On Constrained Minimum Vertex Covers of Bipartite Graphs: Improved Algorithms[⋆]

Jianer Chen and Iyad A. Kanj

Department of Computer Science, Texas A&M University,
College Station, TX 77843-3112, USA
Email: {chen,iakanj}@cs.tamu.edu

Abstract. The constrained minimum vertex cover problem on bipartite graphs arises from the extensively studied fault coverage problem for reconfigurable arrays. In this paper, we develop a new algorithm for the problem, in which classical results in matching theory and recently developed techniques in parameterized computation are nicely combined and extended. The algorithm is practically efficient with running time bounded by $O(1.26^k + kn)$, where k is the size of the constrained minimum vertex cover in the input graph. The algorithm is a significant improvement over the previous algorithms for the problem.

1 Introduction

As the density of VLSI chips increases, the probability of introducing defects on the chips during the fabrication process also increases. A number of reconfiguration strategies have been proposed. In particular, algorithms for repair of reconfigurable arrays using spare rows and columns have received considerable attention [7, 9, 16, 17]. A typical reconfigurable array consists of a rectangular array plus a set of spare rows and spare columns. A defective element is repaired by replacing the row or the column containing the element with a spare row or a spare column. In general, the number of spare rows and columns is much smaller compared with the total number of rows and columns in the reconfigurable array, and the cost of reconfigurating an array is proportional to the number of replacement rows and columns. Therefore, it is desirable to use the minimum number of spare rows and columns to repair the defective elements, under the constraint that the numbers of replacement spare rows and columns do not exceed the total numbers of the available spare rows and columns, respectively.

Formally, let A be an $n \times m$ rectangular reconfigurable array. A *fault covering* of A consists of a set S_r of rows and a set S_c of columns in A such that every defective element in A is either in a row in S_r or in a column in S_c (or both). The fault covering is *minimum* if $|S_r| + |S_c|$ is the minimum. Suppose the array A is also provided with k_r spare rows and k_c spare columns. Then the problem is to either find a minimum fault covering (S_r, S_c) of A with $|S_r| \leq k_r$ and $|S_c| \leq k_c$, or report no such fault covering exists.

[⋆] This work is supported in part by NSF under Grant CCR-0000206.

This problem can be easily converted to the *constrained minimum vertex cover problem on bipartite graphs*, (shortly MIN-CVCB), as follows. Let $G_A = (U \cup L, E)$ be a bipartite graph with $U = \{u_1, \ldots, u_n\}$ and $L = \{v_1, \ldots, v_m\}$, such that there is an edge between u_i and v_j if and only if the (i, j) element in the array A is defective. It is fairly easy to see that a vertex set K of k_1 vertices in U and k_2 vertices in V is a vertex cover of the graph G_A if and only if the corresponding k_1 rows and k_2 columns make a fault covering of the array A. Therefore, the problem can be formally stated as follows:

MIN-CVCB
Given a bipartite graph $G = (U \cup L, E)$ and two integers k_u and k_l, either find a minimum vertex cover of G with at most k_u vertices in U and at most k_l vertices in L, or report no such a vertex cover exists.

An extensive list of algorithms has been published in the literature for the MIN-CVCB problem and its variations (e.g., [7, 9, 16, 17]). Most of these algorithms are heuristic and have no guaranteed performance. An algorithm proposed in [7] introduced the concept of *critical set* and used branch-and-bound technique based on the A^* algorithm in Artificial Intelligence [14]. No explicit analysis was given in [7] for this algorithm, but it is not hard to see that in the worst case the algorithm running time is at least $\Theta(2^{k_u+k_l} + m\sqrt{n})$. Experimental results show that this algorithm is much more favourable than previous algorithms and the running time is moderate for practical instances for the MIN-CVCB problem.

