Labeling downtown

Eidgenössische
Technische Hochschule
Zürich

Departement Informatik
Institut für
Theoretische Informatik

Labeling Downtown

Gabriele Neyer
Frank Wagner

Technical Report #324, 1999

Mai 1999

ETH Zürich
Departement Informatik
Institut für Theoretische Informatik
Prof. Dr. P. Widmayer

Address:

G. Neyer, Institut für Theoretische Informatik,
ETH Zentrum, CH-8092 Zurich, Switzerland,
e-mail: neyer@inf.ethz.ch

This report is available via http://www.inf.ethz.ch/publications/tech-reports or
<ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/

Labeling Downtown[†]

Gabriele Neyer[‡] and Frank Wagner [§]

Abstract

American cities, especially their central regions usually have a very regular street pattern: We are given a rectangular grid of streets, each street has to be labeled with a name running along its street, such that no two labels overlap. For this restricted but yet realistic case an efficient algorithmic solution for the generally hard labeling problem gets in reach.

The main contribution of this paper is an algorithm that guarantees to solve every solvable instance. So far we are not able to provide a runtime analysis that guarantees efficiency, but the empirical behavior is polynomial without a single exception. The complexity status of the problem is open, we show that a slight generalization, namely the labeling of a cylinder shaped downtown, is $\mathcal{NP}$-hard.

Finally, we present efficient algorithms for three special cases including the case of having no labels that are more than half the length of their street.

1 Introduction

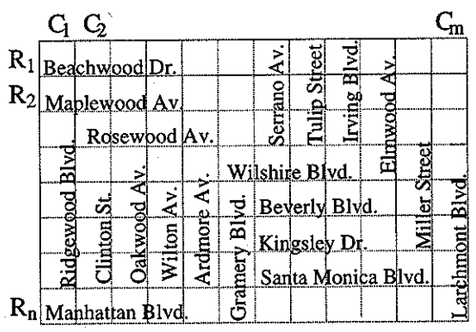

Figure 1: American downtown street map.

The general city map labeling problem is too hard to be automated yet [NH]. In this paper we focus on the downtown labeling problem, a simple special case, that is still non-trivial. In fact, the complexity status is open.

The clearest way to model it, is to abstract a grid-shaped downtown street pattern into a chessboard of adequate size. The names to be placed along their streets we abstract to be tiles that w.l.o.g. span an integer number of fields.

A feasible labeling then is a conflict free tiling of the board placing all the labels along their streets.

Our main algorithm is kind of an adaptive backtracking algorithm that is guaranteed to find a solution if there is one, and that empirically has a strictly bounded depth of backtracking, namely one. We conjecture it to be efficient in general.

[†]This work was partially supported by grants from the Swiss Federal Office for Education and Science (Projects ESPRIT IV LTR No. 21957 CGAL and N0. 28155 GALIA), and by the Swiss National Science Foundation (grant "Combinatorics and Geometry").

[‡] Institut für Theoretische Informatik, ETH Zürich, CH-8092 Zürich, email:neyer@inf.ethz.ch

[§] Transport-, Informatik- und Logistik- Consulting (TLC), Kleyerstraße 27, D-60326 Frankfurt am Main and Institut für Informatik, Freie Universität Berlin, Takustraße 9, D-14195 Berlin, email:Frank.Wagner@tlc.de

The complexity of our problem is unknown, but there is a well studied family of related problems from Discrete Tomography [Woe96, GG95] that yields a NP-hardness result for a slightly more general labeling problem, taking place on a cylinder instead of a rectangle.

We round up the paper by giving efficient solutions to special cases: There is a polynomial algorithm, if

- no label is longer than half of its street length
- all vertical labels are of equal length
- the map is quadratic and each label has one of two label lengths

One general remark that helps to suppress a lot of formal overhead: Often, we only discuss the case of horizontal labels or row labels and avoid the symmetric cases of vertical labels and columns or vice versa.

2 Problem Definition

Let G be a grid consisting of n rows and m columns. Let $R = \{R_1, \ldots, R_n\}$ and $C = \{C_1, \ldots, C_m\}$ be two sets of labels. The problem is to label the i^{th} row of G with R_i and the j^{th} column of G with C_j such that no two labels overlap. We will represent the grid G by a matrix.

Definition 2.1 (Label problem (G, R, C, n, m)**)**

Instance: *Let $G_{n,m}$ be a two dimensional array of size $n \times m$, $G_{i,j} \in \{\emptyset, r, c\}$. Let R_i be the label of the i^{th} row and let r_i be the length of label R_i, $1 \leq i \leq n$. Let C_i be the label of the i^{th} column and let c_i be the length of label C_i.*

Problem: *For each row i set r_i consecutive fields of $G_{i,.}$ to r and for each column j set c_j consecutive fields of $G_{.,j}$ to c.*

Of cause no label can be longer than the length of the row or column, respectively.

Initially, we set $G_{i,j} = \emptyset$ which denotes that the field is not yet set. Let $[a, b[$ be an interval such that $G_{i,x} \in \{\emptyset, r\}$, for $x \in [a, b[$. We say that $G_{i,[a,b[}$ is free for row labeling. Furthermore, this interval has length $b - a$. We also say that $G_{i,.}$ contains two disjoint *intervals* of length at least $\lfloor \frac{b-a}{2} \rfloor$ that are free for row labeling, namely $[a, a + \lfloor \frac{b-a}{2} \rfloor[$ and $[a + \lfloor \frac{b-a}{2} \rfloor, b[$.

3 General Rules

Assume we have a label with length longer than half of its street length. No matter how we position the label on its street, there are some central fields in the street that are always occupied by this label. We therefore can simply mark these fields *occupied.* It is easy to see that these occupied fields can produce more occupied fields. The following rules check whether there is sufficiently large space for each label and determines occupied fields.

Rule 3.1 (Conflict) *Let $I = [a, b[$ be the longest interval of row i that is free for row labeling. If $r_i > b - a$, then row i can not be labeled, since it does not contain enough free space for row labeling. In this case we say that a* conflict *occurred and it follows that the instance is not solvable.*

Rule 3.2 (Large labels) *Let $I = [a, b[$ be the only interval in $G_{i,.}$ that is free for feasible row labeling. Observe that the fields that are occupied simultaneously when R_i is positioned leftmost and rightmost in I have to be occupied by R_i no matter where it is placed. These fields we set to r and call them preoccupied.*

Procedure 1 PREPROCESSING (G, R, C, n, m)

```
1) repeat {
2)       G' = G;
3)       run Rule 3.2 on (G,R,C,n,m) and on (G^T,C,R,m,n);
4)       if Rule 3.1 yields a conflict on (G,R,C,n,m) or on (G^T,C,R,m,n) then
5)         return "conflict";
6)}until (G = G';)
7) return true;
```

Our PREPROCESSING Procedure 1 iteratively executes the Rules 3.1 and 3.2 until none of them yields a further change to the label problem or a conflict occurrs. In the latter case we have that the instance is not solvable. We will spell out special cases where the successful preprocessing implies solvability. Furthermore, the preprocessing underlies the following considerations.

For each unfixed label that is limited to just one interval of at most twice its length or to two intervals of exactly its length we can check whether these labels can be simultaneously positioned without conflicts. This can be done since all possible label positions of these rows and columns can be encoded in a set of 2SAT clauses, the satisfaction of which enforces the existence of a conflict free label positioning of these labels. On the other hand a conflict free label positioning of these labels implies a satisfying truth assignment to the set of clauses. Even, Itai and Shamir [EIS76] proposed a polynomial time algorithm that solves the 2SAT problem in time linear in the number of clauses and variables.

We therefore represent each matrix field $G_{i,j}$ by two boolean variables. We have the boolean variable $G_{i,j} = r$ and its negation $\overline{G_{i,j} = r}$ which means $G_{i,j} \neq r$ or $G_{i,j} \in \{\emptyset, c\}$. As the second variable we have $G_{i,j} = c$ and its negation $\overline{G_{i,j} = c}$ which means $G_{i,j} \neq c$ or $G_{i,j} \in \{\emptyset, r\}$. Of course these two variables are coupled by the relation $(G_{ij} = r) \rightarrow (\overline{G_{ij} = c})$.

