Analyzing Traffic Problem Model With Graph Theory Algorithms

Yong Tan
entermarket@163.com

Abstract
In this paper, author will discuss a practical problem Urban Traffic with graph theory algorithms. We will study those features, and try to simplify the complexity of this dynamic system. These contents mainly contain how to analyze a decision problem with Combinatorial Formula and graph theory algorithms, how to optimize our strategy to gain a feasible solution with other method of Computer Science.

ACM class: I.2.8; MSC class: 90B20, 90C27, 91B74

Keyword. Traffic Problem; Graph Partition Algorithm; BOTS Algorithm; Dynamic System

1 Introduction
This paper derives from the paper Construct Graph Logic[1] written by author before, which was published on arxiv.org. In that paper, it introduces the graph logic and algorithms, including BOTS and graph partition with their proofs where these facts are proved: BOTS can find out each path on an arbitrary instance and we may use graph partition to construct a hypergraph isomorphic of instance. Hence It is natural case for author use these tools sequentially to analyze traffic problem.

In our opinion, this problem is complex, dynamic and random, presently where it is simple to be identified as how to find a shortest path between two endpoints in a city, such as BellmanFord algorithm[2] working on fixed weighted graph.

Therefore, we will discuss different case no longer on a fixed weighted graph. In other word, the status of each street traffic will be decided by a function with several variables which maybe involve to the time frame, oil price, weather, holiday and etc. Consequently, we have to find a strategy in where having a physical quantity can be represented the superposition result affected by these factors. Hence, firstly author asserts the problem is relevant to timing sequence, so that we can round this assertion to find a feasible and integral solution, which is dynamic, flexible and real-time.
Overview. In the rest of this section, these basic knowledge of graph data structure and algorithms in Section 2. We will discuss the features and the solution for the traffic problem in Section 3. Section 4 will give the experimental data of various figures for modeling. Then in section 5, after we discuss the complexity of objects, we are going to look for optimized strategy on those fore conclusions. Section 6 will introduce the model on industrial level and discuss some practical problems.

Finally, we will summarize above contents and pose relevant future work.

2 Preliminaries

We are interested in the finite and connected graph, i.e. for given a node on instance such that we at least can find a path from it to another. We reserve the letter \( n \) as the number of nodes on an instance.

Given a graph \( G \) with a collection \( V \) of nodes, if we can walk on it then there exists a relation \( \tau \) we call it Traversal Relation \([1]\), which is the subset of the Cartesian Product \( V \times V \). The collection \( S \) of Unit Subgraph is the set partition of \( \tau \). Where each Unit Subgraph \( s \) similarly is a Cartesian Product with a root and several leaves: \( R(s) \times L(s) \).

Graph partition \([5]\) is a heuristic method which partition those nodes on instance to an ordered sequence components \( (\sigma_1, \sigma_2, \cdots, \sigma_k) \), showing as following diagram.

![Diagram 1](image)

For each node \( u \in \sigma_i (i > 1) \), at least there exists a node \( v \in \sigma_{i-1} \), having \( (v, u) \in \tau \). Namely, for node \( u \in \sigma \), there is node \( v \in \sigma_{i-1} \) with an arc pointing to \( u \). For a simple graph, we proved the length of shortest path from start-node to another on \( i \)th region \( \sigma_i \) is equal to \( i - 1 \). Where we say that shortest path between two endpoints is the minimum quantity of nodes inserted in it.

Such as nodes \( v_1 \) to \( v_{19} \) on Diagram 1, the length of path from \( v_1 \) to \( v_{19} \) is greater than and equal to 5, eg. \( (v_1 - v_2 - v_3 - v_{10} - v_{11} - v_{19}) \).
3 Features and Impacted Factors

We firstly give those features of Urban Traffic from our observation as follow:

1. For a street, the *Expedite Index* is pass-time nor the length, especially in period of traffic jam.

2. The *Expedite Index* is dynamic, i.e. while you reach a medium node on a path, those indices of latter streets have changed than the corresponding ones at start-time.

