Yugoslav Journal of Operations Research
1 (1991), Number 2, 121–134

GEOMETRICAL/GEOGRAPHICAL OPTIMIZATION AND FAST AUTOMATIC DIFFERENTIATION

Masao IRI

Faculty of Engineering, University of Tokyo, Bunkyo-ku, Tokyo 113, Japan

Koichi KUBOTA

Faculty of Science and Technology, Keio University, Kohoku-ku, Yokohama, Kanagawa 223, Japan

Kazuo MUROTA

Faculty of Engineering, University of Tokyo, Bunkyo-ku, Tokyo 113, Japan

Abstract. With the geographical/geometrical optimization problem as an example, it is shown that "fast automatic differentiation", a graph-theory based computational technique, will resolve a number of difficulties of different kinds which we confront in the solution of large complicated continuous optimization problems.

Key words and phrases: geographical optimization, fast automatic differentiation, Voronoi diagram, facility location, consistent optimization algorithm

1. INTRODUCTION

Combinatorial approaches and tools are useful not only for discrete optimization problems but also for continuous optimization problems, as has been evidenced in many instances. For example, combinatorial properties of linear programming theory have long been recognized; use of combinatorial approaches with fixed-point theorems, simplicial decomposition, homotopy etc. in numerical solutions of continuous optimization problems and nonlinear equations are well known; and combinatorial methods are essential for preliminary structural analysis of specially structured large-scale continuous systems [2], [13].

In this paper, we deal with another aspect of the rôle of combinatorial concepts and methods in an originally continuous optimization problem called the geographical/geometrical optimization problem [4]. Since the problem is geometrical, it is natural that topological concepts should play an important rôle, so that the importance of the newly developing field of computational technology called

"fast automatic differentiation" [1], [3], [5], [6], [9], [10] will be particularly emphasized in the following. (Fast automatic differentiation is nothing but a graphical technique for manipulating computational processes.)

2. GEOMETRICAL/GEOGRAPHICAL OPTIMIZATION

2.1. Formulation

We will consider the following problem of optimum location of facilities. (For a family of related problems called geometrical/geographical optimization problems, see [4].) The unit square S in the 2-dimensional Euclidean plane is densely inhabited with the distribution function $\varphi(\boldsymbol{x})$ of inhabitants ($\boldsymbol{x} = (x, y) \in S$). There are a given number N of facilities, all of the same kind, in S. Each inhabitant uses the nearest facility regularly, say 5 times a month, but some cost is incurred at each time of use and we assume that the cost is a known function $f(t^2)$ of the distance t between the place where an inhabitant lives and that where the nearest facility is located.

The problem is how to determine the locations of those N facilities in the optimum way, optimum in the sense that the total cost incurred be minimum.

The mathematical expression of the problem is quite simple. It is an unconstrained minimization problem in $2N$ variables $\{(x_i, y_i) \mid i = 1, \ldots, N\}$ ($\boldsymbol{x}_i = (x_i, y_i)$): the location of the i-th facility; $\boldsymbol{X} \equiv (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$: the location vector with $2N$ components) with the objective function expressed in the form of the integral:

$$F(\boldsymbol{X}) = F(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \int_S f(\min \|\boldsymbol{x} - \boldsymbol{x}_i\|^2)\varphi(\boldsymbol{x})\, \mathrm{d}\,\boldsymbol{x}, \tag{2.1}$$

where the minimum in the integrand function is taken over the N points $\boldsymbol{x}_i$ for each $\boldsymbol{x}$. The expression is merely theoretical, because the minimum must be taken at infinitely many points $\boldsymbol{x}$.

2.2 Reformulation of the Problem in Terms of the Voronoi Diagram

In order to compute the objective function F, it is natural to consider the partition of the unit square S into the territories V_i of the facilities defined by

$$V_i = \{\boldsymbol{x} \in S \mid \|\boldsymbol{x} - \boldsymbol{x}_i\| < \|\boldsymbol{x} - \boldsymbol{x}_j\|, j \neq i\}, \qquad i = 1, \ldots, N, \tag{2.2}$$

where, obviously,

$$V_i \cap V_j = \varnothing, \qquad i \neq j, \tag{2.3}$$

$$\bigcup_{i=1}^{N} \overline{V_i} = S \qquad (\overline{V_i} \text{ indicates the closure of } V_i), \tag{2.4}$$

and we set

$$W_{ij} = \mathrm{Int}(\overline{V_i} \cap \overline{V_j}), \tag{2.5}$$

$$U_{ijk} = \overline{V_i} \cap \overline{V_j} \cap \overline{V_k}. \tag{2.6}$$

The territory of $\boldsymbol{x}_i$ is called the *Voronoi region* V_i; the common boundary of two adjacent V_i and V_j is called the *Voronoi edge* W_{ij}; and if three territories V_i, V_j and V_k meet at a point, the point is called the *Voronoi point* U_{ijk}. This partition into the territories of $\boldsymbol{x}_i$'s or the corresponding two-dimensional combinatorial-topological complex is called the *Voronoi diagram*. Then, it is obvious that the integral (2.1) can be rewritten in a more practical form:

$$F(\boldsymbol{X}) = F(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \sum_{i=1}^{N} \int_{V_i} f(\|\boldsymbol{x} - \boldsymbol{x}_i\|^2)\varphi(\boldsymbol{x})\, \mathrm{d}\,\boldsymbol{x}. \tag{2.7}$$

Here, it should be noted that the variables $\boldsymbol{x}_i = (x_i, y_i)$, $i = 1, \ldots, N$, appear on the right-hand side of (2.7) not only as the arguments of the integrand function $f(\cdot)$ but the domains of integration V_i also depend on $\boldsymbol{x}_i$'s.

3. A. BRUTE-FORCE SOLUTION

It seems that no practically useful analytical solution is available to the problem except in the cases where N has a special value and φ has a special form. Furthermore, the uniqueness of solution obviously does not hold in general, so that F is not a convex function, nor is it continuously differentiable in general. Therefore, we may give up looking for any theoretically elegant approach to the problem but may be contented with a brute-force iterative numerical method to get a local minimum of F.

