Задача о максимальном потоке в графе

команда группы АС-22-05

Арчаков Ильяс
Епишов Сергей
Дмитриева Екатерина
Котков Роман
Сахаров Данила
Смирнова Анна
Шпагин Алексей

Москва, 2023

Введение

постановка задачи и ее актуальность

Постановка задачи

Дана транспортная сеть $N = (V, E)$ с источником $s \in V$ и пропускными способностями $c(u, v) \geq 0$.

Величиной **потока** называется сумма потоков из источника $|f| = \sum_{u \in V} f(w, t)$.

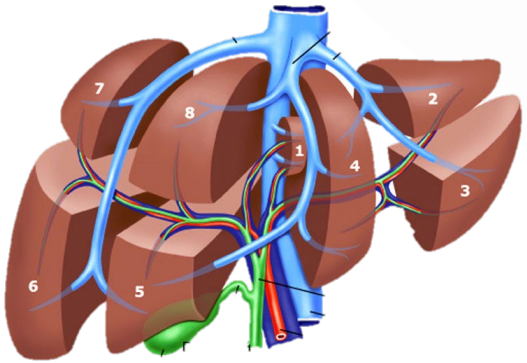
Задача о максимальном потоке заключается в нахождении такого потока, где величина потока максимальна.

Мы проанализируем различные алгоритмы и сравним их время работы, на основе чего сделаем выводы.

Актуальность

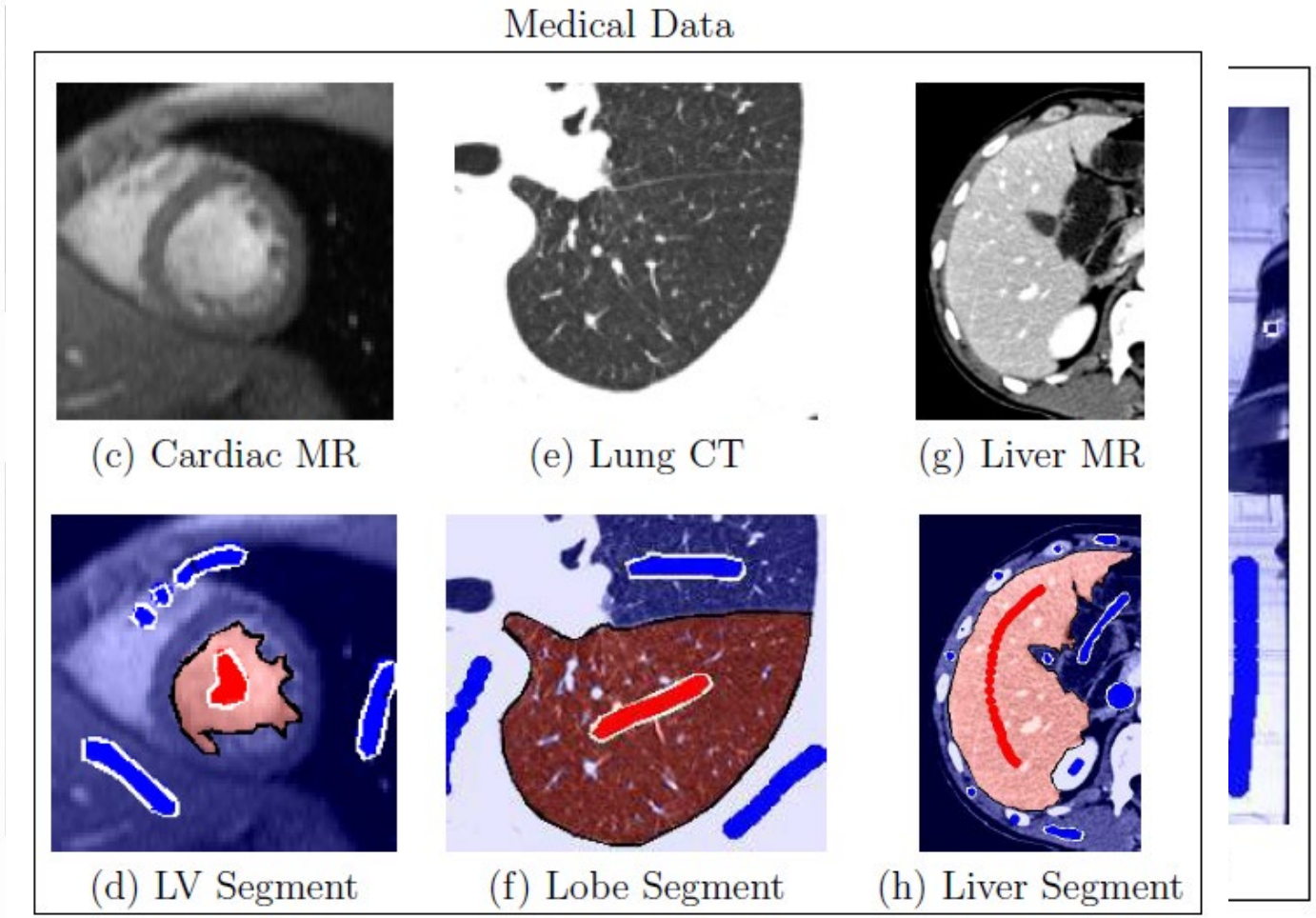
Максимальный поток — это широко применимая модель решения проблем.

- **Транспортная логистика**
- **Телекоммуникационные сети**
- **Электроэнергетика**
- **Банковские операции**
- **Расписание рейсов**
- **Сегментация изображения**
- **Сегментация васкуляризации печени**



Актуальность

- Сегментация изображения



Алгоритмы

основные алгоритмы, решающие данную задачу

Алгоритмы

Алгоритм	Сложность
Алгоритм Форда-Фалкерсона (FF)	$O(EU)$
Алгоритм Эдмонса-Карпа (EC)	$O(VE^2)$
Алгоритм Диница (D)	$O(V^2E)$
Алгоритм проталкивания предпотока (PR)	$O(V^2E)$
Алгоритм масштабирования потока (FS)	$O(E^2 \log U)$
Алгоритм Голдберга-Тарьяна (GT)	$O(V^3)$
Алгоритм «поднять в начало» (RTF)	$O(V^3)$

Алгоритм Форда-Фалкерсона

Идея алгоритма заключается в следующем. Изначально величине потока присваивается значение 0: $f(u, v) = 0$ для всех $u, v \in V$.

