19th International Computer Science and Engineering Conference (ICSEC)
Chiang Mai, Thailand, 23-26 November, 2015

On the Parallel Complexity of Minimum Sum of Diameters Clustering

Nopadon Juneam*† and Sanpawat Kantabutra†
*Department of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai, THAILAND, 50200
†The Theory of Computation Group, Department of Computer Engineering
Faculty of Engineering, Chiang Mai University, Chiang Mai, THAILAND, 50200
juneam.n@gmail.com, sanpawat.k@gmail.com

***Abstract*—Given a set of n entities to be classified, and a matric of dissimilarities between pairs of them. This paper considers the problem called MINIMUM SUM OF DIAMETERS CLUSTERING PROBLEM, where a partition of the set of entities into k clusters such that the sum of the diameters of these clusters is minimized. Brucker showed that the complexity of the problem is NP-hard, when $k \geq 3$ [1]. For the case of $k = 2$, Hansen and Jaumard gave an $O(n^3 \log n)$ algorithm [2], which Ramnath later improved the running time to $O(n^3)$ [3]. This paper discusses the parallel complexity of the MINIMUM SUM OF DIAMETERS CLUSTERING PROBLEM. For the case of $k = 2$, we show that the problem in parallel in fact belongs in class NC.[1] In particular, we show that the parallel complexity of the problem is $O(\log n)$ parallel time and n^7 processors on the COMMON CRCW PRAM model. Additionally, we propose the parallel algorithmic technique which can be applied to improve the processor bound by a factor of n. As a result, we show that the problem can be quickly solved in $O(\log n)$ parallel time using n^6 processors on the COMMON CRCW PRAM model. In addition, regarding the issue of high processor complexity, we also propose a more practical NC algorithm which can be implemented in $O(\log^3 n)$ parallel time using $n^{3.376}$ processors on the EREW PRAM model.**

Keywords*—parallel complexity, PRAM algorithm, clustering, minimum sum of diameters, application in social networking

I. INTRODUCTION

Clustering is a well-known and well-studied problem. Applications in clustering are known in a wide variety of areas such as data mining, machine learning, image processing, bioinformatics, and social networking. Conceptually, a *cluster* is a collection of similar entities, where entities within the same cluster are more similar to each other than to those in different clusters. A basic problem of clustering is to partition a given finite set of n entities into k clusters according to the criteria of similarity defined for clusters in the partition. The type of clustering as one desires is application dependent. However, the clustering criteria are most often based on two fundamental notions of analysis—*homogeneous* and/or *well-separated* classes [4], [5]. The concepts of seperation and homogeneity can be made precise in several ways.

Commonly, a measure used in characterizing seperation is assumed by the *dissimilarity* between entities. Homogeneity of a cluster is often characterized by the *diameter* of the cluster [6], which is defined as the maximum dissimilarity between any pair of entities in that cluster. *Minimum diameter clustering* is the problem to determine a partition such that the maximum diameter of its clusters' diameters is minimized. A variation of the problem considering a partition in which the sum of its clusters' diameters is minimized is called *minimum sum of diameters clustering*.

Minimum diameter and minimum sum of diameters clusterings are of interest in many situations, where homogeneity is the main concern in the analysis. However, previous studies suggested that minimum diameter clustering suffers from the *dissection effect* [1], [7], [6], [8], as is known as very similar entities may be assigned to very different clusters. The cause of this effect is found as minimum diameter clustering is likely to require clusters to have fairly equal diameters, and a natural cluster could get split for this reason. In the case of minimum sum of diameters clustering, the dissection effect is usually much less damaging, since no such equalizing factor is at play when the sum of diameters is considered.

Complexity of minimum sum of diameters clustering has been studied. All concentrated on the sequential aspect. In [1], Brucker showed that the problem is NP-hard, where $k \geq 3$. For the case of $k = 2$, Hansen and Jaumard gave an $O(n^3 \log n)$ algorithm [2], which Ramnath later improved the running time to $O(n^3)$ [3]. Emphasize that though the case of two clusters is quite restrictive, it appears that in practice a bipartitioning algorithm can also be recursively applied to approximate a partitioning into three or more clusters [2]. This technique is also known as *divisive hierarchical clustering*, which perhaps is one of the most popular topics that has been studied extensively in clustering literature [9].

This paper considers minimum sum of diameters clustering from the parallel perspective. Our work presented here is motivated by a number of existing applications of clustering in social networking in which a system is expected to deal with massive data set provided by users in real-time. Grouping/matching users based on similarities of temporal information are relevant examples of such applications. For large systems of social networking, it is commonly found that many efficient sequential algorithms have failed in real situations. Hence, from a theoretical point of view, parallel algorithms are of interest for substantial improvements in time serving such systems.

