MAkE: Multiobjective algorithm for \(k\)-way equipartitioning of a point set

Sriparna Saha\textsuperscript{a,}\textsuperscript{*}, Susmita Sur-Kolay\textsuperscript{b}, Parthasarathi Dasgupta\textsuperscript{c}, Sanghamitra Bandyopadhyay\textsuperscript{d}

\textsuperscript{a}Machine Intelligence Unit, 203 B.T. Road, Indian Statistical Institute, Kolkata, West Bengal 700108, India
\textsuperscript{b}Faculty of the Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India
\textsuperscript{c}Faculty of the MIS group, Indian Institute of Management Calcutta, Kolkata, India
\textsuperscript{d}Faculty of the Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India

\begin{abstract}
The classical problem of partitioning a given set of points, has applications in several areas such as facility location, scattered network, and in hierarchical design of VLSI circuits. While equipartitioning is traditionally associated with the single objective of minimum cutcost, the above application areas appear to demand more. In this paper, we introduce the problem of multiobjective \(k\)-way equipartitioning of a point set. Brief discussions on the above applications are followed by their generic formulation as a multiobjective \(k\)-way equipartitioning problem of a given point set. The non-commensurate multiobjective criteria addressed include (i) minimizing overall areas of the partitions, (ii) maximizing area of the individual partitions, (iii) minimizing the total compactness of the partitions, and (iv) minimizing the total geometric diversity of the obtained partitions. Since this optimization problem is computationally expensive in time and space, a technique based on genetic algorithm is proposed in order to obtain high quality results. Crossover and mutation operators specific to the \(k\)-way equi-partitioning problem, have been designed and a new greedy operator named compaction is proposed to accelerate convergence. To illustrate the utility of the proposed formulation and the algorithm, a problem in VLSI layout design is considered. Results on synthetic data sets as well as those extracted from layouts of benchmark circuits demonstrate the effectiveness of the proposed multiobjective approach.
\end{abstract}

\section{Introduction}

The problem of partitioning a set of objects into a given number of subsets is well-known and has practical applications in several areas including data mining, market segmentation, facility location, scatternets and VLSI layout design automation. For a given set of points in a 2-dimensional plane, typical partitioning criterion is based on the Euclidean distance between pairs of points. In practice, however, the problem is more complex and involves additional optimization criteria depending on the application domain.

This paper considers partitioning a given number of points in a 2-dimensional plane such that the partitions are balanced with respect to the number of points assigned to each of them, each partition occupies minimum area, and total geometric skew (a parameter capturing the geometric diversity; which is defined formally in Section 3) of the obtained partitioning is minimum. The non-commensurate properties of the objectives warrant the use of a multiobjective optimization technique.

Multiobjective optimization (MOO) problem, usually has a rather different perspective. While in single objective optimization there is only one global optimum, in multiobjective optimization there is a set of global optimum solutions called \textit{Pareto-optimal set}; all these solutions having equal importance. A single objective approximation of multiple objectives, in the form of a weighted sum, unfortunately often fails to capture the full Pareto optimal front. Over the past decade, a number of multiobjective evolutionary algorithms have been suggested [1,2]. The prime motivation for using evolutionary algorithms (EAs) to solve multiobjective problems is their population-based nature and ability to find multiple optima simultaneously. A simple EA can be easily extended to maintain a diverse set of solutions.

The problem of multiobjective \(k\)-way equipartitioning of a point set being hard [3], we propose a method for this multiobjective optimization based on genetic algorithm (GA): Multiobjective Genetic Algorithm for \(k\)-way Equi-partitioning (MAkE). Our proposed algorithm is based on a MOO technique. In this paper,
2. Motivating applications

The idea of equipartitioning a point set is very important in several fields, such as facility location, computer-aided design (CAD) of VLSI circuits for flip-flop partitioning and in ad-hoc network configuration, specifically in scatternets. A brief overview of some of these applications is provided below.

2.1. Facility location

The general facility location problem states: given a set of facility locations and a set of customers to be served from the facilities, assign a facility location to each customer such that the total cost of serving all the customers is minimized. A variant of this problem is defined in point domain as follows: Given a distribution of $n$ points representing locations of customers or clients, a subset of $k$ points has to be chosen from these, such that the facilities (warehouses) can be located there to serve the clients and the appropriate cost function(s) is(are) optimum. As each facility is serving a subset of clients, multiple objectives such as the sum of the distances of a facility from its clients, the maximum distance of a facility from its clients (retailers) are to be minimized. Fig. 1(a) shows a graphical representation of the facility location problem where the facility locations and customer locations are represented by two different types of points.

It is interesting to note that this can be modeled as k-way equipartitioning problem. The algorithm proposed in this paper is capable of solving the problem where each partition of points represents a distinct set of client locations and their respective facilities are located at the centre of each partition.

2.2. Scatternet formation

Bluetooth is a protocol defining method for data communication that uses short-range radio links (piconets) to replace cables between computers and their connected units. For data transmission in Bluetooth technology at first a piconet is created. Bluetooth Scatternets are concerned with the interconnection of Bluetooth piconets. An example of a Scatternet is illustrated in Fig. 1(b). In Scatternet formation, for a given set of device locations, it is useful to consider the performance of the network in terms of several objectives, namely (i) the number and the location of the masters, the slaves and the bridges, (ii) the sizes of the piconets, (iii) the length of the average shortest path. These factors influence the efficient flow of data within the Scatternet, and other aspects such as reliability and connectivity. Thus several of conflicting objectives may have to be considered for efficient design of Scatternets.

Given a set of devices (slave nodes), the formation of Scatternet can be considered as equipartitioning the given set of slave nodes such that there is a specific number of nodes in each partition. As discussed above, several objectives specific to Scatternet formation, have to be optimized. Thus the above problem can also be treated as k-way equipartitioning problem with multiple objectives.

