A MATRIX SYNTHESIS APPROACH TO THERMAL PLACEMENT*

Chris C. N. Chu and D. F. Wong
cnchu@cs.utexas.edu and wong@cs.utexas.edu
Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712.

ABSTRACT

In this paper, we consider the thermal placement problem for gate arrays. We introduce a new combinatorial optimization problem MSP (Matrix Synthesis Problem) to model the thermal placement problem. Given a list of mn non-negative real numbers and an integer t, MSP constructs a $m \times n$ matrix out of the given numbers such that the maximum sum among all $t \times t$ sub-matrices is minimized. We show that MSP is NP-complete and present several provably good approximation algorithms for the problem. We also demonstrate that our thermal placement strategy is flexible enough to allow simultaneous consideration of other objectives such as wiring.

1. INTRODUCTION

High performance circuits consume a considerable amount of power due to increases of frequency, bandwidth, and system integration. For examples, the two recent high-performance chips, Alpha 21164 and PowerPC 620, consume 50 W and 30 W, respectively, on 3 cm^2 dies. It can be extrapolated that a 10 cm^2 next-generation microprocessor, clocked at 500 MHz would consume 300 W [6]. Consumed power is converted directly into dissipated heat. In the past decade, heat produced by a chip has increased from 2.2 to 10 W/cm^2 due to the continuous increase of the clock frequency and the total number of transistors [8]. Higher temperature not only affects circuit performance directly by slowing down the transistors on CMOS chips but also decreases their reliability. A circuit with considerable power consumption requires extra expensive cost to remove heat at the packaging level, and therefore the reduction of power dissipation is required at the chip design stages. (See [6] for a survey of current research efforts in power minimization in IC design.) Even when the total power consumption of a chip is constrained, an unevenly distributed heat dissipation by the gates in the chip may produce hot spots which can lead to reliability problems. It is also desirable to have an even temperature distribution for the temperature-sensitive circuit (whose characteristic, such as the gain factor, β, of a CMOS or bipolar circuit, affects its output). Therefore, during physical design of a VLSI chip, it is important to place the gates such that heat dissipation by the gates are evenly distributed.

*This work was partially supported by a grant from the Avant! Corporation.

The thermal placement problem has been studied in the past for placing chips during the packaging stage (for PCBs and MCMs) [2, 4, 5]. However, since thermal placement of gates within a single chip was not of major concern in the past, existing placement algorithms [7] only focus on minimizing area and delay but do not consider heat dissipation. One exception is [1] but it only addresses thermal issues during IC floorplanning. In this paper, we consider the thermal placement problem for gate arrays. We introduce a new combinatorial optimization problem MSP (Matrix Synthesis Problem) to model the thermal placement problem.

Basically, MSP is to synthesize a matrix out of a given list of numbers such that no sub-matrix of a particular size has a large sum. In this paper, sub-matrix means those consisting of consecutive rows and columns. For any matrix M, let $S_t(M)$ be the set of all $t \times t$ sub-matrices of M. Let $\sigma(M)$ be the sum of all entries in M. Let $\mu_t(M) = \max_{S \in S_t(M)} \sigma(S)$. MSP can be defined formally as follows:

MATRIX SYNTHESIS PROBLEM (MSP)
INSTANCE: Integers t, m, n, and a list of mn non-negative real numbers $x_0, x_1, \ldots, x_{mn-1}$.
QUESTION: Synthesize a $m \times n$ matrix M out of $x_0, \ldots, x_{mn-1}$ such that $\mu_t(M)$ is minimized.

See Figure 1 for an example.

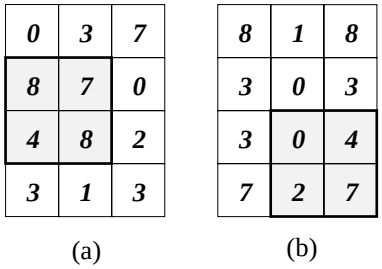

Figure 1. *An example of MSP with $m = 4, n = 3$ and $t = 2$. The problem is to synthesize a 4×3 matrix out of 12 numbers (8,8,7,7,4,3,3,3,2,1,0,0) and to minimize the maximum sum over all 2×2 sub-matrices. (a) is a bad solution (maximum sum is 27). (b) is an optimal solution (maximum sum is 13). The sub-matrices with maximum sum are shaded.*

It is not difficult to see that MSP models the thermal placement problem for gate arrays. We represent the amount of heat generated by each gate by a non-negative real number. (If we have less gates than the number of array slots, we can add some zeros.) A sub-matrix in $S_t(M)$ corresponds to a region of size $t \times t$ on the chip. The sub-matrix with the largest sum corresponds to the hottest region on the chip. So MSP is equivalent to finding a placement of the gates such that the temperature of the hottest region is the lowest among all possible placements.