This paper discusses the parallel complexity of minimum sum of diameters clustering, for the case of $k = 2$, where

[1]Class of problems for which a polylogarithmic time parallel algorithm with polynomial processor complexity is known.

the clustering is defined as the optimization problem called MINIMUM SUM OF DIAMETERS CLUSTERING PROBLEM. Our results reveals that the problem in parallel in fact belongs in class NC. More details on parallel complexity class NC can be found in [10]. In particular, we show that the parallel complexity of the problem is $O(\log n)$ parallel time and n^7 processors on the COMMON CRCW PRAM model. Additionally, we propose the parallel algorithmic technique which can be applied to improve the processor bound by a factor of n. As a result, we show that the problem can be quickly solved in $O(\log n)$ parallel time using n^6 processors on the COMMON CRCW PRAM model. In addition, regarding the issue of high processor complexity, we also propose a more practical NC algorithm which can be implemented in $O(\log^3 n)$ parallel time using $n^{3.376}$ processors on the EREW PRAM model.

The rest of this paper is organized as follows: section II introduces notations and preliminaries. Section III discusses the parallel complexity of MINIMUM SUM OF DIAMETERS CLUSTERING PROBLEM. Section IV describes the parallel algorithmic technique for the processor improvement. A more practical NC algorithm is given in section V. Finally, we give the conclusion and dicuss our work in section VI.

II. NOTATIONS AND PRELIMINARIES

This section introduces some notations and preliminaries related to the minimum sum of diameters clustering. The parallel model of computation used in this paper is described.

A. Notations and Definitions

Throughout this paper, let $S = \{1, 2, \ldots, n\}$ denote the finit set of n *entities*. We assume *dissimilarities* between pairs of entities in the set S be given by a *distance* metric $d : S^2 \to \{0\} \cup \mathbb{R}^+$. A dissimilarity between entities i and j in S, denoted by d_{ij}, satisfies the conditions $d_{ii} = 0$, and $d_{ij} = d_{ji}$, for all $1 \leq i, j \leq n$. A family $P = \{C_1, C_2, \ldots, C_k\}$ of k subsets of S is called a *partition* of S into k *clusters* if for all $1 \leq i, j \leq k$, $C_i \neq \emptyset$, $C_i \cap C_j = \emptyset$, and $C_1 \cup C_2 \cup \cdots \cup C_k = S$. For a cluster $C \subseteq S$, the largest dissimilarity between entities in C defines the *diameter* of the cluster C, denoted by $d(C)$, where $d(C) = max_{i,j \in C}\{d_{ij}\}$. The optimization problem called MINIMUM SUM OF DIAMETERS CLUSTERING PROBLEM (MSDCP) is defined as follows.

<u>MINIMUM SUM OF DIAMETERS CLUSTERING PROBLEM (MSCDP)</u>
Given: A tuple (S, d) and integer $k \leq n$.
Problem: Determine a partition $P = \{C_1, C_2, \ldots, C_k\}$ of S such that $d(C_1) + d(C_2) + \cdots + d(C_k)$ is minimized.

This paper discusses the parallel complexity of the MSDCP. Throughout this paper, we assume $k = 2$, and without loss of generality, $d(C_1) \geq d(C_2)$.

B. The Parallel Model of Computation

The model of parallel computation in our discussion in this paper is called the *Parallel Random Access Machine* (PRAM). The basic PRAM model is a machine consisting of a collection of infinite number of RAM processors and a collection of infinite number of shared memory cells in which multiple accesses to these cells by two or more processors are allowed. Inputs and outputs to the computation are placed in shared memory to allow the concurrent access. There are quite a few variations of this parallel machine. However, most of our complexity results are based on the *Common Concurrent Read Concurrent Write* PRAM (COMMON CRCW PRAM) model, which permits the simultaneous reads, and the simultaneous writes succeed only if all processors try to write the same value. And our more practical parallal algorithm are based on the *Exclusive Read Exclusive Write* PRAM (EREW PRAM) model, which only one processor can read and write to a memory cell at any time. More information on the PRAM model and its variations can be found in [10], [11].

III. MSDCP'S PARALLEL COMPLEXITY

This section discusses the parallel complexity of the MSDCP, showing that the problem can be quickly solved in $O(\log n)$ parallel time using n^7 processors on the COMMON CRCW PRAM model. Essentially, the parallel complexity of the MSDCP in our discussion is based on that of the decision problem called POSSIBLE PAIR OF DIAMETERS PROBLEM (PPDP). Formally, the problem is defined as follows.

<u>POSSIBLE PAIR OF DIAMETERS PROBLEM (PPDP)</u>
Given: A tuple (S, d) and a pair of dissimilarities (d_1, d_2), with $d_1 \geq d_2$.
Problem: Is there a bipartition $P = \{C_1, C_2\}$ of S such that $d(C_1) \leq d_1$ and $d(C_2) \leq d_2$?

