Improved Algorithms for Hypergraph Bipartitioning

A. E. Caldwell, A. B. Kahng and I. L. Markov
UCLA
http://vlsicad.cs.ucla.edu

Outline

- Hypergraph Bipartitioning
- Existing "Flat" Partitioning Heuristics
  - Greedy, KL, FM
- Partitioning with Fixed Vertices
- *New improvement:* FM-VRW
- Multi-level Partitioning
- *New improvement:* UCLA MLPart
- Leading-edge Results
- Conclusions

Hypergraph Bipartitioning

- Goal
  - split vertices into two sets of approx. equal size (*balance* tolerance: often *2%-10%*, can be < or >)
  - minimize a given objective (netcut etc)
- *Min-cut partitioning* is most common variant
  - minimizes #nets incident to vertices in both sets
  - can be performed very efficiently
- Relevant to *circuit partitioning* and *placement*
  - signal nets represented by hyperedges
  - *placement* done by *recursive bipartitioning*

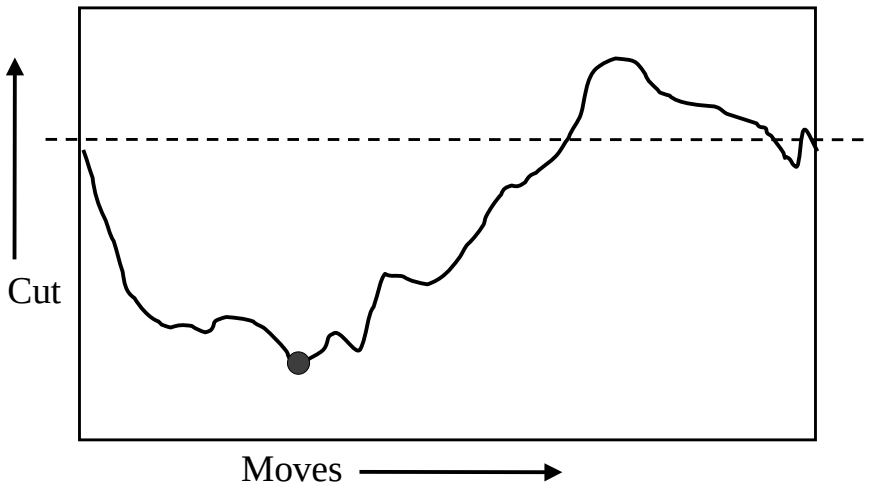
Existing Partitioning Heuristics

- *Most heuristics start with a random solution*
- Greedy moves: move one vertex to the other set
  - only if this *improves cut*
  - only if no worsening of balance ("legal" move)
  - continue until no moves left
  - can move the same vertex twice
- Greedy: *very fast* but produces *very poor solns*
  - final cut can be 10x greater than best known solns
- Pass-based heuristics (next slide)
- Multi-level heuristics (also in this talk)
  - terminology: "flat" == *not multi-level*

(Flat) Pass-based Heuristics

- ***Start pass*** with all vertices free to move (unlocked)
- ***Choose the best move*** every time
  - (greedy does not)
  - update priorities after each move
- ***Once moved, each vertex is locked***
  - can't be moved until end of pass
- ***End of pass: no moves left***
  - all vertices unlocked
  - *restore best-seen solution* (e.g., by undoing moves)
- Continue passes until solution fails to improve

Cut During One Pass



(Flat) Pass-based Heuristics (cont'd)

- ***Pass-based heuristics can do hill-climbing***
  - perform moves that *increase cut*
  - *escape* some *local minima*
  - have much better performance than greedy
- Kernighan-Lin (KL, first pass-based heuristic)
  - proposed in early 70s (similar to their TSP heur.)
  - move: *swap* a pair of vertices, one from each set
- Fiduccia-Mattheyses (FM, faster than KL)
  - similar to Kernighan-Lin
  - move: single-vertex set-to-set
- FM variants LIFO FM (`96), CLIP FM (`94)

