Институт прикладной математики им. М.В.Келдыша РАН
Москва

An algorithm for mapping of global adjacency lists to local numeration in a distributed graph in the GridSpiderPar tool

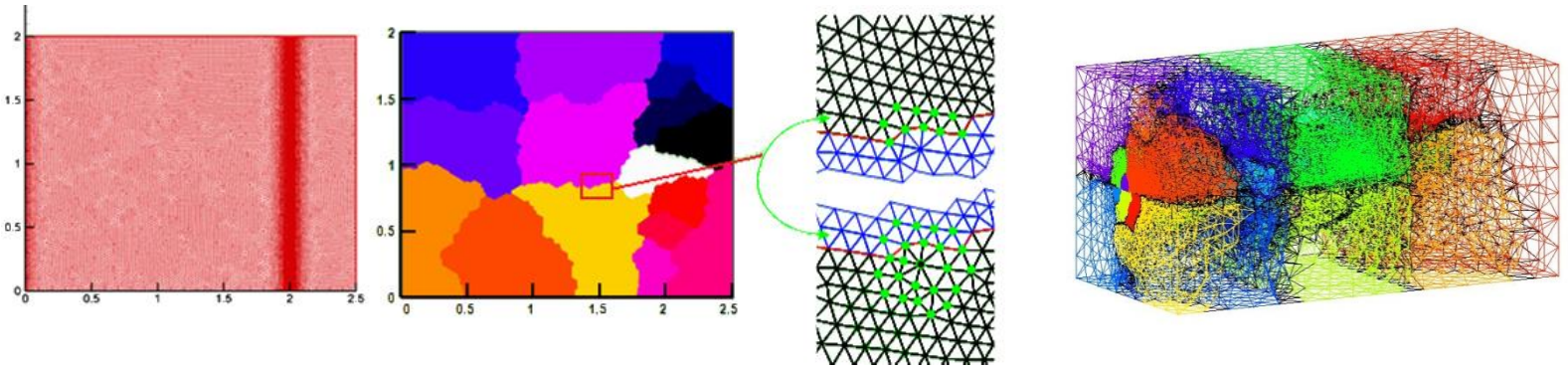
Е.Н. Головченко

Актуальность декомпозиции

- численное решение сеточных задач механики сплошных сред, электродинамики и других на распределенных вычислительных системах

↓

Геометрический параллелизм



Эффективность загрузки процессоров

↙ равномерность распределения сетки по процессорам

↘ минимизация объема передач данных между процессорами

Последовательные пакеты декомпозиции сеток

METIS, Jostle, Scotch, Chaco, Party

Параллельные пакеты декомпозиции сеток

ParMETIS, Jostle, PT-Scotch, Zoltan

Область исследования

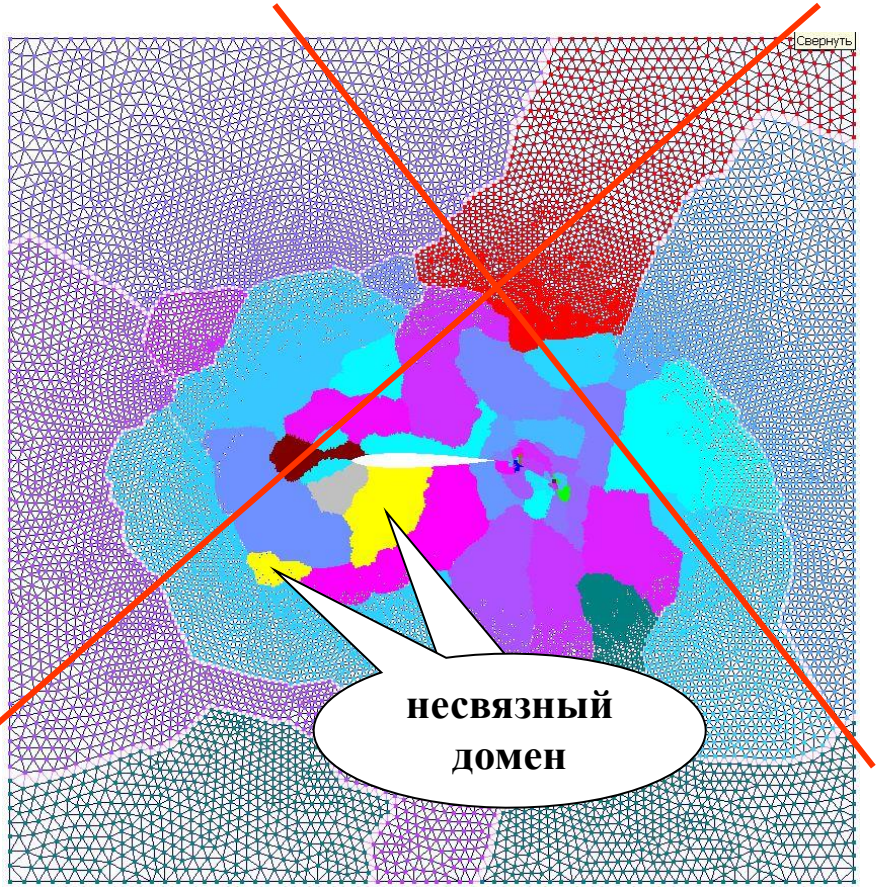
- **нерегулярные сетки, содержащие 10^9 и более вершин**

Недостатки алгоритмов декомпозиции графов, реализованных в существующих пакетах

- образование несвязных доменов
- формирование сильно несбалансированных разбиений (ParMETIS: количества вершин в доменах могут отличаться в 2 раза)
- не всегда удается получить разбиения на большое число микродоменов

Связность важна:

- **алгоритмы решения систем линейных уравнений**
- **компрессия сеточных данных**
- **алгоритм композиции подобластей[1]**
- **распараллеливание[2] методики ТИМ-2D**



[1] А. И. Илюшин, А. А. Колмаков, И. С. Меньшов. Построение параллельной вычислительной модели путем композиции вычислительных объектов // Математическое моделирование. 2011. Т. 23. № 7. 97-113.

[2] А. А. Воропинов. Декомпозиция данных для распараллеливания методики ТИМ-2D и критерии оценки ее качества // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование:», вып. 4. 2009. №37(170). 40-50.

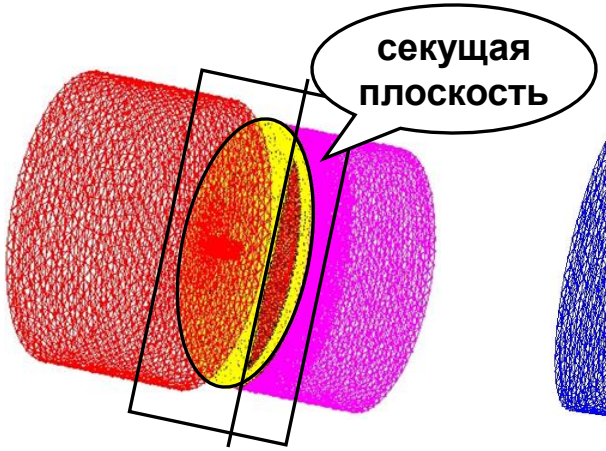
Разработаны Алгоритмы,

- **позволяющие разбивать нерегулярные сетки, содержащие 10^9 и более вершин, на большое количество микродоменов при выполнении следующих критериев:**
  - **сбалансированность получаемых разбиений,**
  - **связность формируемых микродоменов**
  - **минимизация суммарного веса разрезанных ребер**

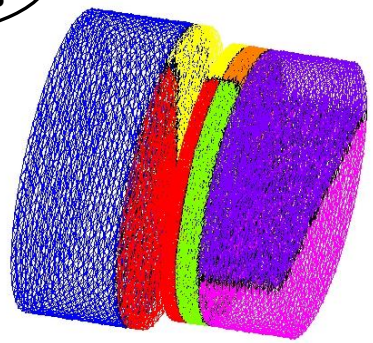
Библиотека параллельной декомпозиции больших сеток GridSpiderPar

(Головченко Е.Н., Якобовский М.В.)

