An Approximation Algorithm Based on Chain Implication for Constrained Minimum Vertex Covers in Bipartite Graphs*

Jianxin Wang[1], Xiaoshuang Xu[1], and Jianer Chen[1,2]

[1] School of Information Science and Engineering, Central South University, Changsha 410083, China
jxwang@mail.csu.edu.cn
[2] Department of Computer Science Texas A&M University
College Station, TX 77843, USA
chen@cs.tamu.edu

Abstract. The constrained minimum vertex cover problem on bipartite graphs (the Min-CVCB problem) is an NP-complete problem. This paper presents a polynomial time approximation algorithm for the problem based on the technique of chain implication. For any given constant $\varepsilon > 0$, if an instance of the Min-CVCB problem has a minimum vertex cover of size (k_u, k_l), our algorithm constructs a vertex cover of size (k_u^*, k_l^*), satisfying $\max\{k_u^*/k_u, k_l^*/k_l\} \leq 1 + \varepsilon$.

1 Introduction

With the development of VLSI technology, the scale of electric circuit chip becomes larger and larger, and the possibility of introducing defects also increases along with the manufacture craft. With the increasing in the chip integration, it is not allowed that the wrong memory element appears in the manufacture process. A better solution is to use reconfigurable arrays. A typical reconfigurable memory 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. Since the cost of reconfiguration is proportional to the number of replaced rows and columns, people often replace as few as possible rows and columns to repair the array. This problem can be formulated as a constrained minimum vertex cover problem on bipartite graphs [3,7], which is formulated as follows.

Definition 1. [Constrained minimum vertex cover in bipartite graphs (Min-CVCB)] Given a bipartite graph $G = (V, E)$ with the vertex bipartition $V = U \cup L$ and two integers k_u and k_l, determine whether there is a minimum vertex cover of G with at most k_u vertices in U and at most k_l vertices in L.

* This work is supported by the National Natural Science Foundation of China (60433020) and the Program for New Century Excellent Talents in University (NCET-05-0683).

J.-Y. Cai, S.B. Cooper, and H. Zhu (Eds.): TAMC 2007, LNCS 4484, pp. 760–769, 2007.

For simplify our description, we will say that a bipartite graph $G = (U \cup L, E)$ has a minimum vertex cover of size (k_u, k_l) if G has a minimum vertex cover with at most k_u vertices in U and at most k_l vertices in L.

The Min-CVCB problem is NP-complete [3]. The problem was proposed by Hasan and Liu [7] in 1988, who introduced the concept of the critical set to develop a branch-and-bound algorithm for solving the Min-CVCB problem, based on the A^* algorithm [12]. No explicit analysis was giving in [7] for the running time of their algorithm, but it is not hard to see that in the worst-case the running time of the algorithm is at least $O(2^{k_u+k_l} + mn^{1/2})$. The Min-CVCB problem has been extensively studied in last two decades, mainly on heuristic algorithms. Interested readers are referred to [1,2,8,9,11,13] for the progress.

Recently, people used the parameterized computation theory to study exact algorithms for the Min-CVCB problem [3,6]. After analyzing the structures of bipartite graphs, Fernau and Niedermeier [6] used the branching search technology to develop an algorithm of time complexity $O((k_u + k_l)n + 1.4^{k_u+k_l})$. Chen and Kanj [3] proved that the Min-CVCB problem is NP-complete. By using classical results in matching theory and recently developed techniques in parameterized computation theory, they proposed an exact algorithm with time complexity $O((k_u + k_l)|G| + 1.26^{k_u+k_l})$, which is currently the best exact algorithm for the Min-CVCB problem.

Both heuristic algorithms and exact algorithms for the Min-CVCB problem have drawbacks. Heuristic algorithms are unable to provide solutions in a guaranteed time; while exact algorithms could get optimal solutions but run in exponential time. In this paper, we are interested in polynomial time approximation algorithms for the Min-CVCB problem. While a polynomial time approximation algorithm may not be able to find an optimal solution, it finds a near-optimal solution with a guaranteed error bound. In practice, near-optimality is often good enough, and acceptable in many application fields.