First, we note that the gradient of F looks like

$$\frac{\partial F}{\partial x_i^\lambda} = \int_{V_i} (x_i^\lambda - x^\lambda) f'(\|\boldsymbol{x}_i - \boldsymbol{x}\|^2)\varphi(\boldsymbol{x})\, \mathrm{d}\,\boldsymbol{x} \qquad (\lambda = 1, 2) \tag{3.1}$$

and the Hessian like

$$\frac{\partial^2 F}{\partial x_i^\kappa \partial x_j^\lambda} = \begin{cases} H_{\lambda\kappa}^i + G_{\lambda\kappa}^i & \text{if } j = i, \\ G_{\lambda\kappa}^{ji} & \text{if } V_i \text{ and } V_j \text{ are adjacent } (i \neq j), \\ 0 & \text{if } V_i \text{ and } V_j \text{ are not adjacent } (i \neq j), \end{cases} \tag{3.2}$$

$$H_{\lambda\kappa}^i = \int_{V_i} [\delta_{\lambda\kappa} f'(\|\boldsymbol{x}_i - \boldsymbol{x}\|^2) + 2(x_i^\kappa - x^\kappa)(x_i^\lambda - x^\lambda) f''(\|\boldsymbol{x}_i - \boldsymbol{x}\|^2)]\varphi(\boldsymbol{x})\, \mathrm{d}\,\boldsymbol{x},$$

$$G_{\lambda\kappa}^{ji} = K_{\lambda\kappa}^{jji}, \qquad G_{\lambda\kappa}^{i} = -\sum_{j:V_j \text{ adjacent to } V_i} K_{\lambda\kappa}^{iji},$$

$$K_{\lambda\kappa}^{kji} = \int_{W_{ij}} \frac{(x_i^\kappa - x^\kappa)(x_k^\lambda - x^\lambda)}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|} f'(\|\boldsymbol{x}_i - \boldsymbol{x}\|^2)\varphi(\boldsymbol{x})\, \mathrm{d}\,\boldsymbol{x},$$

where $x_i^1 = x_i$ and $x_i^2 = y_i$ and $\int_{V_i} \cdot\, \mathrm{d}\,\boldsymbol{x}$ and $\int_{W_{ij}} \cdot\, \mathrm{d}\,\boldsymbol{x}$ are, respectively, the surface integral over the Voronoi region V_i and the line integral along the Voronoi edge W_{ij}. In order to numerically compute the integrals we must use numerical quadrature formulas. (We chose the 7-point formula of the Gaussian type for the numerical quadrature on a triangle based on which the integral on a polygonal Voronoi region V_i is approximated by dividing the polygon into triangles, and for a numerical

quadrature formula on a straight-line segment we chose the 4-point Gaussian formula.) The approximate values computed by those numerical quadrature formulas will be denoted with tilde in the following.

Now, the optimization algorithm follows.

0° Initialization: — Set, rather arbitrarily, the initial values $\{\boldsymbol{x}_i^{(0)} \mid i = 1, \ldots, N\}$ for $\boldsymbol{x}_i$'s and $\nu := 0$.

1° Construct the Voronoi diagram $\{V_i^{(\nu)}, W_{ij}^{(\nu)}, U_{ijk}^{(\nu)}\}$ for points $\boldsymbol{x}_i^{(\nu)}$'s.

2° Compute the approximate value $\tilde{F}$ of the function F of (2.7), the approximate values $\widetilde{\nabla F}$ of its gradient ∇F of (3.1) and those $\widetilde{\nabla\nabla F}$ of the Hessian (3.2) by means of the numerical quadrature formulas.

3° Modify $\boldsymbol{x}_i^{(\nu)}$ using the formula:

$$\boldsymbol{x}_i^{(\nu+1)} = \boldsymbol{x}_i^{(\nu)} + \alpha^{(\nu)} \boldsymbol{d}_i^{(\nu)}, \qquad i = 1, \ldots, N, \tag{3.3}$$

where $(\boldsymbol{d}_1^{(\nu)}, \ldots, \boldsymbol{d}_N^{(\nu)})$ is the Newton vector determined by $\widetilde{\nabla F}$ and $\widetilde{\nabla\nabla F}$, and $\alpha^{(\nu)}$ the parameter to be determined by line search. We may resort to the Marquardt method to stabilize the iterative process.

4° If the modification $\boldsymbol{x}_i^{(\nu+1)} - \boldsymbol{x}_i^{(\nu)}$ is small enough, or if $\widetilde{\nabla F}$ is small enough, or if the improvement $F(\boldsymbol{x}^{(\nu)}) - F(\boldsymbol{x}^{(\nu+1)})$ on the value of the objective function is small enough, then stop. Otherwise, increase ν by 1 and go to 1°.

This kind of brute-force iteration really works to some extent. Here, we shall show two examples in [7] and [17].

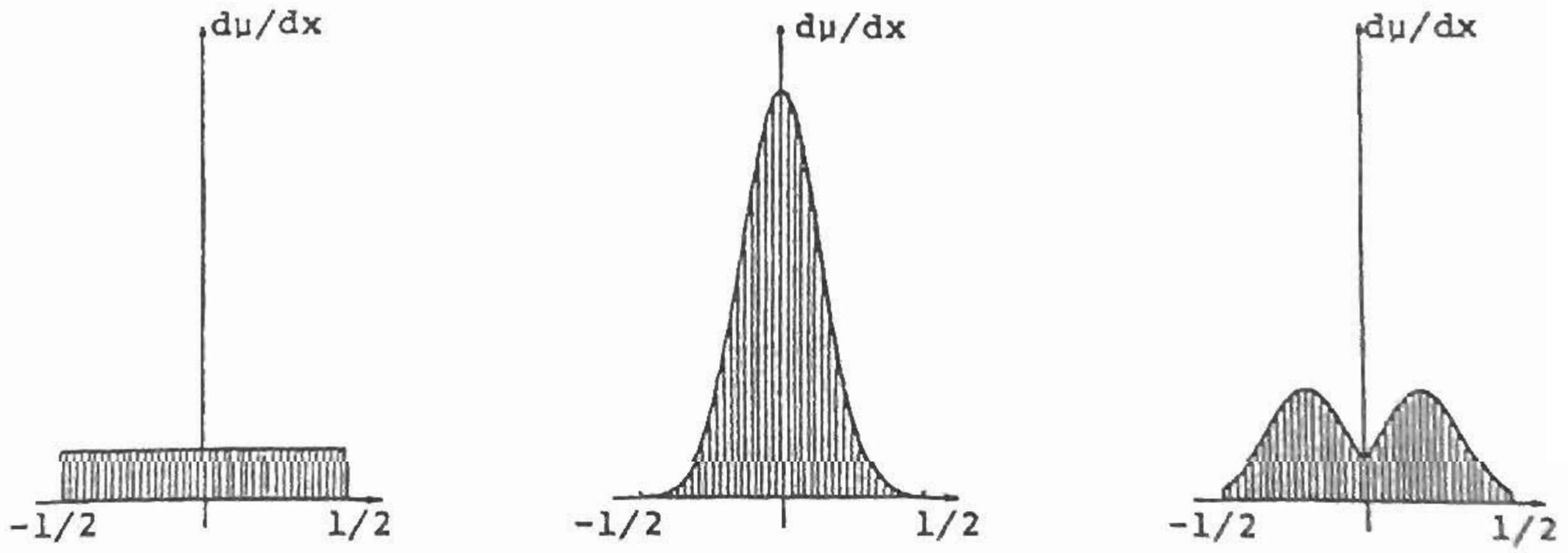

Fig. 1. Population distribution used in the experiments (one-dimensional section) [7]