Параллельный алгоритм геометрической декомпозиции сеток



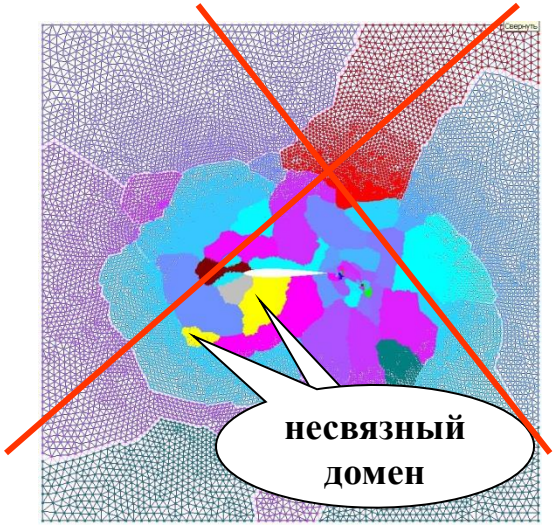
результат



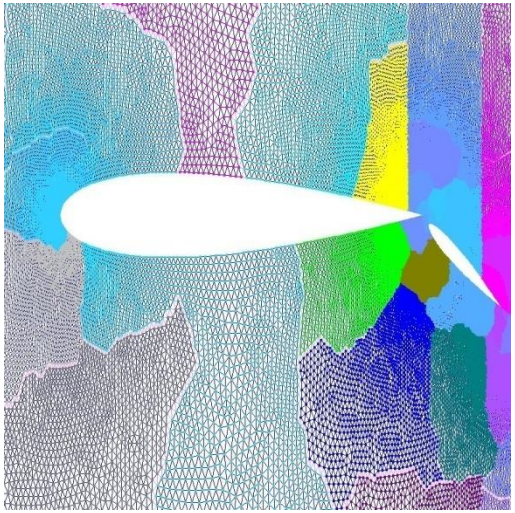
Достоинства

- секущая плоскость (медиана) разрезается по нескольким координатам
- разница числа вершин в доменах не превышает единицу

Параллельный инкрементный алгоритм декомпозиции графов



результат



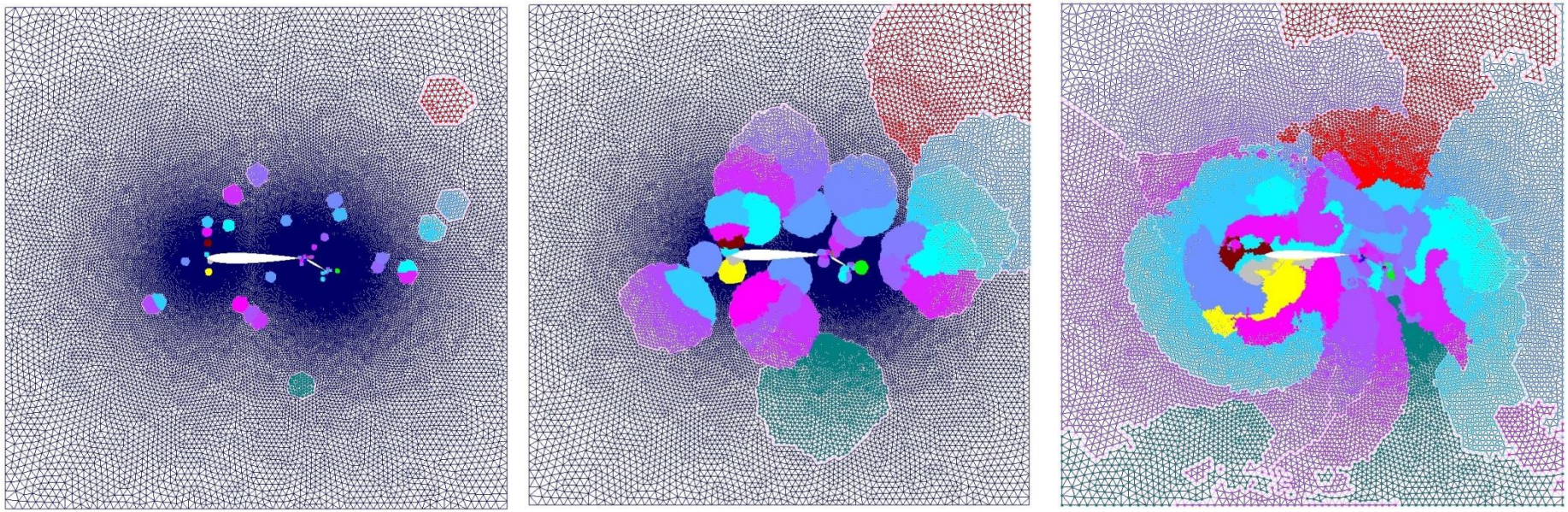
Достоинства

- ориентирован на формирование связных доменов
- сбалансированность разбиений лучше, чем разбиений, получаемых другими методами декомпозиции графов (5% (60%) → 0.05%)

Инкрементный алгоритм декомпозиции графов

(Якобовский М.В., ИПМ им. М.В.Келдыша РАН)

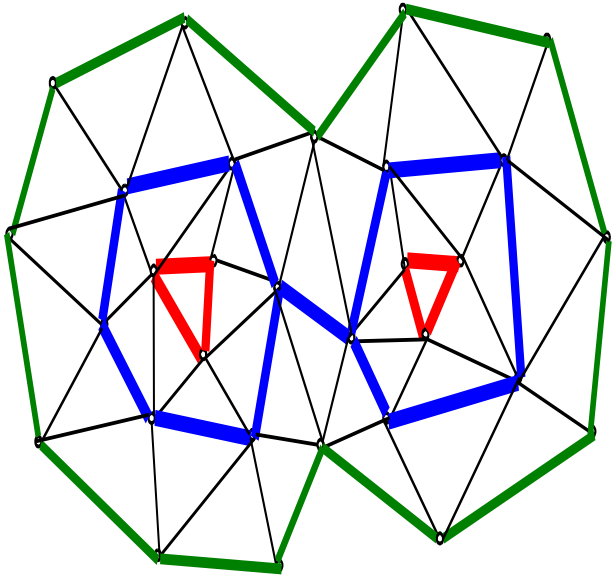
- **инкрементный рост доменов**
- **диффузное перераспределение вершин между доменами**



