MALOS: A Movement-Aware Location Selection System

Di Yang, Hui Li
yanw.d@foxmail.com,hli@xidian.edu.cn
School of Cyber Engineering, State Key Laboratory of Integrated Services Networks
Xidian University
Xi'an, China

Meng Wang
School of Computer Science
Xi'an Polytechnic University
Xi'an, China
wamengit@sina.com

Dan Li, Jiangtao Cui
i.danli@outlook.com,cuijt@xidian.edu.cn
School of Computer Science and Technology
Xidian University
Xi'an, China

ABSTRACT

Facility placement has been receiving considerable research attention due to the proliferation of GPS equipped mobile devices. Establishing new facilities has a wide spectrum of applications including billboard placement, wildlife monitoring, supermarket/restaurant placement, etc. In all these applications, finding the best position to place a facility can lead to the optimal benefit given plenty of target users, which are associated with many historical position logs, e.g., user check-in data, wildlife monitor logs. In this work, we demonstrate a general system, namely MALOS, for answering the majority of facility placement problems given movement logs of a series of target users. Specifically, MALOS leverages on three academic works in the following scenarios. Find the best one/k position/s to place one/k new facilities in order that the overall benefit can be maximized; find one position to place an extra facility given that there are k existing facilities in different places, such that the overall benefit can be maximized. Notably, in MALOS we consider only the geographical issues that are common in all facility placement applications and choose not to take into account the specialized factors that vary across applications, such as billboard size, restaurant type, etc. We shall provide an opportunity for demonstration users to experience all kinds of these facility placement queries with a real-world historical movement logs and corresponding maps interface.

1 INTRODUCTION

Facility placement has been receiving considerable research attention due to the proliferation of GPS equipped mobile devices. Establishing new facilities has a wide spectrum of applications, for example, setting up billboards, monitoring wildlife, running restaurants, building charge stations and urban planning. Majorities of existing research work focus on solving a specific problem, such as setting up billboards or running restaurants, etc. If we change the application, these researches cannot be applied directly to the new one. Therefore, we are motivated to design a general system to address facility placement problem, namely Movement-Aware LOcation Selection system (MALOS).

Compared with the maximum coverage problem of existing work [7], the advantages of MALOS are as follows. Firstly, as a ready-to-use integrated system, instead of loading offline dataset as input, MALOS incorporates a data acquisition module that collects historical check-in data from the Internet and is updated in real-time; secondly, [7] can only be applied in outdoor advertising applications but fails to fit in other applications. In comparison, we take into account the common factor of different facility placement applications, i.e., distance, and allows user-configuration for other factors specialized for particular scenarios. Thirdly, MALOS is a general system that supports up to three different use cases, while [7] considers only one of them.

MALOS is a *general* movement-aware facility placement system, where *general* can be interpreted as follows: 1) We generalize the common spatial-correlated issues within facility placement applications, and avoid falling into application-dependent factors, *e.g.,* size and price of billboard, categories of restaurant and etc., which can be extremely different across applications. 2) We consider the following representative scenarios in facility placement, given the historical movement logs for a large scale of users.

- *Place-One*: query the optimal location which can influence the maximum moving objects [6].
- *Place-k*: return k locations which can influence the largest number of moving objects [4].
- *Incremental-One*: add a location into the existing locations set which could provide the best marginal [1].

Our key contributions in this work are summarized as follows.

- We develop a general system based on user movement data, namely MALOS, answering general facility placement queries with user-friendly browser-based interface.
- The system provides three different queries to meet the major requirements for facility placement applications.
- We provide a module to further incorporate check-ins data for each moving object in real-time.

The rest of the demonstration proposal is organized as follows. Section 2 formally defines the specific queries supported by MALOS. Then, we introduce MALOS system in Section 3. Section 4 offers the demonstration details. Finally, Section 5 concludes the demonstration.

2 DEFINITION OF QUERIES

In this section, we will introduce the definitions of three queries mentioned before.