Those rows and columns, where the possible label positions are limited to just one interval of at most twice its length or to two intervals of exactly it length, we call *dense.* We now encode all possible label positions of the dense rows and columns in a set of 2SAT clauses the satisfaction of which yields a valid labeling of these rows and columns and vice versa.

Property 3.1 (Density Property I) *Let $G_{i,.}$ be a row that contains exactly two maximal intervals each of length r_i that are disjoint and free for feasible row labeling. Let these intervals be $[a, b[$ and $[c, d[$, $1 \leq a < b < c < d \leq n + 1$. Then, a valid labeling exists if and only if the conditions*

1. $(G_{i,a} = r) \leftrightarrow (G_{i,a+1} = r) \leftrightarrow (G_{i,a+2} = r) \leftrightarrow \cdots \leftrightarrow (G_{i,b-1} = r)$,
2. $(G_{i,c} = r) \leftrightarrow (G_{i,c+1} = r) \leftrightarrow (G_{i,c+2} = r) \leftrightarrow \cdots \leftrightarrow (G_{i,d-1} = r)$,
3. $(G_{i,a} = r) \nleftrightarrow (G_{i,c} = r)$

are fulfilled.

$(G_{i,a} = r) \leftrightarrow (G_{i,a+1} = r)$ can be written as the 2SAT clauses $(\overline{G_{i,a} = r} \vee G_{i,a+1} = r)$, $(G_{i,a} = r \vee \overline{G_{i,a+1} = r})$ and since the condition $(G_{i,a} = r) \nleftrightarrow (G_{i,c} = r)$ can be written as $(\overline{G_{i,a} = r} \vee \overline{G_{i,c} = r})$, $(G_{i,a} = r \vee G_{i,c} = r)$ it is easy to see that the complete Densitiy Property 3.1 can be written as a set of 2SAT clauses. The feasible label placements are $(G_{i,a} = r, \ldots, G_{i,b-1} = r)$ and $(G_{i,c} = r, \ldots, G_{i,d-1} = r)$.

Property 3.2 (Density Property II) *Let $G_{i,}$. be a row that contains only one maximal interval $[a, b[$ that is free for feasible row labeling, $r_i < b - a \leq 2r_i$. Then, a valid labeling for the row exists if and only if the conditions*

1. $(G_{i,a} = r) \rightarrow (G_{i,a+1} = r) \rightarrow (G_{i,a+2} = r) \rightarrow \cdots \rightarrow (G_{i,b-r_i-1} = r)$,
2. $G_{i,b-r_i} = r, \ldots, G_{i,a+r_i-1} = r$, *and*
3. $(G_{i,a} = r) \nleftrightarrow (G_{i,a+r_i} = r), (G_{i,a+1} = r) \nleftrightarrow (G_{i,r_i+1} = r), \ldots,$
   $(G_{i,b-r_i-1} = r) \nleftrightarrow (G_{i,b} = r)$

are fulfilled.

Analogously to the first Density Property, the conditions of the second Density Property can be formulated as a set of 2SAT clauses. All feasible label placements are $(G_{i,a} = r, G_{i,a+1} = r, \ldots, G_{a+r_i-1} = r)$, $(G_{i,a+1} = r, G_{i,a+2} = r, \ldots, G_{a+r_i} = r)$, $(G_{i,a+2} = r, G_{i,a+3} = r, \ldots, G_{a+r_i+1} = r)$, $\ldots$, $(G_{i,b-r_i} = r, G_{i,b-r_i+1} = r, \ldots, G_{i,b} = r)$. Note that the properties work analogously for columns.

Theorem 3.1 *The 2SAT formula of all dense rows and columns can be created in $\mathcal{O}(nm)$ time. The 2SAT instance can be solved in $\mathcal{O}(nm)$ time.*

Proof: The number of variables is limited by $2nm$. For each dense row we have at most $\frac{3}{2}n$ clauses. Analogously, for each dense column we have at most $\frac{3}{2}m$ clauses. Altogether we have $\mathcal{O}(nm)$ clauses. Thus, the satisfiability of the 2SAT instance can be checked in $\mathcal{O}(nm)$ time [EIS76]. □

Procedure 2 calls Procedure 1, our *preprocessing*. In case of success, all dense rows and columns are encoded as a set of 2SAT clauses with the aid of Density Property 3.1 and 3.2. Then, their solvability is checked e.g. by invoking the 2SAT algorithm of Even, Itai and Shamir [EIS76].

Procedure 2 DRAW_CONCLUSIONS (G, R, C, n, m)

```
1) if PREPROCESSING(G,R,C,n,m) then{
2)       F := the set of 2SAT clauses of the dense rows and column;
3)       if F is satisfiable then return true;}
4) return false;
```

Lemma 3.1 *The* PREPROCESSING *Procedure 1 and the* DRAW_CONCLUSIONS *Procedure 2 can be implemented in $\mathcal{O}(nm(n+m))$ time.*

Proof: The rules only need to be applied to those rows and columns in which an entry was previously set to r or c. A setting of a field $G_{i,j}$ can only cause new settings in row i or

column j, which by themselves can again cause new settings. The application of the rules on a row and a column takes time $\mathcal{O}(n+m)$. Since at most $2nm$ fields can be set we yield that the preprocessing can be implemented such that its running time is $\mathcal{O}(nm(n+m))$. In Theorem 3.1 we proved that the 2SAT clauses can be generated and checked for solvability in $\mathcal{O}(nm)$ time. Thus, in total we need at most $\mathcal{O}(nm(n+m))$. □

Thus, we can solve dense problems:

Theorem 3.2 *In case that each row and each column of a preprocessed labeling instance (G, R, C, n, m) either fulfills the Density Property 3.1 or 3.2, Procedure 2* DRAW_CONCLUSIONS *decides if the instance is solvable. In case of solvability we can generate a valid labeling from a truth assignment. The overall running time is bounded by $\mathcal{O}(nm(n+m))$.*

4 A General Algorithm

In this section we describe an algorithmic approach with a backtracking component that solves any label problem. Empirically it uses its backtracking ability in a strictly limited way such that its practical runtime stays in the polynomial range. After performing the PREPROCESSING and satisfyability test for dense rows and columns (see Procedure 2 DRAW_CONCLUSIONS), we adaptively generate a tree that encodes all possible label settings of the label problem. Each node in the first level of the search tree corresponds to a possible label setting for the first row label. In the i^{th} level the nodes correspond to the possible label settings for the i^{th} row, depending on the label settings of all predecessor rows. Thus, we have at most m possible positions for a row label and at most n levels. Our algorithm searches for a valid label setting in this tree by traversing the tree, depth-first, generating the children of a node when necessary.

In the algorithm, we preprocess matrix G and check the solvability of the dense rows and columns by invoking Procedure 2 DRAW_CONCLUSIONS. We further mark all these settings permanently. When we branch on a possible label setting for a row, we increase the global timestamp, draw all conclusions this setting has for the other labels by invoking Procedure 2 DRAW_CONCLUSIONS and timestamp each new setting. These consequences can be a limitation on the possible positions of a label or even the impossibility of positioning a label without conflicts. After that, we select one of the newly generated children, increase the timestamp and again timestamp all implications. When a conflict occurs, the process resumes from the deepest of all nodes left behind, namely, from the nearest decision point with unexplored alternatives. We mark all timestamps invalid that correspond to nodes that lie on a deeper level than the decision point. This brings the matrix G into its previous state without storing each state separately. Let the algorithm return a valid label setting for all rows. Since Procedure 1 ensures that each column i contains an interval of length at least c_i that is free for column labeling we can simply label each column and yield a valid label setting. The algorithm is given in Algorithm 1, and in the Procedures 1, 2, and 3.

