Multilevel Cooperative Search for the Circuit/Hypergraph Partitioning Problem

Min Ouyang
Department of CS&E, University of Nebraska-Lincoln, mouyang@cse.unl.edu

Michel Toulouse
Department of Computer Science, University of Manitoba, toulouse@cs.umanitoba.ca

Krishnaiyan Thulasiraman
School of Computer Science, University of Oklahoma, thulasi@cs.ou.edu

Fred Glover
Hearin Center for Entreprise Science, University of Mississippi, fglover@bus.olemiss.edu

Jitender S. Deogun
Department of CS&E, University of Nebraska-Lincoln, deogun@cse.unl.edu

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 [20], 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 hMETIS. Bisections of hypergraphs based on our algorithm also outperform hMETIS, although more modestly. We suggest some possible improvements in the implementation of the cooperation strategy. The experimental results demonstrate the effectiveness of the cooperative multilevel search paradigm for solving the netlist partitioning problem. Our outcomes disclose 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.

Submitted to *IEEE Transactions on Computer Aided Design*

1. INTRODUCTION

Netlist partitioning is an important and well-studied research area in VLSI CAD. Several classes of heuristics have been proposed to address this problem [3]. Recently, multilevel algorithms have been applied to the netlist partitioning problem [13]. This approach has since become the standard to partition netlists.

The multilevel paradigm in the context of netlist partitioning enables Fiduccia-Mattheyses (FM) types of move-based heuristics to execute moves involving static clusters (blocks) of modules in the netlist (usually a move only involves one or two modules). This strategy of multilevel algorithms is a variant of k-exchange [1], a well-known technique to create variations in the neighborhood structure of local search algorithms. Usually, applications of k-exchange to define the neighborhood structures do not change the optimization problem but they have an impact on local optimality and consequently on the local search problem solved by the move-based heuristics. The multilevel variant of k-exchange is a little more drastic. Besides changing the neighborhood structures, it also reduces the size of the solution space through the coarsening of netlists. This coarsening constitutes a relaxation of the optimization problem which enables to achieve gains in computational speed. But coarsened netlists are static. Consequently multilevel algorithms can only swap fixed blocks of modules. This constraint may strongly impair FM and other move-based heuristics by restricting their operations to work with a limited set of blocks which may be potentially flawed. This 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 [9; 13] using more dynamic coarsening strategies. In the present paper we provide a broader design strategy to address this problem.

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 an independent program that runs in a time sharing manner with the other programs on a sequential computer or in parallel if several computers 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 a same algorithm. 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. Cooperation, as framed in Huberman's paper [12], is an exchange of "hints" that may confuse some processes, but will also help others. Over all, hint sharing improves the performance because we only care about the best performers and some of them do even better with hints. This approach has been used with success to design search heuristics in the context of constraint satisfaction problems [4; 10] and to parallelize some metaheuristics [15; 16; 18; 19]. More recently, in [20], we have applied cooperative search to graph partitioning quite successfully.

The present paper introduces the Cooperative Multilevel Hypergraph Partitioning algorithm (CoMHP), an asynchronous variation for hypergraph partitioning of the cooperative algorithm in [20]. 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. We motivate this coarsening approach and analyze its complexity. The main contribution of this paper is a parallel design of multilevel algorithms based on the cooperative search paradigm. We describe how to obtain independent searches out of clusters in coarsened hypergraphs combined with the same FM heuristic. Next, once H_0, the hypergraph representation of a given netlist, has been coarsened into hypergraphs H_1, H_2, ..., H_l, each hypergraph becomes the input of one independent move-based search process. The $l+1$ search processes are run concurrently. The cooperation protocol, which is the main ingredient of cooperative search, uses three different forms of strategies for interactions among the processes. Let $p_0, \ldots, p_l$ be the search processes (process p_i has coarsened hypergraph H_i as input). In the first strategy, each process p_i obtains partitionings from coarsened hypergraph H_{i+1} and uses these partitionings as initial solutions to the move-based heuristics that constitute the search algorithm of process p_i. (This operator is similar to the interpolation operator of multilevel partitioning algorithms.) The two other interaction strategies borrow partitionings from H_{i-1} to modify the coarsening of hypergraph H_i, either by splitting vertices of H_i or by aggregating vertices of H_{i-1}, to yield a new coarsening of the vertices of H_0 in H_i. We call these last two strategies the *local partitioning operator* and the *local clustering operator*, respectively. The execution of the three strategies is triggered by the internal state of the processes. Therefore, as a whole, the computation is asynchronous.

