFS (SIS) FOR THE GRAPH:

- $x_1^{(k+1)} = \ldots - x_2^{(k)};$
- $x_2^{(k+1)} = \ldots - x_1^{(k)} - x_3^{(k)} - x_6^{(k)};$
- $x_3^{(k+1)} = \ldots - x_2^{(k)} - x_4^{(k)} - x_7^{(k)};$
- $x_4^{(k+1)} = \ldots - x_3^{(k)};$
- $x_5^{(k+1)} = \ldots - x_6^{(k)};$
- $x_6^{(k+1)} = \ldots - x_2^{(k)} - x_5^{(k)} - x_7^{(k)};$
- $x_7^{(k+1)} = \ldots - x_3^{(k)} - x_6^{(k)} - x_8^{(k)};$
- $x_8^{(k+1)} = \ldots - x_7^{(k)}.$

BFS. ITERATION 1.

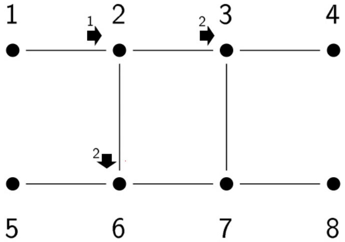
$x^{(0)} = e_i,\ x^{(k+1)} = Ax^{(k)}$.

$x_j^{(k-1)} \neq 0 \Leftrightarrow$ vertex j is reached before the k-th iteration.

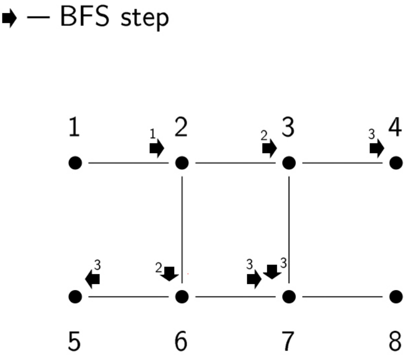


BFS. ITERATION 2.

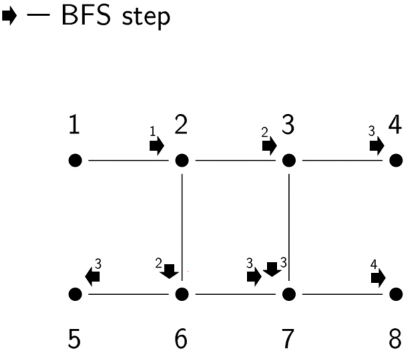
➧ — BFS step



BFS. ITERATION 3.



BFS. ITERATION 4.



AN ITERATION OF BFS FOR THE GRAPH:

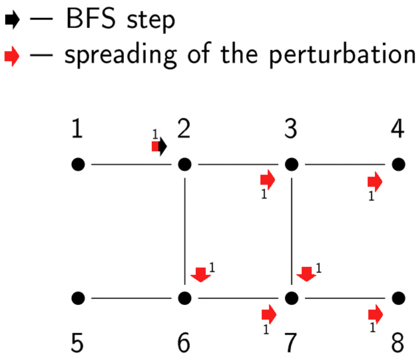
- $x_1^{(k+1)} = \ldots - x_2^{(k)};$
- $x_2^{(k+1)} = \ldots - x_1^{(k)} - x_3^{(k)} - x_6^{(k)};$
- $x_3^{(k+1)} = \ldots - x_2^{(k)} - x_4^{(k)} - x_7^{(k)};$
- $x_4^{(k+1)} = \ldots - x_3^{(k)};$
- $x_5^{(k+1)} = \ldots - x_6^{(k)};$
- $x_6^{(k+1)} = \ldots - x_2^{(k)} - x_5^{(k)} - x_7^{(k)};$
- $x_7^{(k+1)} = \ldots - x_3^{(k)} - x_6^{(k)} - x_8^{(k)};$
- $x_8^{(k+1)} = \ldots - x_7^{(k)}.$

AN ITERATION OF GSS FOR THE GRAPH:

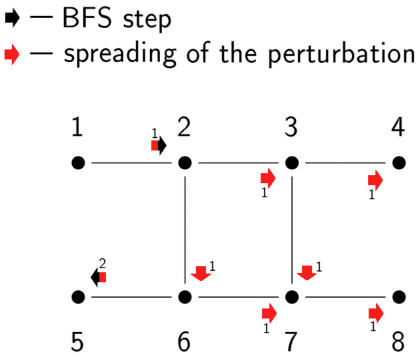
- $x_1^{(k+1)} = \ldots - x_2^{(k)};$
- $x_2^{(k+1)} = \ldots - x_1^{(k+1)} - x_3^{(k)} - x_6^{(k)};$
- $x_3^{(k+1)} = \ldots - x_2^{(k+1)} - x_4^{(k)} - x_7^{(k)};$
- $x_4^{(k+1)} = \ldots - x_3^{(k+1)};$
- $x_5^{(k+1)} = \ldots - x_6^{(k)};$
- $x_6^{(k+1)} = \ldots - x_2^{(k+1)} - x_5^{(k+1)} - x_7^{(k)};$
- $x_7^{(k+1)} = \ldots - x_3^{(k+1)} - x_6^{(k+1)} - x_8^{(k)};$
- $x_8^{(k+1)} = \ldots - x_7^{(k+1)}.$

THE FIRST ITERATION OF GSS

The perturbation of diagonal element a_{11} spreads through traverses over all **ordered chains** that originated in the vertices reached at the first iteration:

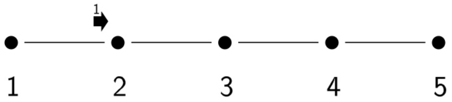


THE SECOND ITERATION OF GSS

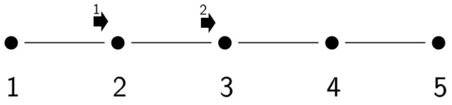


If the perturbation is delivered to a vertex,
then it starts another BFS traversal from the vertex at the next iteration.

