Packing Squares into a Square*

JOSEPH Y-T. LEUNG, TOMMY W. TAM, AND C. S. WONG

Computer Science Program, University of Texas at Dallas, Richardson, Texas 75083

GILBERT H. YOUNG

Department of Computer Science, Tulane University, New Orleans, Louisiana 70118

AND

FRANCIS Y. L. CHIN

Department of Computer Science, University of Hong Kong, Hong Kong

The problem of determining whether a set of rectangles can be orthogonally packed into a larger rectangle has been studied as a geometric packing problem and as a floor planning problem. Recently, there is some renewed interest in this problem, as it is related to a job scheduling problem in a partitionable mesh connected system. In this paper we show that the problem of deciding whether a set of squares can be packed into a larger square is strongly NP-complete, answering an open question posed by Li and Cheng. (c) 1990 Academic Press, Inc.

1. INTRODUCTION

The problem of determining whether a set of rectangles can be packed into a larger rectangle has been studied as a geometric packing problem in [1] and as a floor planning problem in [4]. Recently, there is some renewed interest in this problem, as it is related to a job scheduling problem in a partitionable mesh connected system [3]. In this case, the larger rectangle represents a partitionable mesh connected system and the smaller rectangles represent jobs that require a submesh for processing. The problem is to determine whether a set of jobs can be simultaneously processed by the processors of the partitionable mesh connected system.

Li and Cheng [2] showed that the problem of determining whether a set of squares can be packed into a larger rectangle is strongly NP-complete. They also showed that the problem of packing a set of rectangles into a square is NP-complete, and they asked the question of whether the problem remains NP-complete for packing a set of squares into a square. In this paper we show that the problem is strongly NP-complete.

* Research supported in part by the-CNR Gift N00014-87-K-0839 and in part by a grant from Texas Instruments, Inc.

As in [3], the packings we consider are orthogonal packings, i.e., the sides of the squares are parallel to the vertical and horizontal axes.

2. PROBLEM DEFINITION AND PRELIMINARIES

Let $R$ be a square. The symbol $D(R)$ denotes the size of $R$ and is given by the positive integer $h$. In this section we formally define the square packing problem and derive some preliminary results.

Square Packing. Given a packing square $S$ and a set of packed squares $L = \{l_1, l_2, \ldots, l_k\}$, is there an orthogonal packing of $L$ into $S$?

Before we show that the square packing problem is strongly NP-complete, we consider a special instance of the problem called the $X$-instance. An $X$-instance has the interesting property that the packed squares can only be packed in a very restricted manner. It is defined as follows. For any integer $x \geq 2$, an $X$-instance consists of a packing square $S$ with $D(S) = x(x + 2)$ and a set of $x + 1$ packed squares $L = \{l_1, l_2, l_3, \ldots, l_{x+1}\}$ with sizes given as follows: $D(l_i) = x$ for $1 \leq i \leq x + 2$, $D(l_{x+3}) = x + 1$ for $x + 3 \leq i \leq 2x + 2$, and $D(l_{2x+4}) = x(x + 1) - 1$. We call the first $x + 2$ squares in $L$ the $x$-squares, the next $x$ squares the $(x+1)$-squares, and the last square the big square.

As an $X$-instance always has a solution; see Fig. 3 for a packing. In the remainder of this section we characterize the possible packings of an $X$-instance. Let $P$ be an orthogonal packing of $L$ into $S$. Clearly, the big square $l_{2x+4}$ must be packed in $S$ against at least one of the four sides of $S$, or otherwise the $(x+1)$-squares cannot be packed into $S$.

Without loss of generality, we may assume that the big square is packed against the side $C_yC_y$ of $S$, as shown in Fig. 1. This leaves five rectangular regions $R_1, R_2, R_3, R_4,$ and $R_5$ for
packing the remaining squares; see Fig. 1. In the next four lemmas, we characterize the possible packings of the remaining \(2x+2\) squares.

**Lemma 2.1.** In \(P\), at most one of the regions \(R_1\) and \(R_2\) can be used for packing the \(x\)-squares and the \((x+1)\)-squares.

**Proof.** Using the notation of Fig. 1, the heights of \(R_0\) and \(R_0\) are \(\mu\) and \(\gamma\), respectively. Since \(a + b = x + 1\) and since the remaining squares have size \(1\), it is impossible that both \(R_1\) and \(R_2\) are used for the packing of the \(x\)-squares and the \((x+1)\)-squares.

