Multilevel Cooperative Search for the Circuit/Hypergraph Partitioning Problem

Min Ouyang, Michel Toulouse, Krishnaiyan Thulasiraman, Fred Glover, Jitender S. Deogun

Abstract— **Our objectives in this paper are twofold: design an approach for the netlist partitioning problem using the cooperative multilevel search paradigm introduced by Toulouse, Thulasiraman and Glover [18], and study the effectiveness of this paradigm for solving combinatorial optimization problems, in particular, those arising in the VLSI CAD area. We present a cooperative multilevel search algorithm CoMHP and describe a parallel implementation on the SGI O2000 system. Experiments on ISPD98 benchmark suite of circuits show, for 4-way and 8-way partitioning, a reduction of 3% to 15% in the size of hyperedge-cuts compared to those obtained by hMETIS. Bisections of hypergraphs based on our algorithm also outperform hMETIS, although more modestly. We present experimental results to demonstrate that the cooperation scheme plays a key role in the performance of CoMHP. In fact, the improvement in the quality of the solutions produced by CoMHP is to a large extent independent of the partitioners used in the implementation of CoMHP. The experimental results also demonstrate the effectiveness of the cooperative multilevel search paradigm for solving the netlist partitioning problem and show that the cooperative multilevel search strategy can be used as a paradigm for designing effective solution techniques for combinatorial optimization problems such as those arising in the VLSI CAD area.**

Keywords— **Multilevel algorithms, cooperative search, VLSI Physical Design, graph partitioning, combinatorial optimization.**

I. Introduction

Netlist partitioning is an important and well-studied research topic in the VLSI CAD area. Several classes of heuristics have been proposed to address this problem [2]. Recently, multilevel algorithms have been applied to the netlist partitioning problem [11]. This approach has since become the standard to partition netlists.

In the multilevel paradigm, Fiduccia-Mattheyses (FM) types of move-based heuristics execute moves in coarsened hypergraphs (hypergraphs are a common mathematical representation of netlists) that involve static clusters (blocks) of modules from the original netlist instance. Since the clusters are static, the search space of coarsened hypergraphs is often a tiny fraction of the search space of the original optimization problem. This reduction in the search spaces enables a speedy execution of multilevel algorithms. Unfortunately, it also imposes serious limitations on the ability of multilevel algorithms to provide good quality partitionings. These limitations of the multilevel paradigm have been recently addressed in [7], [11] using more dynamic coarsening strategies. In this paper we present a broader strategy to address this issue.

Our approach is based on a bottom-up algorithm design technique called cooperative search. According to this approach, a set of different search algorithms is first selected. Each algorithm is implemented as an independent program that runs in time sharing with the other programs on a sequential computer or in parallel if several computing units are available. If the difference among the programs is only based on the stochastic properties of a generic algorithm or on different search parameters, then we can think of those programs as the multiple restarts of the same algorithm. However, unlike restart, programs in a cooperative search interact with each other based on a *cooperation protocol* that specifies how the search programs cooperate at run time. Intuitively, as framed in Huberman's paper [10], cooperation is an exchange of "hints" that may confuse some search processes, but will also help others. Overall, hint sharing improves the performance and has been used with success to design search heuristics in the context of constraint satisfaction problems [3], [8] and to parallelize some metaheuristics [13], [14], [16], [17].

The present paper introduces the Cooperative Multilevel Hypergraph Partitioning algorithm (CoMHP), an asynchronous variation for hypergraph partitioning of the cooperative algorithm in [18]. Our hypergraph partitioning method uses a new netlist coarsening strategy which is based on partitioning rather than clustering as it is usually done by multilevel algorithms. Next, we introduce a cooperation protocol which supports a dynamic "re-coarsening" strategy addressing the convergence problems of standard multilevel algorithms. Finally, we give an intuitive description of the convergence behavior for this system of cooperating search algorithms.

The rest of the paper is structured as follows. Section II introduces a few definitions and our coarsening strategy. Section III describes the cooperative algorithm and its convergence behavior. Section IV reports and discusses the results of the tests conducted on the ISPD98 benchmark suite of circuits. Finally, Section V concludes with some suggestions for future work.

M. Ouyang is with Synopsys Inc., 700 E. Middlefield Rd., Mountain View, CA 94043, (mouyang@synopsys.com).

M. Toulouse is with the Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, R3T 2N2 (email: toulouse@cs.umanitoba.ca).

K. Thulasiraman is with the School of Computer Science, University of Oklahoma, (email: thulasi@cs.ou.edu).

F. Glover is with the Hearin Center for Entreprise Science, School of Business Administration, University of Mississippi, University, MS 38677 (email: fglover@bus.olemiss.edu)

J. Deogun is with the Computer science & Engineering Department, University of Nebraska-Lincoln, Lincoln, NE, 68588-0115 (email: deogun@cse.unl.edu).

II. Basic definitions and partition-based coarsening

Hypergraphs are commonly used as a formal representation of netlists. Let $H_0 = (V_0, E_0)$ be a hypergraph representation of a given netlist instance. V_0 is a set of n vertices and E_0 a set of m hyperedges which represent, respectively, the modules or vertices and signal nets of the netlist. The set E_0 is a subset of the powerset 2^{V_0} of the vertices in H_0, i.e., $e \in E_0$ is a subset of V_0. Note that the superset of a set S is defined as the collection of all subsets of S. Given this formalization, the problem of partitioning the modules of a netlist into k subsets $P_1, P_2, \ldots, P_k$ can be stated as a combinatorial optimization problem where one tries to find an instance $\{P_1, P_2, \ldots, P_k\}$ of the mapping

$$\mathcal{P} : V_0 \longrightarrow 2^{V_0}. \tag{1}$$

that minimizes the cost function:

$$f(x) = \sum_{i=1}^{m} w(e_i) \tag{2}$$

where $w(e_i) = 1$ if e_i is a hyperedge that spans more than one P_i, and $w(e_i) = 0$ otherwise. The subsets P_i are subject to the constraints:

1. $P_i \bigcap P_j = \emptyset$ $(i \neq j)$;
2. $\frac{|V_0|}{ck} \leq |P_i| \leq \frac{c|V_0|}{k}$ for some constant $c \geq 1.0$;
3. $\bigcup_{i=1}^{k} P_i = V_0$.

Constraints 1 and 3 ensure that $\{P_1, P_2, \ldots, P_k\}$ is a partition and constraint 2 sets bounds on the cardinalities of P_i in the partitions. The set of instances from Mapping (1) satisfying these constraints are identified as the *solution space* X_0 of this problem. The above problem is a generalization of the graph partitioning problem, an NP-complete problem [6]. Consequently, search heuristics come handy to provide good solutions in reasonable computational time.

