Swarming in the Urban Web Space to Discover the Optimal Region

Chandan Kumar, Uwe Gruenefeld
University of Oldenburg
firstname.lastname@uni-oldenburg.de

Wilko Heuten
OFFIS Institute of Information Technology
wilko.heuten@offis.de

Susanne Boll
University of Oldenburg
susanne.boll@uni-oldenburg.de

***Abstract*—People moving to a new place usually look for a suitable region with respect to their multiple criteria of interests. In this work we map this problem to the migration behavior of other species such as swarming, which is a collective behavior exhibited by animals of similar size which aggregate together, milling about the same region. Taking the swarm intelligence perspective, we present a novel method to find relevant geographic region for citizens based on Particle Swarm Optimization (PSO) framework. Particles represent geographic regions which are moving in the map space to find a region most relevant with respect to user's query. The characterization of geographic regions is based on the multi-criteria distribution of geo-located facilities or landscape structure from the OpenStreetMap data source. We enable end users to visualize and evaluate the regional search process of PSO via a Web interface. The proposed framework demonstrates high precision and computationally efficient performance for regional search over a vast city based dataset.**

Keywords*—*Geographic Regions, Location-based Search, Swarm Intelligence, Particle Swarm Optimization, Regional Search

I. Introduction

In several planning and decision making scenarios people search for relevant geographic regions, e.g., persons moving to a new urban area/city would like to find a region with a similar neighborhood of geo-located facilities as their current living environment. Current location based services can be used to acquire information about facilities, popular landmarks or points of interest. However characterization and search of regions is currently not well supported by such services [10]. There have been some recent efforts to recommend interesting regions for end users based on query by example scenarios [9]. These approaches support the comparison of few selected regions; however there could be situation when users want to explore the entire urban space to discover the most suitable region. Due to the complexity of spatial databases and huge amount of geo-located facilities in urban areas it would be a computationally expensive task. Hence in this work we propose a heuristic based approach where few intelligent particles cordially move in the map space to identify most relevant regions. The proposed approach is derived from the popular particle swarm optimization framework [7].

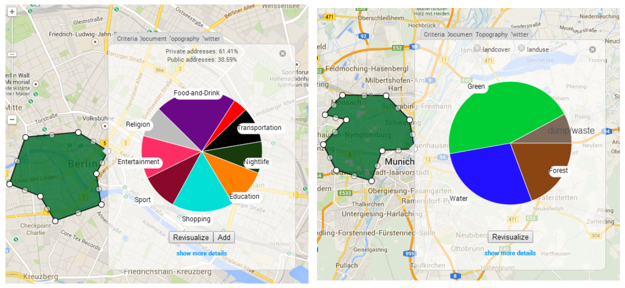
In recent years the particle swarm optimization algorithm has emerged as a new meta-heuristic approach derived from nature and has attracted many researchers' interests. The algorithm has been successfully applied to several optimization problems and application scenarios such as Web search, information retrieval and classification [2], [18], [4]. Nevertheless, the use of PSO for multi-criteria search in the context of geospatial maps is still a research area very few have tried to explore [11], [12]. PSO searches a space by adjusting the path of individual vectors, called "particles". The individual particles are drawn stochastically toward the position of their own best performance and the best performance of the swarm. In this work we propose a framework where particles are represented by geographic regions moving in the map space of definite boundaries such as cities. We inherit the simple concept of PSO algorithm in the regional search scenario, and capitalize on the PSO qualities such as robustness to control parameters, and computational efficiency. In addition to conventional PSO, we propose slight variations of PSO with congestion prevention and local exploration in the geospatial scenario. To the best of our knowledge, the approach presented in this paper is the first that applies PSO to assist users in locating the most suitable regions. To evaluate the efficiency of our PSO approach, we performed experiments on huge geospatial data for several major cities of Germany and Austria. Results indicate that our PSO framework is able to find precise regions based on user selected criteria in a computationally efficient manner.

In this paper, the objective is to investigate the capability of the PSO algorithm to locate the best geographic region based on user selected criteria. The rest of the paper is organized as follows. We give a brief description about regional search and exploration in section 2. In section 3, we present the basic idea of the PSO algorithm. In section 4, the PSO-based regional search algorithm and its variations has been proposed. Experimental results are given in Section 5. We discuss our findings and observations in Section 6. Finally, Section 7 concludes the paper with possible future research directions.