Algorithm 1 LABEL (G, R, C, n, m)

```
1) timestamp:= 1;
2) if DRAW_CONCLUSIONS(G,R,C,n,m) yields a conflict
3)       return "not solvable";
4) timestamp each setting;
5) let w be the first row that is not yet labeled;
6) if POSITION_AND_BACKTRACK(w,G,R,C,n,m,timestamp) {
7)       label all columns that are not yet completely labeled;
8)       return G; }
9) else
10)      return "not solvable";
```

Procedure 3 POSITION_AND_BACKTRACK $(w, G, R, C, n, m, timestamp)$

```
1) while there are untested possible positions for label r_w in row w {
2)       local_timestamp:=timestamp:=timestamp+1;
3)       label row w with r_w in one of these positions;
4)       timestamp each new setting;
5)       if DRAW_CONCLUSIONS(G,R,C,n,m) then {
6)          timestamp each new setting;
7)          if there is a row w that is not yet labeled {
8)             if POSITION_AND_BACKTRACK(w,G,R,C,n,m) then
9)                return true;
10)         }
11)         else return true;
12)      }
13)      timestamp each new setting;
14)      mark local_timestamp invalid;
15) }
16) return false;
```

We implemented the backtracking algorithm and tested it on over 10000 randomly generated labeling instances with n and m at most 100. All solvable instances were solved by performing at most one backtracking step per branch. This gives rise to the following conjecture:

Conjecture 4.1 *We conjecture that each instance of the label problem (G, R, C, n, m) is solved by Algorithm 1 with backtracking depth one. The worst case runtime is then $\mathcal{O}(n^2m^2(n+m))$.*

In order to calculate the worst case run time according to this conjecture we study the worst case behavior of the algorithm with backtracking depth one. The algorithm behaves in the worst case when each label is positioned first in all places that cause a conflict, before it is positioned in a conflict free place. A row label can be positioned in at most m different places. Each time when a label is positioned the Procedure 2 DRAW_CONCLUSIONS is called, which needs at most $\mathcal{O}(nm(n+m))$ time. Thus, the time for positioning a row label is bounded

by $\mathcal{O}(nm^2(n+m))$ time. Since n rows have to be labeled the backtracking approach with backtracking depth one needs at most $\mathcal{O}(n^2m^2(n+m))$ time. □

5 Complexity Status

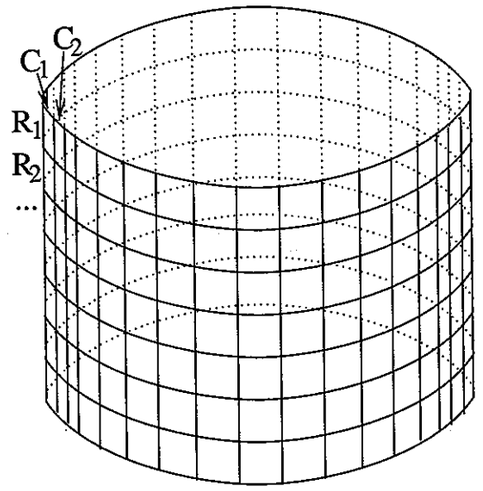

Figure 2: Cylinder label problem

Instead of labeling an array we now label a cylinder consisting of n cyclic rows and m columns. Figure 2 shows an example of an cylinder instance. We show that this problem is $\mathcal{NP}$-complete by reducing a version of the *Three Partition* problem to it. Our proof is similar to an $\mathcal{NP}$-completeness proof of Woeginger [Woe96] about the reconstruction of polyominoes from their orthogonal projections. Woeginger showed that the reconstruction of a two-dimensional pattern from its two orthogonal projections H and V is $\mathcal{NP}$-complete when the pattern has to be horizontally and vertically convex. This and other related problems, also discussed in [Woe96] show up in the area of discrete tomography.

Definition 5.1 (Cylinder Label problem (Z, R, C, n, m))

Instance: *Let $Z_{n,m}$ be a cylinder consisting of n cyclic rows and m columns. Let R_i be the label of the i^{th} row and let r_i be the length of label R_i, $1 \le i \le n$. Let C_i be the label of the i^{th} column and let c_i be the length of label C_i, $1 \le i \le m$.*

Problem: *For each row i set r_i consecutive fields of $Z_{i,\cdot}$ to r, for each column j set c_j consecutive fields of $Z_{\cdot,j}$ to c.*

Our reduction is done from the following version of the $\mathcal{NP}$-complete Three Partition problem [GJ79].

Problem 5.1 (Three Partition)

Instance: *Positive integers $a_1, \ldots a_{3k}$ that are encoded in unary and that fulfill the two conditions (i) $\sum_{i=1}^{3k} a_i = k(2B+1)$ for some integer B, and $(ii)(2B+1)/4 < a_i < (2B+1)/2$ for $1 \le i \le 3k$.*

Problem: *Does there exist a partition of $a_1, \ldots, a_{3k}$ into k triples such that the elements of every triple add up to exactly $2B+1$?*

Theorem 5.1 *The Cylinder Label problem is $\mathcal{NP}$-complete.*

Proof: Cylinder Problem $\in \mathcal{NP}$: The cylinder problem is in $\mathcal{NP}$ since it is easy to check whether a given solution solves the problem or not.
Transformation: Now let an instance of Tree Partition be given. From this instance we construct a cylinder label problem consisting of $n = k(2B+2)$ rows and $m = 3k$ columns. The vector r defining the row label length is of the form:

$$(m,\underbrace{m-1,\ldots,m-1}_{(2B+1)\text{-times}},m,\underbrace{m-1,\ldots,m-1}_{(2B+1)\text{-times}},\ldots)$$

Since a row label of length m occupies the whole row, those rows with label length m have no free space for column labeling. Therefore the rows with label length m subdivide the rows in k blocks, each containing $2B+1$ rows each of which has one entry that is free for column labeling when the row is labeled. The vector defining the column label length is of the form:

$$(a_1, a_2, \ldots, a_{3k})$$

The transformation clearly is polynomial.

The Tree Partition instance has a solution $\Leftrightarrow$ the Cylinder Label instance has a solution:

"$\Rightarrow$": Let $(x_1,y_1,z_1),\ldots,(x_k,y_k,z_k)$ be a partition of $a_1,\ldots,a_{3k}$ into k triples such that $x_i+y_i+z_i = 2B+1$, $1 \le i \le k$. For each i, $(x_i,y_i,z_i) = (a_f,a_g,a_h)$, for some indices f,g, and h, $1 \le i,f,g,h \le 3k$. We now label the columns f, g, and h among themselves in the i-th block of rows. More precisely, in column f we label the fields $Z_{f,(i-1)(2B+2)+2} = c,\ldots,Z_{f,(i-1)(2B+2)+1+c_{a_f}} = c$. In column g we label the fields $Z_{g,(i-1)(2B+2)+2+c_{a_f}} = c,\ldots,Z_{g,(i-1)(2B+2)+1+c_{a_f}+1+c_{a_g}} = c$. In column h we label the fields $Z_{h,(i-1)(2B+2)+2+c_{a_f}+c_{a_g}} = c,\ldots,Z_{h,(i-1)(2B+2)+1+c_{a_f}+c_{a_g}+c_{a_h}} = c$. It then follows that the rows $j(2B+2)+1$ are free for row labeling, for $0 \le j \le k$. Thus, we can label them with their labels of length $3k = m$. All other rows have exactly one entry occupied by a column label. Since the rows are cyclic we can label each of these rows with a label of length $3k-1$.
"$\Leftarrow$": Let Z be a solution of the Cylinder Label instance. Each row contains at most one entry that is occupied by a column label. Each column label a_i has length $(2B+1)/4 < a_i < (2B+1)/2$, $1 \le i \le 3k$. Therefore, exactly three columns are label in the rows $j(2B+2)+2,\ldots,(j+1)(2B+2)$, for $0 \le j \le k-1$. Furthermore the label length of each triple sums up to $3k$ and thus partitions $a_1,\ldots,a_{3k}$ into k triples. Thus solves the Three Partition instance. □

6 Solvable Special Cases

In the following section we derive an $\mathcal{O}(nm)$ time algorithm for the special case where no label is longer than half of its street length. We think that this case applies especially to large downtown maps, where the label length is short in respect to the street length. In Section 6.2 we solve the label problem when each vertical label is of equal length. In many american cities the streets in one orientation (e.g. north-south) are simply called *1-st Avenue, 2-nd Avenue, ...* These names have all the same label length and thus the label problem can be solved with the algorithm in Section 6.2. In Section 6.3 we give an algorithm for quadratic maps, where each label has one of two label lengths. The algorithm of the last two cases have runtime $\mathcal{O}(nm(n+m))$.

