Ex	FMx50	Prev. Best	LSMC		
			Min(Avg)	#descents	CPU(sec)
p1	54	59	53(54.1)	222.2	107
p2	194	154	146(160.5)	200.2	535
st	40	38	34(37.7)	197.0	89
bio	98	88	83(89.4)	120.7	1645
t2	111	91	93(100.6)	208.3	101
t3	69	58	58(61.3)	204.3	61
t4	75	58	51(61.3)	218.5	239
t5	110	82	84(99.2)	193.3	430
t6	68	81	63(66.8)	171.6	234
ind2	649	254	248(313.8)	140.2	1641

Table 2. Comparison of bisection solution quality, using unit module areas.

each run consists of 1000 passes of FM. LSMC outperforms the previous best reported results in the literature by 6.4% for unit area and averages 21.4% improvement over FMx50. With actual module areas, LSMC is 1% better than previous results, and averages 36% improvement over FMx50.

Ex	FMx50	Prev. Best	LSMC		
			Min(Avg)	#descents	CPU(sec)
p1	47	47	47(48.1)	224.6	109
p2	212	146	136(145.2)	192.7	430
st	41		36(39.1)	195.7	173
bio	98		86(96.4)	66.8	785
t2	104	42	42(57.5)	347.8	212
t3	61	50	39(54.6)	235.5	178
t4	51	12	13(19.1)	292.8	168
t5	96	24	28(31.8)	229.6	418
t6	62	63	61(64.9)	204.9	155
ind2	508		222(300.4)	135.5	1691

Table 3. Comparison of bisection solution quality, using actual module areas.

4. CONCLUSIONS AND FUTURE WORK

We have presented an experimental study of several local perturbation strategies (kick moves) for the Large-Step Markov Chain (LSMC) heuristic applied to VLSI netlist bisection. Our experimental results show that the LSMC heuristic consistently outperforms the standard random multistart strategy, given the same computational resources. The quality of our results compare favorably to existing published results.

We are currently pursuing the following extensions to our work:

- [9] showed that for the TSP, the efficacy of the kick moves depends on the underlying greedy descent engine. This may be the case for netlist partitioning as well. We are currently studying whether implementation details such as lookahead [11] or the gain bucket tie-breaking scheme used [6] will significantly affect the choice of kick move.
- Problem reduction techniques can significantly improve the performance of iterative improvement partitioning algorithms [7]. In such hierarchical approaches, it is possible to use LSMC in place of the standard FM algorithm to improve solution quality, at the cost of additional runtime. For example, in a 2-phase FM approach [3], the calls to the FM algorithm can be replaced with calls to LSMC. We are currently experimenting with this approach; preliminary experiments show that this is quite promising.
- The LSMC heuristic can be straightforwardly generalized to k-way partitioning (each of the kick moves described in Section 2 has an obvious k-way generalization). In preliminary experiments, k-way LSMC with clustering kick move is promising, yielding better results than equivalent computational efforts with a multistart k-way FM.
- Finally, [9] has shown that more elaborate variants of LSMC, e.g., using non-zero temperature schedules and population-based optimization can be effective on the TSP. Similar improvements may be found for netlist partitioning.

.

ACKNOWLEDGMENTS

This work was supported in part by NSF CDA-9303148 and NSF Young Investigator Award MIP-9257982.

REFERENCES

[1] C.J. Alpert and A.B. Kahng, "Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning", in *Proc. ACM/IEEE Design Automation Conf.*, Dallas, June 1993, pp. 743–748.

[2] E. B. Baum, "Iterated Descent: A Better Algorithm for Local Search in Combinatorial Optimization Problems", *Unpublished Manuscript*, 1986.

[3] T. Bui, C. Heighham, C. Jones and T. Leighton, "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms," in *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 775–778.

[4] S. Dutt and W. Deng, "A Probability-Based Approach to VLSI Circuit Partitioning", to appear in *Proc. ACM/IEEE Design Automation Conf.*, June 1996.

[5] C.M Fiduccia and R.M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," in *Proc. ACM/IEEE Design Automation Conf.*, 1982, pp. 175–181.