Multilevel algorithms initiate processing by a hierarchical clustering of the vertices of H_0, yielding hypergraphs with fewer vertices, i.e., *coarsened hypergraphs*. Usually, recursive *matching-based clustering* algorithms such as *edge-coarsening*, *hyperedge-coarsening*, *maximal matching* and *modified-hypergraph-coarsening* [4], [11] are used to coarsen hypergraphs. Our clustering strategy addresses the problem of contracting the netlist as a partitioning problem. In this paper, hierarchical clustering and coarsened hypergraph are defined in the following manner:

Definition 1: Let

$$\mathcal{C}_i : V_0 \longrightarrow 2^{V_0},\ i = 1, \ldots, l \tag{3}$$

be a family of l mappings where $\mathcal{C}_i$ maps the vertices of H_0 into $|V_i|$ clusters $C_{i1}, C_{i2}, \ldots, C_{i|V_i|}$ such that $C_{iu} \cap C_{iv} = \emptyset$ if $u \neq v$ and $\cup_{j=1}^{|V_i|} C_{ij} = V_0$. The family of mappings in (3) defines a *hierarchical clustering* of the vertices of H_0 whenever $|V_i| > |V_{i+1}|$ for all $i = 1, 2, \ldots, l-1$.

Definition 2: A coarsened hypergraph $H_i = \{V_i, E_i\}$ is the set of vertices V_i and hyperedges E_i such that: 1) $v \in V_i$ is a cluster (subset) C_{ij} of vertices from V_0 as defined by the mapping $\mathcal{C}_i$ (Note: to simplify notation, v may also be denoted by the corresponding cluster C_{ij}); and 2) a subset e of V_i is a hyperedge of H_i if and only if there exists an hyperedge $e' \in E_0$ such that e' has nonempty intersection with each cluster represented by the vertices in e.

Basically, the above definition formally states how coarsened hypergraphs H_i are formed; each cluster in the partition defined by the mapping $\mathcal{C}_i$ represents a vertex of H_i and a hyperedge e of H_0 becomes an appropriately modified hyperedge of H_i if the vertices in e are not all in the same cluster of the partitioning defined by $\mathcal{C}_i$. For instance, if each hyperedge of H_0 is a pair of vertices in V_0, then for $u \neq v$, $e = \{C_{iu}, C_{iv}\} \in E_i$ iff $\exists\, a, b \in V_0$ such that $a \in C_{iu}$ and $b \in C_{iv}$ and $\{a, b\}$ is a hyperedge in E_0.

In our hierarchical clustering approach, each mapping $\mathcal{C}_i$ is obtained by solving a hypergraph partitioning problem for H_0. The family of mappings as defined in (3) is derived from the solution of the $\frac{n}{2^i}$-way partitioning problems for H_0, for $0 < i \leq l$. The solution of the $\frac{n}{2^i}$-way partitioning, i.e. the $\frac{n}{2^i}$ subsets, become the set of vertices $V_i = \{C_{i1}, C_{i2}, \ldots, C_{i\frac{n}{2^i}}\}$. Next the coarsened hypergraphs H_i are generated from the mappings $\mathcal{C}_i$ as described in Definition 2. We identify this hierarchical coarsening strategy as *partition-based coarsening*. This algorithm can be trivially parallelized by running the partitioning processes and the hypergraph generations on different processors. The parallel time requirement is dominated by the processor that computes the partitioning for $k \approx \frac{n}{2}$.

III. CoMHP: Description and convergence behavior

In this section, we present in detail, the different components of our multilevel cooperative search algorithm CoMHP, and a model to study its convergence behavior. There are $l+1$ processes in CoMHP, the same as the number of hypergraphs. Each process p_i takes as input the hypergraph H_i. The processes run in parallel, applying the same composition of FM-like hypergraph partitioning heuristics to partition their hypergraph. According to Definition 1, moving a vertex v in a coarsened hypergraph H_i is equivalent to moving a cluster of vertices in H_0, the cluster corresponding to vertex $v \in H_i$. Consequently, while a process p_i searches for good partitionings in the coarsened hypergraph H_i, it actually explores the solution space X_0. Though all the processes use the same local search methods, the searches in X_0 do not overlap completely. This is because, as in standard multilevel algorithms, hierarchical clustering in CoMHP creates different neighborhoods to solutions in X_0.

The computational cost of executing a move in neighborhoods based on coarsenings is independent of the size of the clusters in H_0. It only depends on the number of neighbors of the current solution; this number goes on decreasing as the hypergraphs are more highly coarsened. This is a clear advantage of these neighborhoods. On the other hand, if the coarsening cannot be undone, the logical moves in H_0 only involve the clusters of vertices in H_0 associated by

the Mapping (3). All the possible combinations of such clusters in k subsets can only generate a small fraction of the solutions in X_0. This is not a problem by itself since it is usual for local searches to explore only a fraction of the solution space. Rather, the problem is when the small search spaces of coarsened hypergraphs have no good solutions, their exploration is then hopeless, even using the best search methods.

This limitation of the coarsening approach has in fact been partially identified in [7], [11] by observing that an initial partitioning can be refined in different ways depending upon how the coarsening is executed [11]. Those papers propose a multi-phase refinement as a mechanism to refocus the search spaces. Such multi-phase refinement is based on a recursive call of the multilevel algorithm from the same coarsened hypergraph in the multilevel structure. A randomized coarsening is used during the multi-phase refinement. In [11], this coarsening is initiated from the best partitioning obtained from the refinement or from the previous iteration of the multi-phase refinement. The multi-phase refinement iterates until the best solution cannot be improved further. The initial refinement phase of the multilevel procedure is then resumed.

Our multilevel cooperative algorithm is similar in spirit to multi-phase refinement. It also produces many "re-coarsenings" of the hypergraphs which allow the search processes to explore new regions in the solution space X_0.

A. Refinement Phase

The initial coarsening may fail to provide access to good regions of the solution space. We now describe how the partitionings discovered by the search heuristics during the refinement phase can be used to develop a dynamic coarsening strategy. Our dynamic coarsening strategy is supported by the cooperation protocol of CoMHP. This protocol specifies the processes' neighborhood structure, the kind of interactions (hint exchanges) allowed among search processes and what to do with the hints exchanged.

The neighborhood structure in the current implementation is an array of processes: process p_i can only interact with processes p_{i-1} and p_{i+1}. This neighborhood structure mirrors the hierarchical clustering of H_0, where H_i has for neighbors hypergraphs H_{i-1} and H_{i+1} in the sequence of increasingly coarsened hypergraphs. The boundary conditions in the array are handled in the following manner: p_0 only interacts with p_1, p_l only interacts with p_{l-1}.

