JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 14, Number 2, 2007
© Mary Ann Liebert, Inc.
Pp. 131–143
DOI: 10.1089/cmb.2007.0004

Fast Molecular Shape Matching Using Contact Maps

PANKAJ K. AGARWAL,[1] NABIL H. MUSTAFA,[2] and YUSU WANG[3]

ABSTRACT

In this paper, we study the problem of computing the similarity of two protein structures by measuring their *contact-map overlap*. Contact-map overlap abstracts the problem of computing the similarity of two polygonal chains as a graph-theoretic problem. In $\mathbb{R}^3$, we present the first polynomial time algorithm with any guarantee on the approximation ratio for the 3-dimensional problem. More precisely, we give an algorithm for the contact-map overlap problem with an approximation ratio of σ, where $\sigma = \min\{\sigma(P_1), \sigma(P_2)\} \leq O(n^{1/2})$ is a decomposition parameter depending on the input polygonal chains P_1 and P_2. In $\mathbb{R}^2$, we improve the running time of the previous best known approximation algorithm from $O(n^6)$ to $O(n^3 \log n)$ at the cost of decreasing the approximation ratio by half. We also give hardness results for the problem in three dimensions, suggesting that approximating it better than $O(n^\varepsilon)$, for some $\varepsilon > 0$, is hard.

Key words: shape matching, molecular structures, contact maps, graph theory.

1. INTRODUCTION

PROTEINS, the machines and building blocks of living cells, fulfill a variety of roles such as acting as enzymes, transporting small molecules, forming many cellular structures, regulating cell processes, and carrying signals. A protein is simply a polymer consisting of a long chain of amino acid residues. Although a linear sequence, inter-atomic forces between residues bend and twist the chain in a way characteristic for each protein. They cause the protein molecule to curl up into a specific 3-dimensional structure called the *folded state* of the protein. The structure of a protein can be captured by its amino acid residue sequence and its folded structure. A commonly used simplified model to represent the backbone of a protein is the so-called lattice model (Kolinski and Skolnick, 1994), in which the backbone is mapped to a non-self-intersecting path on an integer grid, called a *self-avoiding walk* (Fig. 1).

It is a fundamental axiom of biology that the 3-dimensional structure of a protein has a crucial influence on its function—two proteins that are similar in their 3-dimensional structure will likely have similar functions (Leach, 1996). Comparing two protein structures for similarity is therefore a crucial task and has been extensively investigated (Halperin et al., 2002).

Since it is not clear what quantitative measure to use for comparing protein structures, a multitude of measures have been proposed. Each measure aims at capturing the intuitive notion of similarity. The measures fall into two main categories.

[1]Dept. of Computer Science, Duke Univ., Durham, North Carolina.
[2]Lahore University of Management Sciences, Lahore 54792, Pakistan.
[3]Dept. of Computer Science and Engineering, Ohio State Univ., Columbus, Ohio.

FIG . 1. Protein backbone and its corresponding 3D lattice model.

Inter-distance shape matching methods compute a mapping of residues of protein $\mathcal{A}$ to residues of protein $\mathcal{B}$ such that the distance between the mapped residues is minimized (Gerstein and Levitt, 1998; Godzik, 1996; Maiorov and Crippen, 1994). A commonly used measure is *root mean square distance* (RMSD), which is the root of sum of the squares of Euclidean distances between mapped residues (Maiorov and Crippen, 1994). One problem with these types of methods is that the distance of two residues depends on their relative position in space. Therefore, the matching process typically consists of two intertwined subtasks, namely, finding the mapping of residues from $\mathcal{A}$ to $\mathcal{B}$, and finding the best rotation and translation such that the metric distance (e.g., RMSD) is minimized under that mapping.

Intra-distance matching methods try to find a set of residues $\mathcal{A}'$ of protein $\mathcal{A}$ and subset $\mathcal{B}'$ of protein $\mathcal{B}$, such that the pairwise distance matrix of $\mathcal{A}'$ is similar to the pairwise distance matrix of $\mathcal{B}'$ (Goldman et al., 1999; Holm and Sander,1993,1996; Taylor and Orengo,1989, 1996). These methods have the advantage that they are motion invariant since the distance between two residues of $\mathcal{A}$ remains the same under rigid transformations.

We study the *contact-map overlap* measure, first proposed in Goldman et al., (1999). Contact-map overlap measures the similarity between two proteins (in the lattice model) based on the pairwise distances of the C_α-atoms of each protein. It falls within the intra-distance similarity measure category. The contact-map of a protein is a graph, which represents the three dimensional structure of the protein by modeling the neighborhood of each residue by edges (or contacts) to the neighbors. Then two protein structures are considered similar if one can find mapping of vertices from one to the other such that the pattern of the neighborhoods is similar for a large number of mapped vertices.

This measure has been found to be very useful for measuring protein similarity—it is robust, takes partial matching into account, translation invariant and captures the intuitive notion of similarity very well for details (Goldman et al., 1999; Lancia et al., 2001). Using various heuristics, it was shown in Lancia et al. (2001) that the contact-map overlap measure captures strongly the established notions of similarity and dissimilarity of protein structures: the heuristics to maximize contact-map overlap were run on 33 proteins (due to computational reasons, only small protein structures were tried), classified into four families, and the evaluation of similarity computed by their heuristic very strongly correlated with the current classification, achieving 98.7% accuracy (with 1.3% false negatives and 0.0% false positives). Thus the problem of designing efficient algorithms that guarantee the contact-map overlap quality is an important one that has eluded researchers so far.

1.1. Problem definition

Let Z be the integer grid in $\mathbb{R}^2$ or $\mathbb{R}^3$. A self-avoiding walk $P = \langle v_1, \ldots, v_n \rangle$ is a directed path on Z such that $\|v_i v_{i+1}\| = 1$, and $v_i \neq v_j$, $1 \leq i, j \leq n$. The *contact map* of P is an ordered graph $G(P) = (V, E)$ (an ordered graph is a graph with its vertices ordered by a linear sequence), where $V = \langle v_1, \ldots, v_n \rangle$, and $E = \{(v_i, v_j) \mid |j - i| > 1 \text{ and } \|v_i v_j\| = 1\}$. (Fig. 2).

