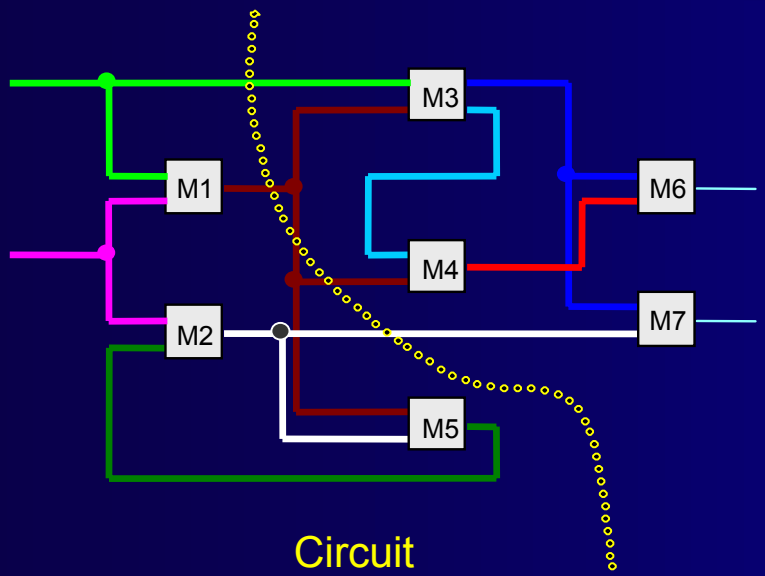
# Multilevel Algorithms for Hypergraph Partitioning: Single & Multiple Constraints

George Karypis
karypis@cs.umn.edu
Department of Computer Science & Engineering
University of Minnesota

# Circuit Partitioning

Given a circuit consisting of a set of *modules* and a set of *signal nets*, we want to divide it into clusters such that the number of inter-cluster signal nets is minimized subject to some balance constraints.



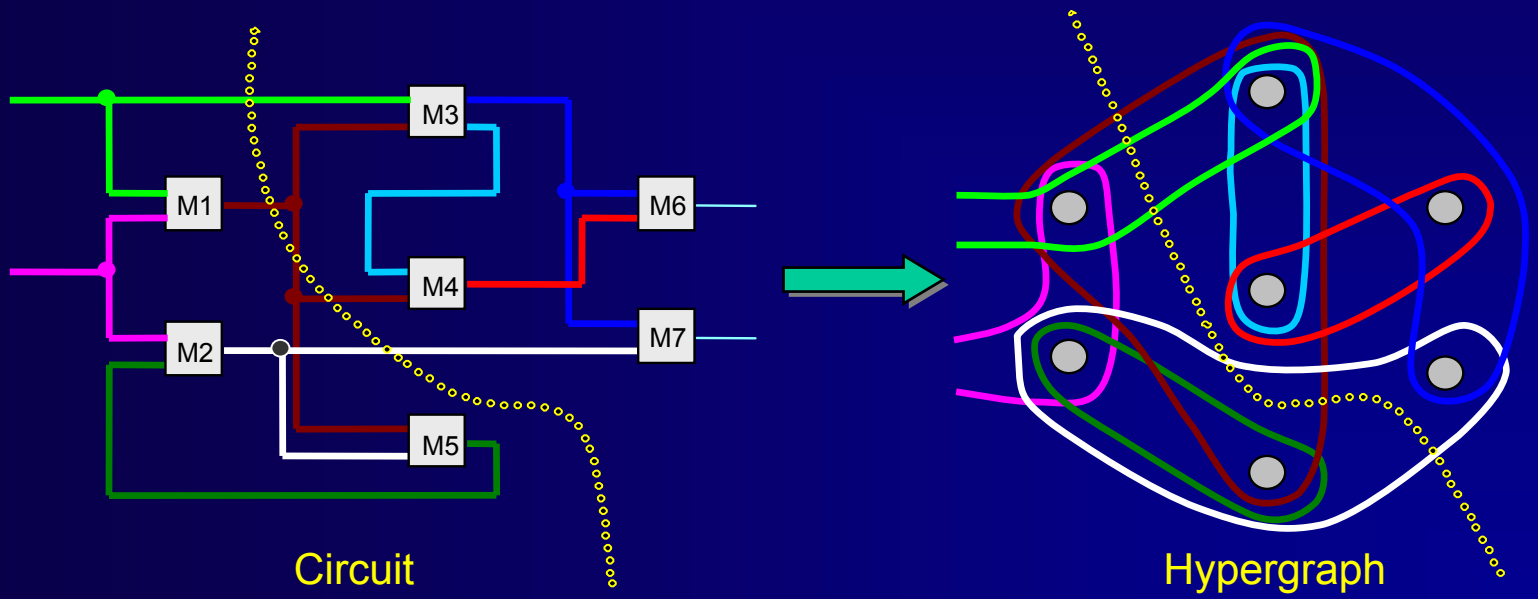

Circuit

## Applications

- Placement
- Design Packaging
- Synthesis
- Design Optimization
- Simulation & Test

# Circuit Partitioning

Given a circuit consisting of a set of *modules* and a set of *signal nets*, we want to divide it into clusters such that the number of inter-cluster signal nets is minimized subject to some balance constraints.




**Circuit Partitioning** → **Hypergraph Partitioning**

# METIS & ParMETIS

- Software packages for partitioning unstructured graphs and computing fill reducing orderings.
  - METIS was released in 1995 (current version 4.0).
  - ParMETIS was released in 1997 (current version 3.0).
- They are freely distributed.
- They are used extensively:

  National Labs
  - Livermore, Los Alamos, Sandia, Argonne, JPL, NASA, Oak Ridge, ...

  Defense Labs
  - CAA, CE-WES, ARL, NRL, AHPCRC, ...

  Commercial Organizations
  - Sun, HP, NEC, Ansys, Boeing, Ford, Rockwell, MSC, Centric, HKS, AEA, Adapco, Altair, CSAR, STAR Inc., NAG, ...

  Numerous Educational Institutions Worldwide

Can we just do the same for hypergraphs?