[6] L. Hagen, D. J.-H. Huang and A.B. Kahng, " On Implementation Choices for Iterative Improvement Partitioning Algorithms", *Proc. European Design Automation Conf.*, 1995, pp. 144–149.

[7] L. Hagen and A.B. Kahng, "Combining Problem Reduction and Adaptive Multi-Start: a New Technique for Superior Iterative Partitioning", to appear in *IEEE Trans. on Computer-Aided Design.*

[8] M.R. Hartoog, "Analysis of Placement Procedures for VLSI Standard Cell Layout", *Proc. ACM Design Automation Conference*, 1986, pp. 314-319.

[9] I. Hong, A. B. Kahng and B.-R. Moon, "Improved Large-Step Markov Chain Variants for the Symmetric TSP", *UCLA Computer Science Dept. Tech. Report UCLA-CSD-950035*, 1995.

[10] D.S. Johnson, "Local Optimization and the Traveling Salesman Problem", *Proc. 17th Intl. Colloquium on Automata, Languages and Programming*, 1990, pp.446-460.

[11] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks," *IEEE Trans. on Computers* 33(5) (1984), pp. 438–446.

[12] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Teubner, 1990.

[13] O.C. Martin, S.W. Otto and E.W. Felten, "Large-step Markov chains for the traveling salesman problem", *Complex Systems*, 5(3) (1991), pp. 299-326.

[14] O.C. Martin and S.W. Otto, "Combining Simulated Annealing with Local Search Heuristics", to appear in *Annals of Operations Research.*

[15] T. Shibuya, I. Nitta and K. Kawamura, "SMINCUT: VLSI Placement Tool Using Min-Cut", *Fujitsu Sci. Tech. Journal*, December 1995.

[16] Y.C. Wei and C.K. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning," in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1989, pp. 298–301.

cut; after performing greedy descent in the reduced netlist, N_u is restored. The combination of these operations constitutes the kick move. In our implementation $MoveSize$, the percentage of uncut nets removed, is varied between 25-75%.

Net Removal

Input: Partition N_{orig} of the netlist, a fraction $MoveSize$, $0 < F < 1$,
Output: A modified partitioning solution

1. Randomly select a subset n_u of the uncut nets, with $|n_u| = MoveSize \times NumNets$, where $NumNets$ is the number of nets in the original netlist.
2. Reduce the netlist by removing all nets in n_u to obtain $N_{reduced}$.
3. Perform a FM descent on $N_{reduced}$.
4. Restore the original netlist by adding the nets n_u back into $N_{reduced}$.

Figure 3. Net removal kick move template.

3. EXPERIMENTAL RESULTS

We compared LSMC performance using the four kick moves described above on benchmarks Struct, Primary2, Test2, Test3 and Biomed maintained by ACM SIGDA. Each entry in Table 1 is based on 50 runs of LSMC, where the termination condition for each run was to stop when a local minimum has been reached and more than 1000 FM passes have been executed.[3] For the net removal kick move, FM passes based on the reduced netlist and on the full netlist were weighted equally.[4]

The third column of Table 1 shows the kick move size. A "randomized" $MoveSize$ means that every time the kick move was applied, a random value of $MoveSize$ was chosen from a uniform distribution (this value was varied between 0.05 and 0.25 for the random and clustering kick moves, and between 0.25 and 0.75 for the net removal kick move).

We make the following observations: (1) The random, net removal, and the clustering kick moves consistently outperform the standard random multistart strategy. Thus, the idea of making local perturbations (as opposed to complete randomization) seems to be a promising idea. (2) Overall, the clustering kick move yielded the best performance, followed by the net removal kick move. (3) In general, the larger values of $MoveSize$ resulted in better performance; the randomized $MoveSize$ performed well overall.