Затем величина потока **итеративно увеличивается** посредством поиска увеличивающего пути (путь от источника s к стоку t, вдоль которого можно послать больший поток). Процесс повторяется, **пока можно найти увеличивающий путь**.

Алгоритм Форда-Фалкерсона

Особенности:

- Алгоритм не конкретизирует, какой именно путь мы ищем или как мы это делаем. В реализации использовался **обход в глубину (DFS)**.
- Гарантированно сходится только для целых пропускных способностей, но даже для них при больших значениях пропускных способностей **он может работать очень долго**.
- Если пропускные способности **вещественны**, алгоритм может работать **бесконечно долго**, не сходясь к оптимальному решению.
- Сложность: $O(EU)$, где U — величина максимального потока.

Алгоритм Эдмондса-Карпа

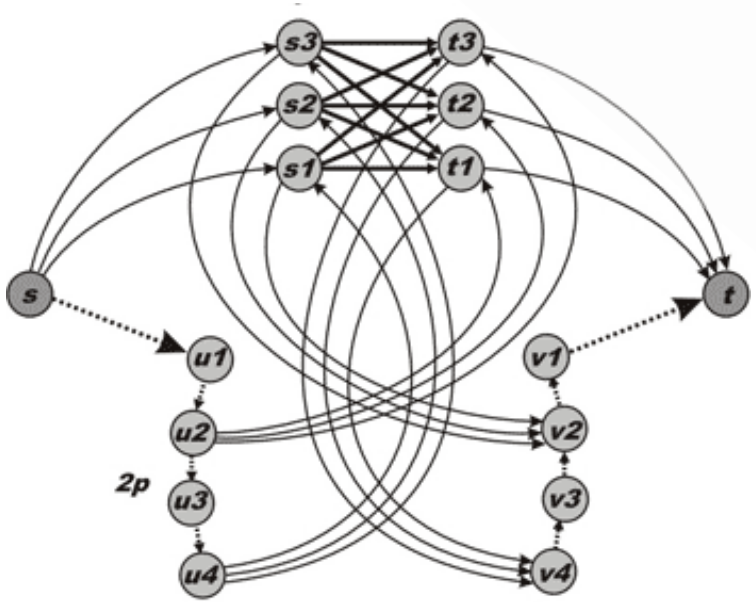
Алгоритм представляет собой **частный случай** метода Форда-Фалкерсона. Впервые был опубликован в 1970 году советским учёным Е. А. Диницом. Позже, в 1972 году, был независимо открыт Эдмондсом и Карпом.

Алгоритм Эдмондса-Карпа — это вариант алгоритма Форда-Фалкерсона, при котором на каждом шаге выбирают **кратчайший дополняющий путь** из s в t в остаточной сети (полагая, что **каждое ребро имеет единичную длину**).

Алгоритм Эдмондса-Карпа

Особенности:

- На каждом шаге выбирают **кратчайший** дополняющий путь.
- Кратчайший путь находится **поиском в ширину(BFS).**
- Есть граф «Грибок», на котором алгоритм дает плохую асимптотику (нижняя граница времени работы)
- Сложность: $O(VE^2)$.



Алгоритм Диница

Полиномиальный алгоритм, предложенный в 1970 году советским (впоследствии израильским) математиком Ефимом Диницем, **частный случай метода Форда-Фалкерсона.**

Используются понятия **вспомогательной сети, остаточной сети и блокирующего потока.**

Блокирующим потоком *в данной сети называется такой поток, что любой путь из истока* s *в сток* t *содержит насыщенное этим потоком ребро. Иными словами, в данной сети не найдётся такого пути из истока в сток, вдоль которого можно беспрепятственно увеличить поток.*

Алгоритм Диница

Остаточной сетью G^R *по отношению к сети* G *и некоторому потоку* f *в ней называется сеть, в которой каждому ребру* $(u, v) \in G$ *с пропускной способностью* c_{uv} *и потоком* f_{uv} *соответствуют два ребра:*

$$(\boldsymbol{u, v})\colon\ \boldsymbol{c^R_{uv} = c_{uv} - f_{uv}}$$

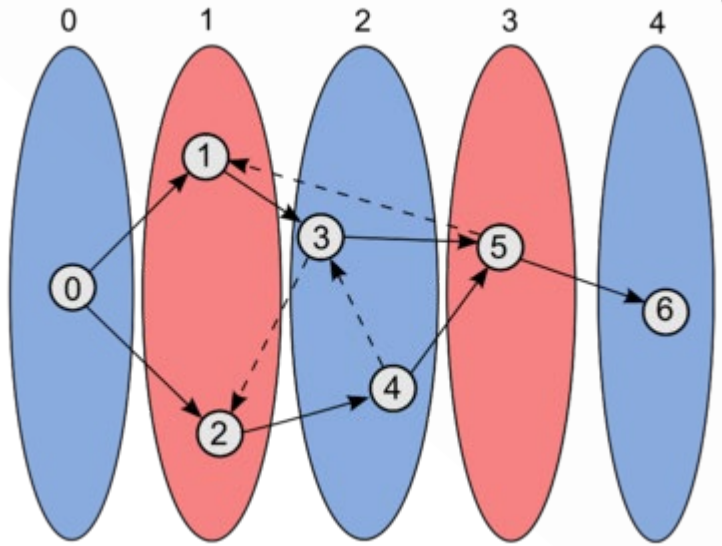
$$(\boldsymbol{v, u})\colon\ \boldsymbol{c^R_{vu} = f_{uv}}$$

Алгоритм Диница

Для начала определим для каждой вершины v данной сети G длину кратчайшего пути $s \to t$ и обозначим её $d[v]$ (для этого можно воспользоваться **обходом в ширину**).

В слоистую сеть включаем только те рёбра (u, v) исходной сети, для которых $d[u] + 1 = d[v]$. Полученная сеть **ациклична**, и любой путь $s \to t$ в слоистой сети является **кратчайшим путём** в исходной, из свойств обхода в ширину.

*Слоистую сеть для графа G будем называть **вспомогательной сетью**.*



Алгоритм Диница

Особенности:

- Представляет собой **несколько фаз**: построение остаточной сети (обход в ширину), поиск блокирующего потока в слоистой сети (обход в глубину) и прибавление его к текущему потоку.
- Схож с алгоритмом Эдмондса-Карпа, но на каждой итерации поток увеличивается не вдоль одного кратчайшего $s \rightarrow t$, а **вдоль целого набора таких путей**.
- Используются алгоритмы поиска в глубину**(DFS)** и поиска в ширину**(BFS)**.
- Сложность: $O(V^2E)$.