Given two ordered graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $V_1 = \langle v_1, \ldots, v_n \rangle$ and $V_2 = \langle u_1, \ldots, u_m \rangle$, the *maximal (partial) contact-map overlap* of G_1 and G_2 is

$$\nu(G_1, G_2) = \max_f |\{(f(v_i), f(v_j)) \in E_2 : (v_i, v_j) \in E_1\}|$$

where the maximum is taken over all $f : V_1' \to V_2$, where $V_1' \subseteq V_1$ and f is an order-preserving bijection, i.e., $i < j$, $u_l = f(v_i)$, and $u_k = f(v_j)$ implies that $l < k$. Note that a vertex of G_1 need not be mapped

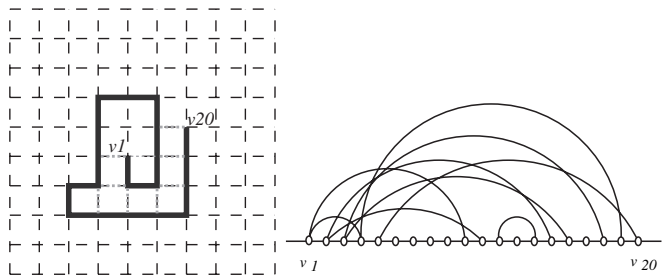

FIG . 2. A self-avoiding walk (bold) on a planar grid and its contact map.

to a vertex of G_2 in the above definition. We modify the problem to include this restriction (Fig. 3). More precisely, given $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, define the *complete contact-map overlap* as

$$\mu(G_1, G_2) = \max_g |\{(g(v_i), g(v_j)) \in E_2 : (v_i, v_j) \in E_1\}|$$

where the maximum is taken over all order-preserving bijections $g : V_1 \to V_2$.

The case that a vertex of G_1 is not mapped to any vertex of G_2 (while preserving the order) corresponds to the deletion of a C_α-atom in the protein backbone during the comparison. Therefore $\mu(G_1, G_2)$ enforces no deletion of vertices for one of the protein backbone, e.g., a motif structure; while $\nu(G_1, G_2)$ allows unlimited deletions for both structures. We refer the reader to Francesco et al. (1996) for a more detailed discussion on the effects of deletions on protein matching. Note that both measures can be viewed as two special cases of a more general and more important problem in which at most k deletions are allowed for one of the structures.

1.2. Previous results

Goldman et al. (1999) show that computing the optimal value of $\mu(G_1, G_2)$ is NP-hard. They propose a 3-approximation algorithm with $O(n^6)$ running time if the contact-maps are derived from self-avoiding walks on a 2-dimensional lattice. They raise open questions whether the structure of 3-dimensional self-avoiding walks can be used to aid in the contact-map overlap problem, and if any algorithms exist for $\mathbb{R}^3$.

Recently, Lancia et al. (2001) gave a branch-and-bound method (based on a linear programming relaxation) for the contact-map overlap problem in $\mathbb{R}^3$. They formulate the problem as a maximum independent set problem, and then employ various branch-and-bound techniques to solve the problem efficiently in practice. Even with heuristics and branch-and-bound techniques, they state proteins of size about three hundred as a computationally limiting size (Lancia et al., 2001). A similar optimization method (based on Lagrangian relaxation) was given in Caprara and Lancia, (2002).

We also note that the *contact-map overlap* measure is in fact an ordered version of the *maximum common subgraph* problem (Garey and Johnson, 1979) (the vertex set of the two graphs is given as a linear ordering and the mapping must be order preserving). It is known that the maximum common subgraph problem is not approximable with an absolute error guarantee of $\min(|V_1|^\varepsilon, |V_2|^\varepsilon)$, for some $\epsilon > 0$ (Verbitsky, 1994).

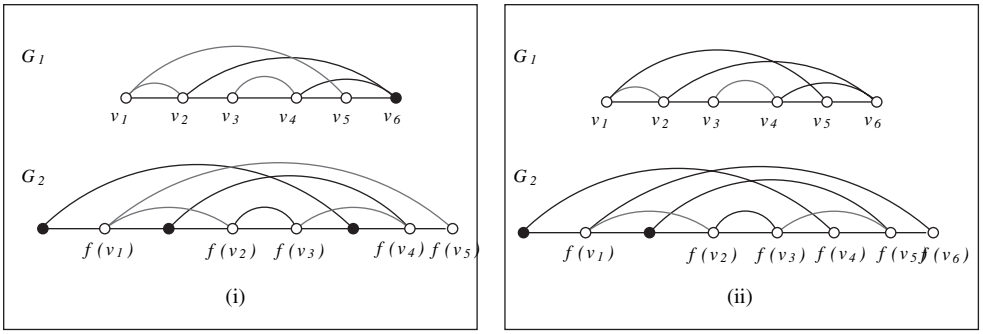

FIG . 3. (i) Partial contact-map overlap ($\nu(G_1, G_2) = 3$), (ii) complete contact-map overlap ($\mu(G_1, G_2) = 2$). Shaded vertices are not matched.

1.3. Our results

We focus on the computation of the contact-map overlap both in $\mathbb{R}^2$ and $\mathbb{R}^3$. Our main contributions are:

- We present a 6-approximation algorithm for computing the maximum contact-map overlap of two self-avoiding walks in $\mathbb{R}^2$, with running time $O(n^3 \log n)$. A c-approximation algorithm for some function $f(G_1, G_2)$ returns a value at least $\frac{1}{c} f(G_1, G_2)$.
- Based on a procedure that decomposes the input graph P_1 into $\sigma(P_1) \leq O(n^{1/2})$ subgraphs (satisfying certain constraints), we propose a σ-approximation algorithm for computing the contact-map overlap of two three-dimensional self-avoiding walks, where $\sigma = \min\{\sigma(P_1), \sigma(P_2)\}$ is a parameter depending on the input chains P_1 and P_2. The running time of our algorithm is $O(n^3 \log n)$. This solves the open problem of Goldman et al. (1999), and is the first provably good approximation algorithm for the three-dimensional case. This result is of additional interest since while no approximation algorithms exist for the maximum common subgraph problem, our results give a $\sqrt{m}$-approximation algorithm for the "ordered" maximum subgraph problem, where m is the number of edges in the graph.
- We test the graph decomposition procedure on a set of 22 protein lattice structures, showing that in practice, the input graph is decomposed into only a small number of subgraphs, instead of the $O(\sqrt{n})$ theoretical worst case bound.
- We show that any degree-1 graph can be embedded as a 3-dimensional self-avoiding walk, which solves the other open problem of Goldman et al. (1999). In particular, this implies that the contact-map overlap problem for two graphs derived from 3-dimensional self-avoiding walks is *MAXSNP*-hard.
- We present various hardness results suggesting that approximating the complete contact-map overlap better than approximation ratio $O(n^{\varepsilon})$, $\varepsilon > 0$, is hard.

Overview. We present our results for two dimensions in Section 2, and our algorithm, a heuristic, and the empirical feasibility of our decomposition algorithms in Section 3. Hardness results for three dimensions are presented in Section 4, and we conclude with discussion in Section 4.

2. TWO DIMENSIONS

Given two self-avoiding walks G_1 and G_2, Goldman et al. (1999) proposed an approximation algorithm by decomposing G_1 into "simpler" graphs which are then matched optimally against G_2. These simpler graphs are defined as follows.

Let $e = (v_i, v_j)$ and $e' = (v_{i'}, v_{j'})$ be two edges of G. We say that e and e' are *disjoint* if $(i, j) \cap (i', j') = \emptyset$, and that e *contains* e' if $[i, j] \supseteq [i', j']$. Two edges e and e' *overlap* if $[i, j] \cap [i', j'] \neq \emptyset$ and one edge does not contain the other (unless they share one endpoint).

