Department: Head
Editor: Name, xxxx@email

Approximating region boundaries based on qualitative and quantitative information

Zhiguo Long
School of Computing and Artificial Intelligence, Southwest Jiaotong University, China, 611756

Michael Sioutis
Faculty of Information Systems and Applied Computer Sciences, University of Bamberg, Germany, 96047

Qingqian Li
School of Computing and Artificial Intelligence, Southwest Jiaotong University, China, 611756

Hua Meng
School of Mathematics, Southwest Jiaotong University, China, 611756
(Corresponding author: menghua@swjtu.edu.cn)

Heng-Chao Li
School of Information Science and Technology, Southwest Jiaotong University, China, 611756

***Abstract*—Approximating regions is a topic that can have important applications in artificial intelligence whenever uncertain, incomplete, or inconsistent/contradictory spatial information is involved. This paper devises a new method to generate region approximations based on rough qualitative direction and distance information. The main idea is first to give a suitability score to smaller regions, *cells*, that are obtained by partitioning the area of interest, and then to identify candidates to form an approximation by evaluating score contribution ratios under a certain threshold. This paper designs a novel mechanism that compares the actual information of cells with the provided rough information in order to calculate suitability scores, and proposes to exploit a regressor model that can predict a threshold given certain suitability scores. Experimental results show that, given a good threshold, this new method can approximate target regions effectively, and that good thresholds can be reliably obtained through a trained regressor.**

NATURAL language-based spatial information has always been common and played an important role in applications, such as place descriptions in web documents and social media texts, and instructions with spatial expressions for collaborative building tasks. In fact, as observed in the literature, e.g. [1], generating geometric depictions out of natural language descriptions could facilitate decision making in many situations; for example, automatically generating a sketch map from text descriptions can be critical for emergency response, as seen in [2, Figure 12], which illustrates an application for emergency call services. Approximating regions naturally also extends to domains and applications that make use of regions for representation of and/or reasoning with uncertain knowledge, as is the case with probabilistic spatio-temporal knowledge bases for instance [3], where regions are spatially and temporally related with other objects over probability intervals, e.g., an object can be inside a region at a given point of time with probability in a certain interval $[\lambda, u]$. Clearly, as with all of our aforementioned examples, regions may not always be provided, and may need to be approximated on the basis of incomplete, uncertain, and even inconsistent/contradictory knowledge.

Motivation The problem considered in this article is to approximate or *delineate* a region when given some vague natural language descriptions. This problem could be encountered in many important applications. Specifically, the work discussed here could contribute to detection and monitoring systems that are based on volunteer public reporting of fires via mobile/smart phone calls, which are becoming increasingly common for detecting fires early [4, Section 2]; we explain as follows. Suppose there is a wildfire and several people saw smoke and posted it on social media like "Bushfire burning about 7km *northwest* from Yallourn North",[1] and "I saw fire about 11km *northeast* from Moe", without providing the exact location information of the fire. Figure 1 illustrates such a situation. It should be noted that the distance and direction information here is just an estimation, and by no means accurate, and

[1]See, e.g., https://twitter.com/andrew_lund/status/1091210815011614721 from Twitter.

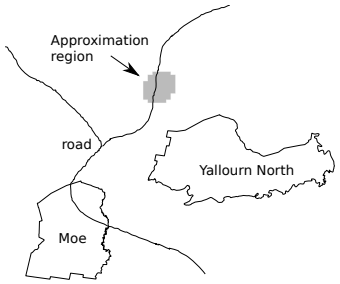


Figure 1. Illustration of locating a region by natural language descriptions, with the help of the region approximation method proposed in this article.

the approach proposed in this article takes such inaccurate information into consideration and is reasonably robust to it (cf. Figure 6). Based on information from the above non-expert descriptions, an approximation region for the wildfire location will then be automatically generated by our approach. With this approximation region, there is a variety of possibilities, such as helping with fire detection, suppression, engagement, evacuation, and so on. For example, warnings could be generated automatically by calculating intersections of geometric representations of roads, towns, or power stations with this approximation region of fire, like "the road could be blocked by fire and the public should reconsider their travel plan"; authorities and other involved parties (e.g., news agencies) could also use the approximation region to better illustrate the affected area of the fire. The above settings are also common for other emergency situations, e.g., sudden terrorist attacks might be witnessed by people around, who would proceed to post statements on social media like "someone was shot, less than 1km north from the square". Social media companies or other agencies could automatically share this kind of information visually and rapidly with the approach proposed in this article, which is more intuitive and clear than pure text and can thus better help others understand and deal with the situation.

Contribution Previous methods mostly concentrated on delineating a region by using points that are known to be contained or not in the region [5], [6], [7], [8]. Other works considered generating sketch maps from natural language

Published by the IEEE Computer Society

descriptions involving qualitative spatial relations, but the end result is a very rough region approximation based on rectangles that is much more inaccurate than the one proposed in this paper (cf. [2, Figure 7]), and may often not correspond to the actual spatial scene [2], [9] (in these works, the size, shape, or distance values have no connection to the actual ones but are just illustrative). Here, we consider the problem of delineating a region without the information of points contained in the region, but by combining vague qualitative and quantitative information, which we argue that it is also a more realistic approach, grounded on real-world examples such as the ones mentioned earlier. Our experiments show that this novel method can approximate regions quite well by using only rough direction and distance information.