In the current paper, we develop simpler and more efficient algorithms for the MIN-CVCB problem. Classical results in matching theory and recently developed techniques in parameterized computation are nicely combined and extended. In particular, we observe that the classical Gallai-Edmonds Structure Theorem [8] applied to bipartite graphs will allow us to concentrate on the MIN-CVCB problem on bipartite graphs with perfect matching. In fact, Gallai-Edmonds Structure Theorem is a stronger (and more systematical) version of the critical set theory developed in [7]. Gallai-Edmonds Structure Theorem also provides a solid basis so that the recently developed technique in parameterized computation, *reduction to problem kernel*, can be nicely applied. We then use the classical Dulmage-Mendelsohn Decomposition [8] to decompose a bipartite graph with perfect matching into elementary bipartite subgraphs. This decomposition makes the other technique, *bounded search tree*, in parameterized computation become much more effective. Combining all these enables us to derive an $O(1.26^k + kn)$ time algorithm for the MIN-CVCB problem, where k is the size of a minimum vertex cover in the input graph. To see how significant this improvement is over the previous best algorithm [7], take a typical reconfigurable array of size 1024×1024 with 20 spare rows and 20 spare columns. A simple estimation shows that our algorithm is more than 10^8 times faster!

We point out that our results are also of theoretical interests in the current study of "efficient" exponential time algorithms for NP-hard problems [2, 4, 15], in particular they make a nice contribution to the study of parameterized algorithms for the minimum vertex cover problem, which has drawn much attention recently [1, 3, 12]. A recently developed algorithm by Fernau and Niedermeier

[6] has running time $O(1.40^k + kn)$ and constructs a constraint (not necessarily minimum) vertex cover for bipartite graphs. This algorithm could also be used to solve the MIN-CVCB problem, but our algorithm is significantly faster and much simpler using elegant results in graph theory (instead of lengthy case-by-case combinatorial enumerations).

2 Reduction to kernel by GE-Theorem

A vertex set K in a graph G is a *vertex cover* if every edge in G has at least one end in K. A graph $G = (V, E)$ is *bipartite* if its vertex set V can be bipartitioned into two sets U (the "upper part") and L (the "lower part"), such that every edge in G has one end in U and the other end in L. A bipartite graph is written as $G = (U \cup L, E)$ to indicate the vertex bipartition. The vertex sets U and L are called the *U-part* and the *L-part* of the graph G. A vertex is a *U-vertex* (resp. an *L-vertex*) if it is in the U-part (resp. the L-part) of G. A *matching M* in a graph G is a set of edges in G such that no two edges in M share a common end. A vertex v is *matched* if it is an end of an edge in M and *unmatched* otherwise. The matching M is *perfect* if all vertices in G are matched. It is well-known that for a bipartite graph, the number of edges in a maximum matching is equal to the number of vertices in a minimum vertex cover [8]. Given a matching M in a graph G, an *alternating path* (with respect to M) is a simple path $\{v_1, v_2, \ldots, v_r\}$ such that v_1 is unmatched and the edges $[v_{2k}, v_{2k+1}]$ are in M for all k. An alternating path is an *augmenting path* if it has odd length and both endpoints are unmatched. A matching M in a graph G is maximum, if and only if there is no augmenting path in G with respect to M [8]. A maximum matching of a graph with n vertices and m edges can be constructed in time $O(m\sqrt{n})$ [10], and for a bipartite graph G, a minimum vertex cover of G can be constructed from a maximum matching in G in linear time [8].

We start with a theorem, which answers an open problem posed in [7].

Theorem 1. *The* MIN-CVCB *problem is NP-complete.*

We now show the classical Gallai-Edmonds Structure Theorem (shortly, GE-Theorem) [8] can be applied to significantly reduce the size of the MIN-CVCB problem. Let $G = (V, E)$ be a bipartite graph. For a subset S of vertices in G, let $G(S)$ be the subgraph induced by S. Let D be the set of vertices in G that are unmatched in at least one maximum matching, let A be the set of vertices in $V - D$ that are adjacent to a vertex in D, and let $C = V - D - A$.

