Load Balancing for Domain Decomposition Problems

Irene Moulitsas
moulitsa@cyi.ac.cy

The Cyprus Institute

Outline

- **Problem Definition – Motivation**
- **Simultaneously Balancing Iterative and Direct Methods workload**
- **Experimental Results**
- **Summary**

Motivation

- **Traditional Graph Partitioning Techniques aim at balancing the number of vertices, while minimizing the edgecut.**
- **They can address qualities defined *a priori* on the Graph, such as vertex and edge weights.**

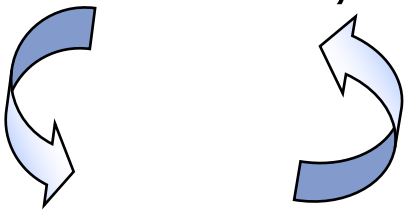
  BUT…
- **They fail to address dynamic issues, for example, to balance the computations performed by a per-subdomain direct factorization.**
- **Crucial for domain decomposition based numerical simulations, where subproblems corresponding to the subdomains are solved using sparse direct factorization methods (FETI).**
- **We would like to balance**
  - **the amount of time required to factor the local subproblem using direct factorization, and**
  - **the number of elements assigned to each processor.**

Challenges

- **The fill incurred in each submatrix depends on the topological structure of the partition.**

Chicken and egg problem

- **The fill cannot be estimated, until a partitioning is found.**



- **A *good* partitioning cannot be found unless we have an estimate of the fill.**

- **To date, there are no good methods for solving this problem.**
  - **Compute different partitioning followed by the associated fill-reducing orderings and pick the one that is most balanced!**
    - **Too expensive, not guaranteed to improve load imbalance.**

Our contribution

Predictor – Corrector Approach

Predictor Step

- We compute a high quality partition of the underlying graph.

Corrector Step

- We modify the partitioning, to achieve the desired balancing constraints.
  - Compute a fill-reducing ordering for each partition.
  - Modify the initial partitioning and the initial ordering in order to meet our objectives.

Our Goal

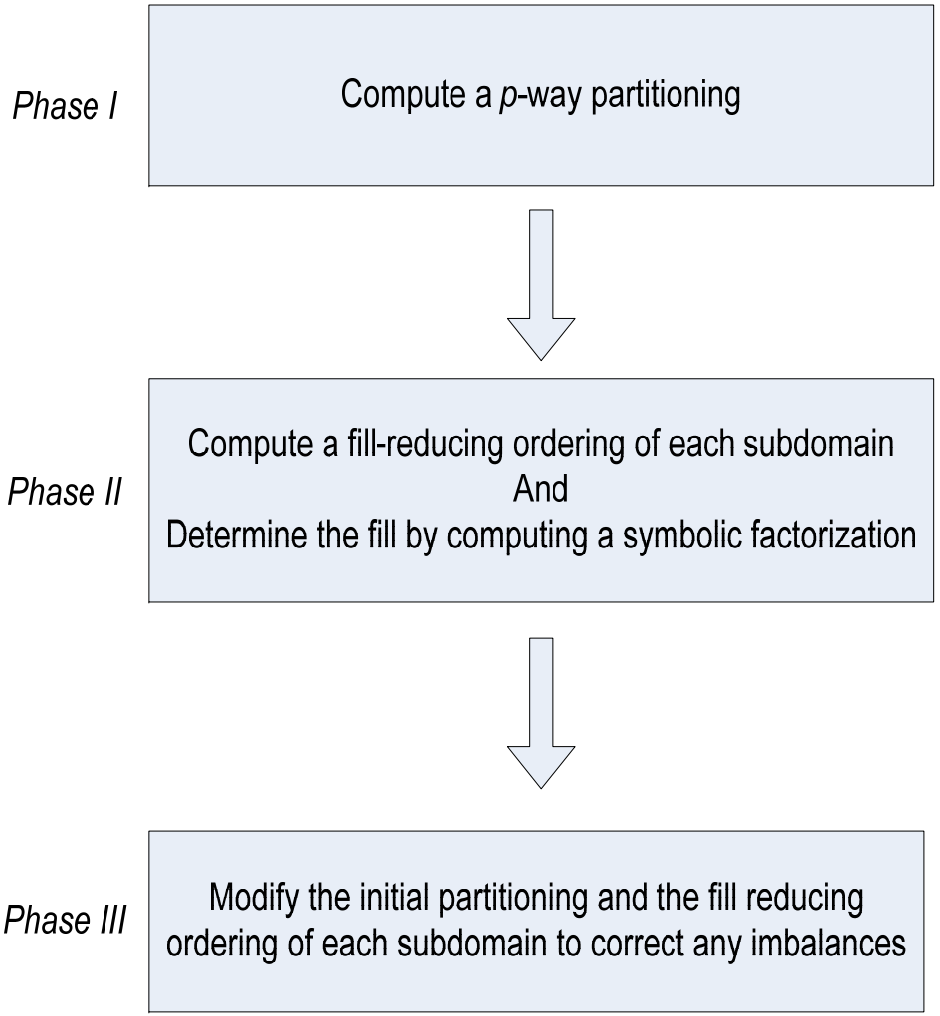
- Each partition has *n/p* vertices.

The fill incurred by each submatrix is the same (balanced).

The fill of each submatrix is minimized.

- The edgecut is minimized.