Алгоритм проталкивания предпотока

Обобщенный алгоритм нахождения максимального потока в сети. В отличие от алгоритма Эдмондса-Карпа и алгоритма Диница **не является частным случаем метода Форда-Фалкерсона**.

Используются понятия **предпотока, избыточного потока и высоты вершины.**

Функция h_u называется **высотой вершины**, если она удовлетворяет условиям:

$$h(s) = |V|,$$

$$h(t) = 0,$$

$$\forall (u, v) \in E_f \; h(u) \leq h(v) + 1.$$

Алгоритм проталкивания предпотока

Предпотоком (preflow) будем называть функцию $f : V \times V \to \boldsymbol{R}$, удовлетворяющую следующим свойствам:

$$f(u,v) = -f(v,u) \textit{ (антисимметричность),}$$

$$f(u,v) \leq c(v,u) \textit{ (ограничение пропускной способностью),}$$

$\forall u \in V \setminus \{s,t\}$ $\sum_{u \in V} f(u,v) \geq 0$*(Неотрицательность избыточного потока).*

Содержащаяся в этом свойстве сумма $\sum_{u \in V} f(u,v) \geq 0$ называется **избыточным потоком (excess)** и обозначается e_u.

Мы называем вершину **переполненной**, если она не является источником или стоком, а избыточный поток в эту вершину **строго положителен**.

Алгоритм проталкивания предпотока

Алгоритм применяет две операции: **проталкивание(push)** и **подъём(relabel).**

Операция **проталкивания(push)** из вершины u в вершину v может применяться тогда, когда вершина u является **переполненной**:

```
function push(Node u, Node v)
    d = min(e(u), c(u, v) − f(u, v))
    f(u, v) += d
    f(v, u) = −f(u, v)
    e(u) −= d
    e(v) += d
```

Алгоритм проталкивания предпотока

Операция **подъёма(relabel)** применима для вершины u, если $e(u) > 0$ и $\forall (u, v) \in E_f \quad h(u) \leq h(v)$:

```
function relabel(Node u)
        h(u) = min{h(v): f(u, v) – c(u, v) < 0} + 1
```

Алгоритм:

- Инициализировать предпоток, избыточные потоки и высоты.
- Пока возможно проталкивание или подъём, выполнить любую возможную операцию.

Алгоритм проталкивания предпотока

Особенности:

- Считается одним из **наиболее эффективных** алгоритмов максимального потока.
- Сложность: $O(V^2E)$.
- Конкретные варианты алгоритмов обеспечивают **еще более низкую временную сложность:** $O(V^3)$, $O\left(V^2\sqrt{E}\right)$ и $O\left(VE\log\frac{V^2}{E}\right)$.

Алгоритм масштабирования потока

Является **модификацией** алгоритма Форда-Фалкерсона для нахождения максимального потока в сети.

Идея алгоритма заключается **в нахождении путей с высокой пропускной способностью в первую очередь**, чтобы сразу сильно увеличивать поток по ним, а затем по всем остальным.

Для этого используют **масштабом** Δ, изначально предполагая, что

$$\Delta = 2^{\lfloor \log_2 U \rfloor}.$$

На каждой итерации в дополняющей сети алгоритм находит **дополняющие пути** с пропускной способностью **не меньшей** Δ, и увеличивает поток вдоль них. **Уменьшив** масштаб Δ **в 2 раза**, переходит к следующей итерации.

Алгоритм масштабирования потока

Для этого используют **масштабом** Δ, изначально предполагая, что

$$\Delta = 2^{\lfloor \log_2 U \rfloor}.$$

На каждой итерации в дополняющей сети алгоритм находит **дополняющие пути** с пропускной способностью **не меньшей** Δ, и увеличивает поток вдоль них. **Уменьшив** масштаб Δ **в 2 раза**, переходит к следующей итерации.

Очевидно, что **при** $\Delta = \mathbf{1}$, алгоритм вырождается в алгоритм **Эдмондса-Карпа**, вследствие чего является корректным.

Количество необходимых увеличений путей, основанных на кратчайших путях, может быть много больше количества увеличений, основанных на путях с высокой пропускной способностью.

Алгоритм масштабирования потока

Особенности:

- Введение **масштаба** Δ позволяет **уменьшить** количество фаз алгоритма, **ускоряя сходимость**.
- Алгоритм **эффективен** для сетей **с большими пропускными способностями**, так как он уменьшает количество фаз.
- Может **не эффективно** работать на графах **с малыми пропускными способностями**, так как масштаб Δ может стать **слишком большим**.
- **Не работает** с графами, где пропускные способности имеют **нецелочисленные** значения (аналогично алгоритму Форда-Фалкерсона).
- Сложность:$O(E^2 \log U)$.

Алгоритм Голдберга-Тарьяна

Модификация алгоритма **проталкивания предпотока**, использующая правило выбора **FIFO** (First In First Out).

Изначально все вершины, кроме источника и стока, имеют высоту 0, а все ребра имеют нулевой поток.

Алгоритм ищет **наилучший путь** из источника s в сток t, используя операции **push** (проталкивание) и **relabel** (переустановка высоты).

Алгоритм продолжает выполнять операции **push** и **relabel** до тех пор, пока не будет достигнута максимальная пропускная способность (максимальный поток) между источником s и стоком t.

Алгоритм Голдберга-Тарьяна

Особенности:

- Алгоритм организует активные узлы в **очередь**. Вершины, которые протолкнули поток, записываются в очередь для дальнейшего рассмотрения **по очереди**. Используется правило **FIFO**.
- Используется **перемаркировка** вершин, **переполненная** вершина ($e(u) > 0$) здесь называется **активной**.
- Сложность: $O(V^3)$.

Алгоритм «поднять в начало»

Алгоритм «поднять в начало» **(Relabel To Front)** представляет собой более эффективную, чем описано выше, реализацию алгоритма **проталкивания предпотока**.

Алгоритм использует понятие **допустимых рёбер,** применяется дополнительная операция: **разрядка (discharge) вершины.**

Кроме предпотока, высот и избыточного потока, **алгоритм хранит** следующее:

- Для каждой вершины u – список её соседей $N[u]$;
- Для каждой вершины u – указатель $current[v]$ *(в терминах C++ - итератор)*;
- Список L всех вершин, кроме источника и стока;
- Указатель it на один из элементов списка L *(в терминах C++ - итератор).*

Алгоритм «поднять в начало»

Ребро (u, v) называется **допустимым (admissable)**, если выполнены два условия:

1. $f(u, v) < c(u, v)$, или, что то же самое, **ребро** (u, v) присутствует **в остаточной сети**.
2. $h_u = h_v + 1$.

