Title
Finding Nonoverlapping Dense Blocks of a Sparse Matrix

Authors
Vassilevska, Virginia
Pinar, Ali

Publication Date
2004-02-06

Peer reviewed

Finding Nonoverlapping Dense Blocks of a Sparse Matrix*

Virginia Vassilevska
Computer Science Department
Carnegie Mellon University
virgi@cs.cmu.edu

Ali Pınar
Computational Research Division
Lawrence Berkeley National Laboratory
apinar@lbl.gov

Abstract

Many applications of scientific computing rely on computations on sparse matrices. The design of efficient implementations of sparse matrix kernels is crucial for the overall efficiency of these applications. Due to the high compute-to-memory ratio and irregular memory access patterns, the performance of sparse matrix kernels is often far away from the peak performance on a modern processor. Alternative data structures have been proposed, which split the original matrix A into A_d and A_s, so that A_d contains all dense blocks of a specified size in the matrix, and A_s contains the remaining entries. This enables the use of dense matrix kernels on the entries of A_d producing better memory performance. In this work, we study the problem of finding a maximum number of nonoverlapping dense blocks in a sparse matrix, which is previously not studied in the sparse matrix community. We show that the maximum nonoverlapping dense blocks problem is NP-complete by using a reduction from the maximum independent set problem on cubic planar graphs. We also propose a 2/3-approximation algorithm that runs in linear time in the number of nonzeros in the matrix. This extended abstract focuses on our results for 2×2 dense blocks. However we show that our results can be generalized to arbitrary sized dense blocks, and many other oriented substructures, which can be exploited to improve the memory performance of sparse matrix operations.

Keywords: memory performance, memory-efficient data structures, high-performance computing, sparse matrices, independent sets, NP-completeness, approximation algorithms.

This paper is submitted as a regular paper.

*This work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract DE-AC03-76SF00098.

1 Introduction

Sparse matrices lie in the hearts of many computation-intensive applications such as finite-element simulations, decision support systems in management science, power systems analysis, circuit simulations, and information retrieval. The performance of these applications directly relies on the performance of the sparse matrix kernels. However, the performance of sparse matrix operations on modern processors is limited due to the high compute-to-memory ratio, and irregular memory-access patterns.

Conventional data structures for sparse matrices have two components: an array that stores floating-point entries of the matrix, and arrays that store the nonzero structure (i.e., pointers to the locations of the numerical entries). Exploiting sparsity invariably requires using pointers, but pointers often lead to poor performance. One reason for the poor memory performance is that pointers cause an irregular memory access pattern and thus poor spatial locality. Another important reason, which is often overlooked, is the extra load operations. Each operation on a nonzero entry requires loading the location of that nonzero before loading the actual floating point number. For instance, sparse matrix vector multiplication, which is one of the most important kernels in numerical algorithms, requires three load operations for each nonzero in the matrix. It has been observed that this overhead might be as costly as the floating point operations [5].

Recent studies have investigated improving memory performance of sparse matrix operations by reducing the number of extra load operations [5, 8, 9]. In [8], Pınar and Heath proposed exploiting dense blocks of a sparse matrix, along with reordering techniques to increase the sizes of these blocks. One approach considered is splitting a matrix as $A = A_{12} + A_{11}$, where A_{12} includes 1×2 blocks of the matrix (two nonzeros in consecutive positions on the same row), and A_{11} covers the remaining nonzeros. Notice that it is sufficient to store a pointer for each block in A_{12}. Significant speedups in large experimental sets have been observed, which gives motivation to search for larger blocks in the matrix for further improvements in performance. One can split the matrix into $A = A_d + A_s$, where A_d contains all dense blocks, and A_s contains the remaining entries. Clearly, for a constant block size, having more entries in A_d yields fewer load operations, thus better memory performance. This calls for efficient algorithms to find a maximum number of nonoverlapping blocks of a specified size in a sparse matrix. A greedy algorithm is sufficient to find a maximum number of nonoverlapping $m \times n$ blocks when $m = 1$ or $n = 1$. However, this problem is much harder when $m, n \geq 2$.

In this work, we study the problem of finding a maximum number of nonoverlapping $m \times n$ dense blocks of a sparse matrix, which we call the *maximum nonoverlapping dense blocks* problem. In the next section, we define the problem formally and investigate its relation to the maximum independent set problem. We define a class of graphs where the independent set problem is equivalent to the maximum nonoverlapping dense blocks problem. In Section 3, we use this relation to prove that the maximum nonoverlapping dense block problem is NP-complete. Our proof uses a reduction from the maximum independent set problem on cubic planar graphs and adopts orthogonal drawings of planar graphs. Section 4 presents an approximation algorithm for the problem. Since we are motivated by improving memory performance of sparse matrix operations, we are interested in fast and effective heuristics for the preprocessing cost to be amortized over the speedups in subsequent sparse matrix operations. Our algorithms require only linear time and space, and generate solutions whose sizes are within 2/3 of the optimal.

In this extended abstract, we only focus on finding 2×2 blocks due to space considerations and clarity of presentation. However, our results can be generalized as we discuss in Section 5. We prove that the problem is NP-complete for $m \times n$ blocks for $m, n \geq 2$. We also work on alternative dense patterns to replace rectangular blocks, which might be employed to speedup sparse matrix operations.

