Supporting Asynchronous Parallel Computation in a Large Graph

Md. Rizwan Parvez
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000, Bangladesh
rizwan.incipient@gmail.com

Turash Mosharraf
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000, Bangladesh
mturash@yahoo.com

Sarana Yi Nutanong
Department of Computer Science and Engineering
City University of Hong Kong (CUHK)
Hong Kong
snutanon@cityu.edu.hk

Mohammed Eun
Department of Com
Science and Engine
Bangladesh Univers
Engineering and Tech
(BUET)
Dhaka 1000, Bangl
eunus@cse.buet.

ABSTRACT

In our day to day life, voronoi diagram with k nearest generators has several applications. For example, while driving a car, the driver might be interested to know the location of 3 nearest gas stations. There are several methods to efficiently solve this problem when the graph is in one computer. However when the graph is too large to keep in a single computer the computation becomes more complex. Road networks in real world are usually very large with millions of nodes and they must be stored in a distributive environment.To solve this problem we not only consider the computational complexity but also message passing and synchronization among different process running in different machines. Several existing models can be used to solve this problem but they does not support enough degree of parllelism. We developed a system that works asynchronously and effectively use high degree of parallelism to find k nearest generators from n total generators in a very large graph within a reasonable time. We tested the performance as well as correctness of our model on USA road network on 23.6 million nodes and 1.6 million gas stationsas generators and found a signicant improvement compared to existing models.

Keywords

1. INTRODUCTION

Continuous queries in road networks have gained significant research interests due to advances in Geographic Information System (GIS) and mobile computing. Advances in GIS and mobile computing have led to research interests in continuous spatial queries in road networks. Single Source Shortest Path (SSSP) and Multiple Source Shortest Path (MSSP) are widely used and have wide range of applications in real life. While both problems can be solved by simple Dijkstra algorithm, more complex procedure is required when they are applied in distributed environment. Unfortunately in real life graphs are often distributed in several machines. Running any of the above algorithm requires message passing between processes. This cause the process to 1)wait for message 2)Do its job on its own part of the graph 3)generate further messages for other processes 4)Send messages. Efficiency of one procedure depends on minimizing delay on step 1 and 4.

The rapid growth in the volume of any real world graph(road network, web, social network) has lead to the development of various vertex centric and block centric distributed graph computing systems in recent years. However real world graphs from different domains have very different characteristics.

balancing the workload is a crucial part when designing graph computing system. graph based systems undergoes a lot of effort to balance workload. If load is not balanced properly between processes some process will have a lot more work than others which will cause it to take longer than others to complete its computation step. Resulting in overall performance degradation. Moreover, all existing systems use intra process parallelism but not inter process parallelism. Processes use sequential computing within itself. So in case of assigning a process a large workload will definitely lower its performance.

In order to balance workload existing systems tries to assign almost equal number of vertices and edges to one process. The main problem is, load distribution not only depends on number of vertices and edges, but also incoming messages which varies according to applications. Depending on application a vertex with many incoming edge can have fewer incoming messages and vice versa. Therefore same vertex distribution may lead to higher performance in one application whereas poor performance in another. There is no application independent global load balancing scheme that can ensure perfect load balancing. It is natural that all the processes will not be able to terminate their works at the same time. Hence it may be wiser to introduce asynchronous parallel computations rather than trying to finish all the processes at the same time or as a batch.

We developed a system that uses both inter process and intra process parallelism for finding k nearest generator in distributed system. for this purpose we used k threads, one for a subset of expansion. Each process consists k number of expansion threads and these threads depends only on the similar thread from other processes. In this way dependancy is shifted from process level to thread level. As threads have lighter workload, the delay is much less.

The delay is further minimized by stopping each expansion earlier when necessary. As we only need k nearest generator, we dont need one expansion to cover full graph. the expansion is cut down in one direction as soon as it proves to be inferior than any k expansions to some node .

We ran several experiments on USA road networks and found out that our system significantly outperforms Blogel. It takes same round of computation as Blogel, but each round is much faster.

The one problem remains in heavy message passing. Graphs like social media, web graphs have very high density and a very high degree of messages are exchanged between neighbours. In our system each thread in all processes exchandes messages resulting in a very high amount of messages. Moreover, to distinguish messages from one thread to another each massage has to contain id of its sender and reciever thread. We developed a technique for this task which will be discussed later. However we found out that there are many messages that can be safely ommitted or combined before sending, this preprocessing step can be done within process by one separate thread. It can significantly reduce unnecessary messages without causing additional delay.

