

Multilevel Cooperative Search: Application to the Circuit/Hypergraph Partitioning Problem

Min Ouyang¹ Michel Toulouse² Krishnaiyan Thulasiraman³
Fred Glover⁴ Jitender S. Deogun⁵

¹ U. Nebraska-Lincoln, Dept CS&E, mouyang@cse.unl.edu

² U. Manitoba, Dept CS, toulouse@cs.umanitoba.ca

³ U. Oklahoma, School of CS, thulasi@cs.ou.edu

⁴ U. Mississippi, Hearin Center for Enterprise Science, fglover@bus.olemiss.edu

⁵ U. Nebraska-Lincoln, Dept CS&E, deogun@cse.unl.edu

ABSTRACT

In this paper, we present an adaptation for hypergraph partitioning of the multilevel cooperative search paradigm first introduced by Toulouse, Thulasiraman, and Glover [23]. We also introduce a new approach for coarsening hypergraphs, and describe a parallel implementation of this algorithm 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 outperforms hMETIS, although more modestly.

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 [2]. 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, beside 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 allows for 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 are potentially flawed. This poses serious limitations in the capacities of future multilevel algorithms to provide good quality partitionings. These limitations of the multilevel paradigm have been recently addressed in [10; 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 completely unrelated search algorithms is first selected. Then a cooperation protocol specifies how the search algorithms can share states at run time. This approach has been used with success to design search heuristics in the context of constraint satisfaction problems [3; 11] and to parallelize some metaheuristics [15; 16; 20; 21]. More recently, in [23], cooperative search has been applied 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 [23]. We first propose a semantically different perspective regarding coarsened hypergraphs. Rather than adopting the usual approach where coarsened hypergraphs are seen as graphs to be partitioned, we see them as functions. More specifically, the vertices of coarsened hypergraphs are treated as mappings on the modules of a netlist. This is so because those vertices set the sizes and configurations of the moves executed by search heuristics which, at the end, partition the netlist, not the coarsened hypergraphs. Mathematically, this functional representation is a natural transition to set theory used here to formally describe the multilevel related aspects of our algorithm. It also conveniently provides a framework to integrate the coarsening phase of multilevel algorithms into the bottom-up approach of the cooperative search paradigm, where one needs a generator of unrelated search algorithms. Finally, it supports a brief analysis of the limitations imposed by static coarsening on standard multilevel algorithms.

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. Once the coarsened hypergraphs are generated, together with a set of move-based heuristics, each of them can be considered as a search process executed concurrently with other search processes in time sharing on a sequential computer or in parallel if several processors are available. To allow cooperation among the search processes at run time, states need to be shared among them. Assume H_0, \dots, H_l are the search processes (as well as the hypergraphs). In the current implementation of CoMHP, the cooperation protocol is based on three different state sharing patterns. The first one uses partitionings from H_i as initial solutions to move-based heuristics computing in hypergraph H_{i-1} (this operator is similar to the usual multilevel interpolation operator). The two other sharing strategies use good partitionings from H_{i-1} to partition clusters (vertices) in H_i or to generate new clusters in H_i . We identify respectively these last two state sharing strategies as the *local partitioning operator* and the *local clustering operator*. We like to think of those operators as repairing coarsened hypergraphs H_i based on information obtained from H_{i-1} . If we see coarsened hypergraphs as functions, these two operators change the mapping used at level H_i to partition the netlist. The execution of the three operators is triggered by the internal state of the processes, therefore as a whole, the sharing of states occurs asynchronously.

The rest of the paper is structured as follows. In Section 2 we introduce the mathematical setting and the motivations that support the design of the cooperative multilevel algorithm presented in Section 3. Section 3 also introduces a framework to analyze the convergence behavior of a heuristic like CoMHP. Section 4 reports the results of the tests conducted on the ISPD98 benchmark suite of circuits. Finally Section 5 concludes with some suggestions for future work.

2. MULTILEVEL ALGORITHMS: ISSUES

Hypergraphs are commonly used in the multilevel paradigm as a formal representation of netlists. Let $H_0 = (V_0, E_0)$ be a hypergraph reduction 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 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 in k subsets P_1, P_2, \dots, P_k can be stated as a combinatorial optimization problem where one tries to find an instance $\{P_1, P_2, \dots, P_k\}$ of the mapping

$$\mathcal{P} : V_0 \longrightarrow 2^{V_0} \quad (1)$$

that minimizes the cost function:

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

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

1. $P_i \cap P_j = \emptyset$ ($i \neq j$);

2. $\frac{|V_0|}{ck} \leq |P_i| \leq \frac{c|V_0|}{k}$ for some constant $c \geq 1.0$;

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

Constraint (1) indicates that module replications are not allowed and constraint (2) set bounds on the modules cardinality.

2.1 Hierarchical Clustering

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

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

$$C^i : V_0 \longrightarrow 2^{V_0}, \quad i = 1, \dots, l \quad (3)$$

where each mapping C^i defines a mapping instance of the vertices of H_0 into k^i clusters $C_1^i, C_2^i, \dots, C_{k^i}^i$ such that $C_u^i \cap C_v^i = \emptyset$ if $u \neq v$ and $\bigcup_{u=1}^{k^i} C_u^i = V_0$. Furthermore, $k^i > k^{i+1}$ and $k^1 < |V_0|$.

DEFINITION 2. A coarsened hypergraph $H_i = \{V_i, E_i\}$ is a set of vertices V_i and hyperedges E_i such that $v \in V_i$ is a cluster C_v^i of vertices from V_0 as defined by the mapping C^i and $e \in E_i$ iff $e \in E_0$ and e has at least two vertices $a, b \in V_0$ where $a \in C_u^i$ and $b \in C_v^i$, $u \neq v$.

Given that $k^i = |V_i|$, the number of vertices in H_i , 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 to clusters of C^i by merging randomly pairs of vertices of the coarsened hypergraph H_{i-1} .

