PAPER Special Section on Foundations of Computer Science —New Trends in Theory of Computation and Algorithm—

Worst Case Analysis of Approximation Algorithm of Abrams et al. for the Set k-Cover Problem*

Satoshi FUJITA[†a)], ***Member***

SUMMARY In this paper, we consider the problem of partitioning a given collection of node sets into k collections such that the average size of collections is the largest, where the size of a collection is defined as the cardinarity of the union of the subsets contained in the collection. More concretely, we give an upper bound on the performance ratio of an approximation algorithm proposed by Abrams et al., which is known to have a performance ratio of at least $1 - \frac{1}{e} \simeq 0.6321$ where e is Napier's constant. The proposed upper bound is $1 - (2 - \sqrt[d+1]{2})^{d+1}/2$ for any $d \geq 1$ provided that $k = o(n)$ which approaches to 0.75 as d increases.

key words: *Set k-cover, approximation algorithm, upper bound on the performance ratio*

1. Introduction

Let S be a finite set of nodes and $\mathcal{T}$ be a collection of subsets of S. In this paper, we consider the problem of partitioning $\mathcal{T}$ into k collections such that the average size of k collections is the largest. This problem, which is known as the **set k-cover problem** in the literature, is formally described as follows. A subset c of $\mathcal{T}$ is said to **cover** nodes in set $\bigcup_{S' \in c} S'$. For any partition Π of $\mathcal{T}$, the **weight** of Π is defined as

$$N(\Pi) \stackrel{\text{def}}{=} \sum_{c \in \Pi} N(c),$$

where $N(c)$ denotes the number of nodes covered by c. The set k-cover problem is the problen of given $\mathcal{T}$ and a natural number k, calculating a k-partition Π of $\mathcal{T}$ with a maximum weight.

Example 1: Let $S = \{1, 2, 3, 4\}$ and $\mathcal{T}$ be a collection of the following four subsets of S: $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, $S_3 = \{3, 4\}$ and $S_4 = \{4, 1\}$. The weight of 2-partition $\Pi_1 = \{\{S_1, S_3\}, \{S_2, S_4\}\}$ is $2 \times 4 = 8$ and since there is no 2-partition with weight nine or more, Π_1 is an (opitimum) solution to this instance for $k = 2$.

The set k-cover problem was proposed by Slijepcevic and Potkonjak in 2001 [9] with a proof of the NP-completeness (in fact, the problem becomes max k-cut [5] by restricting the number of occurrences of each node to the subsets to exactly two). A motivation of the problem is to realize a power-saving control of Wireless Sensor Networks (WSNs). In WSNs, a large number of (sensor) nodes are scattered over a given area so that the whole area is continuously monitored by the nodes. In general, those nodes are placed *redundantly* so that each point in the area is covered by the sensing region of several nodes because it is quite likely that a part of nodes suddenly crashes due to physical forces such as storms and a fire. On the other hand, such a redundancy brings us the possibility of enhancing the *sustainability* of WSNs. More concretely, by dividing the given node set into several subsets so that each subset covers the whole area and by periodically inactivating nodes so that the activated subset is periodically shifted to the next subset (e.g., every 10 minutes), we could reduce the average power consumption of each node to increase the lifetime of the overall WSN.

A key issue to realize such a switching of active subsets without degrading the monitoring performance of the WSN is how to partition the given node set to satisfy the requirement on the coverage of the given area. In the literature, such a requirement is often modeled as the **domatic partition** of a particular network [3], [4], [7], [8], by associating each subset S_i in $\mathcal{T}$ to the region sensed by node i in S. However, such a rigid model is too strong to be used in practical situations (e.g., there is no feasible solution for $k \geq 3$ if the given network has a leaf). The notion of set k-cover was introduced to overcome such issues in previous approaches. In fact, since $N(c)$ indicates the region covered by nodes in c, a large weight of Π indicates that a large portion of the area is covered by each subset in Π, on average. Thus, given k-partition Π of $\mathcal{T}$ with a maximum weight, a periodical switching of active nodes according to Π can reduce the power consumption of the overall WSN to $1/k$ without (significantly) decreasing the average portion of sensed regions.

In this paper, we analyze the performance of an approximation algorithm of Abrams et al. [1] for solving the set k-partition problem. Although it is known that a lower bound on the performance ratio of the algorithm is $1 - \frac{1}{e}$, the derivation of a non-trivial upper bound remained open. In this paper, we show that there is an instance such that the performance ratio of the algorithm is no better than $1 - (2 - \sqrt[d+1]{2})^{d+1}/2$ for any $d \geq 1$ provided that $k = o(n)$, which approaches to 0.75 as d increases.