II. Regional Search

Geographic Information Retrieval (GIR) systems [1], [8] and various local search services[1] provide information access about specific locations, facilities and geographic points of interest. However, in various information seeking situations users are not only interested in the specific geo entities but the composition of urban areas and geographic regions. Users need to specify, analyze and compare the geographic regions, which is not feasible with the current local search systems as their querying, ranking and presentation methods are liable towards definite locations and entities. Users go beyond the popular place names and location for the characterization of

[1]http://www.maps.google.com



(a) Facilities (b) Landscape

Fig. 1: Distribution of geographic regions

their region of interest. Users could be interested in finding appropriate regions in many decision making scenarios. A person moving to a new city may look for an appropriate region to live; a businessman opening a new shop/store looks for a beneficial neighborhood, or even a tourist would want to visit interesting regions to fulfill her sightseeing desires in less time. In such scenarios multiple criteria of interest become simultaneously significant, e.g., a good geographic region for living might contain different criteria of geo-entities, e.g., availability of shopping facilities, medical facilities, and a good connection to public transport.

Exploration of geographic regions and their comparison was found as one of the key desires of current local search users [10]. It might not even be the concrete entities, but rather the atmosphere, composition, and spatial distribution that make up the "feeling" of a neighborhood that best capture the intention of a user. There have been some recent efforts to recommend interesting regions for end users based on query by example scenarios [9]. These approaches provide a facility based overview of geographic regions using georeferenced entities from the OpenStreetMap or spatial Web documents. However, suitable computational support is required to search the entire urban space for relevant geographic regions in an easy and effective manner.

Figure 1a shows an example of a geographic region selected by a user on the map interface, and how the distribution of facilities makes it an interesting living environment for the end user. The user prefers majority of shopping, food and drink facilities, but at the same time facilities like religion and sports also have some significance. There could be scenarios when a user moving to a new city would like to find a similar region with specified categorizations to fulfill his requirements. Figure 1b shows the distribution of a region based on the topographical structure, i.e., the landscape arrangement of a region based on water, greenery, forest etc. This could also be a significant factor while characterizing a region by end users. Hence in this work we propose a system where users could provide regional query by example characterizations and the algorithm would search for the most suitable geographic region respectively.

III. Particle Swarm Optimization

The particle swarm optimization (PSO) is a probabilistic optimization algorithm which discovers the optimal solution using a population of particles. It is inspired by the social behavior of biological organisms, specifically the ability of groups of species of animals to work as a group in locating desirable positions in a given area, e.g., bird flocking or fish schooling [14], [7], [6]. PSO is based on the exchange of information between individuals, which are called particles. Each particle represents a possible solution to the optimization task. During each iteration each particle accelerates in the direction of its own personal best solution found so far, as well as in the direction of the global best position discovered so far by any of the particles in the swarm. This means that if a particle determines a promising new solution, all the other particles will move closer to it, exploring the region more thoroughly in the process. Some of the attractive features of the PSO include ease of implementation and the fact that no gradient information is required. Hence it has been used to solve a wide array of different optimization problems [13], [17].

In PSO methodology, each particle p_i has a set of attributes: current velocity V_i, current position X_i. Each of the particles start with randomly initialized velocities and positions. The particle updates its velocity and position based on the equations (equation 1, 2). The particle is evaluated by a fitness function and the local best position of particle (P_{best}) and the global best position of swarm (G_{best}) are updated based on the fitness value of particle.

$$\begin{aligned} V_{i+1} = w * V_i + c1 * rand() * (P_{best} - X_i) \\ + c2 * Rand() * (G_{best} - X_i) \end{aligned} \tag{1}$$

$$X_{i+1} = X_i + V_{i+1} \tag{2}$$

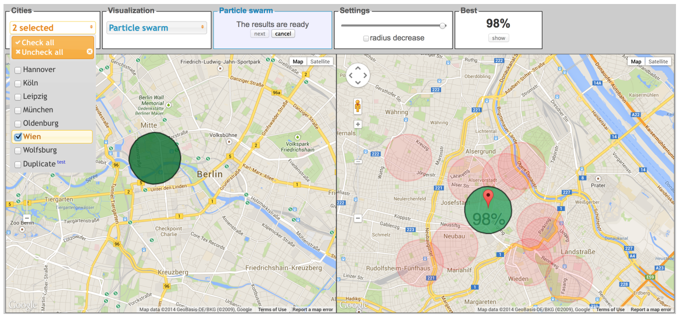
- Here i stands for current generation of the iteration process.
- $w, c1, c2$ are the momentum coefficient, recognise coefficient, social coefficient respectively.
-rand(), Rand() are the functions that generate random numbers between 0 and 1.

IV. Regional Search with Particle Swarm Optimization

We propose a system inspired of swarm intelligence to find the most appropriate region for end users. In this section we first showcase the Web interface of our PSO framework. Subsequently we discuss the methodological details of PSO algorithm for the regional search.