Algorithm Description

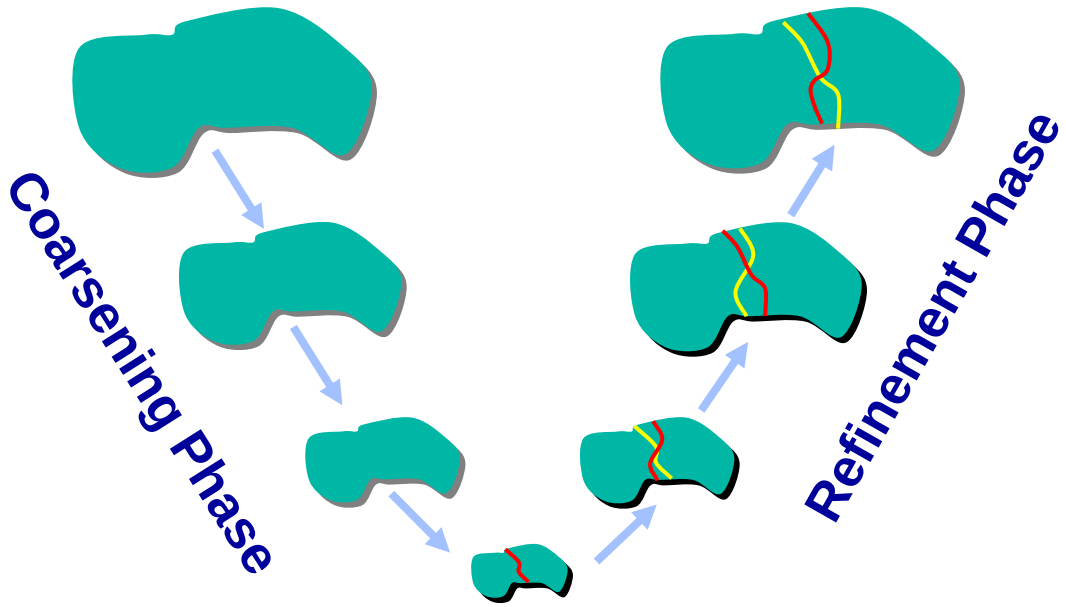


- **We employ METIS to obtain a high quality initial partitioning using its multilevel recursive bisection algorithm.**
- **METIS multilevel nested dissection algorithm is used to obtain the fill reducing ordering per subdomain.**
- **We propose different refinement partitioning schemes to achieve our goals.**

Phase I : Initial Partitioning

- **METIS yields an initial *p*-way partitioning that has**
  - **a balanced load in terms of the size of every subproblem,**
  - **minimal communication requirements,**
  - **low computational requirements.**

The Multilevel Paradigm For Graph Partitioning



A sequence of coarse graphs is constructed.

The partitioning is successively projected back to the original graph. At each finer graph, a refinement algorithm is applied.

A partitioning of the coarsest graph is computed quickly.

Phase II : Fill Reducing Ordering

- **Perform the Multilevel Nested Dissection Algorithm on each subproblem.**

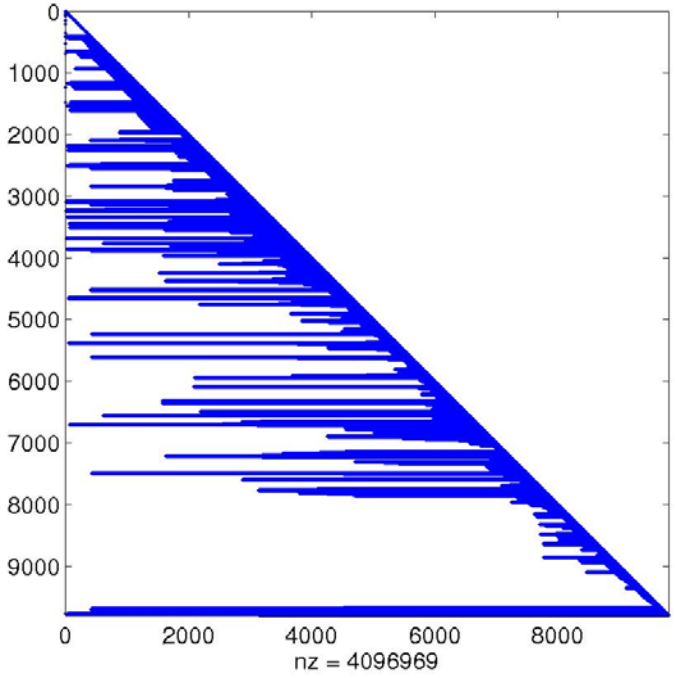
  <u>Divide and Conquer approach</u>

1. **Find a vertex separator *S*, that partitions the local graph into two balanced disconnected subgraphs *A* and *B*.**
2. **Order A and B, followed by S.**
3. **Repeat the same idea on *A* and *B*.**

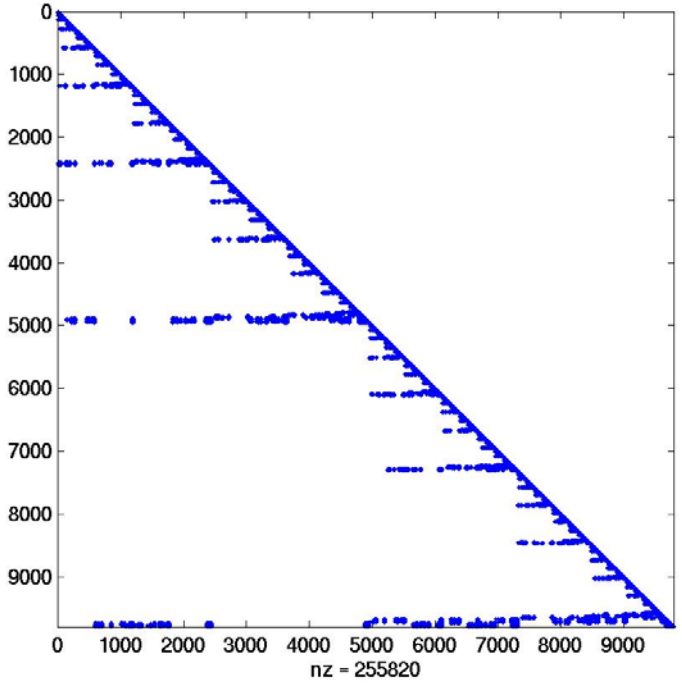
- **Separator S translates to a dense submatrix.**

Fill Reducing Ordering

The original Cholesky factor



The permuted Cholesky factor



We can manipulate the amount of fill by modifying the size of the separators.

