Applying one approach to geometric modelling

Vasyl Tereshchenko
Faculty of Cybernetics
National Taras Shevchenko University , Kiev, Ukraine
v_ter@ukr.net

Abstract

The current state of computer technology allows formulate of new problems and develop effective solutions for them. This requires developing of complex computer models and ways of their formation.Thus, when designing and building a visual model of thermal processes at welding turbine blades of aircraft engines, must be extremely accurate model the welding bath, the welding seam and region adjacent to them. Getting the exact solution of this problem requires simultaneous solution the whole complex of heat physical and geometrical problems which demand significant computing resources. This paper's main result is presenting a generalized effective parallel-and-recursive algorithm with the optimal bound of time complexity $O(log^2N)$ which solves in unified manner the variety of interrelated geometrical problems for the construction of visual models of complex phenomena and processes. This algorithm belongs to the class of the solvability NC_2.

Keywords: *parallel-and-recursive algorithm, visual model,* interrelated geometrical problems.

1. INTRODUCTION

The paper's main result is presenting the generalized effective parallel algorithm which solves in unified manner the variety of interrelated geometrical problems for the construction of visual models of complex phenomena and processes. Such an algorithm is based on well-known technique "divide-and-conquer". An algorithm consists of two basic stages: recursive dividing, which is general for the whole set of problems, and recursive merging. Steps of the merge stage could be executed simultaneously for each problem. The parallel computing model is asynchronous, so merge stage is executed independently for each problem. An application of this algorithm can be, for example, the modeling of thermo-physical and thermo-mechanical processes which arise up at welding. In this case it is necessary to build the exact visual model of the welding bath, the welding seam and region adjacent to them. To have an exact model of welding bath at every moment of time there is the need to simultaneously solve the following geometrical problems: constructing convex hull, triangulation, Voronoi diagram, searching for all nearest neighbors, nearest pairs, geometrical search, spline approximation and so on. Figure 1 shows a fragment modelling welding bath.

In this paper a paradigm is offered - an effective algorithm of parallel solution the set of interrelated geometrical problems for multitask modeling of complex dynamic processes. This paradigm is illustrated on the examples of Voronoi diagram and "all nearest neighbors" problem. It is important to note that main idea of this paper is not to describe the well-known paradigm "divide-and-conquer". In works [1-4] there have been described efficient parallel algorithms for solving some problems of computational geometry, including the mentioned above technique.

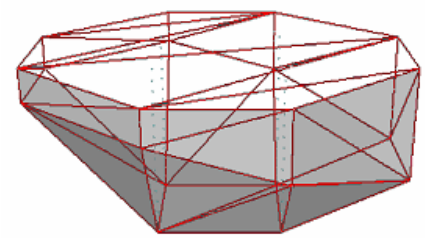

Figure 1: Fragment modeling welding baths: convex hull and triangulation.

For accurate modelling it is necessary to solve simultaneously a set of interrelated problems. The best strategy in this case would be one that uses a common tool for implementing the tasks: data structures, steps of procedures and presentation of results. The most suitable in this case, in our opinion, may be technique "divide-and-conquer". In this technique, the stage of dividing is a common and unified for all tasks, and at the merge stage is proposed to use a common and unified data structure (weighted concatenable queue) at which the procedures are performed quickly. Moreover, the results of the individual steps of some procedures used by other procedures, that ensures high efficiency.

Problem. Let S be the given set of N points in space E^d. It is necessary to develop the generalized effective parallel-and-recursive algorithm for simultaneous solving problems of computational geometry, which are defined on single set of data S, for which the low bound of complexity is $\Omega(NlogN)$ (for one processor machines).

2. THE GENERALIZED ALGORITHM

One of main application problems of the "divide-and-conquer" technique for solving problems of computational geometry is nonlinearity of the merge stage and linear inseparability of the set of points. In the considered approach for problems, due to the composition of successful data structures at the stage of preliminary processing and the use of parallel processing at stages of dividing and merge it is possible to construct the effective parallel-and-recursive algorithm which removes the restrictions specified above. We will consider the technique of the algorithm for the two dimensions case.