The problem of finding nonoverlapping dense blocks of a sparse matrix has not been studied in the sparse-matrix community. We have been recently aware of the work by Berman *et al.* [2], where a similar problem is discussed as the optimal tile salvage problem. In the optimal tile salvage problem, we are given an $\sqrt{N} \times \sqrt{N}$ region of the plane tiled with unit squares, some of which have been removed. The task is to find a maximum number of functional nonoverlapping $m \times n$ tiled rectangles. The difference between our problem and the optimal tile salvage problem is that in the tile salvage problem the tiles are allowed to be in any orientation ($m \times n$ or $n \times m$), whereas in our case the orientation is fixed (only $m \times n$). The two problems coincide in the case of square dense blocks. Berman *et al.* proved the NP-completeness of the tile salvage problem, however their proof exploits the flexibility in the orientation of the dense block, and thus our proof is significantly different. Berman *et al.* also describe an $(1 - \epsilon)$-approximation algorithm, which would work for square blocks, for $\epsilon = O(1/\sqrt{\delta \log M})$, where M is the optimal solution value. Their algorithm is based on maximum planar H-matching which runs in $O(N^{1+\delta})$ steps for small $\delta > 0$. Baker [1] also has an algorithm for the case of square blocks, which runs in $O(8^k N)$-time and $O(4^k N)$ space and produces a $(k-1)/k$-approximation. Both of these algorithms however are complex and hard to implement. The greedy 2/3-approximation algorithms we propose are very simple. It requires linear time and space, with very small constant factors in the time and space bounds. Our algorithm requires only one pass through the matrix, and thus is I/O -efficient.

2 Preliminaries

In this section we define the problems formally, and present definitions and some preliminary results that will be used in the following sections.

2.1 Problem Definition

This work investigates the problem of finding a maximum number of nonoverlapping matrix substructures of prescribed form and orientation.

Definition 2.1 *An $m \times n$* pattern *is a 0-1 $m \times n$ matrix σ. An* oriented σ-substructure *of a matrix A is an $m \times n$ submatrix M in A so that $M(i,j) \neq 0$ if $sigma(i,j) = 1$ for $1 \leq i \leq m$, and $1 \leq j \leq n$. Two substructures M and N overlap if they share nonzero entry e in M with coordinates (i_M, j_M) in M and (i_N, j_N) in N and $\sigma(i_M, j_M) = \sigma(i_N, j_N) = 1$.*

Given a particular pattern σ, we define the *maximum nonoverlapping σ-substructures* (MNS) problem as follows.

Definition 2.2 ***Maximum Nonoverlapping σ-Substructures (MNS) Problem***
INSTANCE: An $M \times N$ matrix A, integer K.
QUESTION: Does A contain K disjoint σ-substructures?

In this paper, we focus on *dense blocks*, due to their simplicity, and their effectiveness in speeding up sparse matrix operations. A *dense block* of a matrix is a submatrix of specified size all of whose entries are nonzero, i.e., it is a σ-substructure where σ is the all 1s matrix. We identify a dense block with its upper left corner. Two blocks overlap if they share a matrix entry. Formal definitions follow.

Definition 2.3 *Given an* $M \times N$ *matrix* $A = (a_{ij})$*, we say* b_{ij} *is an* $m \times n$ *dense block in* A *iff* $a_{kl} \neq 0$ *for all* k *and* l *such that* $i \leq k < i + m \leq M$ *and* $j \leq l < j + n \leq N$*. Two* $m \times n$ *blocks* b_{ij} *and* b_{kl} *overlap iff* $i \leq k < i + m$ *and* $j \leq l < j + n$*, or* $k \leq i < k + m$ *and* $l \leq j < l + n$*.*

We specify the MNS problem for dense blocks as follows.

Definition 2.4 ***Maximum Nonoverlapping Dense Blocks (MNDB) Problem***
INSTANCE: An $M \times N$ *matrix* A*, positive integers* m *and* n *that define the block size, and a positive integer* K*.*
QUESTION: Does A *contain* K *disjoint* $m \times n$ *dense blocks?*

In this paper, we will focus on 2×2 blocks for space considerations, and clarity of presentation, although our results can be generalized to varying block sizes, and different substructures.

2.2 Intersection Graphs

It is easy to find all specified patterns in a matrix, however what we need is a subset with nonoverlapping blocks. In this sense, the MNS problem is related to the *maximum independent set* (MIS) problem, which is defined as finding a maximum cardinality subset of vertices I of a graph G, such that no two vertices in I are adjacent. Below we define an *intersection graph*, which reveals the relation between the independent set and the nonoverlapping blocks problems more clearly.

Definition 2.5 *A graph* G *is an intersection graph of the* σ*-substructures of a matrix* A *if there is a bijection* ϕ *between the vertices of* G *and the substructures of* A*, such that there is an edge in* G *between* $\phi(s_1)$ *and* $\phi(s_2)$ *if and only if* s_1 *and* s_2 *overlap in* A*.*

We will use $G(A, m, n)$ to refer to the intersection graph of dense $m \times n$ blocks in matrix A. A maximum independent set on $G(A, m, n)$ gives a maximum number of nonoverlapping blocks in A, thus the MNDB problem can be reduced to the maximum independent set problem, which is known to be NP-complete [4]. However it is important to note that the block intersection graphs have special structures, which can be exploited for efficient solutions. For instance, a greedy algorithm is sufficient to find a maximum number of nonoverlapping $1 \times n$ and $m \times 1$ blocks, since these problems reduce to a family of disjoint maximum independent set problems on interval graphs. In the remainder of this section, we define the class of graphs that constitute block intersection graphs. An intersection graph of a set of 2×2 dense blocks is an induced subgraph of the so called X-grid which consists of the usual 2 dimensional grid, and diagonals for each grid square. In general, the intersection graph of a set of $m \times n$ dense blocks is an induced subgraph of the X_{mn} grid. Below, we first define an X_{mn} grid, and then restrict the definition to define the graph class $X\Gamma_{mn}$ that represent graphs that can be an intersection graph for a matrix.

