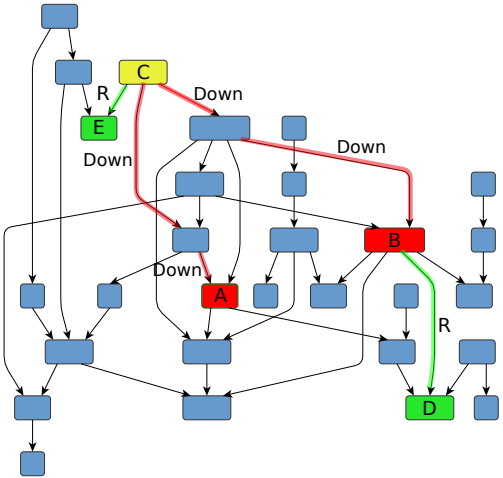
Multiple-Source Context-Free Path Querying in Terms of Linear Algebra

Arseniy Terekhov, Vlada Pogozhelskaya, Vadim Abzalov,
Timur Zinnatulin, **Semyon Grigorev**

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg State University

March 24, 2021

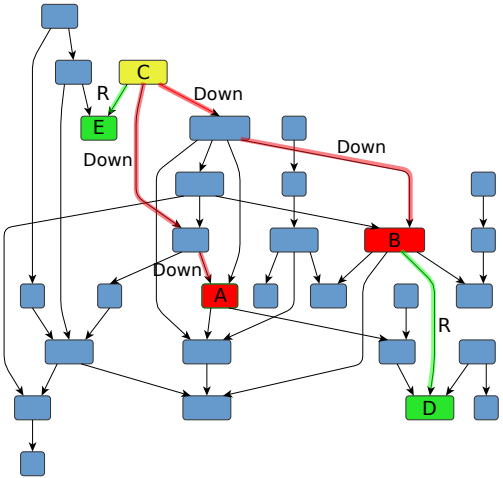
Formal Language Constrained Path Querying



Navigation through an edge-labeled graph

- **Path** specifies a **word** formed by labels of edges
- **Paths constraint** is a **language**: the path specified word should be in the given language
- Constraints expressiveness is related to **formal languages classes**

Formal Language Constrained Path Querying



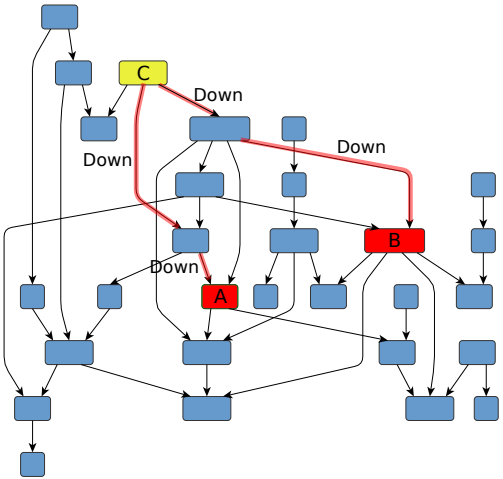
Navigation through an edge-labeled graph

- **Path** specifies a **word** formed by labels of edges
- **Paths constraint** is a **language**: the path specified word should be in the given language
- Constraints expressiveness is related to **formal languages classes**

Regular path queries (RPQ)

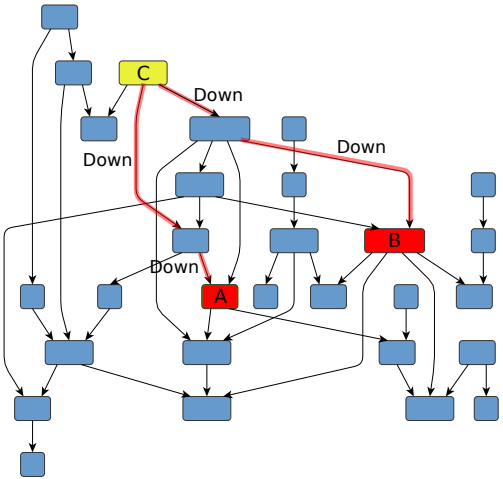
- **Regular** languages are used as constraints
- Which nodes are reachable from **C** by arbitrary number of **R** and **Down** edges?
- $\mathcal{L} = (R \mid Down)^*$