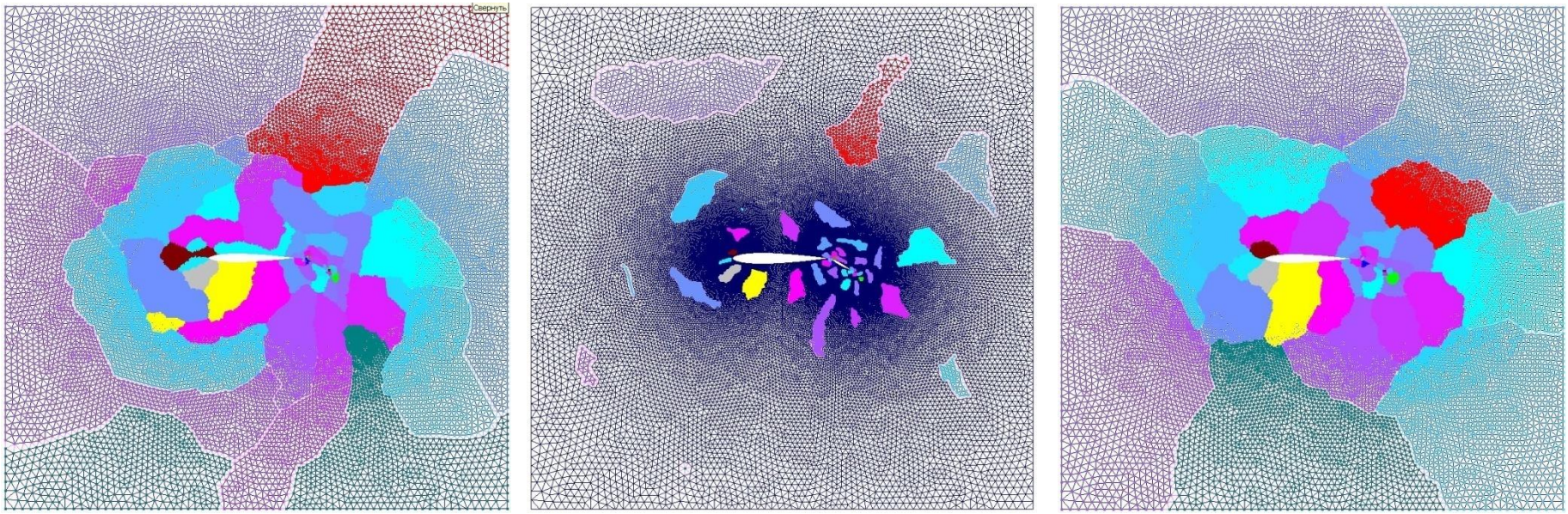
сетка вокруг крыла самолета с закрылком

Инкрементный алгоритм

- **локальное уточнение доменов**
- **проверка качества доменов**
- **освобождение части вершин плохих доменов**



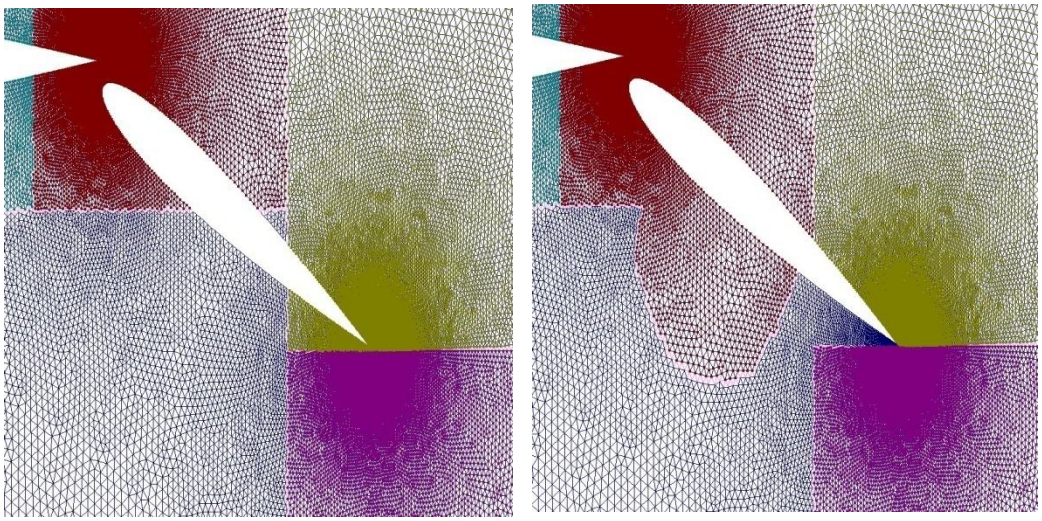
$$T_{k+1} = \mathbf{A}T_k \setminus T_k \setminus T_{k-1}, \quad T_0 = \phi$$



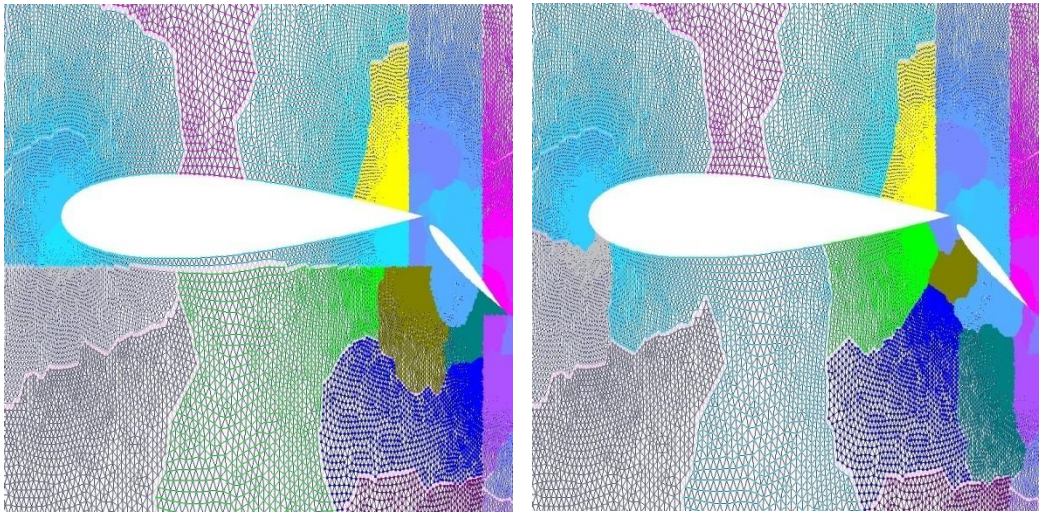
сетка вокруг крыла самолета с закрылком

Параллельный инкрементный алгоритм

- геометрическое разбиение сетки по процессорам
- перераспределение малых блоков вершин



- локальное разбиение
- сбор плохих групп доменов и их повторное разбиение



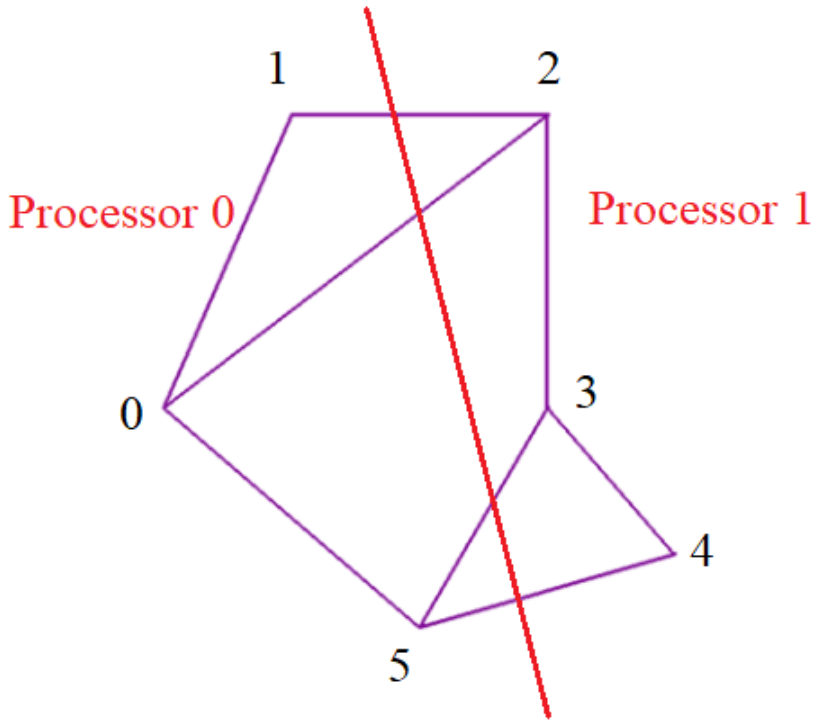
Параллельный инкрементный алгоритм декомпозиции графов:

Достоинства

- ориентирован на формирование связных доменов
- сбалансированность разбиений лучше, чем разбиений, получаемых другими методами декомпозиции графов (5% (60%) → 0.05%)

Проблема преобразования глобальных номеров вершин в локальные

- **возникает после перераспределения малых блоков вершин**
- **и при перераспределении плохих групп доменов**



пример небольшого графа, распределенного между двумя процессорами

Преобразование глобальных номеров вершин в локальные

Processor 0:

0:	1	2	5
1:	0	2	
5:	0	3	4

Processor 1:

2:	0	1	3
3:	2	4	5
4:	3	5	

Processor 0:

0(0):	1	3	2
1(1):	0	3	
2(5):	0	4	5
3(2):	0	1	
4(3):	2		
5(4):	2		

Processor 1:

0(2):	3	4	1
1(3):	0	2	5
2(4):	1	5	
3(0):	0		
4(1):	0		
5(5):	1	2	

Проблема преобразования глобальных номеров вершин в локальные

- возникает после предекомпозиции графа по процессорам
- или когда часть вершин передается между процессорами в процессе счета

- для эффективного параллельного счета нужна непрерывная локальная нумерация

- на процессорах могут оказаться любые глобальные номера вершин и в любом порядке

Проблема не описана в других работах

- нет глобальной нумерации: части сетки состыковываются по геометрическим границам
- нет локальной нумерации:
  - Zoltan: матрица смежности распределена между $\sqrt{p}\cdot\sqrt{p}$ процессорами. Выполняются коллективные операции по строкам или по столбцам
  - Jostle: поиск вершин графа по глобальным номерам с использованием хэш таблицы и бинарных деревьев

Проблема не описана в других работах

- нет локальной нумерации:
  - GraphA (adaptive scheme for efficient partitioning): в списках смежности вершины индексируются хэш значениями вершин-владельцев списков с помощью Adaptive Radix Tree (ART)

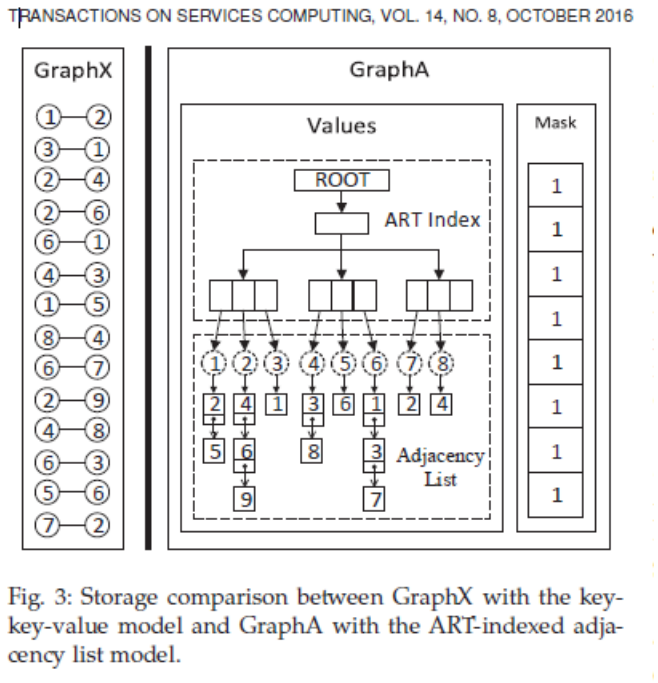
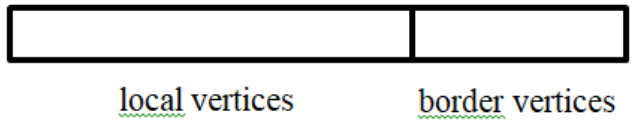


Fig. 3: Storage comparison between GraphX with the key-key-value model and GraphA with the ART-indexed adjacency list model.

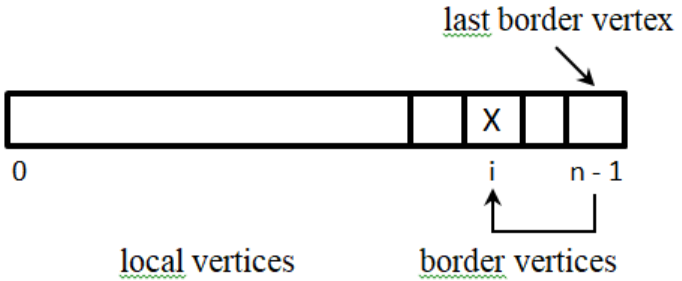
- на каждом процессоре вводится новая локальная нумерация для элементов сетки и строится новый локальный дуальный граф

Хранение графа в параллельном инкрементном алгоритме

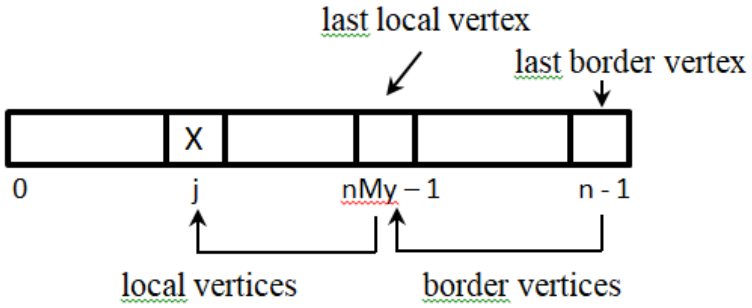
хранение графа



удаление граничной вершины



удаление локальной вершины



Алгоритм преобразования глобальных списков смежности в локальную нумерацию

Mapping Algorithm. The algorithm maps global numbers of adjacent vertices to their indexes in the local graph *g* and adds missing border vertices. *nOld* – initial number of vertices in the graph *g*.

1. [Fill arrays.] Array *A* ← *{vertex global number; vertex index}.*
Array *B* ← *{adjacent vertex global number; vertex index; adjacent vertex index}.*
2. [Sort.] Sort the arrays *A* and *B*.
3. [Initialize.] Set *i* ← 0, *j* ← 0.
4. [Compare]. if *j* == *nA* OR *B[3 · i] < A[2 · j]* go to step 5, otherwise go to step 8. (There is no vertex with the global number *B[3 · i]* in the initial configuration of the graph *g)*.
5. [Compare]. if *g→n* == nOld OR *B[3 · i] > global number of vertex with the index (g→n – 1)* then go to step 6, otherwise go to step 7. (If a border vertex with the global number *B[3 · i]* exists then it is the last vertex in the graph *g*, because border vertices are added to the end of the graph *g* and the array *B* is sorted).