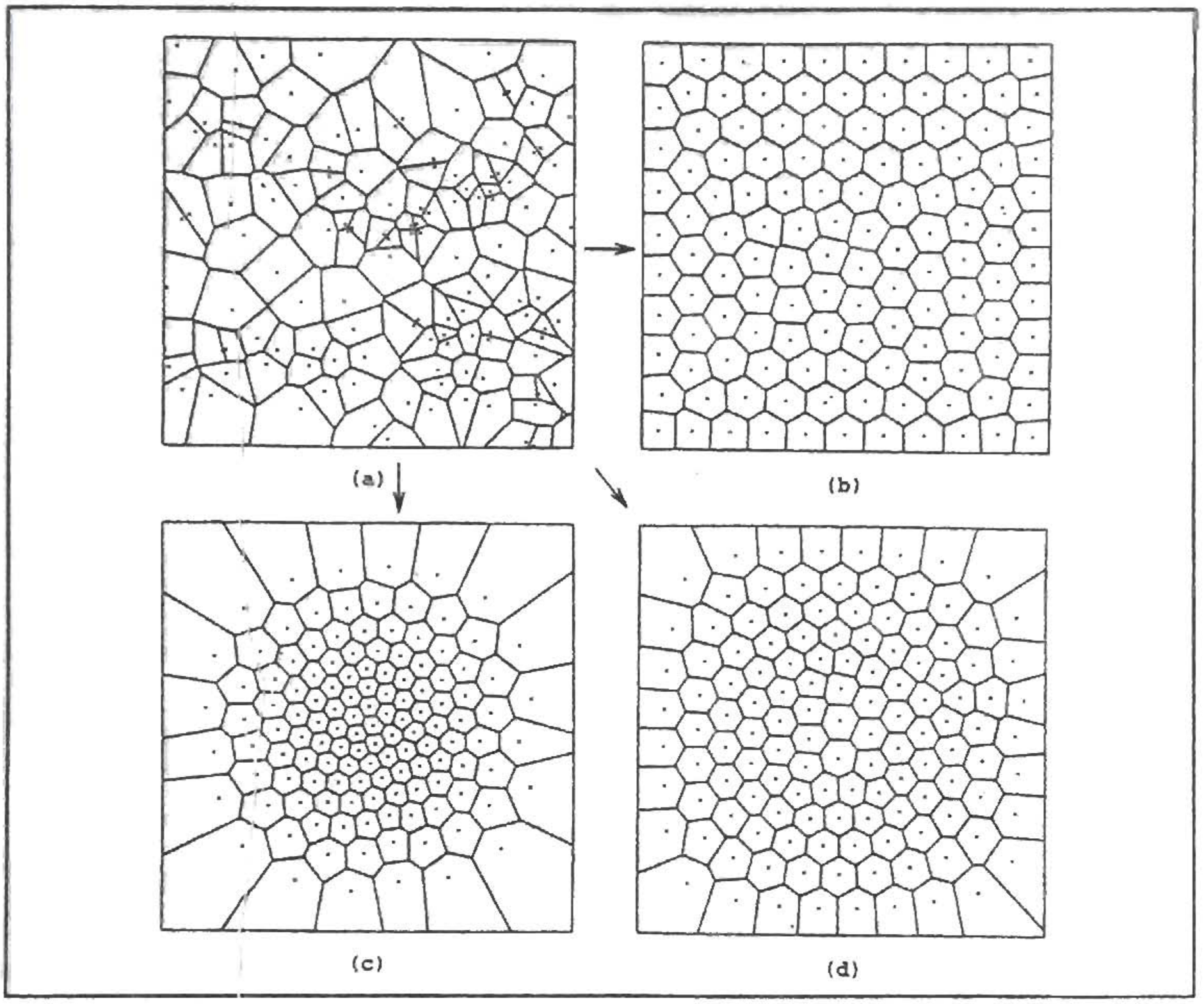

Fig. 2. (a) Initial distribution
(b) Near Optimum distribution for the population density of type 1°
(c) Near Optimum distribution for the population density of type 2°
(d) Near Optimum distribution for the population density of type 3°
([7])

Fig. 2 is the first example of simplest facility location problem in [7], with 128 facilities. Starting from the random initial distribution of facilities (Fig. 2(a)), we got to a nearly regular hexagonal distribution (Fig. 2(b)) by means of the brute-force iteration for the uniform distribution of inhabitants 1° in Fig. 1. We got to near optimum distributions Fig. 2(c) and (d), respectively, for the distributions of 2° and 3° in Fig. 1.