Context-Free Path Querying (CFPQ)



- **Context-free** languages are used as constraints
- Are nodes A and B on the same level of hierarchy?
- Is there a path of form $\overline{\textbf{Down}}^n \textbf{Down}^n$ between A and B?
- Context-free grammar:
  $sameLvl \to \overline{Down}\ sameLvl\ Down \mid \varepsilon$

Context-Free Path Querying (CFPQ)



- **Context-free** languages are used as constraints
- Are nodes A and B on the same level of hierarchy?
- Is there a path of form $\overline{\textbf{Down}}^n \textbf{Down}^n$ between A and B?
- Context-free grammar: $sameLvl \rightarrow \overline{Down}\ sameLvl\ Down \mid \varepsilon$

Applications

- Static code analysis [T. Reps, et al, 1995]
- Graph segmentation [H. Miao, et al, 2019]
- Biological data analysis [P. Sevon, et al, 2008]
- ...

Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

- J. Kuijpers, et al[1]: existing algorithms are too slow to be practical (in the context of Neo4j)

[1] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods.

Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

- J. Kuijpers, et al[1]: existing algorithms are too slow to be practical (in the context of Neo4j)
- A. Terekhov, et al[2]: linear algebra based CFPQ algorithm can be performant enough

[1] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods.

[2] Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, and Semyon Grigorev. 2020. Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication.

Problem Statement

There is no support of CFPQ in real-world graph analysis systems (graph databases)

- J. Kuijpers, et al[1]: existing algorithms are too slow to be practical (in the context of Neo4j)
- A. Terekhov, et al[2]: linear algebra based CFPQ algorithm can be performant enough
- There is no full-stack support of CFPQ
  - Grammars instead of full-featured queries
  - Custom graph storage instead of a mature graph database

[1] Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods.

[2] Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, and Semyon Grigorev. 2020. Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication.

Proposed Solution

- **Multiple-Source CFPQ** to process only required subset of a graph

Proposed Solution

- **Multiple-Source CFPQ** to process only required subset of a graph
- **Cypher** extended with **path patterns**[3] to express context-free constraints

[3] Tobias Lindaaker, Path Patterns for Cypher, 2017,
https://github.com/thobe/openCypher/blob/rpq/cip/1.accepted/CIP2017-02-06-Path-Patterns.adoc

Proposed Solution

- **Multiple-Source CFPQ** to process only required subset of a graph
- **Cypher** extended with **path patterns**[3] to express context-free constraints
- **RedisGraph** database
  - Provides graph storage with matrix-based representation
  - Contains linear algebra based query engine (SuiteSparse:GraphBLAS[4] is used)
  - Allows one to use Cypher for querying (libcypher-parser[5] is used)

[3] Tobias Lindaaker, Path Patterns for Cypher, 2017, `https://github.com/thobe/openCypher/blob/rpq/cip/1.accepted/CIP2017-02-06-Path-Patterns.adoc`

[4] Timothy A. Davis. 2019. Algorithm 1000: SuiteSparse:GraphBLAS: Graph Algorithms in the Language of Sparse Linear Algebra

[5] Chris Leishman, `https://github.com/cleishm/libcypher-parser`

Multiple-Source CFPQ

An improved version of Rustam Azimov CFPQ algorithm[6]

- The set of start vertices can be specified

[6] Rustam Azimov and Semyon Grigorev. 2018. Context-free path querying by matrix multiplication.

Multiple-Source CFPQ

An improved version of Rustam Azimov CFPQ algorithm[6]

- The set of start vertices can be specified
- Only required subgraph will be processed

[6] Rustam Azimov and Semyon Grigorev. 2018. Context-free path querying by matrix multiplication.