We define various classes of graphs in terms of the relations between their edges (Fig. 4). A graph $G = (V, E)$ is called a *stack* if any two edges in E either contain one another, or are disjoint. G is called a *queue* if any two edges in E either overlap each other or are disjoint. Finally, G is called a *staircase* if E can be partitioned into a family of sets so that every pair of edges within a set mutually overlap (Fig. 4) and edges from different sets are disjoint.

A graph of a particular type with maximum degree k is referred to as a k-graph (e.g., a 1-stack is a stack with maximum degree 1). The degree of each vertex in Z is four in $\mathbb{R}^2$ and six in $\mathbb{R}^3$, and therefore the maximum degree of a vertex of G is at most two in $\mathbb{R}^2$, and at most four in $\mathbb{R}^3$ (except possibly for v_1 and v_n, whose degrees can be at most three or five respectively). The left and right endpoints of an edge $e = (v_i, v_j) \in E$, $i < j$, are v_i and v_j respectively.

In the rest of this paper, we sometimes refer to edges in E as contacts. A contact map generated from a d-dimensional self-avoiding walk is also called a d-dimensional contact map. The overall approach for computing both $\mu(G_1, G_2)$ and $\nu(G_1, G_2)$ is similar so we only describe the algorithm for the latter. In the following two subsections, we first describe how to match a 1-staircase or a 2-stack with any contact map.

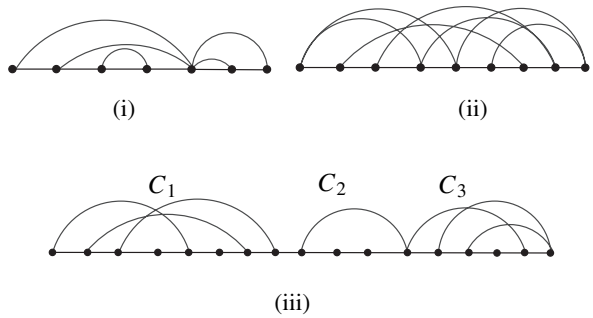

FIG . 4. (i) Stack, (ii) queue and (iii) staircase.

2.1. Staircase and self-avoiding walk

In this subsection, we describe an algorithm for computing partial and complete contact map overlaps between a staircase and the graph of a self-avoiding walk. We begin by stating an obvious structural claim whose proof is omitted.

Lemma 2.1. *Let a and b be two adjacent vertices of the integer grid in the plane. Then the length of any simple path from a to b is odd.*

Lemma 2.2. *Let $T = (V, E)$ be a 2-staircase which is a subgraph of some 2-dimensional contact map. Then E can be partitioned in $O(|E|)$ time into two subsets E_1, E_2 so that (V, E_1) and (V, E_2) are 1-staircases.*

Proof. Let V' be the set of degree two vertices in V. It follows from Lemma 2.1. that the subgraph induced by V' does not contain any cycle of odd length (a cycle in E corresponds to a cycle in the grid). For each degree-2 chain (Fig. 5) or even-length cycle, we assign edges to E_1 and E_2 alternately. The edges induced by $V \setminus V'$ are adjacent to degree-1 vertices, and can simply be assigned to E_1. Since a subgraph of a staircase is also a staircase, (V, E_1) and (V, E_2) are both 1-staircases. ■

We refer to a set of mutually overlapping edges as a cluster. Then each 1-staircase consists of a family of disjoint clusters. Let $T = \langle u_1, u_2, \ldots, u_m \rangle$ be a 1-staircase, and let $\langle C_1, \ldots, C_k \rangle$ be the ordered set of clusters of T. Set $|C_i| = m_i$. Let $G = (V, E)$ be a contact map, where $V = \langle v_1, \ldots, v_n \rangle$. For $1 \le i \le j \le n$, let G_{ij} be the subgraph of G induced by $v_i, \ldots, v_j$, and for $1 \le i \le k$, let T_i be the subgraph of T induced by $C_1, \ldots, C_i$. For $1 \le i \le k$, and $1 \le j \le n$, let $D(i, j) = \nu(T_i, G_{1j})$. Among the optimal contact maps from T_i to G_{1j}, let $f^* : T_i \to G_{1j}$ be the map so that the sequence $(f^*(u_1), \ldots, f^*(u_M))$, $M = \sum_{r=1}^{i} m_r$, is largest in reverse-lexicographic order. Here we are assuming that if $u_r \in T_i$ is not mapped to a vertex of G_{1j}, then $f^*(u_r) = 0$.

Suppose v_{h^*} is the smallest vertex of G_{1j} to which a vertex of C_i is mapped and r^* edges of C_i map to G_{1j} under f^*. By definition of ν,

$$D(i, j) = D(i - 1, h^* - 1) + r^*.$$

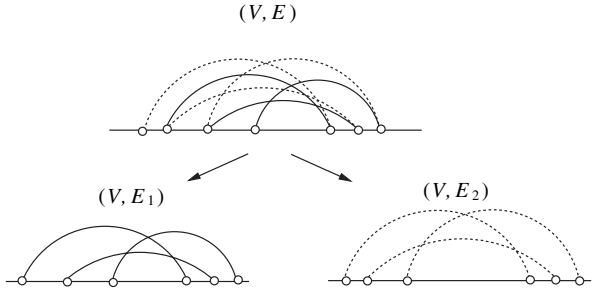

FIG . 5. A 2-staircase (V, E), and its decomposition into two 1-staircases (V, E_1) (solid) and (V, E_2) (dotted) .

For $1 \le j \le n$ and $0 \le r \le m_i$, we define $\Phi(j, r)$ to be the largest integer h so that G_{hj} contains a cluster of size r as its subgraph.

Lemma 2.3. $h^* = \Phi(j, r^*)$.

Proof. Since G_{h^*j} contains a cluster of size r^*, $h^* \le \Phi(j, r^*)$. If $h^* < \Phi(j, r^*) = h$, then let $V_i = \langle v_{i_1}, v_{i_2}, \ldots, v_{i_{2r^*}} \rangle$ be the sequence of vertices in G_{hj} that form a cluster of size r. We map $2r^*$ vertices of C_i to V_i and obtain another map $\tilde{f} : T \to G$ such that $\tilde{f}(T) >_{\text{rev}} f^*(T)$, a contradiction. ■

Lemma 2.3 implies the following

$$D(i, j) = \max_{0 \le r \le m_i} D(i-1, \Phi(j, r) - 1) + r.$$

Assuming that we have $\Phi(j, r)$ at our disposal, we can compute $D(i, j)$ in $O(m_i)$ time, thereby implying that the total time spent in computing $D(\cdot, \cdot)$ is $O(\sum_i m_i \cdot n) = O(mn)$.