Partitioning with Fixed Vertices

- Most works partition "free" hypergraphs
  - in applications, vertices may be fixed in set 1 or set 2
- FM *passively* accommodates such fixed constraints
  - by *not moving* fixed vertices
- Caldwell, Kahng and Markov in DAC99 show:
  - performance of LIFM ***worsens*** when a very small number (5%) of vertices are fixed
  - faster partitioning possible with fixed vertices
  - open question: heuristics that exploit fixed vertices
- **Can one <u>actively</u> exploit fixed vertices ?**

New Improvement (VRW)

- Initial solution
  - typically chosen to be balanced ("legal")
    - all moves preserve balance
  - we choose "very illegal" initial solution (**V**ILE)
    - all vertices are initially in one partition
    - vertex moves must not worsen the balance
    - balance is easily restored in 1-2 FM passes
- **R**andomization
  - classic FM is *deterministic* (except for init. sol.)
  - we *randomly reorder vertices* before each pass
  - $\Rightarrow$ FM can use VILE and better escape local min.

New Improvement (VRW)

- Recall
  - classic FM and its variants ***prioritize*** moves/vertices
  - leaves ties among equal-priority moves
  - when vertex moves, its neighbors are reprioritized
  - tie-breaking seriously affects performance of FM
    - Krishnamurthy used it in look-ahead partitioner (slow)
    - Hagen/Kahng: LIFO FM is better than FIFO FM
  - Unused degrees of freedom exist
- We break ties toward neighbors of fixed vertices
  - at the start of each pass,
    move fixed vertices back and forth ("**w**iggle")
  - ***this reprioritizes the neighbors of fixed vertices***

Experimental Methodology

- Benchmarks with fixed vertices
  - ISPD `99 suite
  - described in Alpert/Caldwell/Kahng/Markov paper
  - produced from ISPD `98 circuits from IBM
  - http://vlsicad.cs.ucla.edu/benchmarks
- Comparison methodology
  - run-time / solution quality trade-off
  - DAC `99 Caldwell/Kahng/Markov
  - records solution quality achieved in given time
- Compared average results of *K* starts of
  - LIFO FM (Hagen/Kahng `94)
  - CLIP FM (Dutt/Deng `95)
  - FM VRW (this paper)

Sample Results

Algo	1 start	2 starts	4 starts	8 starts
IBM01D_H 2%, 6139 movable, 2155 fixed, 7330 nets				
FM	706.6(1.7)	648.8(3.4)	600.5(6.7)	566.2(13.4)
CLIP	654.7(5.4)	593.6(10.8)	**537.2(21.7)**	**492.0(43.4)**
VRW	**624.2(1.7)**	**588.6(3.4)**	**570.6(6.7)**	**555.5(13.4)**
IBM01D_H 10%				
FM	676.9(1.6)	612.3(3.2)	562.8(6.5)	504.2(12.9)
CLIP	482.4(6.0)	412.0(12.1)	**368.3(24.2)**	**348.3(48.4)**
VRW	**419.9(1.4)**	**403.8(2.9)**	**395.9(5.8)**	**390.4(11.6)**
IBM01D_V 2%				
FM	664.3(2.1)	600.0(4.2)	557.8(8.3)	521.9(16.6)
CLIP	587.7(5.5)	536.8(11.1)	499.0(22.2)	**471.2(44.4)**
VRW	**535.6(1.8)**	**508.3(3.7)**	**489.4(7.4)**	**480.1(14.8)**
IBM01D_V 10%				
FM	623.9(1.8)	582.2(3.51)	551.3(7.0)	528.9(14.1)
CLIP	559.4(5.6)	521.3(11.2)	488.3(22.4)	465.2(44.8)
VRW	**519.8(1.4)**	**494.6(2.8)**	**474.6(5.5)**	**461.9(11.1)**