Multiple-Source CFPQ

1: **function** MultiSrcCFPQ($D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$, $G = (N, \Sigma, P, S)$, Src)
2: $T \leftarrow \{T^A \mid A \in N, T^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
3: $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
4: **for all** $v \in Src$ **do** $TSrc^S[v,v] \leftarrow \mathit{true}$
5: $MSrc \leftarrow TSrc^S$
6: **for all** $A \to x \in P \mid x \in \Sigma_E$ **do**
7: **for all** $(v, to) \in E \mid x \in \lambda_E(v, to)$ **do** $T^A[v, to] \leftarrow \mathit{true}$
8: **for all** $A \to x \in P \mid x \in \Sigma_V$ **do**
9: **for all** $v \in V \mid x \in \lambda_V(v)$ **do** $T^A[v,v] \leftarrow \mathit{true}$
10: **while** T *or* $TSrc$ *is changing* **do**
11: **for all** $A \to BC \in P$ **do**
12: $M \leftarrow TSrc^A * T^B$
13: $T^A \leftarrow T^A + M * T^C$
14: $TSrc^B \leftarrow TSrc^B + TSrc^A$
15: $TSrc^C \leftarrow TSrc^C +$ getDst(M)
16: **return** $MSrc * T^S$

Multiple-Source CFPQ

1: **function** MULTISRCCFPQ($D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$, $G = (N, \Sigma, P, S)$, Src)
2: $T \leftarrow \{T^A \mid A \in N, T^A[i,j] \leftarrow false, \text{for all } i,j\}$
3: $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i,j] \leftarrow false, \text{for all } i,j\}$
4: **for all** $v \in Src$ **do** $TSrc^S[v,v] \leftarrow true$
5: $MSrc \leftarrow TSrc^S$
6: **for all** $A \to x \in P \mid x \in \Sigma_E$ **do**
7: **for all** $(v, to) \in E \mid x \in \lambda_E(v, to)$ **do** $T^A[v, to] \leftarrow true$
8: **for all** $A \to x \in P \mid x \in \Sigma_V$ **do**
9: **for all** $v \in V \mid x \in \lambda_V(v)$ **do** $T^A[v,v] \leftarrow true$
10: **while** T *or* $TSrc$ *is changing* **do**
11: **for all** $A \to BC \in P$ **do**
12: $M \leftarrow TSrc^A * T^B$
13: $T^A \leftarrow T^A + M * T^C$
14: $TSrc^B \leftarrow TSrc^B + TSrc^A$
15: $TSrc^C \leftarrow TSrc^C + \text{GETDST}(M)$
16: **return** $MSrc * T^S$

Multiple-Source CFPQ

1: **function** $\textsc{MultiSrcCFPQ}(D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E),\ G = (N, \Sigma, P, S),\ Src)$
2: $\quad T \leftarrow \{T^A \mid A \in N, T^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
3: $\quad TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
4: $\quad$ **for all** $v \in Src$ **do** $TSrc^S[v,v] \leftarrow \mathit{true}$
5: $\quad MSrc \leftarrow TSrc^S$
6: $\quad$ **for all** $A \to x \in P \mid x \in \Sigma_E$ **do**
7: $\quad\quad$ **for all** $(v, to) \in E \mid x \in \lambda_E(v, to)$ **do** $T^A[v, to] \leftarrow \mathit{true}$
8: $\quad$ **for all** $A \to x \in P \mid x \in \Sigma_V$ **do**
9: $\quad\quad$ **for all** $v \in V \mid x \in \lambda_V(v)$ **do** $T^A[v,v] \leftarrow \mathit{true}$
10: $\quad$ **while** T *or* $TSrc$ *is changing* **do**
11: $\quad\quad$ **for all** $A \to BC \in P$ **do**
12: $\quad\quad\quad M \leftarrow TSrc^A * T^B$
13: $\quad\quad\quad T^A \leftarrow T^A + M * T^C$
14: $\quad\quad\quad TSrc^B \leftarrow TSrc^B + TSrc^A$
15: $\quad\quad\quad TSrc^C \leftarrow TSrc^C + \textsc{getDst}(M)$
16: $\quad$ **return** $MSrc * T^S$