Definition 2.1 (Place-One). Given a set of candidate locations C, a set of moving objects Ω, a certain distance-based probability function PF and a user-specified influence threshold τ, the *Place-One* query aims to find the optimal candidate $c \in C$ such that $\forall c' \in C - \{c\}, inf(c) \geq inf(c')$, where $inf(c)$ is the number of moving objects in Ω that are influenced by c [6].

Definition 2.2 (Place-k). Given a set of candidate locations $C = \{c_1, c_2, ..., c_n\}$, a set of moving objects Ω and the budget number of new facilities $k(k \leq n)$. *Place-k* aims to find $\exists S \subseteq C$ to maximize $\sigma(S)$. $\sigma(S)$ denotes the total number of moving objects that are influenced by candidate set S [4].

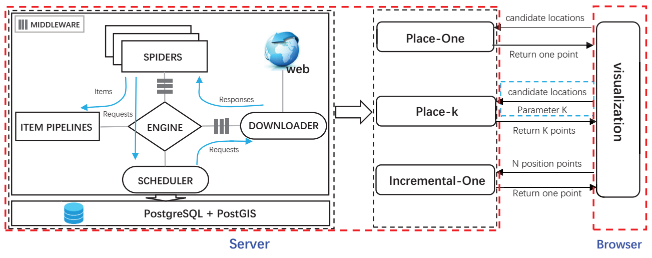


Figure 1: Architecture of MALOS.

Definition 2.3 (Incremental-One). Given a set of candidate locations C, a spatial network graph $G(V, E)$, a set of existing facilities F and a set of moving objects Ω, each of whose movement can be modeled as a set of reference locations, *Incremental-One* aims to find the optimal location c among a set C of query candidates such that $\forall c' \in C, \Delta(c) \geq \Delta(c')$, where $\Delta(c)$ is expected reduction of total distance for c [1].

In particular, in our default settings, we follow the same suggestions from original corresponding academic works for different queries. For instance, in *Place-One*, the probability of a user checking-in at a point-of-interest decays as the power-law of the distance between them. We set the distance-based probability function PF as $\rho(d_0 + dist(c, p))^{-\lambda}$ by default as suggested in [6], where ρ is a factor to describe behavior pattern, d_0 is a distance factor, $dist(c, p)$ is the distance between candidate c and position of moving object p, λ is a exponential factor to configure the mapping from distance to influence probability. In MALOS system, the default values of the number of candidates, probability threshold τ, behavior factor ρ, and exponential factor λ are 600, 0.7, 0.9, and 1.0, respectively. In fact, it can also be manually adapted to other functions by MALOS users.

3 MALOS PROTYPE

In this section, we cover the framework of MALOS, the acquisition of check-in data, modeling of check-in data in database and interaction with data.

3.1 The MALOS architecture

MALOS adopts the browser-server model and is implemented with JavaScript and Java, built on top of PostgreSQL. MALOS implements three algorithms: PINOCCHIO algorithm [6], GreedyPS algorithm [4] and Local Network Based (LNB) algorithm [1]. However, MALOS is not a simple aggregation of these three algorithms, which consider only how to address particular use cases given the data sources. Instead, as a real-time and ready-to-use system, MALOS has to additionally address two other grand challenges, 1) where and how to get the data these algorithms rely on; 2) how to store the data such that these algorithms can be efficiently carried out. Our solutions towards both challenges are discussed in detail within Section 3.2 and 3.3, respectively.

The architecture of MALOS is shown in Figure 1, which includes data acquisition, query processing and visualization modules. The data acquisition module obtains the historical position logs from third-party resources (e.g., Sina Weibo, Twitter). In order to efficiently crawl data from the web, we use the Scrapy architecture, which can efficiently obtain historical location check-in data to ensure real-time performance [5]. Users submit query requests through the browser, which executes a particular facility placement algorithm (*Place-One*, *Place-k* or *Incremental-One*) based on the request. Finally, the results are returned to the users, and are displayed over a Map interface (*e.g.,* Google Maps, Baidu Map) in the browser.