The parameter t is to model how good the heat transfer is. If the heat transfer is poor such that the effect of a gate is mostly on neighbor gates, then MSP with $t = 2$ probably is a good model to use. On the other hand, if the heat transfer is good, we may want to consider larger regions and hence a larger t.

A summary of the remainder of this paper is given below. In Section 2, we show that MSP with any fixed $t \geq 2$ is NP-complete. (MSP with $t = 1$ is trivially in P.) Then in Section 3, we give a simple algorithm (called $A1$) that approximates MSP to within a factor of 2 for every $t \geq 2$. In Section 4, we give a modified version of $A1$ (called $A2$). For $t = 2$, $A2$ approximates MSP to within a factor of $5/3$. If a simple condition on the input is satisfied, $A2$ approximates MSP to within a factor of 1.5 for every $t \geq 2$. $A1$ and $A2$ output a placement which is good for a particular t only. In Section 5, we give a recursive algorithm (called $A3$) which outputs a single placement such that besides approximating MSP with parameter t, it also approximates MSP with parameter t' to within a factor of at most 5 for all $t' < t$. In Section 6, some experimental results are given. Firstly, note that the approximation factors shown in Sections 3, 4 and 5 are worst-case bounds only and we show that the algorithms work much better in practice. Secondly, we consider thermal placement and optimization of other objectives at the same time. It is because when we place gates into a chip, we may have other concerns besides heat consideration. We show that the placements by $A1$ and $A3$ are so flexible that the flexibility can be used in optimizing other objectives simultaneously. We demonstrate the idea by considering thermal distribution and wiring at the same time. In Section 7, we conclude by discussing some directions for future work.

2. NP-COMPLETENESS

MSP with $t = 1$ is very easy since every placement is optimal. However, we will show that MSP with every fixed $t \geq 2$ is NP-complete. In order to prove this result, we need the following definitions.

DECISION VERSION OF MSP
INSTANCE: A positive real bound B, integers t, m, n, and a list of mn non-negative real numbers $x_0, x_1, \ldots, x_{mn-1}$.
QUESTION: Is it possible to synthesize a $m \times n$ matrix M out of $x_0, x_1, \ldots, x_{mn-1}$ such that $\mu_t(M) \leq B$?

3-PARTITION
INSTANCE: A positive real bound B, and a multi-set X of $3q$ positive real numbers such that $\sum_{x \in X} x = qB$ and $\forall x \in X, B/4 < x < B/2$.
QUESTION: Can X be partitioned into q multi-sets $X_0, \ldots, X_{q-1}$ such that for $0 \leq r \leq q-1$, $\sum_{x \in X_r} x = B$?

Note that 3-PARTITION is NP-complete [3].

Theorem 1 *For every fixed $t \geq 2$, MSP is NP-complete.*

Proof outline: Let t be any fixed integer greater than or equal to 2. Given an instance of 3-PARTITION, we can reduce it to an instance of MSP with that particular value of t. The bound B for the MSP is the same as the B for the 3-PARTITION problem. We set $m = t$ and $n = tq$. The mn non-negative real numbers are those in X together with $mn - 3q$ zeros. We can show that the instance of 3-PARTITION returns "YES" if and only if the instance of MSP returns "YES". The details are omitted here. □

3. A SIMPLE APPROX. ALGORITHM

In this paper, we assume that the indices of matrices start at 0. Let $S_t^{ij}(M)$ be the $t \times t$ sub-matrix in $S_t(M)$ at the intersection of rows $i, \ldots, i+t-1$ and columns $j, \ldots, j+t-1$. Let $\widehat{S}_t(M)$ be the set of all $t \times t$ sub-matrices $S_t^{ij}(M)$ such that $i \equiv j \equiv 0 \pmod t$.

From now on, we assume for simplicity that $m = n = tq$ for some integer q. In other words, we are placing t^2q^2 numbers into a $tq \times tq$ matrix. Note that in this case $\widehat{S}_t(M)$ is a set of q^2 non-overlapping sub-matrices that covers the whole matrix M. We can obtain similar results if $m \neq n$, or m or n is not a multiple of t. Without loss of generality, we also assume that $x_0 \geq x_1 \geq \ldots \geq x_{n^2-1}$.