2.3. VLSI design automation (CAD VLSI)

In computer aided design (CAD) of VLSI circuits, appropriate distribution of clock signal to the different circuit components is a major design requirement of circuits. Flip-flops play a very significant role in an integrated circuit, and with the semicon-
ductor technology in the gigascale regime, the number of flip-flops could be of the order of several millions. As such, sending clock signals to each of these flip-flops appropriately in order to have correct circuit functionality is indeed a great challenge. Appropriate placement of flip-flops (or registers) is regarded in literature as a means of reducing the wirelength for clock trees. It may be noted that in the partitioning of set of registers or flip-flops for the purpose of minimization of skew, wirelength, and interconnect power consumption [4], the performance of the clock tree depends to a large extent on the partitioning quality. It can be shown that the equipartitioning of a set of flip-flops, such that total area in each partition is minimum, tends to reduce the total skew of the clock tree. The commonality of issues in the three different domains naturally suggest a k-way equipartitioning problem as introduced formally in the next section.

3. Problem formulation

First, a few relevant definitions are in order. Let \( P \) denote a set of points.

1. Bounding box of \( P \): A rectangular bounding box of a set \( P \) of points in a plane is the smallest rectangle that encloses all points in \( P \).
2. Area of a partition: The area of a partition is the area of the bounding box enclosing all the points in that partition.
3. Average radius of a point in \( P \): For a point \( p_j \in P \), its average radius is the average of the Euclidean distances \( d(p_i, p_j) \) for points \( p_i \in P \), \( p_j \neq p_i \).
4. Centre of \( P \): A point \( p_c \in P \) is the centre of \( P \) if its average radius in \( P \) is the minimum over all points in \( P \).
5. Skew of \( P \): Suppose the point set \( P \) has \( n \) number of points and the centre of the point set is denoted as \( p_c \). Then total skew of this point set is calculated as

\[
\delta = \max_{j=1, \ldots, n} d(p_c, p_j) - \min_{j=1, \ldots, n} d(p_c, p_j).
\]

Fig. 2 shows an example of a point set. Here centre of the point set is denoted by \( p_c \). Then the skew of this point set is defined as:

\[
\delta = d(p_c, p_1) - d(p_c, p_3).
\]

The problem is then formally stated as follows: k-way Equipartitioning Problem: Given a set \( P \) of \( n \) points in the 2-dimensional plane, and a positive integer \( k \), determine \( k \) crisp partitions of the points, such that the number of points in each partition lies in the range \( \lceil (n/k) \rceil \leq |P_i| \leq \lfloor (n/k) \rfloor \) and the four objective functions to be minimized are the following:

- \( f_1 \), the sum of the areas of all partitions,
- \( f_2 \), the maximum of the areas of all partitions,
- \( f_3 \), the total deviation over all partitions given by \( \sum_{i=1}^{k} (\text{size of } P_i - \text{size of } \bar{P}_{(i-1)k+1}^{i-1}) \), where \( P_i \) and \( P_j \) denote the centre and the \( j \)th point of the \( i \)th partition.
- \( f_4 \), the overall skew due to partitions is determined in the following way: Let there are \( k \) partitions of the data set. Suppose the \( i \)th partition has \( n_i \) number of points in it and the centre of the \( i \)th partition is denoted as \( p_{ci} \). For each partition \( i \) in a solution set \( S \), total skew, \( \delta_i \), is calculated using Eq. 1. Then \( \delta = \max_{i=1, \ldots, k} \delta_i \). The skew of the reduced graph of \( k \) partition centres, whose centre is denoted by \( p_r \), is calculated as \( \delta = \max_{i=1, \ldots, k} d(p_{ci}, p_r) - \min_{i=1, \ldots, k} d(p_{ci}, p_r) \). The bound for skew due to partitioning is \( (\delta + \delta_r) \). Thus, we define, \( f_4 = \delta + \delta_r \).

Since the CAD VLSI application is considered for our implementation and experiments, we have provided the above formulation primarily in line with that application. However, minor variants of this formulation can be conveniently used for the other applications as well.

The single objective k-way equipartitioning problem is NP-hard [3]. Thus, the multiobjective formulation of the k-way equipartitioning problem is more complex. Hence, the design of an efficient optimization technique is needed. For this problem a GA-based method is appropriate as GAs are known to suit such requirements well.

4. Background

In this section, we summarize the existing GA-based partitioning algorithms along with some preliminaries of multiobjective optimization.

4.1. Existing partitioning algorithms

The existing partitioning algorithms, in general, partition the given data based on different criteria. Several algorithms for partitioning data are available in the literature viz. k-means [5], branch and bound procedure [6], maximum likelihood estimate procedure [7] and graph-theoretic approaches [8]. The simplest and most popular among iterative and hill climbing partitioning algorithms is k-means. But this algorithm may converge to a suboptimal partition based on the choice of initial partition centres. Since stochastic optimization approaches are good at avoiding convergence to a locally optimal solution, these approaches could be used to find a globally optimal partition. The stochastic approaches used in partitioning include those based on simulated annealing, genetic algorithms, evolution strategies and evolutionary programming [9–14]. Genetic algorithms (GA) are the best known evolutionary technique [15]. There exists a large number of GA-based partitioning techniques in literature [10,11,13,14,16,17]. All the above mentioned algorithms cannot tackle more than one partitioning criterion effectively. In recent years, few multiobjective partitioning techniques have been proposed to handle different shaped partitions [18,19]. To the best of our knowledge, none of these algorithms consider the equipartitioning constraint.