We also compare LSMC with the best previous partitioning results in the literature (e.g., [1, 4, 7]). Table 2 and Table 3 show the comparison using unit are and actual area respectively. Each entry is based on 50 runs of LSMC, where

[3] For the Biomed benchmark in Table 1, our results are based on 500 passes of FM. Recall that an FM pass is linear-time; it moves and locks each node exactly once, then adopts the prefix of this move sequence with maximum gain in cutsize. We use the number of FM passes to bound each run, since this should result in as fair a comparison as possible, and allows experiments to be run on several different CPUs of varying speeds. As a baseline, recent efficient implementations of FM engines by the authors of [4] can perform a complete FM descent on our largest test case (Biomed) in only a few seconds on a Sun Sparc-5 machine.

[4] Note that FM passes on the reduced netlist are significantly faster (because the netlist is smaller), causing some disparity in comparisons between the net removal kick move and the other kick moves. However, it is not clear how significant this disparity is; our current efforts are aimed at developing a more accurate method for equalizing the CPU resources used in our experiments.

Ex	Kick	Size	Min(avg-Max)(σ)	#Descents
st	m		45.1(41-51)(2.1)	117.7(2.1)
st	r	0.05	48.4(41-64)(5.7)	330.7(11.8)
st	r	0.125	42.2(39-46)(1.8)	189.0(5.4)
st	r	0.25	44.6(42-49)(1.6)	117.4(2.6)
st	r	rand	41.5(36-45)(2.1)	144.6(3.7)
st	n	0.25	53.5(42-69)(7.8)	200.4(60.6)
st	n	0.50	55.2(43-79)(9.0)	176.0(54.6)
st	n	0.75	60.3(44-98)(12.0)	186.7(43.5)
st	n	rand	45.4(42-65)(3.8)	163.7(33.2)
st	c	0.05	39.8(34-46)(2.4)	277.6(10.3)
st	c	0.125	38.5(35-46)(2.2)	219.6(5.1)
st	c	0.25	38.7(36-43)(1.5)	185.9(5.5)
st	c	rand	39.1(36-43)(1.9)	195.7(5.1)
p2	m		257.5(212-293)(15.9)	69.3(2.5)
p2	r	0.05	232.6(148-309)(28.6)	305.7(14.4)
p2	r	0.125	223.3(144-308)(32.4)	146.7(12.6)
p2	r	0.25	259.0(227-281)(13.5)	69.8(2.7)
p2	r	rand	231.8(188-255)(13.6)	96.8(5.9)
p2	n	0.25	204.8(136-281)(34.0)	131.3(17.8)
p2	n	0.50	203.3(144-235)(21.3)	104.7(6.3)
p2	n	0.75	185.9(138-230)(20.0)	98.6(8.2)
p2	n	rand	186.1(136-226)(25.5)	103.2(7.7)
p2	c	0.05	159.3(136-222)(24.8)	227.2(6.8)
p2	c	0.125	149.1(136-187)(14.3)	197.5(7.0)
p2	c	0.25	146.4(136-178)(10.4)	179.7(6.1)
p2	c	rand	146.5(136-176)(10.8)	188.4(5.7)
t2	m		118.0(104-136)(7.9)	132.4(3.5)
t2	r	0.05	98.6(42-131)(11.8)	217.5(8.6)
t2	r	0.125	42.0(42-43)(0.1)	147.6(4.2)
t2	r	0.25	42.0(42-42)(0.0)	141.9(3.4)
t2	r	rand	42.0(42-42)(0.0)	152.6(3.3)
t2	n	0.25	131.4(109-169)(13.5)	180.4(26.1)
t2	n	0.50	108.2(93-131)(9.5)	137.4(17.6)
t2	n	0.75	100.5(48-126)(11.2)	110.5(6.5)
t2	n	rand	102.6(64-127)(10.3)	126.0(5.6)
t2	c	0.05	84.4(42-107)(13.3)	285.7(22.9)
t2	c	0.125	65.5(42-98)(19.2)	318.2(79.8)
t2	c	0.25	49.6(42-92)(12.7)	375.3(71.6)
t2	c	rand	57.6(42-96)(17.2)	347.8(80.5)
t3	m		71.8(61-82)(4.7)	106.5(2.5)
t3	r	0.05	71.5(59-104)(11.7)	305.0(10.0)
t3	r	0.125	64.3(60-92)(7.7)	159.1(5.9)
t3	r	0.25	63.8(43-71)(4.4)	132.0(6.7)
t3	r	rand	63.1(58-68)(2.3)	134.7(4.4)
t3	n	0.25	82.9(61-121)(13.7)	174.6(27.4)
t3	n	0.50	69.5(55-99)(9.2)	139.0(26.9)
t3	n	0.75	64.2(55-80)(6.0)	115.0(22.8)
t3	n	rand	62.8(55-85)(6.4)	122.5(6.2)
t3	c	0.05	56.1(55-75)(3.3)	306.6(11.0)
t3	c	0.125	55.6(43-67)(3.5)	266.0(10.3)
t3	c	0.25	52.3(39-59)(7.0)	208.3(19.3)
t3	c	rand	54.6(39-61)(4.6)	235.5(10.1)
bio	m		118.3(98-152)(12.2)	27.3(1.5)
bio	r	0.05	100.3(86-165)(15.4)	96.4(5.1)
bio	r	0.125	99.4(88-119)(7.4)	38.7(2.6)
bio	r	0.25	124.9(94-160)(14.1)	27.4(1.8)
bio	r	rand	104.9(87-139)(9.9)	31.4(2.0)
bio	n	0.25	107.4(86-149)(13.5)	91.9(43.4)
bio	n	0.50	105.4(87-142)(10.2)	48.6(16.3)
bio	n	0.75	105.5(92-126)(8.3)	39.3(3.8)
bio	n	rand	104.0(85-118)(7.2)	43.7(4.7)
bio	c	0.05	96.2(85-143)(10.5)	116.6(7.8)
bio	c	0.125	95.0(85-112)(7.4)	84.3(7.7)
bio	c	0.25	100.0(86-117)(8.2)	55.8(3.0)
bio	c	rand	96.4(86-112)(6.0)	66.8(6.1)