The algorithm $A1$ below approximates MSP to within a factor of 2. The basic idea of the algorithm is to distribute the numbers evenly among the matrix. We divide the numbers into $t \times t$ groups according to their magnitudes. We observe that it is possible to have a placement with the property that every $t \times t$ sub-matrix contains exactly one number from each group.

ALGORITHM $A1$

1. For $0 \leq k \leq t^2 - 1$, let group G_k contains the numbers $x_{kq^2}, \ldots, x_{kq^2+q^2-1}$.
2. For $0 \leq k \leq t^2-1$, for all $i \equiv \lfloor k/t \rfloor \pmod t$ and for all $j \equiv (k \bmod t) \pmod t$, label $m_{i,j}$ (entry (i,j) of matrix M) as L_k.
3. For $0 \leq k \leq t^2 - 1$, place each number in group G_k arbitrarily into a distinct position of M labeled with L_k.

For example, let $t = 2$, $m = n = 6$, $x_i = 35 - i$ for $0 \leq i \leq 35$. In other words, we are placing the numbers $35, 34, \ldots, 0$ into a 6×6 matrix. Then G_0 contains $35, \ldots, 27$, G_1 contains $26, \ldots, 18$, G_2 contains $17, \ldots, 9$, and G_3 contains $8, \ldots, 0$. The labeling is as shown in Figure 2. A possible placement is in Figure 3. Note that those numbers from group G_0 are evenly distributed in the matrix. This is also true for all other groups.

L_0	L_1	L_0	L_1	L_0	L_1
L_2	L_3	L_2	L_3	L_2	L_3
L_0	L_1	L_0	L_1	L_0	L_1
L_2	L_3	L_2	L_3	L_2	L_3
L_0	L_1	L_0	L_1	L_0	L_1
L_2	L_3	L_2	L_3	L_2	L_3

Figure 2. *Labeling of algorithm $A1$ with $t = 2$ and $n = 6$. Note that there is exactly one of each of L_0, L_1, L_2 and L_3 inside every 2×2 sub-matrix.*

Let OPT_t be the optimal placement for MSP with parameter t. Before proving the approximation factor for $A1$, we first give two lower bounds on $\mu_t(OPT_t)$.

30	**19**	**33**	**25**	**27**	**24**
17	**1**	**16**	**5**	**15**	**4**
35	**21**	**29**	**18**	**32**	**26**
14	**0**	**13**	**2**	**12**	**8**
31	**20**	**28**	**22**	**34**	**23**
11	**7**	**10**	**3**	**9**	**6**

Figure 3. *A possible placement by algorithm A1 for the numbers $35, 34, \ldots, 0$. The entries with label L_0 (i.e. numbers from group G_0) are shaded.*

Lemma 1 *For every $t \geq 2$, $0 \leq k \leq t^2 - 1$, $\mu_t(OPT_t) \geq (k+1)x_{kq^2}$.*

Proof: $x_0, \ldots, x_{kq^2}$ are $kq^2 + 1$ numbers at least as large as x_{kq^2}. Consider the q^2 sub-matrices in $\widehat{S}_t(OPT_t)$. By pigeonhole principle, there must be a sub-matrix containing at least $k+1$ numbers larger than or equal to x_{kq^2}. So $\mu_t(OPT_t) \geq (k+1)x_{kq^2}$. □

Lemma 2 *For every $t \geq 2$, $\mu_t(OPT_t) \geq \frac{1}{q^2}\sum_{i=0}^{n^2-1} x_i$.*

Proof:

$$\mu_t(OPT_t) \geq \frac{1}{q^2} \sum_{S \in \widehat{S}_t(OPT_t)} \sigma(S) \quad (1)$$

$$= \frac{1}{q^2} \sum_{i=0}^{n^2-1} x_i \quad (2)$$

Line (1) follows from the fact that $\mu_t(OPT_t) \geq \sigma(S)$ for any $S \in \widehat{S}_t(OPT_t)$ and $|\widehat{S}_t(OPT_t)| = q^2$. Line (2) follows from the fact that $\widehat{S}_t(M)$ is a set of non-overlapping sub-matrices that covers the whole matrix M. □