Definition 2.6 *An $M \times N$ X_{mn} grid is a graph with a vertex set V and an edge set E, so that*

- $V = \{v_{ij} : 1 \leq i \leq M - m; 1 \leq j \leq N - n\}$
- $E = \{(v_{ij}, v_{kl}) : 1 \leq i, k \leq M - m; 1 \leq j, l \leq N - n : |i - k| < m; |j - l| < n\}$

In an X_{mn} grid, vertex v_{ij} corresponds to the block b_{ij} in the matrix, and edges correspond to all possible overlaps between blocks. Note that not all induced subgraphs of the X_{mn} grid are intersection graphs of a matrix. We define a graph class $X\Gamma_{mn}$ in which each graph corresponds to an intersection graph $G(A, m, n)$ of the set of $m \times n$ dense blocks of a matrix A, and each such intersection graph is in the class.

Definition 2.7 *A graph $G = (V, E)$ is in the graph class $X\Gamma_{mn}$ if and only if it is an induced subgraph of an X_{mn} grid and has the closure property so that $v_{ij} \in V$ if*

$$\forall i \leq k < i + m, j \leq l < j + n; \quad \exists v_{st} : s \leq k < s + m \text{ and } t \leq l < t + n$$

The closure property enforces that there is a vertex in the graph for each block in the matrix. Being an induced subgraph of an X grid guarantees that there is an edge for each overlap. The graphs in this class are exactly the intersection graphs of the $m \times n$ blocks in a matrix, thus finding a maximum independent set of a graph in this class is equivalent to solving the MNDB problem of the corresponding dense matrix blocks. This claim is formalized by the following lemma.

Lemma 2.1 *An instance of the MNDB problem for finding K $m \times n$ nonoverlapping dense blocks in a matrix A is polynomially equivalent to an instance of MIS for a graph in $X\Gamma_{mn}$.*

Proof: As we discussed earlier, the MNDB problem can be reduced to the problem of finding an independent set on its intersection graph. Here we show the one-to-one correspondence between intersection graphs, and graphs in $X\Gamma_{mn}$. Remember that each dense block b_{ij} corresponds to the vertex v_{ij} in $G(A, m, n)$. By definition of the class $X\Gamma_{mn}$, $G(A, m, n) \in X\Gamma_{mn}$, thus any instance of an MNDB problem can be reduced to an independent set problem in a graph in $X\Gamma_{mn}$.

Given a graph G in $X\Gamma_{mn}$, define $A = (a_{ij})$, so that a_{ij} is a nonzero iff $k \leq i < k + m$ and $l \leq j < l + n$ for some vertex v_{kl} in G. Observe that any dense block in A must be represented by a vertex in G due to the closure property. Also, for any two adjacent vertices in G, corresponding blocks intersect in A, and no other blocks overlap, due to the definition of edges in X_{mn}. Thus, a maximum-cardinality subset of nonoverlapping blocks in matrix A corresponds to a maximum independent set in $G \in X\Gamma_{mn}$. ∎

In this paper we will use the graph class $X\Gamma_{22}$ to prove the NP-completeness of the MNDB problem for 2×2 blocks. Our proof can be generalized to arbitrary sized blocks, showing the NP-completeness of the MNDB problem for $m \times n$ blocks, and hence the NP-completeness of the maximum independent set problem for graphs in class $X\Gamma_{mn}$.

The following lemma shows that removing a subset of the vertices along with their neighbors preserves the characteristics of the graph, providing the basis for greedy approximation algorithms as will be presented in Section 4.

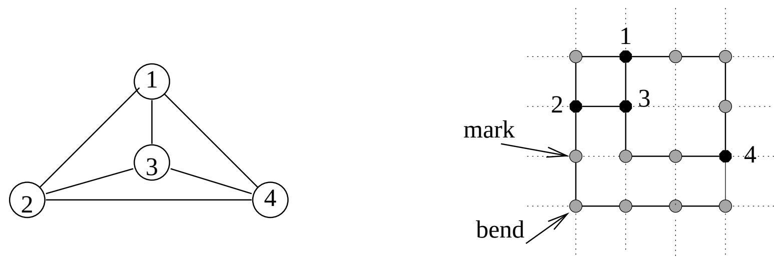

Figure 1: Planar Orthogonal Drawing

Lemma 2.2 *Let $G = (V, E)$ be a graph in $X\Gamma_{mn}$, $S \subseteq V$ a subset of vertices, and $N(S) = \{u \mid (u,v) \in E,\ v \in S,\ u \notin S\}$ be the neighborhood of S in G. Then the graph G' induced by $V \setminus (S \cup N(S))$ is still in $X\Gamma_{mn}$.*