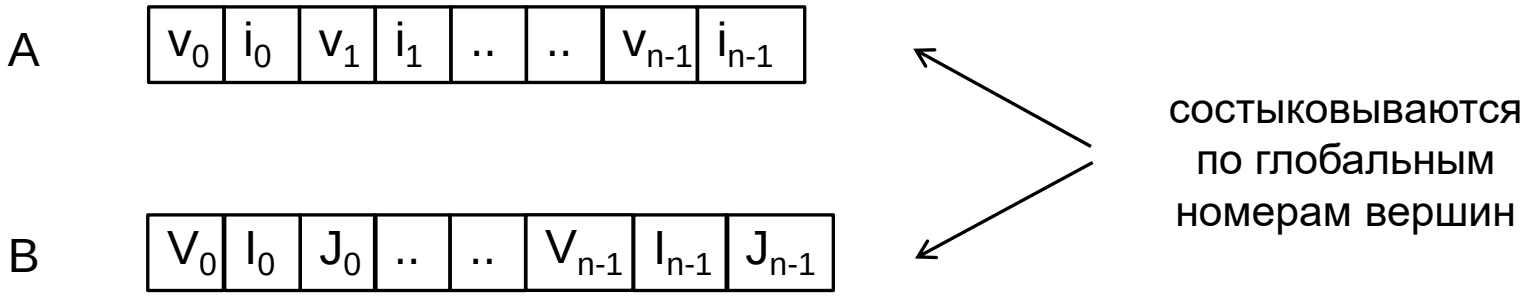
Алгоритм преобразования глобальных списков смежности в локальную нумерацию

6. [Add]. Add a new border vertex with the index $g{\rightarrow}n \rightarrow$ graph g. Set global number of this vertex $\leftarrow B[3 \cdot i]$, $(g{\rightarrow}n) \leftarrow (g{\rightarrow}n) + 1$.
7. [Map]. Set adjacent vertex with the index $B[3 \cdot i + 2]$ of the vertex with the index $B[3 \cdot i + 1] \leftarrow g{\rightarrow}n - 1$. Add $B[3 \cdot i + 1]$ to the adjacent list of the vertex with the index $g{\rightarrow}n - 1$. Set weight of this edge equal to the weight of the edge with the index $B[3 \cdot i + 2]$ of the vertex with the index $B[3 \cdot i + 1]$. Increase i by 1 and go to step 10.
8. [Compare]. If $B[3 \cdot i] > A[2 \cdot j]$ increase j by 1 and go to step 10, otherwise go to step 9.
9. [Map]. In this case $B[3 \cdot i] = A[2 \cdot j]$. Set adjacent vertex with the index $B[3 \cdot i + 2]$ of the vertex with the index $B[3 \cdot i + 1] \leftarrow A[2 \cdot j + 1]$. If the vertex with the index $A[2 \cdot j + 1]$ is boundary then add $B[3 \cdot i + 1]$ to its adjacent list. Set weight of this edge equal. Increase i by 1 and go to step 10.
10. If $i < nB$ go to step 4, else finish the algorithm.

Алгоритм преобразования глобальных списков смежности в локальную нумерацию

- **алгоритм работает даже с неоптимальной нумерацией вершин**
- **может быть применен как к целому графу, так и к его части**

sort A: $v_0 \le v_1 .. \le v_{n-1}$,
sort B: $V_0 \le V_1 .. \le V_{n-1}$



v_k – глобальный номер вершины
i_k – индекс (локальный номер) вершины
V_k – глобальный номер соседа
I_k – индекс вершины
J_k – индекс соседа

Результаты

Преобразование глобальных списков смежности в локальную нумерацию

Mesh	Number of vertices in the graph	time of the algorithm with the binary search, sec.	time of the algorithm where the proposed algorithm is applied, sec.
Mesh1	$2 \cdot 10^8$	97.9	5.4
Mesh2	$2.6 \cdot 10^8$	128	6.7
Mesh3	$1.1 \cdot 10^8$	25.1	2.7

вычисления проводились на суперкомпьютере К-100 (ИПМ им. М.В.Келдыша РАН)

512 MPI processes (6 processes per processor, 12 processes per node, 43 compute nodes).

Результаты

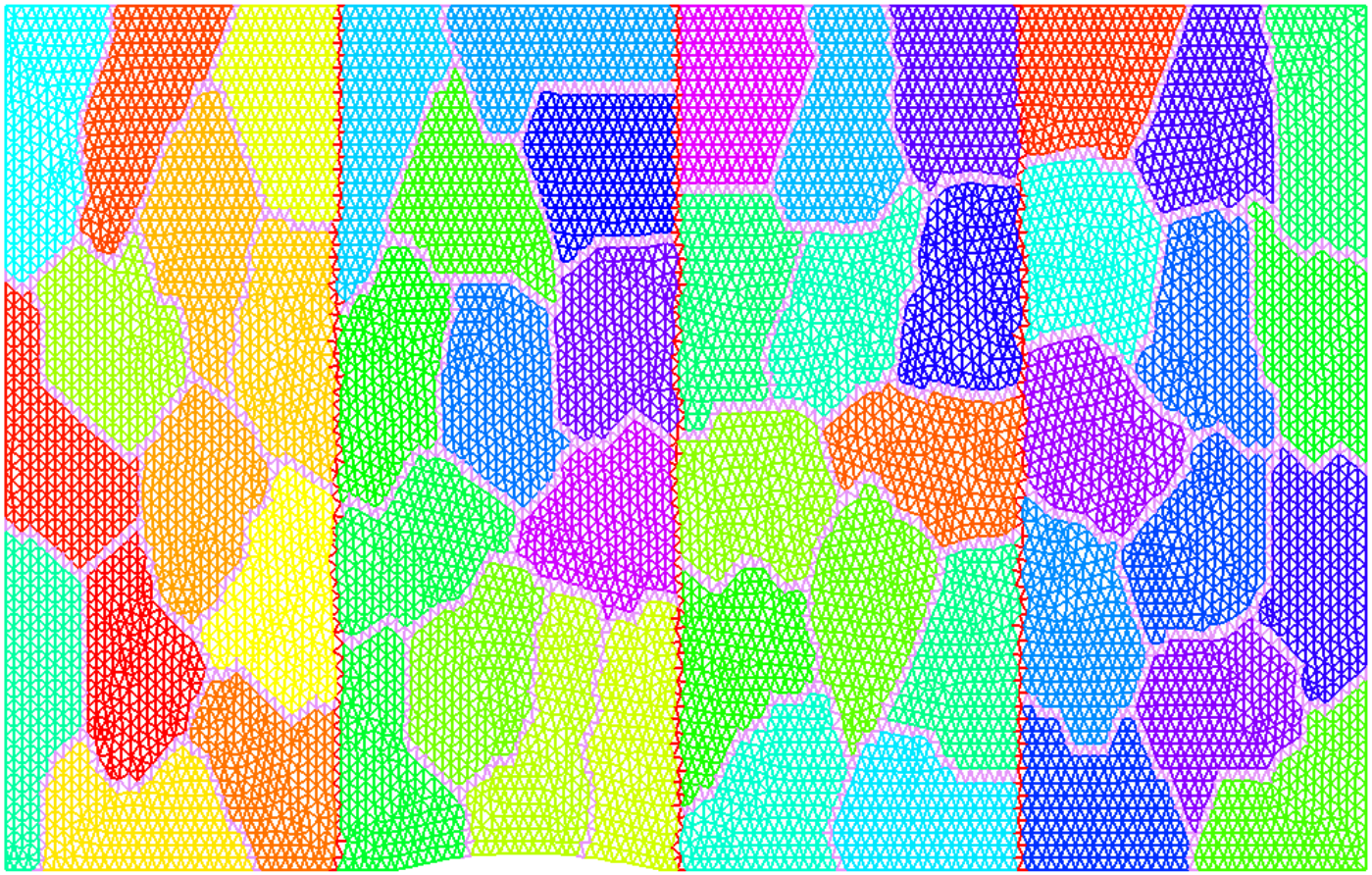
Добавление вершин в граф в процессе перераспределения плохих групп доменов