In order to compute $\Phi(\cdot, \cdot)$, we define an auxiliary function $\tilde{\Phi}(e, r)$, for each edge $e \in G$ and for $r \le n$, which is the largest integer h so that G_{hj} contains a cluster of size r with e as the rightmost edge of the cluster. Then

$$\Phi(j, r) = \max\{\Phi(j-1, r), \max_{e=(i,j) \in E} \tilde{\Phi}(e, r)\}.$$

Let $e = (i, j)$ be an edge of G. Define $E_{ij} = \{(x, y) \mid x < i, i < y < j\}$. We regard each edge $e = (x, y)$ as a point $\tilde{e} = (x, y)$ in $\mathbb{R}^2$. As in Knuth (1973), by sweeping a horizontal line from top to bottom, we can compute, in time $O(n \log n)$, for each integer $l \le j$, the length λ_l of the longest monotonically decreasing sequence (both in x- and y-directions) whose y-coordinates are in the interval $[l, j]$. Then $\tilde{\Phi}(e, r) = k + 1$ if k is the largest y-coordinate with $\lambda_k = r$. Since G has $O(n)$ edges, the total time spent in computing $\tilde{\Phi}(\cdot, \cdot)$ is $O(n^2 \log n)$.

Putting everything together, we obtain the following theorem.

Theorem 2.4. *Let G be any two-dimensional contact map with n vertices, and let T be a 1-staircase with m vertices. Then $\nu(G, T)$, as well as $\mu(G, T)$, can be computed in $O(mn + n^2 \log n)$ time.*

2.2. *Stack and self-avoiding walk*

Theorem 2.5. *Let G be any two-dimensional contact map with n vertices, and let T be a 2-stack with m vertices. Then $\nu(G, T)$, as well as $\mu(G, T)$, can be computed in $O(mn^2 \log m)$ time.*

Proof. A stack T can be viewed as a tree (Fig. 6), where each node of the tree corresponds to an edge of T, and an edge e containing another edge e' in T translates to e' being the descendent of e in the tree.

We first describe how to find the optimal match between a 1-stack $T = (V, E)$ and a degree-1 contact-map $G = (V', E')$, of size m and n respectively. Let $e \in E$ be the edge with the highest right endpoint. Since T is a stack, e is not contained in any other edge. Similarly, let $e' \in E'$ be the rightmost edge of G. Then, in the optimal matching, if both e and e' are present, then they must match to one another. This leads to a dynamic programming algorithm, where we compute the optimal match between subgraphs of T and G

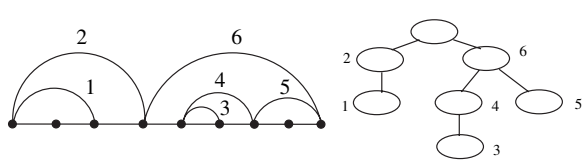

FIG. 6. A stack with its corresponding ancestor-descendant tree.

induced by every pair of vertices — Compute the table D, where $D(i, j, i', j')$ denotes the optimal match between subgraphs induced by $v_i, \ldots, v_j$, and $v'_{i'}, \ldots, v'_{j'}$ as:

$$D(i, j, i', j') = \max\{D(i, j-1, i', j'), D(i, j, i', j'-1), \\ (D(i, l-1, i', l'-1) + D(l, j-1, l', j'-1) + 1)\}$$

where $(v_l, v_j) \in E$ and $(v'_{l'}, v'_{j'}) \in E'$. Thus D can be computed in $O(m^2 n^2)$ time. This is essentially the approach used in Goldman et al. (1999).

The above recursion matches the rightmost edges of current subgraphs, resulting in a "*right matching*" algorithm. But note that we could also have matched leftmost edges as well — say denoted as a "*left matching*" algorithm, with still $O(n^4)$ running time. Furthermore, the two algorithms can be interleaved—at some steps we match rightmost edges, and at some steps the leftmost. Using a tree-decomposition scheme by Klein (1998), we can compute a sequence of left and right matchings steps such that only $O(m \log m)$ pairs of V need to be visited, and thus computed (though $O(n^2)$ pairs of V' still need to be computed) during the dynamic programming procedure. As a result, the size of table D is reduced to $O(mn^2 \log m)$, and each entry can be computed in constant time. With some more care, vertices with degree two can also be treated within the same time bound. This proves the theorem. ■

2.3. Overall algorithm

The overall algorithm is similar to the one described in Goldman et al. (1999). Given two two-dimensional contact maps G_1 and G_2, we first decompose G_1 into two 2-stacks and two 2-staircases. Each 2-staircase is then decomposed into two 1-staircases in linear time by Lemma 2.1. We then compute the maximum overlap of G_2 with these six graphs (Theorems 2.4 and 2.5), and take the maximum of them. This gives a 6-approximation of $\nu(G_1, G_2)$ and $\mu(G_1, G_2)$.

Theorem 2.6. *Given two self-avoiding walks G_1 and G_2, $\nu(G_1, G_2)$ and $\mu(G_1, G_2)$ can be approximated within a factor of 6 in time $O(n^3 \log n)$.*

3. THREE DIMENSIONS

In this section, we inspect the partial and complete contact-map overlap problems for 3-dimensional self-avoiding walks. In what follows, we first give a $\sqrt{n}$-approximation algorithm for solving $\nu(G_1, G_2)$ and $\mu(G_1, G_2)$, and then present a local improvement algorithm that achieves a worse approximation ratio, but performs better in practice. Finally, we present empirical results showing that for protein structures the approximation ratio of both the algorithm and the local improvement algorithm is much better than the worst case bounds.

Let $G = (V, E)$ be an ordered graph. We sort the edges in E in increasing order of their left endpoints, $\{e_1 = (v_{s_1}, v_{t_1}), \ldots, e_u = (v_{s_u}, v_{t_u})\}$, where $s_1 \leq \ldots \leq s_u$, and let $\tau = \langle t_1, t_2, \ldots, t_u \rangle$. Note that each s_i need not be unique, and if $s_i = s_j$, then the edge with smaller t comes first. Define $\sigma(G)$ to be the smallest integer k such that τ can be decomposed into k monotonically non-increasing or monotonically nondecreasing sequences. Set $\sigma = \min\{\sigma(G_1), \sigma(G_2)\}$.

3.1. $\sqrt{n}$-approximation algorithm

Theorem 3.1. *Given two ordered graphs G_1 and G_2 of some constant degree and size n, one can compute a σ-approximation to $\mu(G_1, G_2)$ and $\nu(G_1, G_2)$ in time $O(n^3 \log n)$, where $\sigma \leq \sqrt{n}$.*

The approach is similar to the one taken in the previous section—we will decompose G_1 or G_2 into at most σ stacks and staircases, in $O(\sigma n \log n)$ time. As shown in the last section (Theorems 2.4 and 2.5), the contact-map overlap of each stack or staircase can be computed in polynomial time. We finally return $\nu(G_1, G_2)$

as the maximum contact-map overlap of the smaller graphs with G_1 or G_2 as our solution, achieving an approximation factor of at most σ (by the pigeonhole principle). Since the smaller graphs partition the input graph, summing the running time of matching each smaller graph yields the running time $O(n^3 \log n)$.