A. Web Interface for PSO regional search

Figure 2 shows the Web interface of the proposed system. Here users could select a particular city or multiple cities for the regional search. In the shown example, the user intends to find the most suitable region in Vienna with respect to a specified region in Berlin (query region). The system allows user to specify the region of interest via a spatial query (polygon or circular selection) on Berlin's map. In the example shown in Figure 2, the user selected a circular region of interest on the Berlin city map. Once the user has provided her area of interest, the system triggers several particles in the city map of Vienna. Each particle has the same diameter as the query



The results are ready

Fig. 2: PSO regional search

region. These particles start moving across the Vienna map to find the best region with respect to user's criteria. Users could view the particles and swarm movement on the map interface, and they could also control the speed of the swarm via the top panel of interface. Here they could also spot the current status (found similarity) of the search process. The swarm would keep searching the map space until it discovers the best region with 100% similarity or if the user terminates the search process. In this example the user terminated the search process when the swarm had found a region with 98% similarity. The best region is indicated with the dark green color and its similarity value. The transparent red circles are showing the last positions of particles when the search process was stopped. Users could also visualize the regional distribution of results. Figure 3 shows the exploration of regions with the pie chart divisions of criteria, which inherently verifies the estimated similarity.

1) Realization Framework: We used Google Maps as a map framework, the visualizations were built using Data Driven Documents[2] (D3), a JavaScript library for visualization design. The characterization of geographic regions is based upon the availability of geo-located facilities. Our geospatial database consists of a huge collection of geo-entities for various cities from Germany and Austria. These geo-entities are extracted from the rich geospatial data source OpenStreetMap[3] (OSM). All the geo-entities are classified in a pre-defined set of categories such as shopping, education, medical facilities, sports, etc. (the category description is a generalization of the OpenStreetMap category listing). We also characterize the topographic detailing of OSM such as natural landscape (categories such as water, greenery, mountains, forests etc.) and landuse (road, railways, industrial, commercial areas etc.). For this purpose we developed an OSM map parser using lucene spatial boundaries[4].

B. PSO methodology

Inspired by the success of PSO as a common heuristic technique for multi-criteria optimization and decision making problems, we derive a PSO-based method to search for geographic regions with multiple criteria of interests and assist the spatial decision making of end users. The goal is to find an appropriate region in less computational time. To apply PSO algorithm for regional search, we mapped some concepts of PSO to the spatial decision problem. Particle stands for a circular region on the map as a candidate solution, which means it is similar to the query region. For a particle p_i, the parameter position X_i stands for the current location of p_i on the map, i.e., geocoordinate of the centre of circular region. Velocity V_i implies for the movement direction of particle on the map surface. During the search process particles are moving within the map boundaries. Inherently we translate the city frame into a torus, i.e., if a particle is moving over the right/top border it will enter the map from the left/down border.

[2]http://www.d3js.org/
[3]http://www.openstreetmap.org/
[4]http://lucene.apache.org/core/400/spatial

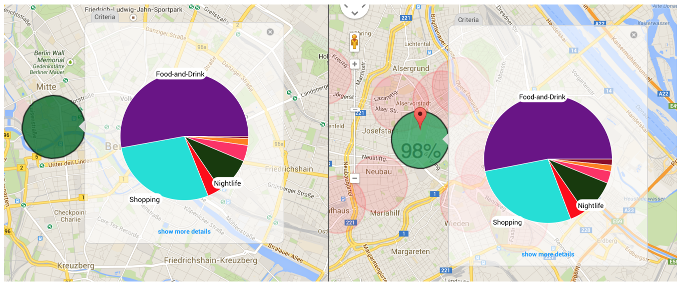


Fig. 3: PSO results with visualizations

The proposed model executes a regional search process as described below. In the initial step, a set of particles as candidate solutions are generated. In the regional search scenario the search space is restricted to the map boundaries; hence the random position and velocity of initial set of particles are set based on equation 3, 4. Then the particle updates its velocity and position based on the equations 1, 2. The particle is evaluated by a fitness function (equation 5) and the local best position of the particle (P_{best}) and the global best position of the swarm (G_{best}) are updated based on the fitness value of particle. Finally, if the swarm finds the optimal region, i.e., a particle representing 100% similarity, or the user aborts the search process, then the algorithm outputs the best solution. Otherwise, the algorithm would continue the iteration until a satisfied region is found.

$$\begin{aligned} X_0 = Vector(&minLat + (rand() * midLat), \\ &minLng + (Rand() * midLat) \end{aligned} \tag{3}$$

$$\begin{aligned} V_0 = Vector(&(rand() - 0.5) * (midLat/C), \\ &(Rand() - 0.5) * (midLng/C)) \end{aligned} \tag{4}$$

- Here $minLat, maxLat, minLng, maxLng$ are the minimum/maximum latitude and longitude within the boundaries of the city
- $midLat$ is the space size between $minLat$ and $maxLat$, hence $midLat = maxLat - minLat$, similarly $midLng = maxLng - minLng$
-$rand()$, $Rand()$ are the functions that generate random numbers between 0 and 1. However we subtract it with 0.5 to generate negative values as well, so that the particle movement in both directions is possible

- C is the constant parameter to control the movement speed of particles, which has been set to $C = 10$ in this work