The remainder of this paper is organized as follows. Section 2 overviews related work. Section 3 reviews the

Manuscript received March 26, 2013.
Manuscript revised June 6, 2013.

[†]The author is with the Department of Information Engineering, Institute of Engineering, Hiroshima University, Higashihiroshima-shi, 739–8527 Japan.

*Earlier version of this paper was presented at "Distributed Algorithm for Set k-Cover Problem and Its Worst Case Analysis," by Satoshi Fujita, In Proc. the Third International Conference on Networking and Computing (ICNC '12), December 2012.

a) E-mail: fujita@se.hiroshima-u.ac.jp

DOI: 10.1587/transinf.E97.D.399

deterministic algorithm of Abrams et al. Section 4 gives a proof of an upper bound on the performance ratio of the algorithm. Finally, Sect. 5 concludes the paper with future work.

2. Related Work

Recall that $N(c) \leq n$ for any $c \subseteq \mathcal{T}$. A dual of the set k-cover problem is to regard the weight of the resultant partition as a constraint. More specifically, we can define the problem of finding the maximum k such that the weight of the resultant k-partition is exactly $k \times n$. Such a variant is apparently NP-complete since it includes the domatic partition problem ([GT3] in [5]) as a special case (as a good news, there is a $O(\log |S|)$-approximation scheme for the maximization problem, since it is a variant of the set cover problem [2]). Unfortunately however, such a constraint on the weight is too strong if we wish to apply it to the power management in WSNs. In fact, if the given instance contains a node which is covered by exactly one element in $\mathcal{T}$, $k = 1$ is the trivial solution.

The set k-cover problem becomes easy if k is sufficiently large. When $k \geq |S|$, we could easily have an optimum solution such that $N(\Pi) = \sum_{S' \in \mathcal{T}} |S'|$ by letting $c_i := \{S_i\}$ for each $1 \leq i \leq |S|$. The reader should note that the actual value of $N(\Pi)$ decreases from the trivial uppder bound $\sum_{S' \in \mathcal{T}} |S'|$ due to the intersection of subsets contained in the same subset c_i. Thus, even for $k < |S|$, if we could partition $\mathcal{T}$ such that node sets contained in c_i are mutually disjoint for each i (or if k is large enough to allow such a partition), we can attain the trivial upper bound. This indicates we could hope that the following approach works well: 1) pack node sets to c_i's so that sets packed into the same c_i are mutually disjoint, and 2) if there is no such c_i, it selects c_j such that the amount of intersection with the nodes sets contained in c_i is the smallest. Abrams et al. proved that such a greedy approach based on the amount of intersection could attain a performance ratio of 0.5, which is (unfortunately) lower than the best known bound of $1 - \frac{1}{e}$. In addition, any polynomial time algorithm could not beat a certain bound on the performance ratio, since the performance ratio can not be better than $\frac{15}{16}$ unless $\mathcal{P} = \mathcal{NP}$ [1].

Another possibility to give a better bound for the set k-cover problem is to restrict the class of instances such as interval graphs and unit disk graphs. For example, it is known that the Minimum Dominating Set (MDS) problem admits a PTAS if the given instances are disk graphs, e.g., in a recent paper of Gibson and Pirwani [6], the technique of local search is effectively adopted to derive a PTAS for MDS. Although it is not clear whether we could apply the same technique to the set k-cover problem, if the value of k is fixed so that it is sufficiently smaller than the domatic number of the given graph, we could obtain an approximated solution by repeating the local search.

3. Approximation Algorithm of Abrams et al.

3.1 Randomized Scheme

Consider the following randomized algorithm for solving the k-set cover problem: 1) for each $1 \leq j \leq k$, initialize c_j to $\emptyset$; 2) for each $S_i \in \mathcal{T}$, randomly select j out of $1, \ldots, k$ and add S_i to c_j; 3) output the resultant $\{c_1, c_2, \ldots, c_k\}$ as the solution. We will call this algorithm RAND. In this subsection, we review the performance analysis of RAND conducted by Abrams et al. [1]. They proved the following claim.

Theorem 1: [1] The weight of the solution obtained by RAND is at least $1 - \frac{1}{e}$ times of an optimum weight.