Lemma 3.2. *Given a graph $G = (V, E)$, we can compute graphs $T_1 = (V, E_1), \ldots, T_{\sigma(G)} = (V, E_{\sigma(G)})$ in $O(\sigma n \log n)$ time such that the edges $E_1, \ldots, E_{\sigma(G)}$ partition E, and each T_i is either a stack or a staircase and $\sigma(G) \leq O(\sqrt{n})$.*

Proof. Let $V = \langle v_1, \ldots, v_n \rangle$ be the ordered set of vertices of G. We perform many iterations over the sequence of integers τ which we defined earlier. Each iteration extracts the largest monotonically increasing or monotonically decreasing subsequence of τ in $O(n \log n)$ time (Knuth, 1973). From the Erdös and Szekeres (1935) theorem, the number of iterations needed are at most $O(\sqrt{n})$, since the maximum degree of G is constant so τ has length $O(n)$.

Let the subsequence extracted at the k-th iteration be $\tau_k = \langle t_{i_1}, \ldots, t_{i_l} \rangle$. Then set T_k to be the subgraph induced by the edges $E_k = \{(s_{i_1}, t_{i_1}), \ldots, (s_{i_l}, t_{i_l})\}$. We now show that each T_k obtained this way is either a stack or a queue. First consider the case of increasing subsequence. Take two edges, say $e_i = (v_{s_i}, v_{t_i})$ and $e_j = (v_{s_j}, v_{t_j})$ such that $t_i \in \tau_k$ at index s_i, and $t_j \in \tau_k$ at index s_j. Assume w.l.o.g. that $t_i < t_j$. Since τ_k is an increasing subsequence, it must be that $s_i < s_j$. Then the edge e_i does not contain the edge e_j (since $t_i < t_j$) and e_j does not contain e_i (since $s_i < s_j$). And therefore T_i is a queue.

Now consider the case when τ_k is a decreasing subsequence. Take two edges, say $e_i = (v_{s_i}, v_{t_i})$ and $e_j = (v_{s_j}, v_{t_j})$, such that $t_i \in \tau_k$ (at index s_i), and $t_j \in \tau_k$ (at index s_j). Assume w.l.o.g. that $t_i < t_j$. Since τ_k is an decreasing subsequence, it must be that $s_i > s_j$. Then we know that the left endpoint index of edge e_j, s_j is less than the left endpoint index of e_i, s_i and the right endpoint index of e_j, t_j is larger than the right endpoint index of e_i, t_i. Thus e_j contains the edge e_j. Thus, for any two edges, one must contain the other, and T_i is a stack.

Since each queue can then be decomposed into two staircases, the number of subgraphs $\sigma(G)$ is at most $O(\sqrt{n})$. The overall running time is $O(\sigma n \log n)$. ■

It is natural to ask whether these bounds on decompositions are tight, since the same integer sequence can have many graph realizations. Also, the decreasing subsequence, although a stack graph, is not necessarily the largest stack subgraph, i.e., there could exist a large stack as a subgraph of G, but the decreasing sequence is unable to compute it. For example, Figure 7a shows a stack which has only a small decreasing integer subsequence. However, there are degree-1 ordered graphs such that any decomposition produces at least $\Omega(\sqrt{n})$ stacks and queues, as illustrated in Figure 7b. This shows that our decomposition algorithm is optimal in worst case. Furthermore, it implies that if one wants to have an algorithm approximating the contact map overlap better than $O(\sqrt{n})$, one should use some other approaches than decomposing one graph into stacks and queues.

3.2. Approximation factor σ

Recall the problem with the stack decomposition of G_2, as shown in Figure 7a, where the graph could actually be a stack, yet the decreasing sequence is unable to identify it. This is because although every decreasing sequence is a stack, every stack does not have a corresponding decreasing sequence. One can find

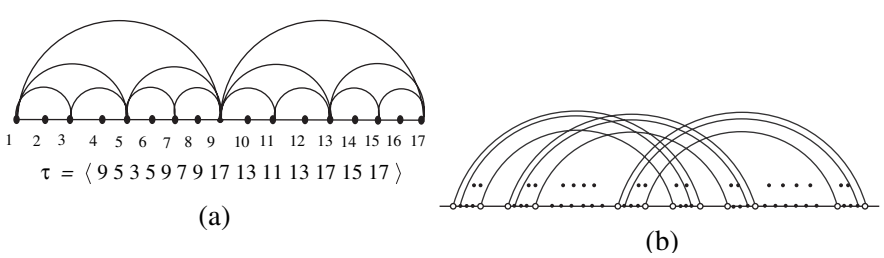

FIG. 7. (a) A stack with its corresponding sequence τ (b) Worst case example for our decomposition.

the largest stack in a graph using dynamic programming. However, one would like to find a stack that uses the maximal number of edges that are "unusable" for queues, so that in the next iteration a large queue could be found.

Local improvement heuristic. We now give a recursive heuristic for decomposition of the graph G into a set of graphs $T_1, \ldots, T_k$, where each T_i is either a queue, or a stack. The algorithm explained in the previous section and this heuristic are called `Decomp` and `GreedyDecomp`, respectively.

Given a graph $G = (V, E)$, define an edge $e \in E$ as a *top edge* if and only if there does not exist any edge $e' \in E$ that contains e. This defines an partial ordering of the edges. Define the weight of each top edge e' as $w(e') = |\{e \in E \; s.t. \; e \subseteq e'\}|$. The algorithm first finds the set E' consisting of all the top edges of E (typically, $|E'| \leq 10$). Then find the maximum weigh non-overlapping set of edges and for every edge $e' = (v_i, v_j)$ chosen, recursively apply the same procedure on the subgraph induced by vertices $\langle v_i, \ldots, v_j \rangle$ to find the largest stack contained inside e'. Note that by definition, no two edges in E' contain one another, and hence define a queue. There is a trade-off involved. If the number of top edges is large, then since they form a queue, we can always take them as a graph T_i, and apply `GreedyDecomp` on the remaining graph.

Empirical value of σ. We now give results on the decomposition of graphs by algorithm `Decomp` and `GreedyDecomp`. We test our decompositions on 20 proteins of size 300–1100 residues obtained from the Protein Data Bank. In almost all cases, the number of decomposed graphs is less than ten. The testing protein backbones are projected onto the *face centered cubic* (FCC) lattice, where each vertex has forty two potential contact-overlap neighbors. The two algorithms are used to decompose them into a set of queues and stacks.

Table 1 shows the input proteins, with their PDB file ID, the total number of residues, and the number of contacts. For each protein, we give the decompositions by both algorithms: for each algorithm, we give the number of stacks, and queues produced. Note that each queue has to be further decomposed into two staircases in order to be matched exactly, so we upper-bound the approximation ratio by the pigeon-hole principle: $\sigma = (\#\text{stack} + 2 * \#\text{queue})$.