Approach

Preliminaries

A *region* on the plane is a nonempty regular closed point set, that is, a set of points that is the same as the closure of its interior. Suppose that there is a region a on the plane, and a set of other k regions $N(a) = \{b_j : 1 \leq j \leq k\}$, known as the *neighbours* of a. Note that a neighbouring region b_j does not necessarily share common parts with a. Then, there will be some qualitative directional relations between a and each b_j. Here, we consider the following eight relations from Cardinal Relation Algebra [10]: $\mathsf{NW}, \mathsf{N}, \mathsf{NE}, \mathsf{W}, \mathsf{E}, \mathsf{SW}, \mathsf{S}$, and SE, representing *northwest*, *north*, *northeast*, *west*, *east*, *southwest*, *south*, and *southeast*, respectively. These relations are ambiguous in that, given two regions, it is usually hard to determine exactly what the directional relation between them is.

There will also be some distance information between a and each neighbouring region b_j. An example of this is, "a is 19 km away from b_j". Note that this information is also vague, as for regions there are different ways for measuring the distance between them, e.g., distance between centroids, between landmarks, or even between boundaries, and even with a fixed way of measurement, the measured distance might be a bit unreliable due to error.

In our discussion, for a region a and a set of its neighbours $N(a)$, we suppose that there is a set of tuples (r_j, d_j^*, b_j) associated with a, where $b_j \in N(a)$, representing the directional relation r_j and the distance d_j^* between a and b_j.

Now we can define the *region approximation problem* tackled in this article.

Definition 1:
The *region approximation problem* is to construct a region A that approximates the true (yet unknown) region a with respect to both its location and area, given a set of neighbouring regions $N(a)$ and a set of directional relation and distance tuples $R = \{(r_j, d_j^*, b_j) : b_j \in N(a)\}$.

For example, in some situation, one might need to find out the approximate location and area of a place a, but the only known information could be that a is 1.2 km north of b, and a is 2.3 km southeast of c, where b and c are places with known location and area. In such a situation, the problem to be solved is exactly an instance of the region approximation problem, with $N(a) = \{b, c\}$ and $R = \{(\mathsf{N}, 1.2, b), (\mathsf{SE}, 2.3, c)\}$.

Algorithms

In this paper, we propose a probabilistic way for approximating a target region based on directional relations and distance information between its neighbours. As mentioned above, the directional relations are qualitative and rough, at least compared to angles that can be used for directions, and the distance information is also vague because of the various ways of measurement and potential error in measured values. The general idea is illustrated in Algorithm 1. First, we need to partition the area of interest into a set of *cells* (smaller regions). The area of interest could be any area that one considers as possible to contain the region being approximated. For each of the cells, we can assign a score to it based on its suitability for the given directional relations and distance information. With the scores, we can then identify a set of cells that form an approximation to the region a.

To obtain a set of cells, here we choose to partition the area of interest into a set of axis-aligned rectangles. This algorithm actually builds a grid of size $n_x \times n_y$ over the area of interest. The cells could also be other shapes, e.g., triangles

Algorithm 1: General procedure of the proposed approach for approximating a region, given direction and distance information tuples.

in : A set of direction and distance information tuples $R = \{(r_j, d_j^*, b_j) : b_j \in N(a)\}$, where $N(a)$ is a set of reference regions of a.

out : A set of cells that are identified as part of the region: A.

```
1 begin
2     C ← getCells(); // partition with cells C
3     foreach cell c_i in C do
4         s(c_i) ← getScore(c_i, R); // calculate the score for each cell
5     A ← identifyCells(C, s(C)); // identify cells form the region
6     return A;
```

generated by the *Delaunlay triangulation* of some points. We note that as long as the cells are small enough, then the shape of the cells will not significantly affect the quality of approximation, although it might affect the efficiency of obtaining an approximation. Here, as a balance between efficiency and quality of approximation, we set the grid size to be $n_x = n_y = 100$. Figure 2 illustrates a partition with grid cells.

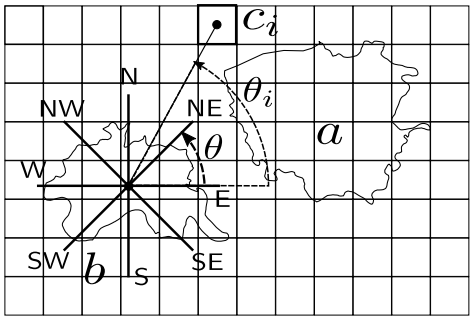


Figure 2. Illustration of constructing cells, and translating qualitative directional relations into angles.