In 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 move-based heuristics (mostly FM like heuristics) in a sequence going from H_l to H_0 . Move-based heuristics are iterative search methods where a partitioning $x(t)$ is obtained by a simple transformation of the partitioning at iteration $t-1$. Typically, a transformation corresponds to swapping either a pair of vertices or a single vertex between two subsets of partitioning $x(t-1)$. A transition from a solution (partitioning) $x(t-1)$ to solution $x(t)$ is a *move* executed by the search heuristic in the solution space. (The sequence of solutions $x(0), x(1), \dots, x(\text{last_iteration})$ visited by move-based heuristics is often pictured as a walk in the solution space.) Formally, in the context of partitioning,

DEFINITION 3. A move is a set of swaps between two consecutive evaluations of the cost function.

FM heuristics evaluate a “gain” function after each swap of a module in partitioning¹ $x(t-1)$. According to Definition 2, swapping a vertex $v \in V_i$ of a coarsened hypergraph H_i is equivalent to swapping the cluster C_v^i of modules in the netlist. We identify a move in a coarsened hypergraph as a *coarsened move* and we identify each member of a set of swaps in the netlist related to a coarsened move as a *netlist swap*. Although a coarsened move involves several netlist swaps, according to Definition 3, the set of netlist swaps is a single move in the netlist because no evaluation of the gain function takes place during the execution of the set of netlist swaps. Each coarsened move is a single move in hypergraph H_0 . As an example, consider Figure 1 with the family of mappings:

$$C^1 = \{C_1^1 = \{1, 6\}, C_2^1 = \{2, 3\}, C_3^1 = \{5, 9\}, C_4^1 = \{7, 12\}, C_5^1 = \{13, 14\}, C_6^1 = \{10, 11\}, C_7^1 = \{4, 8\}, C_8^1 = \{15, 16\}\}$$

$$C^2 = \{C_1^2 = \{1, 2, 3, 6\}, C_2^2 = \{5, 9, 13, 14\}, C_3^2 = \{4, 7, 8, 12\}, C_4^2 = \{10, 11, 15, 16\}\}.$$

Assume that bisection $x(t-1)$ of H_2 is $P_2^2 = \{\{1, 2\}, \{3, 4\}\}$.

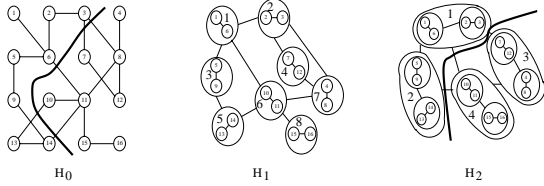


Figure 1: Hierarchical clustering of H_0 using recursive maximal matching.

Using a 2-swap move-based heuristic we seek to improve the cost of bisection $x(t-1)$. One potential move is to swap vertices 1 and 3, moving from $x(t-1)$ to the bisection $P_2^2 = \{\{2, 3\}, \{1, 4\}\} = x(t)$. One can see that bisection P_2^2 is identical to bisection $P_0^2 = \{\{4, 5, 7, 8, 9, 12, 13, 14\}, \{1, 2, 3, 6, 10, 11, 15, 16\}\}$ of H_0 . One can also observe that the coarsened move of vertices 1 and 3 in H_2 has an equivalent move in H_0 involving four netlist swaps of pairs of vertices in H_0 . The one step $x(t-1)$ to $x(t)$ of the walk in the solution space of H_2 is also one step of a walk in the solution space of H_0 . Any coarsened move in a hypergraph $H_i, i > 0$, has an equivalent 2^i swaps move with respect to the same partitioning in hypergraph H_0 .

From this perspective, multilevel algorithms extend move-based heuristics by allowing moves for which the size and configuration have been defined by a clustering algorithm. We can consider coarsened hypergraphs as mappings or functions specifying which modules are involved in one move of a search heuristic. In some instances, the clustering algorithm may even have a closed form, in which case the coarsening phase and coarsened hypergraphs can be eliminated and replaced by a family of mappings as in (3). Obviously maximal matching has a closed form: a simple random function. In several other cases, we may have to keep the coarsening phase as well as the coarsened hypergraphs as the most convenient way to express the function that defines the move sizes and configurations.

Finally, we can see how the coarsening phase of standard multilevel algorithms can be considered to be a generator

¹Others consider the whole sequence of swaps evaluated by the gain function as a single move [24].

of search algorithms in the context of cooperative search. Let \mathcal{W}_i be the set of different walks that can be executed in the solution space of H_0 by a move-based heuristic of the coarsened hypergraph H_i . Provided $V_i \neq V_j$, it is not too difficult to prove that $\exists w_x \in \mathcal{W}_i, w_y \in \mathcal{W}_j$ such that $w_x \neq w_y$. In other words, the same move-based heuristic executed in two different coarsened hypergraphs H_i and H_j do not see the same set of solutions in H_0 . We are therefore justified to consider each coarsened hypergraph as providing a different search algorithm.

2.2 Multilevel: a k -exchange heuristic

By contracting hypergraphs, multilevel algorithms gain run time efficiencies proportional to the number of modules in the cluster that defines a move. As we explain in this Section, the price to be paid for speeding-up computation is to create deficiencies in the search algorithms. Multilevel algorithms are a special case of k -exchange heuristics, a strategy much used to design move-based heuristics and metaheuristics. Let X_0 be the solution space of the minimization problem (2) and let the neighborhood of $x \in X_0$ be defined as $\mathcal{N}(x) = \{y \in X_0 \mid y \text{ can be reached from } x \text{ with a single move of the } k\text{-exchange heuristic}\}$. A *neighborhood structure* is a mapping \mathcal{S} on the partitionings of the solution space X_0 :

$$\mathcal{S} : X_0 \longrightarrow 2^{X_0}. \quad (4)$$

This mapping defines for each solution $x \in X_0$ a set of neighbors $\mathcal{N}(x) \in 2^{X_0}$. In the context of k -exchange heuristics, the neighborhood structure is a function of the value of the parameter k . Some k -exchange heuristics use this parameter to diversify the search. More advanced heuristics such as tabu search use a complicated pattern of values of k to provoke diversification in unexplored regions of the solution space, to initiate intensification of the search around good solutions, to cross infeasible regions of the search space, to probe the solution space, etc. The *crossover operator* of genetic algorithms which partitions solutions and recombines the partial solutions in new and hopefully better solutions is also an implicit application of the k -exchange technique. The crossover operator executes a k -exchange by replacing the state of k variables of a solution x by the state of k variables from another solution y .