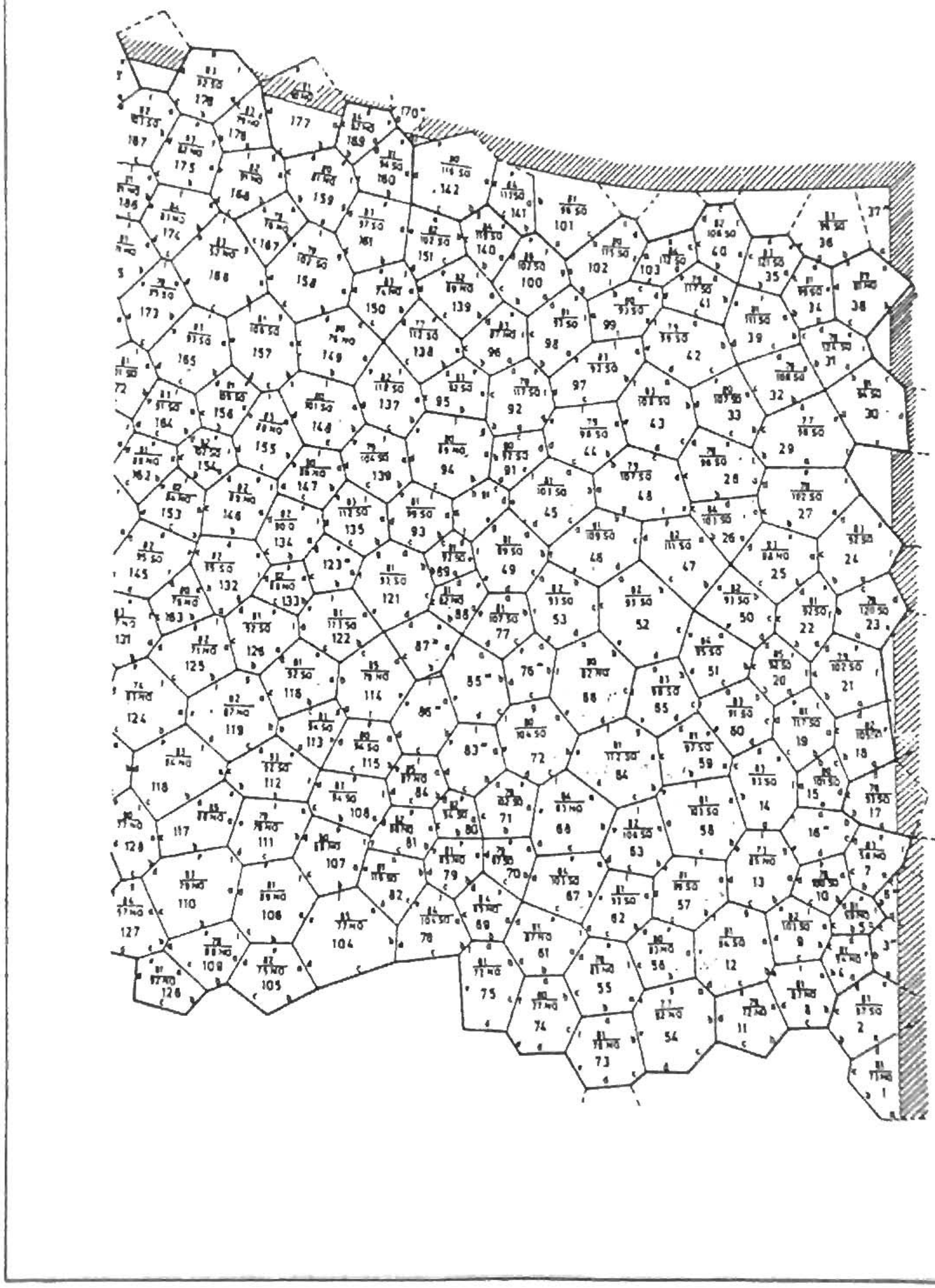

Fig. 3. Columnar structure of basalt [15]

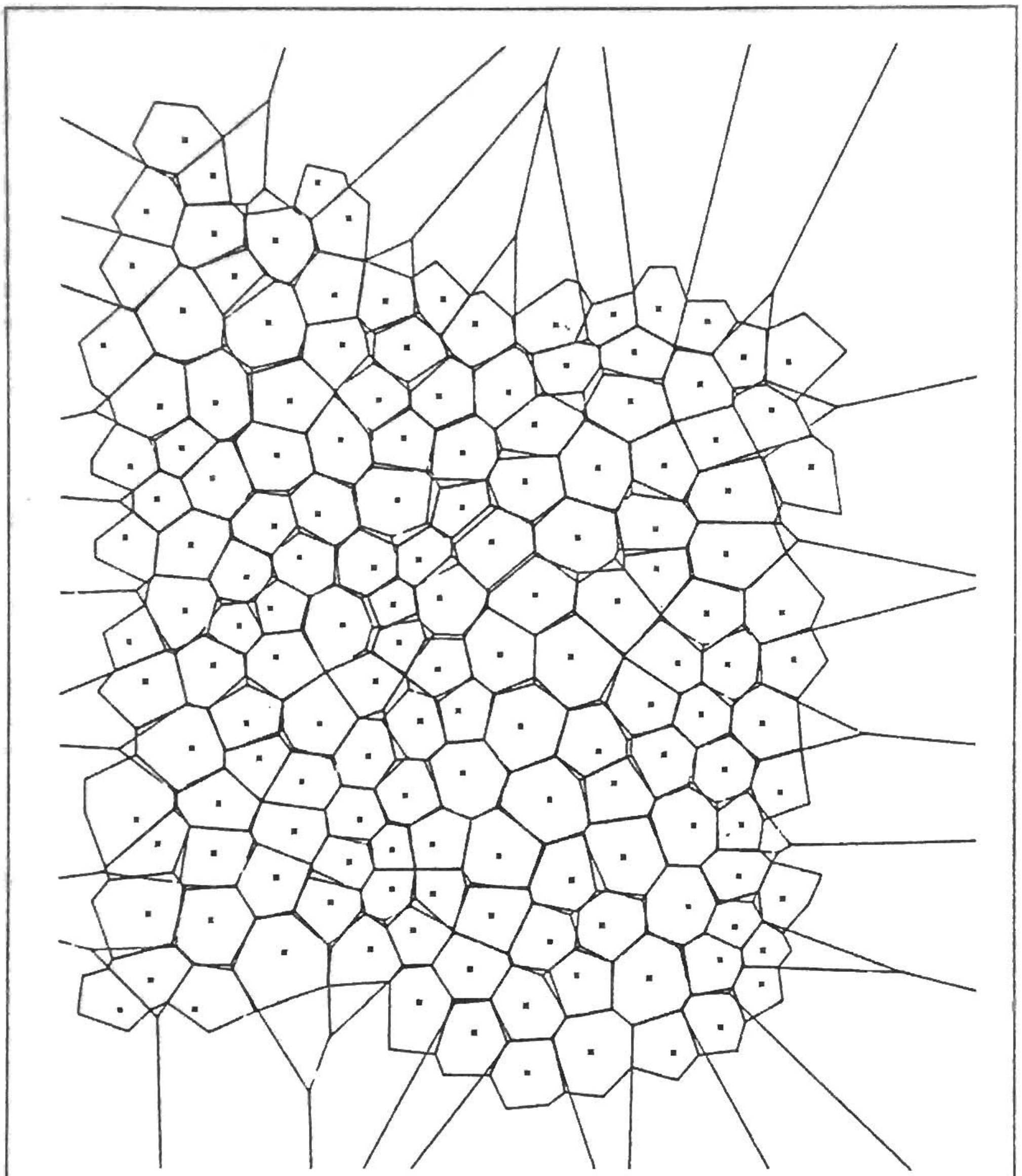

Fig. 4. Voronoi-diagram approximation of Fig. 3 [17]

The other example is shown in Fig. 3 and Fig. 4. Fig. 3 indicates an observed horizontal section of columnar structure of basalt [15]. There is a controversy among crystallographers on the underlying physical process of the growth of this kind of structure, particularly about whether this section diagram is a Voronoi diagram or not. An observed diagram cannot be exactly a Voronoi diagram due to various noises and observation errors so that our problem must be how well the observed diagram can be approximated by a Voronoi diagram. The problem is formulated in a manner similar to the problem of facility location but with a different from of objective function integral which represents the total area of discrepancy between the observed diagram and an approximate Voronoi diagram. Fig. 4 shows the result for Fig. 3. Thick lines indicate the observed grain boundaries, and thin lines the best approximation by a Voronoi diagram. Whether the approximation is good or bad will be evaluated by crystallographers.