4.2. Multiobjective optimization

The multiobjective optimization can be formally stated as follows [20]. Find the vectors \( X = [X_1, X_2, \ldots, X_M]^T \) of decision variables that simultaneously optimize the \( M \) objective values \( \{ f_1(X), f_2(X), \ldots, f_M(X) \} \), while satisfying the constraints, if any.
An important concept of multiobjective optimization is that of domination. In the context of a maximization problem, a solution \( \overline{\pi} \) is said to dominate \( \overline{\pi}' \) if \( \forall k \in 1, 2, \ldots, M, f_k(\overline{\pi}) > f_k(\overline{\pi}') \) and \( \exists k \in 1, 2, \ldots, M, \text{ such that } f_k(\overline{\pi}) > f_k(\overline{\pi}') \).

Among a set of solutions \( P \), the nondominated set of solutions \( P' \) are those that are not dominated by any member of the set \( P \). The nondominated set of the entire search space \( S \) is the globally Pareto-optimal set. In general, a multiobjective optimization algorithm usually admits a set of solutions that are not dominated by any solution encountered by it. These notions can be explained with a two-objective optimization problem which has five different solutions, as shown in Fig. 3.

### 4.2.1. MOO performance measures

In multiobjective optimization, basically two functionalities must be achieved regarding the obtained solution set [20]. It should converge as close to the true Pareto-optimal front as possible, and it should maintain as diverse a solution set as possible. The first condition clearly ensures that the obtained solutions are near optimal and the second condition ensures that solutions with a wide range of trade-off objectives are obtained. Clearly, these two tasks cannot be measured with one performance measure adequately. A number of performance measures for MOO algorithm have been suggested in the past. Here we have mainly used two such performance measures.

The first measure called Purity [21] is used to compare the solutions obtained using different MOO strategies. It calculates the fraction of solutions produced by a particular algorithm which remain nondominated in the union of the Pareto front solutions obtained by all used algorithms. A value near to 1(0) indicates better (poorer) performance. The second measure named MinimalSpacing [21], reflects the uniformity of the solutions over the nondominated front. Smaller values of MinimalSpacing for a particular MOO algorithm indicate better performance.

### 4.2.2. Nondominated sorting genetic algorithm-II (NSGA-II)

Genetic algorithms are known to be more effective than classical methods such as weighted metrics, goal programming [20], for solving multiobjective problems primarily because of their population-based nature. NSGA-II [22] is widely used in this regard, where initially a random parent population \( P_0 \) is created and the population is sorted based on the partial order defined by the non-domination relation—a sequence of nondominated fronts is obtained. Each solution of the population is assigned a fitness which is equal to its non-domination level in the partial order. The authors have assumed the minimization of fitness. A child population \( Q_0 \) of size \( N \) is created from the parent population \( P_0 \) by using binary tournament selection, recombination, and mutation operators. According to this algorithm, in the \( t \)th iteration, a combined population \( R_t = P_t \cup Q_t \) is formed. The size of \( R_t \) is \( 2N \). All the solutions of \( R_t \) are sorted according to non-domination. If the total number of solutions belonging to the best nondominated set \( F_1 \) is smaller than \( N \), then \( F_1 \) is totally included in \( P_{t+1} \). The remaining members of the population \( P_{t+1} \), are chosen from subsequent nondominated fronts in the order of their ranking. To choose exactly \( N \) solutions, the solutions of the last included front are sorted using the crowded comparison operator [22] and the best among them (i.e., those with lower crowding distance) are selected to fill in the available slots in \( P_{t+1} \). The new population \( P_{t+1} \) is then used for selection, crossover and mutation to create a population \( Q_{t+1} \) of size \( N \). The pseudocode of NSGA-II is provided in Fig. 4.

### 5. Proposed multiobjective genetic algorithm for \( k \)-way equipartitioning (MAkE)

A multiobjective genetic algorithm, MAkE, along the lines of NSGA-II, is now proposed for solving the \( k \)-way equipartitioning problem where the objective functions \( \{f_1, f_2, f_3, f_4\} \) defined above in Section 3, are minimized. Note that although the steps in MAkE have some similarity with those in NSGA-II, any other existing multiobjective genetic algorithms could have been used as the underlying MOO technique in MAkE. The pseudocode of the proposed MAkE is provided in Fig. 5.

#### 5.1. Representation/encoding scheme

A chromosome is an array of \( n \) integers where \( n \) is the total number of points on the 2D plane representing the co-ordinates of flip-flops in a given placement. The allele value at the \( i \)th position of a chromosome represents the index of the partition, to which the \( i \)th point belongs. A chromosome represents a particular way of partitioning the given point set \( P \) into \( k \) equipartitions as depicted
in Fig. 6. The chromosome can be looked upon as an arrangement of the \( k \) partition indices, each index having \( \left\lfloor \frac{n}{k} \right\rfloor \) copies.

5.2. Initialization

The initialization of each chromosome is done as follows:

1. randomly select a point \( p \) whose partition index has not been assigned yet;
2. assign the same partition index to \( p \) and its \( \left\lfloor \frac{n}{k} \right\rfloor - 1 \) nearest neighbors;
3. repeat steps 1 and 2 \( (k-1) \) times.

5.3. Genetic operators

The crossover and mutation operators designed for this particular problem are as follows:

5.3.1. Crossover

The crossover operator used here is inspired by order crossover operator [23]. A pair of strings are chosen as parents and two offsprings are constructed by choosing a substring from one parent and preserving the relative order of the partition indices in the other parent string. Suppose there are \( n = 6 \) points, and \( k = 3 \) partitions. Let us consider the following parent chromosomes: \((1 2 3 1 2 3)\) and \((3 3 1 1 2 2)\), and suppose that we select a first cut point between the second and the third positions, and a second one between the fourth and the fifth positions. The parent chromosomes may be re-written as \((1 2 3 1 2 3)\) and \((3 3 1 1 2 2)\).