Phase III : Modifying the ordering

Develop refinement strategies to modify the existing partitionings and orderings.

- **Reduce the separator size of partitions bearing separators whose size is above average.**
  - **Load imbalance is reduced.**
  - **High fill is reduced.**

- **Top level separator vertices are to be moved, while ensuring that the nested dissection qualities are not violated.**

- **Such movements are allowed until a larger separator has been sufficiently reduced.**

Defining our goals

$$F_i = c \cdot s_i^{\,2}, c = const$$

(1)

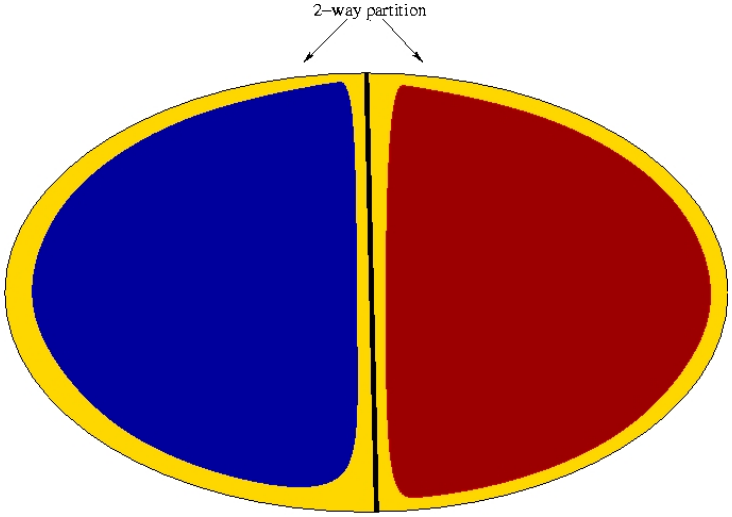
$$F_i^{'} = c \cdot s_i^{'2} \Leftrightarrow F_i^{'} = \frac{F_i}{s_i^{2}} \cdot s_i^{'2} \Leftrightarrow s_i^{'} = \sqrt{\frac{F_i^{'}}{F_i}} \cdot s_i$$

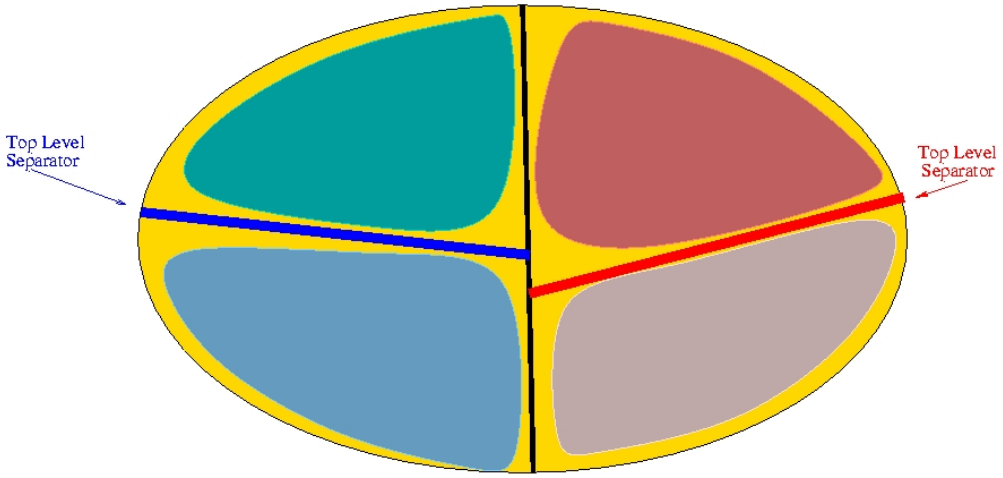
- **If we know the desired target fill of a subdomain, we can estimate the target size of the top level separator.**