# hMETIS (http://www.cs.umn.edu/~metis)

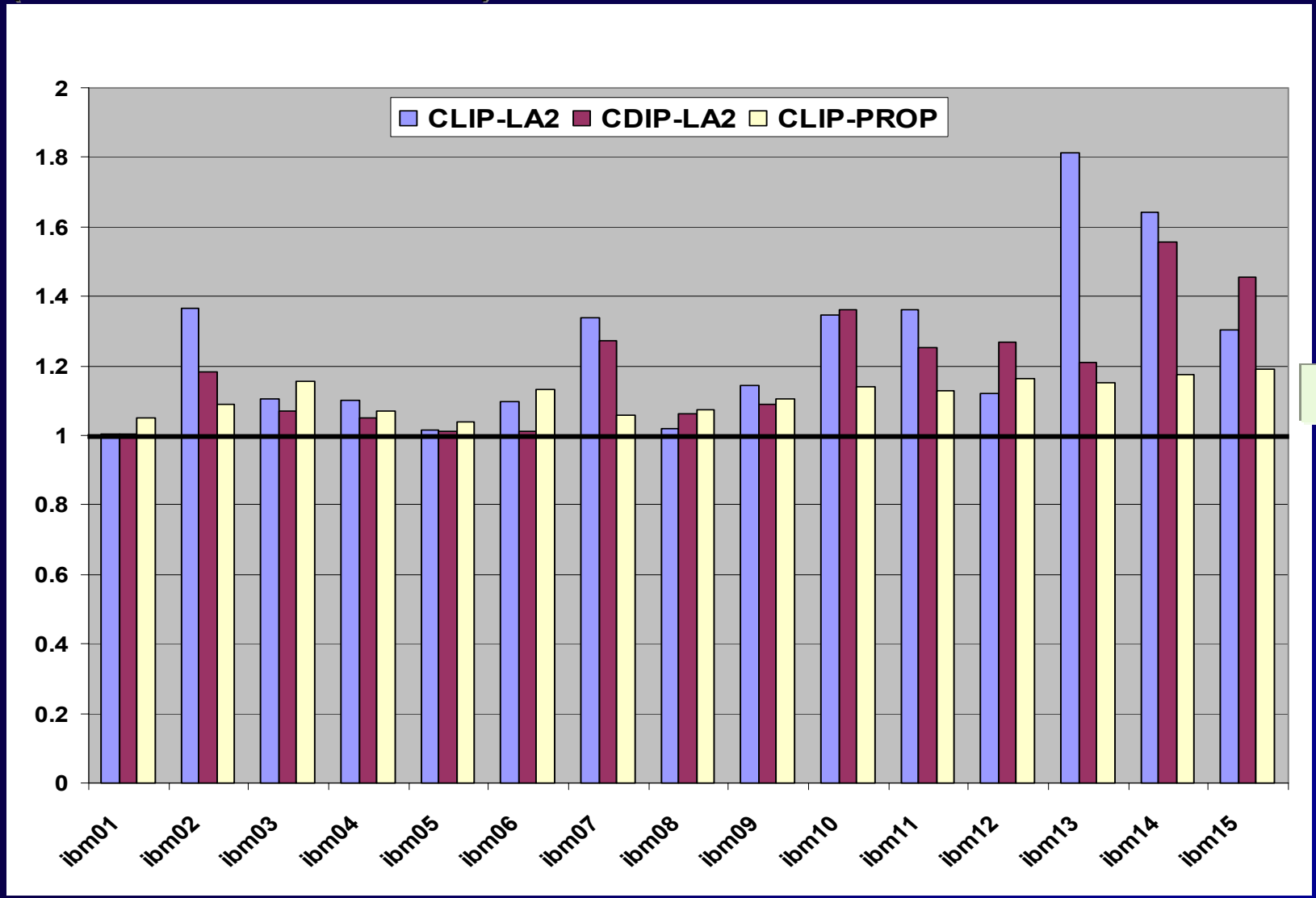
- A set of multilevel-based algorithms for partitioning hypergraphs
  - hMETIS initially released in 1997 (current version 1.5.3)
  - Contains both recursive-bisection as well as direct *k*-way partitionings
  - It is reasonably fast
  - It produces high-quality partitions
    - Both in terms of edge-cut, (k-1)-metric, or SOED.
  - Widely available & used

# Experimental Results

- We evaluated the performance of hMETIS on the ISPD98 benchmark suite.
- We compared it against the state-of-the art partitioners
  - CLIP-LA2
  - CDIP-LA2
  - CLIP-PROP

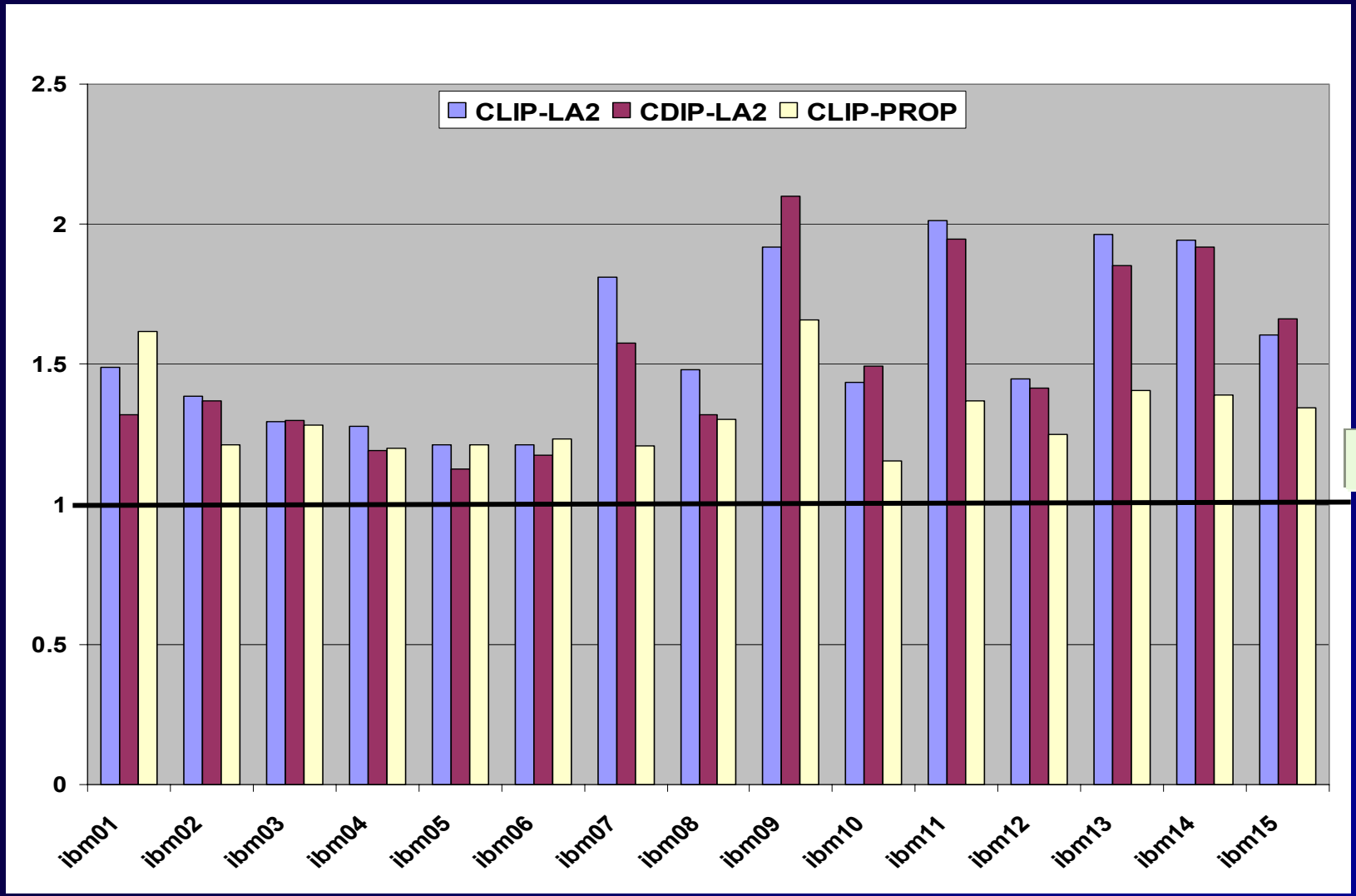
| | *Number of Vertices* | *Number of Hyperedges* |
|---|---|---|
| ***Ibm01*** | 12,506 | 14,111 |
| ***Ibm02*** | 19,342 | 19,584 |
| ***Ibm03*** | 22,853 | 27,401 |
| ***Ibm04*** | 27,220 | 31,970 |
| ***Ibm05*** | 28,146 | 28,446 |
| ***Ibm06*** | 32,332 | 34,826 |
| ***Ibm07*** | 45,639 | 48,117 |
| ***Ibm08*** | 51,023 | 50,513 |
| ***Ibm09*** | 53,110 | 60,902 |
| ***Ibm10*** | 68,685 | 75,196 |
| ***Ibm11*** | 70,152 | 81,454 |
| ***Ibm12*** | 70,439 | 77,240 |
| ***Ibm13*** | 83,709 | 99,666 |
| ***Ibm14*** | 147,088 | 152,772 |
| ***Ibm15*** | 161,187 | 186,608 |