Our algorithm proceeds as follows. We firstly reduce the input size of a given instance of the Min-CVCB problem by applying the technique of kernelization proposed in parameterized computation theory [5]. Then we make full use of the classic matching theory on bipartite graphs to give a deep analysis of the structures of bipartite graphs. Based on the analysis, we get a $1 + \varepsilon$ approximation algorithm based on the technique of chain implication for the Min-CVCB problem, in the following sense: for a given constant $\varepsilon > 0$, if the instance of the Min-CVCB problem has a constrained minimum vertex cover of size (k_u, k_l), then our algorithm produces a vertex cover of size (k_u^*, k_l^*) satisfying $\max\{k_u^*/k_u, k_l^*/k_l\} \leq 1 + \varepsilon$.

The paper is organized as follows. In section 2, we formally define what is an approximation algorithm for the Min-CVCB problem. Section 3 describes some related definitions and lemmas. The heart of this paper is section 4, which proposes a $1 + \varepsilon$ approximation algorithm for the Min-CVCB problem based on the formulation of a directed acyclic graph. Section 5 gives the description of our algorithm and an explicit analysis of its approximation ratio and time complexity. Conclusions and future research are given in section 6.

2 Approximation Algorithms for the Min-CVCB Problem

First, we give some basic concepts in approximation algorithm theory. More details could be found in [4].

Definition 2. Given an NP optimization problem Q, we say that an algorithm A for Q has an *approximation ratio* $f(n)$ if, for any given instance I of Q, the value $A(I)$ of the solution produced by the algorithm A is within a factor of $f(|I|)$ of the value $Opt(I)$ of an optimal solution:

$$\max\{A(I)/Opt(I), Opt(I)/A(I)\} \leq f(|I|)$$

Note that $f(n)$ can be a constant.

Definition 3. Let Q be an NP optimization problem. If there exists an approximation algorithm A for Q that takes (x, ε) as its input, where x is an instance of Q and $\varepsilon > 0$ is a constant, and outputs a solution y to x such that the approximation ratio of y over the optimal solution to x is bounded by $1 + \varepsilon$, and for any fixed constant $\varepsilon > 0$, the algorithm A runs in time polynomial in the size of its input instance, then we say that the approximation algorithm A is a *polynomial-time approximation scheme* (shortly a PTAS) for the problem Q.

Therefore, approximation algorithms and PTAS are for computational optimization problems, in which a solution of optimal value is searched. On the other hand, the Min-CVCB problem is a decision problem, in which we only need to answer "yes" or "no" by determining if a given bipartite graph $G = (U \cup L, E)$ has a minimum vertex cover of size (k_u, k_l). To study approximability of the Min-CVCB problem, we formally define, as follows, what is a PTAS for the Min-CVCB problem.

Definition 4. An algorithm A is a *polynomial time approximation scheme* (i.e., PTAS) for the Min-CVCB problem if for any instance $(G = (U \cup L); k_u, k_l)$ of the Min-CVCB problem, and for any given constant $\varepsilon > 0$, the algorithm A either claims that the bipartite graph G has no minimum vertex cover of size (k_u, k_l), or, in case G has a minimum vertex cover of size (k_u, k_l), constructs a vertex cover of size (k_u^*, k_l^*) for G, satisfying $\max\{k_u^*/k_u, k_l^*/k_l\} \leq 1 + \varepsilon$, and the algorithm A runs in polynomial time for any fixed $\varepsilon > 0$.