The proof is as follows. In the following explanation, to make the exposition clear, we call each element in partition Π of $\mathcal{T}$ a "color." Let N_j denote the number of elements in $\mathcal{T}$ containing node j. Note that it holds $\sum_{j \in S} N_j = \sum_{i \in S} |S_i|$. Given node $j \in S$ and color $c \in \Pi$, the probability that j is *not* covered by a subset assigned to c under RAND is $\left(1 - \frac{1}{k}\right)^{N_j}$ (i.e., it should repeat "not to assign c to a subset covering j" N_j times). Thus, the probability that node j is covered by color c is

$$1 - \left(1 - \frac{1}{k}\right)^{N_j}.$$

Since the same argument holds for all colors, by the linearity of expectation, the expected number of colors covering j is $k - k\left(1 - \frac{1}{k}\right)^{N_j}$. Thus, the expected weight attained by RAND is given as

$$A = \sum_{j \in S} \left\{ k - k\left(1 - \frac{1}{k}\right)^{N_j} \right\}.$$

On the other hand, since the weight attained by an optimum algorithm is at most $\sum_{j \in S} \min\{k, N_j\}$, we could complete the proof by showing that $A / \sum_{j \in S} \min\{k, N_j\} \geq 1 - \frac{1}{e}$. In fact, we can prove the claim by showing the following slightly stronger claim:

$$\frac{k - k\left(1 - \frac{1}{k}\right)^{N_j}}{\min\{k, N_j\}} \geq 1 - \frac{1}{e} \tag{1}$$

for any j. Let us consider the following two cases.

1) When $k \leq N_j$, since $\min\{k, N_j\} = k$, the left hand side of Eq. (1) can be bounded as $1 - \left(1 - \frac{1}{k}\right)^{N_j} > 1 - \left(1 - \frac{1}{k}\right)^{k} \geq 1 - \frac{1}{e}$, where the first inequality is due to $k \leq N_j$ and the second inequality comes from a well known mathematical formula $\left(1 - \frac{1}{k}\right)^{k} \leq \frac{1}{e}$.

2) When $k \geq N_j$, the left hand side of Eq. (1) becomes

$$\frac{k}{N_j} \left\{ 1 - \left(1 - \frac{1}{k}\right)^{N_j} \right\}.$$

This value monotonically decreases as N_j increases from 0

to k, since the derivative is negative. Thus the value becomes minimum when $k = N_j$ and by the same argument to Case 1, we can conclude that it is at least $1 - \frac{1}{e}$. Q.E.D.

3.2 Derandomization

Abrams et al. proposed another algorithm which solves the k-set cover problem in a deterministic manner. This algorithm, which will be called DET hereafter, is a derandomization of RAND which repeats an assignment of node sets to the colors in a greedy manner. The order of node sets to be examined is determined by a permutation π over set S, i.e., node set $S_{\pi(1)}$ is assigned a color first, node set $S_{\pi(2)}$ is assigned a color next, and so on (permutation π plays the role of an adversary in the proof of the upper bound given in the next section). A color to be assigned to a given node set is selected so that it maximizes the **cost** of newly covered nodes, where a tie is broken by the subscript of colors (i.e., it prefers c_j to c_{j+1} if they cause the same cost) and the cost of nodes is re-calculated after each assignment (a formal definition of the cost function will be given later at the end of the proof). More concretely, they proved the following claim.

Theorem 2: [1] There is a deterministic algorithm which generates a solution whose weight is at least $1 - \frac{1}{e}$ of optimum.

In the following, we review the proof given by Abrams et al. The goal of the proof is to show that DET is as good as RAND provided that it appropriately defines the cost function. Since the following argument holds for any permutation π, to make the exposition simple, we assume that π is the identity permutation and the assignment is conducted in the order of the subscript of the node sets, i.e., S_1 is assigned first, S_2 is assigned next, and so on. The logic used in the proof is to compare the expected weights derived by the following two scenarios: in the first one, $S_1, S_2, \ldots, S_j$ are deterministically assigned and then $S_{j+1}, \ldots, S_n$ are randomly assigned. In the second scenario, $S_1, \ldots, S_{j-1}$ are deterministically assigned and then $S_j, \ldots, S_n$ are randomly assigned. If we can show that the expectation for the former case is no worse than that for the latter case, by iteratively moving position j from 1 to n, we can conclude that the weight attained by the deterministic scheme, i.e., DET, is no worse than the expected weight of the randomized scheme, i.e., RAND.