Table 1. Comparison of various types of kick moves. The "Kick" column indicates the kick move type: multistart (m), random (r), net removal (n), and clustering (c).

Algorithm LSMC

Input: Hypergraph bisection instance,
iteration bound M,
temperature schedule $temp_i$, $i = 1, \ldots, M$

Output: Partitioning P_{best}

1. Generate a random partitioning P_{init};
   $P_1 = FMDescent(P_{init})$;
   $P_{best} = P_1$;

2. **for** $i = 1$ **to** M
   2.1. $P_i^* = kick\_move(P_i)$;
   2.2. $P_i^{**} = FMDescent(P_i^*)$;
   2.3. $diff = cost(P_i^{**}) - cost(P_i)$;
   if ($diff < 0$) $P_{i+1} = P_i^{**}$;
   else {
   generate a random number $t \in [0, 1)$;
   if ($t < \exp(-diff/temp_i)$);
   $P_{i+1} = P_i^{**}$;
   else
   $P_{i+1} = P_i$;
   }
   2.4. **if** ($cost(P_{best}) > \text{cost}(P_i^{**})$) $P_{best} = P_i^{**}$;

3. **return** P_{best};

Figure 1. LSMC algorithm template.

on variants of zero-temperature LSMC, we note that Hong et al. [9] have recently studied the utility of non-zero temperatures in applying LSMC to the TSP, and have shown that significant performance improvements can result.

In work related to our present application, [13] briefly describes the application of LSMC to graph partitioning on random geometric graphs, using the so-called *clustering kick move* that we describe in the next section. Recently, Shibuya et al. [15] proposed a method that strongly resembles LSMC, using a "Stable-Net Transition" kick move. In their method, a net is *stable* if it remains cut after a min-cut bisection is found. Since their experiments show that a high proportion of cut nets are always in the cut set during min-cut bisection, their method detects stable nets and perturbs cells incident to those nets in order to "uncut" these nets. This is similar to the *net removal kick move* that we study below. Finally, Hartoog [8] mentions what seems to be an early version of the *random kick move* that we study below, but does not give any performance results.