When the k -exchange technique is used without restrictions on the configurations of the k swaps, any solution $x \in X_0$ can be reached by a sequence of moves from any other solution $y \in X_0$. This is not the case for the multilevel variant of k -exchange, coarsened moves of multilevel algorithms restrict the number and configurations of netlist swaps by swapping only the vertices within the same cluster in H_i . This restriction has a significant impact. The static configurations of netlist swaps as defined by the clusters $C_j^i \in H_i$ not only affect the neighborhood structure, they also force some solutions in X_0 to be excluded from any walk executed by a move-based heuristic using clusters of H_i , i.e., some solutions of X_0 are not reachable. Furthermore, the size of the vertices in a coarsened hypergraph (the value of k) dictates the size of the solution space for the move-based heuristics. For example, increasing k reduces the set of reachable solutions in the solution space X_0 . Given a hierarchical clustering algorithm based on maximal matching and a coarsening ratio $\frac{|V_i-1|}{|V_i|}$ of approximately two, the size of the set of reachable solutions (the solution space X_i)

of the partitioning problem for hypergraph H_i is given by $|X_i| \approx \frac{1}{k!} \frac{n_i!}{\binom{n_i!}{k}^{\frac{n_i}{k}}}$, $n_i = |V_i|$. $|X_i|$ is substantially smaller than the size of solution space $|X_0| \approx \frac{1}{k!} \frac{n!}{\binom{n!}{k}^{\frac{n}{k}}}$. Although in the literature this reduction of the size of the solution space is considered to be an advantage for the initial partitioning and refinement phases of multilevel algorithms, it potentially defeats these two phases in at least two ways.

Assume an unbiased maximal matching algorithm is used to reduce the size of the netlist, i.e, the vertices and matching vertices are chosen randomly. Then the following model characterizes the solution space defined by the coarsened hypergraphs. Let $\frac{\sum_{x_i} f(x_i)}{n_i}$ be the *average of the hyperedge-cuts* of the solution space X_i . Given a hierarchical clustering algorithm based on an unbiased maximal matching, coarsening has no impact on the distribution of the cost function in coarsened hypergraphs (because low and high cost partitionings are removed from solution spaces with equal probability during the coarsening phase). Therefore the average of the hyperedge-cuts of the solution space X_i , $i \leq l$ will be approximately constant. This in turn implies that coarsening levels the landscape of the cost function by driving out low and high cost partitionings. The leveling action during the coarsening phase can obliterate optimal valleys from the solution spaces of several coarsened hypergraphs, which will then make it impossible for any move-based heuristic to identify good regions of the solution space X_0 . There is no easy fix for this problem. Simulated annealing, which is the thermodynamic realization of the same multilevel idea (temperature acts as a the refinement parameter), also faces this problem. There, in order to compensate for the leveling impact of the temperature parameter on the landscape of the cost function, the number of levels is increased and more time is spent to search each level. This fix is time consuming.

In practice, most clustering strategies not only aim at reducing the size of the hypergraphs, but also try to help the refinement phase by populating the solution spaces with good partitionings (using strategies like “heavy-edge matching” for example). In this case the average of the hyperedge-cuts might not be constant across the solution spaces. Most likely, low cost partitionings will have proportionally more representatives in solution spaces X_i than in the solution space X_0 . This kind of distribution in the solution space has a negative impact on the diversity of the solutions, a limitation identified in the early stage of the development of genetic algorithms. Heuristics like GA move in the solution space by combining blocks of k variables in different ways. For an extensive exploration, such heuristics need a diversified base of combinations, a diversified set of solutions in the population. GA fail when trying to get different solutions from a self-similar genotype (block of similar genes). They are then trapped in a region of the solution space due to the lack of diversity in the unfavorable distribution of good but non-optimal solutions in a population. The vertices of a coarsened hypergraph constitute the genotype (blocks of modules) used by move-based heuristics to create partitionings. Given the small number of vertices in highly coarsened hypergraphs and given the restriction imposed by coarsening on k -exchange, a biased coarsening phase may leave very little diversity in highly coarsened hypergraphs. This in turn causes the partitioning and refinement phases to be trapped

in local optima. Biased clustering strategies can reduce the work needed during the refinement phase, thereby speeding-up the computation, but they may not help to get better partitionings.

3. CoMHP ALGORITHM

Contracting hypergraphs may speed-up the computation of partitionings and help FM-like heuristics by increasing the degree of the vertices. On the other hand, the brief analysis of Section 2.2 shows that contracting cannot help to find good solutions if they have been removed from the solution space during the coarsening phase. Reducing the size of the solution space is not by itself a sufficient condition to successfully identify optimum regions of the solution space. We can address these inherent problems associated with coarsening by using good partitionings found during the refinement phase of standard multilevel algorithms as a source of information to repair the original set of coarsened hypergraphs. There is a cost associate with this approach because of the extra computation during the refinement phase, but the improvement of the quality of the final partitionings is impressive.

Our approach to implement this idea is based on cooperative search. The design space of cooperative search algorithms is quite large, the implementation proposed here is a rather limited use of those possibilities. Our restraint in the current design has been motivated by our desire to keep intact the aspects of the standard multilevel paradigm that have made successful this approach. In other words, we have tried to guarantee the success of this new algorithm based on the knowledge of generations of researchers that have worked on the partitioning problem before us.

