Ana Paula Tomás, Marta Andrade, Américo Pires da Costa

Technical Report Series: DCC-2001-02

Departamento de Ciência de Computadores – Faculdade de Ciências

&

Laboratório de Inteligência Artificial e Ciência de Computadores

Universidade do Porto

Rua do Campo Alegre, 823 4150 Porto, Portugal

Tel: +351+2+6078830 – Fax: +351+2+6003654

http://www.ncc.up.pt/fcup/DCC/Pubs/treports.html

Obtaining Origin-Destination Data at Optimal Cost at Urban Roundabouts
(Extended Version)

Ana Paula Tomás*
DCC-FC & LIACC, Univ. do Porto
R. do Campo Alegre, 823
4150-180 Porto, Portugal
apt@ncc.up.pt

Marta Andrade
Aenor / Operanor – Auto-Estradas do Norte, S.A.
mandrade@aenor.pt

Américo Pires da Costa
Departamento de Engenharia Civil
Fac. Engenharia, Univ. do Porto
amcosta@fe.up.pt

March 2001 – (rev. July 2001)

Abstract

We investigate the problem of finding the Origin-Destination trip matrix in a roundabout at optimal cost. The goal is to choose the turning movements that will be measured in addition to some independent total volumes at exits, entries and cross-sections inside the roundabout. Three cost criteria are proposed, each one modelling the cost of data collection in a different way. We discuss aspects of our study of this problem on computer and go through interesting mathematical properties we have found the proposed model to have.

1 Introduction

Vehicle flow data is an important source of information while providing a better knowledge about the traffic systems. Collecting traffic data is normally expensive and time consuming, so that the traffic surveys must be carefully designed.

As regards urban intersections, it is important to know not only the total traffic flows but also the turning movement flows, i.e., the Origin-Destination (OD) matrix for the junction. It is easy to count the total entry and exit volumes on each arm and, moreover, the recent advance in technology is rendering this task still easier. The problem arises with the turning movements, which have to be collected by direct observation, because it is not practically possible the use of automatic counters.

To perform direct observations at urban roundabouts is more difficult since not only they occupy a larger land area, but they also contain U-turning movements. So we must plan and design traffic data collection at roundabouts carefully. In particular, we have to find out how many observers we need, where they are to count the traffic volumes and which tasks are addressed to each one. Building on the computational study we present in this paper, a methodology for tackling this problem is given in [1], being considered three cost functions for doing a systematic analysis of roundabouts. The optimisation of the number and location of traffic counting points for the more general case of traffic networks is the subject of two recent publications [2, 8]. Traffic counts are there used in the estimation

*The work presented here has been partially supported by funds granted to LIACC through the Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia and Programa POSI.

of the OD trip matrix for the traffic network, their focus being on the quality of that estimation. By contrast to our work, in [2, 8] it is naturally assumed either some knowledge about the turning probabilities at link junctions or path flow information, which makes our problem essentially distinct from theirs.

The paper is organized as follows. In Section 2 we define the problem in mathematical terms. Then, in Section 3 we go through some aspects of the programs we implemented to study the problem on computer, to conclude in Section 4, by showing interesting mathematical properties we have found this problem to have.

2 The Mathematical Formulation

For a given urban intersection, let $\mathcal{O}$ be the set of entries (origins), $\mathcal{D}$ the set of exits (destinations) and q_{ij} the traffic flow from the entry i to the exit j. Our goal is to accurately find the values of all the q_{ij}'s for a given period of time. We assume that all the vehicles entering the roundabout also exit it, that is translated by (1), so that (2) and (3) hold,

$$\sum_{i\in\mathcal{O}} O_i = \sum_{j\in\mathcal{D}} D_j \tag{1}$$

$$\sum_{j\in\mathcal{D}} q_{ij} = O_i, \text{ for } i \in \mathcal{O} \tag{2}$$

$$\sum_{i\in\mathcal{O}} q_{ij} = D_j, \text{ for } j \in \mathcal{D} \tag{3}$$

with O_i and D_j denoting the total traffic volumes entering from i and exiting at j, respectively. By (1), at least one of the $|\mathcal{O}|+|\mathcal{D}|$ equations (2)–(3) is redundant. In fact, any $|\mathcal{O}|+|\mathcal{D}|-1$ of such equations are non-redundant, which follows quite intuitively in view of the real meaning of each O_i and D_j, but it is also one of the well-known properties of the Transportation Problems studied in Linear Programming (see e.g., [4]).