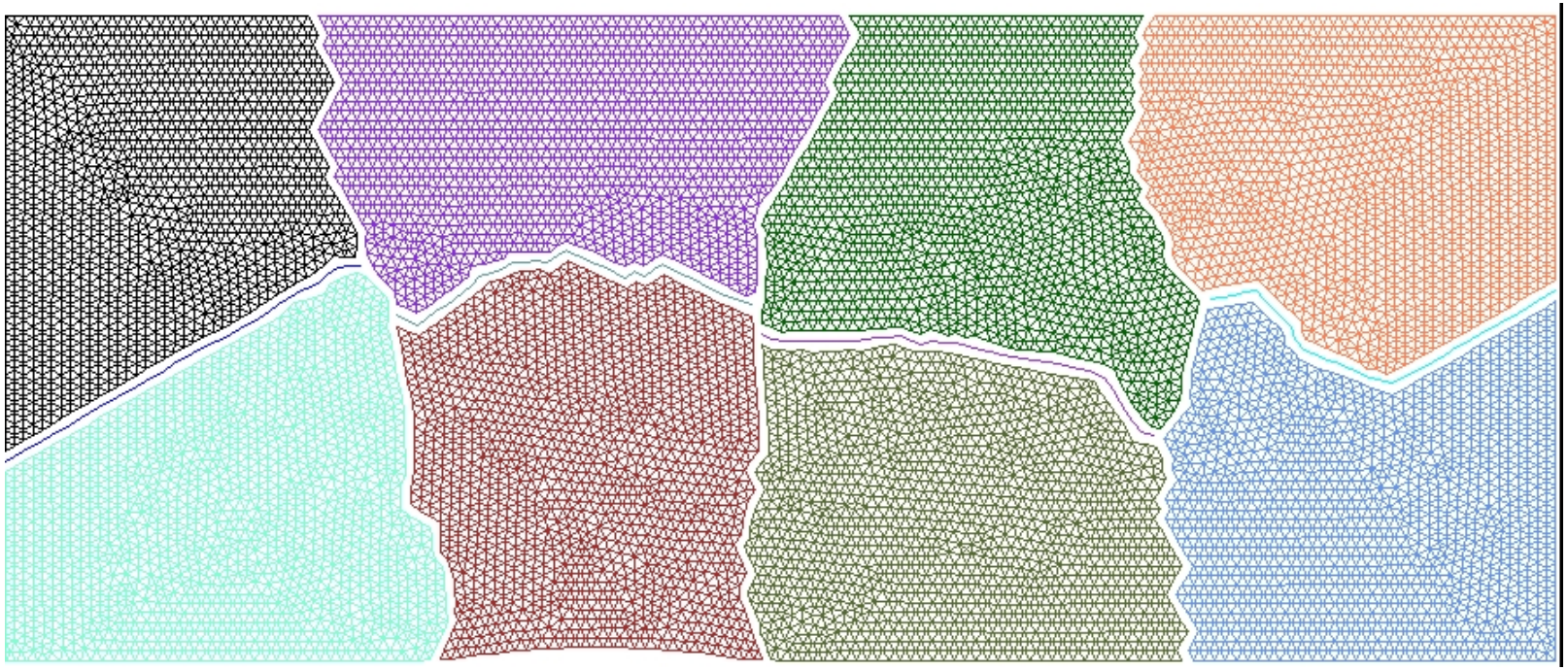
Refinement Procedure

- **All vertices are visited in a random order.**
- **Top level separator vertices are moved out of necessity. They will most likely be moved to adjacent partitions causing**
  - **an increase in the edgecut,**
  - **a perturbation to the fill in of the target partition, and**
  - **a perturbation to the load balance.**
- **Non-separator vertices will be moved in order to cure the afore negative effects.**

Example







Modeling the Fill

$$F_i = n_i^{\alpha}, \alpha = const \Leftrightarrow \alpha = \frac{\log F_i}{\log n_i}$$

(2)

$$F_i^{'} = n_i^{'\alpha} \Leftrightarrow F_i^{'} - F_i = n_i^{'\alpha} - n_i^{a} \Leftrightarrow F_i^{'} = F_i + n_i^{'\frac{\log F_i}{\log n_i}} - n_i^{\frac{\log F_i}{\log n_i}}$$

- **If we perform a move we can estimate the new fill in of both the originating and target partition using the above model.**

Proposed Refinement Schemes

RS1

- Find target separator size using (1)
- Perform refinement
- Update fill using (2)

RS2

- Find target separator size using (1)
- Perform refinement
  - edgecut degradation is relaxed progressively
- Update fill using (2)

RS4

- Find target separator size using (1)
- Perform refinement
  - edgecut degradation is relaxed progressively
- Fill is computed explicitly

RS3

- Find target separator size using (1)
- Perform refinement; edgecut degradation is
  - relaxed progressively,
  - proportional to the size of the separator
- Update fill using (2)

RS5

- Find target separator size using (1)
- Perform refinement; edgecut degradation is
  - relaxed progressively
- Update separator sizes
  - assume larger separator sizes incur higher fill
  - base move decisions solely on this assumption

Experimental Results

Data Sets

	Name	#Vertices	#Edges	Description
1	144	144,649	1,074,393	Graph corresponding to a 3D FEM mesh of a parofoil
2	auto	448,695	3,314,611	Graph corresponding to a 3D FEM mesh of GM's Saturn
3	bcsstk29	13,992	302,748	Graph corresponding to Buckling model of a Boeing 767 rear pressure bulkhead
4	brack2	62,631	366,559	Graph corresponding to a 3D FEM mesh of a bracket
5	cylinder93	45,594	1,786,725	Graph of a 3D stiffness matrix
6	f16	1,124,648	7,625,318	Graph corresponding to a 3D FEM mesh of an F16 Wing
7	f22	428,748	3,055,361	Graph corresponding to a 3D FEM mesh of an F22 Wing
8	finan512	74,752	261,120	Graph of a stochastic programming matrix for financial portofolio optimization
9	inpro1	46,949	1,117,809	Graph of a 3D stiffness matrix
10	m6n	94,493	666,569	Graph corresponding to a 3D FEM mesh of an M6 wing