3 Related Definitions and Lemmas

Our algorithm that searches for a near-optimal solution for the Min-CVCB problem contains the following two steps:

(1) Reducing to the kernel. The main idea of reducing to the kernel is to reduce the algorithm's search space size. To reduce to the Min-CVCB problem's kernel is to reduce the size of the input bipartite graph. In Lemma 1, by using a polynomial time preprocessing algorithm, we reduce a given instance of the

Min-CVCB problem to an equivalent instance $(G'; k'_u, k'_l)$ in which the bipartite graph G' has a perfect matching and has at most $2(k'_u + k'_l)$ vertices. Based on this, Lemma 2 applies the classical matching theory to convert it into a directed acyclic graph consisting of elementary bipartite graphs.

(2) Making full use of the technique of chain implication to enumerate the minimum vertex covers of the elementary bipartite graphs whose size is larger than a constant to search for a vertex cover for the given graph. It makes sure that, if the input bipartite graph G' has a minimum vertex cover of size (k'_u, k'_l), then the algorithm will find a close enough vertex cover for G'.

We start with some related definitions and related known results.

Definition 5. A graph G is *bipartite* if its vertex set can be partitioned into two sets U (the "upper part") and L (the "lower part") such that every edge in G has one endpoint in U and the other endpoint 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, and a vertex is a *U-vertex* (resp. an *L-vertex*) if it is in the U-part (resp. in the L-part) of the graph.

Let $G = (U \cup L, E)$ be a bipartite graph with a perfect matching. The graph G is *elementary* [10] if every edge in G is contained in a perfect matching in G. It is known that an elementary bipartite graph has only two minimum vertex covers, namely U and L [10].

Lemma 1. ([3]) *The time complexity for solving an instance $(G; k_u, k_l)$ of the Min-CVCB problem, where G is a bipartite graph of n vertices and m edges, is bounded by $O(mn^{1/2} + t(k_u + k_l))$, where $t(k_u + k_l)$ is the time complexity for solving an instance (G', k'_u, k'_l) of the problem, with $k'_u \leq k_u$, $k'_l \leq k_l$, and G' has a perfect matching and contains at most $2(k'_u + k'_l)$ vertices.*

Lemma 2. (The Dulmage-Mendelsohn Decomposition theorem [10]) *A bipartite graph $G = (U \cup L, E)$ with perfect matching can be decomposed and indexed into elementary subgraphs $B_i = (U_i \cup L_i, E_i)$, $i = 1, 2, \ldots, r$, such that every edge in G from a subgraph B_i to a subgraph B_j with $i < j$ must have one endpoint in the U-part of B_i and the other endpoint in the L-part of B_j. Such a decomposition can be constructed in time $O(|E|^2)$.*

An elementary subgraph B_i will be called an (elementary) *block*. The *weight* of a block $B_i = (U_i \cup L_i, E_i)$ is defined to be $|U_i| = |L_i|$. Edges connecting vertices in two different blocks will be called *inter-block edges*.

Lemma 3. ([3]) *Let G be a bipartite graph with perfect matching, and let B_1, B_2, ..., B_r be the blocks of G given by the Dulmage-Mendelsohn Decomposition. Then every minimum vertex cover for G is a union of minimum vertex covers of the blocks B_1, B_2, ..., B_r.*

We say that an instance $(G; k_u, k_l)$ of the Min-CVCB problem is *normalized* if $G = (U \cup L, E)$ has a perfect matching, $|L| = |U| \leq (k_u + k_l)$, and a Dulmage-Mendelsohn Decomposition of G is given. By Lemmas 1 and 2, we only need to concentrate on normalized instances of the Min-CVCB problem. Formally, this approach is validated by the following results

Lemma 4. ([3]) *There is an algorithm that, on a given instance $(G; k_u, k_l)$ of the Min-CVCB problem, constructs a subset Z_0 of vertices in G and a normalized instance $(G'; k'_u, k'_l)$ of the problem, where G' is a subgraph of G, such that*

(1) Every vertex cover of G' plus Z_0 is a vertex cover of G; and

(2) A vertex set Y in G' is a constrained minimum vertex cover of size (k'_u, k'_l) for G' if and only if the set $Y \cup Z_0$ is a constrained minimum vertex cover of size (k_u, k_l) for G.