The design of this cooperative multilevel algorithm has two phases: first we define the initial search methods (the coarsening phase), second we define the cooperation protocol for sharing states among the search methods.

3.1 CoMHP Coarsening Strategy

In the context of cooperative search and in accordance with section 2.1, we obtain different search algorithms by contracting the original netlist in different coarsened hypergraphs. The vertices of those coarsened hypergraphs are the building blocks that move-based heuristics combine to generate partitionings. The selection of those blocks is critical for a successful refinement phase. Usually recursive *matching-based clustering* algorithms such as *edge-coarsening*, *hyperedge-coarsening*, *maximal matching* and *modified-hypergraph-coarsening* [4; 13] are used to reduce the size of the netlist. Our approach to coarsening is not based on the recursive definition of levels as in most multilevel algorithms. Rather we propose to generate the coarsened hypergraph H_i directly from the netlist, not from 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 $k = k^i = \frac{n}{2^i}$, $0 < i \leq l$. More specifically, we seek a mapping instance $C^i = \{C_1^i, C_2^i, \dots, C_{\frac{n}{2^i}}^i\}$ 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 identify this hierarchical coarsening strategy as *partition-based coarsening*. In practice, the mapping C^i is the hMETIS k -way hypergraph partitioning software [14] and the partitioning returned by

hMETIS is an instance $\{C_1^i, C_2^i, \dots, C_{\frac{n}{2^i}}^i\}$ that minimizes the cost function (2) according to this software. 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 , etc.

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 a diversified set of building blocks for the initial partitioning and refinement phases. 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 k the number of subsets of the partitioning problem. The worst case for CoMHP is when $k \approx \frac{n}{2}$, the number of vertices of H_1 , the less 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 $k \approx \frac{n}{2}$. 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, like recovering the partitioning levels of a partitioning method based on recursive bisection. We can ask the partitioning method to compute a k -way partitioning where $k = \frac{n}{2}$, 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, \dots, 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 [23].

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, we have vertex $C_j^i \in H_i = C_p^{i-1} \cup C_q^{i-1}$ for some vertices $C_p^{i-1}, C_q^{i-1} \in H_{i-1}$. So if $v \in H_0$ is mapped to $C_j^{i-1} \in H_{i-1}$, then v is automatically mapped to C_j^i , the superset of C_j^{i-1} in the coarsened hypergraph H_i . It is then usual to consider hypergraphs 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 $C_j^i \in H_i$ 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 k -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 it 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 might be a handicap since it may be trapped during the uncoarsening phase in the local optima of the solution spaces.

3.2 CoMHP Cooperation Protocol

The cooperation protocol of cooperative algorithms specifies how the set of unrelated heuristics can share states at run time. The design of this protocol is usually quite complex and involves setting-up parameters to govern the four following considerations:

1. Which data are shared;
2. Which heuristics engage in the sharing;
3. The frequency of the sharing;
4. Whether the sharing takes place synchronously or asynchronously.

The specifics about this setting depend naturally on the application. In CoMHP, the unrelated heuristics are the coarsened hypergraphs generated by the coarsening algorithm. The original set of coarsened hypergraphs happen to contain the objects that need to be repaired for a more efficient refinement phase. Consequently the states shared among the unrelated heuristics have to contribute to repairing the coarsened hypergraphs. The solution to this design issue has come from *scatter search*, first proposed by Glover [6] and *vocabulary building* also first introduced by Glover [7]. Scatter search algorithms are population based search heuristics that seek to exploit information contained in “elite solutions” (i.e., best solutions) by combining them to derive new solutions. Vocabulary building is a variation of scatter search which focuses on components of elite solutions (partial solutions) rather than complete solutions. Vocabulary building uses partial solutions extracted from the elite solutions to form a pool of *solution fragments*, which in turn are combined to form larger fragments until complete solutions are generated. Procedures using this design have been implemented for the Vehicle Routing Problem by Rochat and Taillard [18] among many others.

We have adapted principles from scatter search and vocabulary building to set the parameter that specifies which data are shared among the search heuristics. Our adaptation is based on elite solutions (elite partitionings) that are sent from a hypergraph H_i to another hypergraph H_j . The elite partitionings of a hypergraph H_i is a set $X_i^l \subset X_0$ such that the average of the hyperedge-cuts of the partitionings

of X'_i is better than the average of the hyperedge-cuts of all the partitionings identified in H_i from the beginning of the computation until the current iteration t . We express formally this property of elite partitionings as:

$$\frac{\sum_{x \in X'_i} f(x)}{n'_i} < c \left(\frac{\sum_{x \in X_i} f(x)}{n_i} \right) \quad (5)$$

for some real constant $c < 1$, $n'_i = |X'_i|$ and as usual $n_i = |X_i|$. The hypergraph H_j uses the elite solutions received from H_i as filters to identify: 1) new solution fragments (vertices) capable of producing good partitionings once combined by the move-based heuristics in H_j ; 2) current vertices of H_j that need to be destroyed because they are unlikely to produce good partitionings. We set the second parameter above by allowing the sharing of elite solutions only among search heuristics that partition neighbor hypergraphs in the hierarchy of hypergraphs: H_{i-1} and H_{i+1} share partitionings with H_i . The fourth parameter is set as explained in Section 1, the sharing of partitioning takes place asynchronously among the processes. We describe later how we have set parameter 3.

We use elite partitionings in three different ways for the refinement phase, using three different operators: the *local partitioning*, *local clustering* and *interpolation* operators. The three operators use information provided by those elite partitionings. In order to make the description of the three operators more intuitive, we assume that each hypergraph H_i is associated with a process p_i at run time.