Now we compare the expectations of two scenarios. A key observation is that the change of the color assigned to subset S_j merely affects the number of colors concerned with nodes contained in S_j, since the order of assignments is fixed and we are assuming that the assignment for subsets $S_1, S_2, \ldots, S_{j-1}$ has been determined (in the following, we will say that "color c covers v" if v is covered by a subset assigned color c). Suppose that node v in S_j is covered by x different colors after completing the assignment for S_{j-1}. Consider the set of subsets covering v and let y_v be the number of subsets which covers v but has not yet been assigned a color (thus the subscript of those subsets must be at least j). Note that at the beginning of the algorithm, $y_v = N_v$ for all $v \in S$.

By the same argument to the last subsection, the probability that v is covered by a new color c after completing the second scenario is represented as $1 - \left(1 - \frac{1}{k}\right)^{y_v}$. Since there are $k - x$ such colors and v is covered by x colors before starting the assignment for S_j, the expected weight of node v in the second scenario is estimated as

$$\begin{aligned} W_2 &= x + (k - x)\left\{1 - \left(1 - \frac{1}{k}\right)^{y_v}\right\} \\ &= k - (k - x)\left(1 - \frac{1}{k}\right)^{y_v}. \end{aligned}$$

The reader should note that the value of y_v is decremented by one by conducting an assignment for a subset covering v.

Now let us evaluate how much the expected weight increases by selecting the color so that the number of colors covering v increases by the assignment for S_j (Case 1), instead of selecting the color so that the number does not change by the assignment (Case 2). In Case 2, the expected number of colors covering v (after completing all assignments) is

$$k - (k - x)\left(1 - \frac{1}{k}\right)^{y_v - 1},$$

while in Case 1, the expected number is

$$\begin{aligned} &k - (k - x - 1)\left(1 - \frac{1}{k}\right)^{y_v - 1} \\ &= k - (k - x)\left(1 - \frac{1}{k}\right)^{y_v - 1} + \left(1 - \frac{1}{k}\right)^{y_v - 1}. \end{aligned}$$

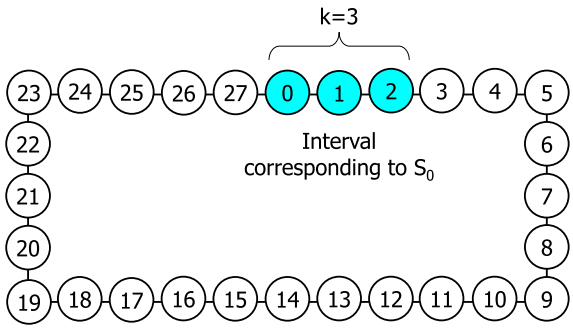
This indicates that by conducting an assignment of a color for S_j such that v is covered by a new color, the expected number of colors covering v increases by $\left(1 - \frac{1}{k}\right)^{y_v - 1}$, which could be regarded as a gain acquired by node v. This implies that by selecting color c for subset S_j such that the value of

$$\sum_{v \in S_j} \left(1 - \frac{1}{k}\right)^{y_v - 1} \tag{2}$$

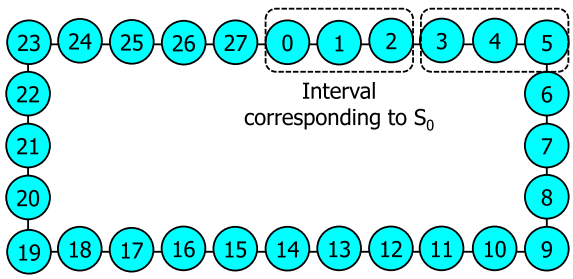
is the largest, we could maximize the gain acquired by subset S_j (recall that the assignment for S_j does not affect the weight of any other node not in S_j). Since the maximum gain is no smaller than the average gain attained by a randomized selection, we can conclude that the expected weight attained by the first scenario is no worse than the expected weight attained by the second scenario, provided that the cost function is defined as in Eq. (2). Hence the theorem follows. Q.E.D.

4. Upper Bound on the Performance Ratio

In this section, we derive an upper bound on the performance ratio of DET. The reader should note that in the paper



(a) Circular arrangement of nodes in S and an interval.



(b) Arrangement of intervals of the same color in an optimum solution.

Fig. 1 Instance for $d = 1$.

written by Abrams et al. [1], they merely give a lower bound on the performance ratio of DET and the analysis of the upper bound remained open.

Our main contribution is described in the following theorem.