To assign each cell a score, we need to consider how suitable a cell is for the given directional relation and distance information. In particular, to measure directional suitability of a cell, we compare the given directional relation r with the actual directional relation between a cell and a reference region. To this end, we first translate the given directional relation into quantitative angles.

As shown in Figure 2, the eight qualitative relations NW, N, NE, W, E, SW, S, SE are translated to angles of $135°$, $90°$, $45°$, $180°$, $0°$, $225°$, $270°$, $315°$ respectively; this is based on the model of Cardinal Relation Algebra [10]. For example, if the qualitative directional relation between a and b is $r = \mathsf{NE}$, then we use the angle $\theta = 45°$ to interpret r.

To figure out the directional relation between a cell and a reference region b, we consider the line connecting the centroid of the cell and the reference region, as shown in Figure 2. The angle θ_i between this line and the x-axis will be used as the quantitative representation of this directional relation. The directional suitability of a cell is then measured using the difference between those angles, with $\cos(\theta - \theta_i) + 1$. Here we added 1 to the cosine part, because we want to keep the value non-negative. Note that the cos function has a maximum value of 1 at $0°$ and decreases to a minimum value of -1 when the angle goes to either $180°$ or $-180°$. This formula correctly captures the property that when the cell has a similar direction to what is given, then the score of the cell is large (approaching 2), and when the cell has a very different direction to what is given (e.g., opposite direction), then the difference between angles is close to $\pm 180°$, and thus the score is small (approaching 0). Note that although one can use some other linear functions instead of the cos function, cos is simple and gives a smooth change of values. Furthermore, by calculating $\cos(\theta_i - \theta_j)$ using inner products, we can avoid calculating angles using inverse trigonometric functions which could be inaccurate and computationally inefficient.

To measure the distance suitability of a cell, we compare the given distance with the actual distance between a cell and a reference region. The distance between the centroids is used as an approximation to the actual distance. Alternatively, one could also use the distance between "nearest" boundaries, or any other reasonable

distance. Using the distance between centroids can be seen as a compromise between accuracy and efficiency. Moreover, as the given distance usually is also not accurate (due to vagueness of natural language and inaccuracy in measurement), it seems not necessary for the distance between a cell and a reference region to be extremely accurate.

The difference between the given distance d_j^* and the distance d_i for c_i and b_j, is calculated by the (adjusted) relative error of d_i w.r.t. d_j^*, i.e. $\epsilon_i = 2 * ((d_i - d_j^*)/d_j^* + 1)$. Then the distance suitability of a cell c_i is obtained using some function $g(\epsilon_i)$. In order to characterize distance suitability well, the function g should have the following properties:

- when $\epsilon_i = 2$ (i.e. $d_i = d_j^*$), $g(\epsilon_i)$ is the maximum value of g, because in this case c_i has the same distance as the given one, d_j^*.
- when $0 < \epsilon_i < 2$, i.e. c_i is too close to the reference region b_j, $g(\epsilon_i)$ should drop quickly to 0 with ϵ_i approaching 0.
- when $\epsilon_i > 2$, i.e. c_i is too far away from b_j, $g(\epsilon_i)$ should decrease to zero as ϵ_i increases. In this case, g should not drop too quickly, because as d_i becomes larger, the same increment in d_i has less impact on the suitability of c_i than in the previous case.

The probability density function, w.r.t. ϵ_i, of the χ^2-distribution with 4 degrees of freedom possesses all of the above properties. Therefore, the distance suitability of a cell is measured by $\epsilon_i * \mathrm{e}^{-\epsilon_i/2}$.

The final score for a cell c_i is then obtained by multiplying the scores of c_i for each $b_j \in N(a)$. The whole procedure of calculating the final score of a cell is shown in Algorithm 2.

Algorithm 2: getScore(c_i, R)

in : A cell c_i. The set of corresponding direction and distance information tuples $R = \{(r_j, d_j^*, b_j) : b_j \in N(a)\}$.
out : The score of c_i.

```
begin
    s ← 1;
    foreach (r_j, d_j^*, b_j) ∈ R do
        d_i ← distance between centroids of c_i and b_j;
        θ_i ← angle for the line connecting the centroids of c_i and b_j;
        θ_j ← angle translated from the qualitative directional relation r_j;
        ε_i ← 2 * ((d_i − d_j^*)/d_j^* + 1);
        s ← s * (ε_i * e^{−ε_i/2} + (cos(θ_i − θ_j) + 1)/3);
    return s;
```