Algorithm 1 describes the stepwise process of the PSO regional search. Let S be the number of particles in the swarm, each having a position X_i in the map space and a velocity V_i. The goal is to find a solution a for which $f(a) > f(b)$ for all b in the search-space. $f()$ is a fitness function explained below.

```
for i:=1,..S do
    Initialize the particle position $X_i$ with formula (3)
    Initialize particle's best known position $P_{best} = X_i$
    if $(f(P_{best}) > f(G_{best}))$ then
        $G_{best} = P_{best}$
    end
    Initialize the particle's velocity $V_i$ with formula (4)
end
while optimal region found or abort do
    for i:=1,..S do
        Update particle's velocity $V_i$ using formula (1)
        Update particle's position $X_i$ using formula (2)
        if $(f(X_i) > f(P_{best}))$ then
            $P_{best} = X_i$
            if $(f(P_{best}) > f(G_{best}))$ then
                $G_{best} = P_{best}$
            end
        end
    end
end
```

Algorithm 1: PSO for regional search

1) Fitness function: The fitness of a particular particle is computed with respect to its similarity to the user specified query region. Each particle is a circular region on the map characterized by the geo-located facilities. The relevance of a particle region is based on its possibility to generate/replicate the queried region. The index contains the region-category matrix where each region R is represented by a category distribution $(C_1, C_2, .., C_n)$, e.g., transportation, education, sport, shopping, etc. Each category C_i is based upon the geo-entities which belong to category i. The relevance of a particular particle region R_p is based on the similarity of its categorical distribution with respect to the query region R_q. There could be various measures to estimate the distribution similarity [16]. However, to compute the association between geographic regions by mean of their criteria distribution, we found the Euclidian-Distance based distribution similarity measure [15] most appropriate.

$$fitness(p) = Similarity(R_p, R_q) \tag{5}$$

C. PSO with Congestion Prevention

During the PSO experiment with regional search we discovered that there are instances when all the particles stuck around the same region for long durations, i.e., all particles running around the best position in the swarm. When all the particles have similar G_{best} and P_{best}, the update in the particles velocity and positions are negligible, hence they stay at the same location and squander the precious iterations. We try to avoid such congestion circumstances by tracking the particles history, so if a particle does not find a better solution in a considerable number of iterations, its history is cleared by resetting the speed and position to a random value using equation 3,4. We believe such a heuristic could be vital in this scenario since a small number of particles are searching in the huge city boundaries. Hence the iteration of each particle becomes a significant factor to be minimized for relevance. Algorithm 2 shows the PSO search mechanism with congestion preventions. The variable $History$ is used to keep track of the particle status, so when the particles do not find a better position for a constant C number of attempts, it gets initialized to random values.

```
while optimal region found or abort do
    for i:=1,..S do
        if History > C then
            Initialize $X_i$ with formula (3)
            Initialize $V_i$ with formula (4)
            Reset $History$ to null
        end
        else
            Update $V_i$ using formula (1)
            Update $X_i$ using formula (2)
        end
        if $(f(X_i) > f(P_{best}))$ then
            $P_{best} = X_i$
            Reset $History$ to null
            if $(f(P_{best}) > f(G_{best}))$ then
                $G_{best} = P_{best}$
            end
        end
        else
            $History++$
        end
    end
end
```

Algorithm 2: PSO with congestion prevention

D. PSO with Local Exploration

In our spatial adaptation of PSO algorithm, the particles are moving in the map space which symbolizes the real geospatial world. In the spatial search scenarios neighborhood has a significant impact, e.g., geo-located facilities could be found closer to each other in the area like city centers. Hence we propose a modification of PSO with local exploration to benefit the particles from their neighboring region. PSO with local search has been used in several recent approaches [17], [11], where the particles communicate with nearest neighbor particles. However in this work we focus specifically on finding a better position for a particle in its restricted neighborhood which is rather close to hill climbing algorithms for PSO [3].

In this model we search in the particle neighborhood only if the new position of a particle is better than its previous position. It's the assumption that a particle has moved to a relevant area on the map and there could be better positions nearby. If better fitness is found in the neighborhood we replace the current position of particle to its new neighboring position. The local exploration is generating a constant number of random vectors using the equation 6, which is similar to the method being used to set the initial velocity of particles (equation 4). However the value of parameter C is much higher here to move only in the close neighborhood. In iterative steps we choose one of the vectors and add it to the position until we do not find a better position with a superior fitness for the particle. In the PSO details described in Algorithm 1, if the condition $(f(X_i) > f(P_{best}))$ is satisfied the local exploration process is triggered, i.e., the algorithm attempts to find other positions near X_i (which has better fitness than X_i) by stepping in the directions of vector generated by equation 6. In a fixed set of iterations if a better position L is found, this would become the new X_i as well as the P_{best} for the subsequent search process.

$$Vector((rand() - 0.5) * (midLat/100), \\ (Rand() - 0.5) * (midLng/100)) \tag{6}$$

V. Evaluation

In this section we describe our experiments to evaluate the performance of proposed regional search with PSO and its variations with congestion prevention (PSO_CP) and local exploration (PSO_LE). The empirical parameters such as number of particles that yields high solution were set by experiments prior to the performance evaluation test. We access the performance of these algorithms against the baseline method of exhaustive complete search.