Intuitively, the PPDP as MSDCP's subroutine verifies whether the given pair (d_1, d_2) is possible to be the clusters' diameters. For a given MSCDP instance, the optimal pair of diameters $(d(C_1), d(C_2))$ can be found among all of PPDP YES-instances in which $d_1 + d_2$ minimized.

We first provide the generic parallel complexity of the MSCDP by the parallel complexity of PPDP, and then show the parallel complexity of PPDP. Subsequently, the parallel complexity of the MSDCP is shown.

A. MSDCP*'s Generic Parallel Complexity*

Let $t(n)$ and $p(n)$ denote the parallel time and processor complexities of the PPDP, and $m(n)$ the number of necessary instances of the PPDP in order to solve the MSDCP. The generic parallel complexity of the MSCDP is as follows.

Theorem 3.1 (MSDCP's Generic Parallel Complexity)**.** *For some $q(n) = O(m(n))$, the parallel time complexity of the* MSDCP *is $O(\frac{m(n)}{q(n)}(t(n) + \log(q(n))))$, while the processor complexity is $p(n)q(n)$.*

Proof: The MSDCP can be solved in two phases. The first phase simultaneously solves $m(n)$ PPDP instances using $p(n)q(n)$ processors in $O(\frac{m(n)}{q(n)}t(n))$ parallel time. The second phase uses $p(n)$ processors determining which YES-instance has $d_1 + d_2$ minimized in $O(\frac{m(n)}{q(n)} \log(q(n)))$ parallel time. ■

B. PPDP*'s Parallel Complexity*

Next, we discuss the parallel complexity of the PPDP, showing that the problem can be solved in $O(\log n)$ parallel time using n^3 processors on the COMMON CRCW PRAM model. Basically, our solving technique relies on the parallel transformation that constructs a 2-SATISFIABILITY (2SAT) instance with clauses in *implicative form* from a given PPDP instance.

Lemma 3.2 (PPDP-2SAT Transformation)**.** *Given a* PPDP *instance, the corresponding* 2SAT *instance can be constructed in* $\Theta(1)$ *parallel time using* n^2 *on a* COMMON CRCW PRAM.

Proof: The transformation associates a boolean varible u_l to each entity $l \in S$ such that u_l is true if and only if $l \in C_1$, and constructs 2SAT implication clauses according to two conditions found for the dissimilarity d_{ij}.

case 1: If $d_1 < d_{ij}$, then construct $(\bar{u_i} \to u_j)$, $(\bar{u_j} \to u_i)$, $(u_i \to \bar{u_j})$ and $(u_j \to \bar{u_i})$.
case 2: If $d_2 < d_{ij} \le d_1$, then construct $(\bar{u_i} \to u_j)$ and $(\bar{u_j} \to u_i)$.

The construction can be performed in $\Theta(1)$ parallel time using n^2 by assigning each processor to each dissimilarity d_{ij}, for all $1 \le i, j \le n$. ■

Aspvall, Plass and Tarjan [12] showed that a 2SAT instance with a set $U = \{u_1, u_2, \ldots, u_n\}$ of n variables, and a collection $C = \{(x_1 \to y_1), (x_2 \to y_2), \ldots, (x_m \to y_m)\}$ of implication clauses can be represented by an *implication graph*. Basically, an implication graph is a directed graph $G = (V, E)$, where $V = \{u_i, \bar{u}_i | 1 \le i \le n\}$ and $E = \{(x_j, y_j) | 1 \le j \le m\}$. A 2SAT instance is satisfiable if and only if there is no variable u_i such that there exists a cycle containing both u_i and $\bar{u}_i$ in G, for all $1 \le i \le n$. Observe that one way to compute a 2SAT instance is by computing *transitive closure* of G.

Theorem 3.3 (PPDP's Parallel Complexity)**.** *The* PPDP *can be computed in* $O(\log n)$ *parallel time using* n^3 *processors on a* COMMON CRCW PRAM.

Proof: We apply the transformation by Lemma 3.2 to construct the corresponding 2SAT instance with $O(n^2)$ implication clauses. Observe that the construction of the implication graph can be done in $\Theta(1)$ parallel time using n^2 processors. We apply a parallel transitive closure algorithm in $O(\log n)$ parallel time using n^3 processors. ■

Remark that the optimal bipartition for the MSDCP can be constructed directly from the satisfying truth assignment for the corresponding 2SAT instance. We can incorporate with a parallel algorithm on the COMMON CRCW PRAM model for finding the satisfying truth assignment in $O(\log n)$ parallel time using n^3 processors by Chen [13] into the bipartition construction. Note that the same parallel algorithm can be simulated on the EREW PRAM model in $O(\log^2 n)$ parallel time using n^3 processors.