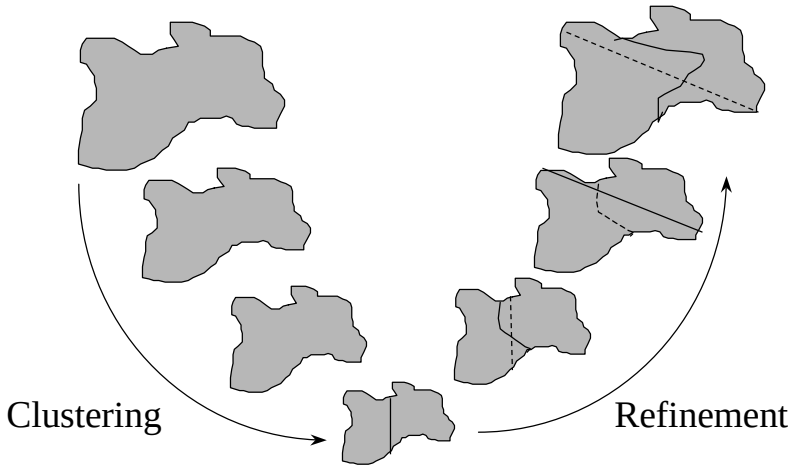
Observations

- Winning combinations are in bold in table
  - all FM VRW combinations are winning
  - FM VRW is superior to LIFO FM and CLIP FM
- Extending LIFO FM to FM VRW
  - took one day
- With very many terminals (see proceedings)
  - FM VRW is faster than hMetis, similar quality
- With few terminals (see proceedings)
  - FM VRW is better on some benchmarks
  - much worse on others (see paper)
  - can perform both LIFO FM and FM VRW starts

Multi-level Partitioning

- ***Used in the most efficient partitioners***
  - Karypis et al., DAC `97
  - Alpert et al., DAC `97
- Basic idea (two levels)
  - group original vertices together (in clusters)
  - induce clustered hypergraph
  - partition clustered hypergraph
  - assign original vertices to set1/set2 accordingly
  - refine by additional FM passes
- Also see a `98 paper by Wichlund and Aas

Multilevel Partitioning



Multi-level Partitioning (cont'd)

- Multi-level FM (ML-FM)
  - does not directly use an initial partitioning
  - much better than "flat" FM alone
  - produces very good solutions in near-linear time
  - critical to performance of top-down placers
- Benchmarks
  - old MCNC benchmarks do not reflect modern circuitry
  - use common benchmarks from IBM (Alpert, ISPD `98)

Multi-level Partitioning (cont'd)

- **Previous #1 partitioner** (downloadable)
  - **hMetis** from U. of Minnesota (Karypis et al., DAC `97)
  - combines a large number of heuristics
    - difficult to describe
    - performance seriously improved since DAC `97
    - some heuristics from DAC `97 paper are not used
- V-cycling ("iterative ML") - Karypis et al., DAC '97
  - a method of using an initial solution
  - ***avoids clustering vertices that are in difft. sets***
  - ***can now run ML to improve results of previous runs***
  - used by hMetis with non-trivial stopping criterion

Our MLFM variant (UCLA MLPart)

- Is simpler to implement than hMetis
- Was developed for *fast* partitioning
  - goal: produce best possible soln. in N sec
- Better quality/runtime trade-off than hMetis
  - produces better solutions given same CPU time
  - each start is much faster: better time granularity
  - scales better with growth of hypergraphs
- Available for download

http://vlsicad.cs.ucla.edu/GSRC/bookshelf/Slots/Partitioning

Main Improvements

- Top-level partitioning with small tolerance
  - first partition top level with lax tolerance
  - use the result as initial solution for another FM run
  - decrease tolerance to what it should be, run FM
- A new clustering algorithm PinEC
  - as simple and as fast as EC of Karypis et al.
  - different pairs of vertices are contracted
  - produces clusterings that work better in MLFM

Clustering Algorithms (details)

- EC (Edge Clustering) of Karypis et al.
  - *for all pairs of adjacent vertices/clusters*, add up weighs of all hyperedges incident to both
  - every hyperedge contributes *weight/degree*
  - merge the pairs in the order of total weights
- A new clustering algorithm PinEC
  - as simple and as fast as EC of Karypis et al.
  - hyperedges of *degree==2* contribute *2*weight*, all others contribute *weight*
    - this corresponds to *#pins removed by merging*
  - produced clusterings work better in MLFM
    - but do not “look better” themselves