Each interaction involves exchange of "hints" or information based on "elite partitionings" and taking appropriate actions. Elite partitionings are those that we expect to give us hints as to the nature of optimum partitionings. For instance, a partitioning, which has the smallest hyperedge-cut among all those partitionings generated by the search method used, may be considered elite at that stage. Let X_i be the search space of H_i, whereas X_i' is the set of elite partitionings of H_i. Different approaches have been tested to create each elite set X_i' of hypergraph H_i. The approach presented in Fig. 4 initializes the elite set to a good partitioning generated by a search method. A newly generated partitioning is added to X_i' if its hyperedge-cut value is not larger than 10% of the smallest of the hyperedge-cuts of the partitionings already in the set X_i'. We also limit the size of X_i'. If the addition of a partitioning increases this limit, we remove from X_i' the partitioning which has the highest hyperedge-cut value among all those already in the set X_i'.

The operators, to be described soon, specify how to use the elite partitionings at the level of each process. The local partitioning operator and the local clustering operator recoarsen or redefine the coarsening of hypergraphs. On the other hand, the interpolation operator re-initiates the search of some of the move based heuristics used by each process. More specifically, in the interpolation operator, one elite partitioning from H_{i+1} is selected as the initial solution of a move based heuristic in hypergraph H_i. We now describe, in detail, these three operators. Fig. 1 is used for illustration of the workings of these operators. Fig. 1(a) gives H_0, H_1 and H_2 with one elite partitioning for each indicated by dashed lines.

Fig. 1. Interaction operators: (a) coarsened hypergraphs, dashed lines indicate partitionings; (b) local partitioning operator; (c) local clustering operator; (d) interpolation operator.

A.1 Local partitioning operator

Local partitioning changes the coarsening of H_i by splitting *some* of its vertices. Vertices (clusters) in H_i are split based on the information provided by the set of elite partitionings X_{i-1}' from hypergraph H_{i-1}. The local partitioning operator finds clusters $v \in V_i$ (line 1 in Fig. 2) such that v has at least two vertices $a, b \in V_0$ that are into two different subsets (line 4) of one of the elite partitionings of X_{i-1}'. When this happens, the vertices of H_0 in the intersection of the sets $v \cap P_s$ form a new vertex v_j of H_i (line 5). Once the local partitioning operator has completed, a special routine is called which generates a new coarsened hypergraph H_i reflecting the changes in the mapping $\mathcal{C}_i$ provoked by the execution of the local partitioning operator. Note that in

```
Local_partitioning( );
1. for each vertex v ∈ V_i
2.     for each elite partitioning x ∈ X'_{i-1}
3.         for each subset P_s of elite partitioning x
4.             if ((v ∩ P_s ≠ ∅) & (v ⊄ P_s)) then
5.                 create new vertex v_j = v ∩ P_s;
6.                 if v not yet marked to "delete" then mark it;
           endfor;
7.         if v is marked to "delete" goto line 1;
       endfor;
   endfor;
```

Fig. 2. Pseudo-code of the local partitioning operator

the pseudo-code of Fig. 2, after the first elite partitioning for which we find at least one nonempty intersection with a subset in this partitioning, we end the "for loop" (beginning at line 1) for this vertex. In other words, we split a vertex with respect to only one elite partitioning, if possible.

We shall first illustrate the local partitioning operator with an hypothetical example. Then the illustration will

be given using Fig. 1.

Consider the hypergraph H_i. Pick any vertex v of H_i (line 1 in Fig. 2). Note that $v \in V_i$ is a cluster of vertices of H_0. Let $v = \{2, 7, 8, 9\}$. Here 2, 7, 8 and 9 are all vertices of H_0. Consider an elite partitioning x of H_{i-1} (line 2 in Fig. 2). Here, we assume that the optimization problem is to search for optimum 3-way partitionings. Let the three subsets in the partitioning x be $P_1 = \{1, 2, 3, 8, 12\}, P_2 = \{9, 11, 13, 14\}$ and $P_3 = \{4, 5, 6, 7, 10\}$. Now we get (line 4 in Fig. 2):

$$v \cap P_1 = \{2, 8\}, v \cap P_2 = \{9\} \text{ and } v \cap P_3 = \{7\}.$$

We then create the following new vertices (line 5 in Fig. 2) as a result of splitting vertex v of H_i:

$$\{2, 8\}, \{9\} \text{ and } \{7\}.$$

Once we identify a partitioning x (as above) and split vertex v, then v is marked to be deleted (line 6) and the operator returns to line 1 to pick a new vertex and check if it could be split as above. In case the elite partitioning x does not help to split the vertex v, we pick another partitioning from the list of elite partitionings X'_{i-1}. If no elite partitioning leads to a splitting of v, then this vertex v in H_i is left unchanged as it was before the execution of the local partitioning operator. The algorithm for the local partitioning operator terminates once all the vertices of H_i have been considered for possible splittings.

We now return to Fig. 1 for another illustration of the local partitioning operator with a concrete example. Consider the hypergraph H_1 of Fig. 1(a). (Vertices of coarsened hypergraphs as well as the subsets of partitionings are sets of vertices from the hypergraph H_0.) This hypergraph has 4 vertices $a, b, c, d \in V_1$, where a is a cluster $\{0, 1\}$ of vertices of H_0, b is a cluster $\{2, 3\}$ of vertices of H_0, and so on. The elite bisection of H_0 is $P_1 = \{0, 1, 3, 7\}$ and $P_2 = \{2, 4, 5, 6\}$. This elite bisection indicates that potentially good solutions exist in the region of the solution space in which vertices 2 and 3 are in the different subsets of each bisection. But in H_1, vertices 2 and 3 form a cluster; swaps on H_1 always move vertices 2 and 3 of H_0 together so that they are in the same subset of the bisection of H_1 because they form a single vertex in H_1. The region of the solution space corresponding to the elite bisection of H_0 is not reachable in the search space defined by the coarsened hypergraph H_1. Although the search space of H_1 is smaller than X_0 and therefore faster to explore, this advantage is lost because search space X_1 does not overlap with good solutions of the basic optimization problem. This problem is detected by the local partitioning operator because the intersection of the sets b and P_1 is not empty and b is not strictly included in P_1. Similarly, the intersection of the sets b and P_2 is non-empty and b is not strictly included in P_2. Also, the intersections of vertex $d = \{6, 7\}$ with P_1 and P_2 are non-empty. In other words

$$b \cap P_1 = \{3\} \text{ and } b \cap P_2 = \{2\}$$
$$d \cap P_1 = \{7\} \text{ and } d \cap P_2 = \{6\}$$

So, the vertex b is split into the two vertices $\{3\}$ and $\{2\}$, and the vertex d is split into the two vertices $\{7\}$ and $\{6\}$. Note that the vertices a and c are not split because they are strictly included in P_1 and P_2 respectively. The resulting coarsening of H_1 is shown in Fig. 1(b). Note that this new coarsening of H_1 enables the search space X_1 to overlap with the regions of X_0 identified by the elite bisection.