3. Like a physics field, said that it is unworthy for you drive a vehicle turn back to the passed node.

4. Periodicity, implies that those similar statuses always present at the same time frame of historical period.

5. We can view each road fork as a node.

Now as the fifth feature, let us select the grid figure to simulate a block in a city. For Periodicity property, it implies that we can forecast the data of traffic status according to historical record in database. Let $P$ be a path with $|P| \geq 1$, an ordered sequence $P = (v_1 \tau v_2, v_2 \tau v_3, \cdots, v_i \tau v_j)$. And we denote the time of arrival on path $P$ by $T(P)$, which node $v_1$ is the start-node. Then we can give a function to express this relationship as follow.

$$T(P) = \sum_{i=0}^{|P|} t(i).$$

The variable $t(i)$ is the pass-time on the $i$th segment on path $P$. Set an assignment function $\omega$ for variable $t(i)$ from database as

$$t(i) = \omega(\tau_i, T(i - 1)) \text{ for } \tau_1 = (u, v).$$

This is an iterative process and starting at $v_1$ on sequence $P$. While a query is sent from the first node $v_1$, system can obtain the initial time-stamp $T(0)$. On the first segment $\tau_1$, it returns the first time-stamp $T(1) = T(0) + \omega(\tau_1, T(0))$, which the value assigned from function $\omega$ to $t(1)$ is the pass-time associated with historical record. By this way, System has the start-time of touring and then to gain offset quantity iteratively, until gaining the overall time stamp on the end of sequence $P$.

Because the pass-time comes from the historical data on server, we can say that it turns out the result centrally presented of the superposition
affected by those known and unknown impacted factors. For a group time quantities $\bigcup_{i=1}^{N} T(P_i)$, the minimum one is the top path naturally.

Now we simplify the traffic problem to searching a top path with min. time-stamp. And we know no retrograding on this hypergraph, which implies that there is not any walk from higher region to low one on the ordered sequence. Therefore we can utilize such orderd structure to filter those useless data, and reduce the search breadth. Thus for BOTS method such that the inner characteristic function $\phi$ should be modifited, that in the leaf-set, those leaves in native or next region are valid nodes for enumerating.

For a length of exploring path $|P|$, having $|P| \leq n$ with respect to graph partition being the set partition of $V$. We denote the length by $L$. Then the time complexity is $O(mL^2B)$ for search a path on instance with BOTS. Hence, the time complexity of overall search is $O(mL^2B)$, by which letter $B$ we denote the search breadth of instance.

4 Breadth of Grid Figure

Based on the 5th feature of Urban Traffic, we selected the grid figure as our simulative object in previous section. Then we need to study the search breadth. Our first type is the $k \times k$ one, where there are $k$ nodes on row or column such that $n = k^2$.

Afterwards we will continue to study other types to find the optimal one. We reserve to denote any grid instance by $k \times m$ with $k \geq m$. For $k \times k$, it just is $m = k$. The search results of $k \times k$ Expt. are listed in following Table 1, which $k \in \{5, 6, 7, 8, 9, 10, 11\}$.

In particular we state those items in the table as the follow for same experiment on different type instance.

1. Column $n$ = shows the $k \times m$ of instance.
2. Column Len. is the length of a shortest path.
3. Column Lo. is the quantity of loop in BOTS.
4. Column P. is the quantity of paths found by BOTS.
5. Column(1)/(2) is of dividing loops by paths, the quotient represents the efficiency of search.
6. Column R.T. is the actual runtime of program

---

1 The tests were executed on one core of an Intel Dual-Core CPU T4400 @ 2.20GHz running Windows 7 Home Premium. The machine is 64bit system type, clocked at 800MHz and has 4.00 GB of RAM memory. The executive program is on console, which compiled by C++11, Code::Blocks 12.11 IDE; [http://www.codeblocks.org](http://www.codeblocks.org)
Table 1: $k \times k$ Expt.

| n= | Len. | Lo.(1) | P.(2) | (1)/(2) | R.T. |
|----|------|--------|-------|---------|------|
| 5  | 8    | 251    | 70    | 3.59    | 5 ms |
| 6  | 10   | 923    | 252   | 3.66    | 22 ms|
| 7  | 12   | 3431   | 924   | 3.71    | 57 ms|
| 8  | 14   | 12869  | 3432  | 3.75    | 655 ms|
| 9  | 16   | 48619  | 12870 | 3.78    | 18 s 160 ms |
| 10 | 18   | 184755 | 48620 | 3.80    | 3 m 18 s 059 ms |
| 11 | 20   | 705431 | 184756| 3.82    | 48 m 91 s 536 ms |

The following Diagram 2 shows that how to calculate the quantity of shortest paths (these shadow parts on diagram show the regions).