Таким образом:

- *Проталкивание осуществляется только по допустимым рёбрам.*
- *Подъём допустим тогда и только тогда, когда поднимаемая вершина переполнена и из неё не исходит допустимых рёбер.*

Алгоритм «поднять в начало»

Опишем функцию, которую называется **разрядка (discharge)** вершины. Разрядка применяется **только к переполненным вершинам** и выполняется следующим образом:

Шаг 1. Пока вершина u **переполнена**, выполнять шаги 2-4.

Шаг 2. Если ***current*** вышел за конец списка, **поднять** вершину u и вернуть ***current*** в начало списка.

Шаг 3. Иначе, если допустимо проталкивание от u к ***current*****[u],** выполнить его.

Шаг 4. Иначе продвинуть ***current*** на 1 элемент вперёд.

Алгоритм «поднять в начало»

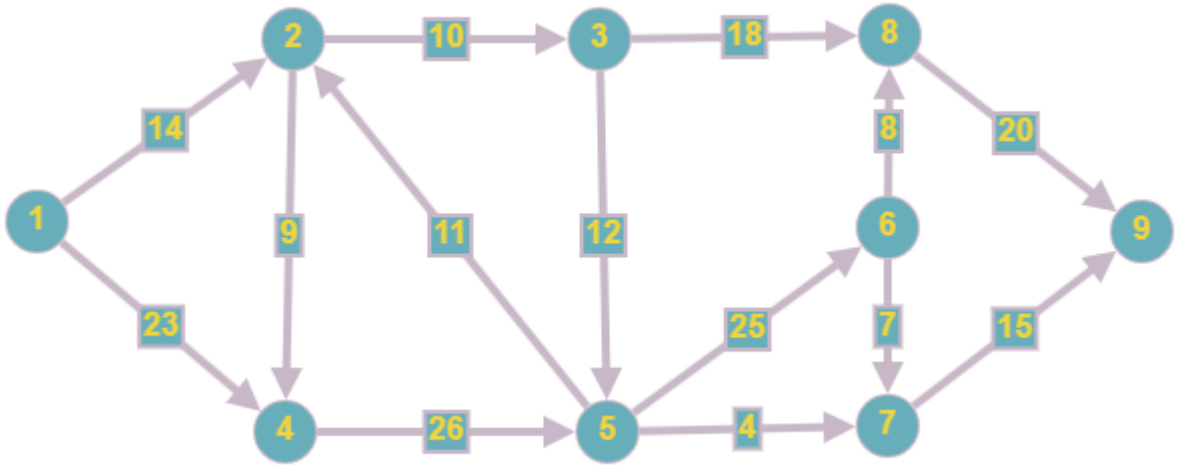
Особенности:

- Алгоритм может работать **плохо** на некоторых графах, особенно **с высокой плотностью** или с большой разницей между максимальной и минимальной емкостью.
- Алгоритм **не подходит для динамических графов**, структура которых часто меняется, поскольку для поддержания структур данных **требуются значительные вычислительные затраты**.
- Сложность: $O(V^3)$, самый быстрый из представленных.