A.2 Local clustering operator

Local clustering changes the coarsening of H_i by merging *some* of the vertices (clusters) of H_{i-1} into new vertices for H_i. A vertex from H_{i-1} becomes a "candidate for merging" if it is strictly included in one of the subsets of each elite partitioning from X'_{i-1} (line 5 in Fig. 3). The identification of which vertex can be candidate for merging is needed because between the time a partitioning enters the set of elite partitionings and the time local clustering is run, many vertices of H_{i-1} may have been split or clustered such that some of them may overlap more than one subset of an elite partitioning. According to the definition of Problem (2), a vertex is strictly included in exactly one subset of a partitioning. Therefore, if $j = k$ in line 6, vertex v is strictly included in one subset of each elite partitioning. Once the candidates for merging have been identified,

```
Local_clustering( );
1. for each vertex v ∈ V_{i-1}
2.     j = 0;
3.     for each elite partitioning x ∈ X'_{i-1}
4.         for each subset P_s in partitioning x
5.             if (v ⊂ P_s) then j = j + 1;
           endfor;
       endfor;
6.     if j is equal to k, mark v as a candidate for merging;
   endfor;
```

Fig. 3. Pseudo-code of the local clustering operator

a pair of vertices can be merged if both vertices satisfy the two following conditions: 1) both vertices lie on the same hyperedge; 2) both vertices are together in the same subset in all elite partitionings. Note that two vertices v_1 and v_2 of H_{i-1} are on the same hyperedge e of H_0 if they are both strictly included in e. For instance, in the case each hyperedge is a pair of vertices, then v_1 and v_2 are on the hyperedge e if $e = \{v_1, v_2\}$. Once a vertex is identified as candidate for merging, the vertex is labeled with the subset of each elite partitioning where it has been found. Vertices with the same label satisfy condition 2 above.

We shall next illustrate the local clustering operator with an hypothetical example. Consider vertices v_1 and v_2 of H_{i-1} (line 1 in Fig. 3). Let $v_1 = \{6, 8\}$ and $v_2 = \{12, 13\}$, and let x_1, x_2 and x_3 be the three elite partitionings of H_{i-1}. That is, they form the set X'_{i-1}. Assuming 3-way partitionings, let the three subsets in x_1, x_2 and x_3 be

$$x_1 = \{2, 7, 9, 11\}, \{1, 3, 5, 10\}, \{4, 6, 8, 12, 13, 14\}$$
$$x_2 = \{1, 3, 4, 5, 10\}, \{2, 7, 9, 11, 14\}, \{6, 8, 12, 13\}$$
$$x_3 = \{6, 8, 12, 13, \}, \{2, 7, 9, 11, 14\}, \{1, 3, 4, 5, 10\}$$

We can see that the vertex $v_1 = \{6, 8\}$ is strictly included in one of the subsets in each of the above elite partitionings

(lines 3, 4, 5 in Fig. 3) and so it is eligible for merging. Similarly, the vertex $v_2 = \{12, 13\}$ is also eligible for merging. If $e = \{3, 6, 8, 9, 10, 12, 13\}$ is an hyperedge, then both v_1 and v_2 lie on this hyperedge, and so we can merge them to form a vertex $\{6, 8, 12, 13\}$ for the hypergraph H_i.

Now we return to Fig. 1 for another illustration of the local clustering operator with a concrete example. According to the elite bisection of H_0 (see Fig. 1(a)), vertices 0 and 1 are in the same subset of this elite bisection. These vertices also lie on the same hyperedge. So, they are merged to form vertex a of H_1 (see Fig. 1(c)). For similar reasons, we merge vertices 4 and 5 to form vertex b of H_1, merge vertices 2 and 6 to form vertex c of H_1 and merge vertices 3 and 7 to form vertex d of H_1. These vertices a, b, c and d now define the new coarsened hypergraph H_1.

Local clustering tends to reduce the number of vertices in a coarsened hypergraph and therefore to reduce the size of the search space. This balances the effect of the local partitioning operator which tends to increase the number of vertices. It also makes the search space to retreat from overlapping with uninteresting regions of the solution space. Together, local partitioning and local clustering operators allow the search spaces to move in the solution space, which is the desired effect of re-coarsening.

A.3 Interpolation Operator

In the interpolation operator, one partitioning from the set of elite partitionings of H_{i+1} is selected to be the initial solution of a move-based heuristic in hypergraph H_i. Our interpolation operator would be identical to the interpolation operator of multilevel algorithms if it were not for the way we compute the coarsened hypergraphs. Recursive coarsening used by most multilevel algorithms is such that vertices $C_{ij} \in H_i$ are formed by the clustering of two vertices from H_{i-1}. For example, when vertices $u, v \in H_0$ are mapped together to an aggregate $C_{(i-1)j} \in H_{i-1}$, those two vertices are necessarily mapped to C_{ij}, the superset of $C_{(i-1)j}$ in the coarsened hypergraph H_i. Hypergraphs generated with recursive coarsening are said to be related level by level. This is not necessarily the case with our coarsening strategy. The vertices of H_0 that are mapped to a cluster $C_{ij} \in H_i$ may be spread over several clusters in each hypergraph H_j, $j > i$.

Returning to Fig. 1, assume the elite partitioning of hypergraph H_2 is selected as initial solution for a move-based heuristic of process p_1 associated with hypergraph H_1. The elite partitioning cannot be used because hypergraphs H_1 and H_2 are not related level by level. For example, vertex a of H_1 spreads over vertices e and f of hypergraph H_2 (same thing for b). The elite partitioning of H_2 cannot be a partitioning for the hypergraph H_1. In order to use the partitioning from H_2, we change the coarsening of H_1. A split of vertices in H_1 that spreads over more than one vertex in H_2 is performed using a similar procedure as for the local partitioning operator. That is, we consider splitting the vertices a, b, c and d of H_1 (see Fig. 1(a)) using the bisection of H_2 shown in Fig. 1(a). As shown in Fig. 1(d), after the split, it becomes possible to use the elite partitioning from H_2 as an initial solution to one of the move-based heuristics in our search method.

A.4 The search heuristic of CoMHP's processes

We are now ready to describe the search heuristic run by the $l + 1$ search processes of CoMHP. This search heuristic combines, in a single iterative loop, several hypergraph partitioning heuristics and the three operators that handle hints from neighbor processes. Each iteration of the loop executes the following sequence of operations: local partitioning, local searches, interpolation, local searches, local clustering, local searches and global searches. Fig. 4 contains an abbreviated pseudo-code for this loop for process p_i.

```
CoMHP( ); /* process p_i */
Initialization:
   Compute H_i using n/2^i-way partitioning
      algorithm (see Section II);
   Compute an initial partitioning x;
   X'_i = x; BestEdgeCut = f(x);
While not terminated /* begin main loop */
1.   Apply local partitioning operator to H_i;
1a.  For each partitioning y found using FMS and PFM on new H_i
1b.     if f(y) < BestEdgeCut then
           X'_i = X'_i ∪ y; BestEdgeCut = f(y);
           if |X'_i| > NumberOfAllowedEliteSolutions then
              Remove the worst elite solution from X'_i;
2.   Get a partitioning x from X'_{i+1} (interpolation operator);
2a.  For each partitioning y found using FMS and PFM with
     x as the initial solution
2b.     run 1b;
3.   Apply local clustering operator to current H_i;
3a.  Run 1a and 1b;
4.   If number of vertices < 500 do random search
     else execute hMETIS;
4a.  Run 1b;
5.   Test termination criterion;
/* end main loop */
End CoMHP
```

Fig. 4. CoMHP for process p_i

The initialization phase computes a coarsened hypergraph and an initial partitioning. The initial partitioning is required to provide a first elite solution to the set X'_i and as an initial solution to the iterative local search methods used in the main loop (unlike constructive methods, iterative methods have to be provided with an initial solution that they then improve).