# Mincut Performance of hMETIS (best of 10 runs)



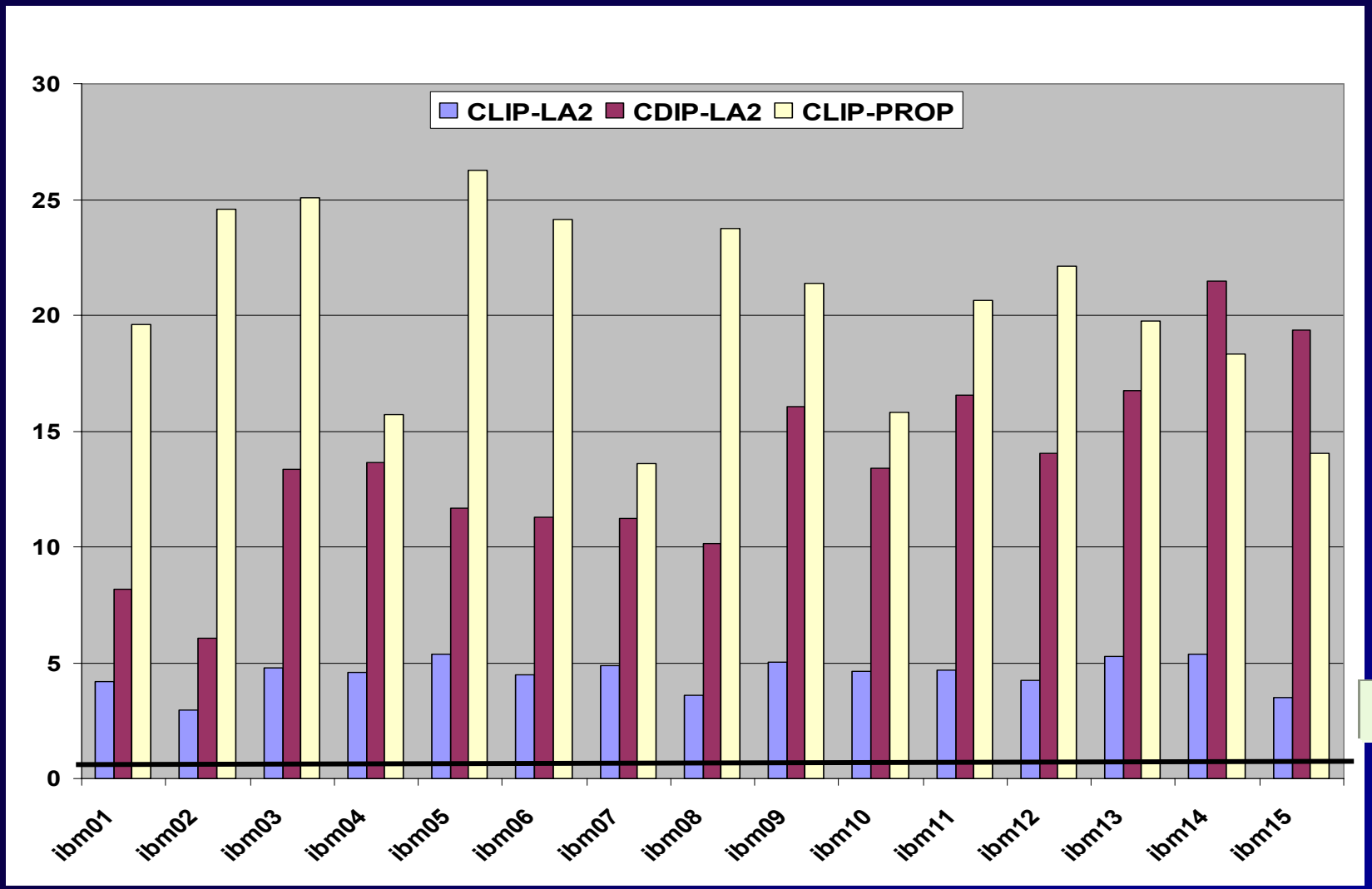

**CLIP-LA2: 25% worse** **CDIP-LA2: 19% worse** **CLIP-PROP: 12% worse**

# Average Cut Performance of hMETIS
## (average of 10 runs)




**CLIP-LA2: 57% worse** **CDIP-LA2: 52% worse** **CLIP-PROP: 32% worse**

# Runtime Performance of hMETIS




hMETIS

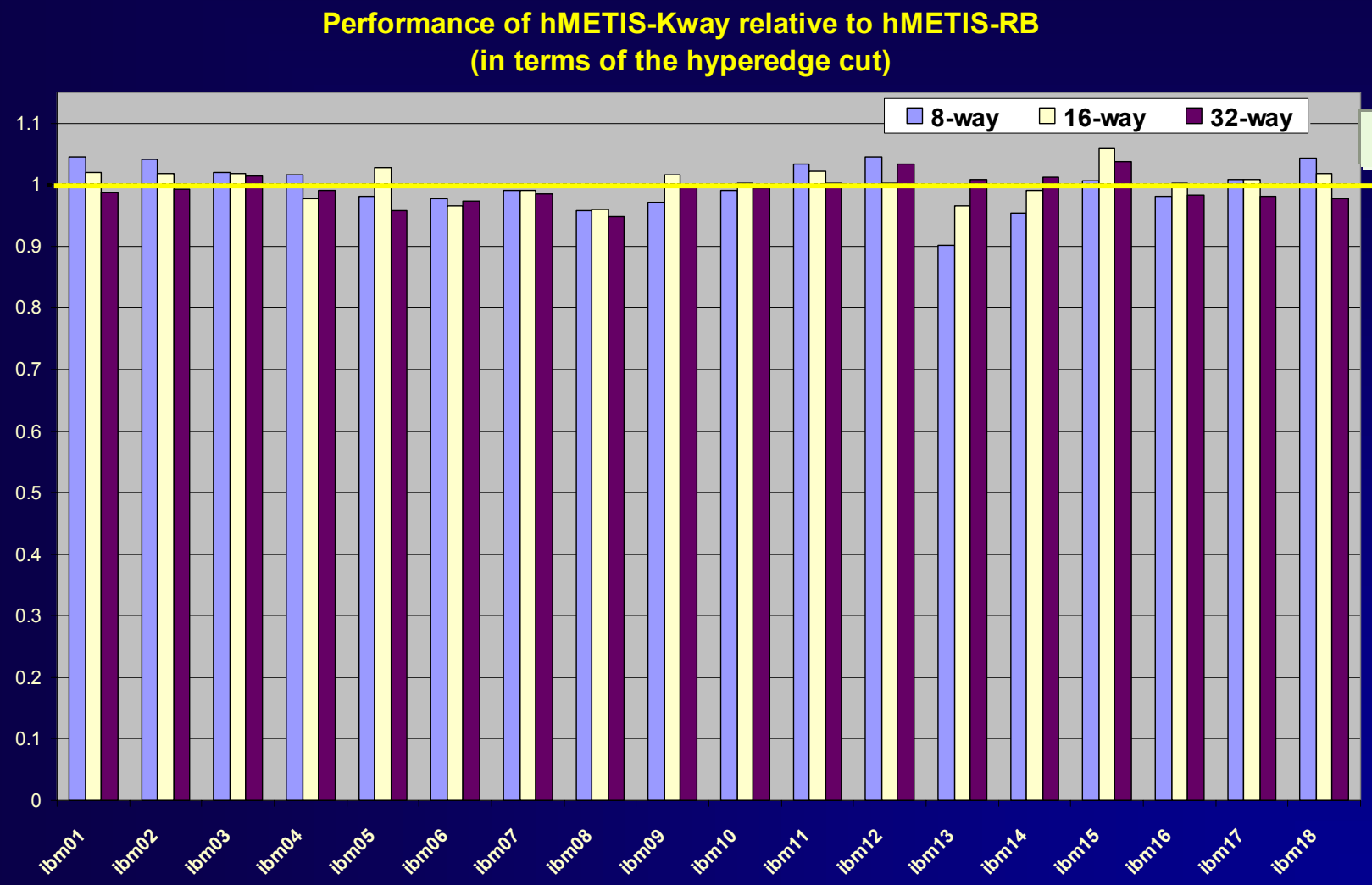
**CLIP-LA2: 4x slower** **CDIP-LA2: 14x slower** **CLIP-PROP: 17x slower**