C. MSDCP*'s Parallel Complexity*

Observe that the optimal pair of diameters $(d(C_1), d(C_2))$ for MSCDP can be found among $O(n^4)$ trivial PPDP instances all of which is generated by trying $O(n^2)$ all possible dissimilarities between d_1 and d_2. Applying PPDP Theorem 3.3, we obtain the parallel complexity of the MSDCP as follows.

Theorem 3.4 (MSDCP's Parallel Complexity)**.** *The* MSDCP *can be computed in* $O(\log n)$ *parallel time using* n^7 *processors on a* COMMON CRCW PRAM.

Proof: This result is by applying $m(n) = O(n^4)$ and $q(n) = n^4$ in the generic parallel complexity of the MSDCP. The functions $t(n)$ and $p(n)$ follows the parallel complexity of the PPDP in Theorem 3.3. ■

We would like to remark that though the result in Theorem 3.4 places the MSDCP in class NC. However, n^7 processors used in the computation are considered generally impractical. Any NC algorithm with less processor complexity for the MSDCP would be a more appropiate parallel approach towards the real world applications. In the next section, we will propose the technique of parallel algorithm which can be applied to reduce the processor complexity to n^6.

IV. PROCESSOR IMPROVEMENT TECHNIQUE

This section presents the technique of parallel algorithm that can be applied to any MSDCP instance to reduce the number of PPDP instances required in solving the MSDCP. As a result, we show that the technique can improve the processor complexity in Theorem 3.4 by a factor of n. Intuitively, this technique is based on the result of Hansen and Jaumard [2], which implies that there can be only at most n possible disimilarities for $d(C_1)$ according to the structure of a Kruskal's *maximum spanning tree* (MST) of a complete graph $K_S = (S, S^2)$ with d as its weight function. Here, we generalize Hansen and Jaumard's result, extending the result in a more restricted case, which only holds for a Kruskal's MST, to any MST in general case. Let $\mathcal{T}$ denote any MST of a complete graph K_S. This result is as follows.

Lemma 4.1 (Possible Larger Candidate Diameters)**.** *The edges whose weights are candidate diameters for* $d(C_1)$ *are a maximum weighted edge* $(u, v) \in E(K_S) \setminus E(\mathcal{T})$ *that closes an odd cycle in* $\mathcal{T}$*, and all the edges* $e \in E(\mathcal{T})$ *with* $d_{uv} \le d_e$.

Proof: Consider an edge (u, v) of maximum weight among the edges not in $E(\mathcal{T})$. Let P_{uv} denote a path (a subset of edges) from u to v in $\mathcal{T}$. The edge (u, v) can either be the edge that closes a cycle of odd or even length in $\mathcal{T}$.

Suppose the edge (u, v) closes an odd cycle in $\mathcal{T}$. Then, $d_{uv} \le d_e$, for all edges $e \in P_{uv}$. Otherwise, $\mathcal{T}$ is not maximum (cycle property). Suppose to the contrary that $d(C_1) < d_{uv}$. Then, vertices along path P_{uv} including edge (u, v) should alternatively belong in the different clusters, but this is impossible because $|P_{uv} \cup (u, v)|$ is odd.

Suppose the edge (u, v) closes an even cycle in $\mathcal{T}$. Again, by cycle property, $d_{uv} \le d_e$, for all edges $e \in P_{uv} \subseteq E(\mathcal{T})$. Suppose that $d(C_1) = d_{uv}$. Then, vertices u and v must belong in the same cluster. Because $|P_{uv} \cup (u, v)|$ is even, at least two end vertices of one edge of P_{uv} must be in the same cluster too. It follows that $d(C_1) \ge d_{uv}$. ■

By Lemma 4.1, the lower bound on possible larger candidate diameters is the weight of a maximum weighted edge (u, v) that closes an odd cycle in $\mathcal{T}$. We formulate the computational problem called LOWER BOUND ON LARGER

CANDIDATE DIAMETER PROBLEM (LLCDP) aiming at determining the edge (u, v). Formally, the problem is defined as follows.

LOWER BOUND ON LARGER CANDIDATE DIAMETER PROBLEM (LLCDP)
Given: A complete graph K_S, a weight function d, and a MST $\mathcal{T}$ of K_S.
Problem: Determine an edge $(u, v) \in E(K_S) \setminus E(\mathcal{T})$ of maximum weight such that (u, v) closes an odd cycle in $\mathcal{T}$.

For a given MSDCP instance, the corresponding LLCDP instance can be constructed using a parallel MST algorithm in $O(\log n)$ parallel time using n^2 processors on the EREW PRAM model [14]. Note that the construction on the COMMON CRCW PRAM model can also be done with the same resource complexities.

We first introduce the parallel computation of the LLCDP. Then, we describe how to incorporate with the parallel computation to compute all possible larger candidate diameters. Subsequently, the improved bound is presented.