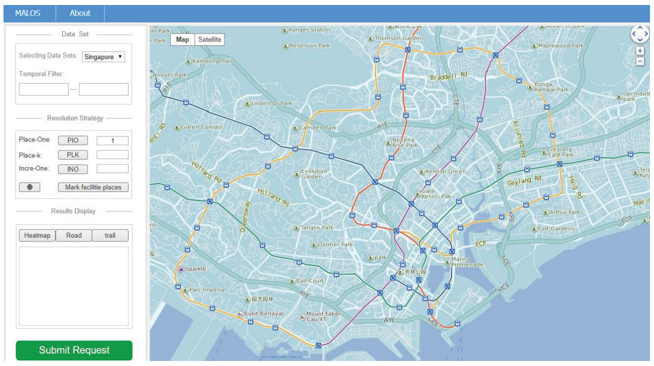
3.2 Data acquisition

How to obtain moving users' historical position data is a first challenge for MALOS. At present, there are two direct ways to address it, one is to be authorized to obtain real-time data directly from giant LBS (location-based service) enterprises (e.g., Uber, Google), the other is to cooperate with government departments. However, neither of them is easy to be carried out for ordinary companies. In this end, MALOS system aims to provide a friendly usage for general public users, who are unable to get the data through the ways discussed above. Therefore, we developed a general web crawler to obtain the historical locations of people through third-party services.

In MALOS system, we use Scrapy framework [5] to collect data. Scrapy is an excellent open source framework for quick crawling of web sites and extracting structured data. There are mainly seven components in Scrapy. Scrapy *Engine* is responsible for controlling the data flow between all components of the system. *Scheduler* receives requests from the engine and enqueues them for further usage by the engine. *Downloader* is responsible for fetching web pages and feeding them to the engine which, in turn, feeds them to the spiders. *Spiders* are custom classes written by us to parse responses and extract items from them or additional URLs to follow. *Item Pipeline* is responsible for processing items once they have been extracted by the spiders. *Downloader Middlewares* are specific hooks that sit between the Engine and the Downloader; and process requests (resp., responses) when they pass from the Engine to the Downloader (resp., Downloader to the Engine). *Spider Middlewares* are specific hooks that sit between the Engine and the Spiders; they are able to process spider input (responses) and output (items and requests). The framework also provides a convenient mechanism for extending Scrapy functionality by plugging custom code.

We present a feasible crawling strategy for MALOS (By default, the crawling site of the MALOS system is Weibo). First, we select one (or more) seed users in a region as the initial target, which is determined by the geographic location label in the personal information list. For each crawled object, we grab personal information in turn, locations of historical check-ins in all Weibo pages, as well as fans list and follower list. Then, add the two lists in the upper level watchlist to the crawl object list. Following this iteration, the reptile radiation can be formed, which takes the seed user as the core and diffuses outward layer by layer. In fact, there are a lot of records with noise information, such as some vulgar marketing numbers. Therefore, we need to set a threshold value for the captured object to determine whether the user is a target user. In MALOS, we parameterize a target moving user as follows: the number of message < 5000, fans < 5000, follower $<$ 5000.

Given the above strategy, Scrapy in MALOS works as follows (illustrated in the left part of Figure 1). When the web *Spider* starts, it will extract each URL from the start URL of a seed user and encapsulate it into a request, which will be sent to the *Engine*. Then the requests are passed over to the *Scheduler* and form a queue. The first item in the queue will be sent to *Downloader* for retrieving the corresponding content. The response file returned from the *Downloader* will be handed over to a parse method. Finally, the parse method calls the a predefined XPath to extract the check-in data from the crawled page content.



Data Set

Selecting Data Sets: Singapore

Temporal Filter:

Resolution Strategy

Place-One: PIO 1

Place-k: PLK

Incre-One: INO

Results Display

Figure 2: The browser interface of MALOS.