In further section:

2. RELATED WORKS

Due to the growing need to deal with massive graphs in various graph analytic and graph mining applications, many distributed graph computing systems have emerged in recent years, including Pregel, GraphLab, PowerGraph, Giraph, GPS, and Mizan and Blogel. Apart from Blogel, all of these systems adopt the vertex-centric model proposed in, which promotes the philosophy of âĂIJthinking like a vertexâĂİ that makes the design of distributed graph algorithms more natural and easier. However, the vertex-centric model has largely ignored the characteristics of real-world graphs in its design and can hence suffer from severe performance problems. Blogel investigated a broad spectrum of real-world graphs and promoted the philosophy of âĂIJthinking like a block of vertexâĂİ. It has outperformed all of previously stated systems. However. It does not support Inter worker parallelism.

3. METHODOLOGY

3.1 Overview

In real life finding the nearest points of intersest (POI) has become a day to day query. Everybody wants to know the the nearest two or more restaurants among which he could choose. In case of emergencies one may want to find out the nearby hospitals or police stations. A travellers may want to know the nearby airports or gas stations etc. Based on these observations, queries are often found to find out the best or nearest k out of all n POIs. But often in case of large topologis or graphs executing SSSP or MSSP become quite time consuming and a lot of computations are also needed. Where the graph is large enough that distributed environments are used to store and process them, the processing workers pass a large number of meassages to communicate among them. And if $k < n$ only few of the generators or sources outperform the rests. Therefore a signigicant amount of expansions or computaions are indeed unnecessary which we can avoid. Our methodology introduces an effective and efficient way of finding nearest k generators out of n with a reduced number of message passing and prunning of unnecessay expansions.

As an instance of a large graph we choose the road network of USA. We store the graph in several computers using the distributed file system of hadoop HDFS. As vertex based computaion has work load unballance problem, we use blockwise computation. In order to partition the graph into blocks we use the voronoi partitioning approach of BLOGEL framework. We allocate these blocks among the workers afterwards. In oreder to process further (e.g: finding the closest genrators by MSSP) we integrate the BLOGEL framework with ours. BLOGEL computes parallely at most at worker level but our approach introduces thread level parallel computation even within these workers. In order to find the k nearest generators we keep a table of k entries which stores the result. Actually we treat the MSSP as a composite of single SSSPs. For each SSSP expansion from each generator we alloacte a thread within the worker. It is quite common that the number of generators n is very much smaller than k. As each expansion is now different they donot need to synchronous anymore. Each parallel exapansion computes now asynchronously and hence could be at different level of computation (i.e: superstep). If the number of generators are also pretty large, we can not create a thread per generator. In that case we allocate each thread to number of generators. Each thread then acts like a MSSP expansion. In case of $k < n$ some expansions actually fails to update the table of best k entries. The boundaries, beyond which they fails to update the table and cannot keep the next vertexes in $ACTIVE$ computation mode, are the limit of these expansion. We terminate or drop these expansion beyond these boundaries as they cost unnecessary computation and message passing beyond the boundaries.

3.2 Probability Calculation Model

There are lots of factors that are needed to consider for calculating the probability of a crime. Some notable factors are space, time, crime type, population of the place, per household annual expenditure on personal care, insurance, pension, monthly income, distance from the critical boarder, drug dealing centers, other social information and so on. In a third world country like us, most of these information are not available. For Dhaka city, we have collected the crime records of the month june 2013 to May 2014 from Dhaka Metropolitan Police Database. But only a few information about the crime were available. Information, we found in the record, includes the followings.

- Crime that occured.
- Location of Crime in Lattitude and Longutitude.
- Date of occurance.
- Time.

With these information, our model calculates the probability from four perspectives.

- Space.
- Time.

- Day of week.
- Crime type.

Police assumes that criminals do crimes when they get chance. In reality, at a very same spot, any crime can occur at anytime. Therefore, our model consider these four factors to be independent. To formally describe our model we denote the probability of occurring crime by Equation 1

$$P_{<Space,Time,Day,Type>} = P_{Space} * P_{Time} * P_{Day} * P_{Type} \quad (1)$$

3.3 Probability Calculation For Space

Our model divides the entire Dhaka city into m * n grids. Any crime that occurred in a grid has also an effect on its surrounding grids. Therefore we add a fraction of the total number of occurrence in a grids to its neighbors. This effect is limited to 3 layers. Layer1 consists of itself only. Layer2 consists of the closest 8 grids and Layer3 consists of 16 surrounding grids of Layer2. Beyond them, the effect can be neglected.