After assigning each cell a score, we need to identify a subset of cells to form an approximation A to the region a. Suppose that there are n cells and each cell c_i has a score s_i. In order to avoid the side effect of having different scales for our scores, we *normalize* all the scores to be in the range $[0, 1]$ by performing $s_i = s_i/S$, where $S = \sum_{k=1}^{n} s_k$. There are many methods to identify a subset of cells. For example, one can use a *direct* threshold σ, and let $A = \{c_i : s_i \geq \sigma\}$ be the approximation to a_i. Here we use a threshold σ in a different way to filter out cells with low scores. In particular, we use $\beta_i = \sum_{k=1}^{i} x_k/S$ as a *contribution ratio* of x_i, where $(x_1, \cdots, x_n) = \mathsf{sorted}(s_1, \cdots, s_n)$ is a *descendingly sorted* list of scores. Given a threshold σ, suppose j is the smallest index such that $\beta_j \geq \sigma$, then the subset of cells with score x_i ordered before x_j (i.e., $i < j$) is identified as the approximation. The advantage of using a contribution ratio over a direct threshold is that it can distinguish between cells with similar scores. For instance, suppose that there is a list of cells with sorted scores $(0.112, 0.109, 0.101, \dots)$ and the sum S of them is 1.0, and selecting the first two would give the best approximation result. Then we can select the first two cells by using the threshold $\sigma = 0.3$ on the contribution ratio, while in the case of a direct threshold we would have to use some σ like 0.107, which is harder to obtain as it involves searching possible values of σ in more significant digits.

The time complexity of Algorithm 1, with the implementations of getCells, getScore, and identifyCells as described in this article, is $\mathcal{O}(mn)$, where n is the number of cells in C and m is the number of neighbours in $N(a)$. In

particular, line 2 of Algorithm 1 takes $\mathcal{O}(n)$ time, because getCells needs to go through each of the cells once. The *for* loop in lines 3 and 4 needs to iterate n times, as it checks each of the cells in C, and for each iteration, getScore needs to go through each of the neighbours in $N(a)$. In total the *for* loop needs $\mathcal{O}(mn)$ time. Line 5 of Algorithm 1 needs $\mathcal{O}(n)$ time, since it compares each of the scores of the cells to a threshold. Therefore, the whole process of Algorithm 1 needs $\mathcal{O}(n)+\mathcal{O}(mn)+\mathcal{O}(n) = \mathcal{O}(mn)$ time for each region to be approximated. Since the number m of neighbours in $N(a)$ when approximating a is usually a small constant, we can consider the time complexity as $\mathcal{O}(n)$. As an intuition, in our experiments, a region can be approximated within one second on average given 10 000 cells.

Choice of threshold

The choice of a threshold will greatly affect the result of approximation. There are several factors that will affect the appropriate value of a threshold: the function used to calculate scores, the relative distance between the target region and its neighbours, the choice of the area of interest, the size of regions, and many others. We observe that for similar scene settings the threshold will also be similar.

For example, consider the three cases in Figure 3. This observation inspires us to consider the scores as features and make use of machine learning techniques to discover any similarity between them and exploit such information.

Specifically, suppose that there are n cells and each cell c_j has a score s_j. As before, we will normalize these scores and obtain $(x_1, \cdots, x_n) = \mathsf{sorted}(s_1, \cdots, s_n)$, a descendingly sorted list of normalized scores, which is then used as a *feature vector* for applying machine learning techniques to obtain a proper threshold for identifying the cells. One can also apply dimensionality reduction or feature selection techniques, such as Principal Component Analysis, to the list of scores, so that the lengths of lists of scores are the same, and noises in data can be reduced. However, this will not be applied in our evaluations.

To obtain a set of training samples, we need to first get proper thresholds for different feature vectors of scores. The method we use here is to perform a search of a threshold from 0.01 to 1.00.

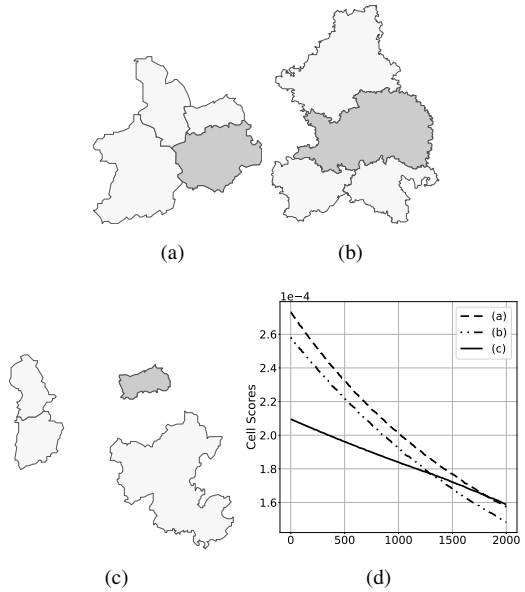


Figure 3. Distribution of cell scores for different tuples. For (a) and (b), the target region has similar size with the neighbouring regions, and the associated distances are not very large, i.e., the target region is not far from any of the neighbouring regions, and, for (c), the size of the target region is smaller than that of any of the neighbouring regions, and the distances are relatively larger too. In (d) we show the first 2 000 largest normalized cell scores for each of the previous three cases. It can be seen that the distributions of cell scores of (a) and (b) are indeed similar, and are also different from that of (c). This suggests that differences in scene settings can be distinguished by monitoring the distribution of cell scores.

