Computational Geometry and Heuristic Approaches for Location Problems

M. Šeda
Institute of Automation and Computer Science
Faculty of Mechanical Engineering, Brno University of Technology
Czech Republic

***Abstract*—In this paper we deal with two problems, whose common basis is to find the location of a service center for potential customers, but with different criterion functions, determining what we consider in these tasks as optimal. While maximizing the coverage of an area by supermarkets, we choose a new supermarket the location that minimises interaction (and thus competition) with existing supermarkets. On the contrary, if we want to provide the availability of certain services for all customers within a reasonable distance, and yet we know in advance where it would be possible to set up servicing points, the goal is to minimize their number. We show that the first type of problem can be solved in polynomial time using the Voronoi diagram, the task of the second type leads to the set covering problem, which is an NP-hard problem, and it is therefore necessary to solve larger instances of a task by heuristics. It is proposed we use a genetic algorithm approach and special attention is paid to implementation of a repair operator for infeasible solutions generated by the operations of crossover and mutation.**

Keywords-computational geometry; location-based service (LBS)

I. INTRODUCTION

One of the basic considerations of retail chains is to cover the area where it can be expected the demand for their goods, but so that any new shop, was as close to customers who have existing shops too far and on the other hand, new shop was far from existing ones not to compete with themselves.

The same considerations require the tasks when we need to obtain a large, contiguous, undeveloped piece of land on which to build a factory, or to find the new site for a source of pollution.

Another possible view on the customer serviceability is given by the maximum distance which still can be considered for shop (or other facility, such as a school, hospital) reachable sufficiently close to be able to satisfy the requested service. Locations, where the service centers could be situated, and we try to find a minimum number of service centers for providing services in reachable distance for all customers

Both aspects bring various possibilities of solving and we deal with them in more detail in the following paragraph.

II. COMPUTATIONAL GEOMETRY APPROACH

A. Voronoi Diagrams

A *Voronoi diagram* of a set of sites in the plane is a collection of regions that divide up the plane. Each region corresponds to one of the sites and all the points in one region are closer to the site representing the region than to any other site (Aurenhammer 1991, de Berg et al. 2000, LaValle 2006, Okabe et al. 2000, Šeda 2007) [1-10]. An example of the Voronoi diagram is shown in Figure 1.

Let $d(p_i , p_j)$ denote the distance between two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ in the plane and consider the *Euclidean metric*

$$d(p_i,p_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \tag{1}$$

Then more formally, we can define Voronoi diagrams in mathematical terms.

Definition: Let $P=\{p_1, p_2, \dots , p_n\}\subset \Re^2$ be a set of points with the Cartesian coordinates $(x_1, y_1), \dots , (x_n, y_n)$ where $2<n<\infty$ and $p_i \neq p_j$ for $i \neq j$. We call the region

$$V(p_i)=\left\{ x\in \Re^2 \mid d(x_i,p_i)\le d(x_j,p_j)\right\} \tag{2}$$

the *planar Voronoi polygon* associated with p_i (or the Voronoi polygon of p_i) and the set given by

$$V=\{V(p_1),\dots V(p_n)\} \tag{3}$$

the *planar Voronoi diagram* generated by P (or the Voronoi diagram of P). We call p_i of $V(p_i)$ the *site* or *generator point* or *generator* of the i-th Voronoi polygon and the set $P=\{p_1, p_2, \dots , p_n\}$ the *generator set* of the Voronoi diagram V. Hence we get

$$\begin{aligned} V(P) &= \bigcup_{p_i\in P} V(p_i) = \\ &= \bigcup_{p_i\in P}\left\{x\in \Re^2 \mid d(x,p_i)\le d(x,q):\forall q\in (P-\{p_i\})\right\}= \\ &= \bigcup_{p_i\in P}\left[\bigcap_{q\in P-\{p_i\}}\left\{x\in \Re^2 \mid d(x,p_i)\le d(x,q)\right\}\right] \end{aligned} \tag{4}$$

Assume that Voronoi diagrams are non degenerate (no four or more of its Voronoi edges have a common endpoint. Then it is satisfied:

– Every vertex of a Voronoi diagram $V(P)$ is a common intersection of exactly three edges of the diagram.

