# CS 124: Programming Assignment 2

Hao Wang Chao Cheng

April 1, 2020

# 1 Determination of Cross-over Point

## 1.1 Analytical Results

**Conventional Algorithm** The conventional algorithm for multiplying two $n \times n$ matrices $A$ and $B$ calculates the product matrix $C = A \times B$. Calculating each entry $C_{i,j} = \sum_k^n A_{i,k} B_{k,j}$ requires $n$ multiplication and $n-1$ addition operations. Since there are $n^2$ total entries in $C$, the overall cost of the conventional algorithm is $n^2(2n-1) = 2n^3 - n^2$ , assuming the cost of any single arithmetic operation is 1.

**Strassen Algorithm** The Strassen algorithm for multiplying two $n \times n$ matrices performs 7 recursive multiplication calls on size $\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$ matrices and 18 matrix additions on size $\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$ matrices. Therefore, the overall cost of Strassen algorithm is $7T\left(\lceil \frac{n}{2} \rceil\right) + 18\left(\lceil \frac{n}{2} \rceil\right)^2$, assuming the cost of any single arithmetic operation is 1.

*Note.* If $n$ is odd, we must pad the matrix with an additional column and row of 0's in order to split into sub-matrices evenly, hence the ceiling function.

**Modified Strassen** If we modify Strassen by switching to the conventional algorithm when $n$ becomes smaller than $n_0$, the cross-over point, then the overall cost of the modified Strassen is:

$$T(n) = \begin{cases} 7T\left(\lceil \frac{n}{2} \rceil\right) + 18\left(\lceil \frac{n}{2} \rceil\right)^2 & n > n_0 \\ 2n^3 - n^2 & n \leq n_0 \end{cases}$$

We want to determine the cross-over point, $n_0$, such that the switch occurs at the optimal time, which happens when the cost of the conventional algorithm is less than the cost of Strassen, or in mathematical terms:

$$2n^3 - n^2 < 7T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 18\left(\left\lceil \frac{n}{2} \right\rceil\right)^2$$

Since $n_0$ is the cross-over point, any recursive call will return the conventional cost $2n^3 - n^2$, so we have

$$2n^3 - n^2 < 7\left(2\left\lceil \frac{n}{2} \right\rceil^3 - \left\lceil \frac{n}{2} \right\rceil^2\right) + 18\left(\left\lceil \frac{n}{2} \right\rceil\right)^2$$

Due to the ceiling function, we now have two cases to consider:

- **Case 1:** $n$ is even.
  In this case, $\lceil \frac{n}{2} \rceil$ simplifies to $\frac{n}{2}$, so we have

$$\begin{aligned} 2n^3 - n^2 &< 7\left(2\left(\frac{n}{2}\right)^3 - \left(\frac{n}{2}\right)^2\right) + 18\left(\frac{n}{2}\right)^2 \\ 2n^3 - n^2 &< \frac{7}{4}n^3 - \frac{7}{4}n^2 + \frac{18}{4}n^2 \\ \frac{1}{4}n^3 &< \frac{15}{4}n^2 \\ n &< 15 \end{aligned}$$

  Thus for even $n$, we have $n_0 = 15$ as the cross-over point.

- **Case 2:** $n$ is odd.
  In this case, $\lceil \frac{n}{2} \rceil$ simplifies to $\frac{n+1}{2}$, so we have

$$2n^3 - n^2 < 7\left(2\left(\frac{n+1}{2}\right)^3 - \left(\frac{n+1}{2}\right)^2\right) + 18\left(\frac{n+1}{2}\right)^2$$
$$2n^3 - n^2 < \frac{7}{4}(n+1)^3 + \frac{11}{4}(n+1)^2$$
$$n < 37.169879057440357182$$

  Thus for odd $n$, we have $n_0 = 37$ as the cross-over point.

## 1.2 Experimental Results

To determine the optimal cross-over point experimentally for a range of matrix sizes, we tested matrices with dimensions that were powers of 2, 3, 5, or 10, up until 1024. For each dimension, we tested from 1 to $\frac{dim}{2}$ as cross-over points, and recorded the cross-over point that had the fastest runtime, as shown in the table below.

| **Optimal Cross-over Points** | | |
|---|---|---|
| *Dim.* | *Cross-over* | *Runtime(ms)* |
| 1 | - | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 0 |
| 4 | 1 | 0 |
| 5 | 2 | 0 |
| 8 | 2 | 0 |
| 9 | 2 | 0 |
| 10 | 3 | 0 |
| 16 | 4 | 0 |
| 25 | 8 | 0 |
| 27 | 7 | 0 |
| 32 | 5 | 0 |
| 64 | 17 | 0 |
| 81 | 21 | 1 |
| 100 | 26 | 1 |
| 125 | 32 | 3 |
| 128 | 52 | 2 |
| 243 | 92 | 12 |
| 256 | 67 | 13 |
| 512 | 140 | 87 |
| 625 | 167 | 160 |
| 729 | 186 | 247 |
| 1000 | 125 | 581 |
| 1024 | 194 | 614 |

As the results show, the optimal cross-over point increases as the dimension increases, and furthermore, the relationship seems to be linear (very roughly, the cross-over point appears to be 1/8th of the dimension). However, if we do not know the size of the dimension beforehand, then we propose setting $n_0 = 32$ as the cross-over point, since it is the median of the cross-over points and should work well on average for a variety of different dimensions.

### 1.3 Optimization on Strassen's Algorithm

**Memory Allocation** A major concern regarding the Strassen Algorithm is that, for each recursion, the matrix must be separated into several block matrices, requiring a large amount of memory allocation. In order to reduce unnecessary memory allocation, in our modified Strassen Algorithm, we only store the block matrices that are required in the multiplication. For example, for the product $(A_{11} + A_{12}) * B_{11}$, we only require the result of $A_{11} + A_{12}$, so we only store the result of this sum, and $B_{11}$, in memory when we calculate the product.

**Cache Locality** Strassen's Algorithm depends on conventional matrix multiplication as its base case, so we can increase the efficiency of Strassen by increasing the efficiency of our conventional matrix multiplication. Because our computers cache data horizontally, we want to access data horizontally as well. However, in conventional matrix multiplication, without optimizing cache locality, we access certain data vertically. Therefore, by changing the order of accessing data in matrix multiplication, we can maximize the hit rate of our cache to increase efficiency.

## 2 Counting Triangles in Graphs

The table below shows the results of using the modified Strassen to compute the number of triangles in a graph represented by an adjacency matrix where each edge has a probability $p$ of being included. The expected values were calculated using the formula $\binom{dim}{2}p^3$.

| **Number of Triangles** | | | |
|---|---|---|---|
| *Dim.* | *p-value* | *Result* | *Expected* |
| 1024 | 0.01 | 184 | 178.433 |
| 1024 | 0.02 | 1433 | 1427.464 |
| 1024 | 0.03 | 4927 | 4817.692 |
| 1024 | 0.04 | 11670 | 11419.714 |
| 1024 | 0.05 | 22478 | 22304.128 |