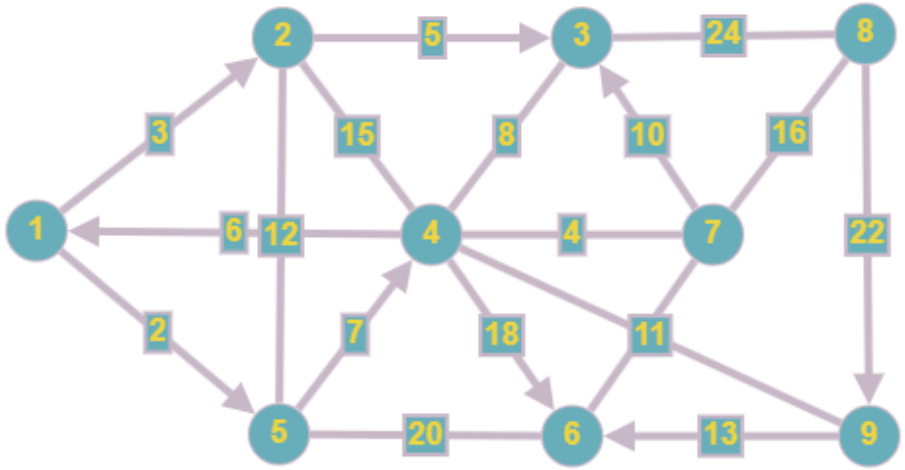
Результаты

сравнение времени работы и обработка измерений

Результаты



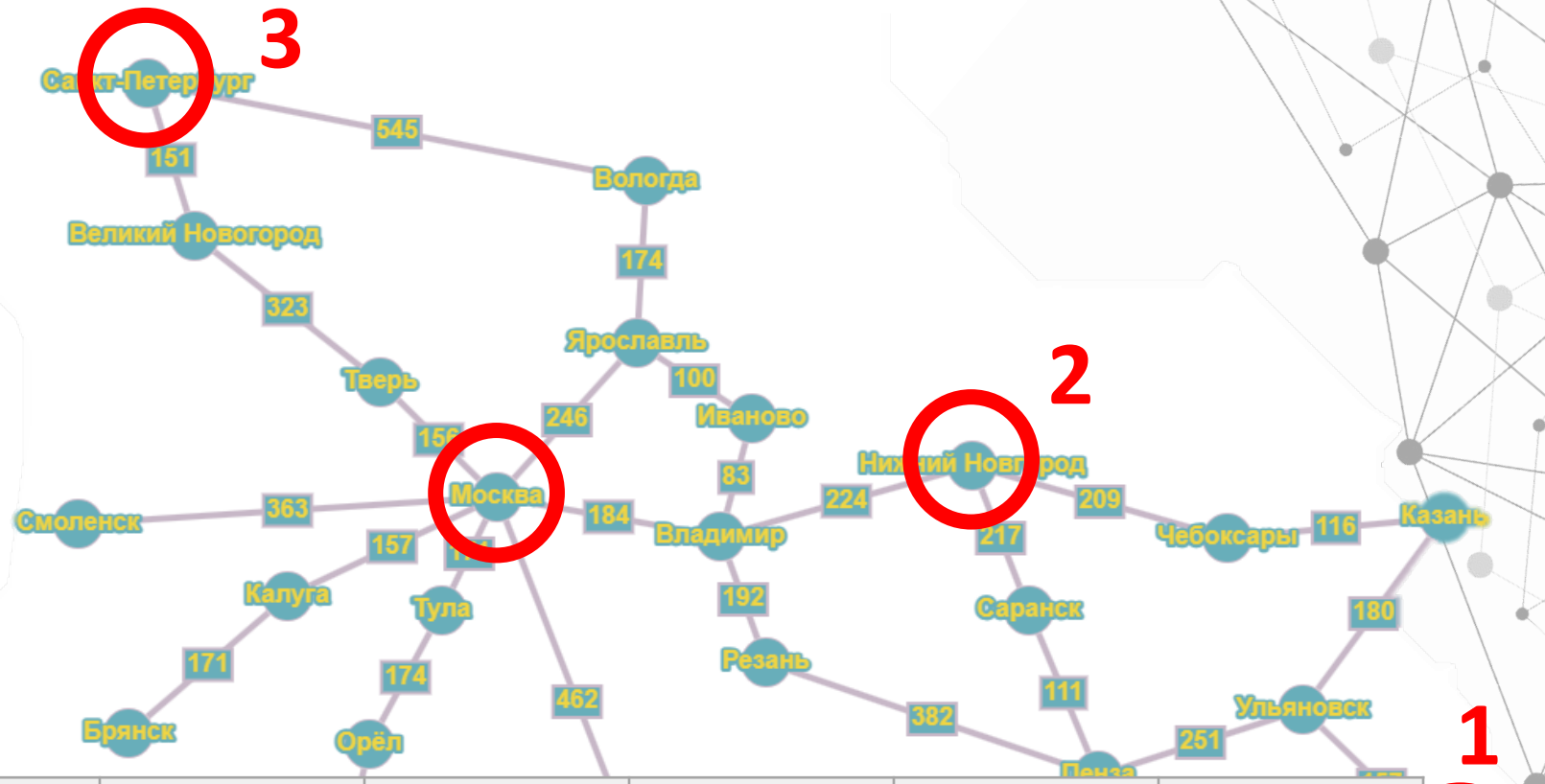
Max flow = 29



Max flow = 26

№	FF	EC	D	PR	FS	GT	RTF
	Время, 10^{-6} с (среднее за 5.000.000)						
1	2,20003	3,7043	2,09825	2,09964	2,10066	2,04848	0,034555
2	4,27919	8,04156	4,21923	4,20167	4,2206	2,688	0,069376

Результаты

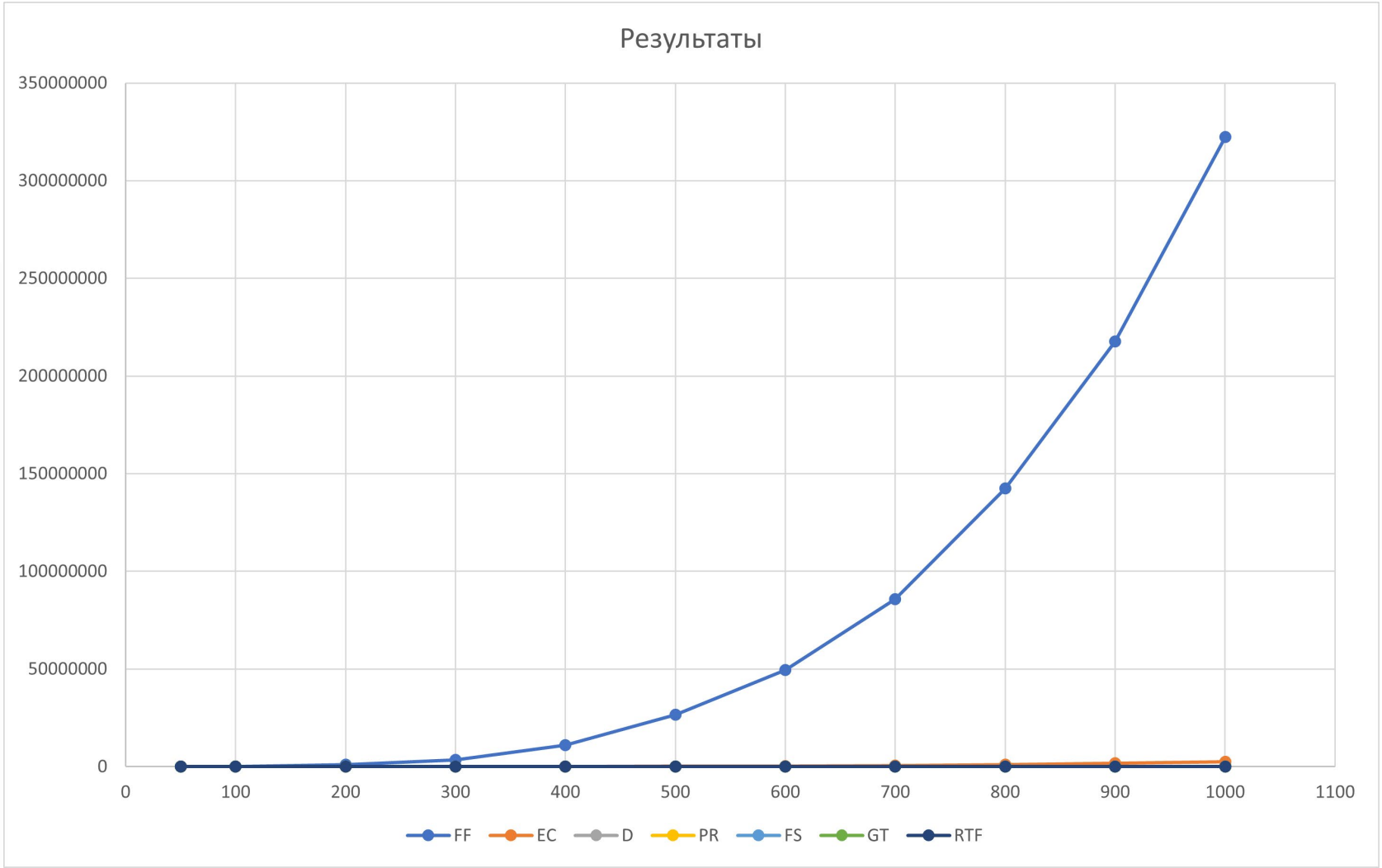


№	FF	EC	D	PR	FS	GT	RTF
	Время, 10^{-6} с (среднее за 1.000.000)						
1	5,53781	11,7614	10,0329	10,0437	9,89717	1,68296	0,034213
2	8,51387	17,9454	14,5506	14,5287	14,3446	3,69998	0,069309
3	12,6396	29,1201	26,8957	26,6517	26,6073	6,01439	0,103685