Mesh	Number of vertices in the graph	Number of bad subdomains (first case, second case)	time of the algorithm with the binary search, sec.	time of the algorithm where the proposed algorithm is applied, sec.
Mesh1	$2 \cdot 10^8$	51, 73	807.6	24.9
Mesh2	$2.6 \cdot 10^8$	19, 9	17.3	1.3
Mesh3	$1.1 \cdot 10^8$	1, 4	0.17	0.4

25600 доменов

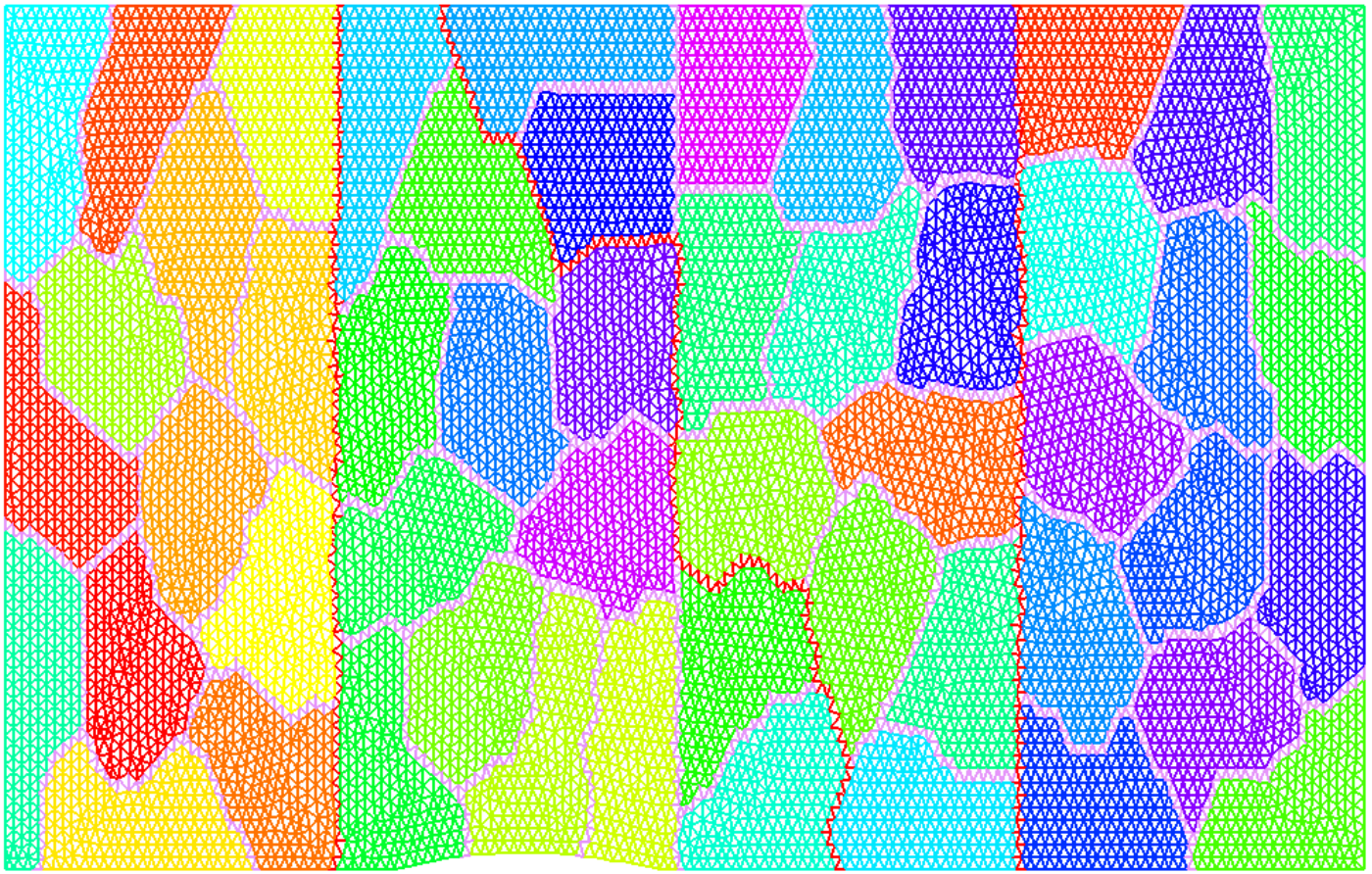
Домены перед перераспределением плохих групп доменов