A. LLCDP*'s Parallel Computation*

We discuss the parallel computation of the LLCDP, showing that the LLCDP can be computed in $O(\log n)$ parallel time using n^3 processors on the COMMON CRCW PRAM model. Basically, we solve the LLCDP in three simple phases in the following procedure.

phase 1: Compute a graph $K'_S = (S, E(K_S) \setminus E(\mathcal{T}))$.
phase 2: Remove from $E(K'_S)$ all the edges that close even cycles in $\mathcal{T}$.
phase 3: Determining a maximum weighted edge (u, v) among the remaining egdes of $E(K'_S)$.

The correctness of the procedure is trivial. Observe that, by using n^2 processors, a graph K'_S in the first phase can be computed in $\Theta(1)$ parallel time, and the edge (u, v) in the third phase can be determined in $O(\log n)$ parallel time. We show that the second phase can be implemented by incorporating with a BFS-NUMBERING algorithm.

Recall that, for a graph $G = (V, E)$ with a rooted vertex $r \in V$, a BFS-NUMBERING on G is a vertex labelling $\beta : V \to \{0, 1, \ldots, p\}$, where p is the depth of the BFS's tree starting from r. The labelling starts with $\beta(r) = 0$, and assigns for each vertex $u \in V \setminus \{r\}$, $\beta(u) = \beta(r) + l$, if u is l steps away from r in the BFS's tree. Let β be a BFS-NUMBERING on $\mathcal{T}$. The property of an edge that closes an even cycle in $\mathcal{T}$ is shown as follows.

Lemma 4.2 (Cycle Evaluation by BFS-NUMBERING)**.** *An edge $(x, y) \in E(K_S) \setminus E(\mathcal{T})$ closes an even cycle in $\mathcal{T}$ if and only if $\beta(x)$ and $\beta(y)$ are not both even or odd.*

Proof: Consider an edge $(x, y) \in E(K_S) \setminus E(\mathcal{T})$. Let P_{xy} denote a path (a subset of edges) from x to y in $\mathcal{T}$. Without loss of generality, we assume $\beta(x) < \beta(y)$.

Suppose the edge (x, y) closes an even cycle in $\mathcal{T}$. Then, $|P_{xy} \cup (x, y)| = 2k$, for some positive integer $k > 1$, and $|P_{xy}| = 2k - 1$. Since x and y are $2k - 1$ steps away in $\mathcal{T}$, it follows that $\beta(y) = \beta(x) + 2k - 1$. Thus, $\beta(x)$ is even if and only if $\beta(y)$ is odd.

Conversely, suppose the edge (x, y) closes an odd cycle in $\mathcal{T}$. Then, $|P_{xy} \cup (x, y)| = 2k + 1$, for some positive integer $k \geq 1$, and $|P_{xy}| = 2k$. Since x and y are $2k$ steps away in $\mathcal{T}$, it follows that $\beta(y) = \beta(x) + 2k$. Thus, $\beta(x)$ is even if and only if $\beta(y)$ is even. ■

Observe that once a BFS-NUMBERING on $\mathcal{T}$ is computed, the second phase can be proceeded by removing all the edges whose end vertices both have odd or even labels. Clearly, using n^2 processors can carry this step in $\Theta(1)$ parallel time on a COMMON CRCW PRAM. It was shown by Gazit and Miller that computing BFS-NUMBERING on a graph with n vertices can be done in $O(\log n)$ parallel time using n^3 processors on the COMMON CRCW PRAM model [15]. Hence, the parallel complexity of the LLCDP is as follows.

Theorem 4.3 (LLCDP's Parallel Complexity)**.** *The* LLCDP *can be computed in $O(\log n)$ parallel time using n^3 processors on a* COMMON CRCW PRAM.

B. *Processing $O(n)$ Possible Larger Candidate Diameters*

By Lemma 4.1, there can be only at most n possible disimilarities for $d(C_1)$. Those are restricted to the weights of the edges $e \in E(\mathcal{T})$ for which $d_{uv} \leq d_e$. Observe that once the edge (u, v) is computed, determining those edge weights can be done in $\Theta(1)$ parallel time using n processors on the COMMON CRCW PRAM model. In fact, this step is relatively easy to see since by assigning each processor to each edge of $E(\mathcal{T})$, all the edge weights strictly greater than d_{uv} can then be determined in $\Theta(1)$ parallel time. We state this result in the following lemma.

Lemma 4.4 (LLCDP's Post-processing)**.** *Given the edge (u, v), a solution of the* LLCDP*. At most n possible disimilarities for $d(C_1)$ can be computed from $\mathcal{T}$ in constant $\Theta(1)$ using n processors on a* COMMON CRCW PRAM.

Next, we integrate the results in Lemma 4.4 and Theorem 4.3. The resource used in processing all possible disimilarities for $d(C_1)$ by Lemma 4.1 is as follows.