Proof: Let $V' = V \setminus (S \cup N(S))$. The lemma would not hold only if for some i, j with $1 \le i \le M - m$ and $1 \le j \le N - n$ we have that for all k, l such that $i \le k < i + m$ and $j \le l < j + n$ there exist s, t with $s \le k < s + m$ and $t \le l < t + n$ and $v_{st} \in V'$, yet $v_{ij} \notin V'$. First note that $v_{ij} \notin S$. This is since if $k = i$ and $l = j$ then the corresponding v_{st} must be adjacent to v_{ij} since $|k - s| < m$ and $|l - t| < n$. Since $v_{st} \notin (S \cup N(S))$ in particular $v_{st} \notin N(S)$, and so $v_{ij} \in N(S)$. Hence there is a $v_{pq} \in S$ such that it is adjacent to v_{ij}, i.e. $|p - i| < m$ and $|q - j| < n$. Then consider $k = \frac{(p+i)+|p-i|}{2}$ and $l = \frac{(q+j)+|q-j|}{2}$. Clearly, $i \le k < i + m$, $j \le l < j + n$ and $p \le k < p + m$, $q \le l < q + n$. Consider the v_{st} corresponding to this choice for k and l. We have $s \le k < s + m$ and $t \le l < t + n$. Hence $|p - s| < m$ and $|q - t| < n$, and so $v_{st} \in N(S)$, which is a contradiction since we must have $v_{st} \in V'$. ∎

2.3 Planar Graphs and Orthogonal Drawings

A graph G is *planar* if and only if there exists an embedding of G on the sphere such that no two edges have a point in common besides the vertices. G is *cubic planar* if every vertex has degree 3.

An *orthogonal drawing* of a graph G is an embedding of G onto a 2-dimensional rectangular grid such that every vertex is mapped to a grid point and every edge is mapped to a continuous path of grid line segments connecting the end points of the edge. When G is planar, the edge paths do not cross. An example of orthogonal embedding of a planar graph is illustrated in Figure 1. As seen in this figure, we refer to a grid point where an edge path changes direction as a *bend*. No two edges share a grid segment or a bend, and no edge path can go through a vertex unless this vertex is an end point of the edge corresponding to the path and is an end point of the path itself. A *mark* in an orthogonal drawing of a graph is a grid point that an edge passes through,but not a vertex in the original graph. The following result has been reported by de Fraysseix *et al.* [3], Kant [6], and Papakostas and Tollis [7].

Theorem 2.3 *Every planar graph G with vertex degree at most 4 can be drawn orthogonally with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends on an $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$ grid in linear time.*

In particular, this shows that every cubic planar graph $G = (V, E)$ can be embedded orthogonally in an $O(|V|) \times O(|V|)$ grid in polynomial time. The NP-completeness proof in the next section uses a reduction from the maximum independent set (MIS) problem on cubic planar graphs, and adopts orthogonal drawings.

3 Complexity

This section proves that the MNDB problem is NP-complete for 2×2 blocks. We use a reduction from the independent set problem on cubic planar graphs, which we know is NP-complete [4]. Throughout this section, we let $X\Gamma$ denote $X\Gamma_{22}$. The next lemma explains how we can retain independent set characteristics of the problems after transformations.

Lemma 3.1 *Let $G = (V, E)$ be a graph, and u, v be two adjacent vertices in G, so that all neighbors of u besides v are also neighbors of v. Let $G' = (V', E')$ be the graph G after vertex v is removed. The size of the maximum independent set in G is equal to the size of the maximum independent set in G'.*

Proof: If vertex v is in a maximum independent set I, then none of its neighbors are in I. Thus $I' = I \cup \{v\} \setminus \{u\}$ is an independent set in G and in G' of the same size as I. ∎

Corollary 3.2 *Let $G \in X\Gamma$ contain the graph H in Figure 5(a) as an induced subgraph so that all vertices except for possibly v_1, v_2 and v_3 have all of their neighbors in H. Then any instance $(G,\ K)$ of MIS is equivalent to the instance $(G',\ K)$ of MIS for the graph $G' = G \setminus \{w_1, w_2\}$.*

Proof: By Lemma 3.1, we can remove w_1 from the graph since all neighbors of x_1 are neighbors of w_1 as well. The reduced graph is illustrated in Figure 5(b). Again using Lemma 3.1, we can remove w_2 since it covers all neighbors of x_2. Note that we can apply the same transformation to add vertices w_1 and w_2 to the graph in Figure 5(c). ∎

The following lemma describes how edges of a graph can be replaced by paths, while preserving independent set characteristics.

Lemma 3.3 *Let $G = (V, E)$ be a graph and $e = (v_i, v_j) \in E$ be an edge. Let $G_{e,k}$ be the graph G with the edge e substituted by a simple path $v_i, w_1, w_2, \ldots, w_{2k}, v_j$ where $k \in \mathbf{Z}^+$ and w_i are new vertices not in the original graph. Then there exists an independent set of size K in G if and only if there exists an independent set of size $K + k$ in $G_{e,k}$.*

Proof: We present the proof for $k = 1$, and the result follows by induction.
Sufficiency: Let I be an independent set in G, then either $v_i \notin I$ or $v_j \notin I$. Without loss generality, assume $v_i \notin I$, then $I' = I \cup \{w_1\}$ is an independent set in $G_{e,k}$.
Necessity: Let I' be an independent set in $G_{e,k}$. If $w_1 \in I'$, then $v_i \notin I'$, thus $I = I' \setminus \{w_1\}$ is an independent set in G. Symmetrically, if $w_2 \in I'$, then $v_j \notin I'$, thus $I = I' \setminus \{w_2\}$ is an independent set in G. If $w_1, w_2 \notin I'$, then $I = I' \setminus \{v_2\}$ is an independent set in G. ∎

Theorem 3.4 *Problem MNDB is NP-complete for 2×2 blocks.*

Proof: As discussed in the previous section, the problem of finding maximum number of nonoverlapping dense blocks in a sparse matrix can be reduced to the problem of finding a maximum independent set in the intersection graph of the matrix, and thus is in NP. For the