Разбиение графа с 9800 вершинами на 52 домена на 4 процессорах

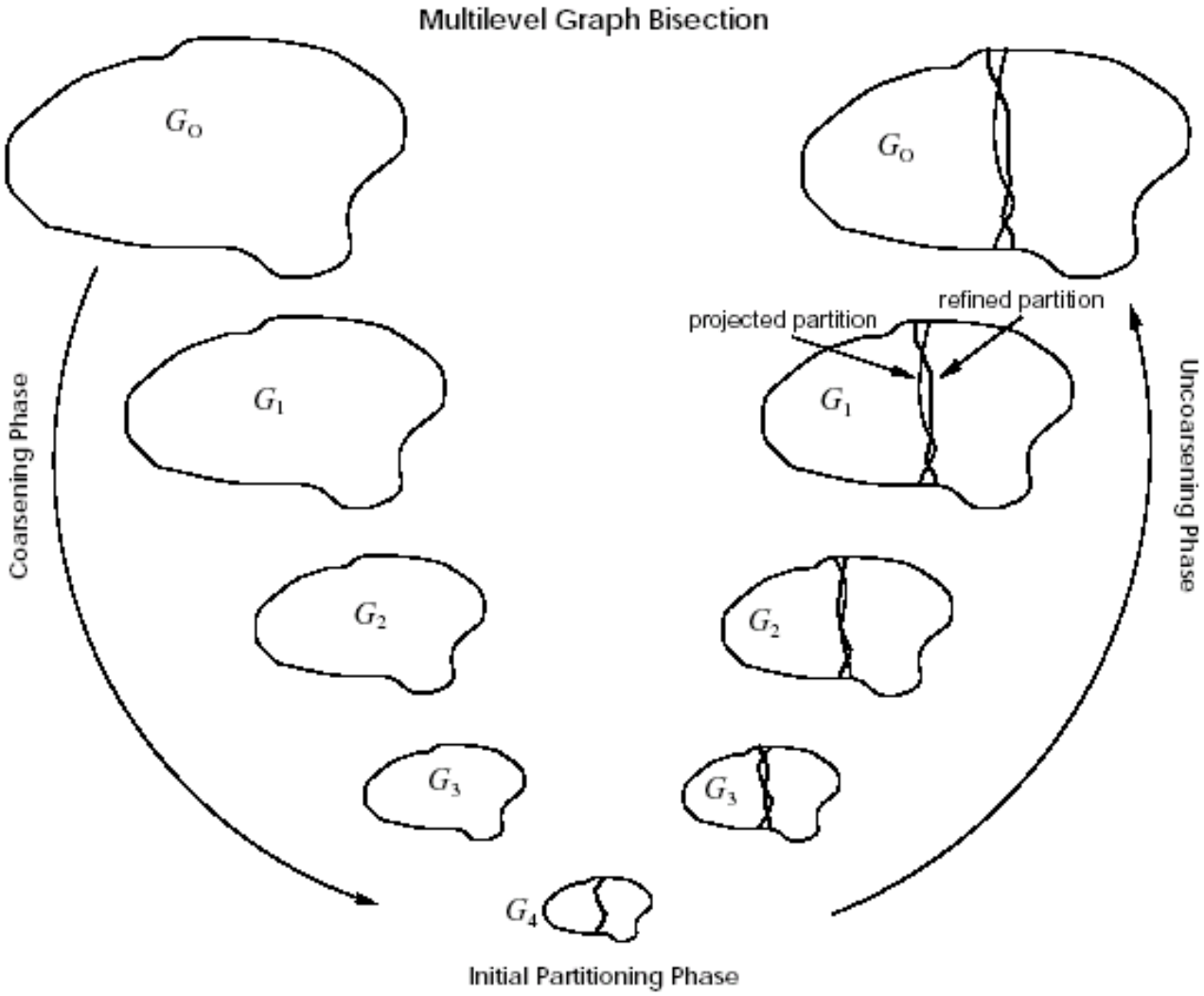


Домены после перераспределения плохих групп доменов

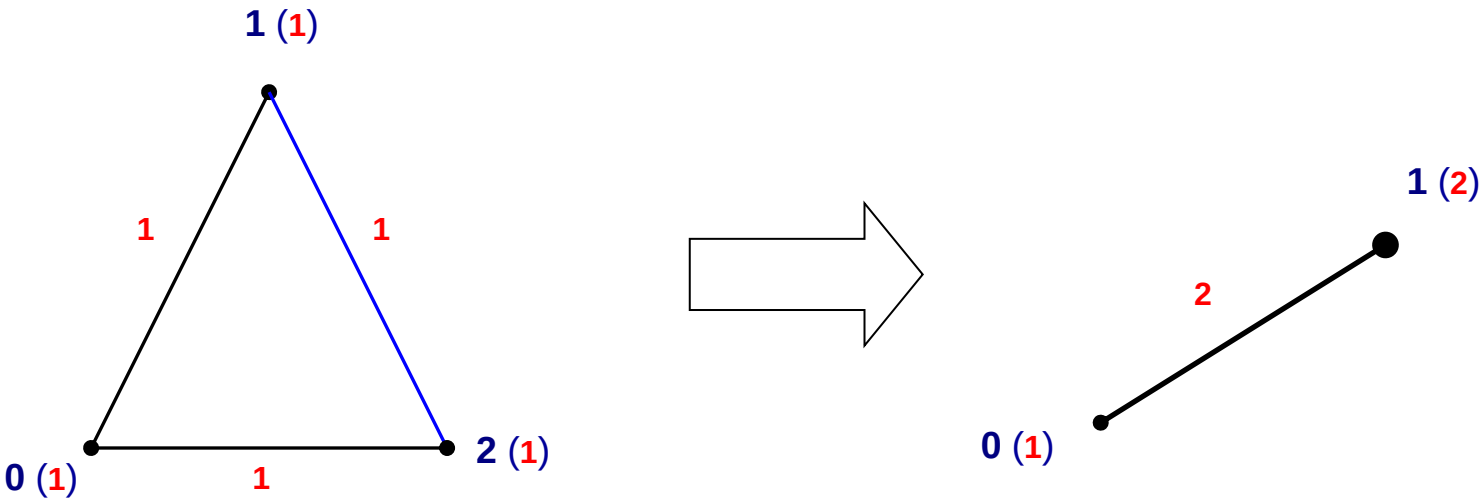
Разбиение графа с 9800 вершинами на 52 домена на 4 процессорах