4. COMPUTATIONAL DIFFICULTIES AND THEORETICAL INCONSISTENCY

Apparently, the iterative algorithm 0°–4° has a number of difficulties. Specifically:

(i) It is not an easy task to reconstruct a Voronoi diagram at each step of iteration ($\leftarrow 1°$).

(ii) The Newton direction $(\widetilde{\nabla\nabla F})^{-1} \cdot \widetilde{\nabla F}$, as well as the steepest-descent direction $\widetilde{\nabla F}$, coincides neither with the corresponding direction for the original analytical problem "min F" nor with that for the approximate problem "min F" ($\leftarrow 2°, 3°$).

(iii) When to stop the iteration is rather arbitrary ($\leftarrow 4°$).

(iv) Deriving the analytical expressions for ∇F, $\nabla\nabla F$ from that for F is not an easy task.

The first difficulty (i) has already been overcome practically. In fact, there is an incremental algorithm for constructing the Voronoi diagram which runs in $O(N)$ time in the average case for many nearly uniformly distributed points $\boldsymbol{x}_i$'s (0.2 ms per point on a standard non-parallel main-frame computer) [14], so that the substantial progress has been made with respect to computational time. On the other hand, according to experience, it is known that naïvely implemented algorithms, especially those of the divide-and-conquer type, are weak against rounding errors, i.e., they are often hung up for large problems due to rounding errors. However, this numerical difficulty has also been practically overcome by modifying the incremental method in [14] laying more stress on the topological condition that "the Voronoi diagram be a planar graph of vertex degree three" than on the values of the coordinates of a Voronoi point obtained by numerical computation [16]. This modification does not deteriorate very much the computational speed of the algorithm. In this way, graphical concepts and techniques did substantial contribution to the resolution of difficulty (i).

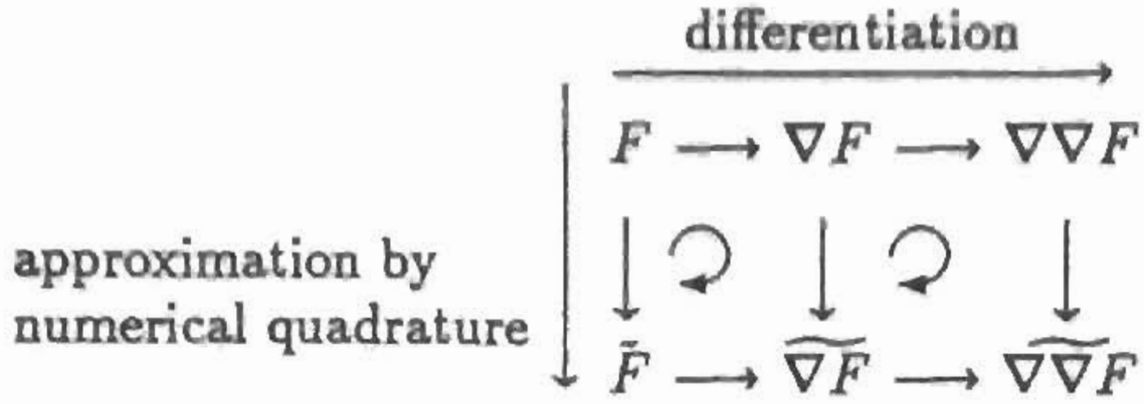

Fig. 5. Desirable scheme

The second difficulty (ii) comes from the fact that the scheme of Fig. 5 is not commutative. In fact, ordinary approximation will be represented by the scheme of Fig. 6. In this respect, there is another possibility. If we had a means for exactly calculating the gradient $\nabla\tilde{F}$ and the Hessian $\nabla\nabla\tilde{F}$ of the approximate objective