Theorem 3: The performance ratio of DET is at most $1 - (2 - \sqrt[d+1]{2})^{d+1}/2$ for any $d \geq 1$.

In the following, we give a proof of the theorem. The proof is by construction, i.e., we will construct an instance such that the performance ratio for the instance is as worse as the indicated bound. In the next subsection, we explain the basic idea by restricting d to one (thus we will show that the upper bound is $1-(2-\sqrt{2})^2/2 = 2(\sqrt{2}-1) < 0.82843$). We then extend the argument to larger d's in the succeeding subsections.

4.1 Basic Idea

4.1.1 Instance

In this subsection, we assume that n is a multiple of k. Let $S = \{0, 1, \ldots, n-1\}$ be the set of nodes and $\mathcal{T} = \{S_0, S_1, \ldots, S_{n-1}\}$ be the set of node sets defined as follows:

$$S_i \stackrel{\text{def}}{=} \{i, i+1, \ldots, i+k-1\},$$

where "+" denotes the addition in modulo n. In the following, we suppose that nodes in S are arranged on a circle of size n in the clockwise order, and each node set in $\mathcal{T}$ is associated with an interval of length k. See Fig. 1 (a) for illustration. The reader should note that this instance is designed such that

- i is contained in exactly k intervals $S_i, S_{i-1}, \ldots, S_{i-k+1}$, i.e., $N_i = k$ for any i, and
- the weight of an optimum solution is $k \times n$.

In fact, an optimum solution of this instance is a partition of $\mathcal{T}$ into k subsets $c_1, c_2, \ldots, c_k$ such that each subset c_j consists of n/k intervals which are arranged on the circle so that there is no uncovered node between two consecutive intervals, i.e., there is no gap between two adjacent intervals. See Fig. 1 (b) for illustration.

4.1.2 Intended Behavior of DET

Recall that in algorithm DET, the order of node sets to be assigned a color is determined by a permutation π; i.e., node sets in $\mathcal{T}$ are assigned to colors in the following order:

$$S_{\pi(1)}, S_{\pi(2)}, \ldots, S_{\pi(n)}.$$

In the proof of the upper bound, permutation π is determined in such a way that DET should follow the following sequence of assignments:

- The sequence is partitioned into two parts.
- In the first part, it assigns a given subset to a color so that it is a new color for all nodes contained in the subset (recall that a tie is broken by the subscript of colors).
- In the second part, it assigns a given subset to a color so that it covers a gap of uncovered nodes which was left at the end of the first part.

In other words, we determine permutation π so that for each color c, there is a gap of length xk ($< k$) for any two adjacent intervals assigned color c, where the value of parameter x will be determined later so that it minimizes the resulting performance ratio (note that x can take discrete values since xk must be an integer). See Fig. 2 for illustration. Concrete permutation π will be given in the next subsection.

Recall that an optimum solution to this instance has weight $k \times n$, i.e., the set of n intervals is partitioned into k subsets so that each subset consisting of n/k intervals covers all nodes in S. In the first part, DET assigns at most

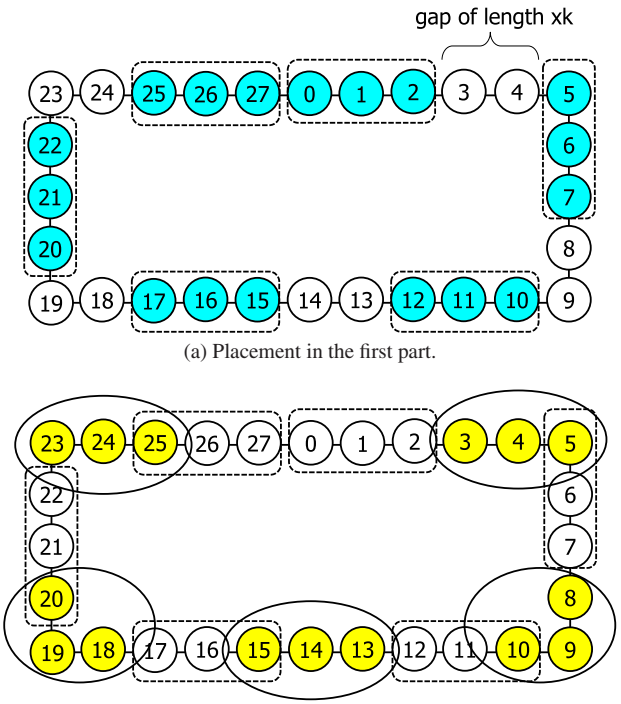
$$\eta \stackrel{\text{def}}{=} \left\lceil \left(\frac{n}{k}\right) \times \left(\frac{1}{1+x}\right) \right\rceil$$