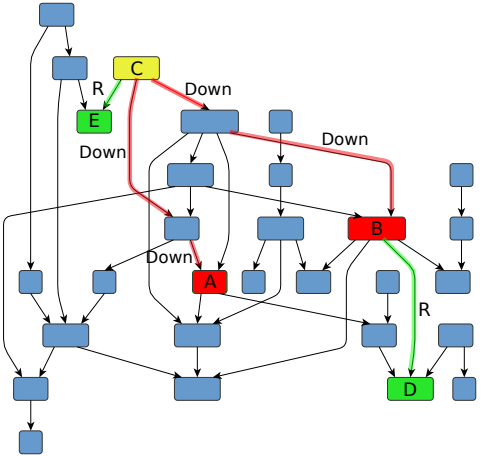
Multiple-Source CFPQ

1: **function** MULTISRCCFPQ($D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$, $G = (N, \Sigma, P, S)$, Src)
2: $T \leftarrow \{T^A \mid A \in N, T^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
3: $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
4: **for all** $v \in Src$ **do** $TSrc^S[v,v] \leftarrow \mathit{true}$
5: $MSrc \leftarrow TSrc^S$
6: **for all** $A \to x \in P \mid x \in \Sigma_E$ **do**
7: **for all** $(v, to) \in E \mid x \in \lambda_E(v, to)$ **do** $T^A[v, to] \leftarrow \mathit{true}$
8: **for all** $A \to x \in P \mid x \in \Sigma_V$ **do**
9: **for all** $v \in V \mid x \in \lambda_V(v)$ **do** $T^A[v,v] \leftarrow \mathit{true}$
10: **while** T *or* $TSrc$ *is changing* **do**
11: **for all** $A \to BC \in P$ **do**
12: $M \leftarrow TSrc^A * T^B$
13: $T^A \leftarrow T^A + M * T^C$
14: $TSrc^B \leftarrow TSrc^B + TSrc^A$
15: $TSrc^C \leftarrow TSrc^C +$ GETDST(M)
16: **return** $MSrc * T^S$

Multiple-Source CFPQ

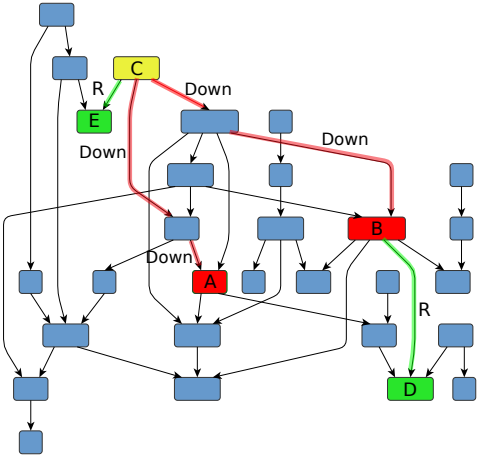
1: **function** MultiSrcCFPQ($D = (V, E, \Sigma_V, \Sigma_E, \lambda_V, \lambda_E)$, $G = (N, \Sigma, P, S)$, Src)
2: $T \leftarrow \{T^A \mid A \in N, T^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
3: $TSrc \leftarrow \{TSrc^A \mid A \in N, TSrc^A[i,j] \leftarrow \mathit{false}, \text{for all } i,j\}$
4: **for all** $v \in Src$ **do** $TSrc^S[v,v] \leftarrow \mathit{true}$
5: $MSrc \leftarrow TSrc^S$
6: **for all** $A \to x \in P \mid x \in \Sigma_E$ **do**
7: **for all** $(v, to) \in E \mid x \in \lambda_E(v, to)$ **do** $T^A[v, to] \leftarrow \mathit{true}$
8: **for all** $A \to x \in P \mid x \in \Sigma_V$ **do**
9: **for all** $v \in V \mid x \in \lambda_V(v)$ **do** $T^A[v,v] \leftarrow \mathit{true}$
10: **while** T *or* $TSrc$ *is changing* **do**
11: **for all** $A \to BC \in P$ **do**
12: $M \leftarrow TSrc^A * T^B$
13: $T^A \leftarrow T^A + M * T^C$
14: $TSrc^B \leftarrow TSrc^B + TSrc^A$
15: $TSrc^C \leftarrow TSrc^C +$ getDst(M)
16: **return** $MSrc * T^S$