The offsprings are created in the following way. First, the substring in each parent string between the cut positions is copied into the corresponding offspring, which gives \((**31**)\) and \((**11**)\). Next, starting from the second cut point of one parent, the rest of the integers are copied in the order in which they appear in the other parent, and omitting a partition index which now already contains \((n/k)\) points. For each partition, there is a counter for the number of points for which it is being assigned in the offspring string. When the end of the parent string is reached, we continue cyclically from its first position. In our example, the following children are obtained: \((3 1 3 1 2 2)\) and \((2 3 1 1 2 3)\). This crossover operation is also demonstrated in Fig. 7.

5.3.2. Mutation

The design of our mutation operator is based on displacement mutation operator [23]. First a substring of the parent chromosome is selected at random. This substring is removed from the original chromosome and inserted at a random position. For example, consider the chromosome \((1 2 3 1 2 3)\), and suppose that the substring \((3 1)\) is selected. After the removal of this substring, we have \((1 2 2 3)\).

Next, we randomly select position three in this remaining string, at which the selected subset \((3 1)\) is inserted. This results in \((1 2 2 (3 1) 3)\), as shown in Fig. 8.

5.4. Compaction

A new greedy operator called compaction is introduced to compact the partitions whenever possible, thereby accelerating
the convergence of MAkE. This operator is applied on every chromosome with a small specific probability, (named probability of compaction \( p_{\text{comp}} \)). It works as follows:

1. for each of the \( k \) partitions, initialize two counters \( \text{InCount}[i] \) and \( \text{OutCount}[i] \) to 0, \( \forall i = 1, \ldots, k \);
2. let the index of the partition whose diameter is maximum be \( i \), and the farthest point from \( p_x \), the centre of partition \( i \), be far;
3. find \( p_x \), the nearest neighbor of far, in \( P \) and let \( k_1 \) be the index of the partition to which \( p_x \) belongs presently;
4. if \( (k_1 \neq i) \) then move point far, to partition \( k_1 \);
5. increment \( \text{InCount}[k_1] \) and \( \text{OutCount}[i] \) by 1;
6. repeat steps 2 to 6 with \( i = k_1 \) until \( \text{InCount}[j] = \text{OutCount}[j] \) \( \forall j = 1, \ldots, k \).

5.5. Time complexity

In the proposed multiobjective optimization technique, MAkE, the length of a chromosome is equal to \( n \), the total number of points in the data set. For the initialization step and the compaction operator, the distances between all pairs of points need to be computed before hand. This takes \( O(n^2) \) time. Initialization step requires \( O(n) \) computational complexity. MAkE employs NSGA-II whose complexity is \( O(M \cdot N^2) \) where \( M \) is the total number of objective functions and \( N \) denotes the size of the population. This comprises the complexity due to its nondominated sorting and the crowding distance operator. Nondominated sorting of a population size of \( 2 \cdot N \) requires at most \( O(M \cdot N^2) \) computations. The crowded tournament selection requires the crowding distance computation of the complete population \( P_{t+1} \) of size \( N \). This requires \( O(M \cdot N \cdot \text{log}(N)) \) computations. Thus the overall complexity of NSGA-II is at most \( O(M \cdot N^2) \).

The crossover and mutation operations of MAkE requires \( O(n) \) time. Compaction operator requires \( O(n) \) time for each chromosome. The computational complexity of objective function calculations for each chromosome is also \( O(n) \).

Thus, the total time complexity of MAkE is \( O((M \cdot N^2 + 4n) \cdot \text{totalgen} + n^3) \).

6. Inheritance in crossover and mutation

One of the important metrics for a genetic operator can be the degree of inheritance in an offspring produced by it [24]. We derive below the degree of inheritance for the crossover and mutation operators defined in Section 5.3.

6.1. Degree of inheritance due to crossover

For the crossover operator defined in the previous section, the role of the length \( l \) and the starting index pos of the substring chosen, on the degree of inheritance (DOI) in the offsprings from parents \( P_1 \) and \( P_2 \) is analyzed, where (DOI) in an offspring off after the above crossover operation is defined as

\[
\text{DOI}(P_1, P_2) = \max \left( \text{DOI}(P_1), \text{DOI}(P_2) \right)
\]

\[
\text{DOI}(P_2) = |S_i, S = \{i|\text{off}[i] = P_2[i]\}|, a = 1.2. \text{ After crossover, off}[i] = P_1[i], \forall i = \text{pos}, \ldots, (\text{pos} + (l - 1)). \text{ Thus, DOI}(P_1) = l.
\]

The remaining positions in off, starting from pos + l and going cyclically up to pos − 1, are filled in by the entries in \( P_2 \) starting from position pos + 1, in the order in which they appear in \( P_2 \).

As an illustration, consider the case when \( l = 1 \). Here DOI\((P_1) = 1 \). Let \( P_1[\text{pos}] = k \). DOI\((P_2) \) depends on the position of the last appearance of \( k \) in \( P_2 \) in the cyclic order starting from \( k \).

- Best case: \( P_2[\text{pos} + 1] = k \) Here DOI\((P_2) = (n - 2) \) (Fig. 9(a)).
- Average case: If \( P_2[i] = k, \forall i = (\text{pos} + 1), \ldots, (\text{pos} + \lceil \frac{n}{k} \rceil) \) mod \( (n - (\text{pos} + l - 1)) \), and every other partition also has a run of \( \lceil \frac{n}{k} \rceil \) (Fig. 9(b)), then DOI\((P_2) = (\lceil \frac{n}{k} \rceil - 1) \cdot k \).
- Worst case: If \( P_2[i] = k, \forall i = (\text{pos} + 1) \) to \( (\text{pos} + \lceil \frac{n}{k} \rceil) \) mod \( (n - (\text{pos} + l - 1)) \), and no other partition has any run \( \geq 2 \) (Fig. 9(c)), then DOI\((P_2) = (\lceil \frac{n}{k} \rceil - 1) \).