Diagram 2

The right diagram shows the fact that the start-node $v_1$ is on the lower left corner and the end-node $v_n$ on the diagonal upper one. They both show the number of regions is $2k - 1$. The diagram shows the $v_n$ coordinates are $(k, k)$. We can say such a fact that the shortest path must contain $k - 1$ segments on $x$ or $y$ axis respectively. Then the quantity of paths is equal to number of combinations $C(L, L/2)$. For the length of path $L = 2k - 2$, this term may be

\[
\text{Breadth} = \binom{2k - 2}{k - 1}.
\]

Obviously, with the data we can approximately evaluate the breadth for our problem, the term is
\[
\binom{2k-2}{k-1} = \frac{(2k-2)!}{(k-1)!(2k-2-(k-1))!}
= (2k-2)!/((k-1)!)^2 \tag{1}
\]

Let \( s = k - 1 \) for Stirling’s approximation \( s! \approx \sqrt{2\pi s} \left( \frac{s}{e} \right)^s \), then fore term\([1]\) having

\[
(2k-2)!/((k-1)!)^2 = (2s)!/(s!)^2
\]

\[
= \frac{\sqrt{2\pi 2s} \left( \frac{2s}{e} \right)^{2s}}{2\pi s \left( \frac{s}{e} \right)^{2s}}
\]

\[
= \frac{(2s)^{2s}}{s^{2s} \sqrt{\pi s}}
\]

\[
= \beta 2^{2s} = \beta 2^{2k-2} \quad \text{for } \beta = \frac{1}{\sqrt{\pi s}} \tag{2}
\]

We can decide the price is expensive. As the term \( C(n, k) = C(n, n-k) \), we can let the \( n = m \cdot k \), and \( m \) be a constant number, then having breadth \( B = \binom{k+m-2}{m-1} \), we can intuitively decide that it can reduce the breadth for term\([1]\). Firstly, we let \( m = 4 \) and \( k >> 4 \), these results in following Table.

**Table 2: m = 4 Grid Expt.**

| k= | length | Loops(1) | Paths(2) | (1)/(2) | runtime |
|----|--------|----------|----------|---------|---------|
| 25 | 27     | 23750    | 2925     | 8.12    | 1 s 399 ms |
| 30 | 32     | 46375    | 4960     | 9.35    | 3 s 591 ms |
| 35 | 37     | 82250    | 7770     | 10.59   | 9 s 565 ms |
| 40 | 42     | 135750   | 11480    | 11.82   | 28 s 034 ms |
| 45 | 47     | 211875   | 16215    | 13.07   | 1 m 0 s 125 ms |

It is clearly for breadth equal to \( C(k + 2, 3) \) that is the term \( \binom{k+m-2}{m-1} \) for
We compare two instances $25 \times 4$ and $10 \times 10$, both $ns$ are equal. We easily observe that the length of path on $25 \times 4$ increases 50% than $10 \times 10$ ones, but for search breadth is 6% of $10 \times 10$ ones. And the runtime of $25 \times 4$ is 0.5%, but the efficiency of search reduces 50%.

We set $m = 3$, and observe the results as follow:

**Table 3: $m = 3$ Grid Expt.**

| k=55 | length | Loops(1) | Paths(2) | (1)/(2) | runtime |
|------|--------|----------|----------|---------|---------|
| 55   | 56     | 30 855   | 1540     | 20.04   | 751 ms  |
| 60   | 61     | 38 710   | 1830     | 21.15   | 2 s 327 ms |
| 65   | 66     | 50 115   | 2145     | 23.36   | 2 s 436 ms |
| 70   | 71     | 62 195   | 2485     | 25.03   | 2 s 837 ms |
| 80   | 81     | 91 880   | 3240     | 28.36   | 6 s 020 ms |
| 90   | 91     | 128 765  | 4095     | 31.44   | 8 s 203 ms |
| 100  | 101    | 176 850  | 5050     | 35.02   | 19 s 606 ms |

It is obviously that the breadth is $C(k + 1, 2)$. On the breadth level, we seem to decide that it is the best solution for reduce the $m$. But it is unequal to the optimal integral solution, we yet need study the time complexity for these figures.