The execution sequence of the interaction operators (local partitioning, interpolation, local clustering) has been chosen arbitrarily. However, once the local partitioning operator is performed at level i, hypergraph H_i is modified. We then run iterative local search methods (line 1a) on this new hypergraph H_i to calculate new hyperedge-cuts as well as to discover potential elite solutions (line 1b). As local search methods we have used the Sanchis partitioning algorithm (FMS) [15], and the multiway partitioning by free moves (PFM) proposed by Dasdan and Aykanat [5]. Next, on line 2, the interpolation operator is applied to get an elite solution x from X'_{i+1}, so that x can be used as an initial solution for the search methods FMS and PFM. Finally, the local clustering operator is ap-

plied, which transforms H_i once again, allowing to repeat the execution of FMS and PFM on a different hypergraph H_i.

The search of line 4 in Fig. 4 serves two different purposes. For highly coarsened hypergraphs, less than 500 vertices for example, several random searches could be executed. As these hypergraphs are very small, the main loop runs very fast and the process may run out of elite partitionings from neighboring processes. By executing many random searches, we slow down the execution of the main loop while having chances to discover good partitionings. The second purpose is to execute a search of H_i which does not depend on any initial partitioning. Both the hMETIS software and random searches have this characteristic (our random search routine consists of an initial partitioning generated randomly, followed by the execution of an iterative search to refine the random partitioning).

All the processes run the same combination of heuristics and interaction operators, except for the special conditions that hold for the boundaries processes p_0 and p_l and for the difference between highly coarsened hypergraphs and the other hypergraphs.

CoMHP is an asynchronous algorithm in the sense that each process executes the iterative loop of the search method without synchronization with other processes. Therefore, processes associated with highly coarsened hypergraphs run more iterations of the search method compared, for example, to the process associated with H_0. The time requirement for the cooperation protocol is the same for all processes. Furthermore, this time requirement is insignificant compared to the time required by the search heuristics.

B. Convergence behavior of CoMHP

Because of the local interactions among the search processes, the convergence behavior of cooperative algorithms is sometime modeled according to the theory of complex systems. For example, Huberman [10] uses a probabilistic model to show that the performance of cooperating processes is log-normally distributed for successful cooperative algorithms, in contrast with the normal distribution of independent searches (or restarts). In this model, the effect of cooperation on the distribution is a smaller number of average quality searches but an increase of the length of the tails on both sides of the distribution. The long tail of the positive side of the distribution produces the overall performance improvement. But to achieve such performance improvement, the cooperative procedure must satisfy the following requirements:

1. A large set of heuristically guided searches.
2. The searches apply successfully different search strategies, leading to non-redundant explorations of the solution space.
3. Processes exchange some useful information (hints) that allows some of them to cut the number of steps required to reach an optimal or acceptable solution.
4. Hints are statistically independent.

Requirements 1 and 2 are necessary to provide statistical independence among the hints. In practice, however, these requirements often conflict with one another. Either the number of guided searches is too small or the explorations of the different searches overlap in the solution space. When this happens, cooperative programs do not provide consistent quality of solutions: they converge well on some instances, yet very poorly on other instances of the same optimization problem.

Beside addressing the limitations induced by static coarsened hypergraphs in standard multilevel algorithms, a second motivation for mixing the cooperative paradigm with the multilevel paradigm has been to address the issue of conflicting requirements facing many implementations of cooperative algorithms. In the context of cooperative search, the stability of coarsened hypergraphs helps to reduce the negative influence that sub-optimal hints can have on the convergence behavior of the system. Hints have first to change the static neighborhood structures defined by coarsened hypergraphs before directing the search in new regions of the solution space. This is unlike any other cooperative algorithms. We believe that this is the reason why the best solutions found by CoMHP are very close to those of cooperative procedures that meet the requirements above. However, the performance of cooperating processes in CoMHP does not settle in a log-normal distribution because there are usually too few processes and hints are often strongly correlated. Rather, the cooperation protocol in CoMHP is such that it tends to minimize the differences between the hyperedge-cuts of the elite partitionings. The evolution of CoMHP settles in a minimum energy state (such as Hopfield networks [9]). The energy function is given by the sum of the differences between the averages for the hyperedge-cuts $f(x)$ of the elite partitionings x of the different processes:

$$E(\mathcal{X}) = \sum_{i=0}^{l-1} \{ \frac{\sum_{x \in X'_i} f(x)}{|X'_i|} - \frac{\sum_{x \in X'_{i+1}} f(x)}{|X'_{i+1}|} \} \qquad (4)$$

The initial state of the system is given by the elite partitionings computed after the coarsening phase. If the coarsening phase is successful, the elite partitionings from neighboring processes will have different hyperedge-cuts. The differentials among the best hyperedge-cuts of neighbor processes create opportunities to change, by percolation, the coarsening of the neighbor hypergraphs. New coarsenings provide new elite partitionings which in turn affect the coarsening of neighboring hypergraphs. This percolation process stops to have an impact on the exploration of the solution space when all the elite partitionings have about the same hyperedge-cuts, which corresponds to a minimum energy level of the system. Once the system has reached such minimum energy state, the quality of the best partitioning does not improve much. In terms of the best partitioning, when the system (as modeled in Equation (4)) is stable, that is, it has reached a fixed point, the computation can then be ended.

IV. Experimental Results

We have evaluated the performance of our CoMHP algorithm on the ISPD98 benchmark suite of netlists [1], comparing the performance of CoMHP with version 1.5.3 of the hMETIS partitioning package. We have implemented a parallel version of our hypergraph partitioning algorithm and have run it on the SGI computer at the RCF (Research Computing Facility) of the University of Nebraska-Lincoln. hMETIS has also been run on this same environment. RCF possesses a shared memory SGI O2000 system with 16 250Mhz R10k CPUs, 4GB main memory, and runs on the IRIX 6.5 Operating System. For each problem instance, we have executed 10 runs of hMETIS with recursive bisection and 10 runs with hMETIS-Kway (the direct approach) [12]. Our algorithm has been run for 10 iterations of process p_0. Since hypergraph H_0 is the largest one in the sequence of hypergraphs, process p_0 takes more time than any other process to complete one iteration of the refinement phase.