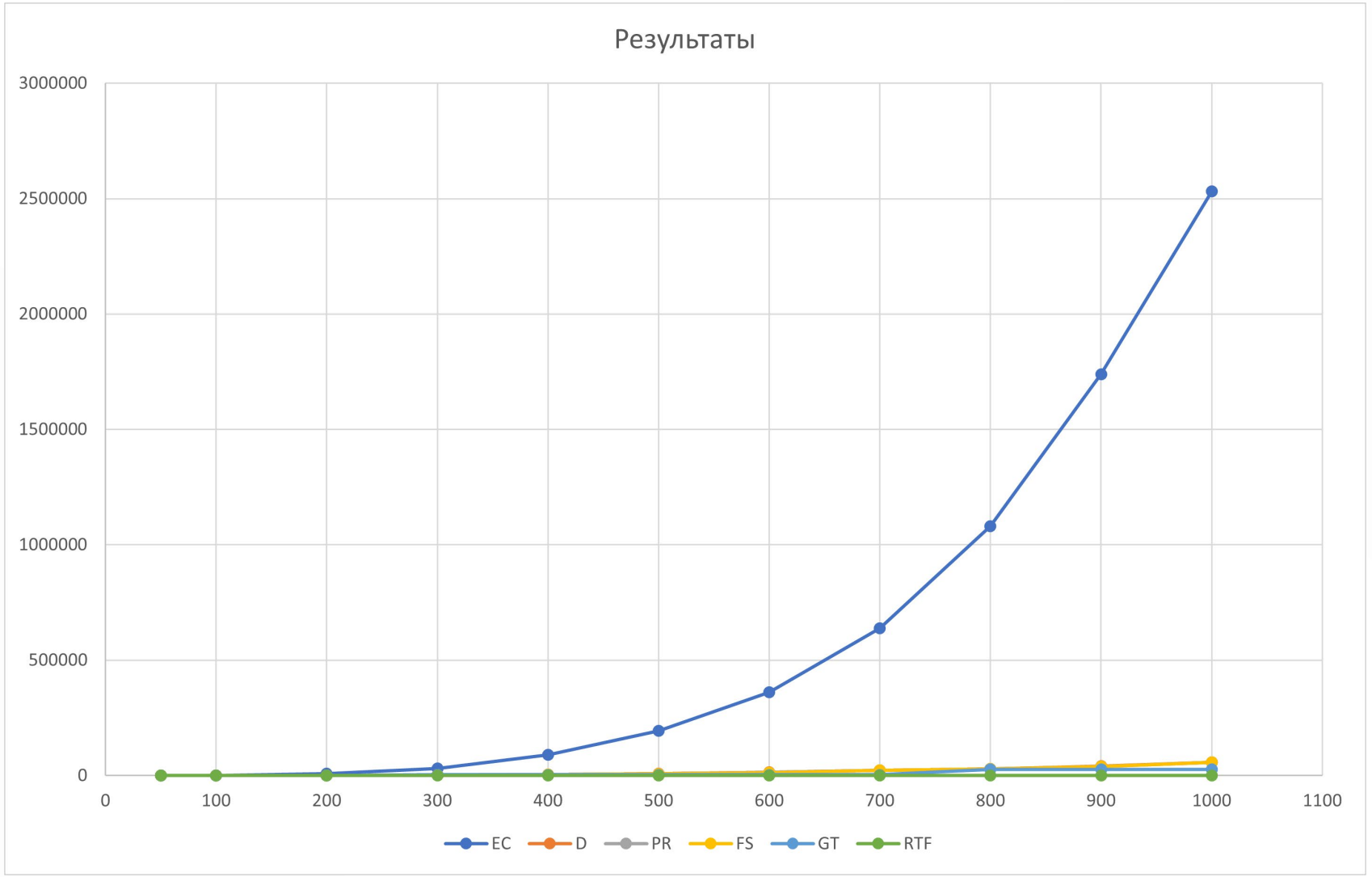
Результаты

V	FF	EC	D	PR	FS	GT	RTF
	Время, 10^{-6} с (среднее за 500), плотность = 0,33						
50	2881,16	162,499	23,0714	28,7416	25,2182	117,517	0,0326
100	50231,4	1089,09	127,313	147,682	133,826	119,029	0,0656
200	945961	9113	1035,38	1058,09	1042,69	121,26	0,1022
300	3,478E+06	31087,2	2137,72	2156,84	2147,22	3265,14	0,1354
400	1,098E+07	90325,3	4134,82	4126,22	4040,2	3268,74	0,1678
500	2,67E+07	194165	7609,27	7379,36	7338,18	3275,48	0,2006
600	4,96E+07	360883	13756,5	13470	13339,6	3280,64	0,2338
700	8,57E+07	638307	22194,1	21841,2	21997,1	3286,55	0,2666
800	1,43E+08	1081040	29153,6	28737,9	28854,5	26220,4	0,2996
900	2,18E+08	1738430	40153,8	39239,5	39424,9	26227,8	0,3324
1000	3,22E+08	2531250	58030,1	56985	57194,2	26236,1	0,3654