The rest of the paper is structured as follows. Section 2 describes the standard multilevel approach to partitioning, identifying the main weakness of this approach and discussing how our cooperative search design addresses the problem. Section 3 describes our algorithm and its implementation. Section 4 reports and discusses the results of the tests conducted on the ISPD98 benchmark suite of circuits. Finally, Section 5 concludes with some suggestions for future work.

2. THE MULTILEVEL ALGORITHM

Hypergraphs are commonly used by multilevel algorithms 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. Given this formalization, the problem of partitioning the modules of a netlist into κ subsets $P_1, P_2, \ldots, P_\kappa$ can be stated as a combinatorial optimization problem where one tries to find an instance $\{P_1, P_2, \ldots, P_\kappa\}$ of the mapping

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

The goal is to minimize 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. $\frac{|V_0|}{c\kappa} \leq |P_i| \leq \frac{c|V_0|}{\kappa}$ for some constant $c \geq 1.0$;

3. $\bigcup_{i=1}^{\kappa} P_i = V_0$.

Constraint (1) indicates that module replications are not allowed and constraint (2) sets bounds on the cardinality of the subsets P_i in the partitions.

Multilevel algorithms initiate processing by a sequence of l different relaxations of the optimization problem (2). These relaxations are obtained by mapping the flattened hypergraph H_0 into l hypergraphs with fewer vertices. This phase of the processing is identified as the coarsening phase, and is usually based on hierarchical clustering strategies applied on H_0. We define hierarchical clustering in the following manner:

DEFINITION 1. *A hierarchical clustering algorithm is a family of l mappings:*

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

where each mapping $\mathcal{C}_i$ defines a mapping instance of the vertices of H_0 into $|V_i|$ clusters $C_{i1}, C_{i2}, \ldots, C_{i|V_i|}$ where $C_{iu} \cap C_{iv} = \emptyset$ if $u \neq v$ and $\cup_{j=1}^{|V_i|} C_{ij} = V_0$. Furthermore, $|V_i| > |V_{i+1}|$, for all $i = 1, 2, \ldots, l-1$.

DEFINITION 2. *A coarsened hypergraph $H_i = \{V_i, E_i\}$ is a set of vertices V_i and hyperedges E_i such that i) $v \in V_i$ 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 ii) 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 $\{a, b\} \subseteq e$ for some hyperedge $e \in E_0$.*

A hierarchical clustering algorithm generates a sequence of increasingly coarsened hypergraphs where H_{i+1} is more coarsened than H_i. In practice, many hierarchical clustering algorithms generate the mapping of the vertices of H_0 based on some characteristics of the coarsened hypergraph H_{i-1} rather than directly from the hypergraph H_0. For example, in *recursive maximal matching*, vertices of H_0 are mapped into clusters of $\mathcal{C}_i$ by merging randomly pairs of vertices of the coarsened hypergraph H_{i-1}. For standard multilevel algorithms, coarsening is followed by a partitioning of the most coarsened hypergraph and by the refinement of this initial partitioning using the other less coarsened hypergraphs. During these last two phases of multilevel algorithms, partitionings are obtained and refined by FM-like heuristics in a sequence going from H_l to H_0.