In general, the value of \( l \) can lie between 1 and \( \lfloor \frac{n}{2} \rfloor \). Let the substring of length \( l \) in \( P_1 \) contains \( k \) unique partition indices \( i_1, i_2, \ldots, i_k \), and let the number of points in partition \( i_j \) be denoted by \( n_{i_j} \). Note that \( n_{i_1} + n_{i_2} + \ldots + n_{i_k} = l \).
Then DOI($P_2$) depends on how far in the cyclic order the $\lfloor (n/k) \rfloor$th point of any of these $k'$ partitions appear, starting from the $(pos+l)$th position in $P_2$. The three cases are:

1. **Best case:** When $\lfloor (n/k) \rfloor$th point of any partition which appeared in the substring occurs at $(pos-k'+\lfloor (n/k) \rfloor + n_1 + n_2 + \ldots + n_{k'})$ position in $P_2$, then
   \[ DOI(P_2) = (n - n_1 - n_2 - \ldots - n_{k'}) = (n - l). \]

2. **Worst case:** If $\lfloor (n/k) \rfloor$ points of any partition $i_j, j = 1$ to $k'$ occur in a run from the $(pos+l)$ th position in $P_2$, and no other partition among these $k'$ partitions has a run $(\geq 2)$, then
   \[ DOI(P_2) = \lfloor (n/k) \rfloor - n_i. \]

3. **Average case:** If $\lfloor (n/k) \rfloor$ number of points of all the $k'$ partitions occur in a run consecutively starting from $(pos+l)$ th position in $P_2$ then in this case
   \[ DOI(P_2) = (\lfloor (n/k) \rfloor - n_{\max})k' \]
   where $n_{\max} = \max(n_1, \ldots, n_{k'})$.

### 6.2. Degree of inheritance due to mutation

Suppose the length of the substring is $l$. At most $2+l$ number of positions can be changed in the offspring, so DOI due to mutation is $(n - 2l)$. 

![Fig. 9. Inheritance in crossover for $n = 9, k = 3, l = 1$ (a) Best case, (b) average case, and (c) worst case.](image)
The DOI of both the operators is within a reasonable range. While low DOI would cause the search space to be explored in a haphazard manner with many sharp changes and longer convergence, high DOI is likely to diminish diversity and the scope of covering the entire search space.

7. Experimental results

Our proposed method has been implemented in C on a 1.6 GHz PIV Linux machine and has been applied on two categories of data sets.

- Category 1: Synthetic point sets.
- Category 2: Data sets or point sets from layouts of benchmark VLSI circuits.

For category 1, each of the three data sets have been generated by first creating \( k \) overlapping boxes, either rectangular or square in shape, and then randomly generating \( n/k \) points within each box. These data sets are described in Table 1.

7.1. Results on synthetic point sets

In order to show a detailed sensitivity study of the parameters of the proposed MAkE, it has been applied on the three synthetic data sets with 5 different parameter settings. For all these five settings the population size is kept equal to 100. For Data1 and Data2, the total number of generations is kept equal to 200 for all these five settings. For Data3, the total number of generations is kept equal to 1200. The five settings of GA parameters ( \( p_c \): probability of crossover; \( p_m \): probability of mutation; \( p_{comp} \): probability of compaction) are as follows:

1. Setting 1: \( p_c = 0.9, p_m = 0.2, p_{comp} = 0.6 \).
2. Setting 2: \( p_c = 0.9, p_m = 0.2, p_{comp} = 0.4 \).
3. Setting 3: \( p_c = 0.9, p_m = 0.1, p_{comp} = 0.4 \).
4. Setting 4: \( p_c = 0.8, p_m = 0.1, p_{comp} = 0.4 \).
5. Setting 5: \( p_c = 0.8, p_m = 0.1, p_{comp} = 0.0 \).

Setting 5 is used to show the effect of the compaction operator on the convergence of the proposed MAkE. With the five different parameter settings, five Pareto optimal sets are obtained for each data set. The two performance metrics, Purity and MinimalSpacing (discussed in Section 4.2.1) are computed in order to compare these five sets of Pareto-optimal solutions. Table 2 shows the minimum skew value over a particular set of Pareto-solutions and the corresponding values of Purity and MinimalSpacing for the three synthetic data sets. It is observed from Table 2 that in general, MAkE performs well with Setting 2 for the above mentioned three synthetic data sets. For Data1, MAkE with Setting 4 provides the best Purity, MinimalSpacing and minimum skew values. MAkE with Setting 3 gives the best performance in terms of Purity and MinimalSpacing for Data2, whereas it attains the minimum skew value with Setting 1. Further, for this particular data set, MAkE with Setting 2 also performs the second best in terms of Purity but the MinimalSpacing values with both Setting 2 and Setting 3 are the same. For Data3, the largest synthetic data set, MAkE with Setting 2 provides the best Purity, MinimalSpacing and Minimum Skew values.

In MOO, algorithms produce a large number of nondominated solutions [20] on the final Pareto-optimal front. Each of these solutions provides a way of partitioning the given data set. MAkE also produces a large number of solutions on its final Pareto-optimal front. Here we have chosen the solution which has the minimum skew value from the final Pareto-optimal front after application of MAkE on all synthetic data sets used here for experiment. The partitions corresponding to the solutions with minimum skew value obtained with different parameter settings for Data1 are shown in Fig. 10(a) and (b), respectively. Figs. 11 and 12 show the point sets in the partitions corresponding to the solutions with minimum skew value for Data2 and Data3, respectively. Here points in different partitions are indicated by different symbols. After carefully inspecting Fig. 10(a) and (b), it is inferred that for the set Data1 in the partitioning corresponding to Setting 4, the individual equipartitions are more non-overlapping (Fig. 10(b)) rather than that corresponding to Setting 1/Setting 2/Setting 3 (Fig. 10(a)). These observations are also in sync with the minimum skew values associated with these two partitions (refer to Table 2). Again for the set Data2 the equipartitions obtained by MAkE with Setting 1 (Fig. 11(a)) are more non-overlapping rather than those obtained by Setting 2/Setting 3/Setting 4 (Fig. 11(b)). These are again supported by the skew values corresponding to these partitions (refer to Table 2). However for the set Data3, the partitionings obtained by MAkE for all the parameter settings are almost the same; very close scrutiny reveals that there are more overlaps in the equipartitions corresponding to the partitioning obtained with Setting 4 (Fig. 12(b)) than those obtained with Setting 1/Setting 2/Setting 3 (Fig. 12(a)).