# Experimental Results

- We evaluated the performance of hMETIS-kway on the ISPD98 benchmark suite.
- We compared it against the state-of-the art partitioners
  - hMETIS-RB (DAC 97)
  - K-PM/LR (ICAD98)
- hMETIS-Kway used a 10% load imbalance tolerance.
- hMETIS-RB used a [48-52] balancing constraint.
- K-PM/LR used a [45-55] balancing constraint.

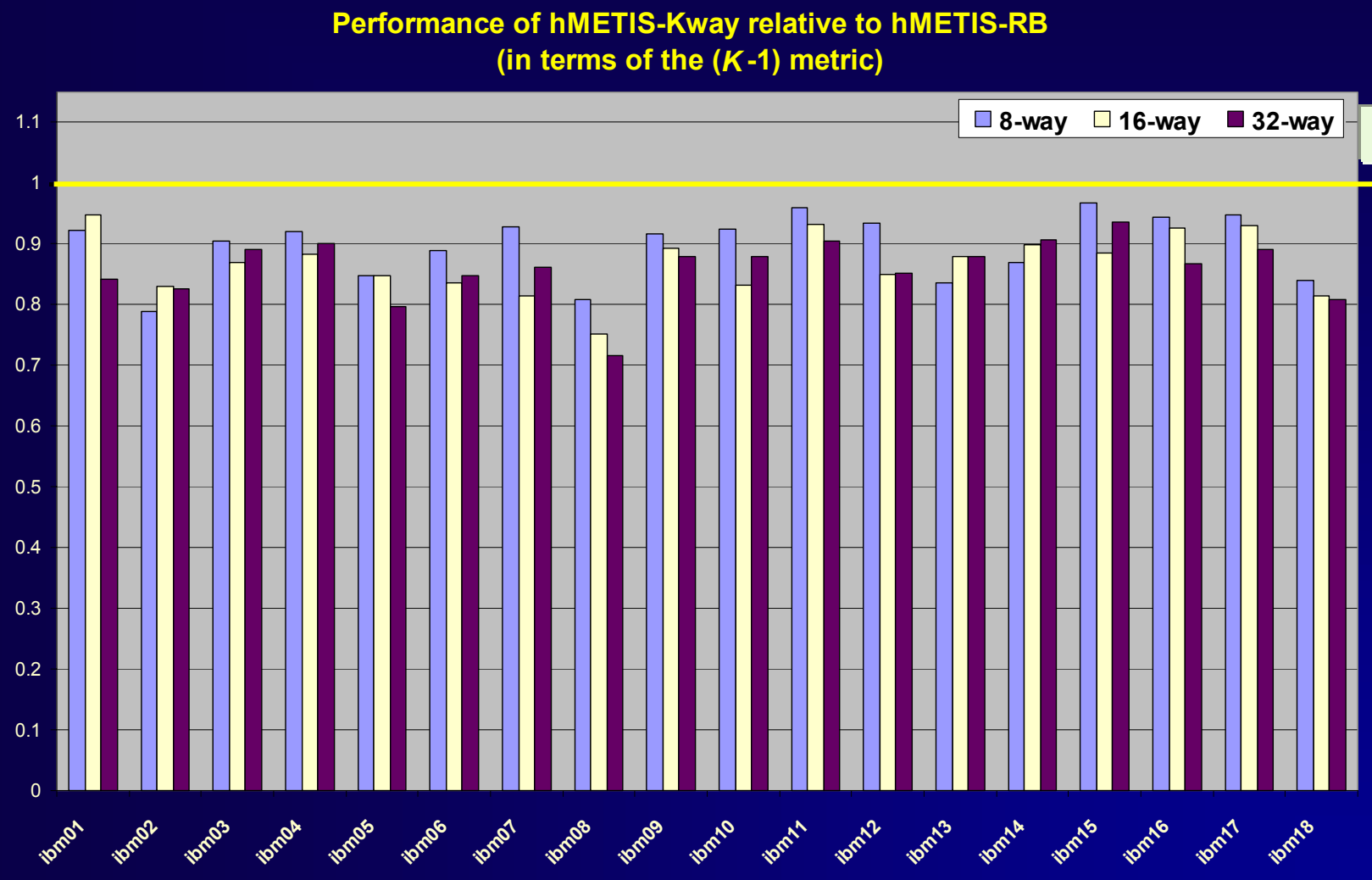
| | *Number of Vertices* | *Number of Hyperedges* |
|---|---|---|
| ***Ibm01*** | 12,506 | 14,111 |
| ***Ibm02*** | 19,342 | 19,584 |
| ***Ibm03*** | 22,853 | 27,401 |
| ***Ibm04*** | 27,220 | 31,970 |
| ***Ibm05*** | 28,146 | 28,446 |
| ***Ibm06*** | 32,332 | 34,826 |
| ***Ibm07*** | 45,639 | 48,117 |
| ***Ibm08*** | 51,023 | 50,513 |
| ***Ibm09*** | 53,110 | 60,902 |
| ***Ibm10*** | 68,685 | 75,196 |
| ***Ibm11*** | 70,152 | 81,454 |
| ***Ibm12*** | 70,439 | 77,240 |
| ***Ibm13*** | 83,709 | 99,666 |
| ***Ibm14*** | 147,088 | 152,772 |
| ***Ibm15*** | 161,187 | 186,608 |
| ***Ibm16*** | 183,484 | 190,048 |
| ***Ibm17*** | 185,495 | 189,581 |
| ***Ibm18*** | 210,613 | 201,920 |