– A point q is a vertex of $V(P)$ if and only if its *largest empty circle* $C_P(q)$ contains three points on its boundary.

– The bisector between points p_i and p_j defines an edge of $V(P)$ if and only if there is a point q such that $C_P(q)$ contains both p_i and p_j on its boundary but no other point.

– For any q in P, $V(q)$ is convex.

– Voronoi diagram $V(P)$ of P is planar.

– Polygon $V(p_i)$ is unbounded if and only if p_i is a point on the boundary of convex hull of the set P (*convex hull* of a set P is the smallest convex set that contains P).

– The number of vertices in the Voronoi diagram of a set of n point sites in the plane is at most $2n–5$ and the number of edges is at most $3n–6$.

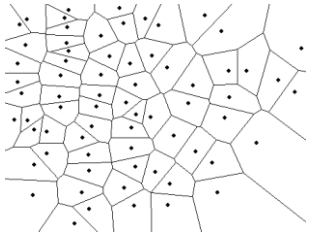

FIGURE I. VORONOI DIAGRAM.

The fundamental algorithms and their modifications include the *incremental algorithm*, *random incremental algorithm*, *divide and conquer algorithm* and *plane sweep algorithm* (or *Fortune's algorithm*). More details can be found e.g. in (Aurenhammer 1991, de Berg et al. 2000, Okabe et al. 2000) [1,2,6]. The time complexity of the incremental algorithm is $O(n^2)$ in the worst case, and $O(n \log n)$ for the other three algorithms.

B. Largest Empty Circle

From the previous properties the most interesting is that concerning the empty circle.

Given n points in the plane, find a largest circle containing no points of the set yet whose center is interior to the convex hull.

The largest empty circles among all empty circles can be found as follows:

1. Compute Voronoi diagram.
2. Compute convex hull.
3. Find all vertices of the Voronoi inside the convex hull.
4. Find all circles in these vertices.
5. Select the largest circle.

Since the time complexity of the first two steps is $O(n \log n)$ (Aurenhammer 1991, de Berg et al. 2000) [1,2] and all other are simpler, the total time complexity of this algorithm is $O(n \log n)$.

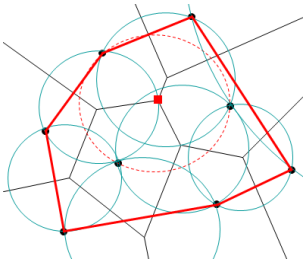

FIGURE II. LARGEST EMPTY CIRCLE.

In Figure 2, points are existing positions of the service center network, the boundary drawn by a thick line is the convex hull, and square is the center of the largest empty circle, determined among the Voronoi vertices that lie inside the convex hull and it is the best place to build a new service center.

C. How to Adapt the Previous Solution

The drawback of the previous approach is that it does not take into account whether a location that is determined by calculation as optimal, can be used for a new service center. It can be occupied by another building or there is no communication to this place, or it is owned by someone who does not want to sell the ground. In these cases it is necessary to move on the nearest possible place.

Another problem is determining of distances. In previous definition of Voronoi diagram, we consider the Euclidean metric. In a real situation, because of other buildings, the distance of two points is not given by direct line, but the road between them. In cities like New York, partly Barcelona, where the streets create the rectangular system, it would be appropriate to use rectilinear metric. The traditional Euclidean Voronoi diagram then changes in rectilinear Voronoi diagram and its shape is significantly modified. changed. It contains edges only in horizontal, vertical and diagonal direction

Consider *rectilinear* (or *Manhattan*) *metric*

$$d(p_i,p_j)=|x_i-x_j|+|y_i-y_j| \qquad (5)$$

If we use the rectilinear metric for a Voronoi diagram, then, due to the rectilinearity, each straight-line segment of a bisector in the now rectilinear Voronoi diagram will be either horizontal, vertical, or inclined at $45°$ or $135°$ to the positive direction of the x-axis

Another way how to determine distances which correspond to real situation is to use GPS systems or a specialized program, such a route planner Trackroad that is available from http://www.trackroad.com.