By Lemma 2.1, we may assume, without loss of generality, that \(R_1\) is used for packing the remaining squares. Let \(R' = R_0 \cup R_0, R'' = R_0 \cup R_1, R_1 = R_0 \cup R_0, \) and \(R'' = R_0 \cup R_0\). The regions \(R_1\) and \(R_2\) satisfy the properties as stated in the next lemma.

**Lemma 2.2.** In \(P\), at most one square can be completely packed along the vertical side of \(R_0\) and along the horizontal side of \(R_0\). Furthermore, there is a \(x\)-square packed against the corner \(C_1\) of \(S\) as shown in Fig. 2.

**Proof.** The width of \(R_0\) is \(x + 1\) and the height of \(R_0\) is at most \(x + 1\). Since the remaining squares have sizes at least \(x\), no two squares can be completely packed along the vertical side of \(R_0\) and along the horizontal side of \(R_0\).

The remaining \(2x+2\) squares are packed in \(R_0\). Let \(zi, 1 \leq i \leq 2x + 2\), be the square packed closest to the corner \(C_1\). Let \(Z\) be the set of squares packed along the horizontal (vertical) side of \(R_0\) (\(R_0\)). Observe that \(zi\) is in both \(Z_0\) and \(Z_0\). Since the width of \(R_0\) and the height of \(R_0\) are exactly \(x(x + 2)\), we have \(\sum_{i=1}^{2x+2} D(z_i) < x(x + 3)\) and hence \(\sum_{i=1}^{2x+2} D(z_i) < x(x + 2)\). This implies that \(\sum_{i=1}^{2x+2} D(z_i) < x(x + 2)\). Since \(\sum_{i=1}^{2x+2} D(z_i) = x(x + 3)\), we have\(D(z_i) = x\), and hence \(z_i\) must be an \(x\)-square. Furthermore, \(z_i\) must be packed against the corner \(C_1\).

**Lemma 2.3.** In \(P\), the \(x\)-squares are all packed in either \(R_0\) or \(R_0\).

**Proof.** By Lemma 2.2, there is an \(x\)-square packed against the corner \(C_1\). This implies that the remaining width of \(R_0\) and the remaining height of \(R_0\) are exactly \(x(x + 1)\). Since the remaining squares have total size \(2x(x + 1)\), they must be packed such that there are no gaps left along the horizontal side of \(R_0\) and along the vertical side of \(R_0\). Let \(k_1\) and \(k_2\) be the numbers of \(x\)-squares and \((x+1)\)-squares, respectively, packed in the remaining area of \(R_0\), where \(0 < k_1 < x + 1\) and \(0 < k_2 < x\). Then, we have

\[
k_1 x + k_2 (x + 1) = x(x + 1).
\]

Equation (1) has only two solutions: (i) \(k_1 = x + 1\) and \(k_2 = 0\) and (ii) \(k_1 = 0\) and \(k_2 = x\). Both solutions imply that the \(x\)-squares are all packed in either \(R_0\) or \(R_0\).

By Lemma 2.3, we may assume, without loss of generality, that the \(x\)-squares are all packed in \(R_0\). With this assumption, the \((x+1)\)-squares must all be packed in \(R_0\). Such a packing is called a canonical packing; see Fig. 3 for an example. Note that there are many canonical packings since the big square and some of the \((x+1)\)-squares can be moved vertically up or down. As we set in the next section, the remaining unused space in the packing square is used to pack the newly added packed squares. Thus, we need to introduce the notion of the best canonical packing. Let \(P^*\) be a canonical packing. We say that \(P^*\) is the best canonical packing if for any canonical packing \(P^*\) and any \(L\) of newly added squares that can be packed into the unused area of \(P^*\), \(L^*\) can also be packed into the unused area of \(P^*\). The next lemma characterizes the best canonical packing.
PACKING SQUARES INTO A SQUARE

Lemm 2.4. Let $P^*$ be the canonical packing such that the big square is packed against the $C_2C_3$ side of $S$ and the $x$-squares are all packed against the $C_2C_4$ side of $S$, as shown in Fig. 3. Then $P^*$ is the best canonical packing.

Proof. For any canonical packing $P$ and any list $L'$ of newly added squares that can be packed into the unshaded area of $P$, we can transform $P$ into $P^*$ such that the squares in $L'$ are all packed in the region $R_i$.