Иерархический алгоритм декомпозиции графов

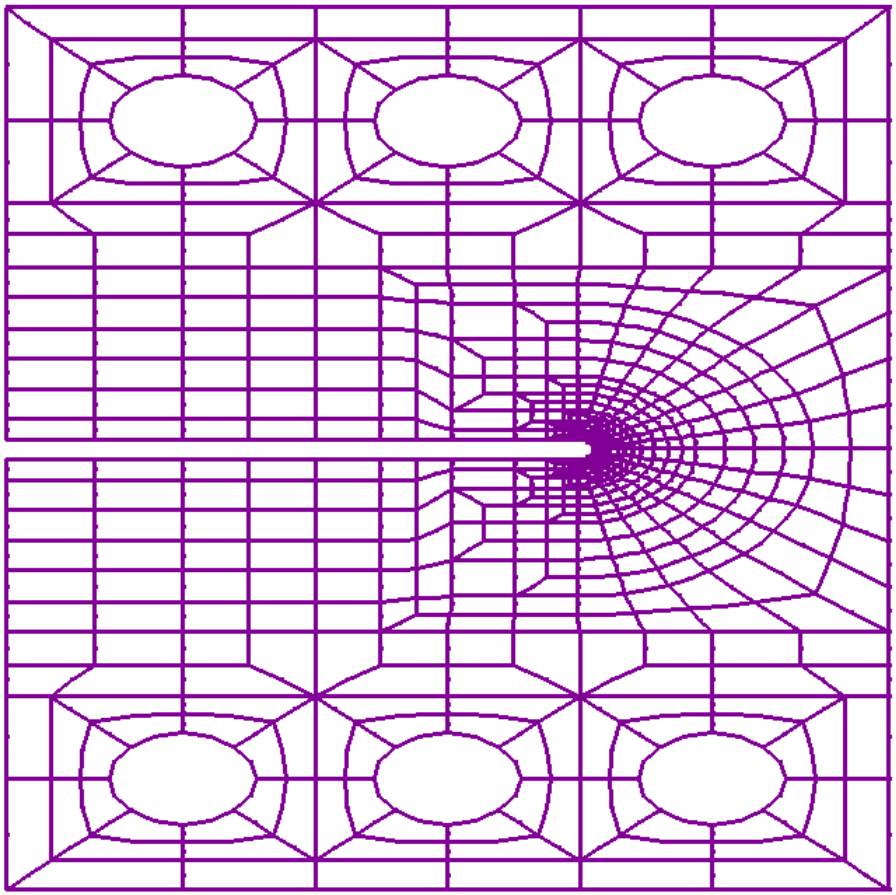


Огрубление графа



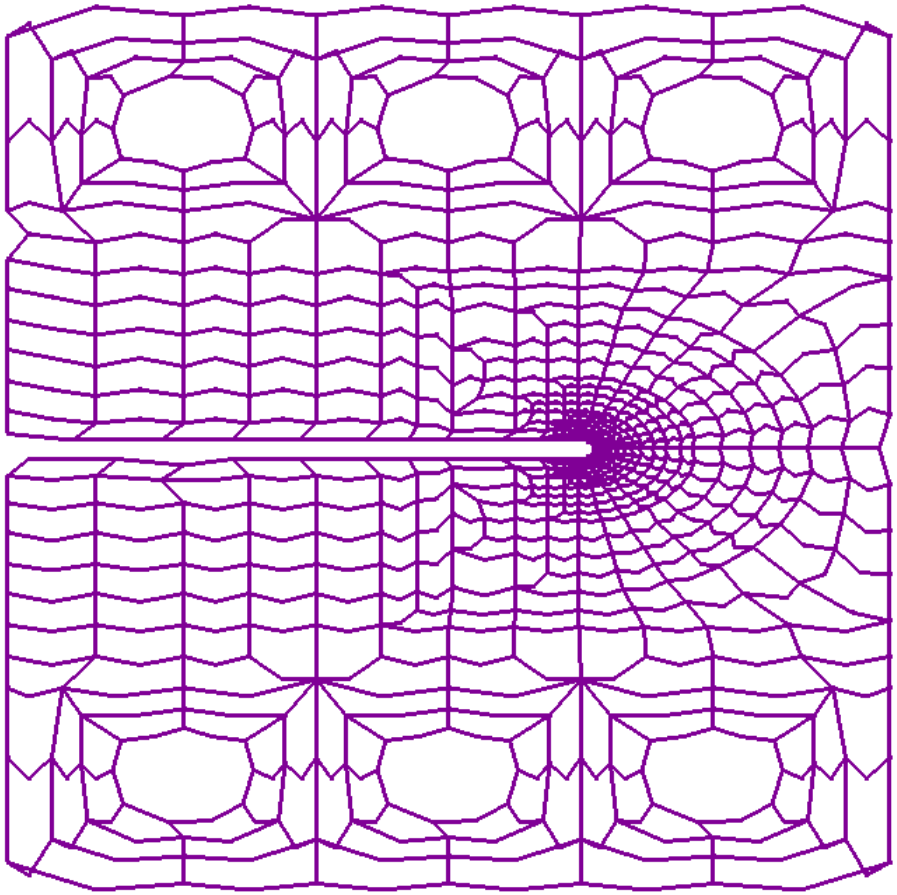
Огрубление графа

Небольшой начальный граф с 5981 вершиной



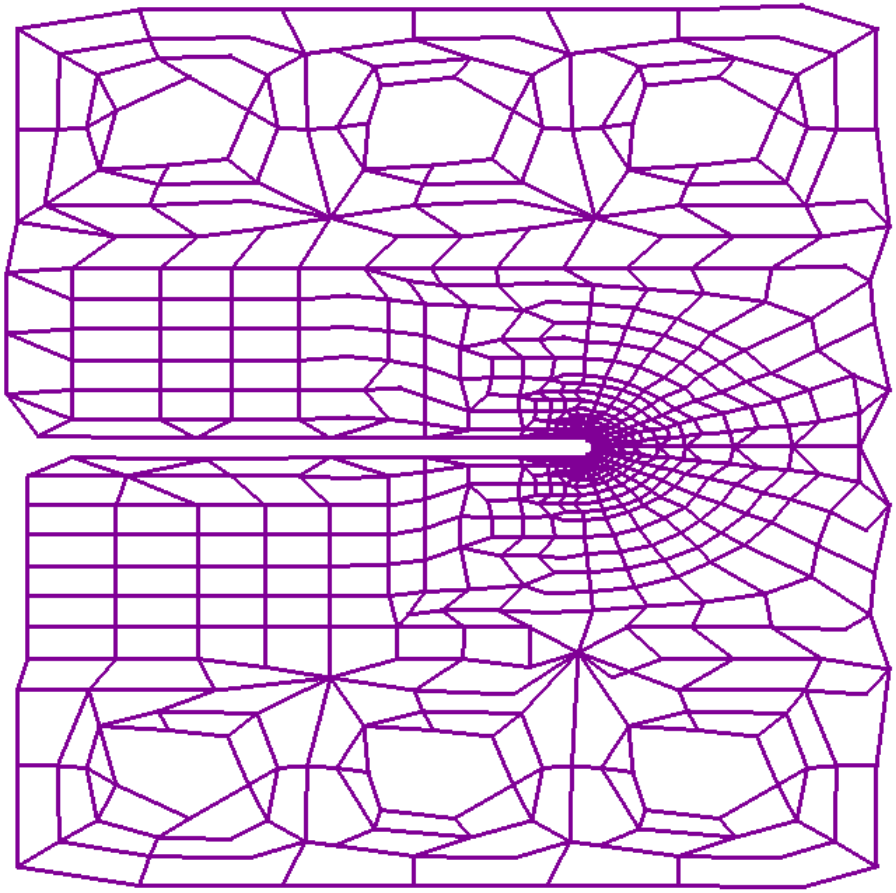
Огрубление графа

Граф после одного этапа огрубления



Огрубление графа

Граф после двух этапов огрубления

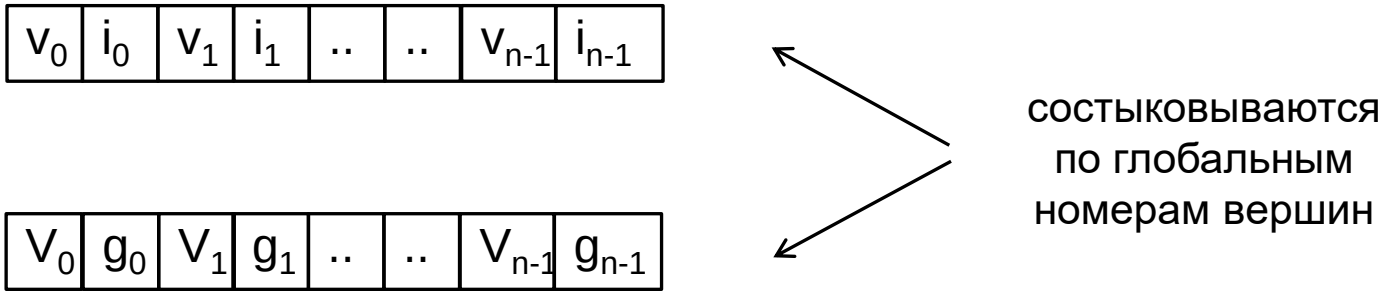


Параллельное восстановление графа

Обмен информацией о доменах вершин

- **родители вершины помещаются в тот же домен, что и вершина в огрубленном графе**

sort: $v_0 \le v_1 .. \le v_{n-1}$
$V_0 \le V_1 .. \le V_{n-1}$



v_k V_k – глобальные номера вершин
i_k – индекс (локальный номер) вершины
g_k – домен вершины

Выводы

- в пакете GridSpiderPar разработан параллельный алгоритм преобразования глобальных списков смежности в локальные в распределенном графе. Он является частью параллельного инкрементного алгоритма
- алгоритм работает даже с неоптимальной нумерацией вершин
- три тетраэдральные сетки с $2 \cdot 10^8$ вершин были разбиты параллельным инкрементным алгоритмом на 25600 микродоменов на 512 процессорах. Отображение глобальных номеров в локальные было выполнено двумя способами: с использованием бинарного поиска в отсортированном массиве и при помощи разработанного алгоритма. Разработанный алгоритм работает на двух сетках в 18 раз быстрее.
- разработанный алгоритм уменьшил в 32 раза время добавления вершин в граф при перераспределении плохих групп доменов на сетке с большим количеством плохих доменов

Выводы

- **разработанный алгоритм позволяет заменить последовательность поисков в большом массиве на один проход по двум отсортированным массивам в поиске совпадений и может быть применен в подобных алгоритмах**