III. SET COVERING PROBLEM (SCP)

A. Dmax Cover

Assume that the transport network contains m vertices, that can be used as operating service centers, and n vertices to be served, and for each pair of vertices v_i (considered as service centers) and v_j (serviced vertex) their distance d_{ij} is given and D_{max} is the maximum distance which will be accept for operation between the service centers and serviced vertices.

The aim is to determine which vertices must be used as service centers so that each vertex was covered by at least one of the centers and the total number of operating centers was minimal.

Note:

1. A necessary condition for solvability of the task is that all of the serviced vertices were reachable from at least one place where an operating service center is considered.

2. Serviced vertex v_j is reachable from vertex v_i, which is considered as an operating service center if $d_{ij} \le D_{max}$. If this inequality is not satisfied, vertex v_j is unreachable from v_i.

If variables a_{ij}, where a_{ij}=1 or a_{ij}=0, express whether operated vertex v_j is reachable from vertex v_i, which is considered as operating service center, respectively. is not reachable, then the set covering problem can be described by the following mathematical model:

Minimise

$$z = \sum_{i=1}^{m} y_i \tag{6}$$

subject to

$$\sum_{i=1}^{m} a_{ij} y_i \ge 1, \quad j = 1, \ldots, n \tag{7}$$

$$y_i \in \{0,1\}, \quad i = 1, \ldots, m \tag{8}$$

The objective function represents the number of operating centers, constraint (7) means that each serviced vertex is assigned at least one operating service center.

Example:

Consider the following *distance matrix* which expresses service centers and serviced vertices (= customer locations) and D_{max}=40.

serviced vertices (customers locations)

service centers	1	2	3	4	5	6	7	8
1	5	41	50	26	38	60	44	59
2	49	82	13	67	68	20	32	31
3	45	17	61	45	67	48	53	127
4	37	170	195	32	77	88	90	30
5	58	42	25	101	133	32	21	78

From D_{max}=40 we get the *reachability matrix* of serviced vertices from service centers.

	1	2	3	4	5	6	7	8
1	1	0	0	1	1	0	0	0
2	0	0	1	0	0	1	1	1
3	0	1	0	0	0	0	0	0
4	1	0	0	1	0	0	0	1
5	0	0	1	0	0	1	1	0

Since only service center 3 is reachable to the second serviced vertex (serviced vertex 2 is covered by the 3rd service center) and only service center 1 is reachable to service center 5, these service centers must not be omitted. These two centers cover serviced vertices 1, 4, 5 and 2

It remains to find the service centers which cover the remaining uncovered vertices 3, 6, 7 and 8. This can be achieved either by selecting the service centers 2 and 5, or 4 and 5.

Thus the example has two solutions, where four vertices are sufficient to cover serviced vertices (either 1, 3, 2, 5, or 1, 3, 4, 5).□

Note:

In the general case, however, the selection of service centers for k uncovered vertices has 2^k possibilities and thus the complexity of tasks increases exponentially with the number of uncovered vertices.

For large k we must solve it using a heuristic method (Galinier & Hertz 2007, Michalewicz & Fogel 2004, Yagiura et al. 2006, Zelinka et al. 2012)[3,5,9,10], e.g., by a genetic algorithm.

IV. GENETIC ALGORITM FOR SCP

The skeleton for GA can be described as follows (Michalewicz & Fogel 2004)[5]:

generate an initial population ;

evaluate fitness of individuals in the population ;

repeat

  select parents from the population;

  recombine (mate) parents to produce children ;

  evaluate fitness of the children ;

  replace some or all of the population by the children

until a satisfactory solution has been found ;

Since the principles of GAs are well-known, we will only deal with GA parameter settings for the problems to be studied. Now we describe the general settings (Michalewicz & Fogel 2004, Zelinka et al. 2012)[5,10].