## 5 Time Complexity of Search

**Pure Grid Complexity.** This section we will discuss the problem of the complexity of our objects. Firstly for a unit subgraph $s$ on instance such that the potential of leaf-set $|L(s)| \in \{2, 3, 4\}$. Then we set each potential equal to 4 in the worst case.

Now we discuss the case $n = k^2$, which is $k \times k$ grid figure. For $L = 2k - 2$, such that we can obtain the runtime polynomial for exploring a path[6]:

\[
O(4(2k - 2)^2) = O(16k^2 - 32k + 16) < O(16k^2) \text{ for } k \geq 1.
\]

We may have the term of runtime complexity:
\[O \left( 16k^2 \cdot \beta 2^{2(k-1)} \right) = O \left( \beta \cdot 2^{2k-2+4k^2} \right)
\]
\[= O \left( \beta 2^{2k+2} \right)
\]
\[= O \left( 4\beta n 2^{2\sqrt{n}} \right) \quad \text{for } k = \sqrt{n}.
\]
\[= O \left( 4n^\frac{3}{2} 2^{2\sqrt{n}} \right) \quad \text{for } \beta = \frac{1}{\sqrt{\pi} \gamma} \quad \text{and } s = k - 1.
\]

(3)

Now if \(2^{\sqrt{n}} \approx n^{9/4}\), then we gain a time complexity term \(O(4n^3)\), having

\[
2^{\sqrt{n}} \approx (\sqrt{n})^\frac{9}{2}
\]
\[
2\sqrt{n} \lg 2 \approx 5 \lg \sqrt{n}
\]
\[
2\sqrt{n} \approx 5 \lg \sqrt{n}
\]
\[
2k \approx 5 \lg k \quad \text{for } k = \sqrt{n}.
\]

It is obviously that if \(k \leq 8\), then \(2k \leq 5 \lg k\) having \(2^{\sqrt{n}} \approx n^{9/4}\). Otherwise \(k > 8\), having \(2k > 5 \lg k\). Of course you can note the runtime changing in Table 1.

Second for \(k \times m\) grid figure, with respect to \(L = k + m - 2\), the price of exploring a path by BOTS method can be

\[
O(1(k + m - 2)^2) \leq O(4(k + m)^2)
\]
\[
< O(4(2k)^2) \quad \text{for } k \gg m.
\]
\[
= O((4n/m)^2)
\]

Consider breadth term \(\binom{k+m-2}{m-1}\) for \(m = 4\), we can have the explicit form

\[
\binom{k+4-2}{4-1} = \binom{k+2}{3} = \frac{1}{6} (k^3 + 3k^2 + 2k)
\]
\[
< \frac{1}{6} (6k^3) = k^3 = (\frac{n}{4})^3 \quad \text{for } k = n/4.
\]

Then the time complexity is

\[
O(mL^2B) = O \left( 4 \left( \frac{4n}{4} \right)^2 \cdot \left( \frac{n}{4} \right)^3 \right) = O \left( 0.625n^5 \right).
\]

8
With the same fashion we can have the breadth for \( m = 3 \) is

\[
\binom{k + 3 - 2}{2} = \binom{k + 1}{2} = \frac{1}{2} (k + 1)k = \frac{1}{2} \cdot \frac{3}{2} \cdot \frac{2}{2} (n^2 + 3n) < \frac{2}{9} n^2.
\]

And the time complexity of \( k \times 3 \) is \( O(4n/3)^2 \cdot \frac{2}{5} n^2) = O(1.6n^4) \). Certainly, we can decide the time complexity of search for \( m = 2 \) is \( O(n^3) \). This is obviously that we can not use such pure grid for our solution.