To measure the properness of a threshold, we will need to check how accurately A approximates a. The accuracy of the approximation is measured by adapting the well-known Jaccard similarity measure:

$$\mathsf{acc}(A, a) = \frac{\mathsf{area}(A \cap a)}{\mathsf{area}(A \cup a)}. \qquad (1)$$

In this way, for each list of scores, we identify a proper threshold that maximizes $\mathsf{acc}(A, a)$.

The problem of identifying a proper threshold, given a list of scores, is then modelled as a regression problem. We can apply various well-established algorithms to solve this problem. Here, we consider using nearest neighbour regression [11], but other algorithms, e.g., decision tree regression and ridge regression, are also feasible.

The idea of nearest neighbour regression is:

1) Training: record the feature vectors and corresponding thresholds in the training set.
2) Predicting: for a new feature vector, find its k nearest neighbouring feature vectors, with thresholds $\sigma_1, \cdots, \sigma_k$; return the predicted threshold $\sigma = f(\sigma_1, \cdots, \sigma_k)$.

In this way, given a list of cell scores, we obtain a prediction model for appropriate threshold selection. In the next section, we will evaluate how this model performs for different region sets.

Results and discussion

Region sets and settings

Two publicly available sets of regions were considered in the experiments, namely, "Germany" (G), which consists of 435 administrative regions, and "China (subset)" (CS), which consists of 416 administrative regions.[2]

From these two region sets, we prepared different sets of *region pairs* in the form $(a, N(a))$, where a is a target region and $N(a)$ a set of its neighbours. Particularly, we use "[Region-set]_[X]S3" to represent a set of region pairs, where for each region in the corresponding "[Region set]" (CS or G), we consider its top 10 nearest neighbours with respect to the distance of their centroids, and from these 10 neighbours, we *select* the first (F), middle (M), or last (L) three nearest ("[X]" $\in \{\mathrm{F}, \mathrm{M}, \mathrm{L}\}$), or randomly select any three in the 10 neighbours ("[X]" $=$ R), as neighbours in $N(a)$; when "[X]" $=$ RT3, it means that we repeated the random selection of any three of the 10 neighbours for 3 times, which should reduce the effects of statistical noise.

In most of the evaluations in the sequel, for each pair $(a, N(a))$, we will consider the area of interest as the minimum bounding rectangle of the regions in $\{a\} \cup N(a)$, and set the size of the grid to 100×100 unless otherwise specified.

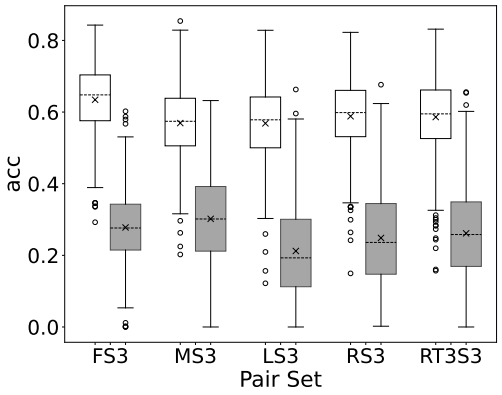
Approximation results

The baseline method to compare is the one (denoted by FanFit) in [12], which is the only comparable method with similar settings. We measure the performance of an approximation algorithm by its accuracy as it is defined in Equation 1. For each pair in a region pair set, we check the accuracy score of an approximation to the target region, obtained by using the direction and distance information from the neighbour regions. For our method, as an approximation is determined by a threshold, for each pair $(a, N(a))$, we first search for an appropriate threshold in the range $[0, 1]$, with a step size of 0.01, and then for the resulting threshold σ_0 with the highest approximation accuracy, we search in the range $[\sigma_0 - 0.05, \sigma + 0.05]$ with a step size of 0.001 to identify a finer threshold. The corresponding approximation accuracy given by the finer threshold of a region pair is then used as the accuracy score for that pair.

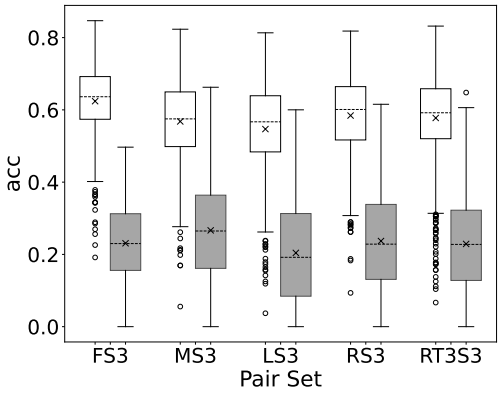
Figure 4 shows the accuracy scores for each of the region pair sets of the region sets CS and G; the results of the method in [12] are coloured gray. As the results between the two region sets are similar, the analysis in the following will focus on the region set CS, yet we note that it also applies to the region set G.