To conclude this section, we note that the best canonical packing is the packing we are concerned with in the next section.

3. NP-COMPLETENESS PROOF

In this section, we show that the square packing problem is strongly NP-complete by reducing the 3-partition problem to it. The 3-partition problem is known to be strongly NP-complete [2]. It can be stated as follows.

3-Partition. Given a list $a = (a_1, a_2, \ldots, a_n)$ of $3$ positive integers such that $\sum_{i=1}^{n} a_i = 3b$ and $b/4 < a_i < B/2$ for each $1 \leq i \leq 3z$, can $l = \{1, 2, \ldots, n\}$ be partitioned into $l_1, l_2, \ldots, l_z$ such that $\sum_{i \in l_j} a_i = B$ for each $1 \leq j \leq z$?

We begin by giving a reduction from the 3-partition problem to the square packing problem. Let $a = (a_1, a_2, \ldots, a_{3n})$ be an instance of the 3-partition problem. Without loss of generality, we may assume that $B > 2z$, as otherwise we can multiply each $a_i$ by $2z$ without changing the solution of the problem. Furthermore, we may assume that $z > 1$, as otherwise the problem is trivially solvable. Let $x$ be the smallest integer such that $x(x + 1) - 1 > 3z$, where $z$ is the smallest integer such that $3z > 2$ and $3z$ is an integral multiple of 3. Let $l = x(x + 1) - 1, \ y = (3l - z)/3, \text{ and } C = 3B^3$.

We call the squares in $L_2$ the $X$-squares, the squares in $L_1$ the partition squares, the squares in $L_4$ the dummy squares, the squares in $L_3$ the inflexible squares. Note that the $X$-squares and the packing square $S$ form an $X$-instance of
defined in the last section. Thus, the "best" way to pack the $X$-squares into $S$ is as in the best canonical packing. The next lemma shows that a solution to the 3-partition problem implies a solution to the square packing problem.

**Lemma 3.1.** If there is a solution to the 3-partition problem, there is a solution to the square packing problem.

**Proof.** Let $I_1, I_2, \ldots, I_k$ be a solution to the instance of the 3-partition problem. We divide the 32 partition squares into $z$ sets $Y_j = \{p_i \mid i \in I_j\}$ for each $1 \leq j \leq z$. If $i \neq z$, then $Y_j = \{(p_{1,j}, p_{2,j}, p_{3,j}) \mid 1 \leq j \leq z\}$. The packed squares can be packed as shown in Fig. 5. We leave it to the reader to verify that the packing in Fig. 5 is indeed a valid orthogonal packing of $L$ into $S$.

Next, we want to show that a solution to the square packing problem implies a solution to the 3-partition problem. Let $P$ be an orthogonal packing of $L$ into $S$. Since the $X$-squares and the packing square $S$ form an $X$- instances, we may assume that $P$ is the best canonical packing, as shown in Fig. 5. Therefore, the partition squares, the dummy squares, and the enforcement squares must all be packed in $R_1$; see Fig. 6. Let $L' = L_1 \cup L_2 \cup L_3$. Now we focus on the packing of $L'$. Let $P_1$ denote the "subpacking" of $P$ restricted to $L_1$. The next lemmas show that a valid packing of $L'$ in $P_1$, there is a solution to the 3-partition problem.

**Lemma 3.2.** In $P_1$, no squares can be packed along the vertical side with an enforcement square. Furthermore, if $P_1$ is a valid orthogonal packing of $L'$ in $P_1$, there is a solution to the 3-partition problem.

**Proof.** The height of $R_1$ is $3B + B$. Since the enforcement squares have sizes $3B$ and the remaining squares have sizes more than $B$, no square can be packed along the vertical side with an enforcement square in $P_1$.