Cypher Extension




```
MATCH (u)-[(R | Down)*]->(v)
RETURN u.name, v.name
```

Cypher Extension




```
MATCH (u)-[(R | Down)*]->(v)
RETURN u.name, v.name
```

Named path pattern

$SameLvl \rightarrow \overline{Down}\ SameLvl\ Down \mid \varepsilon$

```
PATH PATTERN SameLvl =
  ()-/ <:Down [~SameLvl | ()] :Down> /->()
MATCH (u)-/ ~SameLvl /->(v)
RETURN u.name, v.name
```

Implementation Details

- Linear algebra based multiple-source CFPQ is implemented as a part of RedisGraph query engine
- Cypher parser is extended to support path patterns
- Path patterns are partially supported[7] in RedisGreaph query execution workflow

[7] Full support is a nontrivial challenge: formal description of the extension is required

Evaluation Setup

- Ubuntu 18.04, Intel Core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM
- Graphs stored in RedisGraph with our extensions
- Queries are generated with template for given size of start set
- The union of all start sets is V

Evaluation Setup

- Ubuntu 18.04, Intel Core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM
- Graphs stored in RedisGraph with our extensions
- Queries are generated with template for given size of start set
- The union of all start sets is V

Graph	#V	#E	Q
core	1323	4342	g_1
pathways	6238	18 598	g_1
gohierarchy	45 007	980 218	g_1
enzyme	48 815	109 695	g_1
eclass_514en	239 111	523 727	g_1
geospecies	450 609	2 311 461	*geo*
go	272 770	534 311	g_1

Evaluation Setup

- Ubuntu 18.04, Intel Core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM
- Graphs stored in RedisGraph with our extensions
- Queries are generated with template for given size of start set
- The union of all start sets is V

Graph	#V	#E	Q
core	1323	4342	g_1
pathways	6238	18 598	g_1
gohierarchy	45 007	980 218	g_1
enzyme	48 815	109 695	g_1
eclass_514en	239 111	523 727	g_1
geospecies	450 609	2 311 461	*geo*
go	272 770	534 311	g_1

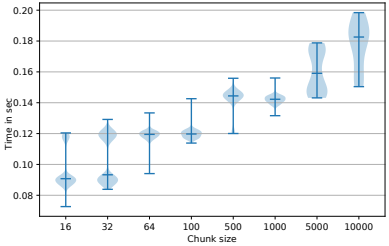
```
PATH PATTERN S =
  ()-/ [<:SubClassOf [~S | ()] :SubClassOf] | [<:Type [~S | ()] :Type] /->()
MATCH (src)-/ ~S /->()
WHERE {id_from} <= src.id and src.id <= {id_to}
RETURN count(*)
```