4. HARDNESS OF CONTACT-MAP OVERLAP

In this section, we present the hardness results of both $\nu(G_1, G_2)$ and $\mu(G_1, G_2)$, indicating that a much better algorithm is not very likely. First, we present an embedding result that yields the hardness of $\nu(G_1, G_2)$. Then we augment it to show the hardness of $\mu(G_1, G_2)$.

4.1. Hardness for $\nu(G_1, G_2)$

The constant-factor approximation algorithm to compute both $\nu(G_1, G_2)$ and $\mu(G_1, G_2)$ for two-dimensional contact maps relies on the fact that G_1 arises from a self-avoiding walk on a two-dimensional

TABLE 1. COMPARISON OF THE APPROXIMATION RATIOS σ BETWEEN DECOMP AND GREEDYDECOMP

PDB	*#Res*	*#Cnts*	`Decomp`			`GreedyDecomp`		
			#Stk	*#Que.*	σ	*#Stk*	*#Que.*	σ
d1ddqc	1112	893	4	9	22	10	0	10
d1qgka	876	812	0	3	6	5	0	5
d2btva	845	652	3	5	13	8	0	8
d2prja	845	727	1	7	15	8	1	10
d5gpb	833	742	2	6	14	7	1	9
d1nok	831	708	2	6	14	5	2	9
d1c50a	830	717	3	6	15	8	1	10
d2skea	830	704	1	7	15	9	0	9
d1axr	829	731	2	6	14	7	1	9
d2amva	829	722	1	7	15	7	1	9

lattice, and thus can be decomposed into two stacks and one queue. In $\mathbb{R}^3$, we show that a self-avoiding walk hardly provides any feature to make either $\nu(G_1, G_2)$ or $\mu(G_1, G_2)$ easier. In particular, given a degree-1 ordered graph $G = (V, E)$, we first describe how to embed it as a three-dimensional self-avoiding walk P, with possibly adding some extra vertices, but without introducing any new edges (i.e., contacts).

Lemma 4.1. *Any degree-1 ordered graph $G = (V, E)$ can be embedded as a self-avoiding walk P on a 3-dimensional lattice such that E is the set of contacts of P.*

Proof. Given any degree-1 ordered graph $G = (V, E)$, where $V = \langle v_1, \ldots, v_n \rangle$ and $E = \{e_1, \ldots, e_m\}$, let $p_i = (x_i, y_i, z_i)$ denote the corresponding point of vertex v_i in the self-avoiding walk, and Π_i be the grid plane $z = i$. We position the points corresponding to V in the plane Π_0 such that (i) for all $e_k = (v_i, v_j) \in E$, $p_i = (0, 4k, 0)$ and $p_j = (1, 4k, 0)$; and (ii) for the remaining vertices of V with zero degree, we align them on the line (in the plane Π_0) $x = 0, y \geq 4m$, with a separation of four between the y-coordinates of any two consecutive vertices. (Fig. 8).

Obviously, the contacts formed on Π_0 correspond exactly to the edges in E. It remains to connect points p_1 through p_n into a self-avoiding walk P so that no new contacts are formed. This is done as follows. For any $p_i = (x_i, y_i, z_i)$, we maintain the invariant that P always enters p_i along the x-direction, and leaves along the y-direction. More precisely, if $x_i = 0$ (resp. $x_i = 1$), $\bar{p}_i$, the point immediately before p_i along P is $(-1, y_i, 0)$ (resp. $(2, y_i, 0)$), while $\hat{p}_i$, the point after p_i along P is $(0, y_i - 1, 0)$ (resp. $(1, y_i + 1, 0)$). (Fig. 9a).

To connect p_i to p_{i+1}, we go from $\hat{p}_i$ straight up (along the positive z-direction) to the point $(\hat{x}_i, \hat{y}_i, i * 2)$ on plane Π_{i*2}, continue along a path γ_i in Π_{i*2} to point $(\bar{x}_{i+1}, \bar{y}_{i+1}, i * 2)$, and then come down to point

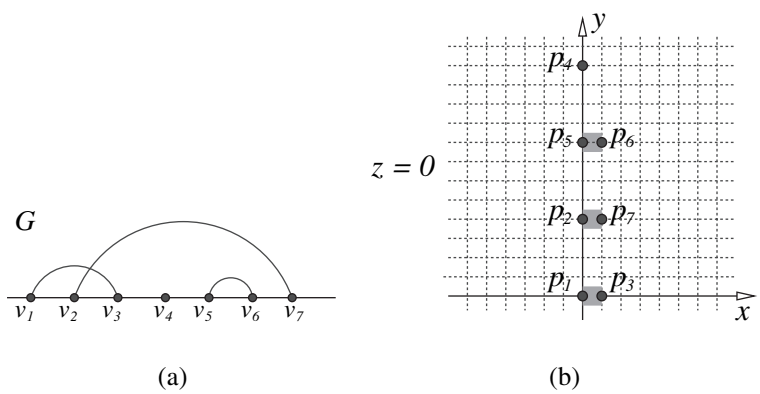

FIG . 8. Position the vertices of contact map G in (a) onto the plane Π_0: $z = 0$, as depicted in (b), where the contacts correspond to edges in G.

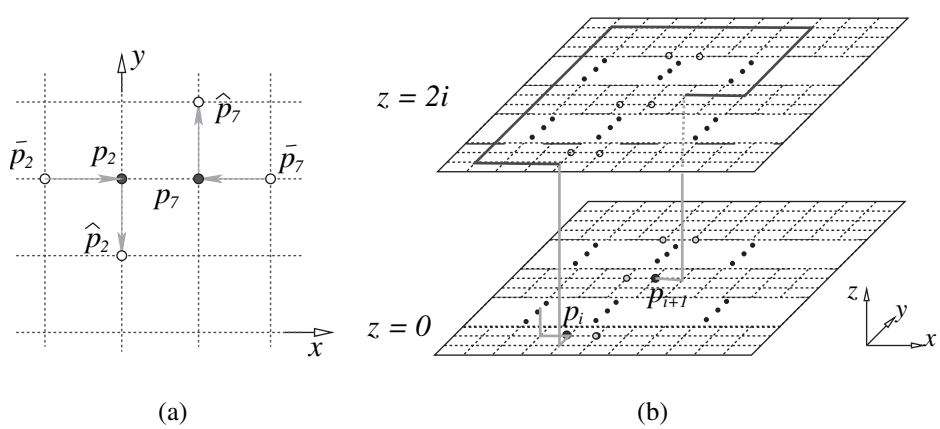

FIG . 9. (a) Depicts how P enters and leaves a vertex, while (b) illustrates how P connects p_i to p_{i+1} (solid points). γ_i consists of the bold segments on plane Π_{2i}.

$\bar{p}_{i+1}$ along -z direction. Note that since P might penetrate plane Π_{i*2} as it enters or leaves any p_j, for $j > i$, γ_i has to be chosen in such a way that it is at least two unit distance away in both x and y direction from any penetrating point. Easy to show that this condition can be easily satisfied, as depicted in Figure 9b. Since there is a separation of two between any γ_i's, no new contacts along z-direction can be formed as well.