intervals of length k to each color c, which implies that at most $k \times \eta$ nodes are covered by color c. In the second part, each interval can cover at most one remaining gap of length xk. Thus, on average, the number of gaps covered by color c is $\frac{n}{k} - \eta$, i.e., it newly covers $xk \times (\frac{n}{k} - \eta)$ nodes. Since n nodes are covered by color c in an optimum solution, by letting $f(x)$ be the performance ratio of DET, we have

$$n \times f(x) = k \times \eta + xk \times \left(\frac{n}{k} - \eta\right).$$

That is,

$$f(x) < \frac{k}{n}\left(\frac{n}{k(1+x)} + 1\right) + \frac{xk}{n}\left(\frac{n}{k} - \frac{n}{k(1+x)}\right)$$



(a) Placement in the first part.

(b) Placement in the second part.

Fig. 2 Arrangement of intervals controlled by the adversary.

$$= \frac{1}{1+x} + \frac{k}{n} + x\left(1 - \frac{1}{1+x}\right)$$
$$= \frac{1}{1+x} + x - \frac{x}{1+x} + \frac{k}{n}$$
$$= x + \frac{1-x}{1+x} + \frac{k}{n}.$$

Since

$$f'(x) = 1 - \frac{1}{1+x} - \frac{1-x}{(1+x)^2}$$
$$= 1 - \frac{1+x+1-x}{(1+x)^2}$$
$$= 1 - \frac{2}{(1+x)^2},$$

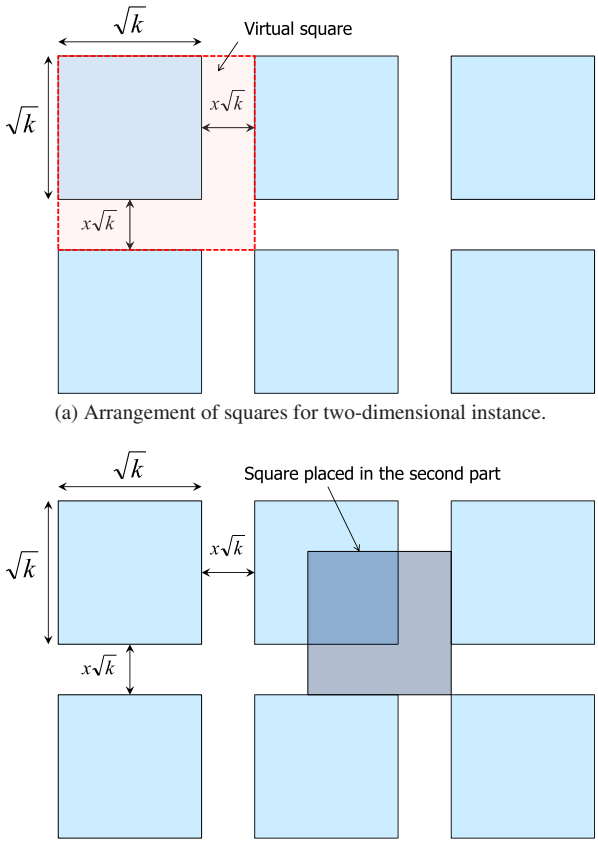
function $f(x)$ takes the minimum value when $x = \sqrt{2} - 1$. Although the actual value of x which could be taken by the algorithm is $\lceil(\sqrt{2}-1)k\rceil/k$ which is smaller than $\sqrt{2}-1$ for any k, it converges to $\sqrt{2} - 1$ as the value of k increases. In addition, the term of k/n converges to zero as n increases provided that $k = o(n)$. Thus, we can conclude that an upper bound on the performance ratio of DET for the above instance is at most

$$f(\sqrt{2}-1) = 2\sqrt{2} - 2 < 0.82843$$

provided that $k = o(n)$.

4.1.3 Permutation

A concrete permutation π satisfying the above requirements is given as follows:



(a) Arrangement of squares for two-dimensional instance.

(b) Partial coverage of a gap.

Fig. 3 Instance for $d = 2$.