6.1 Half Size

Let (G,R,C,n,m) be a label problem. In this section we study the case where each row label has length at most $\lfloor \frac{m}{2} \rfloor$ and each column label has length at most $\lfloor \frac{n}{2} \rfloor$. We show that the label problem is solvable in this case.

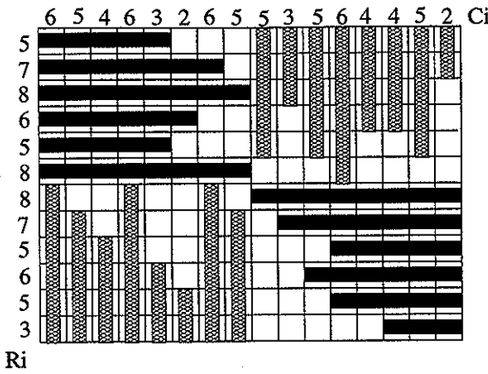

Figure 3: Solution of a typical half size label problem according to Algorithm 2.

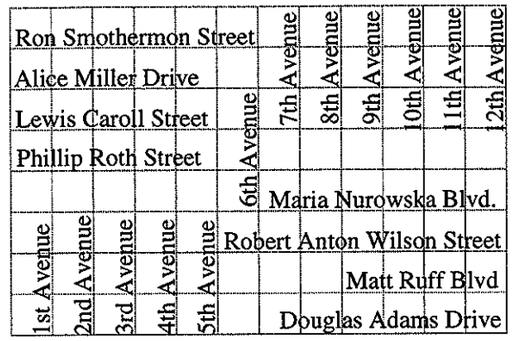

Figure 4: Typical downtown map where the vertical street names have constant length.

Algorithm 2 HALF_SOLUTION (G,R,C,n,m)

1) *label the rows* $1,\ldots,\lfloor \frac{n}{2} \rfloor$ *leftmost;*
2) *label the rows* $\lfloor \frac{n}{2} \rfloor + 1 \le n$ *rightmost;*
3) *label the columns* $1,\ldots,\lfloor \frac{m}{2} \rfloor$ *bottommost;*
4) *label the columns* $\lfloor \frac{m}{2} \rfloor + 1,\ldots,m$ *topmost;*

Theorem 6.1 *Let (G,R,C,n,m) be a label problem. Let $r_i \le \lfloor \frac{m}{2} \rfloor$ and let $c_i \le \lfloor \frac{n}{2} \rfloor$. Then, Algorithm 2 computes a solution to Problem 2.1 in $\mathcal{O}(nm)$ time.*

Proof: Take a look at Figure 3.

6.2 Constant Vertical Street Length

In this section we consider the special case of the label problem (G,R,C,n,m) where all column labels have length l. This problem we denote with (G,R,C,n,m,l). We show that we can decide whether the label problem (G,R,C,n,m,l) is solvable or not. We further give a simple algorithm that labels a solvable instance correctly. All results of this section are assignable for the constant row length case.

Theorem 6.2 (Constant Column Length) *Let (G,R,C,n,m,l) be a label problem with $c_i = l$, $1 \le i \le n$. The instance is solvable if and only if no conflict occurred in the Preprocessing 1.*

We assume that $l \le \lfloor \frac{m}{2} \rfloor$. Otherwise, row $\lceil \frac{n}{2} \rceil$ contains no fields that are free for row labeling. The next lemma states that the preoccupied fields are symmetric to the vertical central axis of G.

Lemma 6.1 *Let (G,R,C,n,m,l) be a successfully preprocessed label problem. After the preprocessing each row has the form [aba], where*

$$G_{i,1} = \emptyset,\ldots,G_{i,a} = \emptyset, G_{i,a+1} = x,\ldots,G_{i,a+b} = x, G_{i,a+b+1} = \emptyset,\ldots,G_{i,m} = \emptyset$$

for $x = r$ or $x = c$, $m \ge b \ge 0$ and $2a+b = m$.

Proof: Initially we have $G_{i,j} = \emptyset$ for $1 \leq i \leq n$, $1 \leq j \leq m$. Executing Rule 3.2 on each row i with length $r_i > \frac{m}{2}$ yields $G_{i,m-r_i+1} = r, \ldots, G_{i,r_i} = r$. Thus, all r-entries of G are symmetric to the vertical mid axis of G. Remember that each column has label length l. Therefore, executing Rule 3.2 on each column i yields that for each entry $G_{i,j}$ that is set to c the fields $G_{i,j+1} = c, \ldots, G_{i,m-j+1} = c$, if $1 \leq \frac{m}{2}$; and $G_{i,m-j+1} = c, \ldots, G_{i,j-1} = c$, if $\frac{m}{2} \leq i \leq m$. Therefore, all c-entries of G are symmetric to the central vertical axis of G. Thus, until now each row i has the form $[aba]$, where $G_{i,1} = \emptyset, \ldots, G_{i,a} = \emptyset, G_{i,a+1} = x, \ldots, G_{i,a+b} = x, G_{i,a+b+1} = \emptyset, \ldots, G_{i,m} = \emptyset$ for $x, y \in \{r, c\}$, $b \geq 0$, $2a + b = m$ and $1 \leq i \leq n$. Assume that the repeated execution of Rule 3.2 on row i of form $[aba]$ and $x = c$ does change an entry of $G_{i,.}$. In this case $a < r_i$ and it follows that the instance is not solvable. Therefore, the repeated execution of Rule 3.2 on a column can not change the instance and the lemma follows. □

Algorithm 3 CONSTANT COLUMN LENGTH (G, R, C, n, m, l)

1) **if** PREPROCESSING(G, R, C, n, m) {
2) *label the columns* $1, \ldots, \lceil \frac{m}{2} \rceil - 1$ *bottommost;*
3) *label the columns* $\lceil \frac{m}{2} \rceil, \ldots, m$ *topmost;*
4) *label the rows* $1, \ldots, n$ *in the free space;*
5) }

Lemma 6.2 *Let (G, R, C, n, m, l) be a successfully preprocessed label problem. Then Algorithm 3 computes a feasible solution to (G, R, C, n, m, l) in $\mathcal{O}(nm(n+m))$ time.*

Proof: Since (G, R, C, n, m, l) is preprocessed successively it follows that each column contains an interval of length at least l that is free for column labeling. Assume that after processing steps 2-3 there exists a row i not containing an interval of length r_i that is free for row labeling. We make a case distinction according to the length of R_i:

Case $r_i > \lceil \frac{m}{2} \rceil$**:** We know that the fields $G_{i,m-r_i+1} = r, \ldots, G_{i,r_i} = r$ were set in the preprocessing. Furthermore, Lemma 6.1 yields that no other entry of $G_{i,.}$ was set to c in the preprocessing. Therefore, each column $G_{.,j}$ with $1 \leq j < m - r_i + 1$ or $r_i + 1 \leq j \leq m$ contains either one interval of length at least $2l$ that is free for column labeling or two intervals each of length at least l that is free for column labeling. From the symmetry of the label problem and since the column labels of the columns j with $1 \leq j < m - r_i + 1$ are labeled bottom most and the labels j with $r_i + 1 \leq j \leq m$ are labeled top most it follows that either the fields $G_{i,1}, \ldots, G_{i,m-r_i+1}$ are free for row labeling or the fields $G_{i,r_i+1}, \ldots, G_{i,m}$. Thus, $G_{i,.}$ contains an interval of length r_i that is free for row labeling. Contradiction.