Individuals in the population (*chromosomes*) are represented as binary strings of length n, where a value of 0 or

1 at the *i*-th bit (*gene*) implies that *xi* = 0 or 1 in the solution respectively.

The *population size N* is usually set between *n* and 2*n*. Many empirical results have shown that population sizes in the range [50, 200] work quite well for most problems.

Initial population is obtained by generating random strings of 0s and 1s in the following way: First, all bits in all strings are set to 0, and then, for each of the strings, randomly selected bits are set to 1 until the solutions (represented by strings) are feasible.

The *fitness function* corresponds to the objective function to be maximised or minimised.

There are three most commonly used methods of *selection* of two parent solution for *reproduction*: proportionate selection, ranking selection, and tournament selection. The tournament selection is perhaps the simplest and most efficient among these three methods. We use the *binary tournament selection* method where two individuals are chosen randomly from the population. The more fit individual is then allocated a reproductive trial. In order to produce a child, two binary tournaments are held, each of which produces one parent.

The *recombination* is provided by the *uniform crossover* operator, which has a better recombination potential than do other crossover operators as the classical *one-point* and *two-point* crossover operators. The uniform crossover operator works by generating a random crossover mask *B* (using Bernoulli distribution) which can be represented as a binary string $B = b1b2b3 \cdots bn\text{-}1bn$ where *n* is the length of the chromosome. Let *P*1 and *P*2 be the parent strings *P*1[1], **...** ,*P*1[*n*] and *P*2[1], **...** ,*P*2[*n*] respectively. Then the child solution is created by letting: *C*[*i*] = *P*1[*i*] if *bi* = 0 and *C*[*i*] = *P*2[*i*] if *bi* = 1. *Mutation* is applied to each child after crossover. It works by *inverting M* randomly chosen bits in a string where *M* is experimentally determined. We use a mutation rate of 5/*n* as a lower bound on the optimal mutation rate. It is equivalent to mutating five randomly chosen bits per string.

When *v* child solutions have been generated, the children will replace *v* members of the existing population to keep the population size constant, and the reproductive cycle will restart. As the replacement of the whole parent population does not guarantee that the best member of a population will survive into the next generation, it is better to use *steady-state* or *incremental replacement* which generates and replaces only a few members (typically 1 or 2) of the population during each generation. The *least-fit* member, or a randomly selected member with *below-average fitness*, are usually chosen for replacement.

Termination of a GA is usually controlled by specifying a maximum number of generations $t_{\max}$ or relative improvement of the best objective function value over generations. Since the optimal solution values for most problems are not known, we choose $t_{\max} \leq 5000$.

The chromosome is represented by an *n*-bit binary string *S* where *n* is the number of columns in the SCP. A value of 1 for the *j*-th bit implies that column *j* is in the solution and 0 otherwise.

Since the SCP is a minimisation problem, the lower the fitness value, the more fit the solution is. The fitness of a chromosome for the unicost SCP (Šeda et al. 2014)[]7,8] is calculated by (9).

$$f(S) = \sum_{j=1}^{n} S[j] \qquad (9)$$

As to the crossover operation, we can use the traditional two-point crossover, where middle parts of the parent chromosomes are changed.

For mutation we considered three operators:

– *exchange mutation* (it exchanges two randomly selected positions in a permutation),

– *shift mutation* (it removes a value at one position and puts it at another position), and*mutation* inspired by well-known *Lin*-2-*Opt change operator* usually used for solving the travelling salesman problem (Šeda et al. 2014)[7,8]. Here first two elements are added to the permutation (into positions 0 and |*n*|+1) and then the same values are assigned to them to simulate a cyclic tour. Two 'edges' (pairs of neighbour elements) are randomly chosen ((p_1, p_2) and (q_1, q_2) say), the inner elements p_2, q_1 are swapped and the elements between p_2 and q_1 are reversed.

The binary representation causes problems with generating infeasible chromosomes, e.g. in initial population, in crossover and/or mutation operations. To avoid infeasible solutions a *repair operator* is applied.

