Shooting Bricks with Orthogonal Laser Beams: A First Step towards Internal/External Map Labeling

Maarten Löffler[*] Martin Nöllenburg[*]

Abstract

We study several variants of a hybrid map labeling problem that combines the following two tasks: (i) a set A of points in a rectangle R needs to be labeled with rectangular labels on the right boundary of R using rectilinear one-bend polylines called *leaders* to connect points and labels; (ii) a maximum subset B' of a set B of fixed internal congruent rectangular labels in R needs to be selected such that B' is an independent set of labels and no leader intersects any label in B'. We also call the points in A *aliens*, the labels of B *bricks*, and the leaders *laser beams*. Then the problem translates into every alien shooting a laser beam so that in total as few bricks as possible are destroyed. We provide algorithms and NP-hardness results for different variants of the problem.

1 Introduction

Assume that we are given a rectangular map R with n points that are to be labeled by rectangular labels. It is required for readability of the map that no label overlaps another label or any of the input points. In *internal* labeling models, each label must be close to the labeled point, i.e., it usually needs to touch the point on its boundary. For instance, in the p-position model for $p \in \{1, 2, 4\}$ labels must touch the points at one of p admissible corners. Maximizing the number of labels that can be placed is NP-hard for any such p [4, 7].

An alternative *external* labeling model, in which all n input points can always be labeled (for sufficiently large R), is known as *boundary labeling* [1–3, 6]. In this model all labels are placed on one, two, or four sides of the boundary of R. Points are connected to their labels with arcs called *leaders*. In order to keep the map clean, leaders are usually required to be crossing-free and to have a prescribed simple shape, e.g., rectilinear polylines with at most one bend. One may require the leaders to be spaced by at least the height of a label, so that there is no label overlap on the boundary; alternatively, one could create the necessary spacing of the labels, e.g., by using two extra bends per leader in the so-called track-routing area [2] outside of R. Typical optimization criteria in boundary labeling are minimizing the total leader length or minimizing the total number of bends.

[*]Department of Computer Science, University of California, Irvine, mloffler@uci.edu, noellenburg@kit.edu

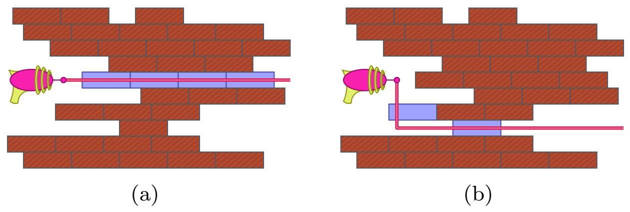

Figure 1: An alien surrounded by bricks. (a) Shooting straight destroys four bricks. (b) The optimal solution.

In the boundary labeling model, however, *all* labels must be on the boundary of R, even though some points would allow for an internal label, which is generally more favorable from the application's perspective. One of the open problems mentioned by Bekos et al. [1] and Kaufmann [6] is to study map labeling in a mixed model, where some points receive internal (fixed-position) labels and the remaining points are labeled externally. In such a mixed model, an additional requirement is that no leader may cross any of the internal labels.

In this paper, as a first step towards map labeling in the mixed model, we assume that the n input points are already partitioned into two sets A and B. The points in A must all be labeled externally using so-called *po*-leaders (they first run *p*arallel to and then *o*rthogonal to the labeled side of R) [2], which have by definition at most one bend. We consider the *one-sided* boundary labeling model, where all labels are placed on the right side of R. The points in B, on the other hand, can only be labeled internally using congruent rectangular labels with a fixed position. Depending on the placement of the leaders for A, some points in B may or may not be labelable. The goal is to label all points in A and to maximize the number of labeled points in B.

In the following we call the points in A *aliens*. We identify each point in B with its fixed label and call the labels in B *bricks*. Instead of connecting each alien with a leader to the right side of R, we say that each alien has a laser gun that can shoot a one-bend rectilinear *laser beam* to the right side of R. Figure 1 shows an example. Each laser beam consists of a (possibly empty) vertical *laser segment* followed by a horizontal *laser ray*. Consequently, we can say that a laser beam *destroys* all

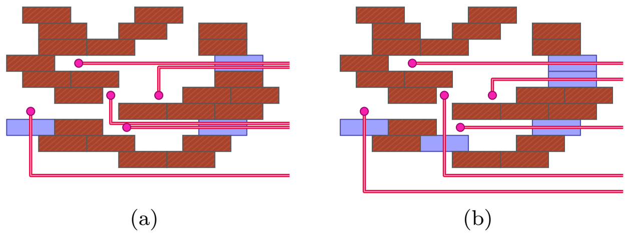

Figure 2: An instance with five aliens. (a) For the non-spaced ABP at least three bricks must be destroyed. (b) For the spaced and crossing ABP at least five bricks must be destroyed.

bricks that are hit by the laser beam. The map labeling problem then translates into the problem that every alien must shoot its laser gun exactly once in a way that minimizes the total number of destroyed bricks. We call this problem the *aliens-and-bricks problem* (ABP).