Case $r_i \leq \lceil \frac{m}{2} \rceil$**:** Lemma 6.1 yields that this row has the form $[aba]$ with $G_{i,1} = \emptyset, \ldots, G_{i,a} = \emptyset, G_{i,a+1} = c, \ldots, G_{i,a+b} = c, G_{i,a+b+1} = \emptyset, \ldots, G_{i,m} = \emptyset$, $b \geq 0$ and $2a + b = m$. Since the instance is solvable it follows that $a \geq r_i$. With the same arguments as above it follows that either the fields $G_{i,1}, \ldots, G_{i,r_i}$ are free for row labeling or the fields $G_{i,m-r_i+1}, \ldots, G_{i,m}$ are free for row labeling. Contradiction.

The running time is dominated by the preprocessing and thus $\mathcal{O}(n^3)$. □

See Figure 5– 8 for an example. Figure 4 shows a typical downtown city map in which all vertical streets have the same length.

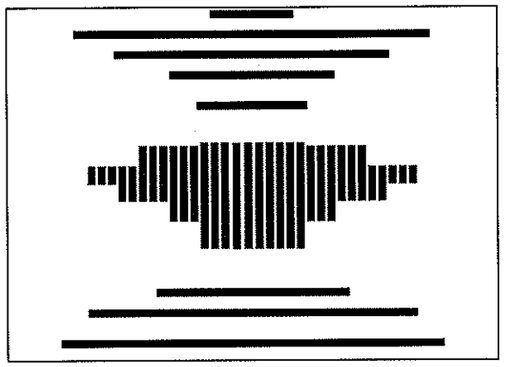

Figure 5: Matrix of a label problem with constant column length after the successful preprocessing. Entries that are set in the preprocessing are colored black and gray.

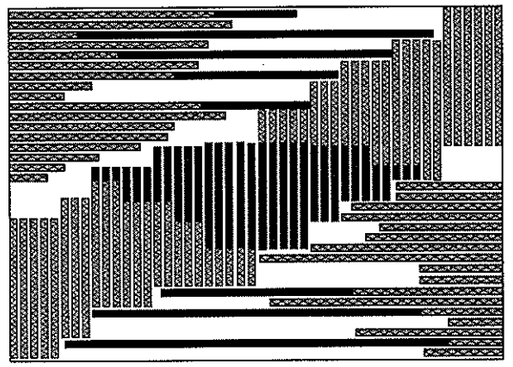

Figure 6: Solution of the label problem of the left figure.

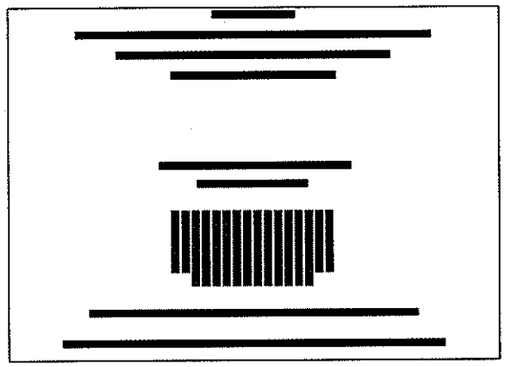

Figure 7: Matrix of a constant column length problem after the successful preprocessing. Entries that are set in the preprocessing are colored black and gray.

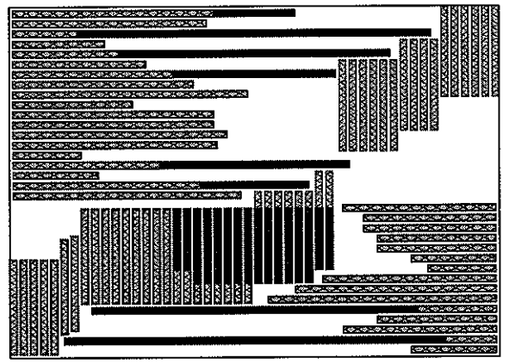

Figure 8: Solution of the label problem of the left figure.

6.3 Two Different Label Lengths

Let (G, R, C, n, n) be a label problem. In this section we study the case when any row label and any column label has length l_1 or length l_2. This special case we denote with $(G, R, C, n, n, l_1, l_2)$. We give an algorithm which solves this problem.

Theorem 6.3 (Two different label lengths) *Let $(G, R, C, n, n, l_1, l_2)$ be a quadratic label problem such that $r_i \in \{l_1, l_2\}$ and $c_i \in \{l_1, l_2\}$. The instance is solvable if and only if no conflict occurred in the Preprocessing Procedure 1.*

W.l.o.g let $l_1 \geq l_2$. In case that $l_1 \leq \lfloor \frac{n}{2} \rfloor$ Theorem 6.1 yields that the instance is solvable. In case that $l_1 > \lfloor \frac{n}{2} \rfloor$ and $l_2 > \lfloor \frac{n}{2} \rfloor$ it is easy to see that the instance is not solvable. Therefore, throughout this section and the next two subsections we assume that $l_1 > \lfloor \frac{n}{2} \rfloor$ and $l_2 < \lfloor \frac{n}{2} \rfloor$.

6.3.1 The Case $l_1 + l_2 > n$

Similarly to the half size solution we are going to label these instances in a windmill like manner. The critical rows and columns are the central rows and columns $n - l_1 + 1, \ldots, l_1$. From the regularity of the label length we can make some useful observations about the existence of long labels (of length l_1) in the central rows and columns.

Observation 6.1 *Let $(G, R, C, n, n, l_1, l_2)$ be a solvable label problem. At most one of the label sets $\{R_{n-l_1+1}, \ldots, R_{l_1}\}$ and $\{C_{n-l_1+1}, \ldots, C_{l_1}\}$ has elements of length l_1.*

The label problem has the nice property that intervals that are free for labeling in the central rows and columns are also free for labeling in the outer rows and columns.

Observation 6.2 *Let $[a, b[$ be an interval of row $G_{i,\cdot}$ that is free for row labeling, $n - l_1 < i \leq l_1$. Then, $[a, b[$ is free for row labeling in all rows.*

Proof: Assume there exists a row i' and a column $j \in [a, b[$ such that $G_{i',j} = c$. In case that $c_j = l_1$ it follows that $G_{i,j} = c$ for $n - l_1 < i \leq l_1$. Contradiction. In case that $c_j = l_2$ it follows that $G_{i',j} = c$ for $n - l_1 < j \leq l_1$. Therefore, row i' contains not enough free space for row labeling and we get a contradiction since the preprocessing would not be successful in this case. □

Observation 6.3 *Let $(G, R, C, n, n, l_1, l_2)$ be a label problem that was successfully preprocessed. There exist two disjoint intervals $[a_1, a_2[$, $[a_3, a_4[$ each of length at least l_2 such that $G_{i,a_1}, \ldots, A_{i,a_2-1}$ and $A_{i,a_3}, \ldots, A_{i,a_4-1}$ are free for row labeling for all rows $1 \leq i \leq n$.*

Proof: Assume there exists a row i with $r_i = l_2$ that does not contain two intervals of length at least l_2 that are free for row labeling (with Observation 6.2 we can assume that $n - l_1 < i \leq l_1$). From the preprocessing Rule 3.2 follows that R_i causes an occupied entry $G_{i,j} = r$ for some $1 < j \leq n$. Then, each row label R_k of length l_2 $(n - l_1 < k \leq l_1)$ causes an occupied row entry $G_{k,j} = r$ for j. Each row label R_k of length l_1 and $n - l_1 \leq k \leq l_1$ causes an occupied row entry $G_{k,j} = r$ for j as well since $l_1 > l_2$. From $c_j \geq l_2$ and since $n - l_1 < l_2$ and $n - l_1 < l_2$ it follows that there is not enough free space for column label C_j. Therefore, the preprocessing would have found a conflict which is a contradiction to our assumption. □

Algorithm 4 TWO DIFFERENT LABEL LENGTH, $l_1 + l_2 > n$ $(G, R, C, n, n, l_1, l_2)$

1) **if** PREPROCESSING (G, R, C, n, m) {
2) *let $[a_1, a_2[, [a_3, a_4[$ be two disjoint maximal intervals of G each of size at least l_2 that are free for row labeling in each row;*
3) *let $[b_1, b_2[, [b_3, b_4[$ be two disjoint maximal intervals of G each of size at least l_2 that are free for column labeling in each column;*
4) *label the rows $1, \ldots, b_3 - 1$ in the leftmost possible sufficiently large interval that is free for row labeling;*
5) *label the rows $b_3, \ldots, n$ in the rightmost possible sufficiently large interval that is free for row labeling;*
6) *label the columns $1, \ldots, a_3 - 1$ in the bottommost possible sufficiently large interval that is free for column labeling;*
7) *label the columns $a_3, \ldots, n$ in the topmost possible sufficiently large interval that is free for column labeling;*
8) }

These observations lead to Algorithm 4 that solves these kind of instances.