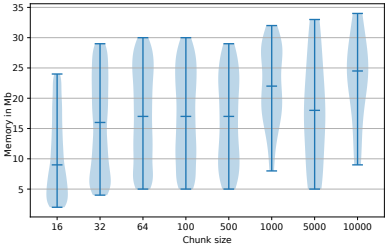
Evaluation Results

Time

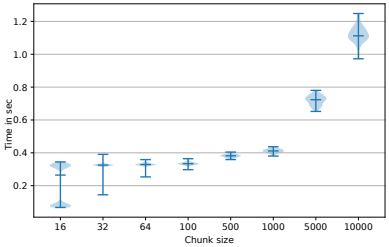
Memory

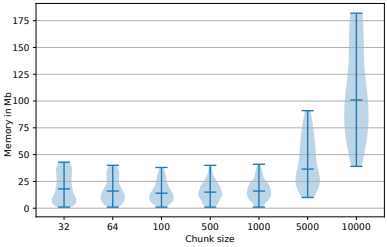
eclass_514en
Query: g1





geospecies
Query: geo





Conclusion

- Full-stack support for CFPQ in real-world graph query language (Cypher) on the top of real-world graph database (RedisGraph)
  - No more context-free grammars
  - No more custom graph formats and storages
- Reasonable performance of context-free path queries
  - Multiple-source scenario
  - Space-time ratio can be tuned
- Context-free path queries can be used in applications with well-established tools

Future Research

- Mechanization of Cypher semantics in Coq
  - Including path patterns
  - Correctness of translation to linear algebra

Future Research

- Mechanization of Cypher semantics in Coq
  - Including path patterns
  - Correctness of translation to linear algebra
- Integration of tensor-based CFPQ algorithm[8] to RedisGraph
  - To construct paths, not only reachability facts
  - The algorithm should be modified to get multiple-source version

[8] Egor Orachev, Ilya Epelbaum, R. Azimov and S. Grigorev. "Context-Free Path Querying by Kronecker Product." ADBIS (2020).

Future Research

- Mechanization of Cypher semantics in Coq
  - Including path patterns
  - Correctness of translation to linear algebra
- Integration of tensor-based CFPQ algorithm[8] to RedisGraph
  - To construct paths, not only reachability facts
  - The algorithm should be modified to get multiple-source version
- Detailed evaluation
  - More graphs and queries, including RPQs
  - Scalability of the solution
  - Comparison with other graph query engines

[8] Egor Orachev, Ilya Epelbaum, R. Azimov and S. Grigorev. "Context-Free Path Querying by Kronecker Product." ADBIS (2020).

Contact Information

- Try it out (Docker image with extended RedisGraph):
  `https://hub.docker.com/r/simpletondl/redisgraph`
- RedisGraph extended with CFPQ:
  `https://github.com/YaccConstructor/RedisGraph`
- Cypher parser extended with path patterns:
  `https://github.com/YaccConstructor/libcypher-parser`

Contact Information

- Try it out (Docker image with extended RedisGraph):
  `https://hub.docker.com/r/simpletondl/redisgraph`
- RedisGraph extended with CFPQ:
  `https://github.com/YaccConstructor/RedisGraph`
- Cypher parser extended with path patterns:
  `https://github.com/YaccConstructor/libcypher-parser`

- Semyon Grigorev: s.v.grigoriev@spbu.ru
- Arseniy Terekhov: simpletondl@yandex.ru
- Vlada Pogozhelskaya: pogozhelskaya@gmail.com
- Vadim Abzalov: vadim.i.abzalov@gmail.com
- Timur Zinnatulin: teemychteemych@gmail.com

Contact Information

- Try it out (Docker image with extended RedisGraph):
  `https://hub.docker.com/r/simpletondl/redisgraph`
- RedisGraph extended with CFPQ:
  `https://github.com/YaccConstructor/RedisGraph`
- Cypher parser extended with path patterns:
  `https://github.com/YaccConstructor/libcypher-parser`

Thanks!

- Semyon Grigorev: s.v.grigoriev@spbu.ru
- Arseniy Terekhov: simpletondl@yandex.ru
- Vlada Pogozhelskaya: pogozhelskaya@gmail.com
- Vadim Abzalov: vadim.i.abzalov@gmail.com
- Timur Zinnatulin: teemychteemych@gmail.com