Theorem 4.5 (Possible Larger Candidate Diameters Processing)**.** *Given a* MSDCP *instance, $O(n)$ possible disimilarities for $d(C_1)$ can be computed in $O(\log n)$ parallel time using n^3 processors on a* COMMON CRCW PRAM.

Proof: We first apply a parallel MST algorithm by Chong [14] in $O(\log n)$ parallel time and n^2 processors on a COMMON CRCW PRAM to construct the corresponding LLCDP instance. We apply the parallel computation by Theorem 4.3 to compute the LLCDP instance. Then, we apply the parallel computation by Lemma 4.4 to compute $O(n)$ possible disimilarities. ■

C. *Improved Bound*

By Theorem 4.5, processing $O(n)$ possible disimilarities for $d(C_1)$ requires the computation tradeoff with $O(\log n)$ parallel time and n^3 processors. Observe that, by this computation, in return, we can generate $O(n^3)$ PPDP instances by trying $O(n)$ possible disimilarities for $d(C_1)$ with d_1, and $O(n^2)$ all possible disimilarities for $d(C_2)$. With these $O(n^3)$ PPDP instances, we improve the parallel complexity of the MSDCP in Theorem 3.4 as follows.

Theorem 4.6 (MSDCP's Improved Parallel Complexity). *The* MSDCP *can be computed in $O(\log n)$ parallel time using n^6 processors on a* COMMON CRCW PRAM.

Proof: This result is by applying $m(n) = O(n^3)$ and $q(n) = n^3$ in the generic parallel complexity of the MSDCP. We apply the parallel computation by Theorem 4.5 trading off with $O(n^3)$ PPDP instances. The functions $t(n)$ and $p(n)$ follows the parallel complexity of PPDP in Theorem 3.3. ■

So far, we have focused our discussion of the parallel complexity of the MSDCP on the COMMON CRCW PRAM model. As a result, when the parallel time complexity of the MSDCP in $O(\log n)$ is considered, we have improved the processor complexity by a factor of n. Nonetheless, n^6 processors used are still too large to be considered practical. In the next section, we will propose a more practical NC algorithm for the MSDCP.

V. MORE PRACTICAL PARALLEL ALGORITHM

In this section, we discuss a more practical NC algorithm for the MSDCP. The parallel algorithm in our discussion can be implemented in $O(\log^3 n)$ parallel time and $n^{3.376}$ processors on the EREW PRAM model. Basically, this parallel algorithm is based on the simplification of MSDCP's subroutines on the EREW PRAM model.

We first describe the simplification. Then, a more practical parallel algorithm on the EREW PRAM model is presented.

A. MSDCP*'s Simplification Subroutines*

We simplify the results in Theorems 3.3 and 4.3, Lemma 4.4, and Theorem 4.5, respectively. Our simplication results on the EREW PRAM model are as follows.

Theorem 5.1 (PPDP's Parallel Complexity). *The* PPDP *can be computed in $O(\log^2 n)$ parallel time using $n^{2.376}$ processors on an* EREW PRAM.

Proof: By making n^2 copies of the pair (d_1, d_2), the PPDP-2SAT transformation by Lemma 3.2 can be implemented in $\Theta(1)$ parallel time using n^2 processors on an EREW PRAM, so can the construction of the implication graph. We apply a parallel transitive closure algorithm in $O(\log^2 n)$ parallel time using $n^{2.376}$ processors on an EREW PRAM. ■

Theorem 5.2 (LLCDP's Parallel Complexity). *The* LLCDP *can be computed in $O(\log^2 n)$ parallel time using $n^{2.376}$ processors on an* EREW PRAM.

Proof: This result is due to the three-phases procedure in section IV-A. We apply a parallel BFS-NUMBERING algorithm by Gazit and Miller [15] that can be implemented in $O(\log^2 n)$ parallel time using $n^{2.376}$ processors on an EREW-PRAM in the second phase. By making n copies of a BFS-NUMBERING, the third phase can be carried in $\Theta(1)$ parallel time using n^2 processors. ■

Lemma 5.3 (LLCDP's Post-processing). *Given the edge (u, v), a solution of the* LLCDP. *At most n possible disimilarities for $d(C_1)$ can be computed from $\mathcal{T}$ in $O(\log n)$ parallel time using n processors on an* EREW PRAM.

Proof: By making n^2 copies of the edge (u, v), the computation by lemma 4.4 can be implemented in $\Theta(1)$ parallel time using n^2 processors on an EREW PRAM. ■

Theorem 5.4 (Possible Larger Candidate Diameters Processing). *Given a* MSDCP *instance, $O(n)$ possible disimilarities for $d(C_1)$ can be computed in $O(\log^2 n)$ parallel time using $n^{2.376}$ processors on an* EREW PRAM.