Parameters The ABP can be further described by three independent parameters that yield in total eight different versions of the problem. Figure 2 shows two examples.

1. Bricks are either all disjoint or they have overlaps; in the latter case we need to find an independent set of bricks. Accordingly, we call an ABP instance *disjoint* or *non-disjoint*. We may further require that the independent set of bricks does not occlude a given, e.g., the top left, corner of any brick.
2. Laser beams can either cross each other at 90-degree angles or they cannot cross each other. The options are denoted as *crossing* and *non-crossing*.
3. Horizontal laser rays can either come arbitrarily close to each other or they have to stay at least one unit apart. Accordingly, we call an ABP instance *spaced* or *non-spaced*. We may further require that the vertical laser segments are also spaced.

2 Non-Spaced Variant

In this section we consider non-spaced laser beams. We note that in this case the issue of crossings is irrelevant: any crossing could be removed at no additional cost by re-routing the laser beam whose vertical segment is involved in the crossing so that the beam turns horizontal just before hitting the other laser beam. Hence, we restrict ourselves to non-crossing lasers beams.

2.1 Disjoint Bricks

When the interiors of the bricks are all disjoint, we can solve the problem in polynomial time using dynamic programming. Assume all laser beams are in an optimal configuration that does not have any crossings. Let $a \in A$ be an alien and consider the right open half-strip S that is defined by the vertical line ℓ_a through a, the y-coordinate y_t of the lowest laser ray above a, and the y-coordinate y_b of the highest laser ray below a. Because we have no crossings, a must shoot somewhere into S, which subdivides S into two smaller strips. Furthermore, let $B_a \subset B$ be the set of bricks that are stabbed by ℓ_a and lie completely between y_b and y_t. Then an optimal solution for all aliens inside S (including a) depends only on S and, additionally, on which bricks of B_a have already been shot by previous aliens outside S. Figure 3 shows an example of such a strip and how it divides into two independent smaller subproblems.

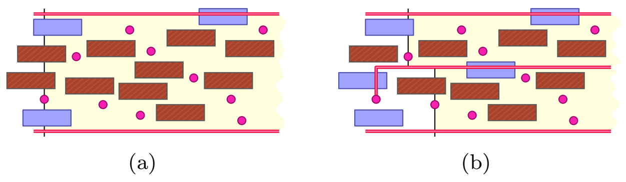

Figure 3: (a) A subproblem defined by a current alien (indicated by the black vertical line ℓ), two laser rays above and below it, and a marking of the bricks stabbed by ℓ. (b) A possible way for the current alien to shoot, and the resulting two subproblems.

The algorithm is now standard dynamic programming. We store the solution of each subproblem, defined by a strip S and a subset of the corresponding set of bricks B_a. To solve a subproblem, we go over the linear number of combinatorially different possible ways to shoot the current beam b, look up the resulting two subproblems, and add to their joint cost the number of bricks newly destroyed by b. We choose the solution with minimum cost. The number of bricks destroyed by b can be computed by sweeping b over the half-strip of the subproblem and keeping track of the number of intersected bricks. This clearly takes linear time for each subproblem.

However, it is not so clear that the algorithm is efficient, since there could in principle be an exponential number of subsets of B_a that are marked as unshot, and therefore an exponential number of subproblems to solve. We proceed by showing that this is not the case and that it suffices to consider a linear number of subsets of B_a.

Lemma 1 *Let S be a half-strip defined as before by an alien a and two horizontal laser rays. There is only a linear number of subsets of B_a that can be shot by aliens on the left of ℓ_a.*

Proof. (Sketch) The basic idea is that the set of bricks that is destroyed depends only on the pieces of laser beams that are visible from ℓ_a. Now, if we fix an alien a' as the leftmost alien that can still see ℓ_a, then this determines the direction in which all aliens between a' and a must shoot (in order not to block a'). We show that this leads to a quadratic number of configurations of visible laser beams, of which only a linear number can hit a different subset of B_a. □

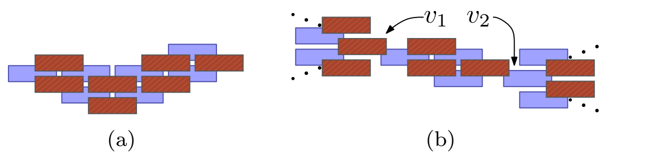

Figure 4: (a) An edge is a diagonal strip of width 2 of overlapping bricks. It has two states with equal cost. (b) A vertex is a single brick that overlaps three edges. Its two vertices overlap different states of an edge: if v_1 is present then v_2 cannot be.