3.2.1 Local partitioning operator

At each iteration of process p_i , the local partitioning operator gets the current set X'_{i-1} of hypergraph H_{i-1} . The local partitioning operator uses elite solutions of X'_{i-1} to identify which vertices (clusters) in V_i need to be destroyed. This identification is achieved by finding clusters $v \in V_i$ such that v has at least two vertices $a, b \in V_0$ that are into two different subsets of at least one of the elite partitionings of X'_{i-1} . In other words, the local partitioning operator destroys vertices of hypergraph H_i that overlap subsets of elite partitionings in X'_{i-1} .

This operator first selects all the vertices $v \in V_i$ that satisfy $v \cap P_p \neq \emptyset, v \cap P_q \neq \emptyset$ for at least one elite partitioning $x \in X'_{i-1}$ (P_p, P_q are the subsets of partitioning x). Let D_{it} be the set of vertices in V_i that overlap more than one subset P of a partitioning $x \in X'_{i-1}$. Local partitioning partitions the vertices of D_{it} so that they do not overlap the subsets of at least one partitioning in X'_{i-1} . Therefore the operator seeks an $x \in X'_{i-1}$ that has at least two subsets overlapped by $v \in D_{it}$. For example, if v overlaps $P_1, P_2 \in x_1$, then v is partitioned in two vertices $v_1, v_2 \in V_i$ such that $v_1 \subset P_1$ and $v_2 \subset P_2$. Since $v_1 \subset v$ and $v_2 \subset v$, $v_1 \cap v_2 = \emptyset$ and v_1, v_2 are legal vertices according to Definition 2 of a vertex in a coarsened hypergraph.

In Figure 2, $C_1 \cap P_1 \neq \emptyset$ and $C_2 \cap P_2 \neq \emptyset$ for $x \in X'_0$. Vertex C_1 overlaps subsets P_1 and P_2 of partitioning x of H_0 . The local partitioning operator identifies C_1 and repairs hypergraph H_1 by partitioning vertex C_1 in two new vertices 10 and 14 in H_1 . Vertex $10 \subset P_2$ and vertex $14 \subset P_1$, $10 \subset C_1, 14 \subset C_1, 10 \cap 14 = \emptyset$, therefore 10 and 14 are two legal vertices for the coarsened hypergraph H_1 . The same situation holds for vertex C_2 of H_1 . As explained in Section 2, the solution space X_i is smaller than X_{i-1} , consequently X_i is faster to search than X_{i-1} . But this advantage may

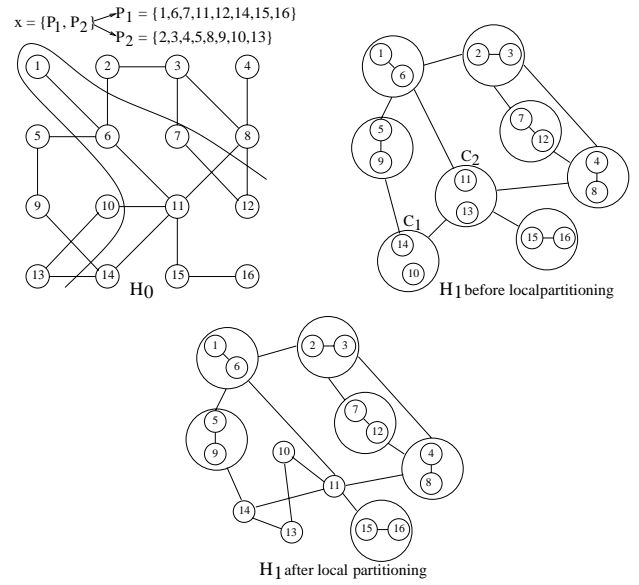


Figure 2: Repair of H_1 by local partitioning.

be lost if H_i has too many flawed blocks (vertices), preventing heuristics from identifying interesting valleys of the solution space X_0 . By partitioning clusters like C_1 and C_2 , the local partitioning operator destroys two partial solutions (C_1 and C_2) that prevent move-based partitioning processes from reaching a good partitioning such as $x \in X_0$. By using elite partitionings from X_{i-1} to identify which vertices of H_i to destroy, the local partitioning operator presumably opens interesting regions of the solution space X_0 to the move-based heuristics at level i .

In terms of implementation, we usually limit 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 [9] and Glover [8].

The execution of local partitioning is followed by the execution of a move-based heuristic on the refined hypergraph H_i .

3.2.2 Local clustering operator

At each iteration of process p_i , the local clustering operator gets the current X'_{i-1} of hypergraph H_{i-1} . Local clustering uses elite partitionings from X'_{i-1} to identify which vertices of H_0 can be clustered together in the coarsened hypergraph H_i . The current implementation is simple. To create a new vertex in H_i , local clustering merges vertices from H_{i-1} that are in the same subsets of all elite partitionings in X'_{i-1} . For example, let $X'_{i-1} = \{x_1, x_2, x_3\}$. Local clustering looks for a vertex $v \in V_{i-1}$ such that $v \subset P_j^1, P_j^2, P_j^3, 1 \leq j \leq k$ (P_j^1, P_j^2, P_j^3 are subset j respectively of partitionings x_1, x_2 and x_3 in X'_{i-1}). Let $C_{it} \subseteq V_{i-1}$ be the set of vertices in V_{i-1} that are in the same subset j of all partitionings in X'_{i-1} . Local clustering creates new vertices in the coarsened hypergraph H_i by merging vertices from C_{it} . The current implementation merges two vertices together from C_{it} according to the two following criteria: 1) The two vertices $v_p, v_q \in C_{it}$ are in the same subset P for all partitionings $x \in X'_{i-1}$; 2) the two vertices lie on the same hyperedge.