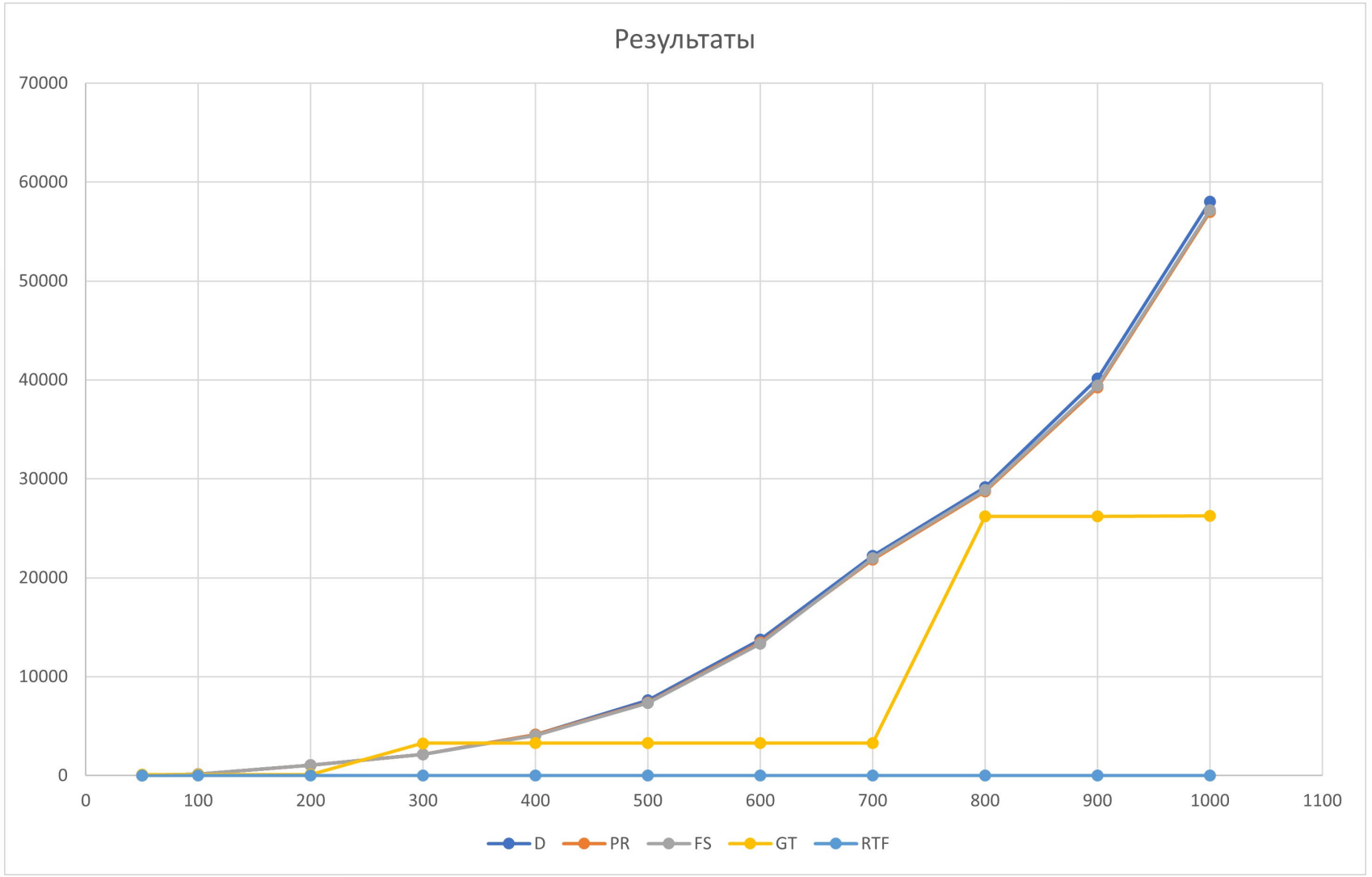
Результаты


Результаты
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
0
100
200
300
400
500
600
700
800
900
1000
1100
FF
EC
D
PR
FS
GT
RTF