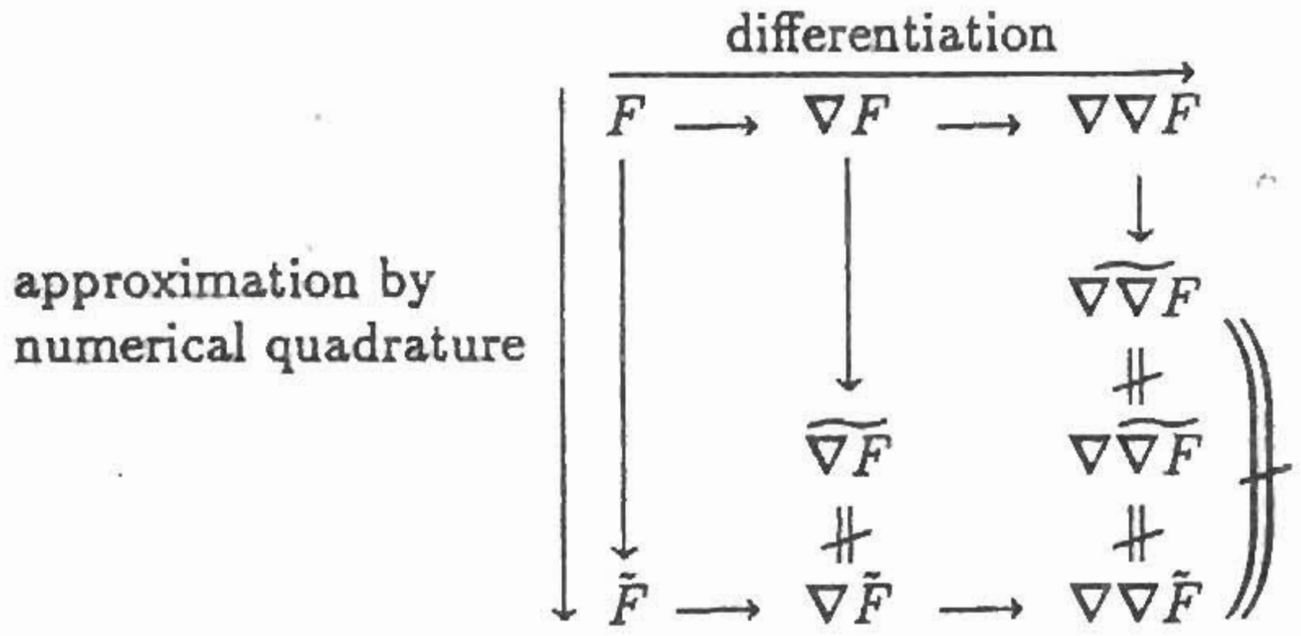

Fig. 6. Actual scheme

function $\tilde{F}$, we could go through all the process of optimization consistently within the approximate world. But, the program of computing $\tilde{F}$ itself is so long and complex containing construction of Voronoi diagrams and numerical quadratures. Furthermore, we are usually interested in problems with tens or hundreds of facilities, i.e., with twice as many variables, so that it would practically be impossible to write down the programs for computing all the exact partial derivatives of $\tilde{F}$. Even numerical differentiation would not be practically feasible because, besides its poor numerical accuracy, it would need to call the program for computing $\tilde{F}$ so many times at each iteration.

The third difficulty (iii) concerning the stopping criterion for the iteration is shared by many iterative methods in numerical analysis. Nobody could confidently judge the relevant quantities to be small enough. In fact, there is no objective reference level of "smallness" in most cases. Numerically, however, a noise level of rounding error to be incurred in the computation of the value of the approximate function $\tilde{F}$ or that of its gradient $\nabla\tilde{F}$, would be, at least conceptually, a good reference level, i.e., we will generally agree upon that the best we can do numerically

is to suppress the computed value of a relevant quantity (which is required to be made equal to zero) below the noise level of rounding error. Thus, we are led to the question how to get a good estimate of the upper bound of the rounding error incurred in the computed value of the quantity in consideration.

The fourth difficulty (iv) may remind us of the possibility of resorting to symbol manipulation software, but it actually is too tough to allow us to do so.

5. FAST AUTOMATIC DIFFERENTIATION

The difficulty (ii) of noncommutativity of differentiation and approximation and that (iii) of rounding error estimation in the geometrical/geographical optimization (which are common in many other optimization-related problems) are practically resolved by the help of the recently developing method of Fast Automatic Differentiation [1], [3], [5], [6], [8], [9], [10], [11], [12]. Using that method, we can automatically convert the original program for computing a function of many variables, however long and complex it may be, into another program for computing simultaneously the function, all the components of its gradient vector and a sharp upper bound on the rounding error in the computed value of the function, in time at most proportional to the time for computing the function alone by the original program. Here, the constant of proportionality is independent of the number of variables and of the complexity of the original program. It is 4, 5 or 6 in terms of operation counts, but, with rather primitive implementation, several tens in terms of computational time due to various overheads. Furthermore, we can compute each row of the Hessian matrix and a sharp upper bound on the rounding error in the computed value of each component of the gradient vector in a similar manner. The difficulty (iv) is, thus, got rid of automatically.

Here again, graphical representation of the computational process of the function plays an essential rôle for the design of the algorithm of the program conversion and for the analysis of its complexity. In brief, we may construct the "computational graph" representing the history of computation, or "computational process", of the function value while we compute it. A node of the computational graph represents an input variable, a constant used in the computation, the function or an intermediate variable, and arcs represent which intermediate variable has been computed from which intermediate variables (or input variables or constants) by a basic operation (corresponding to their terminal node), and to each arc is associated the value of the elementary partial derivative of the corresponding basic operation with respect to the corresponding argument.

A computational graph is acyclic by definition. By traversing the computational graph from the function node down to the input nodes only once, we can find all the first-order partial derivatives of the function with respect to all the intermediate variables v_j and to all the input variables x_i. This way of traversing is similar to that in which we compute the shortest distance on an acyclic graph. $\partial f/\partial x_i$'s are the components of the gradient ∇f. $\partial f/\partial v_j$'s are the amplification factor of the rounding error δv_j generated at the computation of the intermediate

variable v_j to the function value, so that we have an upper bound $|\Delta f|_A$ on the rounding error Δf in the computed value of the function f as follows.

$$|\Delta f| = \left| \sum_j \frac{\partial f}{\partial v_j} \delta v_j \right| \leq \sum_j \left| \frac{\partial f}{\partial v_j} \right| \cdot |\delta v_j| \leq \left(\sum_j \left| \frac{\partial f}{\partial v_j} \right| \cdot |v_j| \right) \cdot \varepsilon \equiv |\Delta f|_A, \quad (5.1)$$

where we assume

$$|\delta v_j| \leq |v_j| \cdot \varepsilon \quad (5.2)$$

with the relative precision ε of the floating-point representation of real numbers.

Since the way of computing the gradient so far explained is itself represented in the form of a program, we can apply the same principle to it to get the Hessian as well as the rounding error estimates for the gradient.

6. NUMERICAL EXAMPLE

This is a small numerical example of a typical location problem. We used Sparc Station 1+(Sun 4-65) with 12Mbyte memory on SUN-OS 4.0.3c and "f77" FORTRAN compiler with option "-O". (All operations are performed with "double precision".)

The problem and the computational method are as follows.

(1) $f(t^2) = t^2$.

(2) $\varphi(\boldsymbol{x}) = \exp(-9\|\boldsymbol{x}\|^2)$.

(3) The number of facilities $N = 30$.

(4) Initial location $\boldsymbol{X}^{(0)}$ is a sample from the uniform distribution over the unit square (Fig. 7).

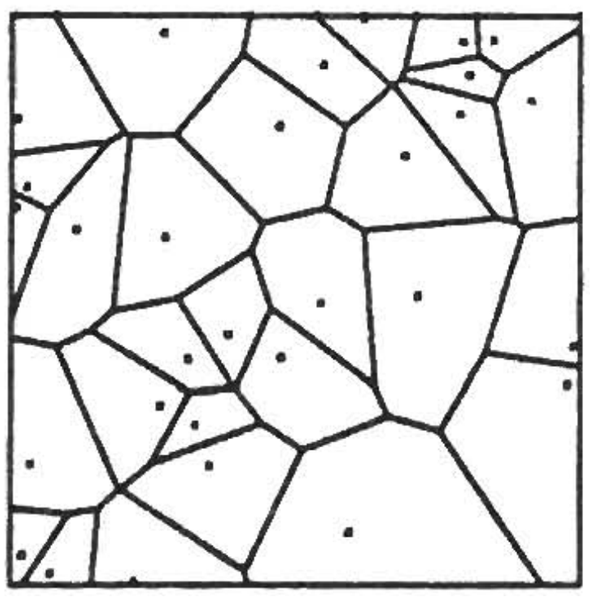

Fig. 7. Initial location

(5) The Newton-Raphson method with Marquardt's modification was used.