Proposition 1. (GE-Theorem) *Let G be a bipartite graph with the sets D, A, and C defined above. Then* (1) *D is an independent set in which no vertex is contained in any minimum vertex cover for G;* (2) *A is the intersection of all minimum vertex covers for G;* (3) *the subgraph $G(C)$ has a perfect mathing.*

The sets D, A, and C can be constructed via a maximum matching M in the bipartite graph G, as follows. Let W be the set of unmatched vertices under

M. Define $\overline{W}$ to be the set of vertices such that a vertex v is in $\overline{W}$ if and only if v is reachable from a vertex in W via an alternating path of even length. By the definition, $W \subseteq \overline{W}$. Since exchanging the edges in M and $E - M$ along any alternating path from a vertex v' in W to a vertex v in $\overline{W}$ will result in a maximum matching that does not contain v, we have $\overline{W} \subseteq D$. Now for any vertex w not in $\overline{W}$, we know $[w, w']$ is an edge in M for some vertex w'. Since w is not in $\overline{W}$, w' is not connected from any vertex in W by an alternating path of odd length. Therefore, after removing w from the graph G, there is no augmenting path in $G - \{w\}$ with respect to the matching $M - [w, w']$. Thus, a maximum matching in the graph $G - \{w\}$ contains $|M| - 1$ edges. This shows that the vertex w must be in every maximum matching in the graph G so w is not in D. In consequence, the set $\overline{W}$ is exactly the set D in Proposition 1. With the set D known, the sets A and C can be constructed easily.

Theorem 2. *The time for solving an instance $\langle G; k_u, k_l \rangle$ of* MIN-CVCB, *where G is a bipartite graph of n vertices and m edges, is $O(m\sqrt{n} + t(k_u + k_l))$, where $t(k_u + k_l)$ is the time for solving an instance $\langle G'; k'_u, k'_l \rangle$ for* MIN-CVCB, *where $k'_u \leq k_u$, $k'_l \leq k_l$, and G' has a perfect matching.*

Proof. Let $G = (U \cup L, E)$, and the instance $\langle G; k_u, k_l \rangle$ asks for a minimum vertex cover for G with at most k_u U-vertices and at most k_l L-vertices. We apply the algorithm described above to construct the sets D, A, and C in GE-Theorem. The algorithm has its time complexity dominated by that of constructing a maximum matching in G, which is bounded by $O(m\sqrt{n})$. Now let $G' = G(C)$. Since the set A is contained in *every* minimum vertex cover for G and no vertex in the set D is in *any* minimum vertex cover, if the set A contains h_u U-vertices and h_l L-vertices, then the graph G has a minimum vertex cover with at most k_u U-vertices and at most k_l L-vertices if and only if the graph G' has a minimum vertex cover with at most $k'_u = k_u - h_u$ U-vertices and at most $k'_l = k_l - h_l$ L-vertices. By GE-Theorem, the graph G' has a perfect matching. $\square$

The technique of *reduction to problem kernel* has been widely used in the study of parameterized computation and in the research in fault coverage of reconfigurable arrays. Theorem 2 provides a much stronger version for this technique than previously proposed versions in both areas.

Hasan and Liu [7] proposed the concept of *critical sets.* The critical set of a bipartite graph G is the intersection of all minimum vertex covers for G, which is, equivalently, the set A in GE-Theorem. However, Hasan and Liu were unable to characterize the structure of the remaining graph when the critical set is removed from the graph G. By lack of this observation, they proposed to interlace the process of branch-and-bound searching and the process of constructing the critical set [7]. On the other hand, Theorem 2 indicates explicitly that the remaining graph G' after removing the critical set A and the independent set D has a perfect matching. This fact immediately limits the size of the graph G'. Moreover, since every minimum vertex cover for the graph G' contains exactly one end from every edge in a perfect matching in G', a careful study can show that the construction of critical sets during the branch-and-bound process proposed in [7] is totally unnecessary. This observation can significantly speed up the searching process. Finally, having a perfect matching in the graph G' will enable us to derive further powerful structure techniques in the searching process, which will be illustrated in detail in the next section.