Lemma 6.3 *Let $(G, R, C, n, n, l_1, l_2)$ be a label problem with two different label lengths. If no conflict occurred in the Preprocessing Procedure 1 then Algorithm 4 computes a feasible solution for $(G, R, C, n, n, l_1, l_2)$ in $\mathcal{O}(n^3)$ time.*

Proof: The rows $i \in \{1, \ldots, b_3 - 1\}$ of label length l_2 are labeled inside $G_{i,1}, \ldots, G_{i,a_2-1}$, not conflicting with any preoccupied column entry from the preprocessing. The rows $i \in \{b_3, \ldots, n\}$ of label length l_2 are labeled inside $G_{i,a_3}, \ldots, G_{i,n}$, not conflicting with any preoccupied column entry from the preprocessing. The columns $j \in \{1, \ldots, a_3 - 1\}$ of label length l_2 are labeled inside $G_{b_3,j}, \ldots, G_{n,j}$, not conflicting with any preoccupied row entry from the preprocessing. The columns $j \in \{1, \ldots, a_3 - 1\}$ of label length l_2 are labeled inside $G_{1,j}, \ldots, G_{b_2-1,j}$, not conflicting with any preoccupied row entry from the preprocessing. Since all four submatrices are disjoint it follows that no conflict occurs between any two labels of length l_2. Assume that there exist two indices i, j, $1 \leq i, j \leq n$ such that $G_{i,j}$ is set to r and to c.

Case $1 \leq i < b_3, 1 \leq j < a_3$: From the observations made above follows that either a label of length l_2 overlaps with a label of length l_1 or two labels of length l_1 overlap.

Let two labels of length l_1 overlap. Since the preprocessing was successful and the j^{th} column is labeled bottommost it follows that all entries $G_{k,j} = c$ with $1 \leq k < b_3$ are preoccupied fields and therefore set in the preprocessing. These preoccupied fields have been considered in the positioning of label r_i. Contradiction. In case that $r_i = l_2$ and $c_j = l_1$ we come to a contradiction with the analog considerations. In case that $r_i = l_1$ and $c_j = l_2$ we know that column j is labeled below row $b_3 - 1$ and yield a contradiction.

Other Cases: All other cases work analogously, since the whole labeling is symmetric.

The preprocessing clearly dominates the running time. Thus the running time is $\mathcal{O}(n^3)$. □

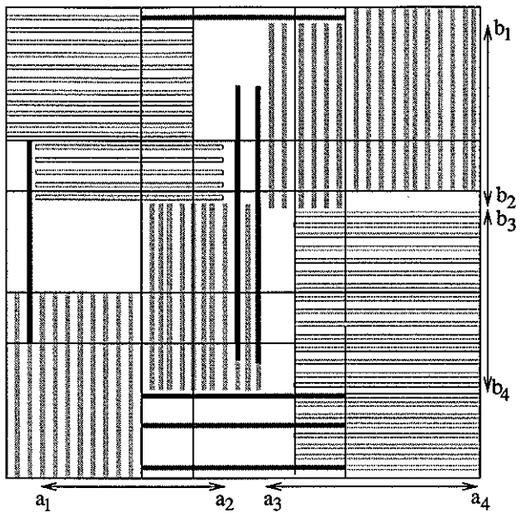

Figure 9: Solution of a problem with two different label length and $l_1 + l_2 > n$ after the execution of Algorithm 4. Entries that were occupied in the preprocessing are colored dark, all others are shaded.

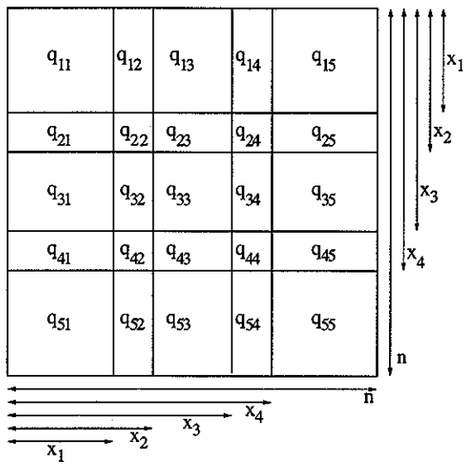

Figure 10: Subdivision of a matrix.

6.3.2 The Case $l_1 + l_2 \leq n$

We subdivide each column and each row into intervals. As interval points we define $x_1 = l_2$, $x_2 = n - l_1$, $x_3 = l_1$ and $x_4 = n - l_2$. Note that $x_1 \leq x_2 \leq x_3 \leq x_4$. These intervals subdivide the matrix G into several areas. We denote these areas with the indices $q_{11}, \ldots, q_{55}$ as defined in Figure 10.

Similarly to the case $l_1 + l_2 > n$ we can make some observations about the label length of solvable instances in the central rows and columns $x_2 + 1, \ldots, x_3$.

Observation 6.4 *Let $(G, R, C, n, n, l_1, l_2)$ be a solvable label problem. At most one of the label sets $\{R_{x_2+1}, \ldots, R_{x_3}\}$ and $\{C_{x_2+1}, \ldots, C_{x_3}\}$ has elements of length l_1.*

W.l.o.g we assume that $R_{x_2+1}, \ldots, R_{x_3}$ have length l_2.

Observation 6.5 *A preoccupied row entry $G_{i,j} = r$ for rows with label length l_1 can only occur if $i \in \{x_2 + 1, \ldots, x_3\}$ or $j \in \{x_2 + 1, \ldots, x_3\}$. For rows with label length l_2 it can only occur if $i \in \{x_2 + 1, \ldots, x_3\}$. Analogously, a preoccupied column entry $G_{i,j} = c$ for columns with label length l_1 can only occur if $i \in \{x_2 + 1, \ldots, x_3\}$ or $j \in \{x_2 + 1, \ldots, x_3\}$. For columns with label length l_2 it can only occur if $j \in \{x_2 + 1, \ldots, x_3\}$.*

Proof: Assume $G_{i,j} = r$ and $0 \leq i \leq x_2$ and $0 \leq j \leq x_2$. In case that $r_i = l_2$ we yield that this preoccupied entry must have been induced by at least one column entry $G_{i',j'} = c$ with $x_3 < i' \leq n$. In case that $r_i = l_1$ we also yield that this preoccupied entry must have been induced by at least one column entry $G_{i',j'} = c$ with $x_3 < i' \leq n$ since otherwise all preoccupied fields are contained in the interval $[x_2 + 1, x_3]$ of row i. Assume $G_{i,j} = r$ and $0 \leq i \leq x_2$, $x_2 < j \leq x_3$ and $r_i = l_2$. We again yield that this preoccupied entry must have been induced by at least one column entry $G_{i',j'} = c$ with $x_3 < i' \leq n$ and one with $1 \leq i' \leq x_2$. Since both possibilities are again caused by preoccupied fields where neither of

the two indices lies inside $[x_2+1, x_3]$ we come to a contradiction. All other cases work exactly analogous. □

Lemma 6.4 *We differ between two classes of instances.*

Case 1: *No label of length l_2 causes a preoccupied entry. Let $[a, b[$ be an interval of row $G_{i,\cdot}$ that is free for row labeling, $x_2 < i \leq x_3$. Then, $[a, b[$ is free for row labeling in all rows. Analogously, let $[c, d[$ be an interval of column $G_{\cdot,j}$ that is free for column labeling, $x_2 < j \leq x_3$. Then, $[c, d[$ is free for column labeling in all columns.*

Case 2: *There exist labels of length l_2 that cause preoccupied fields. Then, there exists an interval $[a, b[$ of length at least l_2 and $[a, b[\cap[x_2 + 1, x_3] = \emptyset$ such that either the fields $G_{i,a}, \ldots, G_{i,b-1}$ are free for row labeling for each $i \in \{x_2 + 1, x_3\}$ or the fields $G_{a,j}, \ldots, G_{b,j}$ are free for column labeling for each $i \in \{x_2 + 1, x_3\}$.*

Proof:

Case 1: Let $[a, b[$ as defined in the corollary, case 1. Assume there exists a row i' and a column $j \in [a, b[$ such that $G_{i',j} = c$. Since no column with a label of length l_2 causes a preoccupied entry we get that $c_j = l_1$. It follows that $G_{i,j} = c$ for $x_2 < i \leq x_3$. Contradiction.