If $P_1$ is a valid orthogonal packing of $L'$ in $R_1$, we may assume that the enforcement squares are all packed on the left side of $R_1$, as shown in Fig. 7. Let $R_i$ be the rectangular region left for packing the squares in $L_{2i-1} \cup L_{2i}$; see Fig. 7. The height of $R_i$ is $3B_i + B$ and the width of $R_i$ is at most $z_iB_i + B$. By our choice of $L_1$, it is easy to see that $i \leq B_i$, and hence the width of $R_i$ is at most $(z_i + 1)B_i$. Since the squares in $L_{2i} \cup L_{2i+1}$ all have sizes more than $B_i$, at most three $(z_i')$ squares can be packed along the vertical (horizontal) side of $R_i$. Since there is a total of $3z_i'$ squares in $L_{2i} \cup L_{2i+1}$, there will be exactly $z_i'$ stacks of squares packed along the horizontal side of $R_i$, with exactly three squares in each stack. Consequently, we can partition the $z_i'$ squares in $L_{2i} \cup L_{2i+1}$ into $z_i'$ groups $Y_1, Y_2, \ldots, Y_{z_i'}$, where each $Y_j$ contains exactly three squares. For each $1 \leq j \leq z_i'$, let $Y_j = \{p_{1,j}, p_{2,j}, p_{3,j}\}$. We have $D(p_{1,j}) + D(p_{2,j}) + D(p_{3,j}) \leq 3B_i + B$ for each $1 \leq j \leq z_i'$. Since the total size of all the squares in $L_{2i} \cup L_{2i+1}$ is $z_i'(3B_i + B)$, we must have $D(p_{1,j}) + D(p_{2,j}) + D(p_{3,j}) = 3B_i + B$ for each $1 \leq j \leq z_i'$.

If $i \neq z$, there are some dummy squares in $L_z$. By our choice of the sizes of the dummy squares, they cannot be in the same group as the partition squares, as otherwise there is no valid packing of $L_z \cup L_{z+1}$ in $R_z$. Without loss of generality, we may assume that the dummy squares are in $Y_{z-1}$, $1 \leq j < z_i'$. Clearly, $I_j = \{p_i \in I \mid 1 \leq j \leq z\}$ must form a solution to the 3-partition problem.

**Theorem 3.1.** The square packing problem is strongly $\text{NP}$-complete.

**Proof.** The square packing problem is clearly in $\text{NP}$, since a nondeterministic Turing machine can guess a position for each square and verify in polynomial time that the packing is valid. To complete the proof, it is easy to see that the 3-partition problem to the square packing problem as given above. Obliquely, the reduction can be done in polynomial time. By
4. CONCLUDING REMARKS

In this paper we have shown that the problem of determining whether a set of squares can be orthogonally packed into a larger square is strongly NP-complete. Our result implies all of the NP-completeness results shown in [3].

REFERENCES


JOSEPH Y.-T. LEUNG received his B.A. degree in mathematics from Southern Illinois University in 1972 and his Ph.D. degree in computer science from Pennsylvania State University in 1977. Since 1978, he has been on the faculty of Virginia Tech., Northern Arizona University, and the University of Texas at Dallas. Currently, he is Professor and Associate Program Head of the Computer Science Program at the University of Texas at Dallas. His research interests include scheduling theory, real-time systems, combinatorial optimization, and algorithms.

TOMMY W. TAM is a Ph.D. student in the Computer Science Program at the University of Texas at Dallas. He received his M.S. degree in computer science from the University of Texas at Dallas in 1988 and his B.S. degree in mechanical engineering from National Taiwan University in 1980. His research interests include real-time systems, scheduling theory, operating systems, and distributed computing. He is a member of IEEE, IEEE Computer Society, and ACM.

C. S. WONG is a Ph.D. student in the Computer Science Program at the University of Texas at Dallas. He received his B.S. degree in computer science from the University of Toronto in 1984. His research interests include operating systems, scheduling theory, real-time systems, and design and analysis of algorithms. He is a member of IEEE, IEEE Computer Society, and ACM.

GILBERT H. YOKO was born on November 9, 1961. He received his B.S. degree with a double-major in computer science and mathematics from the University of Oklahoma in May 1985. His M.S. and Ph.D. degrees in computer science at the University of Texas at Dallas in October 1986 and May 1989, respectively. Since July 1989, he has been an assistant professor at the Computer Science Department at Tuskegee University. His research interests include scheduling and storage allocation theory, real-time systems, algorithms, and combinatorial optimization.

FRANCES L. CHIM received the B.S. degree in engineering science from the University of Toronto in 1973 and the M.S., M.A., and Ph.D. degrees in mechanical engineering and computer science from Princeton University in 1975, 1977, and 1980, respectively. Since 1980, she has taught at various universities in North America, and since 1983, she has been teaching in Hong Kong. He is currently Professor and Head of the Department of Computer Science, University of Hong Kong. He has served as program chairman and committee member of several international conferences. His current research interests include algorithm design and analysis and parallel and distributed computing.