Theorem 2 *For every $t \geq 2$, $\mu_t(A1) \leq 2 \cdot \mu_t(OPT_t)$.*

Proof:

$$\mu_t(A1) \leq x_0 + x_{q^2} + \cdots + x_{(t^2-1)q^2} \quad (3)$$

$$\leq x_0 + \frac{1}{q^2}\sum_{i=0}^{q^2-1} x_i + \cdots + \frac{1}{q^2}\sum_{i=(t^2-2)q^2}^{(t^2-1)q^2-1} x_i \quad (4)$$

$$\leq x_0 + \frac{1}{q^2}\sum_{i=0}^{n^2-1} x_i$$

$$\leq 2 \cdot \mu_t(OPT_t) \quad (5)$$

By the way we place the numbers, each $t \times t$ sub-matrix contains exactly one number from each group G_k. Note that $x \leq x_{kq^2}$ for every number x in G_k. So for any $S \in S_t(A1), \sigma(S) \leq x_0 + x_{q^2} + \cdots + x_{(t^2-1)q^2}$. Line (3) immediately follows. Line (4) follows from the fact that $x_{kq^2} \leq x_{kq^2-r}$ for $1 \leq r \leq q^2$ as the numbers are sorted in decreasing order. Line (5) follows from Lemma 1 with $k = 0$ and Lemma 2. □

4. A BETTER APPROX. ALGORITHM

In step 3 of algorithm $A1$, the placement of numbers from group G_k into entries marked with label L_k is done arbitrarily. The algorithm $A2$ given below makes use of this flexibility on placement to improve the approximation factor.

ALGORITHM $A2$

1. For $0 \leq k \leq t^2 - 1$, let group G_k contains the numbers $x_{kq^2}, \ldots, x_{kq^2+q^2-1}$.
2. For $0 \leq k \leq t^2-1$, for all $i \equiv \lfloor k/t \rfloor \pmod t$ and for all $j \equiv (k \bmod t) \pmod t$, label m_{ij} (entry (i,j) of matrix M) as L_k.
3. Place each number of group G_0 into a distinct position of M labeled with L_0 (i.e. into m_{ij} s.t. i and j are multiple of t) such that $m_{ut,vt} \geq m_{ut+t,vt}$ and $m_{ut,vt} \geq m_{ut,vt+t}$ for all u, v.
4. For $0 \leq r \leq q^2 - 1$, let $S_r \in \widehat{S}_t(M)$ be the sub-matrix where x_r is placed at step 3. For $1 \leq k \leq t^2 - 1$, place $x_{kq^2+q^2-1-r} \in G_k$ into the entry with label L_k in S_r.

One way to do step 3 is to place x_r into $m_{ut,vt}$ where $u = \lfloor r/q \rfloor, v = (r \bmod q)$. Figure 4 illustrates this step.

x_0	x_{17}	x_1	x_{16}	x_2	x_{15}
x_{26}	x_{35}	x_{25}	x_{34}	x_{24}	x_{33}
x_3	x_{14}	x_4	x_{13}	x_5	x_{12}
x_{23}	x_{32}	x_{22}	x_{31}	x_{21}	x_{30}
x_6	x_{11}	x_7	x_{10}	x_8	x_9
x_{20}	x_{29}	x_{19}	x_{28}	x_{18}	x_{27}

Figure 4. *A possible implementation for step 3 of algorithm A2 with $n = 6$ and $t = 2$. The entries with label L_0 are shaded.*

The algorithm matches larger numbers from group G_0 with smaller numbers from other groups. So it prevents all the largest numbers of the groups from being placed into the same $t \times t$ sub-matrix. Intuitively, one might think that it would be better to match larger numbers from half of the groups with smaller numbers from the other half of the groups. However, the worst-case bound is better for our algorithm.

Theorem 3 *For every $t \geq 2$, if $x_{q^2-1} = \alpha x_0$, then $\mu_t(A2) \leq \max(1.5, 2-\alpha) \cdot \mu_t(OPT_t)$.*

Proof outline: By the way we place the numbers in step 3 and step 4, we can show that $\sigma(S_t^{ij}(A2)) \leq \sigma(S_t^{\lceil i/t \rceil t, \lceil j/t \rceil t}(A2))$ for any i, j. In other words, the sum of every sub-matrix is dominated by the sum of some sub-matrix in $\widehat{S}_t(A2)$. Hence we can focus on those sub-matrices in $\widehat{S}_t(A2)$. By a similar (but much more complicated) proof as in Theorem 2, we can prove that for any $S \in \widehat{S}_t(A2)$, $\sigma(S) \leq \max(1.5, 2-\alpha) \cdot \mu_t(OPT_t)$ using the fact that x's are sorted in decreasing order, $x_{q^2-1} = \alpha x_0$, Lemma 1

with $k = 0$ and $k = 1$, and Lemma 2. So $\mu_t(A2) \leq \max(1.5, 2-\alpha) \cdot \mu_t(OPT_t)$. □

Note that Theorem 3 gives a bound worse than 1.5 only when α is small (less than 0.5). In this case, the input should contain a few large numbers and a lot small numbers.