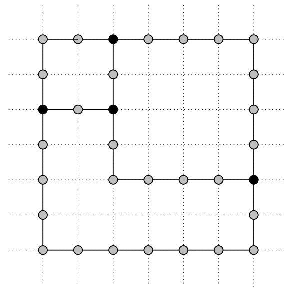

Figure 2: Enlargement operation for $K = 1$

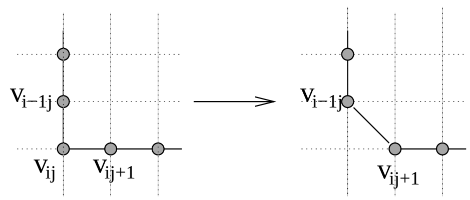

Figure 3: Bend transformation

NP-completeness proof we use reduction from the independent set problem on cubic planar graphs, which is NP-complete [4]. We first use Theorem 2.3 to embed a cubic planar graph onto a grid. Then we transform the embedded graph so that it is in $X\Gamma$. Our transformations preserve independent set characteristics so that an independent set in the transformed graph can be translated to an independent set in the original graph. Finally, we use Lemma 2.1 to relate the independent set problem on a graph in $X\Gamma$, to the MNDB problem, and conclude the MNDB problem is NP-complete.

Our transformations are local, so we first *enlarge* the grid to make room for these transformations. The enlargement operation inserts K new grid points between two grid points in the original. An example is illustrated in Figure 3 for $K = 1$. After the enlargement, each edge is now replaced by a path of K vertices (which we distinguish from the original vertices by calling them marks). Two adjacent vertices in the original graph are now at a distance $K + 1$, which generates a $K \times K$ area around each vertex for local transformations. In this proof, it is sufficient to use $K = 100$.

We can break down our transformations into 2 steps. The first step guarantees that the transformed graph is in $X\Gamma$. For this purpose, we need to have an edge between all pairs of vertices for which the corresponding blocks overlap so that the graph is in $X\Gamma$, and we need to insert vertices into the graph if necessary so that the closure property is satisfied. The second step makes sure that each edge in the original graph is replaced by an even-length path after the orthogonal embedding and transformations. Then we have successfully transformed the independent set problem on the cubic planar graph to an independent set problem on a graph in $X\Gamma$, and we can conclude the NP-completeness of the MNDB problem using the result of Lemma 2.1.

We need to consider two cases for the first step. One is a *bend* neighborhood as illustrated in Figure 3, and the other is a *T- junction.* As illustrated in Figure 4 a T-junction is just a neighborhood of a vertex in the original graph. Notice that the only remaining case is a path of

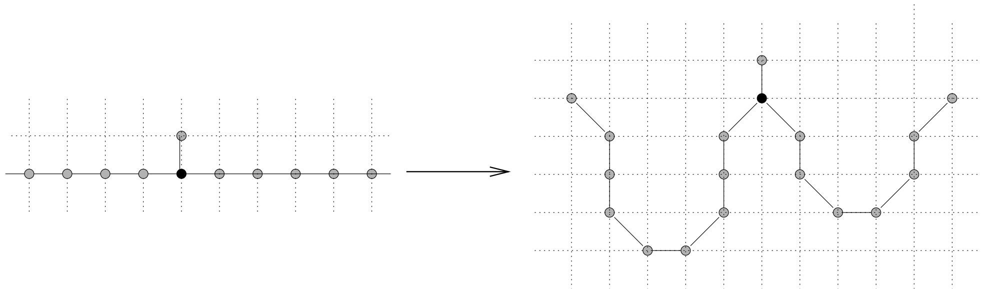

Figure 4: T-junction transformation

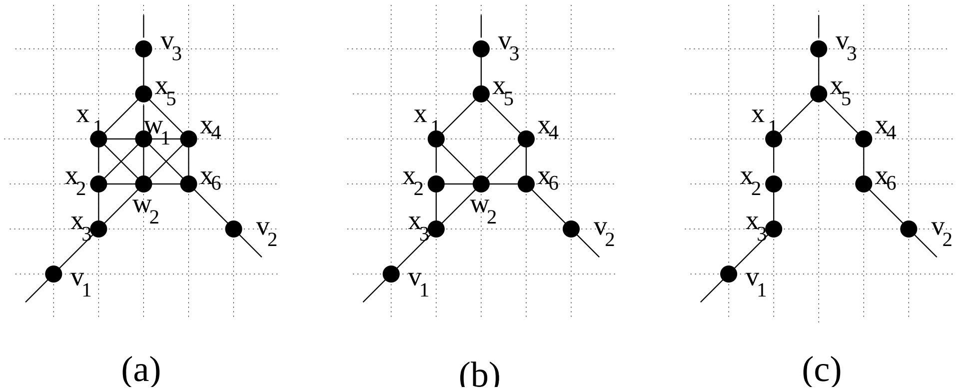

Figure 5: Transformation to preserve closure properties.

vertices, which does not cause any problems. Consider a bend v_{ij} connected to two other marks v_{i-1j} and v_{ij+1}. Note that v_{ij} cannot be a vertex in the original graph, since the original graph is cubic. In a graph in $X\Gamma$, there must be and an edge between v_{i-1j} and v_{ij+1}. We can remove v_{ij}, and connect v_{i-1j} and v_{ij+1} as in Figure 3.