- For $i := 0$ to $\eta - 1$, $\pi(i) := (1+x)k \times i$.
- For $i := \eta$ to $2\eta - 1$, $\pi(i) := (1+x)k \times i + 1$. Similarly, for $i := j \times \eta$ to $(j+1)\eta - 1$ for some $2 \le j \le k-1$, $\pi(i) := (1+x)k \times i + j$.
- The remaining part of the permutation, i.e., $\pi((k-1)\eta)$, $\pi((k-1)\eta + 1)$, ..., $\pi(n-1)$, is given in an arbitrary way.

4.2 Extension to Two-Dimensional Case

Before proceeding to the proof of theorem, we illustrate the way of extending the above argument to two-dimensional case (i.e., the case of $d = 2$) to help the reader to intuitively grasp our idea. Recall that in the last subsection, we consider a circle of size n and associate each subset in $\mathcal{T}$ with an interval of length k on the circle. Our extension to the two-dimensional case is very simple. We consider a ball with surface area n, and associate each subset in $\mathcal{T}$ with a square of size $\sqrt{k} \times \sqrt{k}$ on the surface of the ball. The set of squares is determined in such a way that: 1) each node is covered by exactly k squares and 2) the weight of an optimum solution is $k \times n$.

Then, we consider an adversary (i.e., permutation) such that DET should assign colors to squares so that the distance to an adjacent square with the same color is $x\sqrt{k}$. See Fig. 3 (a) for illustration (in the following, we assume that

Table 1 Upper bound on the performance ratio calculated for each d.

d	1	2	3	4	5	6	7	8	9	10	20	50
upper bound	0.8285	0.7973	0.7839	0.7764	0.7717	0.7684	0.7660	0.7641	0.7626	0.7614	0.7559	0.7524

$x\sqrt{k}$ is an integer since it does not violate the correctness of the analysis provided that $k = o(n)$ and n is sufficiently large). Since the gap virtually increases the size of a square from $\sqrt{k}\times\sqrt{k}$ to $(1+x)\sqrt{k}\times(1+x)\sqrt{k} = (1+x)^2k$, the number of squares of color c placed in the first part of the assignment sequence is $\frac{n}{(1+x)^2k}$. Since any square of size $\sqrt{k}\times\sqrt{k}$ can cover at most

$$k-(1-x)^2k \;=\; k\left\{1-(1-x)^2\right\}$$

nodes in the gap (see Fig. 3 (b)), we can use the same argument to the last subsection to derive an upper bound for the two-dimensional instance (the proof of the claim will be given in the next subsection as a claim for general d's).

4.3 Proof of Theorem

Let d be a positive integer. In the following, we assume that n is a multiple of k and k is an integer represented as $k = h^d$ for some integer $h \geq 2$. Each subset in $\mathcal{T}$ is associated with a d-dimensional hypercube of size $\sqrt[d]{k}\times\cdots\times\sqrt[d]{k}$ each, and those hypercubes fill a d-dimensional space so that: 1) each node is covered by exactly k hypercubes and 2) the weight of an optimum solution is $k\times n$.

We consider an adversary such that in the first part of the assignment sequence, DET assigns hypercubes to colors such that the distance between two adjacent hypercubes with the same color is $x\sqrt[d]{k}$. Note that the number of hypercubes of color c assigned in the first part is

$$\eta = \frac{n}{(1+x)^dk}$$

(again, we assume that η is an integer to clarify the argument). In addition, in the second part of the assignment sequence, any hypercube of size $\sqrt[d]{k}\times\cdots\times\sqrt[d]{k}$ can newly cover

$$k-(1-x)^dk \;=\; k\left\{1-(1-x)^d\right\}$$

nodes in the gap (note that it corresponds to a placement so that the placed hypercube of size k "inscribes" a virtual hypercube of size $(1+x)^dk$).

Thus, since the number of gaps covered by color c is $\frac{n}{k}-\eta$ on average and n nodes are covered by color c in an optimum solution, by letting $f(x)$ be the peformance ratio of DET, we have

$$n\times f(x) \;=\; k\times\eta + k\left\{1-(1-x)^d\right\}\times\left(\frac{n}{k}-\eta\right),$$

that is,

$$\begin{aligned} &f(x)\\ &= \frac{1}{(1+x)^d} + \left\{1-(1-x)^d\right\}\times\left\{1-\frac{1}{(1+x)^d}\right\}\\ &= 1-(1-x)^d + \frac{(1-x)^d}{(1+x)^d}. \end{aligned}$$