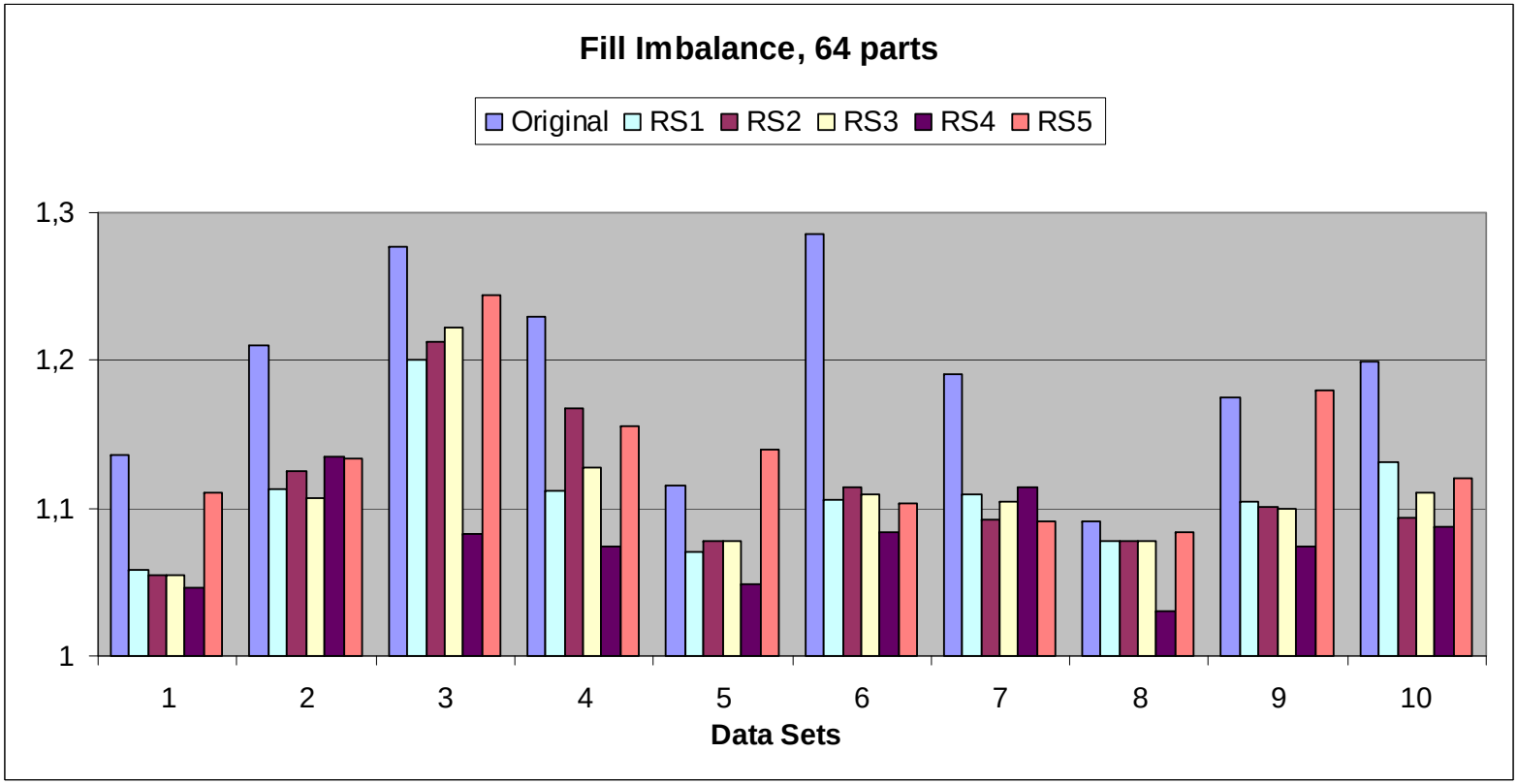
- **We evaluate the different refinement strategies against four different quality measures:**

1. **Fill (Im)balance assessment.**
2. **Maximum Fill assessment.**
3. **Average Fill assessment.**
4. **Edgecut assessment.**