Generally, the average accuracy score of our approximation algorithm is around 0.6, with several approximations having accuracy score as high as about 0.8, while the average score of the baseline (FanFit) is only around 0.2 to 0.3. One should note that an approximation to a region is formed by a set of grid cells, which is relatively rough compared to the original representation of the target region. In fact, with grid cells, even using the *regularization* of a target region a, i.e., $\{c : c \cap a^\circ \neq \emptyset\}$, as the approximation, will only yield an average score of 0.841. Nevertheless, with an average accuracy score of 0.6, an approximation generated by the proposed algorithm on average can cover about 80% of the target region, for all of the pair sets, and about 80% of an approximation coincides with the target region (so, an approximation is similarly sized and shaped). Moreover, as we only have very limited information to approximate a target region, it is natural that the detailed shape of the target region cannot be accurately captured, which results in a lower accuracy score compared to the one provided by regularization. Therefore, considering the circumstances here, the accuracy of the proposed algorithm is already very good, indicating that a large area of the target region is

[2]Germany and China administrative region region sets were derived from GADM (https://gadm.org/data.html).



(a) CS



(b) G

Figure 4. Approximation accuracy boxplots for different region pair sets in the sets of regions CS and G.

covered by the approximation.

To illustrate the effectiveness of the proposed method, for the region set CS, we show the least accurate approximation result, the middle one, and the highest one in Figure 5. Even for the worst result, the approximate region is still close to the actual location, and has similar size to the target region (about 60% of the target region is covered). The middle result represents the most frequent cases in approximation (we remind the reader that the mean and the median values are almost the same), and the approximation is already reasonably good (about 80% of the target region is covered), especially given the fact that only vague distance and direction information is provided.

Effect of uncertain and inconsistent information

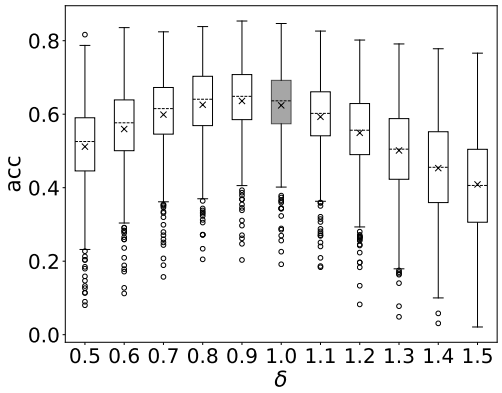
In order to see the effects of different degrees of uncertainty in the information provided for ap-

(a) acc $= 0.157$ (b) acc $= 0.595$ (c) acc $= 0.831$

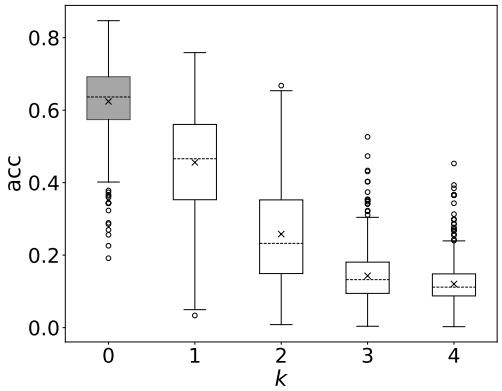
Figure 5. Approximation regions (light gray), compared with target regions (dark gray) and regularization regions (thick boundaries). Neighbour regions are also shown.

proximating a target region, we manually disturb the distance information and relations provided in a tuple (r_j, d_j^*, b_j). In particular, regarding the disturbance of distance information, for each region pair $(a, N(a))$ in the pair set G_FS3, we generate a new set of information tuples $\{(r_j, \delta d_j^*, b_j) : (r_j, d_j^*, b_j) \in R(a)\}$, where $R(a)$ is the original set of tuples used in previous experiments, and δ is a constant value used to disturb the distance information, e.g., $\delta = 1.1$ means the original distance is increased by 10%. In this way, for the disturbance of distance information on G_FS3, we obtained 10 new pair sets by varying the value of δ from 0.5 to 1.5 with a step size of 0.1. As for the disturbance of direction information, we generate a new set of information tuples $\{(f(r_j, k), d_j^*, b_j) : (r_j, d_j^*, b_j) \in R(a)\}$, where f is a function that maps a directional relation r_j to one of its k-neighbour relations, e.g., the 1-neighbour relations of N are NW and NE. In this way, for the disturbance of distance information on G_FS3, we obtained 4 new pair sets by varying the value of k from 1 to 4 with a step size of 1 and randomly choosing one in the k-neighbour relations. The results of optimal approximation accuracies for these pair sets are shown in Figure 6.

It can be seen from the figure that the accuracy of approximation will generally decrease when the provided information is less accurate. Also, the performance does not drop dramatically when the disturbance is minor, as we can see from the results around the original one (coloured gray). It is worth noting that for the case of distance disturbance, when the disturbance is significant (e.g., $\delta = 1.5$), the best approximation accuracies are still around 0.8, which, to some extent, demonstrates the robustness of the proposed ap-



(a) disturbance of distances



(b) disturbance of directions

Figure 6. Approximation accuracy changes with respect to the disturbance of distances and directions.

proach w.r.t. distance disturbance. As for direction disturbance, for inconsistent information, it is expected that the performance of approximation will decrease to some extent. For example, for a target region a with three neighbouring regions b_1, b_2, b_3, and the corresponding directional relations $\mathsf{NE}, \mathsf{SE}, \mathsf{SW}$ (Figure 7), we change the relation between a and b_2 from SE to NW, which introduces significant inconsistencies with respect to other relations too. The approximation accuracy then decreased from 0.659 to 0.313.