Note that MAkE with Setting 5 performs poorly for all the synthetic data sets used here for experiment in terms of purity. Thus, it signifies that without compaction operator MAkE did not converge within the specified number of generations.

7.2. Results on data sets from layouts of benchmark VLSI circuits

Our proposed method has been successfully tested on ten benchmark circuit data sets. The description of these benchmark

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>( n )</th>
<th>( k )</th>
<th>Shape of bounding box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data1</td>
<td>250</td>
<td>5</td>
<td>Rectangular (overlapping)</td>
</tr>
<tr>
<td>Data2</td>
<td>250</td>
<td>5</td>
<td>Square (overlapping)</td>
</tr>
<tr>
<td>Data3</td>
<td>1000</td>
<td>5</td>
<td>Square (overlapping)</td>
</tr>
</tbody>
</table>

Table 1

Synthetic data sets for \( k \)-way equipartitioning of \( n \) points.
Fig. 10. Partitioning obtained by MAkE on Data1 corresponding to minimum skew value \( f_4 \) with parameter (a) \( p_c = 0.9; \) Setting 1 \( (p_m = 0.2, p_{\text{comp}} = 0.6) \), Setting 2 \( (p_m = 0.2, p_{\text{comp}} = 0.4) \) and Setting 3 \( (p_m = 0.1, p_{\text{comp}} = 0.4) \); (b) Setting 4 \( (p_c = 0.8, p_m = 0.1, p_{\text{comp}} = 0.4) \). Here \( k = 5 \) is indicated by the 5 distinct symbols \( \times, \circ, +, \square, \ast \).

Fig. 11. Partitioning obtained by MAkE on Data2 corresponding to minimum skew value \( f_4 \) with parameter (a) Setting 1 \( (p_c = 0.9, p_m = 0.2, p_{\text{comp}} = 0.6) \); (b) Setting 2 \( (p_c = 0.9, p_m = 0.2, p_{\text{comp}} = 0.4) \), Setting 3 \( (p_c = 0.9, p_m = 0.1, p_{\text{comp}} = 0.4) \), and Setting 4 \( (p_c = 0.8, p_m = 0.1, p_{\text{comp}} = 0.4) \). Here \( k = 5 \) is indicated by the 5 distinct symbols \( \times, \circ, +, \square, \ast \).

Fig. 12. Partitioning obtained by MAkE on Data3 corresponding to minimum skew value \( f_4 \) with parameter (a) \( p_c = 0.9; \) Setting 1 \( (p_m = 0.1, p_{\text{comp}} = 0.4) \), Setting 2 \( (p_m = 0.2, p_{\text{comp}} = 0.4) \) and Setting 3 \( (p_m = 0.1, p_{\text{comp}} = 0.4) \); (b) Setting 4 \( (p_c = 0.8, p_m = 0.1, p_{\text{comp}} = 0.4) \). Here \( k = 5 \) is indicated by the 5 distinct symbols \( \times, \circ, +, \square, \ast \).
circuit data sets appear in Table 3. For each data set the algorithm is executed for two different values of $k$, namely, (i) $k = \log(n)$, and (ii) $k = \sqrt{n}$. For MAkE, (as established from the results on the above mentioned synthetic data sets that MAkE with Setting 2 in general performs well) the parameter values are kept as follows: $N = 100$, $p_m = 0.2$, $p_c = 0.8$, and $p_{\text{comp}} = 0.4$. For data sets of size below 200, the total number of generations is set equal to 500, but for the data set s15850 MAkE is executed for 1200 generations. A set of nondominated solutions are obtained for every data set after application of MAkE. Table 3 shows the total number of nondominated solutions, and the MinimalSpacing values obtained after application of MAkE on the above mentioned benchmark data sets. The total number of fitness functions evaluated and the total CPU time taken by MAkE for all the benchmark data sets are also shown in Table 3, the maximum time being less than 36 seconds for all data sets.

For the purpose of illustration, the Pareto-optimal fronts obtained by MAkE for s298, s4863, s15850 projected on different 3-objective spaces are shown in Figs. 13–15, respectively. These figures show the spreadings of the Pareto-optimal solutions over different objective spaces. It is evident that spreading of the Pareto-optimal solutions vary along different objective spaces. From Fig. 13(a), it is inferred that the most diverse set of Pareto-optimal solutions is obtained while optimizing the tuple $f_1, f_2, f_3$ for s298. For s4863, optimizing the tuple $f_1, f_2, f_3$ for $k = 7$ provides more spreading over the Pareto-optimal front (refer to Fig. 14(a)).