# Mincut Performance
## (best of 20 runs)



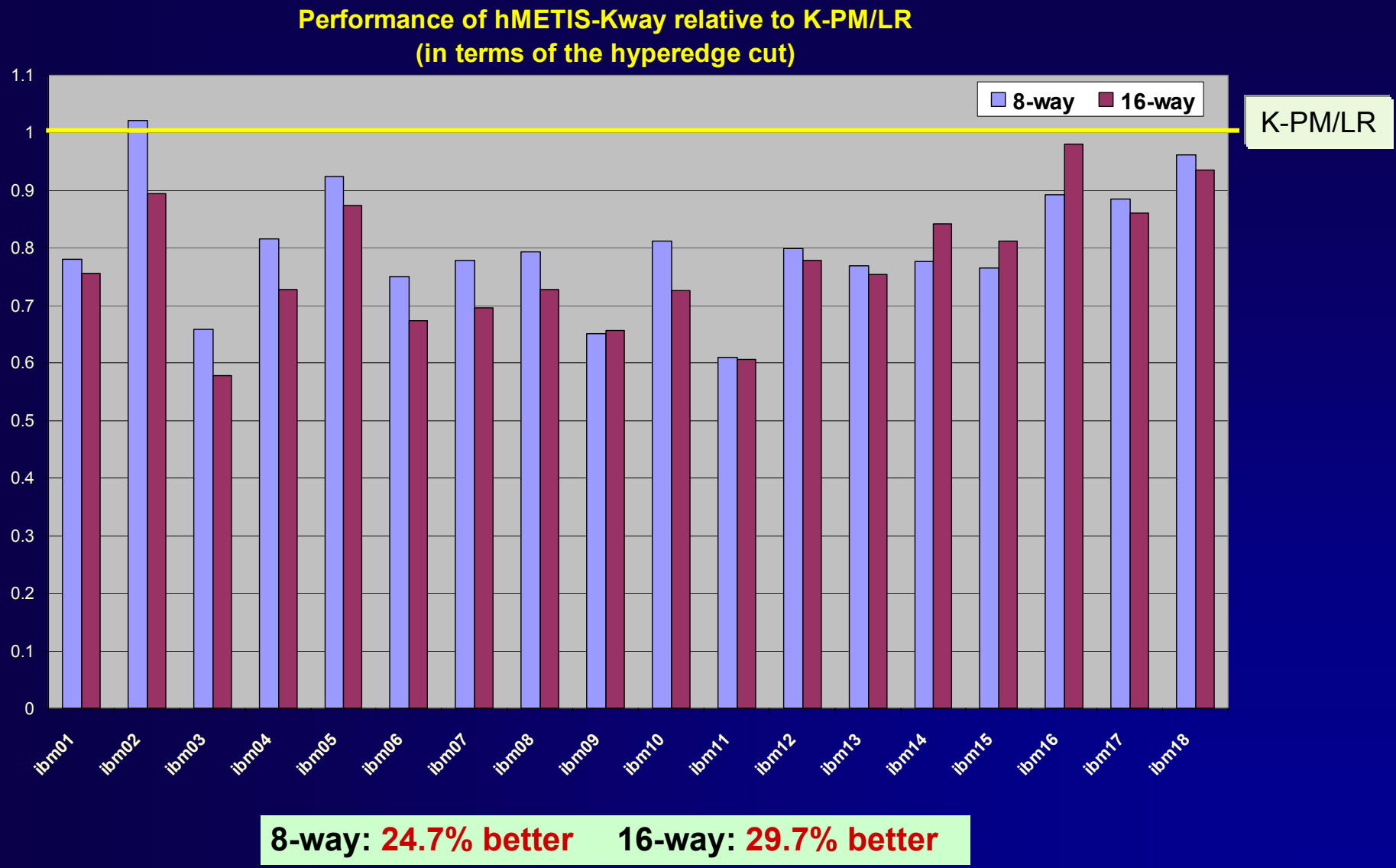

**8-way: 0.2% better** **16-way: 0.4% worse** **32-way: 0.6% better**

# (K-1)-Metric Performance
## (best of 20 runs)



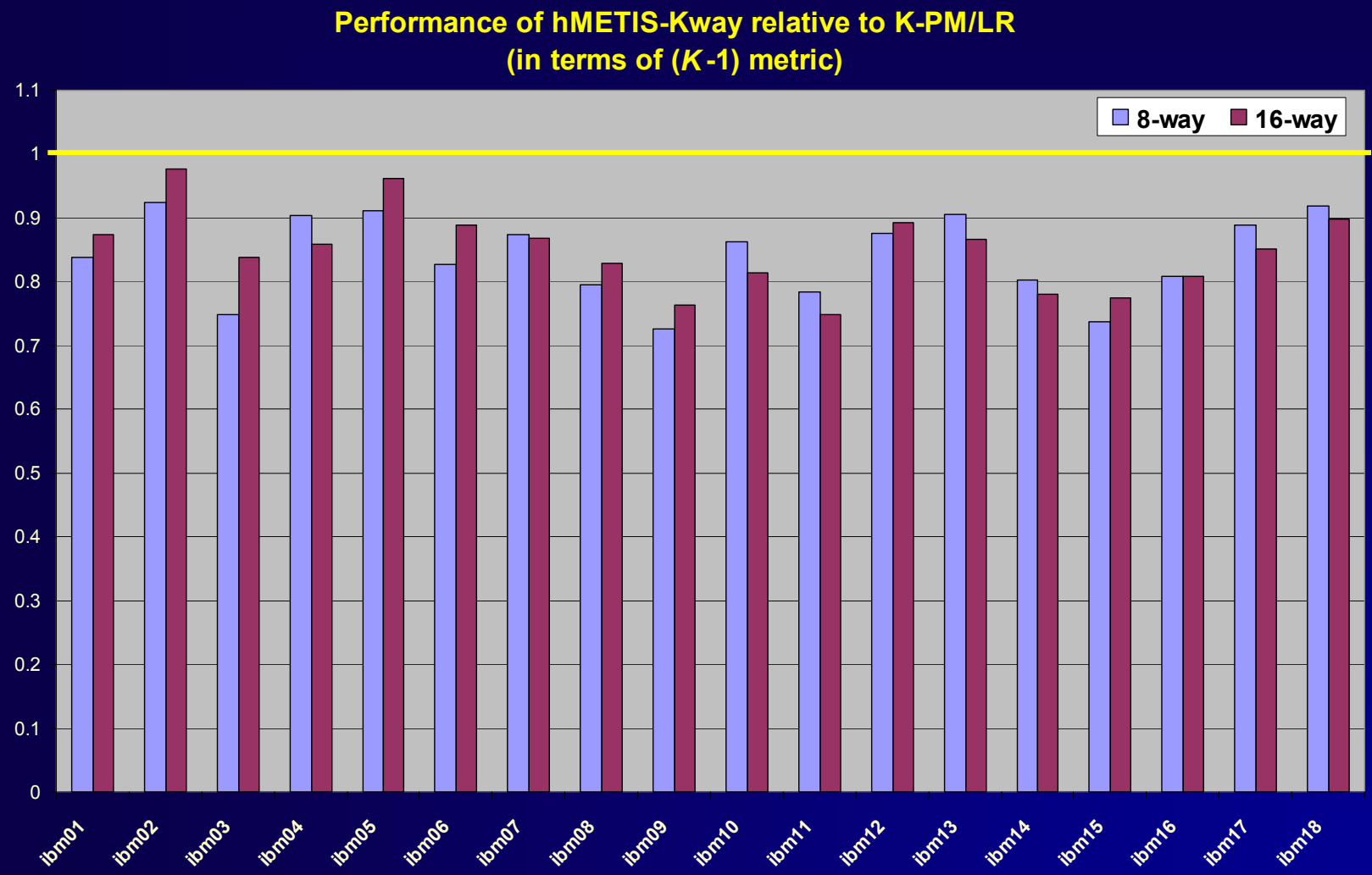

8-way: 11.5% better  16-way: 15.3% worse  32-way: 16.3% better

# Mincut Performance
## (best of 20 runs)




**8-way: 24.7% better  16-way: 29.7% better**

# (K-1)-Metric Performance
## (best of 20 runs)



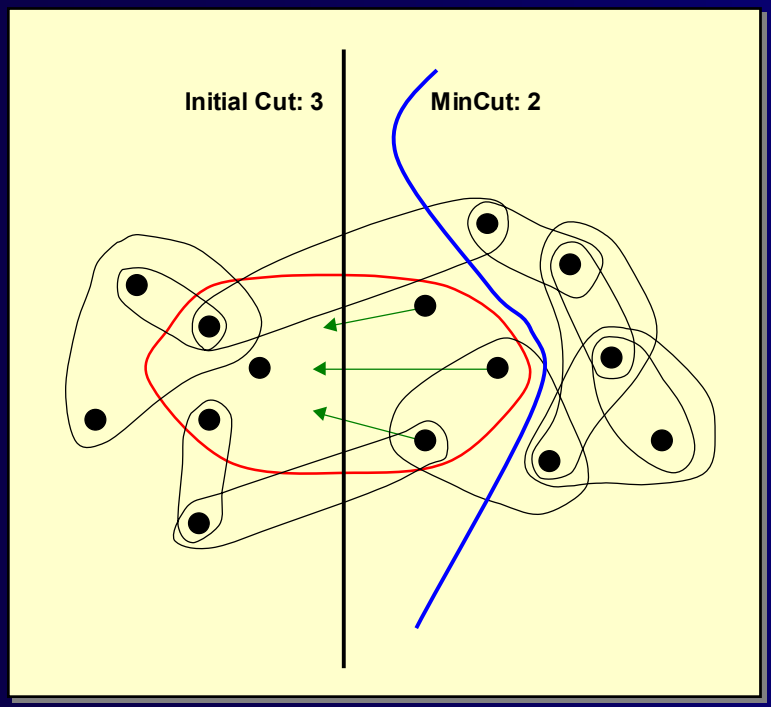

**8-way: 19.1% better** **16-way: 17.8% better**

# How did we get here?

It was not easy!

# Challenges in Hypergraph Partitioning

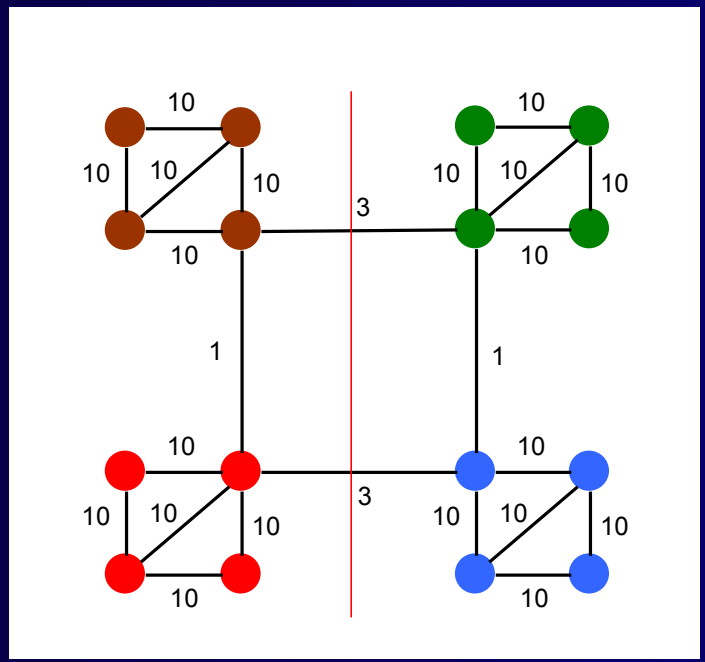
- In general, finding a mincut partitioning for a hypergraph is *harder* than finding a mincut partitioning for a graph.