Now consider a T-junction with vertex v_{ij} at the center, as illustrated in Figure 4. The neighborhood of v_{ij} is composed of (up to a rotation) v_{ij-1}, v_{ij+1}, and $v_{i-1\,j}$, none of which is a vertex in the original graph. As in the case of a bend, the problem here is the absence of edges between v_{ij-1} and $v_{i-1\,j}$, and between $v_{i-1\,j}$ and v_{ij+1}, for which the associated blocks will overlap. Also observe that v_{ij} must be a vertex of the original graph, and cannot be eliminated. We can make the transformation illustrated in Figure 4, yet the resulting graph is still not in $X\Gamma$, since it has missing vertices, and does not satisfy the closure property. We can use Corollary 3.2 to add vertices to the graph as depicted in Figure 5, so that the resulting graph is in $X\Gamma$.

By Lemma 3.3, we need each path replacing an edge of the planar graph to be of even length. For each edge going through an odd number of marks we know that there is a straight line segment going through at least 7 marks, due to the initial enlargement. We can replace this 7 vertex segment with an 8 vertex segment, to guarantee that the path representing an edge is of even length. This transformation is illustrated in Figure 6. After this step, we have a graph in $X\Gamma$ that replaces each edge in the original graph with an even length path.

Notice that all our transformations require polynomial time and space, thus the size of the final embedded graph is polynomial in the size of the original graph.

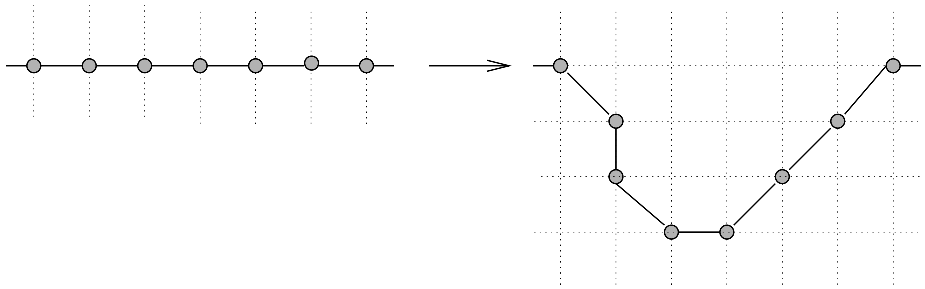

Figure 6: odd-length to even-length transformation to preserve independent set characteristics.

This reduces the independent set problem for cubic planar graphs to an independent set problem in a graph in class $X\Gamma$. By the result of Lemma 2.1, we know the independent set problem on a graph in $X\Gamma$ is equivalent to a MNDB problem in a matrix. Thus we reduced the independent set problem for cubic planar graphs to the MNDB problem, which concludes our proof. ∎

4 Approximation Algorithms

In this section, we present a 2/3-approximation algorithm for the MNDB problem for 2×2 blocks. Now that we know the problem is NP-complete, we have to resort to heuristics for a fast and effective solution. Remember that our motivation for investigating this problem is speeding up sparse matrix-vector multiplication. Our methods will be used in a preprocessing phase, thus they must be fast, for their cost to be amortized by the speedup in subsequent sparse matrix-vector multiplications.

Berman *et al.* [2], propose an approximation algorithm for square blocks, which uses the Lipton-Tarjan planar separator algorithm to get a $(1-\epsilon)$-approximation, where $\epsilon = O(1/\sqrt{\delta log M})$ in $O(n^{1+\delta})$ time, for any $\delta > 0$, where M is the size of an optimal solution. Baker [1] gives an $(k-1)/k$-approximation, which uses $O(8^k n)$ time and $O(4^k n)$ space.

Below we propose a greedy approach for the 2×2 case, which in the 1/2-approximation case is applicable to general $m \times n$ rectangular blocks. Unlike the two algorithms cited, due to its greedy nature it is simple and very easy to implement. It is pass-efficient, and takes time and space linear in the number of blocks of the matrix, with very small constant factors in the bounds.

First note that an easy 1/2-approximation to the MNDB problem with 2×2, which runs in linear time in the number of blocks, is achieved by finding the leftmost block in the topmost row, adding it to the current independent set, and then repeating the same operation after removing this vertex and all its neighbors. Note that at most two of the vertices can be independent among those removed from the graph, thus we have a 1/2-approximation algorithm. In this section we show how to improve this approximation result by looking at an extended neighborhood of the leftmost vertex in the uppermost row. Our algorithm is based on choosing a set of vertices in the neighborhood of the leftmost vertex in the uppermost row, so that the size of this set is no less than 2/3 of a maximum independent set in the induced subgraph of those vertices removed