The running time of the algorithm is $O(|G|^2)$.

By Lemma 4, if we want to develop a polynomial time approximation scheme for the Min-CVCB problem, we can simply concentrate on normalized instances of the problem, as shown by the following corollary.

Corollary 1. *Let $(G; k_u, k_l)$ be an instance of the Min-CVCB problem. Then a normalized instance $(G'; k'_u, k'_l)$ of the problem can be constructed in time $O(|G|^2)$, such that from a vertex cover of size (k''_u, k''_l) for G' satisfying the condition $\max\{k''_u/k'_u, k''_l/k'_l\} \leq 1+\varepsilon$, we can construct in time $O(|G|)$ a vertex cover of size (k^*_u, k^*_l) for G satisfying $\max\{k^*_u/k_u, k^*_l/k_l\} \leq 1+\varepsilon$.*

Proof. For the given instance $(G; k_u, k_l)$ of the Min-CVCB problem, where $G = (U \cup L, E)$ is a bipartite graph, we apply the algorithm in Lemma 4 to construct, in time $O(|G|)^2)$, the set Z_0 and the normalized instance $(G'; k'_u, k'_l)$ satisfying the conditions in the lemma. Suppose that the set Z_0 contains u U-vertices and l L-vertices in the bipartite graph G. By condition (2) in Lemma 4, we must have $k'_u + u = k_u$ and $k'_l + l = k_l$.

Suppose that Y is a vertex cover of size (k''_u, k''_l) for the graph G' satisfying $\max\{k''_u/k'_u, k''_l/k'_l\} \leq 1+\varepsilon$. By condition (1) in Lemma 4, the set $Y \cup Z_0$ is a vertex cover of size (k^*_u, k^*_l) for the graph G, where $k^*_u = k''_u + u$ and $k^*_l = k''_l + l$. Moreover, we have

$$k^*_u/k_u = (k''_u + u)/(k'_u + u) \leq \max\{1, k''_u/k'_u\} \leq 1+\varepsilon, \text{ and}$$
$$k^*_l/k_l = (k''_l + l)/(k'_l + l) \leq \max\{1, k''_l/k'_l\} \leq 1+\varepsilon.$$

That is, $Y \cup Z_0$ is a vertex cover of size (k^*_u, k^*_l) for the graph G that can be constructed from the vertex cover Y of the graph G' in time $O(|G|)$ and satisfies the condition $\max\{k^*_u/k_u, k^*_l/k_l\} \leq 1+\varepsilon$. □

4 The AACI-D Algorithm

We concentrate on normalized instances $(G; k_u, k_l)$ of the Min-CVCB problem, by which we assume that the bipartite graph $G = (U \cup L, E)$ has a perfect matching, $|L| = |U| \leq (k_u + k_l)$, and a Dulmage-Mendelsohn Decomposition $\{B_1, \ldots, B_r\}$ of G is given, where each B_i is a block in G, and every edge between two blocks B_i and B_j with $i < j$ has one end in the U-part of B_i and the other end in the L-part of B_j.

We construct a directed acyclic graph (a DAG) D from the decomposition $\{B_1, \ldots, B_r\}$ of G, as follows. Each block B_i in G corresponds to a vertex w_i in D, and there is a directed arc from w_i to w_j in the DAG D if and only if there is an edge from the U-part of the block B_i to the L-part of the block B_j in the graph G.

Let $\varepsilon > 0$ be a fixed constant. The PTAS for the Min-CVCB problem on normalized instances is given in Figure 1. Without loss of generality, we assume that $k_u \geq k_l$ (otherwise, we simply exchange U and L).