Proof: We first apply a parallel MST algorithm by Chong [14] in $O(\log n)$ parallel time and n^2 processors on an EREW PRAM to construct the corresponding LLCDP instance. We apply the parallel computation by Theorem 5.2 to compute the LLCDP instance. Then, we apply the parallel computation by Lemma 5.3 to compute $O(n)$ possible disimilarities. ■

B. The EREW PRAM *Algorithm*

We discuss the NC algorithm for the MSDCP on the EREW PRAM model. Our parallel algorithm is assembled with the simplication of MSDCP's subroutines on the EREW PRAM model. The parallel algorithm is obtained by adjusting the configuration of the improved parallel complexity of the MSDCP in Theorem 4.6 directly. This parallel algorithm is as follows.

Theorem 5.5 (Basic EREW PRAM Algorithm). *There exists a parallel algorithm for the* MSDCP *with $O(n^2 \log^2 n)$ parallel time using $n^{3.376}$ processors on an* EREW PRAM.

Proof: The parallel algorithm is due to the generic parallel complexity of the MSDCP. We apply the parallel computation by Theorem 5.4 trading off with $O(n^3)$ PPDP instances. We apply $m(n) = O(n^3)$ and $q(n) = n$. The functions $t(n)$ and $p(n)$ follows the parallel complexity of PPDP in Theorem 5.1. ■

Notice that the time complexity of the parallel algorithm is $O(n^2 \log^2 n)$, as $\frac{m(n)}{q(n)} = O(n^2)$. In particular, with $q(n) = n$, any parallel algorithm by the generic parallel complexity of the MSDCP is likely to require at least a polynomial ratio in $\frac{m(n)}{q(n)}$, unless $m(n) = O(n(s(n)))$, for some polylogarithmic $s(n)$. We claim that $m(n)$ can be further reduced from $O(n^3)$ to $O(n \log n)$ under the parallel complexity with $O(\log n)$ parallel time and $n^{3.376}$ processors. Intuitively, we reduce $O(n^3)$ PPDP instances by employing a sorting algorithm which sorts each $O(n^2)$ PPDP instances with a fixed d_1 out. Indeed, the notion behind the sorting is to seperate each $O(n^2)$ PPDP instances with a fixed d_1, and then arrange them in monotonically increasing order on d_2, so that the computation among these $O(n^2)$ PPDP instances can be proceeded by $p(n)$ processors in binary searching fashion to locate a minimum d_2 in $O(t(n) \log n)$ parallel time. Observe that the binary searching is capable because if a pair (d_1, d_2) is possible to be the clusters' diameters, then so is any pair (d_1, d_l) with $d_l \geq d_2$. As a result, this notion leaves just $O(n \log n)$ PPDP instances required solving. Note that it is know that sorting n integers on the EREW PRAM model can be optimally done in $O(\log n)$ parallel time using n processors. Thus, reducing $O(n^3)$ to $O(n \log n)$ PPDP instances requires the computation tradeoff with $O(\log n)$ parallel time and n^3 processors. Hence, the improved version of the parallel algorithm for the MSDCP is as follows.

Theorem 5.6 (Improved EREW PRAM Algorithm)**.** *There exists a parallel algorithm for the* MSDCP *with* $O(\log^3 n)$ *parallel time using* $n^{3.376}$ *processors on an* EREW PRAM.

Proof: The parallel algorithm is due to the generic parallel complexity of the MSDCP. We apply the parallel computation by Theorem 5.4, and the parallel sorting algorithm consecutively, trading off with $O(n \log n)$ PPDP instances. We apply $m(n) = O(n \log n)$ and $q(n) = n$ processors. The functions $t(n)$ and $p(n)$ follows the parallel complexity of PPDP in Theorem 5.1. ■

In fact, any parallel algorithm on a COMMON CRCW PRAM can be simluated on an EREW PRAM within $O(\log n)$ parallel time using the same number of processors. Note that this can be done by making n copies of the input required in the computation in each parallel time step. Straightforwardly, the improved parallel complexity of the MSDCP in Theorem 4.6 yeilds a parallel algorithm with $O(\log^2 n)$ parallel time using n^6 processors on an EREW PRAM. Remark that the parallel algorithm by theorem 5.6 is considered more practical though it performs slower within $O(\log n)$ parallel time since it requires many less processors.