FM heuristics are exchange-based (move-based) heuristics. Typically, one iteration of a FM heuristic yields a new partitioning from the current partitioning by exchanging a vertex $a \in P_i$ with a vertex $b \in P_j$ where P_i, P_j are two different subsets of the current partitioning. However, following Definition 2, a single FM exchange in a coarsened hypergraph H_i involves the exchange of clusters of vertices in V_0. Exchanges executed by FM heuristics in coarsened hypergraphs correspond to k-exchange moves in hypergraph H_0, i.e., the exchange of clusters of vertices in H_0.

k-exchange moves are a key ingredient in the design of heuristics and metaheuristics for discrete optimization problems. The k-exchange moves allow, for example, quick surveys of the solution space. An important feature of multilevel k-exchange moves is that they can only exchange fixed clusters of vertices of H_0 based on the coarsening of hypergraphs. Only a very small subset of the possible combinations of vertices in H_0 are represented in the l different coarsenings. This situation restricts the search performed by each k-exchange FM heuristic to limited regions of the solution space of Problem (2). Furthermore, randomized coarsening strategies drive out, by definition, low and high cost partitionings. (If the randomized coarsening is based on a uniform distribution, low and high cost partitionings have the same probabilities of being eliminated as any other partitionings.) This property of randomized coarsening has a leveling action on the solution space of coarsened hypergraphs that can obliterate optimal valleys from the landscape of the cost function for those coarsened hypergraphs. FM heuristics are gradient heuristics: if the valleys of the optimal solutions have been removed by coarsening, then the initial partitioning and refinement in the most coarsened hypergraphs will be of no use. Some other coarsening strategies (for example, edge-weight based coarsenings) try to populate the coarsened hypergraphs with "better than average" partitionings. This approach however has a tendency to favor some of the better than average partitionings at the expense of others, reducing the diversity of the partitionings represented in the coarsened hypergraphs. This could lead to a poor exploration of the solution space of Problem (2) because FM heuristics are trapped in specific regions of the solution space.

These problems have been partially identified in [9; 13] by observing that an initial partitioning can be refined in different ways depending upon how the coarsening is executed [13]. A multi-phase refinement has also been proposed to improve the search. This multi-phase refinement is based on a recursive call of the multilevel algorithm from the same coarsened hypergraph. A randomized coarsening is used for the multi-phase refinement. In [13] this coarsening is initiated from the best partitioning obtained from the refinement or a previous iteration of the multi-phase refinement. The multi-phase refinement iterates until the best solution cannot be improved further. The refinement of the multilevel procedure is then resumed.

Our algorithm is similar in spirit to multi-phase refinement because it also produces many "re-coarsenings" of the hypergraph. However it is implemented in the context of the cooperative paradigm. The characteristics of our approach are the following:

- Unlike multi-phase refinement, new coarsenings are created at all the levels in parallel.
- We use a new operator to obtain new coarsenings (the partitioning operator).
- The cooperative approach is more diversified given that the solution space is explored concurrently and information sharing is local. For example, in some implementations of multi-phase refinement, the same partitioning is used to generate all the new coarsenings in one multi-phase cycle. In the cooperative approach, a partitioning determines the coarsening of only those neighbor hypergraphs in the hierarchical structure.
- Standard multilevel implementations, including multi-phase approaches, are at the mercy of a bad start (poor initial partitioning). Restarts are used to reduce the influence of bad starts. Cooperative multilevel approach does not apply "independent" restarts; rather it has several processes cooperating with one another.

The next section describes the details of CoMHP and its implementation.