**Composite Grid Network.** For above discussion, we may know there are two impacted factors to affect overall complexity, which are the length of path \( L \) and the inherent breadth \( B \) of instance. They are both relevant to the structure of figure. For above term(3), we can obtain a solution of \( O(n^3) \) if \( k \leq 8 \). Hence, we imagine that while we composite several \( k \leq 8 \) grid figure, we can obtain a optimized solution, no worry about the increasing of quantity of nodes. The diagram we designed is on following.

![Diagram 3](image)

They are two compositing types with two \( 5 \times 5 \) ones. Let node \( v_1 \) is the started-node on both instances. And we set nodes \( v_{46}, v_{44} \) be the end-nodes respectively on left and right figure. It is observed that there are two bridge-nodes on left one, and three bridge-nodes on right instance.

This conclusion is given by counting these nodes in each region after graph partitioning, no by seeing. On left instance, pair \( v_{20}, v_{24} \) are bridge-nodes in \( \sigma_8 \). Similarly, bridge-nodes \( v_{20}, v_{24}, v_{26} \) are in the region \( \sigma_7 \) on right ones. We call region \( \sigma_1 \) bridge-region, for \( |\sigma_{i-1}| \geq |\sigma_i| \leq |\sigma_{i+1}| \). We say that it is a block, which is an ordered subsequence as a component of graph partition. Furthermore for two adjacent blocks, such that the bridge-region is their boundary. Therefore we may separately analyze fore and latter block for above diagram. Then bridge-nodes are the end-nodes on fore block, but for latter they become the start-nodes.

Hence the Combinatorial Number of left instance is
\[2 \cdot \binom{7}{3} \cdot \binom{7}{3} = 2450\]

Right ones is
\[\binom{6}{2} \cdot \binom{6}{2} + \binom{7}{3} \cdot \binom{7}{3} \cdot \binom{6}{2} = 1450\]

We can observe the results both less than \(2n^2\). The lengths of shortest path is 13 and 14 respectively, which both \(L \approx 2\sqrt{n}\). Thus, we seemingly obtain a time complexity \(O(\sqrt{nmn^2})\) optimized solution. Those results of experiment are in following table.

**Table 4: Two 5 x 5 Compositing Expt.**

| Bridges | Length | Loops(1) | Paths(2) | (1)/(2) | Runtime |
|---------|--------|----------|----------|---------|---------|
| 2       | 14     | 8861     | 2450     | 3.62    | 343 ms  |
| 3       | 13     | 5276     | 1450     | 3.64    | 155 ms  |

The both lengths \(L\) close to 8 x 8 figure. But runtime can save more than 50%, and the efficiency of search(loops/breadth) did not reduce. The results encourage us, so that we continue to add the third 5 x 5 on right figure as following diagram.

**Diagram 4**

**Table 5: Three 5 x 5 Compositing Expt.**

| Bridges | Length | Loops(1) | Paths(2) | (1)/(2) | Runtime  |
|---------|--------|----------|----------|---------|----------|
| 3, 3    | 18     | 113051   | 31100    | 3.64    | 1 s 469 ms |

There are 3 blocks on instance for analyzing. For nodes \(v_{20}, v_{24}, v_{26}\) are
in the first bridge-region, and $v_{39}, v_{43}, v_{49}$ are in second ones such that we
need calculate three Combinatorial Number for three stages as follow

\[
\begin{align*}
v_{39} \rightarrow v_{63} & : \binom{3}{4} \binom{6}{2} = 5250 \\
v_{43} \rightarrow v_{63} & : \binom{3}{2} \binom{6}{3} = 7000 \quad (4) \\
v_{49} \rightarrow v_{63} & : \binom{3}{1} \binom{6}{2} = 2100 \\
v_{39} \rightarrow v_{63} & : \binom{3}{5} \binom{6}{2} = 2625 \\
v_{43} \rightarrow v_{63} & : \binom{3}{2} \binom{6}{3} = 7000 \quad (5) \\
v_{49} \rightarrow v_{63} & : \binom{3}{1} \binom{6}{2} = 3150 \\
v_{39} \rightarrow v_{63} & : \binom{3}{5} \binom{6}{2} = 2250 \\
v_{43} \rightarrow v_{63} & : \binom{5}{5} \binom{6}{2} = 1500 \quad (6) \\
v_{49} \rightarrow v_{63} & : \binom{3}{3} \binom{6}{2} = 225 \\
\end{align*}
\]