Quality Measures…

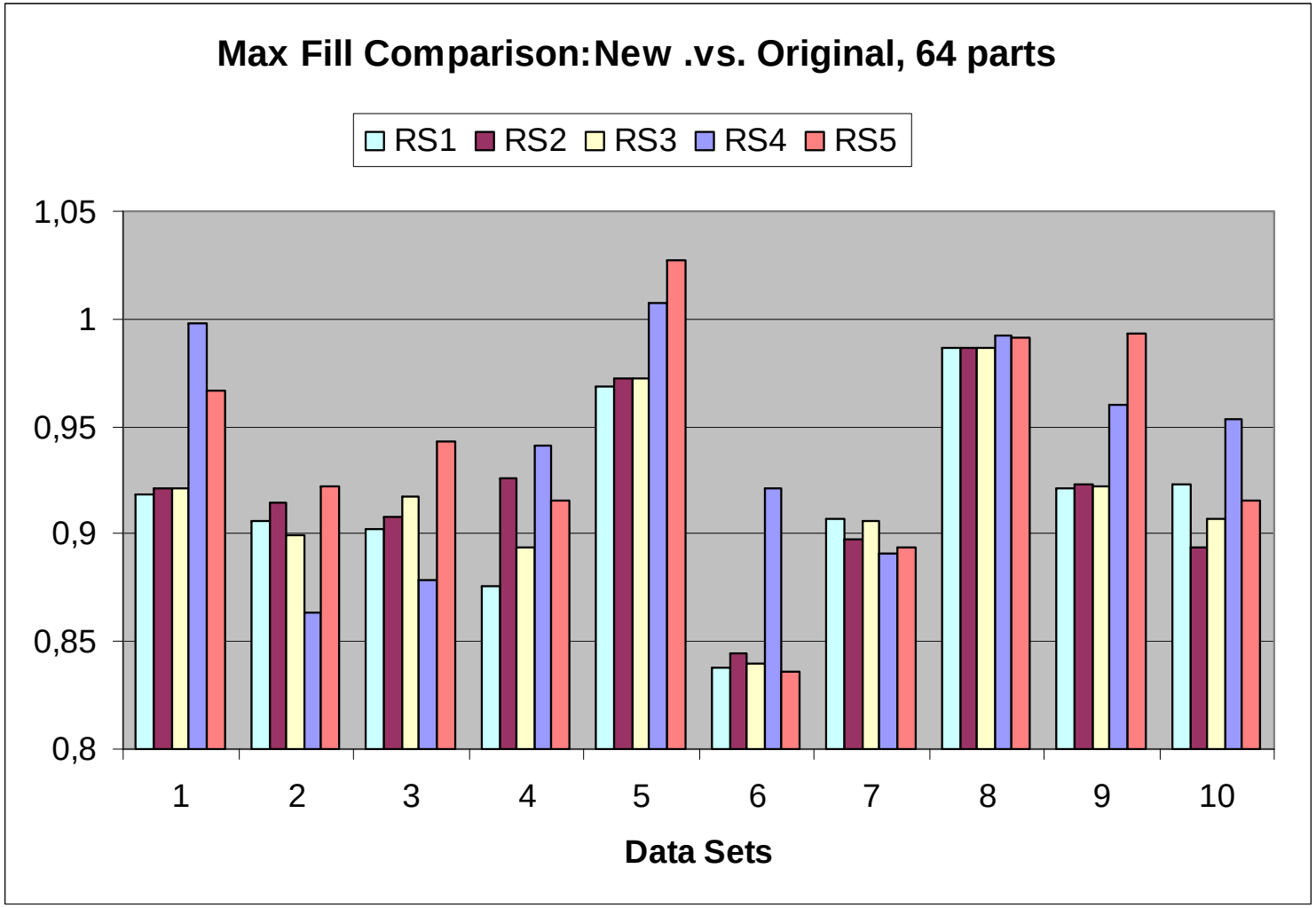
Fill (Im)balance assessment

$$\text{Fill Imbalance}(P) = \frac{\max_i(F_i)}{(\sum_i F_i)/p}$$



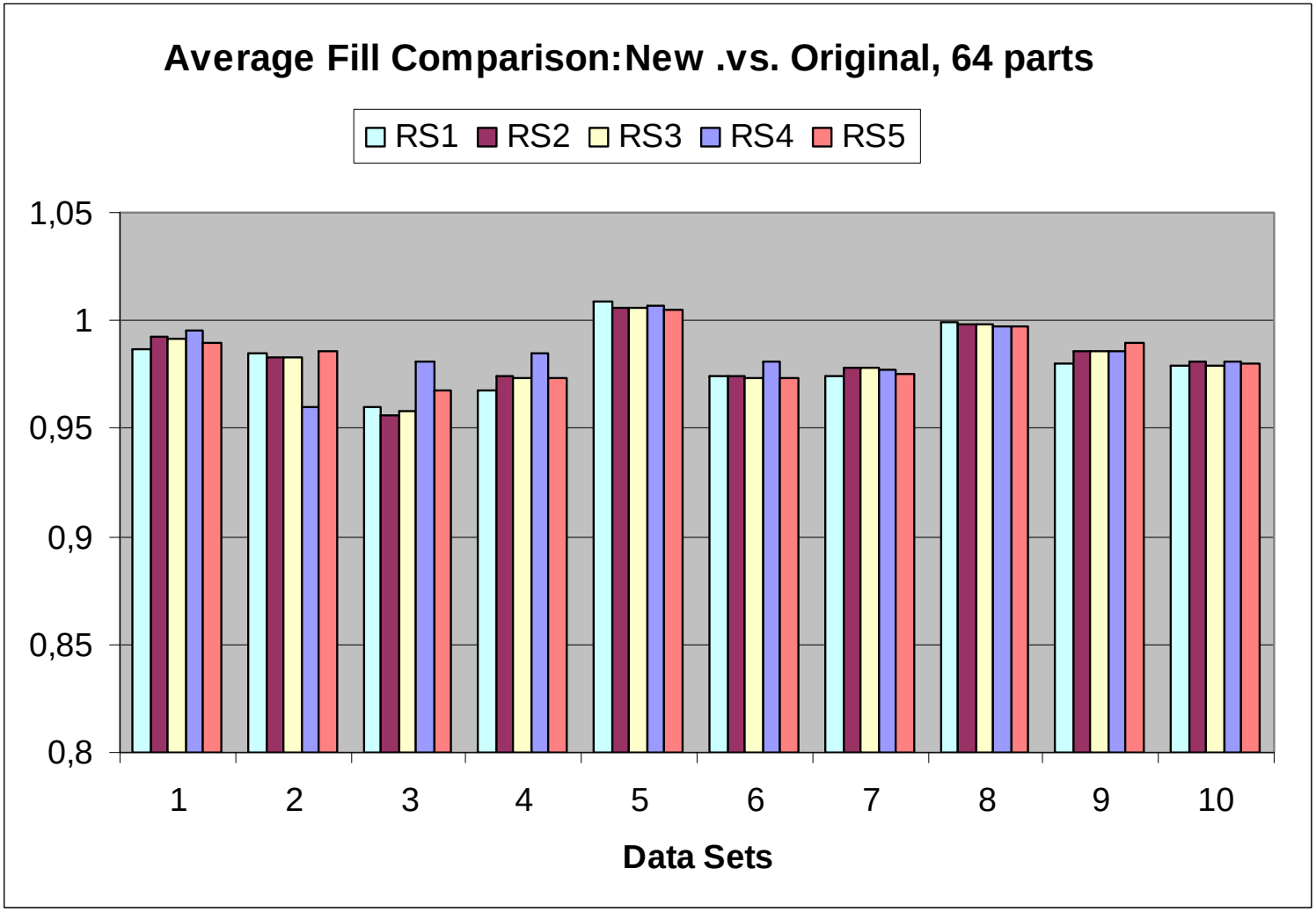
…Quality Measures…