3. COMHP: DESCRIPTION AND IMPLEMENTATION

3.1 CoMHP Coarsening Phase

Usually recursive *matching-based clustering* algorithms such as *edge-coarsening*, *hyperedge-coarsening*, *maximal matching* and *modified-hypergraph-coarsening* [5; 13] are used to reduce (coarsen) the size of the netlist. In our approach to coarsening, we generate coarsened hypergraph H_i directly from the netlist, not from the hypergraph H_{i-1}. We address the problem of contracting the netlist as a partitioning problem. To obtain a hierarchy of coarsened hypergraphs as defined in (3), we solve the partitioning problem (2) for $\kappa = \kappa_i = \frac{n}{2^i}, 0 < i \leq l$. More specifically, we seek a mapping instance $\mathcal{C}_i = \{C_{i1}, C_{i2}, \ldots, C_{i\frac{n}{2^i}}\}$ that minimizes the cost function (2), where 2^i is the size of the clusters at level i and $\frac{n}{2^i}$ is the number of clusters. We call this hierarchical coarsening strategy as *partition-based coarsening*. The coarsening phase of our algorithm uses a $\frac{n}{2}$-way partitioning to get H_1, $\frac{n}{2^2}$-way partitioning to get H_2, and so on.

The partition-based coarsening strategy is a compromise between maximal matching that might obliterate too many good solutions and a coarsening phase biased by the edge-weights. By having a coarsening phase that is biased by several cost functions, we hope to obtain diversified sets of clusters of vertices from V_0 in the different coarsened hypergraphs. This is not however a very good compromise in terms of run time and space efficiency. We need to run a partitioning method to generate each of the l coarsened hypergraphs of CoMHP. The sequential time complexity of a partitioning method generally depends on two inputs: the size of the hypergraph and the number κ of subsets in the partitioning. The worst case for CoMHP occurs when $\kappa \approx \frac{n}{2}$, the number of vertices of H_1, the least coarsened hypergraph of CoMHP. The sequential time complexity of the coarsening phase for CoMHP is then given by the sum of the sequential times of the l partitioning processes.

The coarsening phase of CoMHP can be easily parallelized. One only needs to run each partitioning process on a different processor. The parallel time complexity is then dominated by the processor that computes the partitioning for $\kappa \approx \frac{n}{2}$. The space complexity of sequential CoMHP is $O(2 \times (V_0 + E_0))$, which is the space needed to load H_0 and to store H_1 to H_l. The space complexity of parallel CoMHP is $O((l+1) \times (V_0 + E_0))$ on SMP computers; H_0 is loaded and shared by all the processes while space is required to store the coarsened hypergraphs generated by each multilevel partitioning process. If non-multilevel partitioning methods are used, then the space complexity of parallel CoMHP is the same as for sequential CoMHP. Other options are available to implement our model of the coarsening phase. We can ask the partitioning method to compute a κ-way partitioning where $\kappa = \frac{n}{2}$, the number of vertices in H_1. Hypergraph H_1 corresponds to the $\log_2 n - 1$th bisection of the partitioning method. For the other hypergraphs $H_2, H_3, \ldots, H_l$, we use respectively the partitionings of the bisections $\log_2 n - i$, $1 < i \leq l$. Assuming the partitioning method is not parallelized, this needs about the same computational time as our current parallel coarsening implementation but uses substantially less space. We have used this approach for the version of our algorithm applied to graph partitioning [20].

The way we compute the coarsened hypergraphs directly from H_0 has some impact on the refinement phase. When the mapping of vertices of H_0 to clusters of H_i is based on a recursive coarsening algorithm, a vertex $C_{ij} \in H_i$ is usually constituted from the aggregation of two vertices of H_{i-1}. Therefore $C_{ij} \in H_i = C_{(i-1)p} \cup C_{(i-1)q}$ for some vertices $C_{(i-1)p}, C_{(i-1)q} \in H_{i-1}$. So if $v \in H_0$ is mapped to $C_{(i-1)j} \in H_{i-1}$, then v is automatically mapped to C_{ij}, the superset of $C_{(i-1)j}$ in the coarsened hypergraph H_i. Therefore hypergraphs are considered as related level by level. This is not necessarily the case when using our coarsening strategy. The vertices of H_0 that are mapped to a cluster $C_{ij} \in H_i$ can be spread among several clusters in each hypergraph H_j, $j > i$. For example, in Section 3.2.3, the design of our interpolation operator reflects the fact that the coarsened hypergraphs in CoMHP are not related level by level.