Table 1: Layers in griding system

	L3	L3	L3	L3	L3	
	L3	L2	L2	L2	L3	
	L3	L2	L1	L2	L3	
	L3	L2	L2	L2	L3	
	L3	L3	L3	L3	L3	

For any crime incident we update the count of the occurring grid itself, Layer1 using impact factor α as followed

$$count_{grid} = count_{grid} + \alpha \quad (2)$$

along with the Layer2 grids as following

$$count_{grid} = count_{grid} + \beta \quad (3)$$

and for Layer3 grids as followed

$$count_{grid} = count_{grid} + \gamma \quad (4)$$

But from our observation, any crime incident occurred recently has more impact than a older one. To calculate the probability of occurring a crime on a particular date we assign a fixed priority to every crime incident that occurred previously. This priority follows exponential distribution which decreases for every μ days. The number of days how long before a crime occurred from the day of query is

$$days\_difference = (DATEDIFF(date\_of\_prediction, date\_of\_occurrance)) \quad (5)$$

and using this we calculate the priority of any crime that has already been occurred assuming that the impact changes for every 2 weeks. Therefore, with $\mu = 15$, we get

$$priority = \frac{1}{floor(days\_difference/\mu) + 1} \quad (6)$$

Hence, now Equation 2 for Layer1 grids is modified to the following formula

$$count_{grid} = count_{grid} + \alpha * priority \quad (7)$$

and Equation 3 for Layer2 grids is modified to the following formula

$$count_{grid} = count_{grid} + \beta * priority \quad (8)$$

and Equation 4 for Layer3 grids is modified to the following formula

$$count_{grid} = count_{grid} + \gamma * priority \quad (9)$$

Therefore the probability of occurring a crime in any grid becomes

$$P_{Space} = P_{grid} = \frac{count_{grid}}{\sum count_{grid}} \quad (10)$$

3.4 Probability Calculation For Time

We have divided a day into 6 time periods. These 6 time periods are enlisted bellow.

- From 5.00 a.m to 7:59 a.m into dawn
- From 8.00 a.m to 9:59 a.m into morning
- From 10.00 a.m to 15:59 a.m into working period
- From 16.00 a.m to 18:59 a.m into afternoon
- From 19.00 a.m to 22:59 a.m into evening and early night
- From 23.00 a.m to 4:59 a.m into late night

As we discussed in earlier section about the impact of a crime incident of a grid on its neighbor grid, one time period has effect on its neighbors namely its previous one and next one. Therefore, for every crime incident occurred in a particular time period, we calculate the no of crimes occurred in that time period with an impact factor δ using the following formula

$$count_{time\_period} = count_{time\_period} + \delta * priority \quad (11)$$

where priority term is the very same as described in earlier section. we assumed that the occurrence of this crime has same impact on its previous and next time period. If σ denotes this impact factor the counting formula for the previous and next time period becomes

$$count_{time\_period} = count_{time\_period} + \sigma * priority \quad (12)$$

Accordingly the probability of occurring a crime in any time period becomes

$$P_{Time} = P_{time\_period} = \frac{count_{time\_period}}{\sum count_{time\_period}} \quad (13)$$

3.5 Probability Calculation For Day and Crime Type

To improve the prediction accuracy of our model we unified the similar types of crimes into one. i. e. Decoity, Robbery, Burglary, Pilferage into one category Hijack, Female and child Kidnapping into Kidnapping. Now Regardless of the previously described two factors space and time, day and crime do not have effect on others. We divided the crime database in groups based on priority and we calculate probability for each group.

For any group with priority i, the probability of a crime to occur in a particular day is

$$P_{<Day,i>} = \frac{number\ of\ crimes\ occurred\ in\ Day\ with\ priority\ i}{\sum number\ of\ crimes\ occurred\ with\ priority\ i} \quad (14)$$

And we take the weighted average of these probabilities to calculate the probability P_Day

$$P_{Day} = \frac{\sum P_{<Day,i>} * i}{\sum i} \quad (15)$$

Similarly, for any group with priority i, the probability of a crime of a particular type is

$$P_{<Type,i>} = \frac{number\ of\ crimes\ occured\ of\ Type\ with\ priority\ i}{\sum number\ of\ crimes\ occurred\ with\ priority\ i} \quad (16)$$

and

$$P_{Type} = \frac{\sum P_{<Type,i>} * i}{\sum i} \quad (17)$$

Thus the total probability is calculated.

$$\begin{aligned} P_{<Space,Time,Day,Type>} &= P_{Space} * P_{Time} * P_{Day} * P_{Type} \\ &= \frac{count_{grid}}{\sum count_{grid}} * \frac{count_{time\_period}}{\sum count_{time\_period}} * \frac{\sum P_{<Day,i>} * i}{\sum i} * \frac{\sum P_{<Type,i>} * i}{\sum i} \end{aligned}$$

4. RELATIVE PROBABILITY

We have m * n grids, p time periods of a day, 7 days of week, t types of crimes. Therefore, in case of uniform distribution the average probability of occurring a crime of any type is $\frac{1}{t}$, in any grid is $\frac{1}{m*n}$, in any time period is $\frac{1}{p}$, in any day is $\frac{1}{7}$.