This problem could be tackled by introducing methods for discovering inconsistency. For instance, in the example above, the grid cells in a will get very low scores from the information provided by b_2 while getting very high scores from the information provided by b_1 and b_3. One can set up some rules to detect such significant discrepancy and thus detect and remove inconsistency.

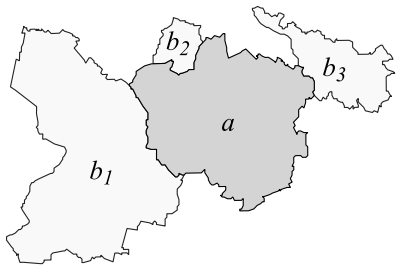


Figure 7. A case to illustrate the effect of inconsistent knowledge.

In summary, the aforementioned results show that the approximation algorithm works and has promising performance to obtain good approximations to target regions, even when only rough direction and distance information is available.

Learning a threshold

Note that in the former evaluations, an appropriate threshold was chosen by searching through a list of candidate values and comparing the resulting accuracy scores. In practice, this way of determining a threshold will not be applicable due to two reasons. First, it requires the geometric information of the target region, which is usually not known. Second, the search is time-consuming, and cannot be used for applications where time is critical. Therefore, we propose to exploit machine learning techniques (a 3-NN regressor, specifically) to automatically assign an appropriate threshold for a given region pair.

We measured the performance using the following absolute error for predicting a threshold:

$$E_{\text{thres}} = |\sigma - \sigma^*|,$$

where σ is the predicted threshold for a pair, and σ^* is the corresponding ground-truth threshold.

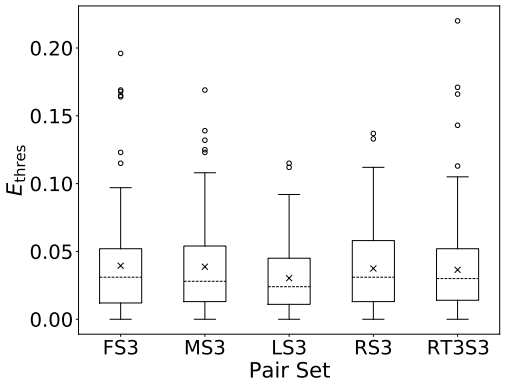
We also calculated the approximation accuracy for each of the predicted thresholds, using the following absolute error regarding approximation accuracy:

$$E_{\mathsf{acc}} = |\mathsf{acc}_{\mathrm{p}} - \mathsf{acc}^*|,$$

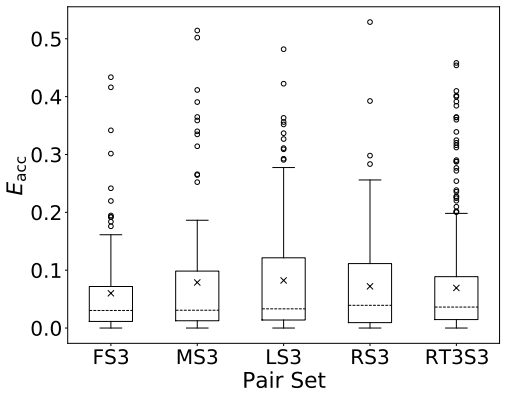
where $\mathsf{acc}_{\mathrm{p}}$ is the approximation accuracy given by the predicted threshold for a pair, and acc^*

is the corresponding ground-truth approximation accuracy.

Figure 8 (a) shows the results for predicting a threshold in different region pair sets in the region set CS. Overall, the prediction of threshold



(a) boxplot of E_{thres}



(b) boxplot of E_{acc}

Figure 8. Prediction results: boxplots showing the distribution of E_{thres} and E_{acc} for different region pair sets in the region set CS.

is accurate, with most of the cases having an absolute error of less than 0.05. By comparing the first three boxes, we can see that when the distance of neighbours to the target region becomes relatively larger, the prediction of threshold gets more accurate: for the three nearest neighbours, the average absolute error is 0.046; for the three middle nearest neighbours, the average absolute error is 0.039, decreased by 0.006; for the three least nearest neighbours, it becomes 0.028. The last box illustrates a more general situation, with an average absolute error of 0.037. The median values are smaller than the corresponding mean values in all of the boxes, meaning that half of the cases have a better performance than the average. Actually, as we can see from the boxplots, the larger mean values are due to some rare outliers with large absolute errors. This illustrates that the machine learning technique can predict a threshold that is close to the most appropriate one.

From Figure 8 (b), it can be seen that the approximation accuracy obtained by using the predicted thresholds can be very close to the ground-truth, with an average absolute error of around 0.082 and a median one of around 0.041 for CS_RT3S3. Again, most cases have a small absolute error and there are some very rare outliers with larger absolute errors. These results indicate that the predicted thresholds are usually close to optimal, and the machine learning technique for predicting thresholds works well.