The reduction to problem kernel technique has been very powerful in designing efficient parameterized algorithms [1,3,6,12]. Fernau and Niedermeier have considered a generalized version of the MIN-CVCB problem (in which the minimization of the vertex cover is not required), and described how to reduce the graph size to $2k_u k_l$. Theorem 2 enables us to improve this bound to $2(k_u + k_l)$: The facts that the graph G' has a perfect matching and that a minimum vertex cover of G' has at most k'_u U-vertices and at most k'_l L-vertices necessarily bound the number of vertices in G' by $2(k'_u + k'_l) \leq 2(k_u + k_l)$. In the development of parameterized algorithms for minimum vertex cover for general graphs, Chen, Kanj, and Jia [3] proposed to apply the classical Nemhauser-Trotter Theorem [11] that can reduce the problem size to $2k$, where k is the size of a minimum vertex cover. Similar to GE-Theorem, Nemhauser-Trotter Theorem decomposes the vertices of a graph into three parts D', A', and C', where D' is an independent set, A' is a subset of *some* minimum vertex cover, and the induced subgraph $G(C')$ has its minimum vertex cover of size at least half of C'. When applied to bipartite graphs, GE-Theorem, which decomposes the graph into three parts D, A, and C, is much stronger than Nemhauser-Trotter Theorem in all aspects: the set A is contained in *every* minimum vertex cover for G while the set A' is only contained in *some* vertex cover for G. In particular, for the application of the MIN-CVCB problem, the set A' cannot be automatically included in the minimum vertex cover being searched. Moreover, since the subgraph $G(C)$ has a perfect matching, it follows directly that the minimum vertex cover for $G(C)$ contains at least half of the vertices in C. By lack of having a perfect matching, the subgraph $G(C')$ also cannot take the advantages we will use in the searching process, which will be described in the next section.

3 Efficient Search by DM-Decomposition

According to Theorem 2, we can concentrate on instances $\langle G; k_u, k_l \rangle$ of the MIN-CVCB problem, where G is a bipartite graph that has a perfect matching and has at most $2(k_u + k_l)$ vertices.

A connected bipartite graph $B = (U \cup L, E)$ is *elementary* if every edge of B is contained in a perfect matching in B. An elementary bipartite graph B has exactly two minimum vertex covers, namely U and L [8]. This property makes our searching process very efficient on the elementary bipartite graph B: we should exclusively include either U or L in the minimum vertex cover.

The classical Dulmage-Mendelsohn Decomposition Theorem (shortly DM-Theorem) [8] provides a nice structure for bipartite graphs with perfect matching so that the above suggested searching process can be applied effectively.

Proposition 2. (DM-Theorem). *A bipartite graph $G = (U \cup L, E)$ with perfect matching can be decomposed into elementary bipartite subgraphs $B_i = (U_i \cup$*

$L_i, E_i)$, $i = 1, \ldots, r$, such that every edge in G between two subgraphs B_i and B_j with $i < j$ must have its U-vertex in B_i and its L-vertex in B_j.

The subgraphs B_i will be called *blocks*. A block B_i is a *d-block* if $|U_i| = |L_i| = d$. Edges connecting two different blocks will be called "inter-block edges".

Lemma 1. *Let $G = (U \cup L, E)$ be a bipartite graph with perfect matching. Then the decomposition of G given in Proposition 2 can be constructed in time $O(|E|^2)$.*

Proof. An edge e in G is an inter-block edge if and only if e is not contained in any perfect matching in G [8]. Consider the algorithm given in Figure 1.

```
DM-Decomposition
Input: a bipartite graph $G = (U \cup L, E)$ with perfect matching
Output: the blocks $B_1$, ..., $B_r$ in DM-Theorem
1. construct a perfect matching $M$ in $G$;
2. for each edge $e = [u, v]$ in $G$ do
       if the graph $G^- = G - \{u, v\}$ has no perfect matching
       then mark $e$ as an "inter-block edge";
3. remove all inter-block edges;
4. sort the remaining connected components: $B_1$, ..., $B_r$, so that every
   inter-block edge connects a $U$-vertex in a lower indexed block to an
   $L$-vertex in a higher indexed block.
```

Fig. 1. Dulmage-Mendelsohn Decomposition

Note that an edge $e = [u, v]$ is in a perfect matching in the graph G if and only if the graph $G - \{u, v\}$ has a perfect matching. Therefore, Step 2 of the algorithm **DM-Decomposition** correctly determines the inter-block edges. By DM-Theorem, each connected component after Step 3 is a block for the graph G. Now applying the topological sorting in a straightforward way will order the blocks so that each inter-block edge in the graph G connects a U-vertex in a lower indexed block to an L-vertex in a higher indexed block.