A. Complete Search

In this method we perform the regional search in the most conventional manner. When the user searches for the best region in a city, each possible region in the complete city boundary becomes the candidate solution. This inherently means that the entire map space would be devised in a grid raster. The similarity calculation with each possible region and query region would be estimated to discover the most relevant region. This method can certainly yield good results but could become a computationally expensive task based on the dimensionality of city space. For a city like Berlin there would be 2 500 000 possible regions (if each region is considered to have distance of 25 meters between their centers). However the complexity could be reduced if the regions are mapped in far distances but that could decrease the performance since the most appropriate region could get omitted.

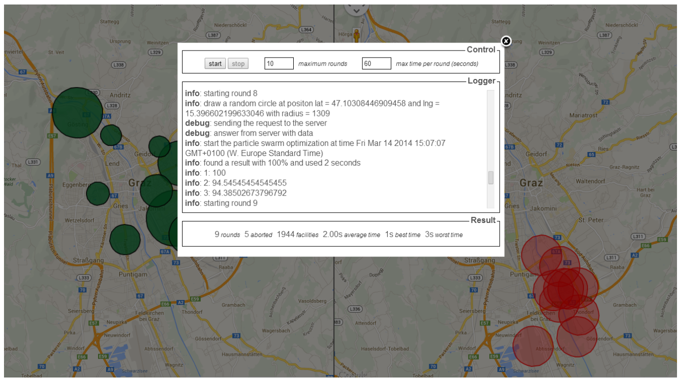
B. Benchmark and Data Description

In this work we proposed a novel problem scenario of regional search so it is unlikely to comprise a benchmark dataset of similar regions to be used for evaluation. Moreover it is difficult to categorize the exact similarity of different regions even from explicit user feedback. Hence we introduced the feature of searching for the same region in a duplicate city. In this scenario if the user has specified a query region R_q in city C_a. The search algorithm which is supposed to find most similar region is applied in the same city C_a. If the system could find a region R_n with 100% similarity where $R_n \sim R_q$, i.e., system could find the same region as of query region, this signifies the efficiency of an algorithm to find the most appropriate region.

Figure 4 shows an example of our automatic evaluation scenario; here the evaluation is conducted in the Austrian city Graz. Duplicate map of the city is shown on both sides of the interface. The system randomly generates queries on the map (shown by dark green circular regions on the left side map) and the PSO algorithm searches the best possible region for each random query. The dialog box on the top describes the stepwise search process.

The evaluation has been conducted in 14 different German and Austrian cities. For each city 100 random queries being generated, i.e., the system performs regional search 100 times per city. Hence the algorithms have been compared for 1400 runs. Moreover we evaluate these algorithms for two different kinds of regional similarity metrics. First, based on the geolocated facilities in the regions, and secondly based on the topography (landscape-landuse structure) of regions. We assess the performance based on three major criteria described as follows:

- Precision: We consider the search process a success only if the algorithm is able to find the accurate region with 100% similarity as of query region. So the precision is the percentage ratio of successes with total runs for a particular algorithm.
- Time: The proposed algorithms could find the most relevant region in few milliseconds or it could take several minutes. Hence to generalize the performance we fixed the maximum time limit to 60 seconds for all the algorithms. If the algorithm is not able to find the best region in the maximum time limit the process is aborted and considered as failure. We imposed this time limit of 60 seconds for evaluation considering that no Web users would like to wait more than a minute for search results. We report the results with respect to the average time taken by algorithms.
- Function calls: One of the major computational costs of regional search is associated with the fitness function or similarity calculation among regions. During each iteration of the search process every particle region calls the fitness function for relevance judgment. We observed that time taken for a particular run is proportional to the number of fitness calls from the algorithm. Hence to showcase the complexity of



10 maximum rounds 60 max time per round (seconds)

```
info: starting round 8
info: draw a random circle at positon lat = 47.10308446909458 and lng =
15.396602199633046 with radius = 1309
debug: sending the request to the server
debug: answer from server with data
info: start the particle swarm optimization at time Fri Mar 14 2014 15:07:07
GMT+0100 (W. Europe Standard Time)
info: found a result with 100% and used 2 seconds
info: 1: 100
info: 2: 94.54545454545455
info: 3: 94.38502673796792
info: starting round 9
```

9 rounds 5 aborted 1944 facilities 2.00s average time 1s best time 3s worst time

Fig. 4: Evaluation with duplicate city

algorithms we state the number of fitness function calls in addition to the time. The measured time unit could be dependent on CPU capability and status but the number of calls would always be the same irrespective of machine configurations.