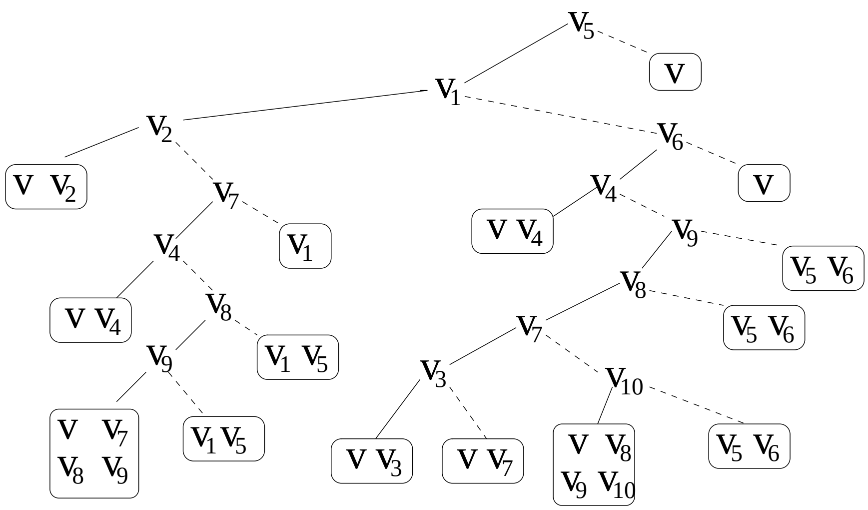

Figure 7: Decision tree for algorithm MNDB-APX. v corresponds to the leftmost vertex in the uppermost row, and the neighboring vertices in the X-grid are marked in Figure 8. We take the left branch if the label vertex is in V, and the right branch otherwise. We proceed until we reach a leaf, which contains the set S that will be added in the independent set.

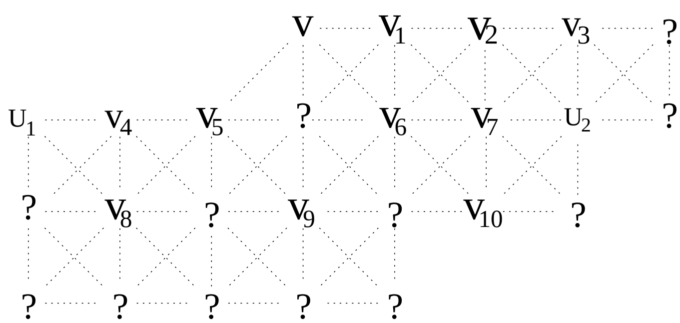

Figure 8: Vertex neighborhood considered for each call to BinTreeDecision. The positions v_i are used in the decision tree, while the positions u_i are only used in the analysis.

from the graph. Clearly this generates a final solution that is 2/3 of the optimal, since all greedy decisions are at least 2/3 of the local optimal. Note that the resulting graph after removing a vertex along with all its neighbors still has the characteristics of the original as proven in Lemma 2.2

Our decision process **BinTreeDecision** is depicted as a binary decision tree in Figure 7. In this tree, internal nodes indicate conditions, and the leaves list the vertices added to the independent set. We present the pseudocode of the algorithm below.

Algorithm MNDB-APX
$I \leftarrow \emptyset$
  while $V \neq \emptyset$
    $v \leftarrow$ leftmost vertex on the uppermost row
    $S \leftarrow$ **BinTreeDecision**(v)
    $I \leftarrow I \cup S$
    remove S and its neighborhood from G
  endwhile
return I

Lemma 4.1 *Algorithm MNDB-APX runs in linear time in the number of blocks in the matrix.*

Proof: Each iteration of the algorithm requires a traversal of the binary decision tree from the root to a leaf, which takes at most 8 steps, thus $O(1)$ time. Also at least one vertex is removed from the graph at each iteration. Thus the time for the decision process is linear in the number of vertices in the graph. The only other operation that affects the cost is finding the leftmost vertex in the uppermost row. In a preprocessing step one can go through the matrix in a left to right fashion and store pointers to the blocks so that v_{ij} appears before v_{kl} iff $i < k$ or $i = k$ and $j < l$. After this it takes constant time to find the current leftmost vertex on the uppermost row. ∎

Lemma 4.2 *The size of the maximal independent set returned by Algorithm MNDB-APX is no smaller than 2/3 of the size of maximum independent set on the intersection graph.*

Proof: The proof is based on case by case analysis. We show that **BinTreeDecision**(v) of Figure 7 always returns an independent set S such that $N(S)$ contains no independent set larger than $1.5\ |S|$, where $N(S)$ denotes the neighborhood of S, i.e., the set of vertices in S or adjacent to a vertex in S. Below we examine the binary search tree case by case:

$v_5 \notin V$ $S = \{v\}$, and v and its neighbors form a clique with MIS size 1.

$v_5 \in V$

$v_1 \notin V$ By the closure property $v_2 \notin V$, and we have the following:

$v_6 \notin V$ $S = \{v\}$, and v and its neighbors form a clique with MIS size 1.

$v_6 \in V$

$v_4 \in V$ $S = \{v, v_4\}$, and $N(S)$ has MIS size at most 3.

$v_4 \notin V$ By the closure property $u_1 \notin V$. In this case, if one of v_9 or v_8 is not in V, then $S = \{v_5, v_6\}$, since their neighborhood has MIS size at most 3. Otherwise, $v_8, v_9 \in V$:

$v_7 \notin V$ This implies $u_2 \notin V$ and:

$v_{10} \notin V$ $S = \{v_5, v_6\}$ and $N(S)$ has MIS size at most 3.

$v_{10} \in V$ $S = \{v, v_8, v_9, v_{10}\}$, and $N(S)$ has MIS size at most 6.

$v_7 \in V$

$v_3 \in V$ $S = \{v, v_3\}$, and $N(S)$ has MIS size at most 3.

$v_3 \notin V$ $S = \{v, v_7\}$, and $N(S)$ has MIS size at most 3.

$v_1 \in V$

$v_2 \in V$ $S = \{v, v_2\}$, and $N(S)$ has MIS size at most 3.

$v_2 \notin V$ By the closure property $v_3 \notin V$, and

$v_7 \notin V$ $S = \{v_1\}$, v_1 and its neighbors form a clique, and the MIS is of size 1.