  - Large hyperedges can easily confuse iterative refinement schemes.
  - Especially in the context of *k*-way refinement.



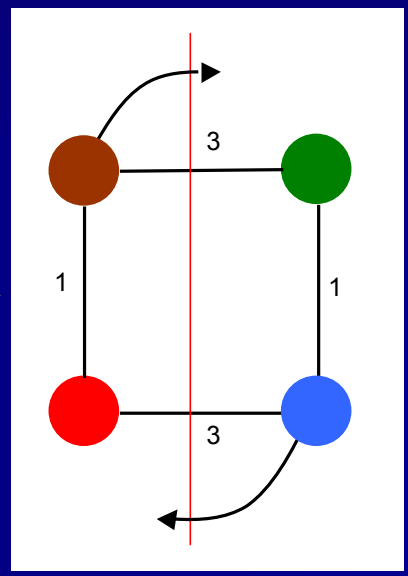

**To address this added difficulty a number of very sophisticated and computationally expensive algorithms have been developed**

# Why does the Multilevel Graph Partitioning Paradigm Work?

- The coarsening phase by hiding a large fraction of the edges, makes the partitioning problem easier.
- Performing refinement at successive finer graphs, enhances the effectiveness of refinement algorithms.
  - Multi-scale refinement



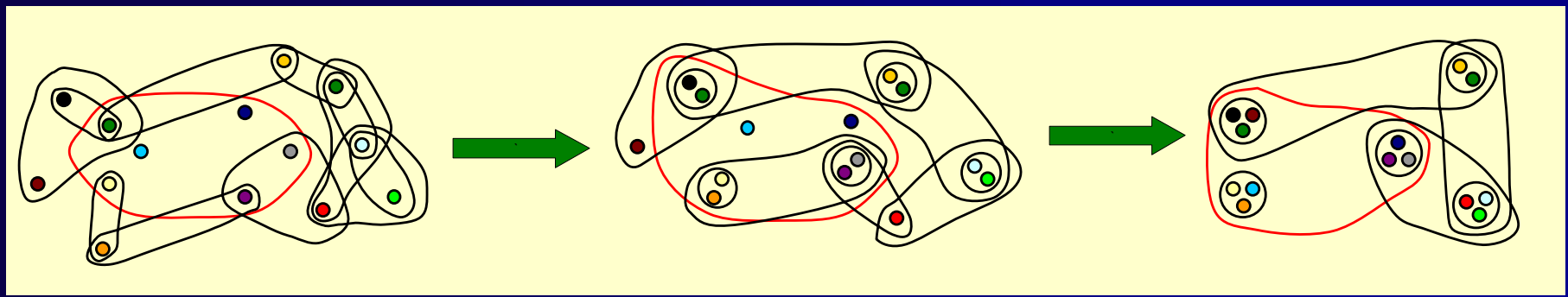




# Multilevel Paradigm & Hypergraphs
## Why does it work?

- The coarsening phase can help us in
  - Hiding a large number of hyperedges.
  - Multiscale refinement.
  - Coarsening reduces the size of the hyperedges