Algorithm AACI-D$(G, k_u, k_l, \varepsilon)$
input: a normalized instance (G, k_u, k_l) of Min-CVCB, and a constant $\varepsilon > 0$
output: a vertex cover for G, or report no vertex cover of size (k_u, k_l) for G

1. construct the DAG D based on the decomposition $\{B_1, \ldots, B_r\}$ of G;
2. let $\mathcal{B}$ be the collection of the blocks B_i whose weight is at least εk_u;
3. **for** each union Y of minimum vertex covers of the blocks in $\mathcal{B}$ **do**
3.1 $Y = Y \cup Y'$, where Y' is the set of vertices that are forced by Y;
3.2 **if** Y is consistent, with u U-vertices and l L-vertices **then**
let $B'_1, \ldots, B'_t$ be the rest of the blocks, sorted as in the DAG D;
let h be the first index such that $u + |B'_1| + \cdots |B'_h| > k_u$;
add the U-parts of $B'_1, \ldots, B'_h$ and the L-parts of $B'_{h+1}, \ldots, B'_t$ to Y;
return Y;
4. return ("no vertex cover of size (k_u, k_l) for G").

Fig. 1. The AACI-D Algorithm for normalized instances of Min-CVCB

We first give some explanations to the algorithm. Let $B_i = (U_i \cup L_i, E_i)$ be a block in the graph G. By Lemma 3, every minimum vertex cover of G either includes U_i but excludes L_i, or includes L_i but excludes U_i. Thus, for two blocks $B_i = (U_i \cup L_i, E_i)$ and $B_j = (U_j \cup L_j, E_j)$ in the graph G with an edge from U_i to L_j (thus $i < j$), if a minimum vertex cover of G includes L_i then it must also include L_j (since it excludes U_i and there is an edge from U_i to L_j). In this case, we say that the set L_j is *forced* (to be in the minimum vertex cover) by the set L_i. Similar derivation shows that the set U_i is forced by the set U_j. More generally, let $B_i = (U_i \cup L_i, E_i)$ be a block in the graph G, and let $\mathcal{B}_i$ be the collection of blocks B_j such that there is a directed path from B_i to B_j in the DAG D. Then all L-parts of the blocks in $\mathcal{B}_i$ are forced by the set L_i. Similarly, let $\mathcal{B}'_i$ be the collection of blocks B_j such that there is a directed path from B_j to B_i in the DAG D, then all U-parts of the blocks in $\mathcal{B}'_i$ are forced by the set U_i. Even more generally, let Y be a set that is a union of U-parts and L-parts of some blocks in G that does not contain both U-part and L-part of any block, then we say that a set Y' is *forced* by Y if the set Y' consists of those vertices that are either L-vertices forced by L-vertices in Y or U-vertices that is forced by U-vertices in Y. This technique of forcing more vertices into a minimum vertex cover using the existing vertices in the minimum vertex cover is called the *chain implication* technique [3].

For the given instance $(G; k_u, k_l)$ of the Min-CVCB problem, we say that a set Y of vertices in the graph G is *consistent* if each block of the graph G either has no intersection with Y, or has exactly one of its U-part or L-part entirely contained in the set Y, and if the number of U-vertices in Y is bounded by k_u and the number of L-vertices in Y is bounded by k_l.

Now we are ready to discuss the algorithm. Let $\{B_1, \ldots, B_r\}$ be the Dulmage-Mendelsohn Decomposition of the graph G, sorted in the order that every edge between two blocks B_i and B_j with $i < j$ has one end in the U-part of B_i and the other end in the L-part of B_j. As described in the algorithm, let $\mathcal{B}$ be the collection of the blocks B_i whose weight is at least εk_u.