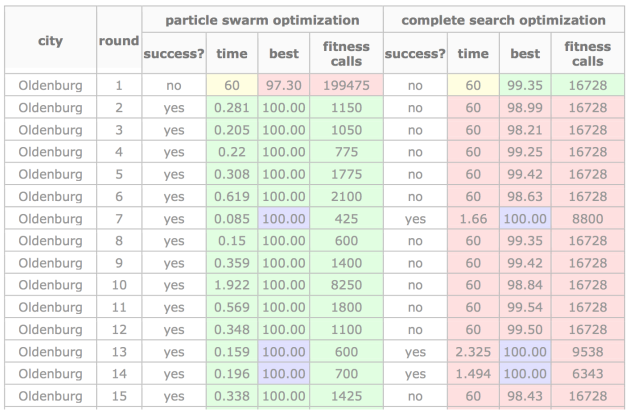
We offer a transparent evaluation process via a Web interface. Figure 5 shows an example of our Web based evaluation sequence, where the summary of results could be viewed with respect to each city and queries via a grid view. Here each row represents the result for a particular regional query. The green entry signifies the superiority of the algorithm against the red entries, and the blue cells signify the equality.

C. Results

Table I shows the performance comparison of proposed algorithms for regional similarity with criteria distribution of geo-located facilities. The results indicate that all the PSO algorithms perform better compared to exhaustive complete search process. The standard PSO method is able to achieve satisfactory ∼82% performance in less time compared to complete search. The PSO with local exploration (PSO_LE) obtained better results in even lesser time. More distinctly PSO_CP could achieve the significant 89% results in 1.85 seconds. The number of function calls is providing the similar indication that PSO methods needs a significantly smaller number of fitness function calls compared to complete search, hence the computational cost should be lower.

Table II shows the performance comparison for regional search based on the landscape-landuse structure (topographic similarity). Once again PSO algorithms are able to achieve high performance in significantly less time compared to CS. PSO_CP could achieve the significant results in 1.05 seconds compared to 13.99 seconds taken by complete search. The number of function calls apparently signifies the differences in the computations efficiency of complete search and PSO algorithms.

We observed that the performance gap between complete search and PSO methods significantly depends on the complexity and size of search space. For a small German city like Oldenburg, complete search and PSO_CP could achieve



city	round	particle swarm optimization				complete search optimization			
		success?	time	best	fitness calls	success?	time	best	fitness calls
Oldenburg	1	no	60	97.30	199475	no	60	99.35	16728
Oldenburg	2	yes	0.281	100.00	1150	no	60	98.99	16728
Oldenburg	3	yes	0.205	100.00	1050	no	60	98.21	16728
Oldenburg	4	yes	0.22	100.00	775	no	60	99.25	16728
Oldenburg	5	yes	0.308	100.00	1775	no	60	99.42	16728
Oldenburg	6	yes	0.619	100.00	2100	no	60	98.63	16728
Oldenburg	7	yes	0.085	100.00	425	yes	1.66	100.00	8800
Oldenburg	8	yes	0.15	100.00	600	no	60	99.35	16728
Oldenburg	9	yes	0.359	100.00	1400	no	60	99.42	16728
Oldenburg	10	yes	1.922	100.00	8250	no	60	98.84	16728
Oldenburg	11	yes	0.569	100.00	1800	no	60	99.54	16728
Oldenburg	12	yes	0.348	100.00	1100	no	60	99.50	16728
Oldenburg	13	yes	0.159	100.00	600	yes	2.325	100.00	9538
Oldenburg	14	yes	0.196	100.00	700	yes	1.494	100.00	6343
Oldenburg	15	yes	0.338	100.00	1425	no	60	98.43	16728

Fig. 5: Automated online evaluation

90% and 93% respective success within 1 minute time limit (topographic similarity). However for a big city such as Berlin which has a vast amount of facilities distributed all over the city, a complete search could achieve just 59% as compare to 86% success for PSO_CP. This indicates that PSO methods are even more beneficial for regional search when the magnitude of spatial database is larger, and could be a practical application in the scenarios when the search space is growing from cities to states or countries.

Algorithm	Precision	Avg time (sec)	Avg funtion calls
PSO	81.93	3.98	44082
PSO_CP	89.14	1.85	3266
PSO_LE	86.86	3.12	21278
CS	75.21	5.61	62817

TABLE I: Results with facilities based similarity metric

One of our main observations is that the PSO algorithms are able to achieve high performance in a small time stamp. However a complete search could achieve high accuracy given the longer time period. Figure 6 represents a performance comparison graph between complete search and PSO algorithms with variable time periods. For the time limit of 10 seconds we notice a huge performance gap between PSO and CS; CS would eventually catch up with the PSO performance given the longer duration of more than a minute. However in real world applications time is a critical factor, so rapid regional search solution from PSO algorithms would be really appreciated.

