

A note on the buildings placement problem

Nicolas Lermé

LAGA UMR CNRS 7539 & LIPN UMR CNRS 7030
Avenue Jean-Baptiste Clément – 93430 Villetaneuse France
`nicolas.lerme@lipn.univ-paris13.fr`

October 9, 2009

1 Introduction

Originally, the towers placement problem comes from a famous game called *Tower Bloxx*, available on some modern mobile phones. Let describe the concept of the game. It consists in building a city block on a 5×5 squared grid. Each tower is built by stacking up floors one by one thanks to a swinging crane. Each time a new floor is added, virtual people come in the tower. The straighter the tower you make, the higher the population you get in the tower. There are four types of towers: residential (blue), industrial (red), commercial (green) and luxury (yellow), increasing in potential population in that order.

In the game, we consider a 4-connexity neighborhood. Then, the rules are the following: blue residential towers can be placed anywhere. Red commercial towers must be placed adjacent to at least one blue tower. Green industrial towers must be placed adjacent to one blue and one red tower. Yellow luxury towers must have one blue, one red, and one green tower neighbor. The picture 1 show a valid city block that follows the previous rules. In the game, you may demolish the required neighboring towers to get a higher population (loophole). For simplicity, we ignore this possibility.



Figure 1: An example of city block.

Then, we could be interested to answer to a deeper question: *what is the optimal city block configuration which maximize the population?* In fact, we guess that this combinatorial optimization problem can be easily expressed in form of an integer linear program. Here, we are only interested to get the optimal solution. First, we generalize this problem with an arbitrary grid size and number of colors. Then, we will give some results with several couples of parameters and finish by a short conclusion.

2 Generalization

Consider a simple non directed graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ with a set of nodes $\mathcal{P} \in \{1 \dots n\}^d$ ($d > 0$) disposed on a multidimensional lattice and a set of edges \mathcal{E} . Each node must takes a color value in a finite set $\mathcal{C} = \{1 \dots k\}$. We denote by (p, q) an edge linking a node p to a node q . A neighborhood system $\mathcal{N} = \{(p, q) \mid p \in \mathcal{P}, q \in \mathcal{N}(p)\}$ is associated to \mathcal{G} , where $\mathcal{N}(p)$ corresponds to the neighborhood of any node p . Also, we can consider several kinds of neighborhoods involving an increasing number of neighbors:

$$\begin{aligned} \mathcal{N}_0(p) &= \{q : \sum_{i=1}^d |q_i - p_i| = 1\} & \forall p \in \mathcal{P} \\ \mathcal{N}_1(p) &= \mathcal{N}_0(p) \cup \{q : |q_i - p_i| = 1 \forall 1 \leq i \leq d\} & \forall p \in \mathcal{P} \\ \mathcal{N}_2(p) &= \mathcal{N}_1(p) \cup \{q : \sum_{i=1}^d |q_i - p_i| = 3, \prod_{i=1}^d |q_i - p_i| \neq 0\} & \forall p \in \mathcal{P} \end{aligned}$$

Next, for any node p we associate a binary vector of variables $x_p = (x_p^1, \dots, x_p^k)$ where $x_p^c = 1$ means that node p has color c , $\forall 1 \leq c \leq k$. Moreover, we need a strictly increasing function $f : \mathcal{C} \mapsto \mathbb{N}$ to count the number of citizens in a tower.

Thus, this leads to the following LP formulation:

$$\begin{aligned} \max \quad & \sum_{c \in \mathcal{C}} f(c) \times \left(\sum_{p \in \mathcal{P}} x_p^c \right), \\ \text{s.t.} \quad & \sum_{c \in \mathcal{C}} x_p^c = 1 & \forall p \in \mathcal{P}, \\ \text{s.t.} \quad & x_p^c \leq \left(\sum_{q \in \mathcal{N}(p)} x_q^d \right) & \forall p \in \mathcal{P}, \forall c, d \in \mathcal{C} \quad \text{s.t.} \quad (c > 1) \wedge (d < c). \end{aligned}$$

The objective function sums the amount of citizens for each color and each point on the grid. The first constraint ensure the sum of vector components to be 1 since we must assign exactly one color to each tower. The second ensure a tower of color $c > 1$ to have at least one neighboring tower for each color stricly lower than c . In the next section, we give some results on our experiments.

3 Experiments

We present some experiments for 2D grid graphs with a \mathcal{N}_0 neighborhood. The table 3 below give the average time resolution for several couples of parameters (n, k) over 10 launchings.

The tests were run on a Intel Xeon @ 3.0 Ghz, 2 Go of RAM using the CPLEX solver. In our experiments, we have chosen the function f as being equal to $(n + 1)^{2^c}$.

First, we see that the time resolution grows extremely quickly with the grid size and the number of colors. This can be explained in the sense the number of constraints to satisfy become more and more important. We also have noticed that the time resolution is faster with a \mathcal{N}_1 neighborhood or beyond.

$n \backslash k$	2	3	4	5
2	$t \simeq 0.0$	$t = 0.01$	$t = 0.03$	$t = 0.03$
3	$t \simeq 0.0$	$t = 0.04$	$t = 0.17$	$t = 0.43$
4	$t \simeq 0.0$	$t = 0.11$	$t = 1.13$	$t = 1.88$
5	$t \simeq 0.0$	$t = 0.59$	$t = 9.30$	$t = 19.41$
6	$t = 0.01$	$t = 3.10$	$t = 161.74$	$t = 123.724$
7	$t = 0.07$	$t = 11.68$	$t = 7237.69$	$t = 22489.28$
8	$t = 0.25$	$t = 162.83$	$t \geq 59040.56$	$t \geq 6749.73$
9	$t = 0.37$	$t = ?$	$t \geq 3154.54$	$t = ?$

Figure 2: Average time resolution (in seconds) for several couples of parameters (n, k) .

4 Conclusion

After introducing the concept of the game, we have seen how our problem can be formulated to be solved efficiently with linear programming. We are able to answer to our initial question by computing the optimal solution of the instance $(n = 5, k = 4)$. For the moment, the results are quite encouraging. The time resolution take a long time (since we want to get the exact solution) and grows extremely quickly with k and n . Nevertheless, we have noticed that the optimal solution is generally obtained only in few iterations. One could be interested to hold further investigations in order to develop basic heuristics.

Moreover, we could imagine some extensions to this problem:

- Use more complex dependency rules between towers.
- Consider the problem with loophole.

Besides, as it is often the case, our initial question call other ones. Especially, it remains unclear to us if this problem have been already studied before or not (eventually presented

under another form). If so, what is its complexity, for any k fixed? In the graph coloring literature, It seems to be a weak coloring problem under particular constraints [1]. However, our problem is harder because we want also to maximize the number of nodes with highest color value.

Also, we are interested to know the complexity for $k = 2$ and $k = 3$. More precisely, it would be interesting to know on the one hand, if there is a polynomial algorithm for $k = 2$ and on the other hand, if this problem is *NP-hard* for $k \geq 3$. Finally, the source code of the linear program is available upon request.

References

- [1] Wikipédia. Weak coloring problem: http://en.wikipedia.org/wiki/Weak_coloring.