In Figure 3, local clustering merges vertices 10 and 13 of H_0 to create a new cluster C_1 in H_1 . The new cluster C_1 represents the fact that vertices 10 and 13 are together in the same subset of the good partitioning x in X_0 . Similarly for cluster C_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

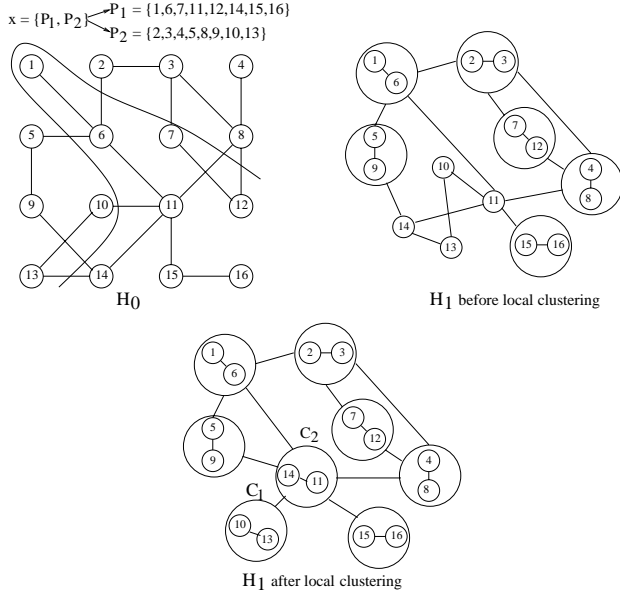


Figure 3: Repair of H_1 by local clustering.

vertices tends to be relatively constant in the coarsened hypergraph although we do not do anything specific to keep this number constant. We hypothesize at this point that the stable number of vertices is related to the set of elite partitionings. We use the same set for both the local clustering and local partitioning operators.

3.2.3 Interpolation Operator

The interpolation operator of process p_i uses the current best partitioning of H_{i+1} as an initial solution of a move-based heuristic in hypergraph H_i . Let x be the best partitioning of hypergraph H_{i+1} at iteration t of process p_i . 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 needs to be performed. Let P_1, P_2, \dots, P_k be the subsets of partitioning x . The interpolation operator of process p_i looks for every vertex $v \in V_i$ that overlaps more than one subset of the partitioning x . Let $v \in V_i$ be such a vertex overlapping subsets P_p, P_q of x . Then two vertices $v_p, v_q \in V_i$ are created in the following manner: $v_p = v \cap P_p$ and $v_q = v \cap P_q$. 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

One iteration of any process p_i corresponds to one iteration of the following main loop: local partitioning, interpolation,