Case 2: There exist labels of length l_2 that cause preoccupied fields. Consider the matrix G with all preoccupied fields of labels of length l_1. Observe that in case that a preoccupied column entry $G_{i,j} = c$, with $i \in [x_2 + 1, x_3]$ exists, then also $G_{i',j} = c$ for all $i \in [x_2 + 1, x_3]$. Furthermore, in case that a preoccupied row entry $G_{i,j} = r$, with $j \in [x_2+1, x_3]$ exists, then also $G_{i,j'} = r$ for all $j' \in [x_2+1, x_3]$. These preoccupied fields induce (a part of the) preoccupied fields of labels of length l_2. Concrete, let a column $j \in [x_2 + 1, x_3]$ with $c_j = l_2$ cause a preoccupied entry $G_{i,j} = c$, then also $G_{i,j'} = c$ for all columns $j' \in [x_2 + 1, x_3]$, which follows from the observation made above.
Case $i \in \{x_2 + 1, \ldots, x_3\}$: Since the preprocessing was successful there exists a row interval $[a, b[$ in row i of length at least l_2 that is free for row labeling and $[a, b[\subseteq [1, x_2]$ or $[a, b[\subseteq [x_3 + 1, n]$. Assume there exists a row $i' \in \{x_2 + 1, \ldots, x_3\}$ with $i' \neq i$ and a $j' \in \{a, \ldots, b - 1\}$ with $G_{i',j'} = c$. Then, $c_{j'} = l_2$ since otherwise also $G_{i,j'} = c$. Since $j' \notin \{x_2 + 1, \ldots, x_3\}$ we yield with Observation 6.5 a contradiction.
Case $i \notin \{x_2 + 1, \ldots, x_3\}$: In case that no $i \in \{x_2 + 1, \ldots, x_3\}$ exists such that $G_{i,j} = c$, $j \in \{x_2+1, \ldots, x_3\}$ and $c_j = l_2$, it follows there exists an interval $[a, b[$ of length at least l_2 and $[a, b[\cap[x_2 + 1, x_3] = \emptyset$ such that the fields $G_{a,j}, \ldots, G_{b,j}$ are free for row labeling for each $j \in \{x_2 + 1, \ldots, x_3\}$

All non listed cases are symmetric and follow analogously. □

Algorithm 5 TWO DIFFERENT LABEL LENGTH, $l_1 + l_2 \leq n\,(G, R, C, n, n, l_1, l_2)$

0) **if** PREPROCESSING(G, R, C, n, n) {
Part 1
1) $x_1 := l_2; x_2 := n - l_1; x_3 := l_1; x_4 := n - l_2;$
2) *label the rows* $1, \ldots, x_2$ *leftmost;*
3) *label the rows* $x_3 + 1, \ldots, n$ *rightmost;*
4) *label the columns* $1, \ldots, x_2$ *bottommost;*
5) *label the columns* $x_3 + 1, \ldots, n$ *topmost;*
Part 2
6) **forall** $i \in \{x_2 + 1, x_3\}$ *such that* $c_i = l_1$
7) *label an interval of* $B_{.,i}$ *of size* l_1 *that is free for column labeling with* C_i*;*
8) *let* $I := \{I_1 := [a_1, b_1[, \ldots, I_p := [a_p, b_p[\}$ *be all maximal intervals of* $G_{i,.}$ *of length at least* l_2 *that are free for row labeling, for all rows and* $x_2 < i \leq x_3$*;*
9) *let* $J := \{J_1 := [e_1, f_1[, \ldots, J_q := [e_q, f_q[\}$ *be all maximal intervals of* $B_{.,i}$ *of length at least* l_2 *that are free for column labeling, for all columns and* $x_2 < i \leq x_3$*;*
10) **if** $\{I_1 \backslash [x_2 + 1, x_3], \ldots, I_p \backslash [x_2 + 1, x_3]\}$ *contains an interval* $[a, b[$ *of length at least* l_2*{*
11) *label the rows* $x_2 + 1, \ldots, x_3$ *leftmost, right of column* $a - 1$*;*
12) **for each** $i \in \{x_2 + 1, \ldots, x_3\}$ *label column* i *in an arbitrarily chosen interval that is free for column labeling;*
13) }
14) **else** {
15) **if** $\{J_1 \backslash [x_2 + 1, x_3], \ldots, J_q \backslash [x_2 + 1, x_3]\}$ *contains an interval* $[e, f[$ *of length at least* l_2*{*
16) **for each** $i \in \{x_2 + 1, \ldots, x_3\}$ *with* $c_i = l_2$ *label columns* i *topmost, below row* $e - 1$*;*
17) }
18) **else** {
19) *let* $[a_1, a_2[, [a_3, a_4[$ *be two disjoint intervals from* $\{I_1, \ldots, I_p\}$ *each of length at least* l_2 *such that* $A_{i,.}$ *is free for row labeling,* $x_2 < i \leq x_3$*;*
20) *let* $[b_1, b_2[, [b_3, b_4[$ *be two disjoint intervals from* $\{J_1, \ldots, J_q\}$ *each of length at least* l_2 *such that* $B_{.,i}$ *is free for column labeling,* $x_2 < i \leq x_3$*;*
21) *label the rows* $x_2 + 1, \ldots, b_3 - 1$ *leftmost, right of column* $a_1 - 1$*;*
22) *label the rows* $b_3, \ldots, x_3$ *rightmost, left of column* a_4*;*
23) **for each** $i \in \{x_2 + 1, \ldots, b_3 - 1\}$ *with* $c_i = l_2$ *label column* i *bottommost, above row* b_4*;*
24) **for each** $i \in \{b_3, \ldots, x_3\}$ *with* $c_i = l_2$ *label column* i *topmost, below row* $b_1 - 1$*;*
25) } } }

Lemma 6.5 *Let* $(G, R, C, n, n, l_1, l_2)$ *be a label problem with two different label lengths and with* $l_1 + l_2 < n$*. If no conflict occurred in the Preprocessing Procedure 1 then Algorithm 5 computes a feasible solution for* $(G, R, C, n, n, l_1, l_2)$ *in* $\mathcal{O}(n^3)$ *time.*

Proof: Let $(G, R, C, n, n, l_1, l_2)$ be a solvable and preprocessed label problem. We first we show that no conflict occurs in part 1 of Algorithm 5. Note that in step 2 only fields in the fields $q_{11}, q_{12}, q_{13}, q_{21}, q_{22}$ and q_{23} of matrix G are set to r. In step 3 only fields in

fields $q_{43}, q_{44}, q_{45}, q_{53}, q_{54}$ and q_{55} of matrix G are set to r. In step 4 only fields in the fields $q_{31}, q_{32}, q_{41}, q_{42}, q_{51}$ and q_{52} of matrix G are set to c. In step 5 only fields in the fields $q_{14}, q_{15}, q_{24}.q_{25}, q_{34}$ and q_{35} of matrix G are set to c. Since no field is named twice and since no conflict occurred in the preprocessing no conflict occurred until now. Note, that the labels of length l_1 only occupy fields in the fields $q_{11}, q_{21}, q_{51}.q_{52}, q_{14}, q_{15}, q_{45}$ and q_{55}. Thus, in the first part of the algorithm we found a conflict free position for the labels $\{R_1, \ldots, R_{x_2}, R_{x_3+1}, \ldots, R_n, C_1, \ldots, C_{x_2}, C_{x_3+1}, \ldots, C_n\}$. Figure 11 shows an example.