There are clearly $O(n^3)$ essentially different half-strips S that can define a subproblem. For each strip, by Lemma 1, we need to consider a linear number of subsets of B_a. So we have a total of $O(n^4)$ possible subproblems. Finally, looking up a specific subproblem in constant time is not trivial since it is not clear how to index a particular subset of bricks. We suggest to order all bricks in B, and then use for each half-strip defined by the three parameters (a, y_t, y_b) a simple search tree according to the order of B to access the correct value corresponding to a particular subset of bricks of B_a. By Lemma 1, this tree has linear size, but its height can also be linear. Therefore, looking up the value of a particular subproblem takes linear time in the worst case. This yields the following result:

Theorem 2 *The disjoint, non-spaced ABP can be solved in $O(n^6)$ time using $O(n^4)$ space.*

2.2 Overlapping Bricks

For non-disjoint bricks it is generally NP-hard to find a maximum independent set of bricks, even when there are no aliens. Klau and Mutzel [7] refer to an unpublished result by Woeginger that this can be proven using a reduction from the NP-complete maximum independent set problem in planar graphs with maximum degree 3 [5]. For completeness, we briefly sketch how this can be done: we "draw" the planar input graph using edges made of overlapping bricks as in Figure 4(a), and using a single brick for each vertex as in Figure 4(b). Then the reduction is immediate.

Theorem 3 *The overlapping, non-spaced ABP without further restrictions is NP-hard.*

However, motivated by the 1-position map labeling model, we may require that no brick in B contains a fixed corner (say, the top left one) of any other brick as that corner coincides with the labeled point. In this case overlapping bricks form only simple monotone structures for which the independent set problem can be easily solved. In fact, the algorithm for disjoint bricks still works with a few modifications. When solving a subproblem, still defined by a half-strip S and a subset of bricks, for each choice of the y-coordinate for the bend in a's beam we need to choose an independent set of bricks in the area of S that is not covered by the strips of the subproblems. We show in the full version of the paper that it suffices to greedily take bricks from left to right. This greedy process may result in some bricks overlapping the left boundary of the subproblem strips to not be chosen, but we can still show an equivalent of Lemma 1 in this case. We arrive at the following result:

Theorem 4 *The overlapping, non-spaced ABP, where bricks cannot contain top left corners of other bricks can be solved in $O(n^6)$ time using $O(n^4)$ space.*

3 Spaced Variant

We now add the restriction that the horizontal laser rays must be vertically spaced. In this case, it does matter whether we allow beams to cross or not. We also discuss horizontal spacing of the vertical laser segments.

3.1 Crossing Variant

We show that the ABP is NP-hard if we require vertically spaced horizontal laser rays and allow crossings between laser beams. The construction does not use overlapping bricks. Let I be an instance of MAX2SAT with n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$ such that each variable x_i appears in at most three clauses. The problem of finding a truth value assignment for the variables that maximizes the number of satisfied clauses in I is NP-complete [8]. Figure 5 shows an example of our reduction with four variable and four clause gadgets.

The variable gadget consists of an alien whose laser beam is vertically blocked by two "indestructible" bricks (actually large blocks of $C \times C$ bricks for large enough C). Towards the right, each variable alien has an upper and a lower choice of shooting its laser beam, both with the same cost, corresponding to the values *true* and *false* of the variable. Furthermore, there is a "wall" of bricks to the far right that has to be crossed by any laser beam, unless it runs below or above the wall.

For a clause c_k in the formula containing variables x_i and x_j, we place an alien a in a new row somewhere between the gadgets for x_i and x_j. Assuming x_i is above x_j, if x_i is a positive (negative) literal of c_k, we add a clause-connector brick vertically above a in the upper (lower) row of x_i, and we analogously add a clause-connector brick for the literal of x_j vertically below a. For every clause we use a new column so that a vertical line through any clause alien intersects exactly the two connector bricks for that clause.