If the calculated probability

$$\begin{aligned} P_{<Space,Time,Day,Type>} &> P_{Avg_Space} * P_{Avg_Time} * P_{Avg_Day} * P_{Avg_Type} \\ &= \frac{1}{m * n * 7 * p * t} \end{aligned}$$

then we can say it is more likely to occur than average one.

5. OBSERVATION

As the Layer3 grids are at a distance from Layer2 grids which is same as the distance between Layer1 and Layer2 grid. Therefore, Our first guess was to test our model with $\gamma = 1$ $\beta = 10$ and $\alpha = 100$. Taking such values the true positive is 73.6%, and true negative is 66.7%.

Then we check it for $\gamma = 1$ $\beta = 5$ and $\alpha = 25$ and got the output which gives true positive 78.9%, and true negative 57.14%.

With $\gamma = 1$ $\beta = 3$ and $\alpha = 9$ we found the true positive is 63.9%, and true negative is 56.8%.

With $\gamma = 1$ $\beta = 2$ and $\alpha = 4$ we found the output which results in true positive is 68.1%, and true negative is 58.7%.

After that, setting different values for α other than β^2 we got better result. Among them with $\gamma = 1$ $\beta = 3$ and $\alpha = 5$ we got the best one. It gives true positive 79.2%, and true negative 67.14%.

After observing the what should be the impact factor of a crime on its own grid, and surroundings, we also varied the impact factor δ, and σ. So far we have observe the results setting $\delta = 5$, and $\sigma = 1$. The following is the result we get by setting δ=5, and σ=2. But it gives us more false positive result. The reason for such result is now we give more impact in some time period where the crime was not actually occurred but it is previous or next time period of the actual. It gives true positive 68.4%, and true negative 64.1%.

Then we set $\delta = 7$ and keep σ unchanged to value 2 to give more impact on the time period in what the crime actually occurred, but no more than the first one where δ=5, and σ=1 which gives better result. It gives true positive 79.24%, and true negative 68.2%.

We have also tested the result with δ=4, and σ=1 but its outcome is nearly same as with δ=5, and σ=1 but worse.

Then we try to vary the grid size. So far we have showed the results with grid size around 600 m * 600 m (i.e., dividing the whole Dhaka city into 1024 grids). We now divide Dhaka into smaller grids to check if gives us better result or not. But we find that more smaller grid does not bring out better result. We find that the places exactly or very nearby of where previous crime occurred become highly probable but actually crime occur some place not too distant nor too close. So the previous one (1024 grids) gives the better result. With 4096(64 * 64) grids our model gives true positive 74.8%, and true negative 51.6%.

Table 2: Different Values of Factors and Corresponding Results

α	β	γ	δ	σ	m	n	true positive	true negative
100	10	1	5	1	32	32	73.6%	66.7%
25	5	1	5	1	32	32	78.3%	57.14%
9	3	1	5	1	32	32	63.9%	56.8%
4	2	1	5	1	32	32	68.1%	58.7%
5	3	1	5	1	32	32	79.2%	67.14%
5	3	1	5	2	32	32	68.4%	64.1%
5	3	1	7	2	32	32	79.24%	68.2%
5	3	1	4	1	32	32	78.9%	66.56%
5	3	1	7	2	64	64	74.8%	51.6%

Table 3: Different Values of Factors and Default values

Factor	Experimented Values	Default value
α	100, 25, 9, 5, 4	5
β	10, 5, 3, 2	3
γ	1	1
δ	7, 5, 4	7
σ	2, 1	2
m	64, 32	32
n	64, 32	32

6. FINDINGS

We have checked with different grid sizes. Neither too small nor too large grid brings out better result. From our experimental results we got the best prediction with around 600m * 600m grid size (i.e., dividing Dhaka into 32 * 32 = 1024 grids). Impact factors of a crime on the grids suits more with these values $\alpha = 5$, $\beta = 3$, $\gamma = 1$ or maintaining this ratio. Impact factors of a crime over the time periods suits more with these values $\sigma = 2$, $\delta = 7$ or maintaining this ratio. With these values, Our model results in true positive 79.24%, and true negative 68.2%.

7. CONCLUSION

In this paper, we have described a spatio-temporal prediction model for crime and evaluated it on real data from Dhaka Metropolitan Police. The pro- posed model is shown to be sufficiently effective when large amount of data is available and can be adapted even when little data is available. Some important characteristics of this approach include:

- Probabibility is measered from different perspective.
- All these probabilities are considered independent.
- priority is distributed exponentially. Recent data are more prioritized.
- This model would predict better if more data were available.

Our prediction modeling can be applied in smart phone applications to help general people to avoid crime as well as the law enforcement agency to take appropriate action.

8. REFERENCES