Tables I and II present the 2,4,8-way hyperedge-cuts for, respectively, the unit cell area and the non-unit (real) cell area with CoMHP (Co) and hMETIS (hM). Out of the 108 tests executed, hMETIS outperforms or yields the same results as CoMHP in 8 instances, while CoMHP outperforms hMETIS in 100 instances. For 2-way partitioning, the improvements of CoMHP over hMETIS are not significant. For 4-way and 8-way partitioning, CoMHP can get up to a 15% improvement in the hyperedge-cuts over hMETIS. For hMETIS, Tables I and II report the best solution of bisection or hMETIS-Kway. In 102 cases, hMETIS with bisection found the best solution while hMETIS-Kway found the best solution in the 6 other instances.

Tables III and IV present the runtimes (parallel computational time) of both algorithms. For CoMHP, the runtime indicates the total time to run 10 iterations of p_0 plus the time to perform the coarsening phase. For hMETIS we report the time to execute 1 run of the bisection approach in order to factor the use of several processors by CoMHP. This biases the results slightly in favor of hMETIS given that CoMHP uses 10 processors only for a few problem instances.

As can be seen from Tables III and IV, on average hMETIS is 20 to 25 times faster than CoMHP for the 108 tests. A time optimized implementation of CoMHP can improve on the current prototype in the following ways. The outer loop of CoMHP has only a few sequential dependencies, therefore it can be easily parallelized. Though this parallelization will not reduce the work ratio between CoMHP and other partitioners, it will considerably improve the time ratio. Secondly, the amount of improvement in the hyperedge-cuts of CoMHP is not significant after 2 or 3 iterations of the search phase by process p_0. At that point the energy function (4) is low and seems stable in its minimum. Running the current prototype implementation of CoMHP only 2 or 3 iterations will not result in any serious degradations of the results obtained using 10 iterations, which means we can get similar results as in Tables I and II with only about 1/5 to 1/3 of run times as

TABLE I

Min-cut 2,4,8-way partitioning results with up to a 10% deviation from exact partitioning, cells are assigned unit area (Columns "hM" and "Co" stand respectively for hMETIS and CoHMP).

Circuit	2-way		4-way		8-way	
	hM	Co	hM	Co	hM	Co
IBM01	180	180	495	430	750	711
IBM02	262	262	616	560	1841	1483
IBM03	953	950	1682	1619	2402	2219
IBM04	529	530	1689	1597	2778	2507
IBM05	1708	1697	3024	2888	4306	3874
IBM06	889	890	1484	1465	2275	2204
IBM07	849	824	2188	2036	3308	3098
IBM08	1142	1140	2363	2241	3469	3240
IBM09	629	620	1670	1606	2659	2474
IBM10	1256	1249	2283	2164	3761	3305
IBM11	960	960	2321	2196	3433	3160
IBM12	1881	1872	3730	3520	5972	5384
IBM13	840	832	1661	1671	2717	2483
IBM14	1891	1816	3278	3097	5060	4263
IBM15	2598	2619	5019	4591	6623	5960
IBM16	1755	1709	3816	3745	6475	5360
IBM17	2212	2187	5395	5194	8695	7960
IBM18	1525	1521	2881	2810	5169	4435

TABLE II

Min-cut 2,4,8-way partitioning results with up to a 10% deviation from exact partitioning, cells are assigned non-unit (actual) area.

Circuit	2-way		4-way		8-way	
	hM	Co	hM	Co	hM	Co
IBM01	217	215	343	340	606	573
IBM02	266	247	470	399	833	762
IBM03	707	608	1348	1220	1981	1879
IBM04	440	438	1321	1209	2408	2241
IBM05	1716	1681	3002	2895	4331	3950
IBM06	367	363	1149	1056	1716	1688
IBM07	716	721	1539	1480	2918	2707
IBM08	1149	1120	2143	1992	3330	3120
IBM09	523	519	1418	1334	2337	2079
IBM10	769	734	1845	1636	3098	2751
IBM11	697	688	1893	1699	2948	2768
IBM12	1975	1970	3577	3402	4957	4762
IBM13	859	832	1698	1568	2439	2298
IBM14	1520	1494	3048	2869	4833	4360
IBM15	1786	1771	4435	4314	6111	5756
IBM16	1681	1639	3562	3149	5580	5146
IBM17	2252	2156	4824	4393	8222	7003
IBM18	1520	1520	3104	2941	4833	4416

in Tables III and IV. Thirdly, the computational time of CoMHP is dominated by the execution of the global and local search subroutines. We believe we can reduce the time spent in the global and local searches by adapting these routines to CoMHP, for example, by not flipping all vertices for refinement, but rather stopping the search after flipping part (20%, for example) of the vertices. However, even if all these optimizations were realized, it is obvious that CoMHP will not be faster than hMETIS, or other partitioners for that matter, given that CoMHP uses repeatedly those partitioners as subroutines. On the other hand, with the same amount of computing resources as given to CoMHP (when run for 10 iterations of p_0), hMETIS didn't improve noticeably the quality of partitionings reported in Tables I and II. The situation is, however, different when hMETIS is embedded in CoMHP. Computational results from a "cooperative hMETIS' to be called ChMETIS are reported in Table V for netlists IBM01 to IBM08. The computation of these hyperedge-cuts is based on the procedure of Fig. 4 after replacing the local partitioners FMS and PFM by calls to hMETIS (except for line 2a, since search after interpolation starts from an initial partitioning). The quality of hyperedge-cuts produced by ChMETIS is very close to the quality of those produced by CoMHP. Since hMETIS is faster than FMS and PFM, computational times were about 10 to 20% better than CoMHP. Table V clearly demonstrates that the cooperation scheme plays a key role in the quality of the solutions produced by CoMHP. The improvement in the quality of the solutions produced by CoMHP is to a large extent independent of the partitioners used in the implementation of CoMHP.

V. Summary and Discussion

We have explored two objectives: design an approach for the netlist partitioning problem using the cooperative multilevel search paradigm introduced by Toulouse, Thulasiraman and Glover [24], and study the effectiveness of this paradigm for solving combinatorial optimization problems, in particular, those arising in the VLSI CAD area. We have presented the design and parallel implementation of an algorithm, called CoMHP, for the netlist partitioning problem. In this algorithm we combine the multilevel paradigm and the cooperative search paradigm and take advantage of the good features of both these paradigms. To date, the most successful approach to the netlist partitioning problem has been the multilevel algorithm hMETIS of Karypis, Aggarwal and Kumar [11] which formulates the netlist partitioning problem as a hypergraph partitioning problem. So, we have chosen this algorithm for a comparative evaluation of the quality of solutions produced.