Whether we use a direct or a bisection κ-way partitioning algorithm for our coarsening phase, the solution spaces are biased by the cost function of the partitioning problem solved at each coarsened hypergraph. We have done some preliminary tests with coarsening strategies not related to our partition-based coarsening approach. We always found better partitionings during the initial partitioning phase as well as during the refinement phase when hypergraphs are clustered by a partition-based coarsening strategy (compared to matching-based coarsening strategies). Partition-based coarsening appears to generate coarsened hypergraphs which have fewer hyperedges spanning several vertices compared to those obtained by matching-based coarsening. In this case, the average of the hyperedge-cuts of the solution spaces should not be constant and should be lower compared to a matching-based coarsening. Most likely, the partitioning phase will be initiated from solution spaces with several local optima. For a standard multilevel algorithm, having fewer hyperedges can be a handicap that induces the method to become trapped during the uncoarsening phase in the local optima of the solution spaces.

3.2 CoMHP Refinement Phase

The coarsening phase defines l new k-exchange FM heuristics. We can then said that the family of mappings in Definition 1 together with the standard FM heuristic constitutes the set of independent search algorithms in the cooperative scheme. Each algorithm (process) p_i takes as input the coarsened hypergraph H_i from which an initial partitioning is computed by each process. We now introduce the cooperation protocol that transforms this set of independent algorithms into cooperating processes.

For standard multilevel algorithms, the sharing of information among hypergraphs is limited to one best partitioning. The design of CoMHP is based on a set of "elite" partitionings, generalizing the scheme for standard multilevel algorithms. The elite partitionings of a hypergraph H_i are collected in $X'_i \subset X_0$ (X_0 is the solution space of H_0) such that the average of the hyperedge-cuts of the partitionings in X'_i is better than the average of the hyperedge-cuts of all the partitionings identified in H_i by the move-based heuristics.

Formally, a set of elite partitionings X'_i is defined by:

$$\frac{\sum_{x \in X'_i} f(x)}{|X'_i|} < c(\frac{\sum_{x \in \bar{X}_i} f(x)}{|\bar{X}_i|}) \quad (4)$$

for some real constant $c < 1$, and where $\bar{X}_i$ is the set of partitions visited by search heuristics at level i.

Usually, multilevel algorithms use a clustering operator where vertices of hypergraph H_i are obtained by aggregating vertices from H_{i-1}. The CoMHP refinement phase uses an operator that we call *local clustering*. But there is second operator that splits vertices of H_i in order to get a new coarsening. We call this operator *local partitioning*.

3.2.1 Local partitioning operator

Local partitioning changes the coarsening of H_i by splitting *some* of its vertices. Vertices (clusters) in H_i are destroyed based on a 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 Figure 1) such that v has at least two vertices $a, b \in V_0$ that are into two different subsets (line 4) of at least one of the elite partitionings of X'_{i-1} (line 2). 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). Following the execution of local_partitioning, a new coarsened hypergraph H_i is generated that reflects the changes in the set of vertices of H_i.

```
Local_partitioning( );
1. for (q = 1; q ≤ |V_i|; q++)
       v = qth vertex of V_i; stop = false; j = 1;
2.     for (r = 1; r ≤ |X'_{i-1}|; r++)
           if (not stop) then
3.           for (s = 1; s ≤ κ; s++)
4.               if ((v ∩ P_s ≠ ∅) & (v ⊄ P_s)) then
5.                   create new vertex v_j = v ∩ P_s; j++;
               if (j > 1) then
                   stop = true; mark v to destroy;
```

Figure 1: Pseudo-code of the local partitioning operator

In terms of implementation, we have limited the set X'_{i-1} to only two partitionings from X_{i-1} each time local partitioning is executed. A future work will test strategies for selecting elite solutions, drawing on guidelines proposed in Glover and Laguna [8] and Glover [7].