2.1 Mathematical model of the algorithm

The mathematical model of the offered parallel algorithm consists of such basic stages: *preliminary processing, dividing set*

of points (recursive descent)and recursive merge of results for subsets (recursive ascent)

***Stage 1*. Preliminary processing.** Let S be the given set of N points in the plane $S = \{P_1, P_2, \ldots, P_N\}$ and there are $O(N)$ processors. The ordered array of points $U = \{P_{ij};\ i, j = 1,\ldots, N\}$ is formed at the stage of preliminary processing. Here, i is an index, which specifies the number of points in the list sorted by the x coordinate, and j is an index, which specifies the number of points in the list sorted by the y coordinate. Constructing the sorted lists for $O(N)$ processors can be carried out by means of one of the algorithms described in details in articles [5-7]. An array formed by such a method is given to the input of the algorithm. The graph of this algorithm it is binary tree, Figure2. In this graph, every node is marked by an integer number k. Number k divides the list of points in nodes into two lists of equal power, on the median, after comparison of the first indexes of points in the array U. And every number of node k is determined by the single iteration on a tree, if the quantity of points of the set is known.

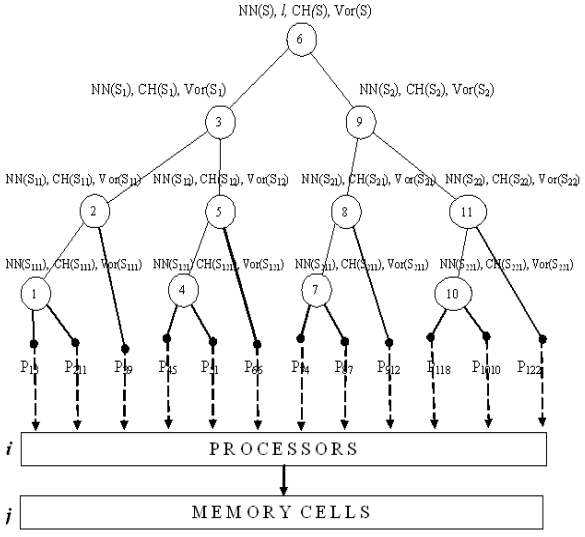

Figure 2: Graph of algorithm. *NN*(*S*), *Vor*(*S*), *CH*(*S*) – merge procedures; l - median.

***Stage 2*. Dividing the set of points (recursive descent).** This stage of an algorithm consists of dividing the set of points in the form of list U into equipollent subsets U_1, U_2, searching for a median l and transferring U_1, U_2 at the next step of recursion. The search median in sorted by x indexed array U is performed in constant $O(1)$ time. Time necessary for recursive descent in the parallel algorithm is defined by a following lemma.

Lemma 1. Using $O(N)$ processors it is possible to execute in time $O(logN)$: a stage of recursive separating the set S from N points on equal capacity of subsets S_1 and S_2 in the plane, search of a median l and transfer of subsets S_1 and S_2.

Proof. Let the given set of points to be presented in the form of an indexed two-dimensional ordered array $U = \{Pij\ ;\ i, j = 1,\ldots, N\}$. For constructing of such a structure of data it is possible to take advantage of parallel algorithm of sorting with complexity $O(logN)$, offered Colle [7]. Such a representation of points set allows constructing a tree of dividing, if the quantity of points N in the list U is known. According to the algorithm, first index i of each point P_{ij} is associated with processor number, and second index j is associated with the number of a memory cell in which a point is stored according to orderliness on y coordinate. On each step of dividing corresponding processors synchronously compare the first indexes from the list of points and dispatch points in corresponding nodes of algorithm, keeping thus the order of an arrangement of points which is defined by their order in memory cells. Considering precise orderliness of points P_{ij} in array U by both indexes and interrelation between processors and the memory elements, time of performance of merge process in each node of a tree will not exceed constant $O(1)$. Thus, general time of dividing will not exceed $O(logN)$ for the worst data input. As it was necessary to prove.

***Stage 3*. Recursive merge of results for subsets (recursive ascent)**. At this stage, the merge procedures of related problems are running in each node of the algorithm graph. These procedures are building a general solution of problems. Process comes to the end with result of merge in root node. In the presented paper, due to the limits on the pages number, and that the step of division is common to all the problems we will consider example of merger procedures construction for ”Voronoi diagram” and ”all nearest neighbors”. The main feature of the proposed procedures - is to use a common data structure ”weighted concatenable queue”. This data structure allows to perform all actions within the logarithmic time.