Let $G' = (V', E')$ be the contact map of P as constructed above. We have $|V'| = O(n^2)$, $|E'| = |E|$, and there is a one-to-one map between edges in E and the contacts of P. ■

Since $\nu(G_1, G_2)$ only considers contacts of the self-avoiding walks, Lemma 4.1 implies that the edges of any degree-1 ordered graph can be the contacts arisen from some self-avoiding walk. Theorem 1 in Goldman et al. (1999) states that $\nu(G_1, G_2)$ is MAXSNP-hard if both inputs are arbitrary degree-1 ordered graphs. Thus, we have the following theorem.

Theorem 4.2. *Computing $\nu(G_1, G_2)$ is MAXSNP-hard even if both input graphs G_1 and G_2 are derived from three-dimensional self-avoiding walks.*

4.2. Hardness of $\mu(G_1, G_2)$

The construction in the last section does not prove that the contact map obtained from a three-dimensional self-avoiding walk can be an arbitrary degree-1 ordered graph, if degree-0 vertices are considered. Recall that in the complete contact map overlap problem, all vertices in G_1, not only the edges, have to be matched as well. Nevertheless, we show in this section that 1-$\mu(G_1, G_2)$ is at least as hard to approximate as 1-$\nu(G_1, G_2)$. Furthermore, we relate 1-$\mu(G_1, G_2)$ to a well-studied NP-hard problem called *dense-k subgraph* (also referred to as *maximum edge subgraph*) problem, which provides more indication of the hardness of the complete contact-map overlap problem.

A degree-1 ordered graph $G = (V, E)$ is considered to satisfy `Cond-A` if (i) at least $4m$ degree-0 vertices separate any two consequent degree-1 vertices p and q, where $m = |E|$; and (ii) there are even number of degree-0 vertices if p and q are both left (or both right) endpoints of some edges; otherwise, there are odd number of such vertices. It is easy to verify that by constructing γ_i's more carefully, any degree-1 ordered graph G under `Cond-A` has a corresponding self-avoiding walk P such that G, including both its vertex set and edge set, is equivalent to the contact map of P.

Theorem 4.3. *1-$\mu(G_1, G_2)$ is MAXSNP-hard even if both input graphs G_1 and G_2 are derived from three-dimensional self-avoiding walks.*

Proof. We reduce an instance of 1-$\nu(G_1, G_2)$ to an instance of 1-$\mu(G_1', G_2')$ whose inputs satisfying `Cond-A` as follows. Given any two contact maps G_1 and G_2, w.l.o.g. assume that there are no degree-0 vertices in them. Let n and m be the number of degree-1 vertices and the number of edges of G_1 respectively. First, construct G_1' by adding $4m$ or $4m + 1$ dummy vertices of degree 0 between any two consecutive degree-1 vertices of G_1, so that `Cond-A` is satisfied. Since any vertex in G_1' has to be mapped to some vertex in G_2' in order, we obtain G_2' by adding dummy vertices to G_2 so that there are $4n^2$ or $4n^2+1$ vertices between any two consecutive degree-1 vertices of G_2. Note that $\nu(G_1, G_2) = \mu(G_1', G_2')$, and any approximation of $\mu(G_1', G_2')$ corresponds to an approximation of $\nu(G_1, G_2)$ within the same approximation factor. As both G_1' and G_2' satisfy `Cond-A`, there exist three-dimensional self-avoiding walks whose contact maps are exactly G_1' and G_2'. This completes the proof of the lemma. ■

We now introduce a well-studied NP-hard problem defined as follows. Given an arbitrary graph $G = (V, E)$, the *dense-k subgraph* problem returns a subset $V' \subseteq V$ such that $|V'| = k$, and the number of edges induced by V' are maximized. Let $\mathrm{DS}_k(G)$ denote this maximum number of edges.

Dense-k subgraph has been the subject of study for over a decade. There are several approximation algorithms for the problem. Kortsarz and Peleg (1993) devised an approximation algorithm which has performance guarantee $O(n^{0.3885})$. This was later improved by Feige et al. (2000) to $O(n^{1/3-\varepsilon})$, for some $\varepsilon > 0$. However, still no algorithm with approximation ratio better than $O(n^{\varepsilon})$ exists.

Theorem 4.4. *If 1-$\mu(G_1, G_2)$ between G_1 and G_2 can be approximated within a factor of $f(\max(|E_1|, |E_2|))$, then $\mathrm{DS}_k(G)$ can be approximated within a factor of $f(|V|^2)$, where $f(x) : \mathbb{R} \to \mathbb{R}$ is*

some function, $G = (V, E)$ is an arbitrary graph, and $G_i = (V_i, E_i)$, $i = 1, 2$, are some three-dimensional contact maps with maximum degree 1.

Proof. Our reduction is similar to Akutsu and Miyano (1999). We reduce an instance of the dense-k subgraph problem to an instance of 1-$\mu(G_1, G_2)$. Given an arbitrary graph $G = (V, E)$, let $n = |V|$. Construct from G a degree-1 ordered graph G_1 that satisfies `Cond-A` as follows (Figure 10).

For any $v_i \in V$, generate n ordered vertices $v_{i_1}, \ldots, v_{i_n}$. Create an edge between (v_{i_j}, v_{j_i}) if there is an edge (v_i, v_j). Next, add $4n^2$ or $4n^2 + 1$ degree-0 vertices between any two v_{i_j} and $v_{i_{j+1}}$, $\forall i \in [1, n]$, $j \in [1, n-1]$, and add $4n^4$ (or $4n^4 + 1$) dummy vertices between v_{i_n} and $v_{(i+1)_1}$, for $\forall i \in [1, n-1]$. The resulting graph $G_1 = (V_1, E_1)$: (i) is a degree-1 ordered graph, with $|E_1| = |E|$ and $|V_1| = O(n^5)$; and (ii) satisfies `Cond-A`.

Finding, $\mathrm{DS}_k(G)$ can be interpreted as finding the maximum subgraph between G and K_k. We construct $G_2 = (V_2, E_2)$ from K_k, using similar techniques as for G_1. So for each vertex $k_i \in K_k$, we generate k ordered vertices $k_{i_1}, \ldots, k_{i_k}$ for V_2. The only differences is that we now add $4n^3$ (or $4n^3 + 1$) degree-0 vertices between k_{i_j} and $k_{i_{j+1}}$, while adding $4n^5$ (or $4n^5 + 1$) dummy vertices between k_{i_k} and $k_{(i+1)_1}$, for $i \in [1, k]$, and $j \in [1, k-1]$. G_2 satisfies `Cond-A`, with $|V_2| = O(n^6)$ and $|E_2| = O(k^2)$.

We sketch now briefly the proof of $\mathrm{DS}_k(G) = \mu(G_1, G_2)$ in this extended abstract.