Let $i \in [x_2 + 1, x_3]$ such that $r_i = l_2$. Let $I_1 = [a_1, b_1[, \ldots, I_p = [a_p, b_p[$ be all maximal intervals of $G_{i,.}$ of length at least l_2 that are free for row labeling, for all $i \in [x_2 + 1, x_3]$ with $r_i = l_2$. Let $J_1 = [e_1, f_1[, \ldots, I_q = [e_q, f_q[$ be all maximal intervals of $G_{.,j}$ of length at least l_2 that are free for column labeling, for all $j \in [x_2+1, x_3]$ with $c_j = l_2$. We assumed w.l.o.g. that $r_{x_2+1} = l_2, \ldots, r_{x_3} = l_2$. Since the preprocessing was successful, for each $i \in \{x_2 + 1, \ldots, x_3\}$ with $c_i = l_1$ there is an interval $[e, f[$ of $G_{.,i}$ that is free for column labeling. These columns are labeled in steps 6 and 7. All fields of $G_{i,j}$ with $i, j \in \{x_2 + 1, \ldots, x_3\}$ that are set to c in steps 6-7 were occupied in the preprocessing. Thus, none of the label placements further influences the amount of free space for row labeling in the rows $x_2 + 1, \ldots, x_3$.

Case $\{I_1 \backslash [x_2 + 1, x_3], \ldots, I_p \backslash [x_2 + 1, x_3]\}$ contains an interval $[a, b[$ of length at least l_2: This case is treated in steps 10-13. The labeling is trivially correct.

Case $\{J_1 \backslash [x_2 + 1, x_3], \ldots, J_q \backslash [x_2 + 1, x_3]\}$ contains an interval $[e, f[$ of length at least l_2: Analogously to the previous case steps 14-17 yield a valid labeling.

Case – all intervals of $\{I_1 \backslash [x_2 + 1, x_3], \ldots, I_p \backslash [x_2 + 1, x_3], J_1 \backslash [x_2 + 1, x_3], \ldots, J_q \backslash [x_2 + 1, x_3]\}$ have size smaller than l_2: With Lemma 6.4 we yield that labels of length l_2 do not produce preoccupied fields. Thus each row $i \in \{x_2 + 1, \ldots, x_3\}$ with $r_i = l_2$ contains at least two disjoint intervals of length at least l_2 that are free for row column labeling. Analogously, each column $j \in \{x_2+1, \ldots, x_3\}$ with $c_j = l_2$ contains at least two disjoint intervals of length at least l_2 that are free for column labeling. With Lemma 6.4 follows that $\{I_1, \ldots, I_p\}$ contains two disjoint intervals $[a_1, a_2[, [a_3, a_4[$ each of length at least l_2 such that $G_{i,a_1}, \ldots, G_{i,a_2-1}$ and $G_{i,a_3}, \ldots, G_{i,a_4-1}$ are free for row labeling, $x_2 < i \leq x_3$. Analogously, $\{J_1, \ldots, J_q\}$ contains two disjoint intervals $[b_1, b_2[, [b_3, b_4[$ each of length at least l_2 such that $G_{b_1,i}, \ldots, G_{b_2-1,i}$ and $G_{b_3,i}, \ldots, G_{b_4-1,i}$ are free for column labeling, $x_2 < i \leq x_3$. Assume that steps 18-24 do not yield a conflict free labeling. Assume that there exists two indices i, j, $1 \leq i \leq n, 1 \leq j \leq m$ such that $G_{i,j}$ was set to r and to c. We make a case distinction according to i and j.
Case $x_2 < i < b_3$ and $x_2 < j < a_3$: No conflict can occur since the columns $x_2 + 1, \ldots, a_3 - 1$ with length l_2 are labeled below $b_3 - 1$. The columns with label length l_1 are labeled right of $a_2 - 1$.
Case $x_2 < i < b_3$ and $a_3 \leq j \leq x_3$: No conflict can occur since the rows $x_2+1, \ldots, b_3-1$ are labeled left of the column a_3.
Case $b_3 \leq i \leq x_3$ and $x_2 < j < a_3$: No conflict can occur since the rows $b_3, \ldots, x_3$ are labeled right of the column $a_3 - 1$.
Case $b_3 \leq i \leq x_3$ and $a_3 \leq j \leq x_3$: No conflict can occur since the columns $x_2+1, \ldots, x_3$ are labeled above row b_3.

Thus, in all cases we found a conflict free labeling and the running time is again dominated by the preprocessing $\mathcal{O}(n^3)$. □

See Figure 11 and 12 for an illustration.

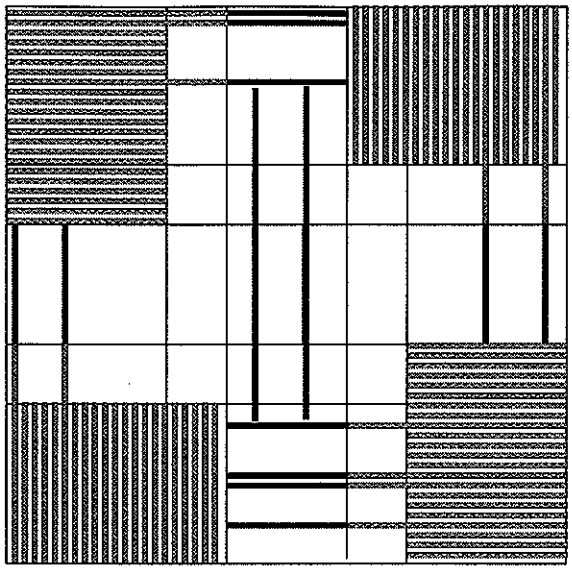

Figure 11: Matrix of a problem with two different label lengths and $l_1 + l_2 \leq n$ after the execution of Algorithm 4 part 1. Entries that were occupied in the preprocessing are colored black, all others are shaded.

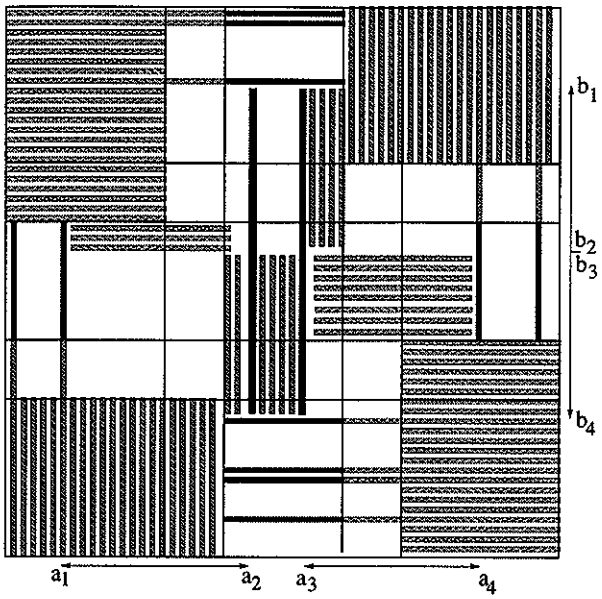

Figure 12: Solution of the label problem of the left figure.

References

[EIS76] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, December 1976.

[GG95] R.J. Gardner and P. Gritzmann. Descrete tomography: Determination of finite sets by x-rays. Technical Report TR-95-13, Institute of Computer Science, University of Trier, Austria, 1995.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractibility, A Guide to the Theory of* NP*-Completeness.* W. H. Freeman and Company, New York, 1979.

[NH] G. Neyer and H. Hennes. Map labeling with application to graph labeling. *GI Forschungsseminar:Zeichnen von Graphen, Teubner Verlag, to appear.*

[Woe96] G.J. Woeginger. The reconstruction of polyominoes from their orthogonal projections. Technical Report SFB-Report 65, TU Graz, Institut für Mathematik, A-8010 Graz, Austria, April 1996.