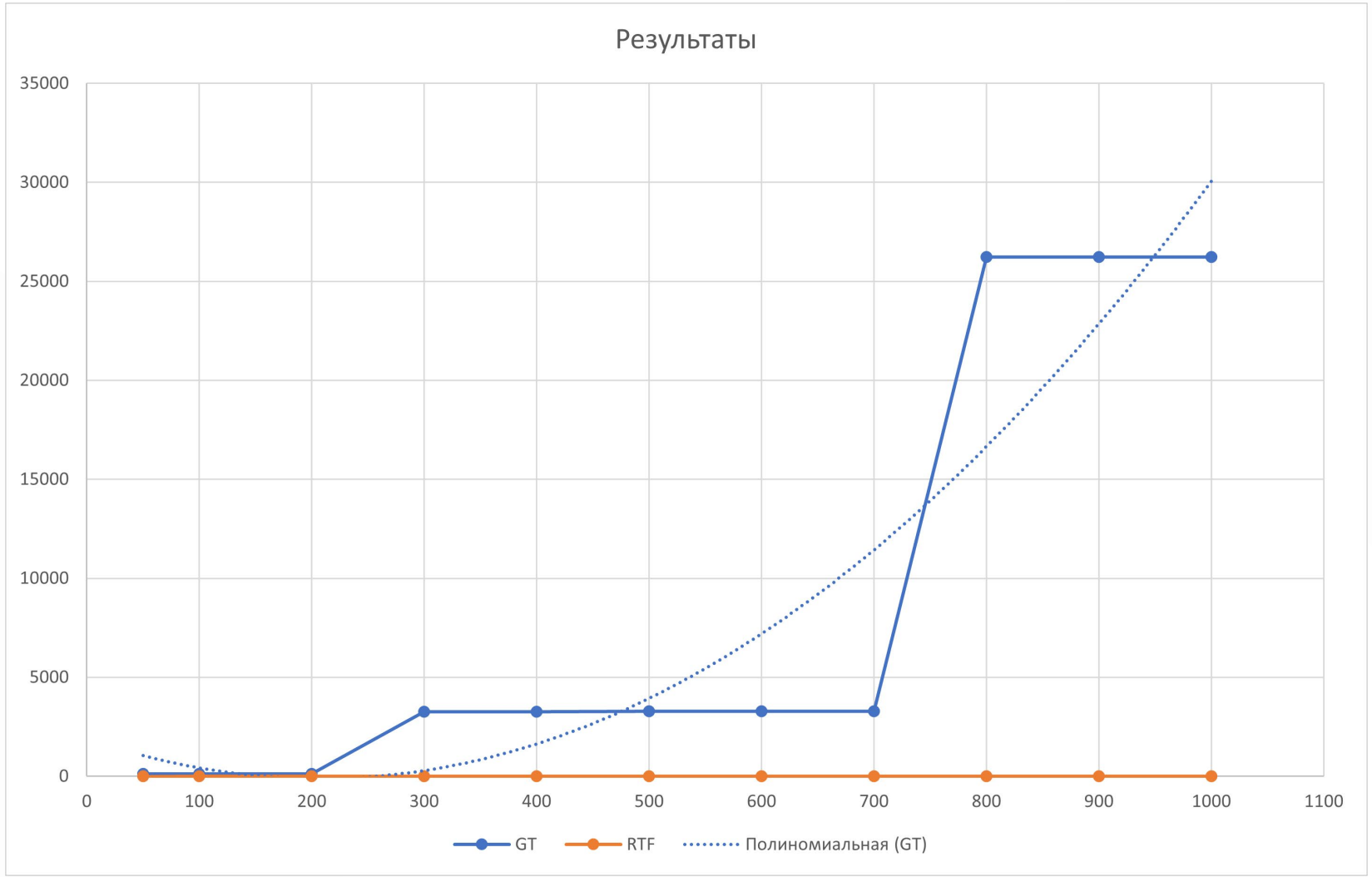
Результаты


Результаты
3000000
2500000
2000000
1500000
1000000
500000
0
0
100
200
300
400
500
600
700
800
900
1000
1100
EC
D
PR
FS
GT
RTF

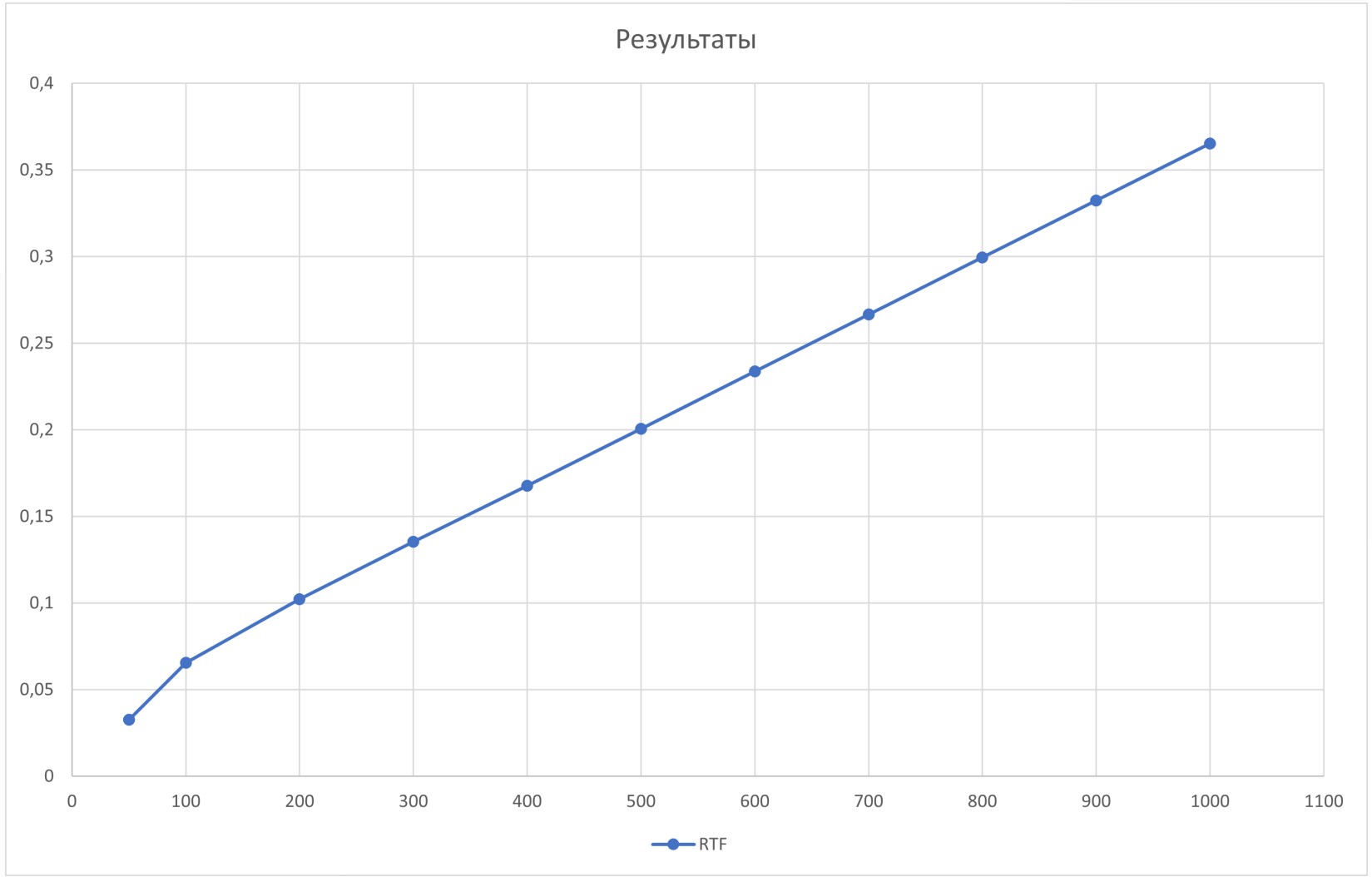
Результаты


Результаты
70000
60000
50000
40000
30000
20000
10000
0
0
100
200
300
400
500
600
700
800
900
1000
1100
D
PR
FS
GT
RTF

Результаты


Результаты
35000
30000
25000
20000
15000
10000
5000
0
0
100
200
300
400
500
600
700
800
900
1000
1100
GT
RTF
Полиномиальная (GT)

Результаты


Результаты
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
100
200
300
400
500
600
700
800
900
1000
1100
RTF

Результаты

Алгоритм	Сложность
Алгоритм Форда-Фалкерсона (FF)	$O(EU)$
Алгоритм Эдмонса-Карпа (EC)	$O(VE^2)$
Алгоритм Диница (D)	$O(V^2E)$
Алгоритм проталкивания предпотока (PR)	$O(V^2E)$
Алгоритм масштабирования потока (FS)	$O(E^2 \log U)$
Алгоритм Голдберга-Тарьяна (GT)	$O(V^3)$
Алгоритм «поднять в начало» (RTF)	$O(V^3)$

Выводы

- Алгоритм Форда-Фалкерсона оказался **самым медленным**, что и ожидалось увидеть.
- Алгоритм «поднять в начало» оказался **самым быстрым** из представленных.
- Три алгоритма с абсолютно разным подходом и разной реализацией показали очень **схожие результаты**, что подтверждает их схожую временную сложность.
- Задача о максимальном потоке **оказалась намного обширнее**, чем изначально предполагалось, и имеет большое количество применений.

Репозиторий GitHub