$v_7 \in V$

$v_4 \in V$ $S = \{v, v_4\}$, and $N(S)$ has MIS size at most 3.

$v_4 \notin V$ By the closure property $u_1 \notin V$, and if one of v_8 or v_9 is not in V, then $S = \{v_1, v_5\}$, and $N(S)$ has a MIS size at most 3. Otherwise if $v_8, v_9 \in V$, then $S = \{v, v_7, v_8, v_9\}$, and $N(S)$ has MIS size at most 6. ∎

Theorem 4.3 *Algorithm MNDB-APX is a linear time,* 2/3*-approximation algorithm.*

Proof: Follows directly from Lemma 4.1 and Lemma 4.2. ∎

5 Extensions and Further Research

We have so far limited our discussions to finding 2×2 blocks in a matrix due to space considerations. However, our results can be generalized to larger blocks and alternative patterns, which can replace dense blocks to speedup sparse matrix operations.

Our NP-completeness proof for 2×2 blocks in Section 3 can be easily extended to arbitrary sized $m \times n$ blocks when $m, n \geq 2$.

Theorem 5.1 *Problem MNDB is NP-complete for* $m, n \geq 2$*.*

The essence of the proof remains the same. We first enlarge the graph (by a factor linear in m and n), then transform the graph so that it is in $X\Gamma_{mn}$, and then finally make sure each edge is replaced by an even length path.

$$\begin{pmatrix} & x & \\ x & x & x \\ & x & \end{pmatrix} \qquad \begin{pmatrix} x & & x \\ & x & \\ x & & x \end{pmatrix} \qquad \begin{pmatrix} & x & x \\ & x & \\ x & x & \end{pmatrix}$$

(a) (b) (c)

$$\begin{pmatrix} x & x & \\ & x & \\ & x & x \end{pmatrix} \qquad \begin{pmatrix} & & x \\ x & x & x \\ x & & \end{pmatrix} \qquad \begin{pmatrix} x & & \\ x & x & x \\ & & x \end{pmatrix}$$

(d) (e) (f)

Figure 9: (a) the cross block (b)-(f) the diagonal versions of the cross block

Details of our proof can be found in `http://www.cs.cmu.edu/~virgi/newm/paper.ps`.

Since our proofs rely solely on the fact that the matrix patterns we consider are bounded by an $m \times n$ rectangle, it does not matter whether the blocks are dense, or there are missing entries in the interior. We have a corollary as follows:

Corollary 5.2 *Let σ be an oriented shape, the outer boundary of which is the boundary of an $m \times n$ rectangular block. Then the MNS problem for this σ is NP-complete*

Another observation is that we need not restrict ourselves to viewing the vertical and horizontal gridlines of the grid as corresponding to the columns and rows of the matrix. We can view one or both of the gridlines to be (parallel) diagonals in the matrix. Moreover, rotating a shape by a right angle (i.e. swapping the gridline directions) does not change the NP-completeness results. Thus any composition of perpendicular rotation and changing gridline directions as described above converts a shape with a known NP-complete MNS problem to a new one with the same complexity. This technique is effective for a large variety of shapes. We were able to derive NP-completeness results for many other simple oriented substructures such as the so called *cross* block consisting of an entry (i, j), and its vertical and horizontal adjacent entries $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$ (Fig. 9a), all of its rotations, cross blocks with vertex subdivided legs, and many shapes with central symmetry.

We have presented an approach to improving the cache performance of sparse matrix operations by searching for a maximum number of nonoverlapping oriented matrix substructures which are to be included in a dense matrix component, leaving all other entries in the sparse component, so that $A = A_s + A_d$. Since many substructure patterns can be considered, a natural extension of this idea is to split the matrix into several dense components, each for a different shape. An alternative approach would be to pack all nonzero entries in a minimum number of disjoint substructures by allowing some of the zero entries to be used as nonzeros. This problem is likely to be hard, yet even a good approximation may prove to be useful as one would not need to split the matrix into a sum.

References

[1] B. Baker, Approximation algorithms for NP-complete problems on planar graphs, *Proc. 24th IEEE Symp. on Foundations of Computer Science* (1983), pages 265–273.

[2] F. Berman, D. Johnson, T. Leighton, P. Shor, L.Snyder, Generalized planar matching, *Journal of Algorithms* **11**, (1990), pages 153–184.

[3] H. de Fraysseix, J. Pach, R. Pollack, How to draw a planar graph on a grid, *Combinatorica*, **10** (1990), pages 41–51.

[4] M. Garey and D. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W.H. Freeman and Co., 1979.

[5] E. Im, K. Yelick, R. Vuduc, SPARSITY: An Optimization Framework for Sparse Matrix Kernels, *International Journal of High Performance Computing Applications*, **1**8 (1), 2004, to appear.

[6] G. Kant, Drawing planar graphs using the canonical ordering, *Algorithmica*, **16**, (1996), pages 4–32.

[7] A. Papakostas and I. Tollis, Algorithms for area-efficient orthogonal drawings. *Computational Geometry* **9** (1998), pages 83–110.

[8] A. Pınar, M. Heath, Improving performance of sparse matrix vector multiplication, *Proc. IEEE/ACM Conf. on Supercomputing 1999*, Portland, OR, November (1999) .

[9] R. Vuduc, J. Demmel, K. Yelick, S. Kamil, R. Nishtala, B. Lee, Performance Optimizations and Bounds for Sparse Matrix-Vector Multiply *Proc. IEEE/ACM Conf. on Supercomputing 2002*, Baltimore, MD, USA, November (2002).