Maximum Fill assessment



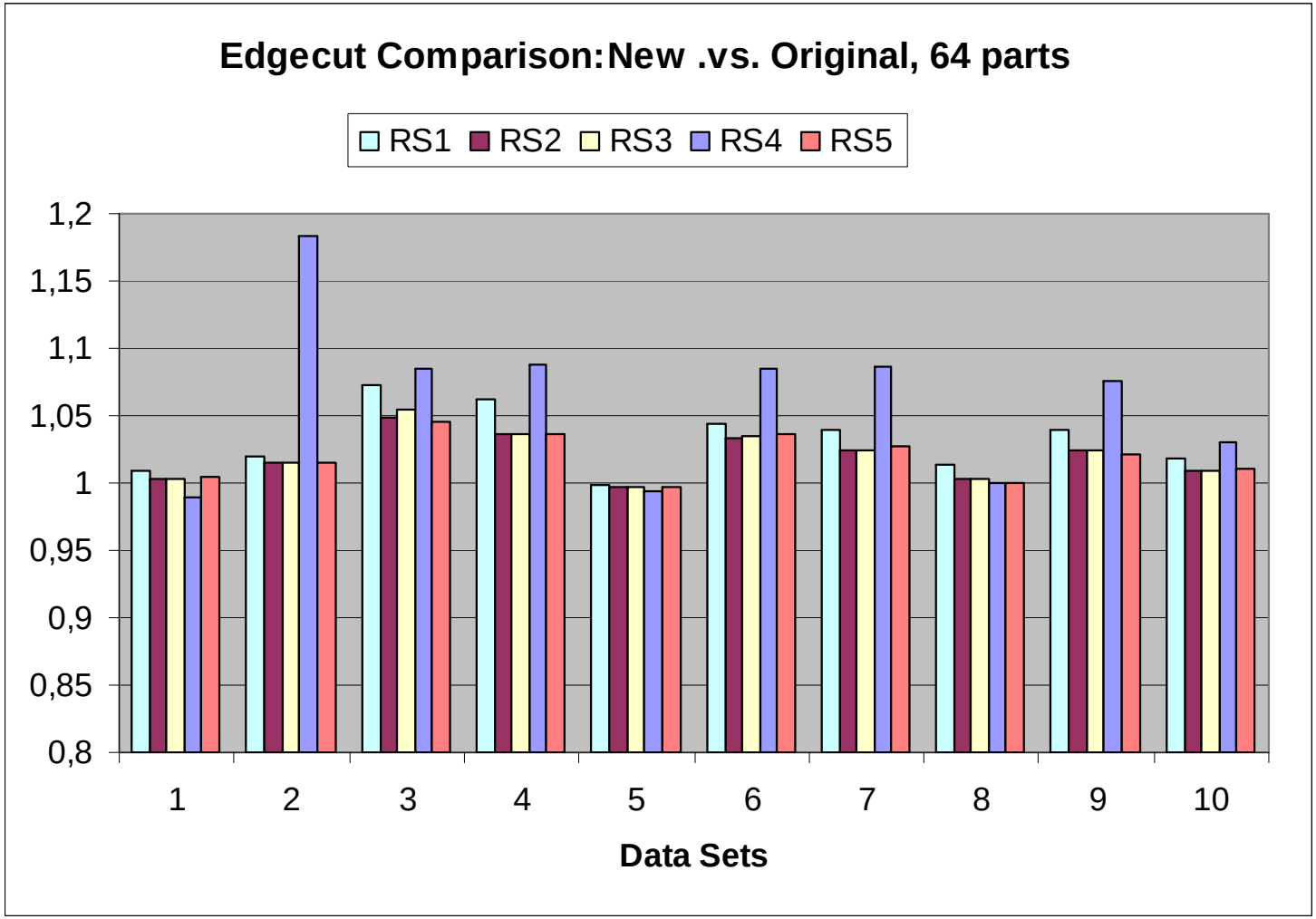
…Quality Measures…

Average Fill assessment



…Quality Measures

Edgecut assessment



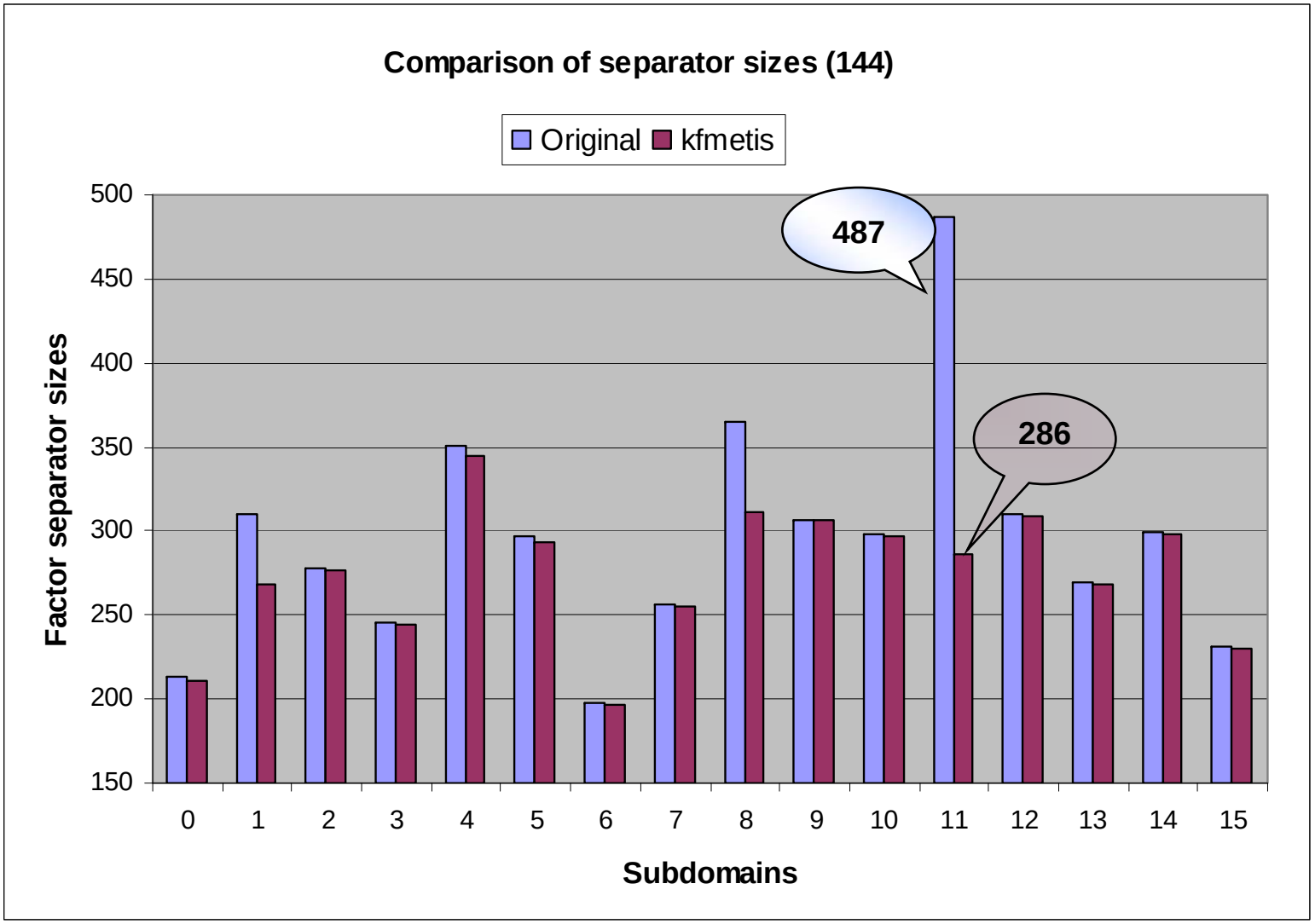
Computational Requirements

Run Times (in secs)