3.2.2 Local clustering operator

Local clustering changes the coarsening of H_i by aggregating *some* of the vertices (clusters) of H_{i-1} into new vertices for H_i. Vertices from H_{i-1} are selected for aggregation based on the fact that they are in the same subsets of each of the elite partitionings in X'_{i-1} (line 5 in Figure 2). That is, local clustering looks for a vertex $v \in V_{i-1}$ (line 1) which is in the same subset P_s (lines 2 and 4) for all elite partitionings of X'_{i-1} (lines 3 and 4).

Once the vertices of H_{i-1} that are candidates for merging have been identified, pairs of vertices are merged according to the two following criteria: 1) the two vertices are in the same subset s for all elite partitionings from X'_{i-1}; 2) the two vertices lie on the same hyperedge. (The merging of vertices is not shown in Figure 2.)

Local clustering tends to reduce the number of vertices in a coarsened hypergraph. This balances the effect of the local partitioning operator which tends to increase the number of vertices. Empirically we have verified that the number of

```
Local_clustering( );
1. for (q = 1; q ≤ |V_{i-1}|; q++)
       v = qth vertex of V_{i-1}; stop = false; j = 1;
2.     for (s = 1; s ≤ κ; s++)
           if (not stop) then
3.           for (r = 1; r ≤ |X'_{i-1}|; r++)
4.               if (v ⊂ P_s) then
                     j++;
5.           if (j == |X'_{i-1}|) then
                 stop = true; mark v as part of subset s;
```

Figure 2: Pseudo-code of the local clustering operator

vertices tends to be relatively constant in the coarsened hypergraph although we do not do anything specific to keep this number constant. In fact we hypothesize that this is related to the way we define the set of elite partitionings. For example, a less stringent definition of eliteness could make it nearly impossible to find vertices to cluster, while a more stringent definition might allow too many vertices to be clustered. It seems likely in some settings that a fairly restrictive definition of eliteness may be needed to allow vertices to be clustered, but then too many will be clustered. To avoid this, the definition of eliteness can be relaxed, and then strengthened indirectly by stipulating that only m-element subsets of the full set qualify as an 'elite set'.

3.2.3 Interpolation Operator

The interpolation operator chooses one partitioning in the set of elite partitionings of H_{i+1} as an initial solution of a move-based heuristic in hypergraph H_i. Let x be this partitioning of hypergraph H_{i+1}. Because of the coarsening strategy used in CoMHP, it is possible that vertices in H_i will overlap several subsets of partitioning x. A split of these vertices in H_i is performed using a similar procedure as for the partitioning operator. Following the split, a FM search is applied to hypergraph H_i using an initial partitioning that reflects the partitioning obtained from H_{i+1}.

3.2.4 The main loop of CoMHP's processes

CoMHP uses elite partitionings in three different ways during the refinement phase based on three different operators: the *local partitioning*, *local clustering* and *interpolation* operators. Each iteration of a process p_i executes the following outer loop: local partitioning, interpolation, local clustering and global search. Special conditions hold for processes p_0 and p_l. Figure 3 give the details this outer loop:

```
CoMHP( ); /* process p_i */
Compute an initial partitioning; /* initial partitioning phase of
                                    standard multilevel algorithm */
While not terminated { /* begin outer loop */
1.   Apply local partitioning to H_i
1a.  Execute FMS and PFM on new H_i;
2.   Apply interpolation to current H_i
2a.  Execute FMS and PFM using good partitionings of H_{i+1};
3.   Apply local clustering to current H_i
3a.  Execute FMS and PFM on new H_i;
4.   GlobalSearch( ) {
       If number of vertices < 500
         do random search;
       else execute hMETIS; }
     Save p_i's good local and global partitioning results;
} /* end outer loop */
End CoMHP
```