Let $I = \{1, \ldots, m\}$ = the set of all rows;

$J = \{1, \ldots, n\}$ = the set of all columns;

$\alpha_i = \{j \in J \mid a_{ij} = 1\}$ = the set of columns that cover row *i*, $i \in I$;

$\beta_j = \{i \in I \mid a_{ij} = 1\}$ = the set of rows covered by column *j*, $j \in J$;

S = the set of columns in a solution;

U = the set of uncovered rows;

w_i = the number of columns that cover row *i*, $i \in I$ in *S*.

The repair operator for the unicost SCP has the following form:

initialise $w_i := |S \cap \alpha_i|, \forall i \in I$;

initialise $U := \{ i \mid w_i = 0, \forall i \in I \}$;

for each row *i* in *U* (in increasing order of *i*) **do**

begin find the first column *j* (in increasing order of *j*)

in α_i that minimises $1/|U \cap \beta_j|$;

$S := S + j$;

$w_i := w_i + 1, \quad \forall i \in \beta_j$;

$U := U - \beta_j$

end ;

for each column j in S (in decreasing order of j) **do**

if $w_i \geq 2$, $\forall i \in \beta_j$

then begin $S := S - j$;

$w_i := w_i - 1, \quad \forall i \in \beta_j$

end ;

{ S is now a feasible solution to the SCP and contains no redundant columns }

Initialising steps identify the uncovered rows. **For** statements are "greedy" heuristics in the sense that in the 1st **for**, columns with low cost-ratios are being considered first and in the 2nd **for**, columns with high costs are dropped first whenever possible.

V. Conclusions

In this paper, we show how to use Voronoi diagram in a situation where we want to add to new service centers to the existing ones. The procedure can also be used in the case of construction, which can be a burden for a given area, such as the construction of nuclear power plants or municipal waste dump. The calculation requires only polynomial time, but the resulting location may need a modification.

Conversely, if potentially useable locations for service centers are given in advance, then we try to minimise their number respecting reachable distance of at least one service center for each element of the system. This this leads to the set covering problem and has exponential time complexity. In the paper, we proposed a genetic algorithm approach for its solving.

Particular attention was paid to the repair operator, because traditional methods of crossover and mutation implementation do not guarantee that generated children would create feasible solutions and penalising them would lead to population with high number of infeasible individuals and the chance to find a good solution would be almost impossible.

References

[1] Aurenhammer, F. 1991. Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys, 23(3): 345-405.

[2] de Berg, M., van Kreveld, M., Overmars M. & Schwarzkopf, O. 2000. Computational Geometry: Algorithms and Applications. Berlin: Springer-Verlag,.

[3] Galinier, P. & Hertz, A. 2007. Solution Techniques for the Large Set Covering Problem. Discrete Applied Mathematics 155(3): 312-326.

[4] LaValle, S.M. 2006. Planning Algorithms. Cambridge: University Press.

[5] Michalewicz, Z. & Fogel, D. B. 2004. How to Solve It: Modern Heuristics. Berlin: Springer-Verlag.

[6] Okabe, A., Boots, B., Sugihara, K. & Chiu, S.N. 2000. Spatial Tessellations and Applications of Voronoi Diagrams. New York: John Wiley & Sons.

[7] Šeda, M. 2007. Comparison of Roadmap and Cell Decomposition Methods in Robot Motion Planning. WSEAS Transactions on Systems and Control 2(2): 101-108.

[8] Šeda, M., Roupec, J. & Šedová, J. 2014. Transportation Problem and Related Tasks with Application in Agriculture. International Journal of Applied Mathematics and Informatics. 8(1): 26-33.

[9] Yagiura, M., Kishida, M. & Ibaraki, T. 2006. A 3-flip neighborhood local search for the set covering problem. European Journal of Operational Research 172(2): 472-499.

[10] Zelinka, I., Snášel, V. & Abraham, A. (eds.). 2012. Handbook of Optimization. From Classical to Modern Approach. Berlin: Springer-Verlag.