### Table 3

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$n$</th>
<th>$k$</th>
<th>MinimalSpacing</th>
<th>Distinct Pareto solutions</th>
<th>Fitness evaluations</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s298</td>
<td>15</td>
<td>4</td>
<td>0.13</td>
<td>54</td>
<td>50,000</td>
<td>5</td>
</tr>
<tr>
<td>s713</td>
<td>20</td>
<td>5</td>
<td>0.13</td>
<td>46</td>
<td>50,000</td>
<td>5</td>
</tr>
<tr>
<td>s953</td>
<td>30</td>
<td>5</td>
<td>0.24</td>
<td>9</td>
<td>50,000</td>
<td>8</td>
</tr>
<tr>
<td>s1269</td>
<td>38</td>
<td>6</td>
<td>0.18</td>
<td>21</td>
<td>50,000</td>
<td>9</td>
</tr>
<tr>
<td>s1494</td>
<td>8</td>
<td>3</td>
<td>0.24</td>
<td>6</td>
<td>50,000</td>
<td>3</td>
</tr>
<tr>
<td>s1512</td>
<td>58</td>
<td>6</td>
<td>0.18</td>
<td>20</td>
<td>50,000</td>
<td>14</td>
</tr>
<tr>
<td>s3271</td>
<td>117</td>
<td>7</td>
<td>0.23</td>
<td>32</td>
<td>50,000</td>
<td>16</td>
</tr>
<tr>
<td>s4863</td>
<td>105</td>
<td>7</td>
<td>0.18</td>
<td>74</td>
<td>50,000</td>
<td>15</td>
</tr>
<tr>
<td>s9234</td>
<td>161</td>
<td>8</td>
<td>0.19</td>
<td>43</td>
<td>50,000</td>
<td>20</td>
</tr>
<tr>
<td>s15850</td>
<td>617</td>
<td>10</td>
<td>0.28</td>
<td>27</td>
<td>120,000</td>
<td>35</td>
</tr>
</tbody>
</table>

**Fig. 13.** Pareto front obtained by MAkE for s298 data set with $k = 4$ (a) projected on the $f_1, f_2, f_3$ objective space (b) projected on $f_1, f_2, f_4$ objective space (c) projected on $f_1, f_3, f_4$ objective space (d) projected on $f_2, f_3, f_4$ objective space.
Again for s4863 with $k = 11$ optimizing the tuple $(f_1, f_2, f_3)$ provides more spreading over the Pareto-optimal front (refer to Fig. 14(e)). From Fig. 15(d) and (h), it is evident that the partitioning on s15850 is more sensitive to $f_4$. Thus we can infer the sensitivity of a data set to a subset of objective functions from these figures. This analysis helps in choosing the appropriate subset of objective functions for a particular data set, if further reduction of complexity and run time of the algorithm is in demand.

Further, in order to show the effectiveness of multiobjective optimization, MAkE has been applied on the above mentioned data sets with the weighted sum of all the four objective functions, $f_1, f_2, f_3, f_4$ as a single objective function. This algorithm is named as kEPA (k-way Equipartitioning technique), kEPA optimizes a single objective function, $f = \sum_{i=1}^{\text{total obj}} w_i f_i$, where $w_i = \frac{1}{\text{total obj}}$, total obj = 4 and

$$f_1 = \frac{f_1}{A}, \quad f_2 = \frac{f_2}{A}, \quad f_3 = \frac{f_3(k)}{k}, \quad f_4 = \frac{f_4}{\max_{i=1..n} d(c, p_i) - \min_{i=1..n} d(c, p_i)}.$$

Here $c$ is the centre and $A$ is the area of the bounding box of the point set $P$. The new set of objective functions $f'_i, i = 1 \ldots 4$ are formulated by normalizing the original four objective functions.
kEPA provides one single solution. In order to compare the results obtained by the single objective optimization and the multi-objective optimization, the original four objective function values are computed corresponding to the final partitioning provided by kEPA. The purity measure is then calculated on the combined results of kEPA and MAkE, reported in Table 4. The values of minimum skew obtained by kEPA and MAkE, are also reported in Table 4. It is evident from this table that except s298, for all the data sets the four tuple provided by kEPA is dominated by the Pareto-optimal solutions provided by MAkE. kEPA also attains larger values of minimum skew compared to MAkE for all the data sets.

Next, experiments are carried out in order to establish that MAkE optimizing four objective functions is better than that for MAkE optimizing any two/three subsets of the four objective functions. The purity measure and MinimalSpacing measure over the set of Pareto-optimal solutions for each benchmark data set are reported in Table 5. It can be seen from the above mentioned results that MAkE optimizing four objectives is, in general, more effective than MAkE optimizing the other ten different combinations. However, for few of the data sets, results show that, certain subsets of the 4-objective functions provide better results than that optimizing all 4-objective functions. For example, as is evident
from Table 5, the results of MAkE for the data set s953 and k = 5, the function subsets \( \{f_1, f_2, f_3\} \), \( \{f_1, f_4\} \), \( \{f_2, f_3\} \), \( \{f_1, f_3\} \) and \( \{f_1, f_2\} \) provide higher purity values than that of optimizing all four objective functions. The corresponding MinimalSpacing values are also very close. Thus though in this paper, all the four objective functions are optimized by MAkE, depending on the data sets used, one may select a proper subset of these 4-objective functions to expedite the algorithm without sacrificing the quality too much.

### Table 5

Effect of optimizing different subsets of the four objective functions with MAkE on Purity (Pu) and MinimalSpacing (MS) of Pareto-optimal fronts for n points from layouts of benchmark VLSI circuits.

<table>
<thead>
<tr>
<th>Data set</th>
<th>k</th>
<th>Measure</th>
<th>Function subset</th>
<th>Pu</th>
<th>MinimalSpacing</th>
</tr>
</thead>
<tbody>
<tr>
<td>s298</td>
<td>4</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.926</td>
<td>514.368467</td>
</tr>
<tr>
<td>s713</td>
<td>5</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.961</td>
<td>514.368467</td>
</tr>
<tr>
<td>s953</td>
<td>5</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.989</td>
<td>514.368467</td>
</tr>
<tr>
<td>s1269</td>
<td>6</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.890</td>
<td>514.368467</td>
</tr>
<tr>
<td>s1494</td>
<td>7</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.830</td>
<td>514.368467</td>
</tr>
<tr>
<td>s4863</td>
<td>7</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.790</td>
<td>514.368467</td>
</tr>
<tr>
<td>s9234</td>
<td>8</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.740</td>
<td>514.368467</td>
</tr>
<tr>
<td>s15850</td>
<td>10</td>
<td>Pu</td>
<td>( {f_1, f_3, f_4} )</td>
<td>0.710</td>
<td>514.368467</td>
</tr>
</tbody>
</table>

### Table 6

Comparison between k-means and MAkE optimizing 4 objective functions for different benchmark circuits with respect to the Best Skew values and Purity measurement.