By Lemma 3, every minimum vertex cover of the graph G is a union of minimum vertex covers of its blocks. Therefore, if the graph G has a minimum vertex cover Y_0 of size (k_u, k_l), then in the enumeration of step 3, we will eventually get a set Y in which the minimum vertex cover for each block B_i in $\mathcal{B}$ is the same as that in the set Y_0. For such a set Y, after forcing further vertices in Y' to the set Y, we still get a set Y that is a subset of the set Y_0. In particular, this set Y should be consistent. We show that on this set Y, the algorithm will return a vertex cover of the graph G with the desired properties. Suppose that Y contains u U-vertices and l L-vertices, $u \leq k_u$ and $l \leq k_l$.

Let $B'_1, \ldots, B'_t$ be the blocks in which no vertices are in the set Y. Note that none of these blocks B'_i is in the collection $\mathcal{B}$ since every block in $\mathcal{B}$ has either its U-part or its L-part included in the set Y by step 3. In particular, the weight of each B'_i is smaller than εk_u. Since h is the first index such that $u + |B'_1| + \cdots + |B'_h| > k_u$, we must have

$$u + |B'_1| + \cdots + |B'_{h-1}| \leq k_u$$

Combining this with $|B'_h| \leq \varepsilon k_u$, we get

$$u + |B'_1| + \cdots + |B'_{h-1}| + |B'_h| \leq k_u + \varepsilon k_u < (1 + \varepsilon) k_u$$

Therefore, in the final returned set Y, the number k^*_u of U-vertices in Y is bounded by $(1 + \varepsilon)k_u$. Moreover, since every block has exactly one of its U-part and L-part in Y, the number k^*_l of L-vertices in the final returned set Y is actually smaller than k_l since $k_u + k_l \geq k^*_u + k^*_l$. In consequence, the numbers k^*_u and k^*_l satisfy the condition $\max\{k^*_u/k_u, k^*_l/k_l\} \leq 1 + \varepsilon$.

We still need to prove that in this case the returned set Y is a vertex cover of the graph G. Let $e = [v_i, v_j]$ be any edge in the graph G. If e is an edge in a block B_i, then since every block in G has either its U-part or its L-part in the set Y, one of the vertices v_i and v_j must be contained in the set Y. If e is an inter-block edge, let v_i be a U-vertex in a block B_i and v_j be an L-vertex in a block B_j, $i < j$. If B_i is a block of weight at least εk_u, and if the U-part of B_i is not in Y, then the L-part of B_i must be in Y by our construction, and the L-part of B_i also forces the L-part of U_j into Y. So the vertex v_j is in the set Y. Similarly, if B_j has weight at least εk_u, then one of the vertices v_i and v_j must be in the set Y. The only remaining case is that both blocks B_i and B_j have weight smaller than εk_u. In this case, if B_i is one of the blocks $B'_1, \ldots, B'_h$,

then the U-part of B_i, thus the vertex v_i, is in the set Y. On the other hand, if B_j is one of the blocks $B'_{h+1}, \ldots, B'_t$, then the L-part of B_j, thus the vertex v_j, is in the set Y. Note that because the sets $B'_1, \ldots, B'_t$ are topologically sorted as in the DAG D, these are the only two possible cases: if the block B_j is in the collection $\{B'_1, \ldots, B'_h\}$ then so is the block B_i, and if the block B_i is in the collection $\{B'_{h+1}, \ldots, B'_t\}$ then so is the block B_j . In summary, it is always the case that one of the ends of the edge e is in the set Y. Since e is an arbitrary edge in G, we conclude that the set Y is a vertex cover of the graph.

This proves the correctness of the algorithm. We have the following theorem.

Theorem 1. *The algorithm AACI-D runs in time $O(m^2 + 2^{2/\varepsilon}m)$, where m is the number of edges in the graph G. If the input instance $(G; k_u, k_l)$ has a minimum vertex cover of size (k_u, k_l), then the algorithm produces a vertex cover for the graph G of size (k^*_u, k^*_l) satisfying $\max\{k^*_u/k_u, k^*_l/k_l\} \leq 1 + \varepsilon$.*

Proof. The second part of the theorem has been proved by the previous discussion. What remains is to prove the time complexity of the algorithm.