In order to meet our requirements, we customize these components by defining some parameters as followings.

ROBOTSTXT_OBEY = False
DOWNLOAD_DELAY = 0.1
CONCURRENT_REQUESTS = 16

3.3 Storage of check-in data

The check-in data reflects the spatio-temporal behavior of the moving object. It generally includes user ID, check-in location, timestamp, check-in area, etc. The check-in location sequence with time mark constitutes the user's historical positions logs. Based on the check-in records, we are able to construct historical positions logs for each individual Weibo user we have crawled. In particular, in PostgreSQL we present the following schema for user historical positions logs.

CI_PAIR (USER_ID varchar(40), LAT real, LNG real, CI_Time timestamp, CITY varchar(256));

Each CI_PAIR object stores the position of a moving user at a particular timestamp. Multiple records of a user are combined to form a spatial-temporal movement history.

We store the dataset in PostgreSQL (version 11.6) extended by PostGIS[1]. In addition to using the embedded distance calculation function ST_Distance ($geometry\ g1, geometry\ g2$), MALOS system extends the integration of PLACEONE($dataset$), PLACEK($dataset, parameter\_k$) and INCREONE ($dataset$, $parameter\_k$) functions into the PostgreSQL database. The extensions are basically customized functions that define specific query processing algorithms. As a mild coupling external module, the extensions can be easily distributed and maintained.

3.4 Usage of MALOS in facility placement

In MALOS system, data operations are initiated by users from browser side. The browser side offers two panels, the function panel and the map panel [2]. The function panel provides the interfaces to users for obtaining check-in data and generating queries. The map panel shows a map that illustrates the returned objects given the required query input of system users.

Users interact with the system through the function panel. The function panel provides data acquisition interaction function and location query interaction function. Note that in case that API authentication settings for real-time crawling mode may require troublesome modifications, in the demonstration we also provide an alternative mode to allow the system to load offline track data.

The MALOS system mainly provides users with three functions: *Place-One*, *Place-k* and *Incremental-One*. *Place-One* and *Place-k* functions are mainly for users who have not yet deployed any facility and want to select locations to deploy new ones in some candidate areas, which can also be manually set by the system users. *Place-One*, which aims to find the optimal location from a set of candidates to place a new facility such that a score (*i.e.,* benefit or influence on some given objects) can be maximized. Further, a user can specify a constraint k to perform *Place-k*, which indicates the number of candidate locations the user plan to choose. Considering that there are existing facilities, and users may intend to expand the scale of the facility network by opening a new one, the system provides *Incremental-One* function, which allows users to enter a series of existing facilities locations as constraints for location selection. Different algorithms with respect to each of these use cases have been implemented in MALOS system.

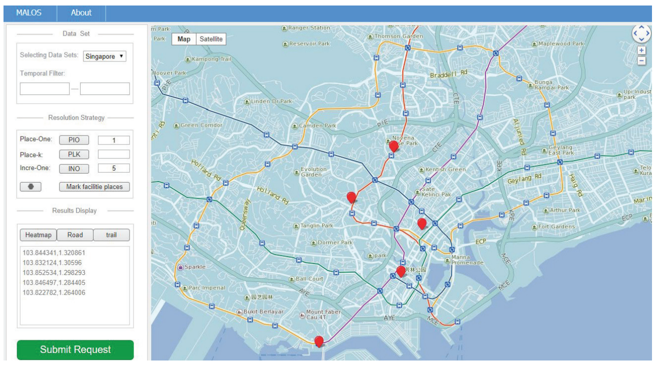
The *Place-One* function is addressed using PINOCCHIO algorithm. PINOCCHIO algorithm employs the R-tree structure to build geographic data index, then leverages two pruning rules based on a novel distance measure. With the help of influence arcs (IA) rule, it identify the candidates that influence the object. For the remnant candidates, non-influence boundary (NIB) rule is used to exclude those cannot influence the object. Lastly, the remnant candidates are verified.

