A Basic Algorithm for Computer-Aided Design of Material Arrangement

Huang Wenqi (黄文奇) and Wang Gangqiang (王刚强)
Dept. of Computer Science, Huazhong University of Science and Technology, Wuhan 430074
Received January 8, 1990; revised August 14, 1990.

Abstract

An algorithm has been obtained for solving the packing problem of placing convex polygons with different shapes and sizes into a rectangular vessel by simulating the elastic mechanics process. It is pointed out that, based on this algorithm, a system of computer-aided design can be developed for arranging two-dimensional materials.

0. Introduction

Let's consider the following problem: It is hoped that N known polygons with different shapes and sizes can be placed into a known rectangular vessel, in which the enclosed boundary of the confined vessel and all the convex polygons are unembeddable rigid objects. If objectively the convex polygons cannot be placed into the vessel, it is required that a judgment be made to that effect, while if they can, it is required that the positions and directions of the objects be given. Obviously, the above problem is in effect a plate cutting problem. It is hoped that N known convex polygons with different shapes and sizes can be cut out on a known rectangular plate.

It has been proved by theoretical computer science that it is very difficult to find a complete exact solution for this problem, which has the so-called \( NP \) hardness. Even if an approximate solution is to be found in accordance with [1], the computing time will more often than not become an astronomical figure in many actual situations. The objective of this paper is to develop the original thoughts in [2] and [3] and obtain the local approximate solution of this problem through the quasi-physical channel, providing basic algorithms to relevant computer-aided designs.

Now we imagine the N polygons as smooth two-dimensional elastic objects and the rectangular vessel as a smooth elastic material filling up the entire two-dimensional space except that a cavity is formed as a rectangle has been cut off. Imagine that the N elastic objects are compressed into the elastic cavity. If our original rigid packing problem can be objectively solved, then a series of motion will take place in the system formed by the elastic objects in which compression exists and the cavity under the action of the elastic forces and ultimately it will be possible for all the objects and the cavity to restore their own shapes and sizes, thus satisfying the condition for the deterministic solution of the packing problem.

Therefore, if we can use a mathematical method to simulate in a certain sense the mode of motion of the objects and cavity under the action of the elastic forces, then we will in fact obtain a numerical solution for the packing problem. Below we shall describe this method.

1. Method Description

We regard the two-dimensional Descartes coordinate system fixed onto the rectangular vessel as an absolute coordinate system. For every convex polygon object, we can assign a two-dimensional Descartes coordinate system to be fixed on it in some way beforehand, and the coordinate system is selected on the geometric centre of the convex polygon. So the state of the...
the \( i \)-th convex polygon object. And now define the elastic deforming potential energy \( U_{vi} \) as
\[
U_{vi} = \begin{cases} 
0, & \text{when } S_{vi} > 0; \\
\frac{S_{vi}^2}{2}, & \text{when } S_{vi} < 0. 
\end{cases}
\] (1.4)

It is quite natural to define \( U_{o,i} \) as
\[
U_{o,i} = \sum_{i=1}^{4} U_{vi}. 
\] (1.5)

Now define the elastic potential energy \( U \) of the system consisting of \( N \) objects and the cavity by the following formula:
\[
U = \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} U_{ij}. 
\] (1.6)

that is, the elastic potential energy of the system formed by \( N+1 \) objects is the total of the elastic potential energy between every two objects.

Obviously, \( U \) is a known function of the system state, i.e.,
\[
U = U(x_1, y_1, \theta_1, \ldots, x_N, y_N, \theta_N). 
\] (1.7)

It can be seen from the construction of function (1.7) that i) \( U \) is always non-negative; ii) \( U > 0 \) indicates that state (1.2) does not satisfy the requirement of placing the \( N \) objects into the vessel; iii) \( U = 0 \) indicates that state (1.2) satisfies the requirement of placing the \( N \) objects into the vessel. We call the state satisfying \( U = 0 \) the feasible point in the state space.

Regarding \( U \) as the objective function in the state space, we can use a computer to find the feasible point in the state space with the slightly modified gradient algorithm described in [4]. When the feasible points are found, the packing problem is said to have been solved, and the array (1.2) indicates the specific posture of placing the \( N \) objects into the vessel.

If the feasible points have not been found after computing a long time, then it is possible that either there is no solution to the problem or that the space is in great demand and the shapes and sizes of the objects are very different from each other and that there are too many objects so that the computation often falls into the non-feasible minimum points. In such a situation we can make use of the graphic display and the relevant input modifying device to interfere artificially with the computation.

The reason the objective function \( U \) is defined with formulae (1.3) — (1.6) rather than the geometrically more natural area of the intersection of two point sets is for the purpose of reducing the frequency of appearance of the non-feasible minimum points in the objective function so as to increase the efficiency of the iterative computation.

2. Algorithm for Finding \( L_{ij} \)

In the process of searching for a solution, it is necessary to repeatedly calculate the distance \( L_{ij} \) between the two polygons with given positions and directions. Now we describe an algorithm for finding \( L_{ij} \).

Connect the geometric centre of polygon \( O_i \) to that of polygon \( O_j \). Fix polygon \( i \), then in the direction \( O_i O_j \) translate polygon \( j \) a distance \( l \) that is great enough so that the two polygons will be far apart (in practice \( l \) is a sufficiently large absolute constant). Now compute the distance \( L_{ij}^* \) between the two polygons in the new position, thus the distance between the two polygons in the original positions is
\[
L_{ij} = L_{ij}^* - l. 
\] (2.1)

Therefore the key situation in the computation of \( L_{ij} \) is one where two polygons are sufficiently far apart. When describing the algorithm for finding \( L_{ij} \), we might as well assume that the two polygons are sufficiently far apart.

First define the positive vertices and edges of polygon \( i \) relative to polygon \( j \). Let a beam