Now, if one of the literals of a clause is *true* then its corresponding clause-connector brick is destroyed. Hence the clause alien can shoot its laser beam vertically through that brick at no cost until it reaches beyond the construction, where it can turn to the right and reach

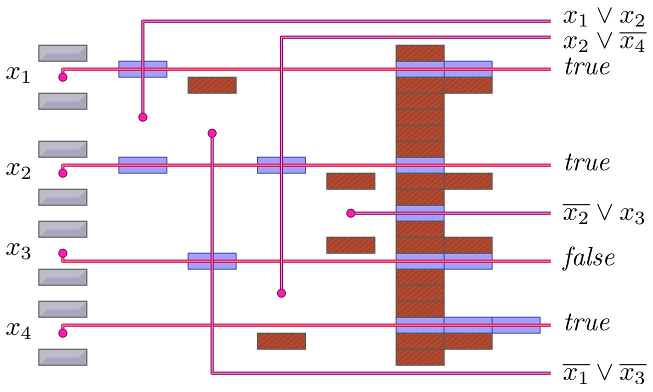
$x_1 \vee x_2$
$x_2 \vee \overline{x_4}$
true
true
$\overline{x_2} \vee x_3$
false
true
$\overline{x_1} \vee \overline{x_3}$
x_1
x_2
x_3
x_4

Figure 5: Aliens (points) and bricks representing a MAX2SAT formula with four variables and four clauses.

the right side of R. If, however, both literals of a clause are *false*, then no matter how the alien shoots its laser beam, it must destroy at least one extra brick.

Theorem 5 *The spaced and crossing ABP is NP-hard. This is independent of the disjointness of the bricks.*

3.2 Non-Crossing Variant

Interestingly, if we restrict the laser beams further by not allowing them to cross, we can solve the problem in polynomial time again. We omit the details, but essentially we need to shrink the half-strip S defining a subproblem by the appropriate amount.

Theorem 6 *The spaced non-crossing ABP can be solved in $O(n^6)$ time using $O(n^4)$ space, both for the disjoint and non-disjoint versions of ABP.*

3.3 Horizontal Spacing

When we require laser beams to be horizontally spaced as well as vertically, a solution is not always possible. If we do allow crossings, then the NP-hardness proof of the previous section can be adapted to show that this problem is also hard. Without crossings, however, we can find an optimal solution (if it exists) in $O(n^4)$ time, assuming unit spacing and constant width bricks. Again we omit the details, but the main reason is that the number of aliens involved in shooting bricks in B_a is limited, so instead of a linear number of subsets we only need to consider a constant number.

Theorem 7 *The doubly spaced non-crossing ABP can be solved in $O(n^4)$ time using $O(n^3)$ space.*

4 Discussion

We studied six main variants of the ABP and presented algorithms or NP-hardness results for each variant. There is a trade-off between the different versions: by adding more restrictions (such as not allowing crossings or requiring vertical spacing) the resulting drawing may look cleaner, but obviously fewer bricks can be placed. Perhaps surprisingly, the least and most restricted variants of the problem can both be solved in polynomial time using essentially the same algorithm, while one of the in-between variants is NP-hard. Apart from the variants we studied, many more variants are conceivable by considering different ways for internal labeling (more or different label positions) or external labeling (more bends per leader, labels at multiple sides of R) that would be interesting to study.

We assumed that the sets of aliens A and bricks B are given as two separate sets. However, in the map labeling formulation of the problem, it may be more realistic to assume that all input points belong to a single set, and that every point is labeled either internally or externally. We then want to maximize the number of internally labeled points. One easy observation is that if no two internal labels (bricks) overlap, the problem becomes trivial since in the optimal solution all points will be labeled internally. The hardness result of Theorem 5 does extend to this situation after some adaptation. Mostly though, this version of the problem, suggested by [1,6], remains open. We hope that our results in this note constitute a first step towards solving it.

Acknowledgments. This research was supported by the German Research Foundation under grant NO 899/1-1 and by the Office of Naval Research under MURI grant N00014-08-1-1015.

References

[1] M. Bekos, M. Kaufmann, M. Nöllenburg, and A. Symvonis. Boundary labeling with octilinear leaders. *Algorithmica*, 57:436–461, 2010.

[2] M. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Com. Geo. Th. Ap.*, 36:215–236, 2007.

[3] M. Benkert, H. Haverkort, M. Kroll, and M. Nöllenburg. Algorithms for multi-criteria boundary labeling. *J. Graph Algorithms Appl.*, 13(3):289–317, 2009.

[4] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Ann. ACM Symp. Comput. Geom. (SoCG'91)*, pages 281–288, 1991.

[5] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, 32(4):826–834, 1977.

[6] M. Kaufmann. On map labeling with leaders. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, LNCS 5760, pages 290–304. Springer-Verlag, 2009.

[7] G. W. Klau and P. Mutzel. Optimal labeling of point features in rectangular labeling models. *Mathematical Programming*, 94(2):435–458, 2003.

[8] V. Raman, B. Ravikumar, and S. S. Rao. A simplified NP-complete MAXSAT problem. *IPL*, 65:1–6, 1998.