In this work, we consider that more expensive or sophisticated means may be required to get origin-destination data (i.e., the q_{ij}'s) than those needed to count vehicles at exits, entries and that pass through cross-sections of the circulatory carriage-way. The system (2)–(3), in the variables q_{ij}, is under-specified in general. At first sight, traffic counts at cross-sections of the circulatory carriage-way could be of some help in reducing this indeterminacy. Fig. 1 schematically represents two types of

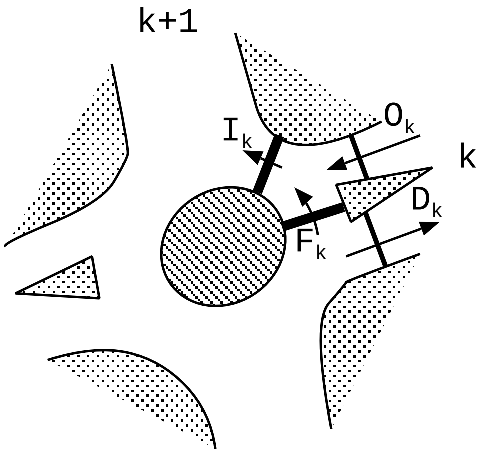


Figure 1: Cross-sections of the circulatory carriage-way

such cross-sections — F_k and I_k denote the traffic flow through the cross-section in frontal alignment with road k and through the cross-section between road k and $k+1$, respectively. Assuming that vehicles exit the roundabout when they reach their destination exit for the first time, we have (4) and (5),

$$\sum_{i\in\mathcal{O}\setminus\{k\}} \sum_{j\in\mathcal{D},\ k\prec j\preceq i} q_{ij} = F_k, \quad \text{for } 1\le k\le n \tag{4}$$

$$\sum_{j\in\mathcal{D}} q_{kj} + \sum_{i\in\mathcal{O}\setminus\{k\}} \sum_{j\in\mathcal{D},\ k\prec j\preceq i} q_{ij} = I_k, \quad \text{for } k\in\mathcal{O} \tag{5}$$

$$F_k = I_k, \quad \text{for } k\in\mathcal{D}\setminus\mathcal{O}$$

where $j\in\mathcal{D},\ k\prec j\preceq i$ stands for "the exits between road k and road i, being k excluded". All of these total traffic volumes are naturally related, being

$$F_{k+1} = F_k + O_k - D_{k+1} \tag{6}$$

and thus (7) holds.

$$F_k = F_1 + \sum_{1\le i<k} O_i - \sum_{1<i\le k} D_i, \quad \text{for all } k \tag{7}$$

Here, each O_i (respectively, D_j) should be read as 0, if $i\notin\mathcal{O}$ (respectively, $j\notin\mathcal{D}$). To simplify notation, by $k+1$ we refer to the road that immediately follows road k, in the ordering naturally induced by the way traffic circulates in the roundabout. We also have (8).

$$I_k = F_k + O_k \qquad \text{and} \qquad I_{k+1} = I_k + O_{k+1} - D_{k+1} \tag{8}$$

These relationships imply at that most $|\mathcal{O}|+|\mathcal{D}|$ of the equations defined by (2)–(5) are non-redundant. An exact characterization of this number is given by Proposition 3, in Subsection 4.3.

2.1 The Problem

Let r denote the rank of the system matrix defined by (2)–(5), that is, the number of non-redundant equations. We have just seen that either $r=|\mathcal{O}|+|\mathcal{D}|-1$ or $r=|\mathcal{O}|+|\mathcal{D}|$.

Under the hypothesis that the required traffic counts can be obtained with accuracy during a certain period of time, (2)–(5) is equivalent to each of its subsystems consisting of r non-redundant equations. Moreover, it admits a unique solution if $|\mathcal{O}||\mathcal{D}|-r$ of the q_{ij}'s are known and the columns of the system matrix associated to the r remaining ones are linearly independent. Hence, the main question is the choice of the traffic flows q_{ij} that will be locally measured, in addition to some r independent O_i's, D_j's, F_k's and I_k's.