VI. CONCLUSIONS AND DISCUSSIONS

In this paper, minimum sum of diameters clustering has been considered as the optimization problem called MINIMUM SUM OF DIAMETERS CLUSTERING PROBLEM (MSDCP). For the case of a partition into $k = 2$ clusters, we have shown that the MSDCP in parallel belongs in complexity class NC, that is, the problem has a polylogarithmic time parallel algorithm with polynomial processor complexity. Fundamentally, our complexity results on the MSDCP are based on the generic complexity defined in the terms of the parallel complexity of the decision problem called POSSIBLE PAIR OF DIAMETERS PROBLEM (PPDP). By revealing that the PPDP can be solved in $O(\log n)$ parallel time using n^3 processors the COMMON CRCW PRAM model, we showed that the parallel complexity of the MSDCP on the COMMON CRCW PRAM model is $O(\log n)$ parallel time and n^7 processors. Accordingly, we have introduced the technique of parallel algorithm which can be applied to a MSDCP instance to improve the processor complexity by a factor of n. Mainly, this technique ties up with the parallel complexity of the computational problem called LOWER BOUND ON LARGER CANDIDATE DIAMETER PROBLEM (LLCDP). On the COMMON CRCW PRAM model, we showed that the LLCDP requires $O(\log n)$ parallel time and n^3 processors. As a result, when the parallel time complexity of the MSDCP in $O(\log n)$ is considered, we have improved the processor complexity to n^6. In addition, regarding the issue of high processor complexity, we have also proposed a more practical NC parallel algorithm for the MSDCP. This parallel algorithm is assembled with the simplication of MSDCP's subroutines on the EREW PRAM model. By showing that both the PPDP and the LLCDP can be solved in $O(\log^2 n)$ parallel time using $n^{2.376}$ processors on an EREW PRAM, the more practical parallel algorithm can be implemented in $O(\log^3 n)$ parallel time using $n^{3.376}$ processors on the EREW PRAM model.

There are some remaining aspects of minimum sum of diameters clustering that are of theoretical interest. For the case of a partition into two clusters, in parallel, the lower bound on processor complexity remains open when the parallel complexity in $O(\log n)$ time is considered. The possibly loose lower bound is n^3 processors, as theoretically suggested by the best known $O(n^3)$-sequential algorithm by Ramnath [3]. Also, any NC algorithm with less processor complexity would be way useful in many practical applications. For the case of a partition into three or clusters, hardness in some approximation classes for minimum sum of diameters remains open.

ACKNOWLEDGMENT

Financial support from the Thailand Research Fund through the Royal Golden Jubilee Ph.D. Program (Grant No. PHD/0267/2552) to student's initial NJ, advisor's initial SK, and co-advisor's initial RG is acknowledged. The authors would like to thank Raymond Greenlaw for many useful comments and discussions.

REFERENCES

[1] P. Brucker, "On the complexity of clustering problems," in *Optimization and Operations Research*, ser. Lecture Notes in Economics and Mathematical Systems, R. Henn, B. Korte, and W. Oettli, Eds. Springer Berlin Heidelberg, 1978, vol. 157, pp. 45–54.

[2] P. Hansen and B. Jaumard, "Minimum sum of diameters clustering," *Journal of Classification*, vol. 4, no. 2, pp. 215–226, 1987.

[3] S. Ramnath, "Dynamic digraph connectivity hastens minimum sum-of-diameters clustering," *SIAM Journal on Discrete Mathematics*, vol. 18, no. 2, pp. 272–286, 2004.

[4] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical Programming*, vol. 79, no. 1-3, pp. 191–215, 1997.

[5] J. A. Hartigan, *Clustering Algorithms*, 99th ed. New York, NY, USA: John Wiley & Sons, Inc., 1975.

[6] M. Delattre and P. Hansen, "Bicriterion Cluster Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, pp. 277–291, 1980.

[7] R. M. Cormack, "A review of classification," *Journal of the Royal Statistical Society. Series A (General)*, vol. 134, no. 3, pp. pp. 321–367, 1971.

[8] M. R. Rao, "Cluster analysis and mathematical programming," *Journal of the American Statistical Association*, vol. 66, no. 335, pp. pp. 622–626, 1971.

[9] R. Greenlaw and S. Kantabutra, "On the parallel complexity of hierarchical clustering and cc-complete problems," *Complexity*, vol. 14, no. 2, pp. 18–28, 2008.

[10] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, *Limits to Parallel Computation: P-completeness Theory*. New York, NY, USA: Oxford University Press, Inc., 1995.

[11] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Array, Trees, Hypercubes*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992.

[12] B. Aspvall, M. F. Plass, and R. E. Tarjan, "A linear-time algorithm for testing the truth of certain quantified boolean formulas," *Information Processing Letters*, vol. 8, no. 3, pp. 121 – 123, 1979.

[13] Z.-Z. Chen, "A fast and efficient parallel algorithm for finding a satisfying truth assignment to a 2-cnf formula," *Information Processing Letters*, vol. 43, no. 4, pp. 191 – 193, 1992.

[14] K. W. Chong, Y. Han, and T.-W. Lam, "Concurrent threads and optimal parallel minimum spanning trees algorithm," *J. ACM*, vol. 48, pp. 297–323, 2001.

[15] H. Gazit and G. L. Miller, "An improved parallel algorithm that computes the BFS numbering of a directed graph," *Information Processing Letters*, vol. 28, no. 2, pp. 61 – 65, 1988.