The *Place-k* function is solved by GreedyPS algorithm. GreedyPS algorithm first calculates the set of objects affected by all candidates. Then it maps these sets to w bitmaps. Finally, it selects the candidates with the largest number of '1' in bitmaps until K is in each iteration process.

The *Incremental-One* function is addressed via LNB algorithm. It first constructs an index structure, Local Network Table (LNT), based on network locality. With the help of LNT, we initialize a Max Heap LNH ordered by $\Delta^+(v_i v_j)$, which is the upper bound of expected reduction of total distance (ERD) if a facility is built on edge $v_i v_j$. Then it iteratively checks the top candidate c in LNH. If $\Delta(ol)$ of current optimal candidate ol is greater than ERD upper bound for the top edge in LNH, the validation is finished. Otherwise, for every reference location whose local network covers $v_i v_j$, it needs to calculate $\Delta(c)$ through relevant rules. If a better candidate is found, it updates ol by c and eventually the optimal answer can be obtained, where $\Delta(c)$ is ERD for c.

After the request is processed by the server, the objects retrieved are returned. In order to enable users to observe the data intuitively, we need to visually present the spatial and temporal distribution of trajectory data. Visual comparison is one of the most fundamental and common visualization tasks [3]. Thus, we add a thermal layer, where we provide users with two types of density maps, namely, heatmap and roadmap. The thermal layer demonstrates the distribution density of the target trajectory in the geographic location by overlaying different color blocks on the map. In case that the volume of the check-in location data heat layer does not clearly reflect the trajectories of the crowd, we extensively add a trajectory layer to the system. The trajectory layer distinguishes the trajectories of different people by different colors, and reflects the overall trend of the trajectories by sampling method, which enhances the interaction experience for users. The marker layer provides a solution for visually presenting the results returned by the server and marking the input location points by users. For the *Incremental-One* function, a user does not need to enter the specific latitude and longitude coordinates of the existing facility in the system function panel. It is only necessary to find the location of the facility in the map panel and perform correlated marking on the marker layer.

[1]www.postgis.net/



```
103.844341,1.320861
103.832124,1.30596
103.852534,1.298293
103.846497,1.284405
103.822782,1.264006
```

Figure 3: Illustration of *Incremental-One* function.

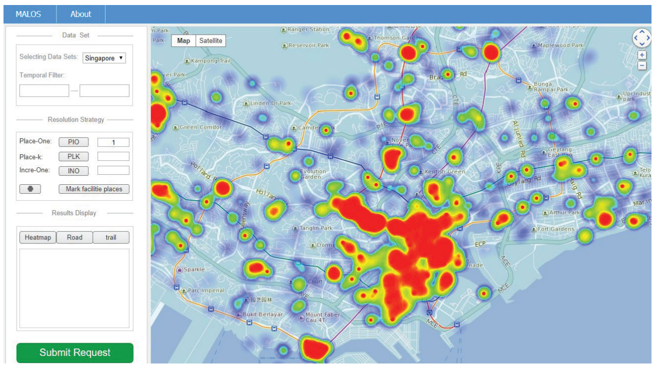


Figure 4: A visualization method of MALOS system(heatmap).

4 DEMONSTRATION

In our demonstration, as discussed before, we additionally implement an offline mode within MALOS system, in case there is troublesome modification for crawler API settings that is brought by the network connection issues. In offline mode, we adopt real-world check-in data in Singapore[2] to give users the opportunity to interact with MALOS and experience how the system can be used to select the location of facilities. The browser interfaces of MALOS has been shown in Figure 2. Users can use the system to find the best location of facilities in various scenarios, including *Place-One*, *Place-k* and *Incremental-One*.