    Allowing iterative refinement schemes to operate effectively



# hMETIS
# Multilevel Hypergraph Partitioner

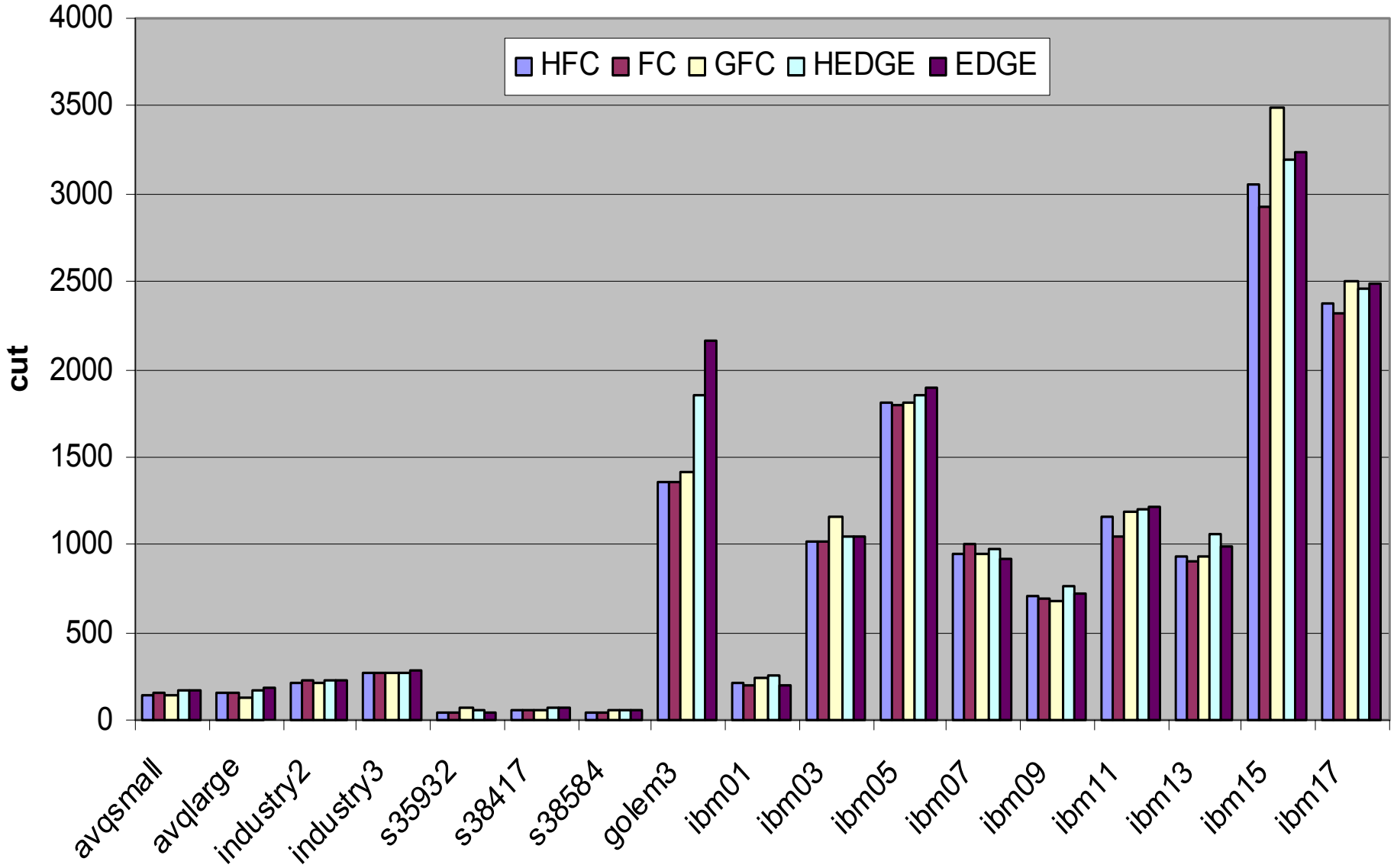
- Coarsening Phase
  - Vertices are grouped and collapsed together.
- Refinement Phase
  - An FM-based iterative refinement algorithm is used.
- V- and W-cycle Refinement Phases
  - A partitioning preserving coarsening is performed followed by multilevel refinement.
    - Allows clusters of vertices to move across partition boundaries.

# Coarsening Phase

Successive coarse hypergraphs are constructed by finding groups of vertices and collapsing together the vertices that belong to the same group.

- We developed a number of different *coarsening schemes*:
  - Heavy Edge Scheme:
    - Pairs of vertices that are part of a large number of hyperedges are grouped together.
  - Heavy Hyperedges Scheme:
    - Vertices belonging to the same hyperedge are grouped together.
  - First Choice Scheme:
    - Vertices that are present in multiple hyperedges are grouped together (*a group can contain an arbitrarily large number of vertices*).
- How to properly coarsen a hypergraph is still an open problem!

# Coarsening Scheme Sensitivity



# What next?

# Are we done?

# Multi-Constraint Hypergraph Partitioning

In many applications we will like the resulting partition to balance multiple weights associated with the vertices

- area, power, degree, flexibility, *etc.*

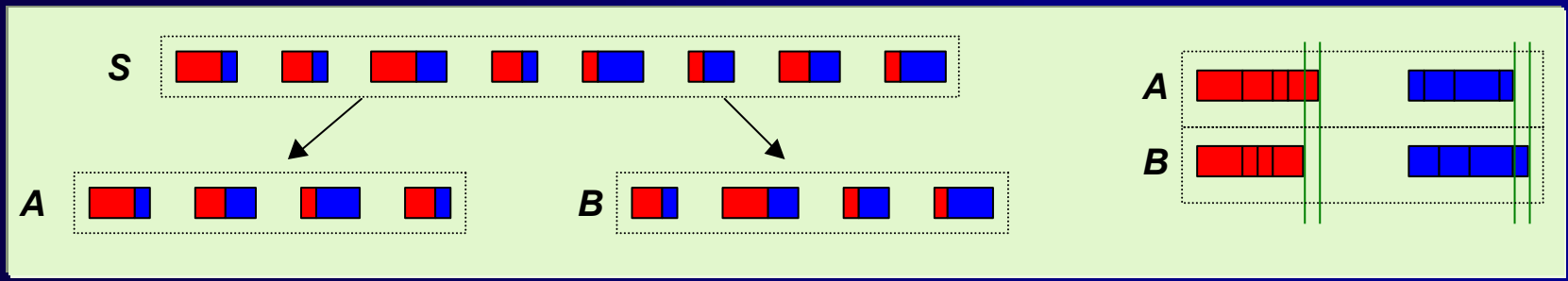
**Multi-constraint hypergraph partitioning**

Multiple balancing constraints can be incorporated by using a vector of vertex weights.

- In a hypergraph $G=(V, E)$, we can associate with each vertex $v \in V$, a weight vector $w^v$ of size $m$, such that $w^v_i$ stores the $i$th weight of vertex $v$.

# Challenges in Multi-Constraint Partitioning

- How can we ensure that the partitioning balances the different weights?
- Given a set of objects with multiple weights. How can we divide them into two groups, such that the weights in each group are balanced?
  - For single weight objects we can use bin-packing based algorithms.
    - The balance difference is bounded by the weight of the *heaviest* object.
  - What can we say about multiple weights?