Main Improvements (cont'd)

- Judging from hMetis logs, our flat FM is stronger
- Main clustering and "V-cycling" use difft algos
- Simplifications compared to hMetis
  - no "hyperedge removal" (cf. hMetis) - "dual FM"
  - only one "V-cycle" (hMetis uses many)
  - independent starts

    (hMetis uses an aggressive pruning heuristic)

Sample Results

Test IBM #	Algo	10% configurations			
		1	2	3	4
01	HM	255(5.3)	252(7.3)	247(11)	237(18)
	OUR	**238(4.6)**	**233(6.9)**	**223(12)**	**220(21)**
02	HM	279(11)	279(14)	279(19)	278(31)
	OUR	**291(8.2)**	**291(13)**	**272(21)**	**268(37)**
03	HM	779(15)	776(18)	768(27)	759(43)
	OUR	**802(11)**	**783(16)**	**732(30)**	**705(52)**
04	HM	506(19)	510(20)	492(29)	478(51)
	OUR	**535(12)**	**512(19)**	**482(32)**	**468(56)**
05	HM	1759(25)	1740(28)	1731(42)	1725(67)
	OUR	**1773(17)**	**1726(26)**	**1728(43)**	**1716(77)**
06	HM	**402(24)**	**383(26)**	**374(35)**	**370(55)**
	OUR	**465(14)**	433(22)	394(34)	375(61)
07	HM	809(40)	806(46)	796(63)	764(102)
	OUR	**790(20)**	**786(32)**	**760(52)**	**747(93)**
08	HM	**1166(43)**	**1162(51)**	**1161(75)**	**1159(123)**
	OUR	**1348(25)**	1330(40)	1195(66)	1168(113)
09	HM	664(33)	**540(42)**	**528(59)**	**525(96)**
	OUR	**572(21)**	**5609(34)**	556(55)	**527(90)**

10	HM	**842(57)**	**830(72)**	**798(102)**	**779(169)**
	OUR	**1196(35)**	1132(55)	1023(91)	938(161)
11	HM	828(59)	**744(71)**	**717(103)**	**710(55)**
	OUR	**788(30)**	**779(48)**	745(77)	728(124)
12	HM	2326(83)	**2165(108)**	**2135(153)**	**2047(238)**
	OUR	**2349(38)**	**2304(57)**	**2261(102)**	2206(176)
13	HM	**1045(97)**	**999(104)**	**940(129)**	**802(197)**
	OUR	**1095(38)**	**1058(61)**	**1027(99)**	963(176)
14	HM	1894(212)	1776(250)	1682(351)	1599(565)
	OUR	**1759(67)**	**1727(104)**	**1638(167)**	**1590(300)**
15	HM	2073(264)	2032(301)	1910(402)	1866(564)
	OUR	**2244(83)**	**2029(133)**	**2035(221)**	**1975(385)**
16	HM	1942(232)	1754(311)	1829(327)	1721(709)
	OUR	**1976(102)**	**1915(155)**	**1752(257)**	**1714(460)**
17	HM	2417(408)	2372(467)	2337(635)	2311(987)
	OUR	**2316(105)**	**2294(170)**	**2264(290)**	**2239(385)**
18	HM	1632(306)	1617(357)	1561(601)	1536(836)
	OUR	**1976(104)**	**1955(160)**	**1666(365)**	**1595(612)**

Conclusions

- Fixed vertices can be actively exploited
  - new FM variant FM-VRW
    can be used together or instead of FM
- Fixed-vertex context still poorly explored
- New variant of multi-level FM: UCLA MLPart
  - each start is much faster than hMetis
  - MLPart's produces better solutions than hMetis, given same time
  - easier to implement than hMetis
- Harder techniques not necessary, but may help
  - hMetis may be improved using our techniques