Since

$$\begin{aligned} &f'(x)\\ &= d(1-x)^{d-1} - \frac{d(1-x)^{d-1}}{(1+x)^d} - \frac{d(1-x)^d}{(1+x)^{d+1}}\\ &= d(1-x)^{d-1} - \frac{d(1-x)^{d-1}}{(1+x)^{d+1}}\times(1+x+1-x)\\ &= d(1-x)^{d-1} - \frac{2d(1-x)^{d-1}}{(1+x)^{d+1}}, \end{aligned}$$

$f(x)$ takes the minimum value when $1 = \frac{2}{(1+x)^{d+1}}$, i.e., when $x = \sqrt[d+1]{2}-1$. Thus the performance ratio is at most

$$\begin{aligned} f(\sqrt[d+1]{2}-1) \;&=\; 1-(2-\sqrt[d+1]{2})^d + \frac{(2-\sqrt[d+1]{2})^d}{(\sqrt[d+1]{2})^d}\\ &=\; 1-(2-\sqrt[d+1]{2})^d\left(1-\frac{1}{(\sqrt[d+1]{2})^d}\right)\\ &=\; 1-(2-\sqrt[d+1]{2})^d\left(1-\frac{\sqrt[d+1]{2}}{2}\right). \end{aligned}$$

Hence the theorem follows. Q.E.D.

The result of numerical calculations is summarized in Table 1. As shown in the table, the upper bound given in the theorem gradually approaches to 0.75 as the value of parameter d increases.

5. Concluding Remarks

In this paper, we give a non-trivial upper bound on the performance ratio of an approximating algorithm proposed by Abrams et al. for solving the set k-cover problem. More concretely, we show that there is an instance such that the performance ratio is at most $1-(2-\sqrt[d+1]{2})^{d+1}/2$ for any $d \geq 1$ provided that $k = o(1)$.

A future work is to improve the algorithm of Abrams et al. so that the performance ratio is at least 0.75. To this end, probably we have to carefully consider the order of assignments so that the "loss" of assignment due to the intersection with a subset with the same color could be minimized.

References

[1] Z. Abrams, A. Goel, and S. Plotkin, "Set k-cover algorithms for energy efficient monitoring in wireless sensor networks," Proc. 3rd Inte'l Symp. on Information Processing in Sensor Networks (IPSN '04), pp.424–432, 2004.

[2] U. Feige, M.M. Halldorsson, and G. Kortsarz, "Approximating the domatic number," Proc. 32nd Annual ACM Symp. on Theory of Computing, pp.134–143, 2000.

[3] S. Fujita, M. Yamashita, and T. Kameda, "A study on r-configurations — A resource assignment problem on graphs," SIAM J. Discrete Math., vol.13, no.2, pp.227–254, 2000.

[4] S. Fujita, "A tight bound on the number of mobile servers to guarantee transferability among dominating configurations," Discrete Applied Mathematics, vol.158, no.8, pp.913–920, 2010.
[5] M.R. Garey and D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, 1979.
[6] M. Gibson and I.A. Pirwani, "Approximation algorithms for dominating set in disk graphs," CoRR abs/1004.3320, 2010.
[7] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater, Fundamentals of Domination in Graphs, Marcel Dekker, 1998.
[8] M. Mito and S. Fujita, "Maximum connected domatic partition of directed path graphs with single junction," Computing and Combinatorics, 14th Annual Int'l Conf., COCOON 2008, LNCS, vol.5092, pp.425–433, June 2008.
[9] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," Proc. IEEE Int'l Conf. on Communications (ICC), pp.472–476, June 2001.
[10] D. Tian and N.D. Georganas, "A node scheduling scheme for energy conservation in large wireless sensor networks," Wireless Communications and Mobile Computing, pp.271–290, May 2003.
[11] T. Yan, T. He, and J.A. Stankovic, "Differentiated surveillance for sensor networks," Proc. 1st Int'l Conf. on Embedded Networked Sensor Systems (ACM SenSys), pp.51–62, 2003.
[12] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Energy efficient robust sensing coverage in large sensor networks," UCLA Technical Report, 2002.

Satoshi Fujita received the B.E. degree in electrical engineering, M.E. degree in systems engineering, and Dr.E. degree in information engineering from Hiroshima University in 1985, 1987, and 1990, respectively. He is a Professor at Graduate School of Engineering, Hiroshima University. His research interests include communication algorithms, parallel algorithms, graph algorithms, and parallel computer systems. He is a member of the Information Processing Society of Japan, SIAM Japan, IEEE Computer Society, and SIAM.