1. $\mathrm{DS}_k(G) \leq \mu(G_1, G_2)$: Let (V', E') be the solution for the dense-k problem for a graph $G = (V, E)$. W.l.o.g., assume $V' = \{v'_1, v'_2, \ldots, v'_k\}$ such that $l_1^* \geq l_2^*$ for $l_1 \geq l_2$, where l^* is a function of l for $l \in [1, k]$, and $l^* = i$ if $v'_l = v_i$. For any $e = (v'_l, v'_j) \in E'$, match k_{l_j} and k_{j_l} in G_2 to $v_{l^*_{j^*}}$ and $v_{j^*_{l^*}}$ in G_1 respectively. One can verify that the matching thus constructed is order-preserving, and the $4n^3$ and $4n^5$ separation of G_2 guarantees that the remaining vertices in G_1 can be matched (without any inconsistencies) to some vertices in G_2.
2. $\mu(G_1, G_2) \leq \mathrm{DS}_k(G)$: Given the mapping of G_1 to G_2 that realizes $\mu(G_1, G_2)$, the $4n^4$ vertex separation in G_1 guarantees that any two v_{i_j} and $v_{i'_{j'}}$ cannot be matched to two vertices $k_{l_{j_1}}$ and $k_{l_{j_2}}$ in G_2. This corresponds to the requirement that any two vertices v_i and $v_{i'}$ in G will not be matched to the same vertex in K_k. Also, it can be shown that the case where two vertices from the same group in G_1 are matched to vertices from two different groups in G_2 can be avoided by modifying the matching without changing $\mu(G_1, G_2)$. This corresponds to the requirement that any vertex in G cannot be matched to two different vertices in K_k. As a result, any match between G_1 and G_2 can be mapped back to a subgraph in G involving at most k vertices.

Since $\max(|E_1|, |E_2|) = O(n^2) = O(|V|^2)$, the theorem follows. ■

As mentioned before, dense-k subgraph problem does not have better than $O(n^{1/3})$ approximation ratio algorithms, despite being studied heavily. Theorem 4.2 shows that complete contact-map overlap problem is at least as hard as this problem, suggesting that an approximation ratio better than $O(n^{\varepsilon}), \epsilon > 0$, might be hard.

5. CONCLUSIONS

In this paper, we give faster algorithms for contact-map overlap variants for two dimensions, and provided first approximation algorithm for three dimensions. We also showed that the contact-map of proteins in three

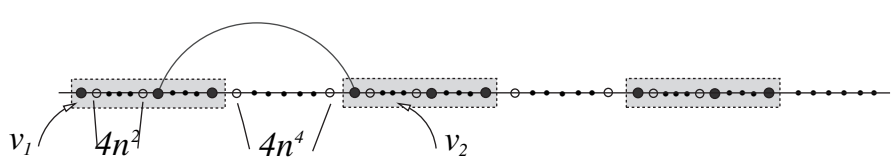

FIG. 10. Convert an arbitrary graph G into a degree-1 ordered graph. Each vertex v_i in G maps to all the points in each rectangle box. The empty dots refer to degree-0 vertices. The bold edge (v_{1_2}, v_{2_1}) in the figure corresponds to an edge (v_1, v_2) in G.

dimensions do not have any special constraining property that might help us get a better bound, and provided experimental results on the decompositions produced by our algorithms.

The problem of hardness of the contact-map overlap still remains open, due to the gap between the theoretical hardness and the bound of the algorithm. It would be interesting to combine the branch-and-bound approach of Lancia et al. (2001) with our decomposition to compute efficiently the contact-map overlap of large proteins.

ACKNOWLEDGMENTS

We thank Sergei Bespamyatnikh and Suresh Venkatasubramanian for helpful discussions and Patrice Koehl for providing the test data. Research by all authors was supported by NSF grants ITR-333-1050, EIA-98-70724, EIA-01-31905, CCR-97-32787, and CCR-00-86013.

REFERENCES

Akutsu, T., and Miyano, S. 1999. On the approximation of protein threading. *Theoret. Comput. Sci. 210*, 261–275.

Caprara, A., and Lancia, G. 2002. Structural alignment of large-size proteins via lagrangian relaxation. *RECOMB.*

Erdös, P., and Szekeres, G. 1935. A combinatorial problem in geometry. *Composit. Math. 2*, 463–470.

Feige, U., Kortsarz, G., and Peleg, D. 2000. The dense k-subgraph problem. *J. Algorithms*.

Francesco, D., Garnier, V., and Munson, P. 1996. Improving protein secondary structure prediction with aligned homologous sequences. *Protein Sci. 5*, 106–113.

Garey, M., and Johnson, D. 1979.*Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Gerstein, M., and Levitt, M. 1998. Comprehensive assessment of automatic structural alignment against a manual standard. *Protein Sci. 7*, 445–456.

Godzik, A. 1996. The structural alignment between two proteins: is there a unique answer? *Protein Sci. 5*, 377–385.

Goldman, D., Istrail, S., and Papadimitriou, C.H. 1999. Algorithmic aspects of protein structure similarity. *IEEE Symp. Found. Comput. Sci.* 512–522.

Halperin, I., Ma, B., Wolfson, H., et al. 2002. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins Struct. Funct. Genet. 47*, 409–443.

Holm, L., and Sander, C. 1993. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol. 233*, 123–138.

Holm, L., and Sander, C. 1996. The FSSP database: fold classification based on structure-structure alignment of proteins. *Nucleic Acids Res. 24*, 206–209.

Klein, P. 1998. Computing the edit-distance between unrooted ordered trees. *Proc. 6th Annu. Eur. Symp.*, 91–102.

Knuth, D.E. 1973. *The Art of Computer Programming, Volume 1, Fundamental Algorithms*. Addison-Wesley, Reading, MA.

Kolinski, A., and Skolnick, J. 1994. Monte Carlo simulations of protein folding: lattice model and interaction scheme. *Proteins* 18, 338–352.

Kortsarz, G., and Peleg, D. 1993. On choosing a dense subgraph. *Proc. 34th Symp. Found. Comput. Sci.* 692–703.

Lancia, G., Carr, R., Walenz, B., et al. 2001. 101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. *RECOMB* 193–202.

Leach, A.R. 1996. *Molecular Modelling: Principles and Applications*. Pearson Education Ltd., London.

Maiorov, V., and Crippen, G. 1994. Significance of root-mean-square deviation in comparing three-dimensional structures of globular proteins. *J. Mol. Biol. 235*, 625–634.

Taylor, W., and Orengo, C. 1989. Protein structure alignment. *J. Mol. Biol. 208*, 1–22.

Taylor, W., and Orengo, C. 1996. SSAP: sequential structure alignment program for protein structure comparison. *Methods Enzymol. 266*, 617–635.

Verbitsky, O. 1994. On the largest common subgraph problem. Personal communication.

Address reprint requests to:
Dr. Pankaj K. Agarwal
Department of Computer Science
Duke University
Durham, NC

E-mail: pankaj@cs.duke.edu