VI. Discussion

The PSO method proved to be more efficient and specifically less time consuming which is of prime importance for real time algorithms. We were able to enhance the performances of PSO even further with proposed variations for regional search. PSO with local exploration (PSO_LE) had achieved good results in relatively lower time and function calls compared to standard PSO. The PSO_CP have obtained exceedingly high performance which has validated our earlier hypothesis of avoiding particles to stay at the same location for a longer duration. In the congestion prevention method particle values are randomized if they are not able to find better position. This eventually prevents the swarm from staying longer on one particular region of the map. In the scenario such as regional search this would have helped the PSO method to explore different regions of the city more aggressively.

In addition to the enhanced performance PSO provides better user control compared to complete search. Users could run PSO algorithm for the desired period of time and view the current best result as per their convenience. However this is not possible with the complete search and the user has to wait until all possible regions in the city space have been assessed. Figure 7 shows the complexity of complete search process by

Algorithm	Precision	Avg time (sec)	Avg funtion calls
PSO	90.79	7.64	47816
PSO_CP	89.29	1.05	5301
PSO_LE	85.36	6.05	9053
CS	84.79	13.99	111626

TABLE II: Results of landscape based similarity metric

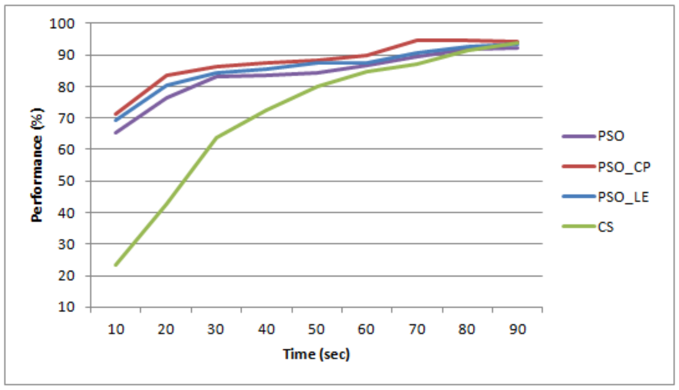


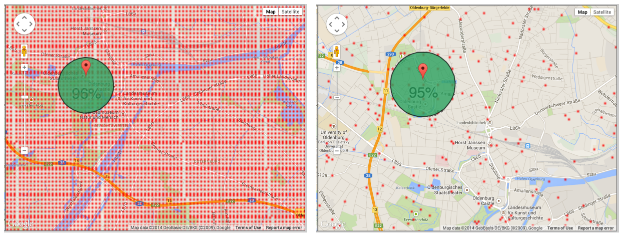
Fig. 6: Performance graph with maximum time limit

means of number of function calls for a particular user query. The system could find a region with 96% similarity, but the whole city map is scanned with respect to different possible region. Each red dot on the map is indicating the region for which the similarity function had been calculated. However, in case of PSO the system is able to find a region with 95% similarity but the distribution of red dots are very sparse, i.e. the system is able to achieve good results in less time and function calls.

One of the distinctive aspects of our work is the visualization and interaction with PSO to end users. There have been several use cases of PSO proposed in popular application scenarios such as Web search, retrieval and classification [5], [18], [4]. But the inference has been more on the algorithmic level, as compared to our approach where user could view and control the PSO search process. The visuals of the particle movements on map interface could provide the relevant indication of important areas to end users. The particle movements could be viewed during the PSO search process; moreover we provide the option of generating a heatmap of the particles movement. Figure 8 shows the example heatmap of swarm movement in the city map space. The red spots in the center are the high density areas where the particles stayed for longer duration; the green, blue and purple colors specify the density in decreasing order respectively. The particles staying in an area for longer duration is an obvious indication of higher relevance in nearby regions.

The Web interface presented to users had the option of searching a region with query by example scenarios, i.e., one could select a model region on the map interface and request the system for a region with similar characterization. If the user is not able to describe her requirement from example regions they could try to modify the pie-chart distribution of categories. However the proposed framework is not limited to such regional queries, other input methods could be simply integrated. For example users could select a list of criteria and weight them according to their interest, and request PSO algorithm to discover regions with similar distribution of criteria.

We have presented a computationally efficient way to search for relevant regions based on facility and landscape compositions. However, the idea could be implied in several other scenarios like regional similarity with news or social media, e.g., if one wants to find which region has the social opinion (similar tweets) with respect to a event happened in a particular region. The proposed framework could be used for several other purposes with different data sources and fitness function. In this work we restricted the regional search within city boundaries but it can be easily extended to states or countries or further. For example a successful European industry wants to open a new plant in India and would like to find the similar region based on the availability of natural resources as of its current infrastructure in a European region.