In the following section we present the computational study we carried out, which led us to realize and prove relevant mathematical properties of this problem.

3 A Computational Study

The need for an exhaustive case analysis that could quickly provide either support or counterexamples to some preliminary conclusions, motivated the use of a computer to proceed the study. We wanted a computer program to generate all the possible combinations for the traffic volumes to be measured for each type of roundabout with n intersection roads. The efficiency of the algorithms was of great concern since the program would involve an highly combinatorial search procedure. Aiming at reducing the implementation effort, we briefly considered the use of a Constraint Programming system for solving the problem. Difficulties in modeling the problem in an suitable way led us to write the programs in C language, although implementing search strategies based on usual techniques in Combinatorial Optimization and Constraint Programming (e.g. [6] for some bibliographic references).

Because interesting programming problems were tackled, we think it worthwhile presenting some of the ideas of the designed algorithms. As we shall see, the analysis of the programs results gave us a deep insight on the problem structure, that allows to greatly improve some aspects of the solving procedure we now go on to describe.

3.1 Generating Distinct Roundabouts

A roundabout where n roads intersect is identified in the programs by a string $R_1R_2\ldots R_n$ for a given numeration of the intersecting roads. Each $R_i \in \{\mathrm{E}, \mathrm{D}, \mathrm{S}\}$ indicates whether road i is just an entry (E), just an exit (S) or both an entry and exit (D) road. All the strings obtained from $R_1R_2\ldots R_n$ by rotation denote exactly the same roundabout, but for different numerations of the roads, so that only one of these strings is used to refer to a given roundabout.

In the implementation, the symbols E, D and S are mapped to the digits 0, 1 and 2, so that the strings are viewed as representations of nonnegative integers in the basis 3. In particular, $R_1R_2\ldots R_n$ was mapped to $\sum_{i=1}^{n} 3^{i-1}R_i$. Among the strings that represent the same roundabout, the string that evaluates to the smallest integer is the one selected by the algorithm. To obtain the set of integers representing distinct roundabouts, the implemented algorithm maintains a boolean array `Id`, where `Id[i]` states whether or not `i` is in such set. Integers are considered from 1 to $3^n - 1$ and each time an integer `i` is found in the set, all the integers whose representation in basis 3 is a rotation of that of `i` are marked as non-eligible. In the end, the roundabouts correspond to the entries with a `KEEP` mark. The algorithm is presented in Fig. 2, where `base3(i,key,n)` yields the `n` digits representation of `i` in `key`, whereas `left_rotate(key,n)` rotates `key` to the left by one position and then returns the integer that `key` represents. The `visited` counter is correct since it can be shown that when a

```
int roundabouts(int n)
{ int nmax=pow(3,n)-2, nroundbs=0, i, k, visited=0;
  char key[NMAX];
  for (i=1; i<nmax; i++) Id[i] = KEEP;
  for (i=1; visited < nmax; i++)
    if (Id[i] == KEEP) {
      base3(i,key,n);  k = left_rotate(key,n);
      while(k != i) {
        Id[k] = DONT_KEEP; visited++;
        k = left_rotate(key,n);
      }
      visited++;  nroundbs++;
    }
  return nroundbs; // number of distinct roundabouts
}
```

Figure 2: Encoding and finding distinct roundabouts

string $\alpha \equiv \alpha_1\alpha_2\ldots\alpha_n$ is successively rotated to the left, α is the first sequence found repeated. The output strings $R_1R_2\ldots R_n$ denote distinct roundabouts and have the property that $R_1 \neq E$ (road 1 is an exit) and $R_n \neq S$ (road n is an entry), being the intersection roads numerated in the way traffic circulates. This casual feature was crucial during the analysis of the experimental results, as we shall see in section 4. Namely, it allowed to recognize the pattern of the optimal solution for one of the optimization criteria studied.

3.2 Finding the Directional Flows to Observe

For each type of roundabout with n intersection roads, the program obtains the constraints (2)–(5), computes the rank r of the corresponding system matrix and enumerates all the alternative solutions for the set of traffic volumes that are to be counted.