In CoMHP, each level is associated with a coarsened (appropriately reduced) hypergraph and a search program derived from known heuristics such as the Fiduccia-Mattheyses (FM) heuristics. These programs execute searches on the coarsened hypergraphs at their respective levels. A distinguishing feature of CoMHP is the use of a cooperation protocol to control the coarsening of the hypergraphs at the different levels. This involves the use of

TABLE III
Run-time performance for min-cut 2,4,8-way partitioning with up to a 10% deviation from exact partitioning, cells are assigned unit area.

Circuit	2-way		4-way		8-way	
	hM	Co	hM	Co	hM	Co
IBM01	0.2	5	0.3	7	0.5	11
IBM02	0.4	10	0.7	12	1.1	21
IBM03	0.4	16	0.8	17	1.1	25
IBM04	0.5	16	1.0	19	1.3	26
IBM05	0.7	18	1.2	24	1.6	30
IBM06	0.6	21	1.2	23	1.7	33
IBM07	1.1	32	2.0	38	2.6	53
IBM08	1.6	36	2.6	51	3.4	59
IBM09	1.0	34	2.0	40	2.6	58
IBM10	2.2	56	3.5	65	5.0	91
IBM11	1.5	50	3.0	59	3.9	78
IBM12	1.9	62	4.6	73	5.1	115
IBM13	2.0	60	3.6	72	5.1	100
IBM14	5.9	79	9.1	141	13.0	169
IBM15	6.6	121	11.0	176	14.1	217
IBM16	7.6	142	13.3	192	19.0	238
IBM17	9.4	219	17.1	196	22.2	374
IBM18	7.7	178	15.1	192	20.4	301

TABLE IV
Run-time performance for min-cut 2,4,8-way partitioning with up to a 10% deviation from exact partitioning, cells are assigned non-unit (actual) area.

Circuit	2-way		4-way		8-way	
	hM	Co	hM	Co	hM	Co
IBM01	0.2	6	0.3	7	0.5	11
IBM02	0.3	10	0.7	13	1.0	20
IBM03	0.4	11	0.8	19	1.2	26
IBM04	0.5	16	0.9	18	1.3	26
IBM05	0.6	18	1.2	23	1.6	35
IBM06	0.5	15	1.2	22	1.7	35
IBM07	1.0	29	2.0	41	2.7	54
IBM08	1.2	25	2.2	35	3.1	57
IBM09	1.1	40	1.8	45	2.6	65
IBM10	1.7	52	3.4	64	4.9	93
IBM11	1.4	44	2.7	53	4.4	88
IBM12	2.0	58	3.8	75	5.1	113
IBM13	1.9	53	3.7	71	4.9	113
IBM14	6.0	81	9.0	145	13.0	151
IBM15	5.6	111	12.0	160	14.2	197
IBM16	6.7	168	13.1	197	18.0	264
IBM17	11.2	243	18.2	286	23.8	354
IBM18	8.7	189	15.9	235	20.5	296

TABLE V
COMPARING HYPEREDGE-CUTS BETWEEN CoHMP AND ChMETIS (COLUMNS "Co" AND "Ch" STAND RESPECTIVELY FOR CoHMP AND ChMETIS). UP TO A 10% DEVIATION FROM EXACT PARTITIONING, CELLS ARE ASSIGNED UNIT AREA.

Circuit	2-way		4-way		8-way	
	Co	Ch	Co	Ch	Co	Ch
IBM01	180	180	430	431	711	705
IBM02	262	262	560	537	1483	1492
IBM03	950	950	1619	1646	2219	2294
IBM04	530	527	1597	1573	2507	2534
IBM05	1697	1703	2888	2905	3874	3875
IBM06	890	892	1465	1467	2204	2222
IBM07	824	824	2036	2033	3098	3113
IBM08	1140	1140	2241	2266	3240	3217

three cooperation operators. The effectiveness of the algorithm depends on the specification and implementation of these operators. They control the coarsening which impacts the solution subspaces explored at the different levels. We have been conservative in exploiting this aspect of the cooperation strategy. Improvements both in terms of computational time and quality of partitionings will result from the choice of elite solutions (those selected at each level for information sharing), the choice of operators for refinement, and the selection of the levels between which cooperation takes place.

Our cooperative search paradigm can be applied to create partitioning methods capable of partitioning hypergraphs with fixed vertices, which could enhance the usefulness of this paradigm in VLSI design. The refinement phase of CoMHP is flexible, and can adapt to local constraints imposed on coarsening by specific needs from the physical design process.

In the case of CoMHP, each iteration of the slowest process executes hMETIS, FM and FMS as subroutines. It is therefore not surprising that CoMHP takes considerably longer time than any of its subroutines. On the other hand, our work supports the hypothesis that individual search algorithms, with the same amount of computing resources as the cooperative computation (through restarts or other means), cannot match the performance of a successful cooperative algorithm. Based on the results presented in this paper, we believe that multilevel design provides such a successful approach to develop cooperation protocols. The cooperative multilevel search paradigm in combination with other heuristics will help produce solutions with better quality than those obtained by the original heuristics. This paradigm will also be useful to design algorithms for other combinatorial optimization problems (besides partitioning) arising in the VLSI CAD area. Our work in this paper is the first study to demonstrate this.

Acknowledgement

We wish to thank Charles Alpert and anonymous referees for their helpful comments.

REFERENCES

[1] C.J. Alpert, J.-H. Huang, and A. B. Kahng. Multilevel Circuit Partitioning. In *Proc. 34th ACM/IEEE Design Automation Conference*, pages 530–533, June 1997.

[2] C.J. Alpert and A.B. Kahng. Recent Developments in Netlist Partitioning: A Survey. *Integration: the VLSI Journal*, 19:1–81, 1995.

[3] S.H. Clearwater, B.A. Huberman, and T. Hogg. Cooperative Solution of Constraint Satisfaction Problems. *Science*, 254:1181–1183, 1991.

[4] J. Cong and M.L. Smith. A Parallel Bottom-Up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design. In *Proc. 30th ACM/IEEE Design Automation Conference*, pages 755–760, June 1993.

[5] A. Dasdan and C. Aykanat. Two Novel Circuit Partitioning Algorithms Using Relaxed Locking. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 16(2):169–78, Feb. 1997.

[6] M.R. Garey and D.S. Johnson. *Computer and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.

[7] A. Gupta. Fast and Effective Algorithms for Graph Partitioning and Sparse Matrix Ordering. Report RC 20496, IBM T.J. Watson Research Center, 1995.

[8] T. Hogg and C. Williams. Solving the Really Hard Problems with Cooperative Search. In *Proceedings of the 11th National Conference on Artificial intelligence (AAAI93)*, pages 231–236. AAAI Press, Aug. 1993.

[9] J.J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79:2554–2558, 1982.

[10] B.A. Huberman. The Performance of Cooperative Processes. *Physica D*, 42:38–47, 1990.