In Fig. 8 are shown the sequences of numerical results:

(i) $\tilde{F}$, $\|\nabla\tilde{F}\|_2$ and $\|\nabla\tilde{F}\|_N \equiv \max_i \dfrac{|(\nabla\tilde{F})_i|}{|\Delta(\nabla\tilde{F})_i|_A}$ computed by the "consistent algorithm" using $\tilde{F}$, $\nabla\tilde{F}$ and $\nabla\nabla\tilde{F}$ ($\nabla\tilde{F}$ and $\nabla\nabla\tilde{F}$ as well as $|\Delta(\nabla\tilde{F})_i|_A$ computed by means of fast automatic differentiation);

(ii) $\tilde{F}$ and $\|\widetilde{\nabla F}\|_2$ computed by the "inconsistent algorithm" using $\tilde{F}$, $\widetilde{\nabla F}$ and $\widetilde{\nabla\nabla F}$ as described in Section 3.

Note that the two sequences $\{\boldsymbol{X}_C^{(0)}, \boldsymbol{X}_C^{(1)}, \boldsymbol{X}_C^{(2)}, \dots\}$ generated by the consistent algorithm and $\{\boldsymbol{X}_I^{(0)}, \boldsymbol{X}_I^{(1)}, \boldsymbol{X}_I^{(2)}, \dots\}$ generated by the inconsistent algorithm which are both seemingly convergent, are considerably different from each other although they start from one and the same initial vector $\boldsymbol{X}_C^{(0)} = \boldsymbol{X}_I^{(0)}$.

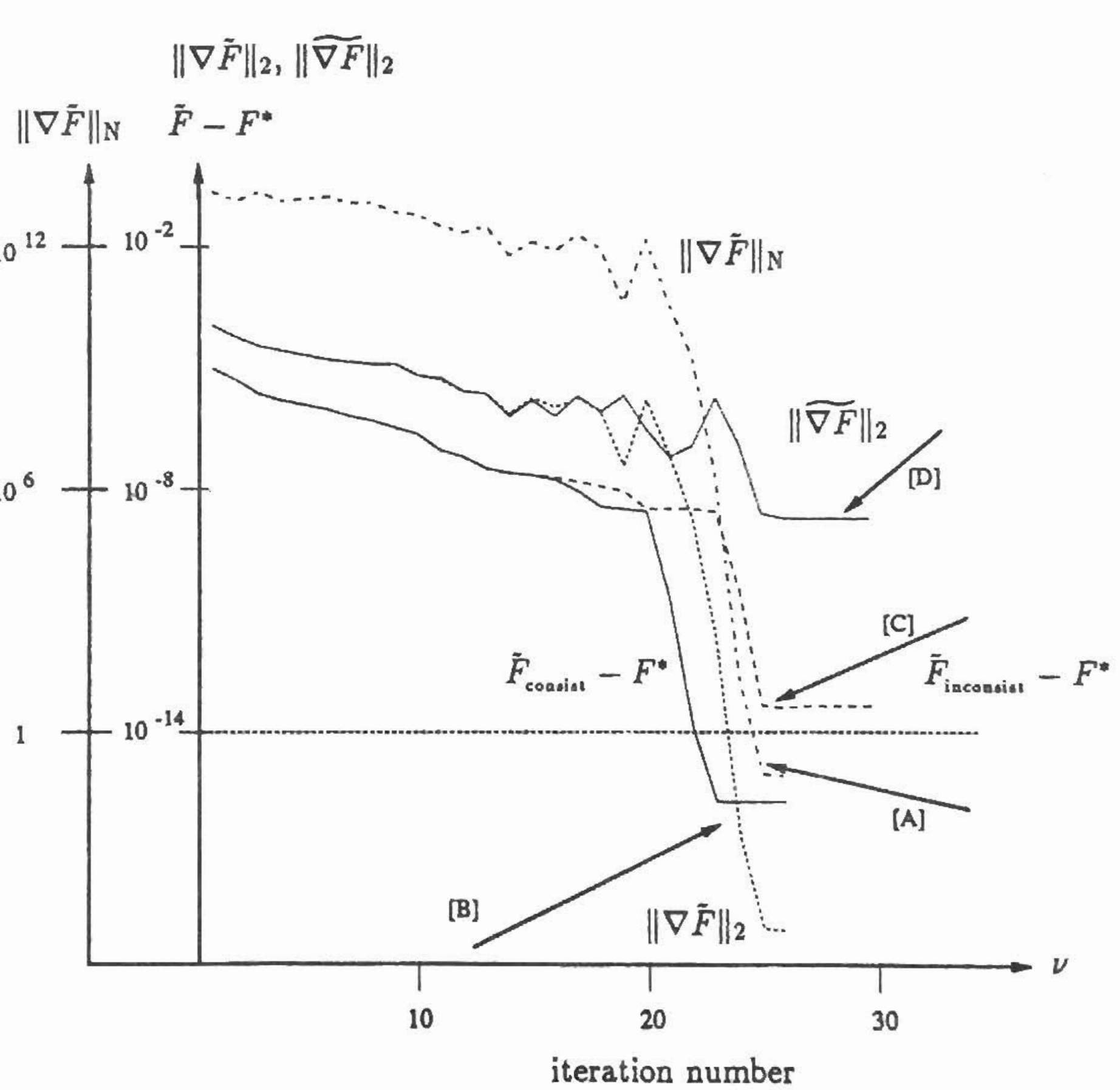

Fig. 8. Numerical results $(F^* = 5.980165886 \times 10^{-6})$

The following may be remarkable points to observe in Fig. 8.

With the consistent system $(\tilde{F}, \nabla\tilde{F}, \nabla\nabla\tilde{F})$ we can stop the iteration at [A] in Fig. 8 because $\|\nabla\tilde{F}\|_N \leq 1$ holds there, i.e., $\nabla\tilde{F}$ becomes small enough in comparison with the noise of rounding errors. Moreover, at [B] in Fig. 8, we can see also that $\tilde{F}$ and $\|\nabla\tilde{F}\|_2$ are minimized simultaneously. Thus, we can be confident that we have got a local minimum (Fig. 9).