Conclusion and future work

In this article, we developed a novel approach for generating approximations to regions given some vague direction and distance information. This approach is shown to perform well, in that relatively accurate approximations can be efficiently found, and an important parameter, namely, the threshold used to identify candidate cells to form an approximation, can be reliably predicted with the help of machine learning regression algorithms.

Regarding future work, it would be interesting to further explore how to deal with noisy, contradictory, or missing knowledge in practice, either via other data-driven techniques or a fuzzification of available knowledge, or even via an interplay of both, and how this approach can be applied to practical scenarios like mobile robot navigation.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 61806170; the Humanities and Social Sciences Fund of Ministry of Education under Grant 18XJC72040001; the National Key Research and Development Program of China under Grant 2019YFB1706104; and the Fundamental Research Funds for the Central Universities under Grant 2682018CX25.

REFERENCES

1. M. J. Egenhofer, "Query processing in spatial-query-by-sketch," *Journal of Visual Languages & Computing*, vol. 8, no. 4, pp. 403–424, 1997.
2. J. Kim, M. Vasardani, and Stephan Winter, "From descriptions to depictions: A dynamic sketch map drawing strategy," *Spatial Cognition & Computation*, vol. 16, no. 1, pp. 29–53, 2015.
3. J. Grant, C. Molinaro, and F. Parisi, "Probabilistic spatio-temporal knowledge bases: Capacity constraints, count queries, and consistency checking," *International Journal of Approximate Reasoning*, vol. 100, pp. 1–28, 2018.
4. A. A. A. Alkhatib, "A Review on Forest Fire Detection Techniques," *International Journal of Distributed Sensor Networks*, vol. 10, no. 3, p. 597368, 2014.
5. Avi Arampatzis, Marc van Kreveld, Iris Reinbacher, Christopher B. Jones, Subodh Vaid, Paul Clough, Hideo Joho, and Mark Sanderson, "Web-based delineation of imprecise regions," *Computers, Environment and Urban Systems*, vol. 30, no. 4, pp. 436–459, 2006.
6. A. Galton and M. Duckham, "What Is the Region Occupied by a Set of Points?" in *International Conference on Geographic Information Science*, 2006, pp. 81–98.
7. O. Şeref and C. W. Zobel, "Recursive voids for identifying a nonconvex boundary of a set of points in the plane," *Pattern Recognition*, vol. 46, no. 12, pp. 3288–3299, 2013.
8. X. Zhong and M. Duckham, "Characterizing the shapes of noisy, non-uniform, and disconnected point clusters in the plane," *Computers, Environment and Urban Systems*, vol. 57, pp. 48–58, 2016.
9. F. A. Twaroch, P. Brindley, P. D. Clough, C. B. Jones, R. C. Pasley, and S. Mansbridge, "Investigating behavioural and computational approaches for defining imprecise regions," *Spatial Cognition & Computation*, vol. 19, no. 2, pp. 146–171, 2019.
10. G. É. Ligozat, "Reasoning about cardinal directions," *Journal of Visual Languages & Computing*, vol. 9, no. 1, pp. 23–44, 1998.
11. G. Biau and L. Devroye, "The nearest neighbor regression function estimate," in *Lectures on the Nearest Neighbor Method*. Springer International Publishing, 2015, pp. 95–103.
12. X. Zhang and S. Zhu, "Contextual spatial relations based spatial modelling of vague place names," in *2010 Second IITA International Conference on Geoscience and Remote Sensing*, 2010, pp. 23–26.

Zhiguo Long is a lecturer with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, Sichuan, China. He received the Ph.D. degree from University of Technology Sydney, Australia, in 2017. His research interests include fundamental and practical techniques in qualitative spatial and temporal reasoning, and representation problems in machine learning and computer vision.

Michael Sioutis is working as a Research Fellow at the University of Bamberg, Germany, in the area of Computer Science since 2020. He received his Ph.D. degree from Artois University, France, in February of 2017. His general interests lie in Artificial Intelligence, Data Mining, Linux System Engineering, Logic Programming, and Semantic Web.

Qingqian Li is currently a Master student with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, Sichuan, China. She received the B.Sc. degree from Southwest Minzu University in 2020. Her research interests include qualitative spatial and temporal information in Geographical Information Science.

Hua Meng currently works at the School of Mathematics, Southwest Jiaotong University, Chengdu, Sichuan, China. He received his Ph.D. degree in mathematics from Sichuan University, China in 2010. His research interests include knowledge representation and reasoning, machine learning, and general topology.

Heng-Chao Li is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan, China. He received his Ph.D. degree from the Graduate University of the Chinese Academy of Sciences, Beijing, China in 2007. He is a senior member of IEEE. He serves as an Associate Editor for the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. His research interests include the statistical analysis of synthetic aperture radar images, remote sensing image processing, and signal processing in communications.