4.1 Data loading and request submission

First, the user needs to select a target city to place the facility, and then select a function in the function panel to initiate the request. In MALOS system, we provide a list of cities for users to choose. For offline mode, the city can be only configured as Singapore. When the user selects a region where he wants to deploy the facilities, the map panel will automatically switches to focus on the corresponding area, and the server loads the historical check-in locations within the region. As shown in Figure 2, when the system loads the offline Singapore dataset, and the map panel also switches to Singapore. To specify the interested location, an audience can select a fine-grained location by drawing a rectangle on map panel (the latitude and longitude of the location are obtained using the Map API). For the *Incremental-One* function, the MALOS system allows users to enter some request constraints before running it. A user can enter a constant k to indicate the number of locations need to be picked. As illustrated in Figure 3, user enters a constant of 5 and marks five locations on the map to indicate the currently deployed facilities. The results display list also shows the longitude and latitude of these five facilities for user input verification.

4.2 Visualization and results display

Users can also explore the distribution of urban population information by visualization. There are three kinds of visualization methods, namely, heatmap, roadmap and trajectory-map for users to choose. As shown in Figure 4, heatmap is shown. In heatmap and roadmap, the darker the color, the greater the population density. The trajectory map shows the migration path of the crowd, and users can view the spatial and temporal distribution of urban population. As illustrated in Figure 3, when the query results are returned to the browser, the result objects are displayed in both the map panel and the result list. At the same time, users can further explore the surrounding environment of the recommended location by the system with the visualization.

[2]The default trajectory data source can be found in the github of PINOCCHIO listed in our MALOS project homepage: https://lihuixidian.github.io/malos/

5 CONCLUSION

In this demonstration, we present a general facility placement system, namely MALOS. MALOS solves a group of representative facility placement problems based on historical check-in location data that comes from the Internet and is updated in real time. MALOS adopts the browser-server model and provides an easy-to-use interface to answer *Place-One*, *Place-k* and *Incremental-One* queries. The queries cover the majority of facility placement problems given historical movement logs of massive users. Moreover, the system focuses on geographical issues that are general in all facility placement applications, and avoids taking into account application-dependent factors, such as the size of billboard, etc. In this way, the system can be used as a basic framework and easily adapted to different applications by taking into account extra specific application-aware factors.

ACKNOWLEDGMENTS

The work is supported by National Natural Science Foundation of China (No. 61672408, 61972309, 61976168), National Engineering Laboratory for Public Safety Risk Perception and Control by Big Data (PSRPC) and China 111 Project (No. B16037).

REFERENCES

[1] Jiangtao Cui, Meng Wang, Hui Li, and Yang Cai. 2018. Place Your Next Branch with MILE-RUN: Min-dist Location Selection over User Movement. *Inf. Sci.* 463-464 (2018), 1–20.
[2] Piotr Jankowski, Natalia V. Andrienko, and Gennady L. Andrienko. 2001. Map-centred exploratory approach to multiple criteria spatial decision making. *International Journal of Geographical Information Science* 15, 2 (2001), 101–127.
[3] Johannes Kehrer and Helwig Hauser. 2013. Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey. *IEEE Trans. Vis. Comput. Graph.* 19, 3 (2013), 495–513.
[4] Dan Li, Hui Li, Meng Wang, and Jiangtao Cui. 2019. k-Collective Influential Facility Placement over Moving Object. *The 20th IEEE International Conference on Mobile Data Management* (2019).
[5] Jing Wang and Yuchun Guo. 2012. Scrapy-Based Crawling and User-Behavior Characteristics Analysis on Taobao. In *2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2012, Sanya, China, October 10-12, 2012.* 44–52.
[6] Meng Wang, Hui Li, Jiangtao Cui, Ke Deng, Sourav S. Bhowmick, and Zhenhua Dong. 2016. PINOCCHIO: Probabilistic Influence-Based Location Selection over Moving Objects. *IEEE Trans. Knowl. Data Eng.* 28, 11 (2016), 3068–3082.
[7] Yipeng Zhang, Zhifeng Bao, Songsong Mo, Yuchen Li, and Yanghao Zhou. 2019. ITAA: An Intelligent Trajectorydriven Outdoor Advertising Deployment Assistant. *Proc. VLDB Endow.* 12, 12 (2019), 1790–1793.