2. KICK MOVE STUDIES

Our work studies the performance of LSMC for VLSI netlist partitioning, with the goal of determining appropriate kick move strategies for this domain. Thus, we examine relationships between the kick move perturbation that is applied between successive FM descents and the quality of the final partitioning solution that results. We assume that the partitioning objective is min-cut bisection; following the usual practice, this means that the total module areas in the two bisections must differ by at most twice the largest individual module area in the netlist. We implemented the following kick moves for our study.

2.1. Multistart Kick Move

The *multistart kick move* completely randomizes the partitioning solution. Thus, it is equivalent to the standard "random multistart" approach of using multiple independent FM calls to yield a "stable" solution. The multistart kick move provides the baseline against which other kick moves can be compared.

2.2. Random Kick Move

The *random kick move* selects from each partition a random subset of modules whose total area is $MoveSize\%$ of the total module area in the smaller of the two partitions. In our experiments, $MoveSize$ is varied between 5-25% of the total area of the smaller partition. This kick move is intended to make random perturbations to the current solution, such that the perturbed solution will lie outside the basin of attraction of the current local minimum. As noted above, Hartoog [8] first reported the use of this technique as a heuristic for improving FM performance, but no performance data was given.

2.3. Clustering Kick Move

The *clustering kick move* is based on the intuition that the appropriate perturbation of a given solution is to swap clusters of connected modules between the partitions. In other words, if there are "islands" of tightly clustered modules in a partition, it is difficult for FM to move the island one node at a time because moving any one node of an island will seem to be a very suboptimal choice.[2] The details of the clustering kick move are given in Figure 2.

Clustering Kick Move

Input: A partitioning solution (A, B)

Output: Two clusters

1. Choose $v1 \in A$ and $v2 \in B$ as seeds,
   where $v1$ and $v2$ are on two different cut edges.
2. From each of the seeds, grow a cluster subject to the constraints:
   2.1. nodes in the cluster must be on the partition *opposite* to the seed node,
   2.2. the cluster is grown by finding neighbors of the current cluster
   2.3. the growth is terminated when the area of the cluster exceeds $MoveSize \times SmallerArea$ or when one cluster can no longer grow

Figure 2. Clustering kick move template.

In our experiments, $MoveSize$ is varied between 5-25% of the area of the smaller partition, and the cluster growth of Step 2.2 is accomplished by breadth-first search. The BFS termination criterion of Step 2.3 forces the swapped clusters to be of equal area, so that the partitions are always balanced. This clustering kick move was first proposed by [14]; we have also experimented with a variant which grows the cluster on the same side as the seed node, but the results are statistically indistinguishable and we do not report them here.

2.4. Net Removal Kick Move

Finally, the *net removal kick move* (Figure 3) is based on the observation that when a local minimum has been reached by the FM algorithm, any node belonging to an uncut net will be frozen in place since moving such a node will likely increase the cutsize. Thus, to escape from a local minimum we propose to *temporarily* "reduce" the netlist by removing some subset of uncut nets N_u from consideration. This allows nodes that are otherwise immobile to move across the

[2] This intuition is validated by recent results showing that FM performs significantly better if tie-breaking is resolved so that the algorithm effectively swaps entire clusters during each pass [6].

LARGE-STEP MARKOV CHAIN VARIANTS FOR VLSI NETLIST PARTITIONING

Alex S. Fukunaga, Dennis J.-H. Huang and Andrew B. Kahng
UCLA Computer Science Department, Los Angeles, CA 90095-1596 USA

ABSTRACT

We examine the utility of the *Large-Step Markov Chain* (LSMC) technique [13], a variant of the iterated descent heuristic of Baum [2], for VLSI netlist bipartitioning. LSMC iteratively finds a local optimum solution according to some greedy search (in our case, the Fiduccia-Mattheyses heuristic) and then perturbs this local optimum via a "kick move" into the starting solution of the next greedy descent. We empirically evaluate several intuitive types of kick moves to determine which is best suited to the VLSI netlist bipartitioning domain. Experiments show that LSMC with an appropriately chosen kick move can yield results that are competitive with the best known results in the literature. Ongoing work examines the variation of the optimal kick move with the underlying partitioning engine. Since LSMC can itself be viewed as a "partitioning engine", other research directions include the use of LSMC within hybrid-genetic and two-phase approaches.