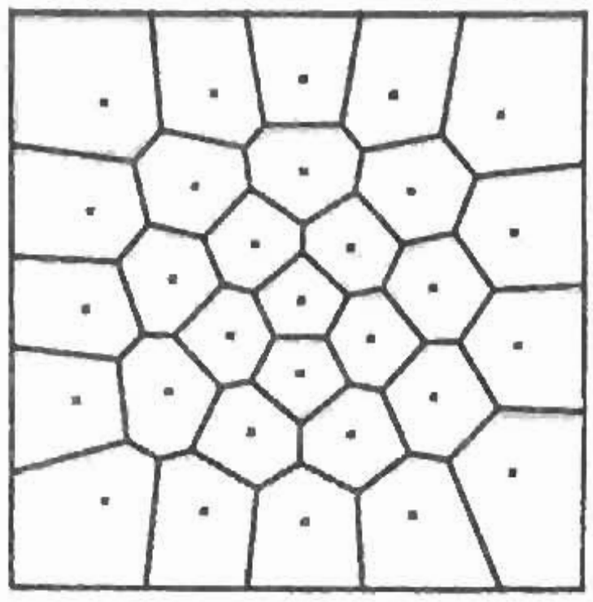

Fig. 9. Near optimum location obtained by the consistent algorithm

On the other hand, with the inconsistent system, at [C] in Fig. 8, we had to stop the iteration because we could not find descent direction although the value of the function could be decreased. (The resulting location is seemingly almost the same as that by the consistent system.) Indeed, the estimate of the rounding error of $\tilde{F}$ at [C] was almost equal to 10^{-19} so that the difference of the value of $\tilde{F}$ at [B] and that at [C] was large enough to separate those values numerically. At [D] in Fig. 8, although $\|\widetilde{\nabla F}\|_2$ was not small enough, the vector $(\widetilde{\nabla\nabla F} + \gamma I)^{-1} \cdot \widetilde{\nabla F}$ (with $\gamma \approx 10^{-2}$) and $\widetilde{\nabla F}$ were almost orthogonal to the steepest descent direction $\nabla\tilde{F}$. Therefore, $\tilde{F}$ failed to decrease any more. Alternatively, we might insist to decrease $\|\widetilde{\nabla F}\|_2$ further, but, even if we did it, $\tilde{F}$ would not be made smaller any longer, which would be meaningless from the point of view of optimization.

Although, of course, the solution computed by the consistent system gives only one of the local minima, we think that it is meaningful to establish a method without theoretical inconsistency. Indeed, we can stop the iteration at [A] and [B] (the 24th iteration) by the consistent algorithm on a sound theoretical basis using the rounding errors incurred in the computed values of $\nabla\tilde{F}$, but we could not stop the iteration at [C] by the inconsistent algorithm.

7. CONCLUSION

We would like to emphasize here again that the two techniques, the topology-oriented algorithm for constructing Voronoi diagrams and Fast Automatic Differentiation, are essential for overcoming the computational difficulties and resolving the inconsistency which appears in the optimization algorithms.

REFERENCES

[1] A. Griewank, *On automatic differentiation*, in: M. Iri and K. Tanabe (eds.), *Mathematical Programming — Recent Developments and Applications*, Kluwer Academic Publishers, 1989, 83–107.

[2] M. Iri, *Applications of matroid theory*, in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming — The State of the Art*, Springer-Verlag, 1983, 158–201.

[3] M. Iri, *Simultaneous computation of functions, partial derivatives and estimates of rounding errors — complexity and practicality*, Japan Journal of Applied Mathematics 1 (2) (1984), 223–252.

[4] M. Iri, *Practical computational methods in geometrical/geographical optimization problems*, in: M. J. Beckmann, K.-W. Gaede, K. Ritter and H. Schneeweiss (eds.), *Methods of Operations Research 54*, Verlag Anton Hain, 1986, 17–37.

[5] M. Iri, *History of automatic differentiation and rounding error estimation*, in: A. Griewank and G. F. Corliss (eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, 1991.

[6] M. Iri and K. Kubota, *Method of fast automatic differentiation and applications*, Research Memorandum RMI 87-02, Department of Mathematical Engineering and Instrumentation Physics, Faculty of Engineering, University of Tokyo, 1987.

[7] M. Iri, K. Murota and T. Ohya, *A fast Voronoi diagram algorithm with applications to geographical optimization problems*, in: P. Thoft-Christensen (ed.): *Proceedings of the 11th IFIP Conference on System Modelling and Optimization*, Copenhagen, 1983, Lecture Notes in Control and Information Science **59**, "System Modelling and Optimization", Springer-Verlag, 1984, 273–288.

[8] M. Iri, T. Tsuchiya and M. Hoshi, *Automatic computation of partial derivatives and rounding error estimates with applications to large-scale systems of nonlinear equations*, Journal of Computational and Applied Mathematics **24** (1988), 365–392.

[9] K. V. Kim, Yu. E. Nesterov, V. A. Skokov and B. V. Cherkasskiĭ, *Effektivnyĭ algoritm vychisleniya proizvodnykh i èkstremal'nye zadachi*, Èkonomika i Matematicheskie Metody **20** (1984), 309–318.

[10] K. Kubota, *PADRE2 — A FORTRAN precompiler yielding error estimates and second derivatives*, in: A. Griewank and G. F. Corliss (eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, 1991.

[11] K. Kubota and M. Iri, *Estimates of rounding errors with fast automatic differentiation and interval analysis*, Research Memorandum RMI 88-12, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1988. (Japanese translation appeared in Transactions of Information Processing Society of Japan **30** (1989), 807–815.)

[12] K. Kubota and M. Iri, *PADRE2, version 1 — User's manual*, Research Memorandum RMI 90-01, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1990.

[13] K. Murota, *Systems Analysis by Graphs and Matroids — Structural Solvability and Controllability*, Algorithms and Combinatorics **3**, Springer-Verlag, 1987.

[14] T. Ohya, M. Iri and K. Murota, *Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms*, Journal of the Operations Research Society of Japan **27** (1984), 306–337.

[15] R. A. Koch, L. Pfeiffer and L. Stammler, *Der Basalt von Stolpen*, Deutscher Verlag für Grundstoffindustrie, Leipzig, 1983.

[16] K. Sugihara and M. Iri, *VORONOI2 reference manual — Topology-oriented version of the incremental method for constructing Voronoi diagrams*, Research Memorandum RMI 89-04, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1989.

[17] A. Suzuki, *Study on the Facility Location Problem in the Plane*, Doctoral Dissertation, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1988 (in Japanese).