local clustering and global search. Processes share partitionings at each iteration of the main loop, this is how we set the parameter regarding the frequency for information sharing (parameter 3 above). Figure 4 details the main loop of the each process.

```

CoMHP(); /* process  $p_i$  */
Compute an initial partitioning; /* initial partitioning phase of
    standard multilevel algorithm */
While not terminated { /* begin outer loop */
    Apply local partitioning to  $H_i$ ;
    Execute FMS and PFM on new  $H_i$ ;
    Apply interpolation to current  $H_i$ ;
    Execute FMS and PFM using a good partitionings of  $H_{i+1}$ ;
    Apply local clustering to current  $H_i$ ;
    Execute FMS and PFM on new  $H_i$ ;
    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 4: 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) [19], and the multiway partitioning by free moves (PFM) proposed by Dasdan and Aykanat [5]. 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 high levels, for coarsened hypergraphs having less than 500 vertices. We use the multilevel k -way hypergraph partitioning algorithm [14] as another global search algorithm in CoMHP.

The three operators of the cooperation protocol are based on l mapping vectors as described in Definition 1. Given a mapping instance $C^i = \{C_1^i, C_2^i, \dots, C_{k^i}^i\}$ for level i , a mapping vector is an array of integers of size $|V_0|$ that maps each vertex $v \in V_0$ to a cluster $C_j^i \in C^i$. Each execution of the partial clustering and partial partitioning operators defines a new mapping instance C^i at level i , which implies that the mapping vector needs to be updated. Partitionings are also defined in the same manner using mapping vectors. The total space requirement for the cooperation protocol is $O(2 \times l \times |V_0|)$. The cooperation protocol can be implemented at no cost in term of space requirement using the *recursive mapping*:

$$C^i : V_{i-1} \longrightarrow 2^{V_{i-1}} \quad (6)$$

rather than the direct mapping to H_0 of Definition 1. This is possible if the vertices of H_i are the union of vertices of H_{i-1} . In that case we simply need a mapping vector of size $|V_{i-1}|$ for each level i , but this mapping vector is already part of standard multilevel data structures. Since we are using mapping instances to find the sets D_{it} and C_{it} of vertices, the time requirement for the three operators is also

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 C^i and the partitionings of X'_{i-1} to get D_{it} . Then a second sweep across the vertices of H_0 is performed to get a new mapping instance 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.

3.2.5 How CoMHP Works

Intuitively it is not too difficult to understand how these three operators help to improve the performance of the local and global searches. In traditional multilevel partitioning, once the hierarchy H_0, \dots, H_l is computed, it remains static as iterative refinement takes place. In our implementation, the hypergraphs H_1, \dots, H_l change dynamically. The local partitioning operator finds those vertices in V_i that overlap subsets of good partitionings in X_{i-1} and destroys them. The local clustering operator populates hypergraph H_i with good vertices (in term of permutations). These changes in the coarsened hypergraphs are based on elite solutions which open for exploration of new and interesting regions of X_0 (by move-based heuristics working with small hypergraphs). This in turn helps the interpolation operator to be more efficient in directing move-based heuristics working with larger and less dense hypergraphs.

There is another critical aspect to the convergence behavior of CoMHP. Cooperative algorithms such as CoMHP are dynamical systems [22], their convergence behavior is governed by an energy function similar to the energy function of Hopfield networks [12] and other similar dynamical systems. The simplest way to describe the convergence behavior of CoMHP is through a brief description of a recursive network like Hopfield networks. A recursive network of l units is defined by a state vector $x = [x_1, x_2, \dots, x_l] \in \{-1, 1\}^l$, a $l \times l$ matrix of couplings $W = [w_{ij} : 1 \leq i, j \leq l]$ and θ a real l -vector. The matrix W and the vector θ are the parameters of the recursive network. The evolution of the system along the time axis is described by a *dynamic equation* of the type:

$$x_i(t+1) = \text{Sgn} \left[\sum_{j=1}^l w_{ij} x_j(t) - \theta_i \right], \quad i = 1, 2, \dots, l \quad (7)$$

One can observe that as the system is iterated, it will go across states among the 2^l possible state vectors, the *state space* of the system. The evolution of such a system is often governed by an *energy function*

$$E(x) = -\frac{1}{2} \sum_{i \neq j=1}^l w_{ij} x_i x_j. \quad (8)$$

Under certain conditions, this energy function is strictly decreasing on a finite interval of time. Any iteration of the system yields a trajectory in the state space of the system that minimizes the energy of the system. Recursive networks can be used to solve combinatorial optimization problems. In this case, the objective function is mapped into an energy function. As the system is iterated, it converges toward a minimum energy level, from which the solution to the optimization problem can be read. An example of energy function for the bisection of a graph is given by:

$$E(x) = -\frac{1}{2} \sum_{i,j=1}^l w_{ij} x_i x_j + \frac{\alpha}{2} \left(\left(\sum_{i=1}^l x_i \right)^2 - \sum_{i=1}^l x_i^2 \right), \quad (9)$$

where the term $\frac{\alpha}{2} \left(\left(\sum_{i=1}^l x_i \right)^2 - \sum_{i=1}^l x_i^2 \right)$ is a penalty function that increases the energy of the system when the subsets of the partitionings are not balanced (see [17] for more details).

The state vector $\mathcal{X}(t)$ of the dynamical system CoMHP is given by the union of $l+1$ boolean vectors of size m , where $m = k \times |V_0|$, i.e., m is the number of variables of the optimization problem times the number of subsets k in the partitioning problem. The value $l+1$ corresponds to the number of processes. Each variable $\mathcal{X}(t)_i$ of the state vector $\mathcal{X}(t)$ is a boolean value that is equal to 1 when $i = v \times q$, where v is a vertex of V_0 , $v = 1, \dots, |V_0|$, and v is in the subset q of the partitioning, $q = 1, \dots, k$. Otherwise $\mathcal{X}(t)_i = 0$. If the number of hypergraphs is equal to $l+1$, a state of the cooperative search is given by the union of the $l+1$ boolean vectors representing the current partitioning at each level. The matrix of couplings W is given by the cooperation protocol. Finally θ_i corresponds to the search heuristic at level i . The dynamics of CoMHP are provided by the search heuristics and the cooperation protocol. The energy function is given by

$$E(\mathcal{X}) = \sum_{i \neq j=0}^l \left\{ \frac{\sum_{x \in X'_i} f(x)}{n'_i} - \frac{\sum_{x \in X'_j} f(x)}{n'_j} \right\} \quad (10)$$

i.e., the sum of the differences of the average of the hyperedge-cuts of the elite solutions among all the levels. The state space is the $2^{m \times (l+1)}$ possible state vectors. Designing a cooperative algorithm like CoMHP consists of selecting a set of search heuristics and a cooperation protocol such that the dynamics of the system converge toward interesting local optima of the optimization function.

The impact of the matrix of couplings on the convergence behavior and the evolution of the CoMHP is obvious. The three local operators keep changing the vertices of the coarsened hypergraphs. These changes define the regions of the state space that are reachable by the cooperative system. The trajectory of the cooperative algorithm in the reachable state space is then set by the cost function of the optimization problem as well as some other parameters related to the move-based heuristics. The reachable state space is redefined every time vertices are created or destroyed. The process of creating and destroying vertices ends precisely when the energy function of the cooperative search has reached a minimum (i.e., when the elite solutions have similar hyperedge-cuts). At that point the trajectory of the system is confined to the same region of the state space forever (as it is the case for a standard multilevel algorithm). 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 stop to have any significant impact on the exploration of the solution space, the system has converged to a minimum energy state.

The use of an energy function to control and coordinate a set of unrelated search heuristics is what makes cooperative algorithms so powerful. Unfortunately, the cooperation protocol and the local heuristics of cooperative algorithms

do not lend themselves to easy functional representations. Consequently designing cooperative algorithms is often very tricky and is based on empirical observations, i.e., trial and error. This state of improvisation often results in cooperative algorithms that are not particularly stable. 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 the k -exchange moves. This makes the dynamics of local heuristics much more stable and allows a better control on the convergence behavior of the whole cooperative search. For this reason, the present design of our cooperative multilevel algorithm for hypergraph partitioning has been mostly oriented toward achieving a stable cooperative algorithm. A fortunate combination of an adaptation of scatter search principles to hypergraph partitioning with the way multilevel algorithms use k -exchange, has produced, to our knowledge, one of the most stable and efficient cooperative search algorithms. The resulting cooperative multilevel algorithm is also a new design approach for efficient cooperative search algorithms. In future designs, this intuitive approach will be supplemented by qualitative studies of the convergence behavior that have been used previously to design other cooperative algorithms [22]. These approaches are based on cellular automata theory and statistical mechanics. We anticipate such an integration will enable the design of cooperation protocols whose resultant energy functions will allow even better partitionings to be found.