	Original	RS1	RS2	RS3	RS4	RS5
144	14.46	27.07	27.08	27.06	112.65	27.00
auto	60.24	116.47	116.86	116.78	494.00	116.63
b29	1.05	1.57	1.62	1.59	4.29	1.6
brack2	3.66	6.16	6.09	6.10	25.89	6.07
cylinder93	9.78	15.96	15.99	16.02	70.73	15.37
f16	157.98	275.27	275.70	277.13	1361.49	275.11
f22	52.10	90.18	90.72	90.72	434.39	90.68
finan512	4.01	6.31	6.29	6.31	16.24	6.78
inpro1	5.53	8.55	8.88	8.87	29.14	8.90
m6n	7.94	13.50	13.49	13.49	60.39	13.47

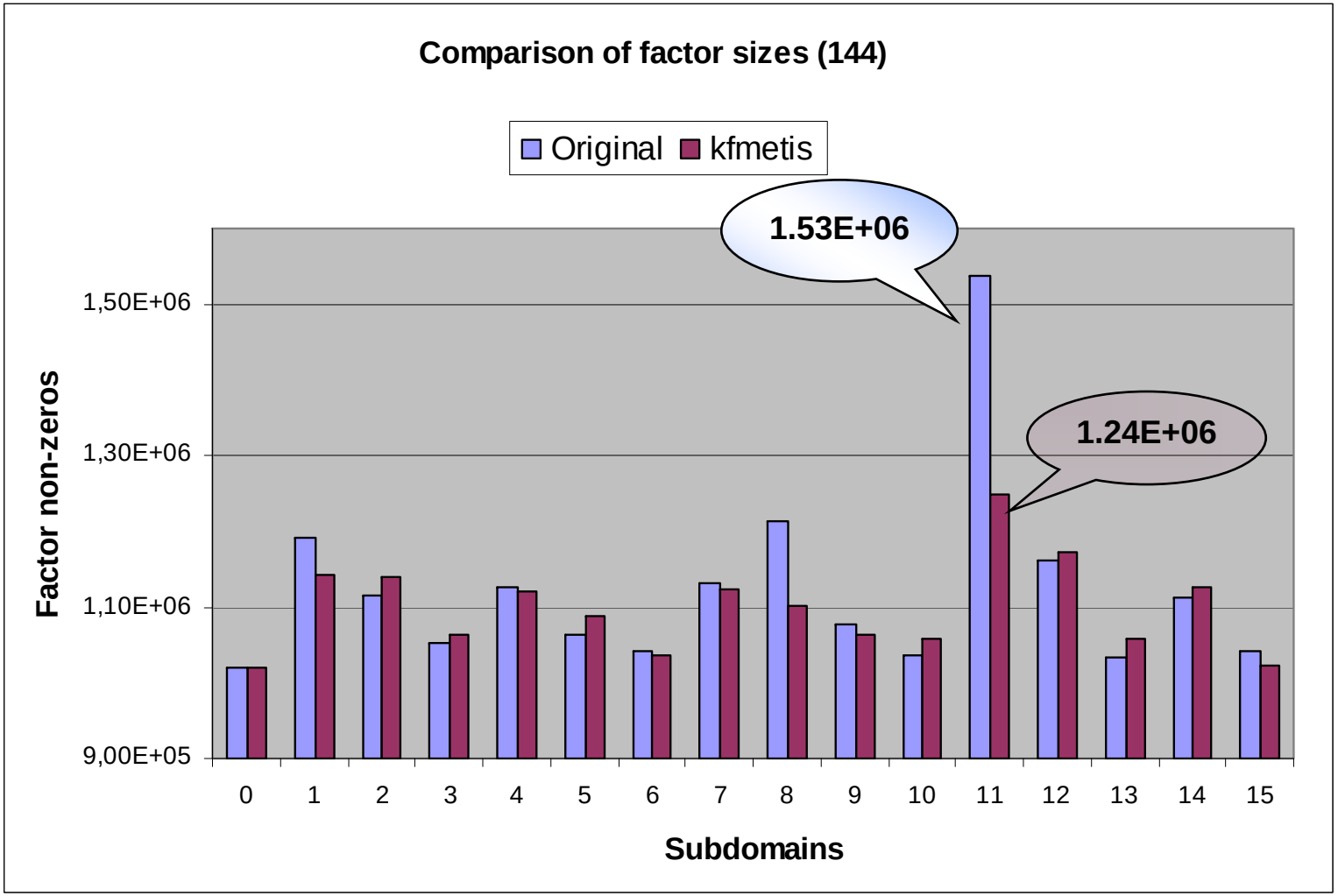
Partitioning Characteristics…

Separator Sizes



…Partitioning Characteristics

Factor Sizes



Summary of Work

New algorithms for computing

- **a partitioning and**
- **a per-subdomain fill reducing ordering**
  - **Simultaneously balance**
    - **the number of elements assigned to each processor**
    - **The fill of the subdomain matrices**
  - **Minimizing Edgecut**

- **Incorporated to the new METIS release.**

Thank You !