Constructing the perfect matching M takes time $O(|E|\sqrt{|E|})$ [10]. Consider an edge $e = [u, v]$ in G. If e is in M then obviously e is not an inter-block edge. If e is not in M, then after removing the vertices u and v and their incident edges, the matching M becomes a matching M^- of $|M| - 2$ edges in the remaining graph G^-. Thus, the remaining graph G^- has a perfect matching if and only if there is an augmenting path in G^- with respect to the matching M^-. Since testing the existence of an augmenting path can be done in linear time [8], we conclude that we can decide whether an edge in G is an inter-block edge in time $O(|E|)$. Therefore, Step 2 of the algorithm **DM-Decomposition** takes time $O(|E|^2)$, which dominates the complexity of the algorithm. □

Now we are ready to describe the main body of our algorithm for the MIN-CVCB problem. Let $\langle G; k_u, k_l \rangle$ be an instance of the MIN-CVCB problem, where $G = (U \cup L, E)$ is a bipartite graph with perfect matching, and G has at most $2(k_u + k_l)$ vertices. We first apply DM-Theorem that decomposes the graph G into blocks $B_i = (U_i \cup L_i, E_i)$, such that every inter-block edge connects a U-vertex in a block of lower index to an L-vertex in a block of higher index. It is easy to see that a minimum vertex cover of the graph G must be the union of minimum vertex covers of the blocks. Since the only two minimum vertex covers of the block B_i are U_i and L_i, the constrained minimum vertex cover of the graph G with at most k_u U-vertices and at most k_l L-vertices must be the union of properly selected U-parts and L-parts from the blocks.

The execution of our algorithm is depicted by a search tree whose leaves correspond to the potential constrained minimum vertex covers K of the graph G with at most k_u U-vertices and at most k_l L-vertices. Each internal node of the search tree corresponds to a branch in the searching process. Let $F(k_u + k_l)$ be the number of leaves in the search tree for finding a minimum vertex cover of at most k_u U-vertices and at most k_l L-vertices in the bipartite graph G. We list the possible situations in which we branch in our search process.

Case 1. A block B_i is a d-block with $d \geq 3$.

Let $B_i = (U_i \cup L_i, E_i)$. Since the constrained minimum vertex cover K of the graph G either contains the entire U_i and is disjoint from L_i, or contains the entire L_i and is disjoint from U_i, we branch in this case by either including the entire U_i in K (and removing L_i from the graph) or including the entire L_i in K (and removing U_i from the graph). In each case, we add at least 3 vertices in the constrained minimum vertex cover K. Thus, this branch satisfies the recurrence relation:

$$F(k_u + k_l) \leq 2F(k_u + k_l - 3) \tag{1}$$

A block sequence $[B'_1, \ldots, B'_h]$ is a *block chain* if there is an edge from the U-part of B'_i to the L-part of B'_{i+1} for all $1 \leq i \leq h-1$. Observe that the *chain implication* can speed up the searching process significantly. For example, if we include the L-part of the block B'_1 in the minimum vertex cover K, then the U-part of B'_1 must be excluded. Since there is an edge in G from the U-part of B'_1 to the L-part of B'_2, we must also include the entire L-part of B'_2 in K, which, in consequence, will imply that the L-part of B'_3 must also be included, and so on. Thus, the L-part of B'_1 in the minimum vertex cover K implies that the L-parts of all blocks in the chain must be in K. Similarly, the U-part of B'_h in K implies that the U-parts of all blocks in the chain must be in K. This observation enables us to handle many cases very efficiently.

Case 2. A 2-block is an interior block in a block chain.

Let $[B', B, B'']$ be a block chain, where B is a 2-block. Then including the L-part of B in the minimum vertex cover K forces at least 3 vertices in K (the L-parts of B and B''), and excluding the L-part of B also forces at least 3 vertices in K (the U-parts of B and B'). Thus, in this case, the branch satisfies the recurrence relation (1).

Case 3. The U-parts of three different 2-blocks are connected to the L-part of the same 1-block, or the U-part of a 1-block is connected to the L-parts of three different 2-blocks.