2.2 Constructing merge procedures

The merge stage of algorithm for constructing Voronoi diagram differs from the merge stage in the convex hull only on a finishing step. In the first case dividing chain is built, which connects the Voronoi diagrams of sons, and in the second case - it is finding of bridges (common tangent segments) to the convex hulls of sons. Thus the results of convex hulls constructing and bridges got for a problem ”convex hull”, is used for next steps of building Voronoi diagram. In addition, construction of dividing chain for the Voronoi diagram in parallel allows to solve the problem of finding all nearest neighbors.

2.2.1. Constructing merge procedure for ”Voronoi diagram” problem. At every step of recursive ascent starting from the second, Voronoi diagrams **vor**(S_L) and **vor**(S_R) for subsets of points from the left and right sons, accordingly, are fed to the input of parent node v of the tree. It is necessary to build Voronoi diagram for the node v. Since the basic step of the constructing merger procedures for the Voronoi diagram and all nearest neighbor, in this algorithm, is building a monotone dividing chain, we offer one of the possible algorithms for its construction.

Constructing the dividing chain. The dividing chain constructing process is executed using $O(N)$ processors for the subsets of points, which are contained in a zone near dividing vertical line l (Figure 3). These subsets are located to the left and to the right from l, and belong to the mutually convex chains of convex hulls of sons and points determined by the edges of diagrams *vor*(S_L), *vor*(S_R), which cross the edges of these chains.

The time required for the construction of the dividing chain is determined by the following lemma.

Lemma 2. Constructing the dividing chain $\sigma(S_1, S_2)$, which "sews" together the Voronoi diagrams **vor**(S_L), **vor**(S_R) at every step of the merge stage can be completed in $O(logN)$ time using $O(N)$ processors.

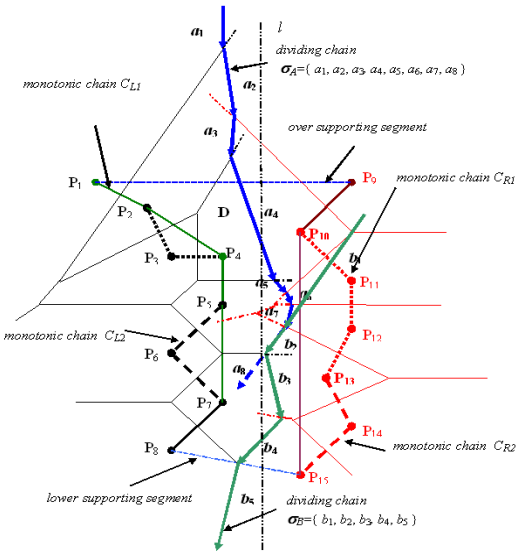

Figure 3. The merge step of two dividing chains (upper σ_A = { a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 } and lower σ_B={ b_1, b_2, b_3, b_4, b_5 }chains) for the pairs of monotonous chains (C_{L1}, C_{R1}), (C_{L2}, C_{R2}), accordingly.

Proof. Between the upper and lower supporting edges of two convex hulls of Voronoi diagrams for sons **vor**(S_L), **vor**(S_R) of some node v of algorithm graph we have two mutually convex chains. They are determine the region of constructing the dividing chain, D. Each of these chains determines the ordered set of the Voronoi diagram edges, which are directed in into the region D, and they cross the edges of mutually convex chains CH_L, CH_R. Each edge of chains CH_L, CH_R can cross with one or a few edges of the Voronoi diagram and consequently, determines the set of points parted by these edges. We will note the set of vertexes as $B_L(S_1)$ ($B_R(S_2)$). It is consists of vertexes the convex chain $CH_L(CH_R)$ and the points, which are determined by edges Voronoi diagram that are crossing the chain. We will name this set as *left-maximum (right-maximum)* ordered set of points (or a list). If we connect the points in the lists $B_L(S_1)$ and $R(S2)$ consistently, we get lists of edges $E_L(S_1)$ and $E_R(S_2)$, which form chains S_L and S_R, respectively.

Lemma 3. Chains S_L and S_R are monotone in relation to direct l.