For the case $t = 2$, we can prove a bound that holds for any input. But we need to use another lower bound of $\mu_t(OPT_t)$.

Lemma 3 *For all $t \geq 2$ and for all r s.t. $0 \leq r \leq n^2 - 1$, $\mu_t(OPT_t) \geq x_r + x_{n^2-1-r}$.*

Proof: $x_0, \ldots, x_r$ are $r+1$ numbers larger than or equal to x_r. Consider the q^2 sub-matrices in $\widehat{S}_t(OPT_t)$. If any two of these numbers are in the same sub-matrix, then the lemma is obviously true. Consider the case when they are in $r+1$ different sub-matrices in $\widehat{S}_t(OPT_t)$. Since there are at most r numbers less than x_{n^2-1-r}, at least one of these $r+1$ sub-matrices must contain some number larger than or equal to x_{n^2-1-r}. Hence, the result follows. □

Theorem 4 *For $t = 2$, $\mu_2(A2) \leq \frac{5}{3} \cdot \mu_2(OPT_2)$.*

Proof outline: As in Theorem 3, we will focus on those sub-matrix in $\widehat{S}_2(A2)$. By a similar (but much more complicated) proof as in Theorem 2, we can prove for any $S \in \widehat{S}_2(A2)$, $\sigma(S) \leq \frac{5}{3} \cdot \mu_2(OPT_2)$, using the fact that x's are sorted in decreasing order, Lemma 1 with $k = 1$ and $k = 2$, Lemma 2 and Lemma 3. So for $t = 2$, $\mu_2(A2) \leq \frac{5}{3} \cdot \mu_2(OPT_2)$. □

5. A RECURSIVE APPROX. ALGORITHM

For the thermal placement problem, if the heat transfer is good, it is reasonable to consider larger regions and hence to use a larger t. Smaller regions will become less important as heat generated will be dissipated to other parts of the chip easily. Even if a lot of heat is generated in a small region, if its surrounding region does not generate much heat, the heat will spread out quickly to a larger region. However, it does not mean that the heat consideration of smaller regions is totally unimportant. One may still want to have some bounds on the amount of heat generated by smaller regions.

In the previous two sections, we present two algorithms $A1$ and $A2$ that give placements which are good for a particular t. If we consider a parameter $t' < t$, those placements generated with parameter t do not give you much guarantee on the approximation factor. For example, if we run $A1$ with $t = 4$, the numbers from G_0, G_1, G_4 and G_5 will be placed next to each other. As the numbers from these 4 groups are relatively large, if we run $A1$ with $t = 4$, $\mu_2(A1)$ may be large.

It can be easily seen that the problem with the previous two algorithms is that there is no intention to distribute the numbers from different groups evenly inside a $t \times t$ sub-matrix. If we do the labeling carefully, we should be able to obtain better bounds for smaller sub-matrices. In this section, we give an algorithm $A3$ which outputs a single placement such that besides approximating MSP with parameter t to within a factor 2, it also approximates MSP with parameter t' to within a factor of at most 5 for all $t' < t$, when t is a power of 2.

The idea is to do the labeling by $A1$ with $t = 2$ recursively. For a $2q \times 2q$ matrix labeled by $A1$ with $t = 2$, if we consider the $q \times q$ matrix formed by removing all the entries other than those marked with L_0, and apply $A1$ with $t = 2$ again to place the q^2 numbers of G_0 into it, then we know that the largest numbers of G_0 will not be placed adjacent to each other in the original matrix. We can continue the idea recursively until the groups we are considering are small enough. Then we can apply the same procedure to G_1, G_2 and G_3. The algorithm is given below.

ALGORITHM $A3$

1. Divide the input numbers into 4 groups G_0, G_1, G_2 and G_3 and label the matrix by L_0, L_1, L_2 and L_3 as in step 1 and 2 of algorithm $A1$ with $t = 2$.
2. Recursively place the numbers in G_0 into the sub-matrix formed by entries marked with L_0 until the size of each group is n^2/t^2. In that case, we do the placement arbitrarily instead of doing it recursively.
3. Apply the same procedure to G_1, G_2 and G_3.