[11] G. Karypis, V. Aggarwal, V. Kumar, and S. Shekhar. Multilevel Hypergraph Partitioning: Application in VLSI Domain. *IEEE Transactions on VLSI Systems*, 07(01):69–79, Mar. 1999.

[12] G. Karypis and V. Kumar. Multilevel k-way Hypergraph Partitioning. In *Proc. 36th ACM/IEEE Design Automation Conference*, pages 343–348. Association for Computing Machinery, June 1999.

[13] K-G. Lee and S-Y. Lee. Efficient Parallelization of Simulated Annealing using Multiple Markov Chains: An Application to Graph Partitioning. In *Proc. 1992 of the Int. Conf. on Parallel Processing*, pages III 177–180. CRC Press, Aug. 1992.

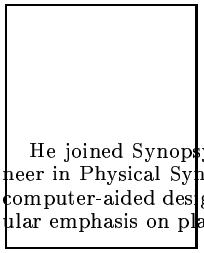
[14] D. Levine. A Parallel Genetic Algorithm for the Set Partitioning Problem. In *Meta-Heuristics: Theory and Applications*, pages 23–35. Kluwer Academic Publishers, 1996.

[15] L.A. Sanchis. Multiple-way Network Partitioning. *IEEE Trans. Comput.*, 38(1):62–81, Jan. 1989.

[16] V. Schnecke and O. Vornberger. An Adaptive Parallel Genetic Algorithm for VLSI-Layout Optimization. In *Proceedings of the Fourth Workshop on Parallel Problem Solving from Nature*, pages 859–868. Springer- Verlag, 1996.

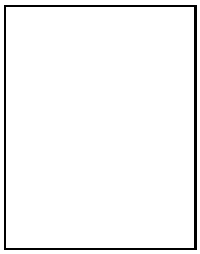
[17] M. Toulouse, T.G. Crainic, and B. Sansó. Self-Organization in Cooperative Tabu Search Algorithms. In *1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2379–2385. Omnipress, Oct. 1998.

[18] M. Toulouse, K. Thulasiram, and F. Glover. Multi-Level Cooperative Search. In *5th International Euro-Par Parallel Processing Conference, volume 1685 of Lecture notes in Computer Science*, pages 533–542. Springer-Verlag, Aug. 1999.

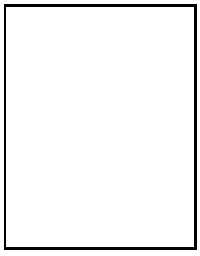


Min Ouyang received a B.S. in Computer Science from Fudan University, China, in 1990, a M.S. in Computer Science from Peking University, China in 1996, and a Ph.D. in Computer Science from the University of Nebraska-Lincoln, Nebraska, U.S.A. in 2000.

He joined Synopsys Inc. in 2000, where he was a Sr. R&D Engineer in Physical Synthesis Group. His research interest includes the computer-aided design of integrated circuit and systems, with particular emphasis on placement algorithms.



Michel Toulouse received a MSc in computer science from Université de Montréal in 1989 and a PhD in computer science & engineering from École Polytechnique, Université de Montréal in 1996. He is currently assistant professor in the Department of Computer Science at the University of Manitoba, Canada. His research interests include metaheuristic algorithms, parallel & distributed algorithms, multilevel algorithms with applications in VLSI CAD area and non-standard computing paradigms (emergent computation, molecular computation, and quantum computing). He is an Associate Editor of the Journal of Heuristic and co-author of a book on Parallel Metaheuristics.

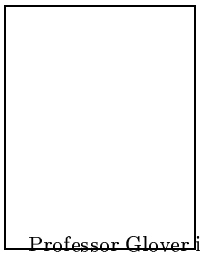


Krishnaiyan Thulasiraman holds the Hitachi Chair and is Professor in the School of Computer Science at the University of Oklahoma, Norman, where he has been since 1994. He received his Ph.D. in EE from the IIT , Madras, India in 1968. Prior to joining the University of Oklahoma, Dr. Thulasiraman was professor (1981-1994) and chair (1993-1994) of the ECE Department in Concordia University, Montreal. He was on the faculty in the EE and CS departments of the IIT during 1965-1981 .

Dr. Thulasiramans research interests have been in graph theory, combinatorial optimization, and algorithms and applications in a variety of areas in CS and EE. He has published more than 100 papers in archival journals, coauthored with M.N.S.Swamy two text books Graphs, Networks, and Algorithms (1981) and Graphs: Theory and Algorithms(1992), both published by Wiley Inter-Science, and authored two chapters in the Handbook of Circuits and Filters (CRC and IEEE, 1995) and recently edited the Circuit Theory section of a forthcoming Encyclopedia.

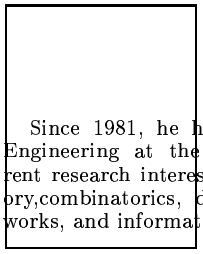
Dr. Thulasiraman has received several awards and honors: IEEE CAS Society Golden Jubilee Medal (1999), Fellow of the IEEE (1990), Senior Research Fellowship of the Japan Society for Promotion of Science (1988), and Guest Professorship of the German National Science Foundation (1990). Dr.Thulasiraman has held visiting positions at the Tokyo Institute of Technology ,University of Karlsruhe, University of Illinois at Urbana-Champaign and Chuo University , Tokyo.

Dr. Thulasiraman has been Vice President (Administration) of the IEEE CAS Society (1998, 1999), Technical Program Chair of ISCAS (1993, 1999), Co-Guest Editor of a special issue on "Computational Graph Theory: Algorithms and Applications" (IEEE Transactions on CAS , March 1988), Associate Editor of the IEEE Transactions on CAS (1989-91, 1999-2001), and Founding Regional Editor of the Journal of Circuits, Systems, and Computers. Recently, he founded the Technical Committee on Graph theory and Computing of the IEEE CAS Society.



Fred Glover is the MediaOne Chaired Professor in Systems Science at the University of Colorado, Boulder, and Distinguished Researcher and co-founder of the Hearin Center for Enterprise Science at the University of Mississippi. He has authored or co-authored more than three hundred published articles and five books in the fields of mathematical optimization, computer science and artificial intelligence.

Professor Glover is the recipient of the distinguished von Neumann Theory Prize, as well as of numerous other awards and honorary fellowships, including those from AAAS, NATO, INFORMS, DSI, CAS, DCA, ERI, AACSB and a variety of others. He serves on the advisory boards of several organizations and is a co-founder of Heuristic, Inc. and OptTek Systems, Inc.



Jitender S. Deogun received his B.S.(Hons.) degree from Punjab University, Chandigarh, India, in 1967, and M.S. and Ph.D. degrees from University of Illinois, Urbana, in 1974 and 1979, respectively.

Since 1981, he has been a Professor of Computer Science and Engineering at the University of Nebraska, Lincoln. His Current research interests include strutural and algorithmic graph theory,combinatorics, design and analysis of algorithms, optical networks, and information retrieval.