Figure 3: Main loop of CoMHP

Each process of CoMHP applies two different move-based heuristics to perform local searches and two global search algorithms. Local search algorithms start a search based on an existing partitioning, while global search algorithms first generate a partitioning and then perform a local search. The local move-based heuristics are the Sanchis algorithm (FMS) [17], and the multiway partitioning by free moves (PFM) proposed by Dasdan and Aykanat [6]. For global search we use a random search algorithm, where an initial partitioning is generated randomly, followed by the execution of a local search to refine this partitioning. The random search algorithm is used for higher levels, for coarsened hypergraphs having less than 500 vertices. We use the multilevel m-way hypergraph partitioning algorithm [14] as another global search algorithm in CoMHP.

Each time a coarsening is modified by one of the three interaction operators, it changes one of the mappings in the family described in Definition 1. A mapping instance $\mathcal{C}_i = \{C_{i1}, C_{i2}, \ldots, C_{i|V_i}\}$ is an array of integers of size $|V_0|$ that maps each vertex $v \in V_0$ into a cluster $C_{ij} \in \mathcal{C}_i$. Partitionings are also defined in the same manner using mapping vectors. The total space requirement for those vectors is $O(2 \times l \times |V_0|)$. Each change to a coarsening or a partitioning forces the update of a mapping vector. The time requirement for the three operators is dominated by the number of vertices in H_0. For example, to execute one iteration of local partitioning at level i, we need to perform one sweep across the vertices of H_0 using the mapping instance $\mathcal{C}_i$ and the partitionings of X'_{i-1} to obtain the set of vertices to destroy. Then a second sweep across the vertices of H_0 is performed to get a new mapping instance $\mathcal{C}_i$ and the new coarsened hypergraph H_i needed by the local and global partitioning methods. Therefore the time requirement for the cooperation protocol is $O(|V_0|)$ for each execution of one of the operators. The time requirement of the cooperation protocol is the same at all levels. Furthermore this time requirement is insignificant compared to the time required by the FM-based search heuristics. For example, the time needed by level 1 to execute one iteration of the outer loop compared with the time taken by the highest level in our tests (level 10) is about $\frac{25}{1}$. That is, level 10 executes about 25 iterations of the outer loop while level 1 executes one.

3.2.5 On the Dynamics of CoMHP

CoMHP is a cooperative search algorithm. The requirements for cooperative algorithms are the following:

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.

Experimental and analytical models of cooperative search [12] show that the performance of cooperating processes is log-normally distributed, contrasting with the normal distribution of independent searches (or restarts). The impact of cooperation on the distribution is a smaller number of average quality searches but with an increases 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.

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 exploration of the different searches overlaps 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. Multilevel algorithms first attracted our attention because of the constraints imposed by the coarsening phase on k-exchange moves. These constraints could lead to poor exploration of the solution space in standard multilevel algorithms. In a system of cooperating search processes, those constraints help to provide conditions similar to what is achieved by the statistical independence of hints. The static nature of k-exchange moves executed by FM heuristics in coarsened hypergraphs slows down the convergence of cooperating processes and diffuses the impact on convergence that can have good but sub-optimal hints provided an operator such as interpolation. On the other hand, operators such as local partitioning and local clustering allow for movement in the solution space by cooperating processes. Those are basically the favorable conditions that a large population of non-redundant searches will bring to the dynamics of a system of cooperating processes.

The global control structure (the algorithm) induced by the cooperation protocol of CoMHP is the minimization of a very simple energy function (similar to the energy function of Hopfield networks [11]). As for multi-phase refinements, the refinement phase of CoMHP ends once there are no improvements in the quality of the best solutions. In the case of CoMHP, the energy function is approximated by the sum of the differences between the averages for the hyperedge-cuts $f(x)$ of the elite partitionings x of the different processes:

$$E(\mathcal{X}) = \sum_{i=0}^{l-1} \{ \frac{\sum_{x \in X'_i} f(x)}{|X'_i|} - \frac{\sum_{x \in X'_{i+1}} f(x)}{|X'_{i+1}|} \} \qquad (5)$$