Note that we assume t is a power of 2 in algorithm $A3$. If t is not a power of 2, we can use the smallest power of 2 bigger than t as the parameter for $A3$ instead.

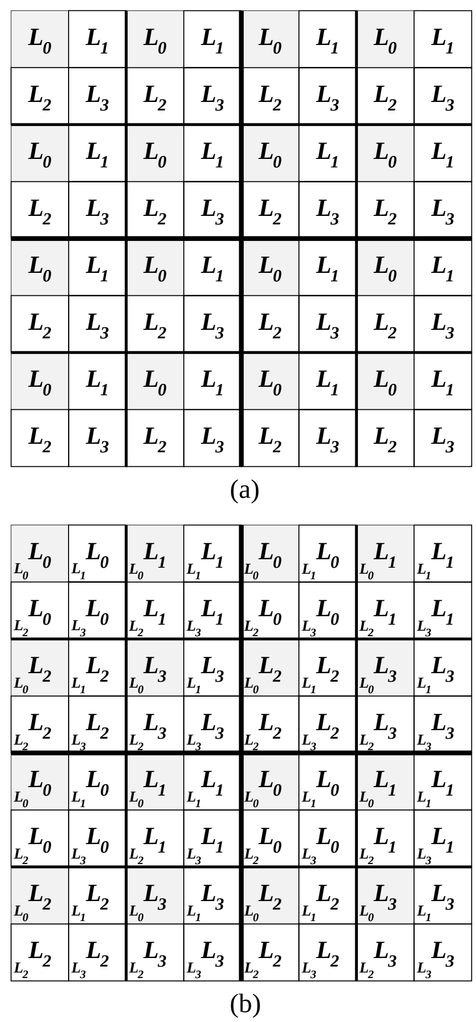

Figure 5. *The labeling of A3 with $t = 4$ and $n = 8$. (a) is the labels for the first level of recursion. Those entries labeled with L_0 at this step are shaded. (b) is the labels for the second level of recursive. The labels for the first level are written at lower left corners.*

An example of the labeling is shown in Figure 5. Basically, as in $A1$ with $t = 4$, we are dividing the input numbers into 16 groups (4 groups in the first level of recursion and then 16 groups in the second level) such that there is exactly one number from each group in every 4×4 sub-matrix. So the sum of every 4×4 sub-matrix will not differ by too much. However, because of the way we do the labeling, numbers from different groups are evenly distributed inside every 4×4 sub-matrix. So we can obtain some bounds for 3×3 and 2×2 sub-matrices too.

Theorem 5 *Suppose t is a power of 2. For any t' such that $2 \leq t' \leq t$, let p be the integer such that $2^{p-1} < t' \leq 2^p$. Then $\mu_{t'}(A3) \leq (1 - (2^p/n)^2 + (2^p/t')^2) \cdot \mu_{t'}(OPT_{t'})$.*

Proof outline: Let r be the integer such that $t = 2^r$. Note that for any $S \in S_{t'}(A3)$, $\sigma(S) \leq x_0 + x_{2^{2r-2p}q^2} + x_{2 \cdot 2^{2r-2p}q^2} + \cdots + x_{(2^{2p}-1)2^{2r-2p}q^2}$. Using the fact that x's are sorted in decreasing order, Lemma 1 with $k = 0$ and Lemma 2, we can prove that $\sigma(S) \leq (1-(2^p/n)^2+(2^p/t')^2) \cdot \mu_{t'}(OPT_{t'})$ and hence the Theorem follows. □

Note that if t' is a power of 2, the approximation factor is at most 2. Otherwise, the approximation factor is at most $(1 + (2^p/2^{p-1})^2) = 5$.

6. EXPERIMENTAL RESULTS

The approximation factor bounds for the algorithms shown in the previous three sections are all worst-case bounds only. We show here that these algorithms perform much better in practice.

As we do not have any actual thermal information for circuits, we generate thermal information uniformly at random. 10 sets of data of size 120×120 are generated. In Table 1, the average approximation factors over the 10 data are shown when algorithms $A1$ and $A2$ with various values of t are used to place them into a 120×120 matrix. For algorithm $A1$, the placement of numbers inside a group is done randomly. We also include the results of random placements for comparison. If the placement of gates is independent of the amount of heat generated by the gates, then the resulting placement should be similar to a random placement in terms of heat distribution.