By Lemma 2, step 1 of the algorithm takes time $O(m^2)$. It is also easy to verify that step 2 of the algorithm takes no more than time $O(m^2)$.

The key observation is that the number of blocks in the collection $\mathcal{B}$ is bounded by $2/\varepsilon$. In fact, by our assumption, $k_u \geq k_l$. Therefore, the total number of U-vertices in the graph G is bounded by $k_u + k_l \leq 2k_u$ (note that the size of a minimum vertex cover of the graph G is equal to the number of U-vertices, which is equal to the number of L-vertices in G). Since all blocks in G are disjoint, and each block in $\mathcal{B}$ has weight (i.e., the number of U-vertices) at least εk, the total number of blocks in $\mathcal{B}$ is not larger than

$$(k_u + k_l)/(\varepsilon k_u) \leq 2k_u/(\varepsilon k_u) = 2/\varepsilon$$

Since each enumeration in step 3 takes either the U-part or the L-part of each block in $\mathcal{B}$, the total number of possible sets Y enumerated in step 3 is bounded by $2^{2/\varepsilon}$. Each execution of the body of step 3 obviously takes time no more than $O(m)$. In conclusion, the total running time of step 3 of the algorithm is bounded by $O(2^{2/\varepsilon}m)$. This concludes the theorem. □

5 Putting All Together

We summarize all the discussions given in the previous sections and present the AACI algorithm in Figure 2.

Some explanation is needed.

Let u be a U-vertex of degree larger than k_l. If u is not included in the vertex cover, then all neighbors of u must be in the vertex cover. Since u has more than k_l neighbors, which are all L-vertices, this implies that the number of L-vertices in the vertex cover would exceed the given bound k_l, which is not allowed in the minimum vertex cover of size (k_u, k_l). This justifies step 2: a U-vertex of degree larger than k_l should be directly included in the vertex cover. Similarly, in step 3,

Algorithm AACI$(G, k_u, k_l, \varepsilon)$
input: an instance (G, k_u, k_l) of Min-CVCB, and a constant $\varepsilon > 0$
output: a vertex cover for G, or report no vertex cover of size (k_u, k_l) for G

1. $Y_0 = \emptyset$;
2. **for** each U-vertex u of degree larger than k_l **do**
   $Y_0 = Y_0 \cup \{u\}$; $k_u = k_u - 1$;
3. **for** each L-vertex l of degree larger than k_u **do**
   $Y_0 = Y_0 \cup \{l\}$; $k_l = k_l - 1$;
4. construct an equivalent normalized instance $(G'; k'_u, k'_l)$ by Corollary 1;
5. call algorithm AACI-D on instance $(G'; k'_u, k'_l, \varepsilon)$;
6. **if** step 5 returns a vertex cover Y' for the graph G'
   then construct a desired vertex cover Y for G from Y' and Y_0
   else return ("no vertex cover of size (k_u, k_l) for G").

Fig. 2. The AACI-D Algorithm for normalized instances of Min-CVCB

all L-vertices of degree larger than k_u are directly included in the vertex cover. The running time of steps 1-3 is obviously bounded by $O((n+m)^2)$, where n and m are the number of vertices and number of edges in the graph G.

By Corollary 1, an equivalent normalized instance $(G'; k'_u, k'_l)$ can be constructed in time $O((n+m)^2)$ in step 4.