Proof. We will prove from opposite. It is known, that dividing chain $\sigma(S_1, S_2)$ is necessarily monotone in relation to direct l. Lets at least one of chains S_L and S_R will be not monotone in relation to l. Then there are edge this chain which will have the angle of rotation in relation to an OX axis with beginning at the end of this edge greater than π. It follows that the corresponding edge Voronoi diagram will not get into the domain D, and the dividing chain will not be monotone, which contradicts the condition.

It is important to mark that merge procedure in every node of algorithm tree can be executed on several processors independently and parallel. In order to execute such actions it is necessary: to determine a data structure which would support a convex hull and Voronoi diagram in every node, would allow to find supporting points, uncouple and couple parts of convex hulls, to conduct supporting segments and to build a dividing chain. As a data structure which would execute the operations mentioned above for logarithmic time, we have chosen the concatenable queue, the same data structure as in problem "Convex hull", with defined procedure MERGE (U_L, U_R), which allows to find supporting points and supporting segments, to build convex hull in $O(logN)$ time and with procedure which would allow to build a dividing chain. For organizing the process constructing the dividing chain, on the basis of lists $B_L(S_1)$ and $B_R(S_2)$ we will create the proper data structures (Figure 4), loading them with needed data.

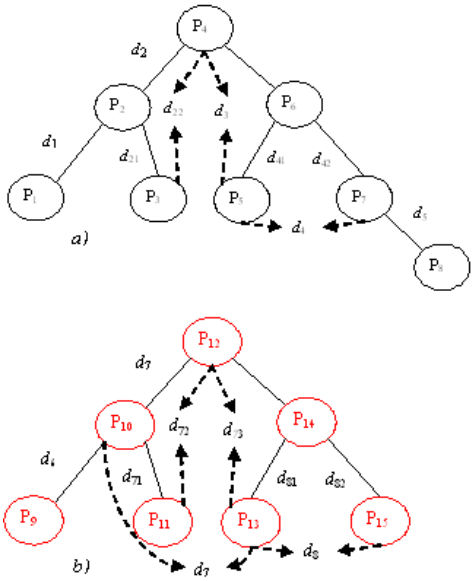

Figure 4. Data structures: the concatenable queues for left C_L and right the C_R chains of merge region D of the Voronoi diagrams: Vor(S_1), Vor(S_2) of figure 3, accordingly.

The concatenable queues of both monotone chains are binary trees with a root, in which nodes we have the coordinates of vertexes, and the arcs of which are the proper edges ek (k = 1,…, N) from the lists of chains $E_L(S_1)$ and $E_R(S_2)$. In addition, nodes are loaded by pointers on the proper edges Voronoi diagram (we will note them through d_{ij} ; i, j $\in$ N). Such data structures allow to build a dividing chain $\sigma(S_1, S_2)$ using $O(N)$ processors in $O(logN)$ time. Graph algorithm for constructing a dividing chain can be represented as binary tree. In the leaves tree, each with $O(N)$ processors builds dividing chain for the corresponding pairs of edges (e_k, e_l) ($e_k \in E_L(S_1)$; $e_l \in E_R(S_2)$). The results are given to the next level of the tree where the step of merge is carried out. As a result, a dividing chain is built as connection the dividing chains of sons. The merge process of dividing chains sons requires in $O(1)$ time in every node tree. As we see all transactions for the

constructing of dividing chains require no more than $O(logN)$ time using $O(N)$ processors.

After determining the supporting points and uncoupling convex hulls of node v sons by them, the left and right parts of trees that support the convex hulls U_L and U_R between the supporting points are deleted respectively. The balanced trees, which will support the upper and the lower convex hull of node v are formed by merge parts of trees which remained. All operations are performed in $O(logN)$ time. The obtained trees support a convex hull and allow to execute the constructing procedures of dividing chain in $O(logN)$ time.

2.2.2. Constructing of merge procedure for "all nearest neighbors" problem. At each stage of recursive ascent, starting from the second, at the input of parent node v of the algorithm graph there are Voronoi diagrams (VD) from the left and right sons $vor(S_L)$, $vor(S_R)$, and the nearest neighbor for each point from subsets S_L and S_R. Voronoi diagram is built for the node v, and the new neighbors at the border of subsets of S_L and S_R are being determined simultaneously, relatively to median l. At the merging stage the algorithm, during the constructing the dividing chain of Voronoi diagram, we find simultaneously nearest neighbors among the points from sets S_L and S_R, which form the current pair of a chain edge $\sigma(S_L, S_R)$, Figure 5.