We may observe that the search breadth is $(4)+(5)+(6) = 31100$, which
less than $2^3 \cdot n^2$ for $n = 63$. Because the length $L$ is $L = 2\sqrt{n} + 2 \approx 18$, then
we compare this figure with $10 \times 10$, which length $L$ is 18 similarly. The
runtime on new figure is only 0.7% to $10 \times 10$.

But while we add the fourth ones, the breadth $B$ increases to 660580
with $n = 82$. We can easily observe that the breadth greater than $82^3$. In
fact, we can have the term of search breadth $n^{N-1}$, which $N$ is the quantity
of composited figure, we call it depth.

Here we seem to forget our target. Our task is looking for a path with
a minimum pass-time. Because there are only bridge-region connecting two
adjacent blocks, therefore we can use the method of divide and conquer to
reduce the complexity, which approach similarly is usual ones in Computer
Science. Then we optimize our strategy like that: for each block, we firstly
find out all shortest paths between start-nodes and end-nodes respectively.
Afterward we alternately find the top path in each block on sequence.

For Diagram 4, we may find out three top paths from $v_1$ to $v_{20}$, $v_{24}$, $v_{26}$
separately and gain 3 top time-stamps on the 3 bridge-nodes in first block.
Then we may gain 3 top paths from those bridge-nodes to $v_{39}$, $v_{43}$, $v_{49}$ with
same fashion. Finally, there are 3 top paths between $v_{39}$, $v_{43}$, $v_{49}$ and target
$v_{63}$, so that we can calculate the top path in stages.

Then the breadth of instance on Diagram 4 is
1. $v_1 \rightarrow \{v_{20}, v_{24}, v_{26}\}$:
   \[
   \left(\frac{7}{3}\right) + \left(\frac{7}{3}\right) + \left(\frac{6}{2}\right) = 35 + 35 + 15 = 85
   \]

2. $\{v_{20}, v_{24}, v_{26}\} \rightarrow \{v_{39}, v_{43}, v_{49}\}$:
   
   \[
   v_{20} \rightarrow \{v_{39}, v_{43}, v_{49}\} = \left(\frac{5}{2}\right) + \left(\frac{5}{2}\right) + \left(\frac{4}{1}\right) = 10 + 10 + 4 = 24 \quad (7)
   \]
   
   \[
   v_{24} \rightarrow \{v_{39}, v_{43}, v_{49}\} = \left(\frac{5}{1}\right) + \left(\frac{5}{2}\right) + \left(\frac{4}{2}\right) = 5 + 10 + 6 = 21 \quad (8)
   \]
   
   \[
   v_{26} \rightarrow \{v_{39}, v_{43}, v_{49}\} = \left(\frac{5}{2}\right) + \left(\frac{5}{1}\right) + \left(\frac{5}{0}\right) = 10 + 5 + 1 = 16 \quad (9)
   \]
   
   Such that having $\square + \square + \square = 61$

3. $\{v_{39}, v_{43}, v_{49}\} \rightarrow v_{63}$:
   \[
   \left(\frac{6}{3}\right) + \left(\frac{6}{3}\right) + \left(\frac{6}{2}\right) = 15 + 20 + 15 = 50
   \]

   We gain the search breadth only $85 + 61 + 50 = 195$, which is approx. equal to $3 \cdot \sqrt{63}$. For each block, having length $L < \sqrt{63}$. While we add a $5 \times 5$ again, the quantity of nodes is $n = 82$ and adding breadth 61, furthermore overall breadth should be $256 \approx 3 \cdot 82$. Because instance adds 19 nodes corresponding to adding a $5 \times 5$, if we have $N \times 5 \times 5$ block on instance, then the search breadth should be $85 + (N - 2) \cdot 61 + 50 = 13 + 61N$ and quantity of nodes is $n = 44 + 19N$. Hence, we can say that is linear growth for breadth $B$ to input $n$. Hence, in the worst case, we can gain the time complexity $O(n^2)$ for search $5 \times 5$ composting grids figure.