<table>
<thead>
<tr>
<th>Benchmark circuit</th>
<th>k</th>
<th>Purity</th>
<th>Minimal skew (( f_4 ))</th>
<th>k-means</th>
<th>MAkE</th>
</tr>
</thead>
<tbody>
<tr>
<td>s298</td>
<td>4</td>
<td>0.00</td>
<td>1.00</td>
<td>4.950186</td>
<td>16.24949</td>
</tr>
<tr>
<td>s713</td>
<td>5</td>
<td>0.00</td>
<td>1.00</td>
<td>79.7656</td>
<td>50.300451</td>
</tr>
<tr>
<td>s953</td>
<td>5</td>
<td>0.00</td>
<td>1.00</td>
<td>112.2716</td>
<td>71.026423</td>
</tr>
<tr>
<td>s1269</td>
<td>6</td>
<td>0.00</td>
<td>1.00</td>
<td>121.8353</td>
<td>80.953487</td>
</tr>
<tr>
<td>s1494</td>
<td>7</td>
<td>0.00</td>
<td>1.00</td>
<td>107.7914</td>
<td>72.857929</td>
</tr>
<tr>
<td>s4863</td>
<td>7</td>
<td>0.00</td>
<td>1.00</td>
<td>149.5869</td>
<td>105.480793</td>
</tr>
<tr>
<td>s9234</td>
<td>8</td>
<td>0.00</td>
<td>1.00</td>
<td>126.2388</td>
<td>97.694893</td>
</tr>
<tr>
<td>s15850</td>
<td>10</td>
<td>0.00</td>
<td>1.00</td>
<td>298.1814</td>
<td>248.388</td>
</tr>
<tr>
<td>s3271</td>
<td>7</td>
<td>0.00</td>
<td>1.00</td>
<td>209.1068</td>
<td>208.127449</td>
</tr>
<tr>
<td>s863</td>
<td>7</td>
<td>0.00</td>
<td>1.00</td>
<td>358.8493</td>
<td>238.295005</td>
</tr>
<tr>
<td>s953</td>
<td>5</td>
<td>0.00</td>
<td>1.00</td>
<td>298.1814</td>
<td>248.388</td>
</tr>
<tr>
<td>s1269</td>
<td>6</td>
<td>0.00</td>
<td>1.00</td>
<td>440.0096</td>
<td>251.5267</td>
</tr>
<tr>
<td>s1494</td>
<td>7</td>
<td>0.00</td>
<td>1.00</td>
<td>358.8493</td>
<td>238.295005</td>
</tr>
<tr>
<td>s4863</td>
<td>7</td>
<td>0.00</td>
<td>1.00</td>
<td>316.3306</td>
<td>370.9284</td>
</tr>
<tr>
<td>s9234</td>
<td>8</td>
<td>0.00</td>
<td>1.00</td>
<td>655.6959</td>
<td>799.4235</td>
</tr>
<tr>
<td>s15850</td>
<td>10</td>
<td>0.00</td>
<td>1.00</td>
<td>701.4056</td>
<td>784.312515</td>
</tr>
</tbody>
</table>

### 7.3. Comparison with k-means, an existing popular partitioning technique

Experiments are also carried out to establish that the proposed MAkE for equi-partitioning a VLSI data set is much more effective than that of the well-known partitioning technique where no equi-partition constraint is imposed. Here, the well-known k-...
means [5] algorithm is executed ten times for each of the above mentioned ten data sets from layouts of benchmark circuits. For each data set, k-means is executed with two different values of $k$, (1) $k = \log n$ and (2) $k = \sqrt{n}$. The four objective values of the obtained partitions over the ten runs are calculated. Then the purity measure is computed between the solution sets provided by k-means and the Pareto front provided by MAkE. These purity measures for all the data sets are also reported in Table 6. To compare the performance of $k$-means with that of MAkE, the best overall skew values, which is concerned for the clock tree design, obtained by both of them for each data set are also reported in Table 6. It is evident from Table 6 that except the s15850 data set, the use of equipartitioning is effective for forming clock tree with minimum skew, thus establishing the efficacy of our proposed method MAkE.

8. Conclusion

In this paper we have discussed a class of applications which involve identifying partitions across a set of physical elements. These applications have the characteristic feature that apart from the traditional requirement of equipartition, several other objectives have to be optimized. A generic formulation for these class of problems is provided, and a GA-based multiobjective optimization algorithm is proposed. The proposed algorithm is tested in one of the application domains, viz., in identifying partitions of flip-flops to aid clock tree synthesis in low power design of nanometer integrated circuits. Results on several synthetic data sets and benchmark circuits from the CAD VLSI domain show that multiobjective optimization by genetic algorithm solves the problem effectively.

Possible future extensions of the work include the analysis of the performance of the proposed algorithm, and computations of the actual reduction in power dissipation of the clock trees generated for the CAD VLSI problem. Moreover, application of the proposed algorithm in the other domains discussed in the paper are to be considered. Incorporating other objective functions into the formulation is another direction to be explored. Extending the proposed MAkE in order to evolve the proper value of number of equipartitions for a particular data set is to be addressed.

Acknowledgements

The authors gratefully acknowledge the comments of the Editor and the anonymous reviewers which helped them in improving the quality of the paper. The authors are also thankful to Sandeep K. Dey, Indian Statistical Institute for helping with extraction of data from layouts of VLSI benchmark circuits.

References