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 the version 1.5.3 the hMETIS partitioning package. We have implemented a parallel version of our hypergraph partitioning algorithm and have run it on 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

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

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. But currently, the amount of improvement in the hyperedge-cuts of CoMHP is not significant after 3 or 4 iterations of the search phase by process p_0 . At that point the energy function (10) is low and seems stable in its minimum. This behavior was unexpected since, in the version for graph partitioning, the energy function goes through several plateaus before stabilizing. This behavior of the energy function for CoMHP is most likely caused by our coarsening algorithm which is strongly biased by the cost function of each coarsening level. The refinement phase starts in good regions of the solution space, causing CoMHP to converge rapidly toward good partitionings, but unable however to generate vertices in the coarsened hypergraphs that will cause the dynamical system to explore substantially different regions of its state space. It could be an indication that, unlike the expectations we had for our coarsening strategy, it fails to provide enough diversity in the original set of coarsened hypergraph. We could either try other coarsening approaches or change the cooperation protocol so as to allow a more extensive exploration of the solution space based on the assumption that the original coarsened hypergraphs are too entrenched in local optima.

At this point our choice of 10 iterations as a stopping criterion does not reflect accurately the current runtime competitiveness of CoMHP. In fact, in the current version of CoMHP, running only 2 or 3 iterations will not result in any serious degradations of the results obtained using 10 iterations. We have also performed tests with hMETIS using 100 restarts and also allowing the same runtime for hMETIS as for CoMHP with 10 iterations. The hyperedge-cuts were not substantially better than with 10 runs, in this regard hMETIS is quite stable.

5. CONCLUSION AND FUTURE WORK

We have described a new netlist partitioning algorithm inspired from the multilevel paradigm and the cooperative search paradigm. We look at the standard multilevel paradigm as a variant of k -exchange, a technique used to define the move sizes of move-based heuristics. We see the vertices of coarsened hypergraphs as a mechanism to select the move sizes and configurations of the local search heuristics that partition the netlist. Following this perspective, the set of coarsened hypergraphs constitutes a set of unrelated search heuristics. Then we introduce a cooperation protocol that specifies how those heuristics share states at run time.

The runtimes of the current CoMHP algorithm do not compare well with a partitioning method like hMETIS. This problem will be addressed in future designs of cooperative multilevel algorithms. Currently, the computational time of CoMHP is dominated by the execution of the global and local search subroutines. A global search like hMETIS is called each time a coarsened hypergraph has been modified by the local clustering and partitioning operators. Local search methods are used with the same philosophy. We believe that we will be able to reduce substantially the use of global and local searches without degrading the quality of

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

partitionings. We can also adapt the local search 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. In the mean time, we can stop the execution after a few iterations (2 or 3 rather than 10 iterations as in our experiments) without important degradations of the results obtained with 10 iterations. Then the runtime performance of CoMHP appears in a more favorable light.

The originality of CoMHP comes from the use of the cooperation protocol to control local and global move-based search methods. As explained in the paper, we have been very conservative in exploiting this aspect of the design space of cooperative algorithms. Improvements both in terms of computational time and quality of partitionings are most likely to result from refining the choice of elite solutions, the set of “repair” operators, the selection of the levels between which sharing takes place, etc. Another important aspect to improve cooperative multilevel algorithms is a better understanding of the dynamics resulting from the couplings specified by the cooperation protocol, the heuristic of the search algorithms and the interactions among these two components of cooperative algorithms. A formal approach to study this aspect of cooperative algorithms is still in its infancy. Because of the difficulties to get a functional representation of the cooperation protocol and search heuristics, in the past, we have used cellular automata (as an alternative to differential equations) to simulate the dynamics of cooperative algorithms [22]. Cellular automata can be programmed to simulate the interactions between the cooperation protocol and search heuristics and then iterated as simplified dynamical models of cooperative search. The qualitative dynamical behavior of some cellular automata is well formalized, such simulations help to understand the convergence behavior of the energy function in cooperative systems. This approach is most likely also applicable to multilevel cooperative algorithms. We think that rigorous studies of the dynamics and energy function of multilevel cooperative algorithms are bound to produce refinements that will yield substantially better partitionings.

Finally this cooperative search paradigm could 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, it could adapt to local constraints imposed on coarsening by specific needs from the physical design process. This flexibility may not always be there however. It comes from the very nature of dynamical systems like CoMHP that the energy function can govern the evolution of such systems toward the same energy levels even when some perturbations are applied to the system. It is also true that minor changes can cause the system to bifurcate into spurious attractors with poor partitionings. To be successful in this project, we will need to adapt a methodology like the *stability analysis* of equilibrium states in nonlinear dynamical systems to cooperative search algorithms. Characterizing the properties of the equilibrium states of multilevel cooperative algorithms will also be profitable to many other aspects of cooperative algorithms.

Acknowledgements

We are greatly indebted to Charles Alpert from IBM Austin Research Laboratory for his helpful comments and sugges-

tions on an earlier draft of this paper.

6. REFERENCES

- [1] E.H.L. Aarts and J.K. Lenstra. Introduction. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 1–17. John Wiley & Sons Inc., 1997.
- [2] C.J. Alpert. The ISPD98 Circuit Benchmark Suite. In *Proc. of the International Symposium on Physical Design (ISPD-98)*, pages 80–85. ACM Press, 1998.
- [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, 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] F. Glover. Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences*, 8(1):156–166, 1977.
- [7] F. Glover. Ejection Chains, Reference Structures and Alternating Path Methods for the Traveling Salesman Problem. Report, University of Colorado, Boulder, 1992.
- [8] 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.
- [9] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- [10] A. Gupta. Fast and Effective Algorithms for Graph Partitioning and Sparse Matrix Ordering. Report RC 20496, IBM T.J. Watson Research Center, 1995.
- [11] 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.
- [12] 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.
- [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] C. Peterson and B. Söderberg. Artificial Neural Networks. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 173–213. John Wiley & Sons Inc., 1997.
- [18] Y. Rochat and E. Taillard. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1):147–167, 1995.
- [19] L.A. Sanchis. Multiple-way Network Partitioning. *IEEE Trans. Comput.*, 38(1):62–81, Jan. 1989.
- [20] 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.
- [21] 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.
- [22] M. Toulouse, T.G. Crainic, and K. Thulasiraman. Global Optimization Properties of Parallel Cooperative Search: A Simulation Study. *Parallel Computing*, 26:91–112, 2000.
- [23] 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.
- [24] M. Yannakakis. Computational Complexity. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 19–57. John Wiley & Sons Inc., 1997.