**Discussion.** We assume that there exists an instance with $N \times k \times k$ grids composting. Let $n = Nk^2$. According to the term $\square$ of a $k \times k$ breadth, we have the search breadth for each block:

   \[
   \text{Breadth} = \beta 2^{k-2}.
   \]

   As above discussion on **Diagram 4**, set $k \geq 2\alpha \geq 1$ and the final block is $\sigma_j$, we have the term of an exploring path length in each region

   \[
   L_1 = L_j = 2k - 2 - \alpha \quad \text{for } 1 < i < j.
   \]

   for BOTS search in the worst case, we let length $L = 2k - 2 - \alpha$ for a shortest path, then we having
\[
O(4(2k - 2 - \alpha)^2) = O(16k^2 + 4\alpha^2 - 32k - 16\alpha + 16)
< O(20k^2)
= O(5 \cdot 2^2 n/N)
\]

Hence the time complexity polynomial is

\[
T = O(5 \cdot 2^2 n/N \cdot N^2 \alpha^{2k-2})
= O(5^\beta n \cdot 2^{2k})
\]

The term is so puzzled for us to obtain a conclusion. If \( n = 2^{2k} = Nk^2 \) and \( k = 8 \), then we have \( 2^{2k} = 2^{16} \) and \( Nk^2 = N^6 \), such that \( N = 2^{10} \) and we gain a nice time complexity of \( O(n^2) \) for such big data. While let \( N = 1 \) and \( k = 8 \), then having \( 2^{2k} = 2^{16} = 65536 \) and equal to \( 16n^2 \), we have the conclusion as fore one in native section.

This case is reasonable for practice and our experiment. It implies that if \( k \) is a constant number, then the search breadth \( B \) in each block should be less than and equal to a constant \( \beta^{2k-2} \), which is the upper boundary to \( B \). If the \( N \gg k \), then we have \( n \approx k^2 \) and \( n \approx 2^{2k} \). It said that we may solve this problem with the method of divide and conquer perfectly, when we face a big data.

Discussion. On program level, for each block on instance, they actually are those collections of regions, which boundary is bridge-region. Algorithm of graph partition assigns an ordered number to each region, thus program can be easy to find out these blocks. For example, there is an instance \( G = (\sigma_1, \sigma_2, \ldots, \sigma_N) \). Set \( \sigma_i \) is the bridge-region with \( 1 < i < N \). The program can cut instance \( G \) to \( G_1 = \bigcup_{k=1}^{i} \sigma_k \) and \( G_2 = \bigcup_{j=i}^{N} \sigma_j \). For each unit subgraph root \( v_t \in \sigma_i \) and \( v_t \in R(s_t) \), there is

\[
L(s_t) = L(s_t)_{\sigma_{i-1}} \cup L(s_t)_{\sigma_i} \cup L(s_t)_{\sigma_{i+1}}.
\]

However consider \((R(s_t) \times L(s_t)_{\sigma_i}) \in G_1\) or \((R(s_t) \times L(s_t)_{\sigma_i}) \in G_2\), these blocks actually may be the partition of set \( \tau \). Therefore we understand if there are \( N \) machines search the paths with \( N \) blocks separately, they may not interfere to each other. It said that really implements the separate Parallel Computing to solve a common problem.

6 Automated Traffic Management

Now we discuss how to build the model of Automated Traffic Management. With the Greedy method, we can decide this fact: each instance being
optimal then overall being too. Namely, if each driver queries the managing system and obtains a top path, then system can optimize the whole Urban Traffic.

To round this idea, we certainly pose the $C-S$ model of distributed computation, in which there are 3 abstract function modules as follow:

1. Client: Terminal device in a car, sends the query with start-node and end-node, afterward waiting for info of top path.