Suppose that the U-parts of the 2-blocks B'_1, B'_2, and B'_3 are connected to the L-part of the 1-block B'_0. Then including the U-part of B'_0 in the minimum vertex cover K forces at least 7 vertices (the U-parts of all the four blocks) in K, and excluding the U-part of B'_0 forces at least 1 vertex (the L-part of B'_0) in K. Therefore, the branch satisfies the recurrence relation

$$F(k_u + k_l) \leq F(k_u + k_l - 7) + F(k_u + k_l - 1) \qquad (2)$$

Similar analysis shows that (2) is also applicable to the other subcase.

Case 4. All Cases 1-3 are excluded.

Because Case 1 is excluded, there are only 1-blocks and 2-blocks. Since Case 2 is excluded, either the U-part or the L-part of each 2-block is not incident on any inter-block edges. Therefore, we can re-order the blocks as:

$$B'_1, \ldots, B'_x, B'_{x+1}, \ldots, B'_y, B'_{y+1}, \ldots, B'_z \qquad (3)$$

where B'_1, …, B'_x are 2-blocks whose L-parts are not incident on inter-block edges, B'_{x+1}, …, B'_y are 1-blocks, and B'_{y+1}, …, B'_z are 2-blocks whose U-parts are not incident on inter-block edges, such that every inter-block edge goes from the U-part of a lower indexed block to the L-part of a higher indexed block. Note that for any index i, $1 \leq i \leq z$, the U-parts of B'_1, …, B'_i plus the L-parts of B'_{i+1}, …, B'_z make a minimum vertex cover for the bipartite graph G. Now to construct a constrained minimum vertex cover K of the bipartite graph G with at most k_u U-vertices and at most k_l L-vertices, we have the following situations.

If $k_u + k_l > k$, where k is the size of a minimum vertex cover of the graph G, then we can pick an index i such that the U-parts of the blocks B'_1, …, B'_i consist of either $k_u - 1$ or k_u vertices (such an index always exists since all blocks are 1-blocks and 2-blocks). Now the U-parts of the blocks B'_1, …, B'_i together with the L-parts of the blocks B'_{i+1}, …, B'_z make a minimum vertex cover with at most k_u U-vertices and at most $k - (k_u - 1) \leq k_l$ L-vertices.

Thus, we can assume that $k_u + k_l = k$.

If $2x \leq k_u \leq 2x + (y - x)$, then let $i = k_u - x$. The minimum vertex cover K consisting of the U-parts of B'_1, …, B'_i and the L-parts of B'_{i+1}, …, B'_z contain $(i - x) + 2x = k_u$ U-vertices and $k - k_u = k_l$ L-vertices. Thus, K is a desired minimum vertex cover satisfying the constraint.

If $k_u < 2x$ and k_u is even, then the U-parts of the first $k_u/2$ 2-blocks plus the L-parts of the rest of the blocks make a desired minimum vertex cover.

Now consider the case $k_u < 2x$ and k_u is odd. If the graph G has no 1-blocks, then obviously there is no minimum vertex cover with k_u U-vertices and k_l L-vertices. Thus we assume that the graph G has at least one 1-block.

It is easy to see that there is a minimum vertex cover of exactly 1 U-vertices if and only if there is a 1-block whose L-vertex has degree 1, and there is a minimum vertex cover of exactly 3 U-vertices if and only if there is a 1-block whose L-vertex is adjacent to the U-part of at most one 2-block. Thus, we can further assume $k_u \geq 5$.

Pick the 1-block B of the minimum index i in sequence (3). Since Case 3 is excluded, there are at most two 2-blocks B' and B'' with their U-parts adjacent to the L-vertex of B. Note that since Case 2 is excluded, no inter-block edges can be incident on the L-parts of the blocks B' and B''. Now re-order the sequence (3) by removing B, B', and B'' from the sequence then re-inserting B', B'', B in the front of the sequence. Since the L-parts of B' and B'' are not adjacent to inter-block edges and inter-block edges from the L-part of B only go to the U-parts of B' and B'', the new sequence still has the property that every inter-block edge links the U-part of a lower indexed block to the L-part of a higher indexed block. Now the U-parts of the blocks B', B'', and B plus the U-parts of the next $(k_u - 5)/2$ 2-blocks in the new sequence and the L-parts of the rest of the blocks give a minimum vertex cover for G with k_u U-vertices and k_l L-vertices. In case the L-vertex of B is adjacent to the U-part of fewer than two 2-blocks, the desired minimum vertex cover K can be constructed similarly.