(a) Complete Search (b) PSO

Fig. 7: Functional distribution on map

VII. Conclusion

In this paper we proposed a swarm intelligence based methodology to search for fitting geographic region in the map interface. Swarm is a collection of particles which represent a solution region and communicate with each other to find the best region. The relevance estimation of geographic regions is based on the multi-criteria distribution of geo located facilities or landscape structure from OpenStreetMap data sources. We provide an end user Web interfaces to visualize, analyze, and evaluate the search process of PSO. We also proposed slight variations of PSO with congestion preventions and local search heuristic in the regional search scenario. In the evaluation conducted with huge city database, all algorithms have shown good performance and efficiency against exhaustive complete search. We believe that PSO is an incredibly powerful method to perform the multi-criteria spatial decision task of regional search, and its applicability becomes more significant as the search space or database becomes larger. In future we would

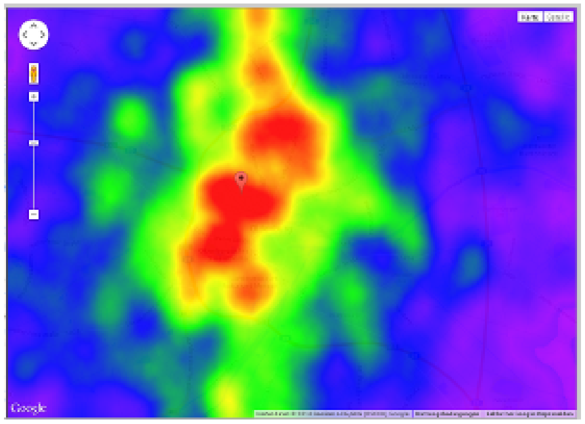


Fig. 8: Heatmap of swarm movement

like to integrate other useful data sources to characterize the regions, e.g., geospatial social media. We would also like to make the system publically available to evaluate it in real world settings and assess the usability among end users.

Acknowledgment

The authors are grateful to the DFG SPP 1335 'Scalable Visual Analytics' priority program[5] which funds the project UrbanExplorer.

References

[1] D. Ahlers and S. Boll. Location-based Web search. In A. Scharl and K. Tochtermann, editors, *The Geospatial Web. How Geo-Browsers, Social Software and the Web 2.0 are Shaping the Network Society*. Springer, London, 2007.

[2] P. I. Borkar and L. H. Patil. Web information retrieval using genetic algorithm-particle swarm optimization. *International Journal of Future Computer and Communication*, 2(6):595–599, 2013.

[3] J. Chen, Z. Qin, Y. Liu, and J. Lu. Particle swarm optimization with local search. In *Neural Networks and Brain, 2005. ICNN&B'05. International Conference on*, volume 1, pages 481–484. IEEE, 2005.

[4] E. Diaz-Aviles, W. Nejdl, and L. Schmidt-Thieme. Swarming to rank for information retrieval. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 9–16. ACM, 2009.

[5] H. Drias. Web information retrieval using particle swarm optimization based approaches. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, volume 1, pages 36–39, Aug 2011.

[6] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[7] J. Kennedy, R. Eberhart, et al. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Perth, Australia, 1995.

[8] C. Kumar. Relevance and ranking in geographic information retrieval. 2011.

[9] C. Kumar, W. Heuten, and S. Boll. A visual interactive system for spatial querying and ranking of geographic regions. In S. N. Lindstaedt and M. Granitzer, editors, *I-KNOW*, page 30. ACM, 2013.

[10] C. Kumar, B. Poppinga, D. Haeuser, W. Heuten, and S. Boll. Geovisual interfaces to find suitable urban regions for citizens: A user-centered requirement study. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 741–744. ACM, 2013.

[11] J. Lane, A. Engelbrecht, and J. Gain. Particle swarm optimization with spatially meaningful neighbours. In *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, pages 1–8. IEEE, 2008.

[12] S. Ma, J. He, F. Liu, and Y. Yu. Land-use spatial optimization based on pso algorithm. *Geo-spatial Information Science*, 14(1):54–61, 2011.

[13] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 603–607. ACM, 2002.

[14] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[15] G. Qian, S. Sural, Y. Gu, and S. Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 1232–1237, New York, NY, USA, 2004. ACM.

[16] J. Shao, Z. Huang, H. T. Shen, J. Shen, and X. Zhou. Distribution-based similarity measures for multi-dimensional point set retrieval applications. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 429–438, New York, NY, USA, 2008. ACM.

[5] http://www.visualanalytics.de

[17] Y. Sun, Z. Wang, and B. Wyk. Local and global search based pso algorithm. In Y. Tan, Y. Shi, and H. Mo, editors, *Advances in Swarm Intelligence*, volume 7928 of *Lecture Notes in Computer Science*, pages 129–136. Springer Berlin Heidelberg, 2013.

[18] Z. Wang, Q. Zhang, and D. Zhang. A pso-based web document classification algorithm. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 3, pages 659–664. IEEE, 2007.