	Avg. Approx. Factor		
t	$A1$	$A2$	Random
2	1.218	1.125	1.899
3	1.079	1.085	1.714
4	1.033	1.054	1.646
5	1.018	1.038	1.480
Average	1.087	1.076	1.685

Table 1. *Average Approximation factors for $A1$ and $A2$.*

As shown in the table, the approximation factors of our algorithms are very close to optimal in practice. They also perform much better than random placements. Note that as we do not know the optimal value $\mu_t(OPT_t)$, we only use the maximum of the lower bounds in Lemma 1, Lemma 2, and Lemma 3 as an approximation of it. The approximation factors should be even better if optimal values are used.

In Table 2, the average approximation factors over the same sets of data for algorithm $A3$ are shown. We use $t = 8$ here and the approximation factors for $t' < 8$ are also shown. The worst-case bounds proved in Theorem 5 and the results of random placements are included for comparison.

As shown in Table 2, the algorithm gives pretty good approximation factors simultaneously for all t'. It performs much better in practice than the upper bounds suggest. It also performs much better than random placements. Again, we can only use the lower bounds in Lemma 1, Lemma 2, and Lemma 3 to approximate the optimal values.

	Worst-case Bound	Avg. Approx. Factor	
t'	for $A3$	$A3$ with $t = 8$	Random
2	2.000	1.247	1.899
3	2.777	1.370	1.714
4	1.999	1.053	1.646
5	3.556	1.222	1.480
6	2.773	1.084	1.388
7	2.302	1.263	1.454
8	1.996	1.006	1.287
Average	2.486	1.178	1.553

Table 2. *The worst-case bounds $(1 - (2^p/n)^2 + (2^p/t')^2$ where $p = \lceil \log_2 t' \rceil)$ and the average values of the approximation factors of algorithm $A3$ with $t = 8$ for different t'.*

Figure 6 and Figure 7 show the heat distribution of a random placement and a placement by $A1$ with $t = 4$ respectively. The brightness at each point is proportional to the total amount of heat generated by a surrounding region of size 4×4. As we can see, there are many hot spots in the random placement. On the contrary, the heat is very evenly distributed in the placement by $A1$.

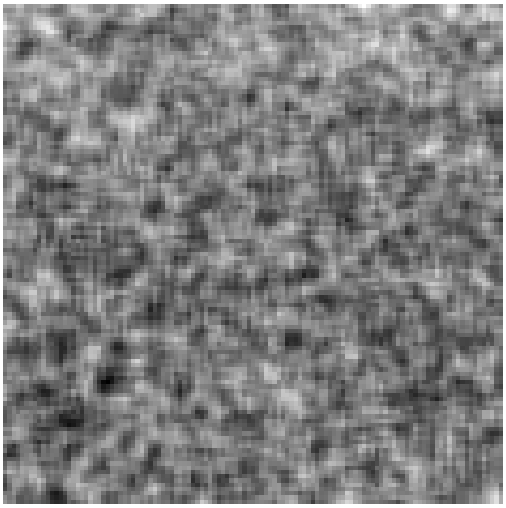

Figure 6. *Heat distribution of a random placement. There are many hot spots (white spots) in this placement.*

When we place gates into a chip, we usually have to optimize other objectives at the same time. For algorithms $A1$ and $A3$, there is large flexibility to do the placement because the algorithms only require a number to be place in any of those entries with a particular label. Moreover the entries with that particular label are plenty and are evenly distributed on the matrix.

We observe that such flexibility can be used to simultaneously optimize other objectives. We demonstrate the idea by considering heat distribution and wiring at the same time. A set of MCNC benchmark circuits was used. Since thermal data of these circuits were not available, we generated a number uniformly at random for each gate representing the amount of heat dissipated by the gate. We first obtain a thermally good placement by our thermal placement algorithm $A1$ with $t = 2$. Then we try to improve the total wiring length by simulated annealing. However, we only al-

Circuit			Wiring			Heat		
name	size	n	Traditional	Our Alg.	inc%	Traditional	Our Alg.	dec%
s5378	2978	55	23912	23912	0.0	1.878	1.224	34.8
s9234	5844	77	58209	58546	0.6	1.882	1.226	34.9
s13207	8727	94	94698	95547	0.9	1.934	1.224	36.7
s15850	10397	102	128369	130003	1.3	1.889	1.216	35.6
s38584	20871	145	375121	375577	0.1	1.949	1.222	37.3
s38417	24061	156	444150	447792	0.8	1.893	1.244	34.3
Average					0.6			35.6