2. Server: Implement approaches including storage of historical data, exploring top path for Client.

3. Accessory Equipment: collecting data sample from streets in real-time; updating traffic data and etc.

We describe these relationships among those modules by using following diagram:

On level of data, there is a problem for big data. Say that a city is a big grid figure with a larger number $n$, but above solution requires $k$ is a small number. There is the possible for a large number nodes between source and target. Then we need a method to clip the big figure, i.e. need clipping geographic information. Of course, the problem can be solved by using the $k \times k$ grid frame to move on certain direction. Here author does not discuss this problem, because it is complex for various cities or requirements.

Second, it is a fact for client that the system should possess the ability, which it can deal with the traffic emergency. It said that there exists the possible for certain accident happened at the few segment on top path, and more other streets in jam at once. Then system should immediately renew to find out a new top path for each client. This time, system can decide the sudden street as a disconnected path, like no incident path between associated nodes, and use the real-time data from Accessory Equipment to re-compute on new relation schema of nodes.
Third, the second note that if in the time frame of traffic jam, server can delete those adjacent relation in table, on which the Expedite Index is approximately to pathological. So that only there is rest resources for client in real-time. That is conform to Greedy Method. Then system possesses these features of flexible and real-time. If system can control the quantity of traffic for a street by statistics of cars, then system will possess dynamic management.

**Summary.** We abstracted those features from practical traffic problem on observation level. Further more we produced a concept *pass-time* to simplify the complex and dynamic system to searching timing sequences. For these sequences, we selected a type of figure as research object for simulating. With graph theory and complex theory, we discussed the strategy for searching and building model on theoretical level. Finally, we gave out the conclusion from calculating combinatorial number and experiment.

Although the two algorithms are Heuristic Method, they still complete several tasks: exploring each shortest path on instance; filtering useless data; finding out the bridge-nodes and partitioning those regions by the abstracted sequence: block $\sim$ region $\sim$ $\tau$. Consequently we can utilize these properties to build a mathematical model of Automated Traffic Management, computer those big data, optimize our strategy and figure objects for solving problem in polynomial time.

On industry level, the traffic data can be public resources for public transportation enterprise, such as driverless taxi. The robot can driver car on the top path given by server.

**Future Work.** For those above contents we discussed, it actually is an instance respecting to solving a problem of dynamic system. This class of problem belongs to decivion problem in Decision Science and Operations Research, that how to optimize strategy and find a top decision. Based on abstract relationship among those factors and decision nodes, the process of decivion can be regarded as a decivion stream flowing from premise start-node to the end-node of conclusion. In most case, those problems present that each mapping relation between two adjacent nodes always depends on the previous output on an ordered sequence. And the final results are not equal possibly for two different sequences.

As the diagram 2, we shew how to put a figure in a 2D coordinate system, such that we can give the combinatorial formula to evaluate the breadth. In natural case, no an urban layout is a pure grid figure. It is unfortunately for more case that there are only less practical models like that. Even we can not give any combinatorial term as on diagram 1.

This is a fact for us that there is not a universal coordinate system as reference system for any figure, although human try to find it for more than a century. Therefore we have not a uniform mathematical method to solve
this NP problem. Then the feature of universal for BOTS naturally is the powerful tool with heuristic exploring. At least we can utilize experiment to study figure model better than helplessness. In particular, graph partition gives us a rough reference sequence for any $r$-regular instance.

Hence this paper introduces such a research instance for practice, industry and society, you can think how to analyze a dynamic system with various tools.

References

[1] Tan, Y. (2013), 'Construct Graph Logic.’, CoRR abs/1312.2209.

[2] Bellman, Richard (1958). "On a routing problem”. Quarterly of Applied Mathematics 16: p. 8790.

[3] Tan, Y. (2013), 'Construct Graph Logic.’, CoRR abs/1312.2209: p. 29.

[4] Tan, Y. (2013), 'Construct Graph Logic.’, CoRR abs/1312.2209: p. 31-32.

[5] Tan, Y. (2013), 'Construct Graph Logic.’, CoRR abs/1312.2209: p. 28.

[6] Tan, Y. (2013), 'Construct Graph Logic.’, CoRR abs/1312.2209: p. 13-14.

[7] Nemes, Gerg (2010), "On the Coefficients of the Asymptotic Expansion of n!", Journal of Integer Sequences 13 (6): 5 pp.