

Re: Algorithm to fit rectangles with different areas inside a matrix with lowest complexity

Source: <http://coding.derkeiler.com/Archive/General/comp.theory/2006-07/msg00045.html>

- *From:* "Dani Camps" <danicamps81@xxxxxxxxxx>
 - *Date:* 12 Jul 2006 03:48:28 -0700
-

Hi Chris,

Thanks for the reply. I had also considered the way you point out as well, these are my impressions:

One thing that I did not say in the first post but it is fair to assume is the following:

"If a certain object under rule 1 has area m , then under rule 2 has area $m*k$ with $k>1$. What I mean with this is that the increase in area that we have for every rule is multiplicative, not additive ($m+k$)."

Then I think, and this is only intuition, if we can assume:

"(1) If object 1 is bigger than object 2 under rule 1 (we could say that the original size is bigger), and the increment when changing rule is the same for all the objects, this means k the same for all the objects, then object 1 will be bigger than object 2 under all the rules."

If (1) holds, then I think the following algorithm should be close to the optimum, if not the optimum:

- 1-I first try to allocate the objects in their smallest area, rule 1.
- 2-If 2 or more objects collide in one place in the matrix and I can not allocate all them, then I place there the object with biggest area. This is the same idea that you are pointing out, I do so because in this way I am minimizing the object with biggest area.
- 3-I try to reallocate the other objects decreasing one rule for all of them.

The point is that every time we have to take a decision, because two or more objects are colliding in one place, we can be sure that we minimize the total area by allocating the biggest one.

But now imagine that we can not assume (1). This basically means that if object 1 is bigger than object 2 under rule 1, maybe object 2 is bigger than object 1 under rule 2, or in another words k is now

Re: Algorithm to fit rectangles with different areas inside a matrix with lowest complexity

different for all the objects.

If now I want to apply an algorithm similar to the one before, I would be in trouble. Imagine that 2 objects collide, then in order to decide which one to place I would have to evaluate the area of placing one under rule 1 and leaving the other under rule 2, and the opposite. But maybe the one I leave under rule 2 then collides with another allocation, then I also have to evaluate this, and the number of possibilities grow and grow yielding to the exhaustive search algorithm.

This is of course if I want to be sure that I get to the optimum solution, I could try to define criterias of the size of an object, and still allocate first the ones with biggest size, for instance:

- 1-The size is the biggest size of the object under all the rules.
- 2-I could get the size averaging the size of the object over all the rules.
- 3-Maybe this average of the size should be a weighted average giving more weight to the lower rules, since I will always try to allocate first according to the lowest rule.

But all these are criterias that I don't see how good or bad they are, maybe the only way to find out is to try them.

Because I am not sure that I can assume (1) I would like to find the best algorithm that compromises optimality and complexity.

BR

Dani

Chris wrote:

Can you point me to some algorithms, that try to solve problems similar to this in an optimum way ?

When you load furniture in a moving van, you put the big objects in first and fill in the small ones later. In this case, "big objects" would be the ones that would be the biggest if they couldn't be placed under rule one or two. Would it be possible to prioritize the objects in this way?

Are there any patterns in the way locations cause objects to be sized? For example, row 2 col 2 always makes an object placed there grow?