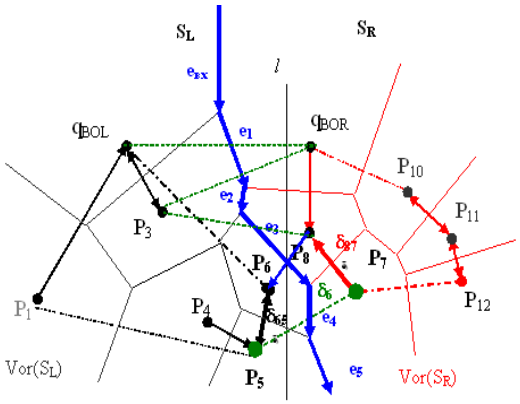

Figure 5. Search nearest-neighbors to pairs of points (P_3, q_{BOR}), (P_3,P_8), (P_6,P_8), (P_6,P_7), (P_5,P_7). For $P_8 \in S_R$ is found a new nearest neighbor $P_6 \in S_L$.

Thus, the edge of the dividing chain determines the next pair of points, which is being checked for the presence of the new nearest neighbors. Let $NN(S)$ be set of pairs the nearest neighbors for set S and $NN(S_L)$, $NN(S_R)$ - for sets S_L and S_R, respectively. The algorithm graph in this case is a binary tree, with the only difference that every node of a tree loaded except for the orderly array of points, median l; Voronoi diagrams $vor(S_L)$ and $vor(S_R)$ children, sets of the pairs points sons of nearest neighbors $NN(S_L)$ and $NN(S_R)$, respectively.

Lemma 4. The stage recursive merging of search results the nearest neighbor to each point of the set S of N points on the plane, using $O(N)$ processors, can be executed in $O(log^2N)$ time.

So as for constructing dividing chain in step merger it is enough $O(logN)$ time, then total complexity of step merger will be the same. This is because at each step constructing edge of dividing chain we must find the nearest neighbors for pair of points which he separates. For it we should compare only two distances.

3. IMPLEMENTATION OF THE ALGORITHM

For implementation of the algorithm is applied MPI and PAROS (Parallel Asynchronous Recursively Operated Systems) technologies (for PRAM model). Those technologies allow to simply and effectively implementing parallel-and-recursive algorithms for the solution of complex problems both on multiprocessing machines, and in a computer network.

4. CONCLUSION

In the given article is proposed approach which allows to develop effective and convenient means of automation the geometric modelling of complex phenomena and processes. The main feature of the implementation approach is that the parallel algorithm simultaneously executes both different steps of one procedure on many processors, and different procedures in one node. It allows develop the generalized algorithm of solution for number geometric problems by single technology.

5. REFERENCES

[1] A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlaing, and C.C. Yap. *Parallel computational geometry*. Algorithmica 3. 1988. 293-327.Springer-Verlag New York Inc.

[2] N.M. Amato, M.T. Goodrich, and E.A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci., pages 683-694,1994.

[3] M.J. Atallah and D.Z. Chen. Parallel computational geometry. In A.Y. Zomaya, editor, Parallel Computations: Paradigms and Applications, pages 162-197, International Thomson Computer Press, Boston, 1996.

[4] J.E. Goodman and J. O'Rourke, eds., Handbook of Discrete and Computational Geometry, Second Edition, Chapman and Hall/CRC Press, 2004.

[5] M. Ajtai, J. Komlos, and E. Szemeredi (1982). An O(nlog(n)) Sorting Network. Proc. 15th ACM Symposium on Theory of Computing, pp.1-9. Also in Combinatorica, 3(1)(1983), pp. 1-19.

[6] T. Leighton (1984). Tight bounds on the complexity of parallel sorting. Proc. 16th ACM Symposium on Theory of Computing, pp.71-80.

[7] R. Cole (1986). Parallel merge sort. Proc. 27th IEEE FOCS Symposium, pp. 511-516.

About the author

Vasyl Tereshchenko is a professor at National Taras Shevchenko University of Kyiv, Faculty of Cybernetics. His contact email is v_ter@ukr.net.