1. INTRODUCTION

Partitioning optimizations are critical to the synthesis of large-scale VLSI systems. Designs with several hundred thousand gates are now common, and entail problem complexities that tax existing back-end physical layout tools. Thus, partitioning is used to divide the design into smaller, more manageable components. A standard model for VLSI layout associates a hypergraph $G = (V, E)$ with the circuit netlist; vertices in V represent modules and hyperedges in E represent signal nets. A bipartitioning of G divides the vertices in V into disjoint subsets U and W, with the *cut size*, $c(U, W)$, being the number of hyperedges in $\{e \in E \mid \exists u, w \in e \text{ with } u \in U \text{ and } w \in W\}$. Two standard partitioning formulations are: *min-cut bisection*, which seeks to minimize $c(U, W)$ subject to $|U| = |W|$, and *minimum ratio cut*, which seeks to minimize $\frac{c(U,W)}{|U| \cdot |W|}$. Both of these formulations are NP-complete, and much work has sought effective heuristic solutions. We focus on min-cut bisection and its relaxation where $|U|$ and $|W|$ can differ by up to a prescribed ratio.

Our work yields insight into the class of iterative partitioning approaches, which iteratively perturb a current solution and accept the new candidate solution according to either a greedy or hill-climbing strategy. Such methods are widely used because pre-placement constraints, module area information, and complex objectives can be transparently integrated. The most widely used iterative algorithm is that of Fiduccia and Mattheyses (FM) [5]. Wei and Cheng [16] use an adaptation of [5] to address the ratio cut objective.

The main weakness of FM and its variants is that solution quality is not "stable", i.e., it is not predictable. Hagen and Kahng [7] report that the distribution of solution costs (nets cut) for independent random FM executions on ACM SIGDA benchmark netlists is approximately normal. This implies not only that the average FM local minimum is significantly worse than the best possible FM solution, but also that FM must be run many times from random starting points to achieve a good result (that is to say, sample from the tail of this distribution of solution costs). Indeed, practical implementations of FM use a number of random starting configurations and then return the best result over all runs in order to attain "stability" [12]: we call this the *random multi-start* approach. The number of runs required to achieve stability via random multi-start grows rapidly with problem size. Despite these shortfalls, the appeal of iterative algorithms – and FM in particular – has led to much research which tries to make such methods more viable in practice.

The LSMC Approach

The Large-Step Markov Chain (LSMC) method of [13] iteratively performs a *descent* using a greedy search engine, and then perturbs the resulting local optimum via a "kick move" to obtain the starting solution for the next greedy descent. We use LSMC to generically encompass earlier "iterated descent" methods, including the Iterated Lin-Kernighan traveling salesman problem (TSP) heuristic of Johnson [10], which is believed to be the best-performing of all heuristics which obtain near-optimal solutions for the TSP. The idea of iterated descent seems to have originated with Baum in 1986 [2], also in the context of the TSP; note that the particular choice of kick move can strongly influence the perceived utility of LSMC, as can be seen from the contrasting results of [2] and [10], as well as the more detailed kick move studies of LSMC for the TSP in [9]. Figure 1 describes the LSMC approach, following the presentation in [13] but adapted to the partitioning domain, i.e., the greedy local optimization procedure is assumed to be the FM algorithm.[1]

As can be seen from the Figure, LSMC actually performs simulated annealing over the set of FM local minima, with $\{$ kick move + $FMDescent$ $\}$ as its neighborhood operator. In other words, if a new local minimum has lower cost than its predecessor, it is always adopted as the current solution; otherwise, its probability of being adopted is given by the Boltzmann acceptance criterion (Step 2.3 in the Figure), with the alternative being to retain the previous local minimum solution as the current solution. Based on experimentally derived intuition, the authors of [13] set the temperature used for Boltzmann acceptance to be zero in most cases. Similarly, Johnson [10] also used zero-temperature LSMC. Although our experiments have concentrated only

[1] An FM *descent* is a series of FM passes, beginning after a kick move perturbation is made and ending when a local minimum is reached (i.e., no cost improvement is made by a pass).