Finally, the case $k_u > 2x + (y - x)$ can be reduced to the previous case by reversing the sequence (3), exchanging the U-part and L-part of the bipartite graph G, and exchanging the numbers k_u and k_l.

Thus, in Case 4 the Min-CVCB problem can be solved in linear time.

4 Putting all together

We summarize all the discussions in the algorithm given in Figure 2.

```
CVCB-Solver
Input: a bipartite graph $G = (U \cup L, E)$ and two integers $k_u$ and $k_l$
Output: a minimum vertex cover $K$ of $G$ with at most $k_u$ $U$-vertices and
            at most $k_l$ $L$-vertices, or report no such a vertex cover exists
1. for each $U$-vertex $u$ of degree $> k_l$ do
       include $u$ in $K$ and remove $u$ from $G$;    $k_u = k_u - 1$;
2. for each $L$-vertex $v$ of degree $> k_u$ do
       include $v$ in $K$ and remove $v$ from $G$;    $k_l = k_l - 1$;
3. apply GE-Theorem to reduce the instance so that $G$ is a bipartite
   graph with perfect matching and has at most $2(k_u + k_l)$ vertices;
4. apply DM-Decomposition to decompose $G$ into blocks $B_1, \ldots, B_r$;
5. while one of the Cases 1-3 in section 3 is applicable, branch accordingly;
6. solve the instance for Case 4 in section 3.
```

Fig. 2. The main algorithm

Steps 1-2, taking time $O(kn)$, make immediate decisions on high degree vertices. If a U-vertex u of degree larger than k_l is not in K, then all neighbors of u should be in K, which would exceed the bound k_l. Thus, such U-vertices should be automatically included. Similar justification applies to L-vertices of degree larger than k_u. After Steps 1-2, the degree of the vertices in the graph is bounded by $k' = \max\{k_u, k_l\}$. Since each vertex can cover at most k' edges, the number of edges in the resulting graph must be bounded by $k'(k_u + k_l) \leq (k_u + k_l)^2$, otherwise the graph cannot have a desired minimum vertex cover.

Theorem 2 allows us to apply GE-Theorem in Step 3 to further reduce the bipartite graph G so that G has a perfect matching. The running time of this step is bounded by $O((k_u + k_l)^3)$. The number of vertices in the graph G now is bounded by $2(k_u + k_l)$.

Step 4 applies DM-Theorem to decompose the graph G into elementary bipartite subgraphs. By Lemma 1, this takes time $O((k_u + k_l)^4)$.

As we have discussed in Section 3, if any of Cases 1-3 is applicable, we can branch in our searching process with the recurrence relations (1) and (2). Eventually, we reach Case 4 in Section 3, which can be solved in linear time.

It is easy to verify that $F(k_u + k_l) = 1.26^{k_u+k_l}$ satisfies the recurrence relations (1) and (2). Thus, the running time of Step 5 in our algorithm is bounded by $O(1.26^{k_u+k_l}(k_u + k_l)^2)$, where the factor $(k_u + k_l)^2$ is the size of the graph resulted from Step 4. Using the general technique given in [13], we can further reduce the time complexity of Step 5 to $O(1.26^{k_u+k_l})$. Combining all these, we conclude with the following theorem.

Theorem 3. *The* Min-CVCB *problem is solvable in time* $O(1.26^{k_u+k_l} + kn)$.

5 Concluding remarks

Parameterized computation theory has proved very useful in designing practical algorithms in industrial applications, in particular for many NP-hard optimization problems. In this paper, we have developed a simple and significantly improved algorithm for the Min-CVCB problem, which arises from the extensively studied fault coverage problem for reconfigurable arrays in the area of VLSI manufacturing. Our algorithm is conceptually simple, easy to implement, and nicely combines classical graph structural results with the recently developed techniques in parameterized computation. For typical applications, our algorithm is faster than the previous known algorithms by several orders of magnitude.