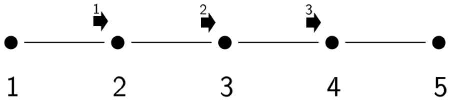
BFS. ITERATION 1.



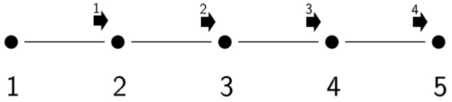
BFS. ITERATION 2.



BFS. ITERATION 3.



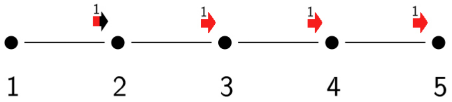
BFS. ITERATION 4.



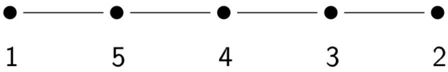
AN ITERATION GSS FOR THE GRAPH:

- $x_1^{(k+1)} = \ldots - x_2^{(k)};$
- $x_2^{(k+1)} = \ldots - x_1^{(k+1)} - x_3^{(k)};$
- $x_3^{(k+1)} = \ldots - x_2^{(k+1)} - x_4^{(k)};$
- $x_4^{(k+1)} = \ldots - x_3^{(k+1)} - x_5^{(k)};$
- $x_5^{(k+1)} = \ldots - x_4^{(k+1)}.$

GSS. ITERATION 1.

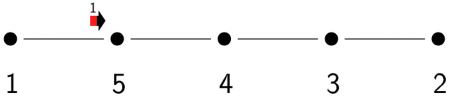


AN ITERATION GSS FOR THE GRAPH:

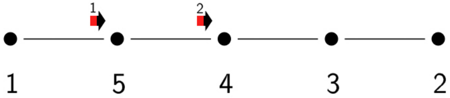


- $x_1^{(k+1)} = \ldots - x_5^{(k)}$;
- $x_2^{(k+1)} = \ldots - x_3^{(k)}$;
- $x_3^{(k+1)} = \ldots - x_2^{(k+1)} - x_4^{(k)}$;
- $x_4^{(k+1)} = \ldots - x_3^{(k+1)} - x_5^{(k)}$;
- $x_5^{(k+1)} = \ldots - x_1^{(k+1)} - x_4^{(k+1)}$.

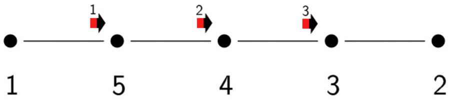
GSS. ITERATION 1.



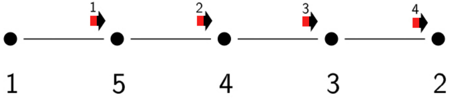
GSS. ITERATION 2.



GSS. ITERATION 3.



GSS. ITERATION 4.



SPARSE GRAPHS

- $n = 90\ 000,\ m = 89\ 100,\ K = 900.$

Connected components are chains on 100 vertices.

- Solving SLAE (Algorithm 1) — *simple iteration method:*
  $N = 110,\ d = 100$

  — **20 minutes**

- Solving SLAE (Algorithm 1) — *Gauss-Seidel method:*
  $N = 80,\ d = 1000$

  — **9 minutes**

- GSS (Algorithm 2):
  $d = 1$

  — **3 minutes**

SPARSE GRAPHS

Graph of a *problem* of connectivity of the transport network:

— the graph edges corresponds to entry of a variables
into equations and inequalities:

- Solving SLAE — *simple iteration method:*
- $n = 367\ 840,\ m = 53\ 404\ 685,\ m = 150n,\ K = 224.$

32 connected components with 11 429 vertices,
192 connected components with 11 vertices (chains) — **97 minutes**

BFS (not algebraic) ≈48 **hours**

THE COMPUTATIONAL COMPLEXITY OF ALG. 2 (GSS)

The complexity of one iteration is $O(m)$.

ℓ is diameter. The number of iterations in the worst case is ℓ.

$$\Rightarrow \text{overall complexity is } O(\ell m).$$

The number of iterations is determined by the numbering of vertices.

Proposition 3.

For every instance of the problem, the computaionial complexity of GSS (Algorithm 2) is not greater than the one of algebraic BFS.

The computational complexity of GSS is less than the one of BFS if we reach the vertices that belong to chain with maximum length

and from which the chains with the correct order are originated.

CONCLUSIONS

- We present an approach to solution of connectivity problems on graphs using perturbations of elements of the adjacency matrix.
- The approach allows to use effective numerical realizations of methods for SLAE to solve connectivity problems on graphs.
- We consider iterative methods of SLAE as realizations of graph traversals and present the algorithm of finding connected components of a graph which uses graph traverse that is not equivalent to the one of BFS.

  Its computational complexity is not greater then complexity of BFS for all instances of the problem.