By Theorem 1, step 5 takes time $O(m^2 + 2^{2/\varepsilon}m)$, which either claims that the graph G' has no constrained minimum vertex cover of size (k'_u, k'_l), or returns a vertex cover Y' of size (k''_u, k''_l) satisfying $\max\{k''_u/k'_u, k''_l/k'_l) \leq 1 + \varepsilon$. By Lemma 4, if the graph G' has no constrained minimum vertex cover of size (k'_u, k'_l), then the graph G has no constrained minimum vertex cover of size (k_u, k_l). On the other hand, if G' has a minimum vertex cover of size (k'_u, k'_l), then by Theorem 1, step 5 of the algorithm returns a vertex cover of size (k''_u, k''_l) satisfying $\max\{k''_u/k'_u, k''_l/k'_l) \leq 1 + \varepsilon$. Now from Corollary 1, step 6 of the algorithm AACI will return a vertex cover of size (k^*_u, k^*_l) for the input graph G satisfying the condition $\max\{k^*_u/k_u, k^*_l/k_l) \leq 1 + \varepsilon$.

This proves the correctness of the algorithm AACI.

The time complexity of the algorithm also follows from the previous discussion. In particular, step 4 of the algorithm takes time $O((n+m)^2)$ by Corollary 1, and step 5 takes time $O((n+m)^2 + 2^{2/\varepsilon}(n+m))$ by Theorem 1.

We summarize these discussions in the following theorem.

Theorem 2. *The algorithm AACI is a polynomial time approximation scheme for the Min-CVCB problem, and its running time is bounded by* $O((n+m)(2^{2/\varepsilon} + n + m))$.

6 Conclusions

In this paper, we have studied the Min-CVCB problem that has direct applications in the area of VLSI manufacturing. Since heuristic algorithms and exact

algorithms may not meet the requirement of industry applications, we studied approximation algorithms for the problem. We developed a polynomial time algorithm with a $1+\varepsilon$ approximate ratio for the problem in the following sense: given an instance $(G;, k_u, k_l)$ for the Min-CVCB, where G is a bipartite graph and looking for a minimum vertex cover of at most k_u U-vertices and at most k_l L-vertices, our algorithm either reports that no such a minimum vertex cover exists, or constructs a vertex cover of k_u^* U-vertices and k_l^* L-vertices in G satisfying the condition $\max\{k_u^*/k_u, k_l^*/k_l\} \leq 1+\varepsilon$. The running time of our algorithm is bounded by $O((n+m)(2^{2/\varepsilon}+n+m))$.

References

1. D. M. Blough, On the reconfigurable of memory arrays containing clustered faults. *Proc. 21th Int. Symp. on Fault-Tolerant Computing* (FTCS'91), (1991), pp. 444-451. .
2. D. M. Blough and A. Pelc, Complexity of fault diagnosis in comparison models. *IEEE Trans.Comput.,41(3)* (1992), pp. 318-323.
3. J. Chen and I. A. Kanj, Constrained minimum vertex cover in bipartite graphs: complexity and parameterized algorithms, *Journal of Computer and System Science 67*, (2003), pp. 833-847.
4. T. H. Cormen, C. E. Leiserson, and R. L. Rovest *Introduction to Algorithms*, McGraw-Hill Book Company, New York, 1992.
5. R. Downey and M. Fellows, *Parameterized Complexity*, Springer, New York, 1999.
6. H. Fernau, and R. Niedermeier, An efficient exact algorithm for constraint bipartite vertex cover, *J. Algorithms 38*, (2001), pp. 374-410.
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. N. Hasan and C. L. Liu, Fault covers in reconfigurable PLAs. *Proc. 20th Int. Symp. on Fault-Tloerant Computing* (FTCS'90), (1990), pp. 166-173.
9. S.-Y. Kuo and W. Fuchs, Efficient spare allocation for reconfigurable arrays. *IEEE Des. Test 4* (1987), pp. 24-31.
10. L. Lovasz and M. D. Plummer, *Matching Theory*, *Annals of Discrete Mathematics Vol.29*, North-Holland, Amsterdam, 1986.
11. C. P. Low and H. W. Leong, A new class of efficient algorithms for reconfiguration of memery arrays, *IEEE Trans. Comput.45(1)*, (1996), pp. 614-618.
12. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA, 1980.
13. 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.