At its initial state, the elite partitionings of neighboring processes have different hyperedge-cuts. This creates energy to change the coarsening of the neighboring hypergraphs. New coarsenings provide new elite partitionings which in turn affect the coarsening of neighboring hypergraphs. This dynamics ends when all the elite partitionings have about the same hyperedge-cuts, yielding the minimum energy level of the system. Once the system has reached a minimum energy, the cooperation protocol has a relatively limited impact on the search process (only the interpolation operator is still active). It is at this point that the three local operators cease to have any significant impact on the exploration of the solution space.

The number of processes in CoMHP is small and the hints are not statistically independent. There is quite a high degree of correlations among the searches and among the hints exchanged. Consequently, the performance of cooperating processes in CoMHP is not log-normally distributed, rather all the processes converge toward about the same quality of solutions at the minimum energy level of the system. This is not unusual for small systems of cooperating processes. However the next section on experimental results shows that

the performance of CoMHP is consistently better than independent (restart) searches. Actually this situation is considered to be the best scenario of cooperative search: an advance in the performance from all the cooperating processes (as opposed to an advance for only a few best performers).

4. EXPERIMENTAL RESULTS

We have evaluated the performance of our CoMHP algorithm on the ISPD98 benchmark suite of netlists [2], 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) [14]. 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 1 and 2 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 for 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 1 and 2 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 3 and 4 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.

In Tables 3 and 4, 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. For example, line 1a can be executed in parallel with line 2, 2a, 3, 3a, and 4. Similarly, line 2a can be executed in parallel with line 1a, 3, 3a and 4, etc. 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 (5) 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

Table 1: 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 2: 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

Table 3: 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 4: 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

we can get similar results as tables 1 and 2 with only about 1/5 to 1/3 run time as shown in tables 3 and 4. 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 this matter, given that CoMHP uses repeatedly those partitioners as subroutines. On the other hand, with the same amount of computing resources as CoMHP (when runs for 10 iterations of p_0), hMETIS didn't improve noticeably the quality of partitionings reported in Tables 1 and 2. We have done similar comparisons in our work on graph partitioning [20], testing several partitioners, with the same conclusions as for the current work.

5. 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 [13] 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 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.

It is expected that a cooperative search algorithm will requested more computational resources than the individual search algorithms cooperating with one another using a cooperation protocol. In the case of CoMHP, each iteration of the slowest process executes hMETIS, FM and FMS as subroutines. It is then 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 the multilevel design provides such a successful approach to develop cooperation protocols. The cooperative multilevel search paradigm in combination with other heuristic 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 the VLSI CAD area. Our work in this paper is the first study to demonstrate this.

6. REFERENCES

[1] E.H.L. Aarts and J.K. Lenstra. Introduction. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Oprimization*, pages 1–17. John Wiley & Sons Inc., 1997.

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

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

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

[5] 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, 1993.

[6] 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.

[7] F. Glover. Scatter Search and Path Relinking. In D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization*, pages 297–316. McGraw-Hill, 1999.

[8] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

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

[10] 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, 1993.

[11] 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.

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

[13] G. Karypis, V. Aggarwal, V. Kumar, and S. Shekhar. Multilevel Hypergraph Partitioning: Application in VLSI Domain. *IEEE Transactions on VLSI Systems*, 1998.

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

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

[16] D. Levine. A Parallel Genetic Algorithm for the Set Partitioning Problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 23–35. Kluwer Academic Publishers, 1996.

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

[18] V. Schnecke and O. Vornberger. An Adaptive Parallel Genetic Algorithm for VLSI-Layout Optimization. In H.-P. Schwefel Y. Davidor and R. Männer, editors, *Proceedings of the Fourth Workshop on Parallel Problem Solving from Nature*, pages 859–868. Springer- Verlag, 1996.

[19] 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, 1998.

[20] 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, 1999.