Our improvement is on the reduction of the base in the exponential running time, which is of great theoretical and practical importance in the study of NP-hard optimization problems. For example, reducing the base by 0.01 in the parameterized vertex cover algorithm has resulted in upto 60% improvement in the running time for the application program used in biochemistry [5]. Our algorithm reduces the base by more than 0.14 (from 1.4^k to 1.26^k) from the previous known algorithms for the Min-CVCB problem.

We point out that our investigation emphasizes on the practicality of the algorithms, which includes simplicity (both conceptual and algorithmic) and effectiveness. The running time of our algorithm could be further improved slightly if we examine in more depth the combinatorial structures in a case-by-case exhaustive manner, which is something that we tried to avoid since it would deprive our algorithm from its simplicity and practicality. We might also apply the dynamic programming techniques, as illustrated in [3], which will improve the algorithm running time to $O(1.247^{k_u+k_l} + kn)$. However, this improvement is at the cost of exponential space.

References

1. R. Balasubramanian, M, R. Fellows, and V. Raman, An improved fixed parameter algorithm for vertex cover, *Information Processing Letters 65*, (1998), pp. 163-168.
2. R. Beigel and D. Eppstein, 3-coloring in time $O(1.3446^n)$: a no-MIS algorithm, *Proc. 36th IEEE Symp. on Foundations of Computer Science*, (1995), pp. 444-452.
3. J. Chen, I. A. Kanj, and W. Jia, Vertex cover: further observations and further improvement, *Lecture Notes in Computer Science 1665* (WG'99), (1999), pp. 313-324.
4. *DIMACS Workshop on Faster Exact Solutions for NP-Hard Problems*, Princeton, February 23-24, 2000.
5. R. G. Downey, M. R. Fellows, and U. Stege, Parameterized complexity: A framework for systematically confronting computational intractability, *AMS-DIMACS Proceedings Series 49*, F. Roberts, J. Kratochvil, and J. Nesetril, eds., (1999), pp. 49-99.
6. H. Fernau and R. Niedermeier, An efficient exact algorithm for constraint bipartite vertex cover, *Lecture Notes in Computer Science 1672* (MFCS'99), (1999), pp. 387-397.
7. N. Hasan and C. L. Liu, Minimum fault coverage in reconfigurable arrays, *Proc. 18th Int. Symp. on Fault-Tolerant Computing* (FTCS'88), (1988), pp. 348-353.
8. L. Lovász and M. D. Plummer, *Matching Theory*, Annals of Discrete Mathematics 29, North-Holland, 1986.
9. C. P. Low and H. W. Leong, A new class of efficient algorithms for reconfiguration of memory arrays, *IEEE Trans. Comput. 45*, (1996), pp. 614-618.
10. S. Micali and V. Vazirani, An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs, *Proc. 21st IEEE Symp. on the Foundation of Computer Science*, (1980), pp. 17-27.
11. G. L. Nemhauser and L. E. Trotter, Vertex packing: structural properties and algorithms, *Mathematical Programming 8*, (1975), pp. 232-248.
12. R. Niedermeier and P. Rossmanith, Upper bounds for vertex cover further improved, *Lecture Notes in Computer Science 1563* (STACS'99), (1999), pp. 561-570.
13. R. Niedermeier and P. Rossmanith, A general method to speed up fixed-parameter-tractable algorithms, *Information Processing Letters 73*, (2000), pp. 125-129.
14. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Co., 1980.
15. R. Paturi, P. Pudlak, M. E. Saks, and F. Zane, An improved exponential-time algorithm for k-SAT, *Proc. 39th IEEE Symp. on Foundations of Computer Science*, (1998), pp. 628-637.
16. W. Shi and W. K. Fuchs, Probabilistic analysis and algorithms for reconfiguration of memory arrays, *IEEE Trans. Computer-Aided Design 11*, (1992), pp. 1153-1160.
17. M. D. Smith and P. Mazumder, Generation of minimal vertex cover for row/column allocation in self-repairable arrays, *IEEE Trans. Comput. 45*, (1996), pp. 109-115.