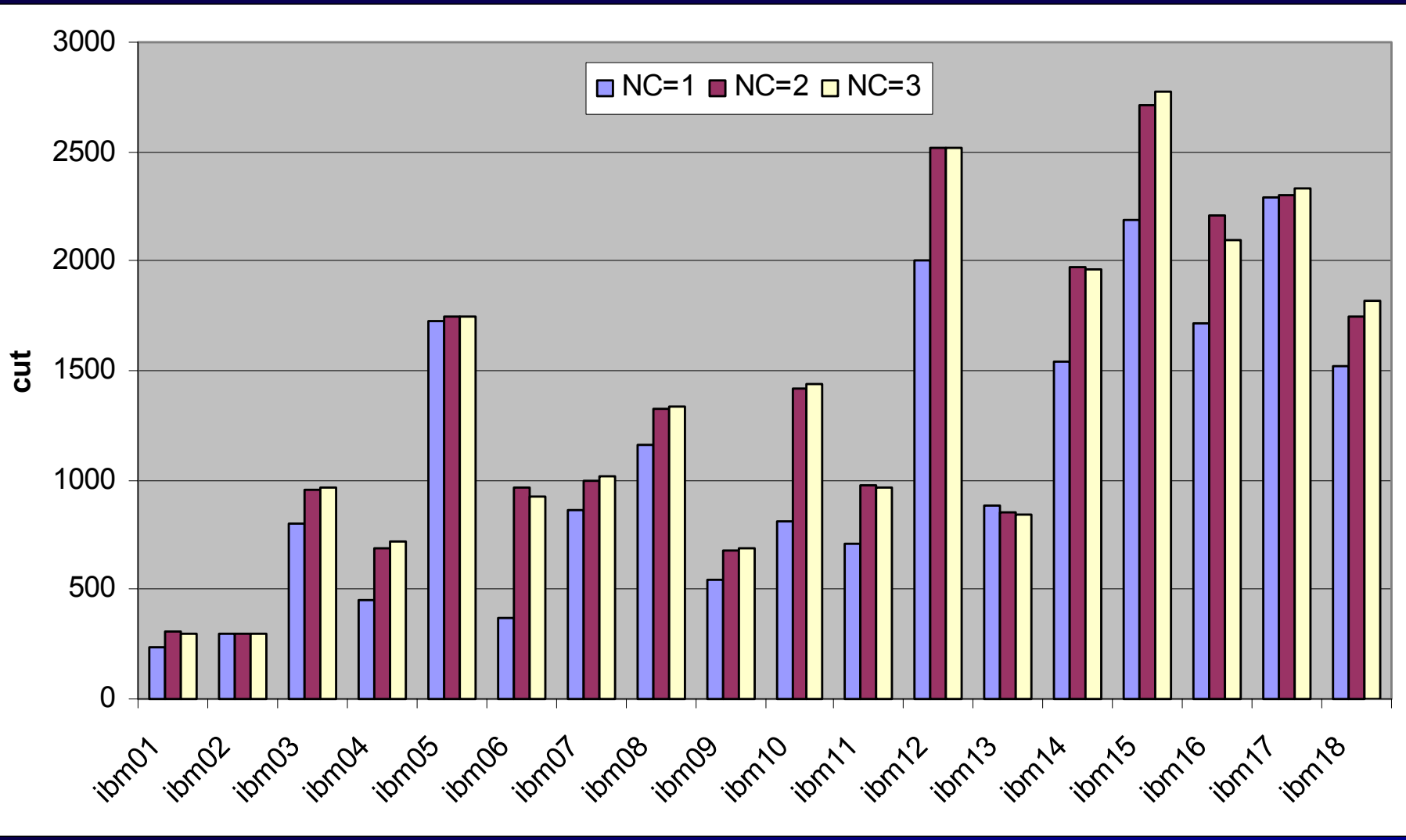

We showed that under certain assumptions on the distribution of the various weights there exist a simple algorithm that can ensure balance (SC 1998).

# Experimental Results

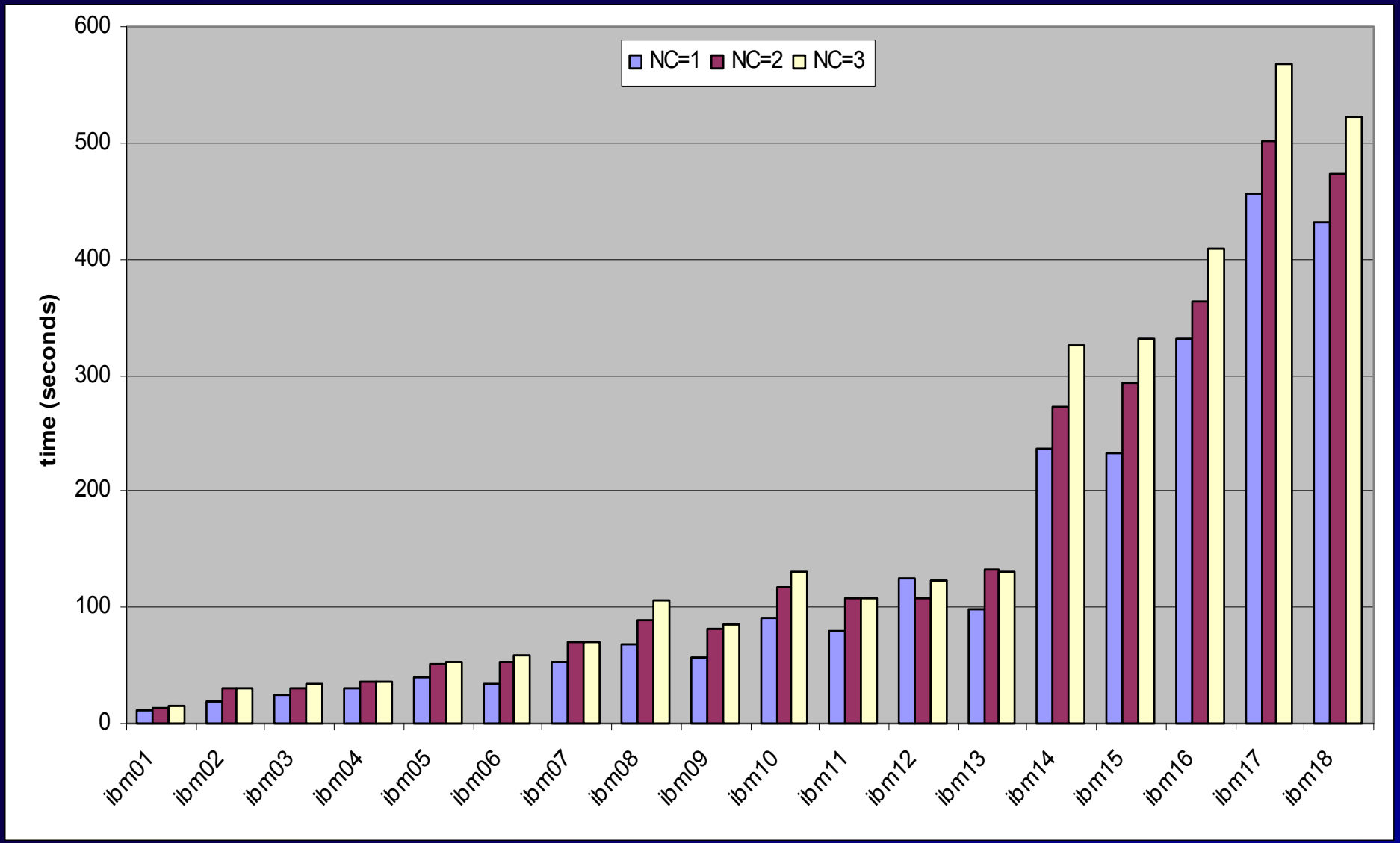
- We evaluated the performance of hMETIS-MC on the ISPD98 benchmark suite.
  - We generated two and three constraint problems
    - Area
    - Incident degree
    - Fan-out
- 45-55 bisections were computed.

| | *Number of Vertices* | *Number of Hyperedges* |
|---|---|---|
| ***Ibm01*** | 12,506 | 14,111 |
| ***Ibm02*** | 19,342 | 19,584 |
| ***Ibm03*** | 22,853 | 27,401 |
| ***Ibm04*** | 27,220 | 31,970 |
| ***Ibm05*** | 28,146 | 28,446 |
| ***Ibm06*** | 32,332 | 34,826 |
| ***Ibm07*** | 45,639 | 48,117 |
| ***Ibm08*** | 51,023 | 50,513 |
| ***Ibm09*** | 53,110 | 60,902 |
| ***Ibm10*** | 68,685 | 75,196 |
| ***Ibm11*** | 70,152 | 81,454 |
| ***Ibm12*** | 70,439 | 77,240 |
| ***Ibm13*** | 83,709 | 99,666 |
| ***Ibm14*** | 147,088 | 152,772 |
| ***Ibm15*** | 161,187 | 186,608 |
| ***Ibm16*** | 183,484 | 190,048 |
| ***Ibm17*** | 185,495 | 189,581 |
| ***Ibm18*** | 210,613 | 201,920 |

# Mincut Performance of hMETIS-MC
## (best of 10 runs)



# Runtime Performance of hMETIS-MC



# Conclusions…

- Multilevel paradigm is very powerful!
- It needs to be used within the context!
- It is widely applicable…
  - An algorithm design paradigm
    - Especially for optimization problems.
- It may not work all the time…
  - Coarsening schemes that preserve the *essense* of the problem and solution may be hard to find…
  - Or it may not lead to sufficient reduction of the problem size…