Table 3. *Comparison of traditional placement based on the wiring objective only and our approach of placement which considers both heat distribution and wiring.*

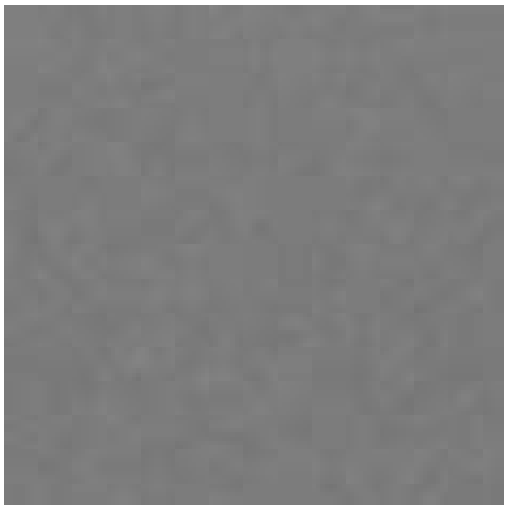

Figure 7. *Heat distribution of a placement by A1. There is no hot spot (white spot) in this placement. The heat is evenly distributed.*

low the exchange of two entries such that the differences in row indices and in column indices are both multiples of t. So as far as heat is concerned, the placement after the simulated annealing is as good as the one before. As for comparison, we also consider traditional placement based on the wiring objective only. That is, in our experiment, we apply simulated annealing to a random initial placement, using total wire length as the objective, and without imposing any restrictions on the gate locations as was done in the other case. It corresponds to the case when heat is not taken into consideration. Table 3 are the results of the experiment.

As expected, our algorithm is not as good as usual simulated annealing in terms of total wire length. However, the increase is very insignificant. On the other hand, our algorithm performs much better in distributing the heat.

7. CONCLUDING REMARKS

We have introduced a new combinatorial problem MSP (Matrix Synthesis Problem) to model the thermal placement problem. We show that MSP is NP-complete and we give several provably good approximation algorithms for it. The algorithms are fast, flexibility and good both theoretically and practically in providing an approximate solution.

A direction of future work is to design algorithms with provably better approximation factors for MSP. As we pointed out at Section 5, one may want to have bounds on several values of t simultaneously. The worst-case bounds given by $A3$ sometimes can be as large as 5. It is good to have algorithms with better worst-case bounds. We can also generalize MSP by considering a weighted average of the approximation factors for different values of t. This model gives more guarantee than MSP and it may be easier to work with than the model of providing several bounds simultaneously. However, we have no idea how the weights should look like. It is worthwhile to investigate what the weights should be and to design approximation algorithms according to the weight distribution. Another direction is to obtain a simple model which gives the temperature for each point on the chip. In fact, the temperature distribution for a given placement can be found by numerically solving differential equations but such calculations are too expensive to be used by a placement algorithm.

ACKNOWLEDGMENT

The authors thank Dr. K.Y. Chao of Intel Corporation for his helpful comments.

REFERENCES

[1] K. Y. Chao and D. F. Wong. Low power considerations in floorplan design. In *Proceedings of the 1994 International Workshop on Low Power Design*, pages 45–50, 1994.

[2] K. Y. Chao and D. F. Wong. Thermal placement for high performance multi-chip modules. In *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, Oct. 1995.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, NY, 1979.

[4] M. D. Osterman and M. Pecht. Component placement for reliability on conductively cooled printed wiring boards. *ASME J. of Packaging*, 111(3):149–156, 1989.

[5] M. D. Osterman and M. Pecht. Placement for reliability and routability of convectively cooled PWBs. *IEEE Transactions on CAD*, 9(7):734–744, 1990.

[6] M. Pedram. Power minimization in IC design: Principles and applications. *ACM Transactions on Design Automation of Electronic Systems*, 1(1):3–56, 1996.

[7] B. T. Preas and M. J. Lorenzetti. *Physical Design Automation of VLSI Systems*. Benjamin Cummings, Menlo Park, CA, 1988.

[8] R. E. Simons. Microelectronics cooling and SemiTherm: A look back. In *Proceedings of the 10th Semiconductor Thermal and Temperature Measurement Symposium*, pages 1–16, 1994.