As we mentioned before, during this work, we assumed that more expensive or sophisticated means are required to get origin-destination data (i.e., the q_{ij}'s) than those needed to count vehicles at exits, entries and cross-sections (i.e., to get the total volumes O_i's, D_j's, F_k's and I_k's). This makes possible to separate the selection of the r independent volumes O_i's, D_j's, F_k's and I_k's that should be collected from that of the $|\mathcal{O}||\mathcal{D}| - r$ directional volumes q_{ij}'s. Solutions are then obtained by putting together any such couple of total and directional counts.

Although our work mainly concerns the choice of the q_{ij}'s, in the first implementation we also tackled the other subproblem. For the sake of efficiency, the program checked whether each selected combination of sections was equivalent by rotation to one previously found. Moreover, while deducing the constraints, it determined the F_k's and I_k's that have the same definition as another flow. That information was used in conjunction with a suitable tabulation of the combinations already studied to avoid exploring twice combinations that were actually equal.

The algorithm we implemented for choosing the directional flows is based on a depth-first search strategy with chronological backtracking, being the q_{ij}'s taken in a certain order. Whenever a solution is found or some failure is detected, backtracking occurs to the most recent alternative to find other solutions. Because the columns corresponding to each set of r directional flows that are not measured must be linearly independent, what the program actually enumerates are the alternatives for such a set of flows. In other words, the program is searching for *bases* of the subspace spanned by the columns of the system matrix.

To test the selected columns for linear independence, Gaussian elimination is applied to the matrix formed by them, to see whether it has full column rank. This method is quite adequate to handle the incremental change of the matrix in an efficient way. When a variable q_{ij} is selected, the program simply applies to its column the transformations done in the previous steps to the matrix corresponding to the variables already in the set, in exactly the same order. This is dealt by a function `incgauss(int ncs, int newcol)`, whose C code is given in Fig. 3, being `ncs` the number of columns

```
int incgauss(int ncs, int newcol)
{ int i, j, ipv, rowpiv;   double newp;
  for (i=0; i < Rank; i++) {
    AuxPvs[i] = i;
    Mat[i][ncs] = CoeffsRestr[i][newcol]); // insert newcol
  }
  if (ncs)
   for (j=0; j < ncs; j++) {
     if (AuxPvs[j] != PrevPvs[j].row) AuxPvs[PrevPvs[j].found] = AuxPvs[j];
     rowpiv =  PrevPvs[j].row;
     Mat[rowpiv][ncs] /= Mat[rowpiv][j];
     for (newp=0, i=j+1; i < Rank; i++) {
       Mat[AuxPvs[i]][ncs] -= Mat[AuxPvs[i]][j]*Mat[rowpiv][ncs];
       if (ABSVALUE(Mat[AuxPvs[i]][ncs]) > newp) {
         newp = ABSVALUE(Mat[AuxPvs[i]][ncs]); ipv = i; }
     }
     if (newp <= EPSILON) return PrevPvs[j].var; // EPSILON means 0
   }
  else { // the given column must be non-null
   for (newp=0, i=0; i < Rank; i++)
     if (ABSVALUE(Mat[i][ncs]) > newp) {newp = ABSVALUE(Mat[i][ncs]); ipv = i;}
  }
  PrevPvs[ncs].found = ipv;  PrevPvs[ncs].row = AuxPvs[ipv];
  PrevPvs[ncs].var = newcol;
  return newcol;
}
```

Figure 3: A Gaussian-elimination based test for linear independence.

already selected and `newcol` the index of the column of q_{ij}. The array `PrevPvs` contains relevant data about the pivots of the previous steps, which is needed to reduce the new column. Notice also that the auxiliary matrix `Mat` is only changed by this function and that, as usually, an array `AuxPvs` is used to keep track of row exchanges, rather than explicitly interchanging rows.

If the new column is found independent, the function returns `newcol`. Otherwise, it returns the index of the first variable that renders q_{ij} non-eligible. This allows the program to mark q_{ij} so as to prevent its selection until that incompatible variable is removed through backtracking. Hence, it detects and remembers conflicts that may arise near the root of the search tree. In addition, the program keeps the number of remaining variables, causing a failure when that number is less than the one needed to complete a basis. In that way, the search space is quite effectively reduced.

While fixing a bug due to roundoff errors, we have finally developed a rather simple method to test for independence, whose idea we present in Subsection 4.2.

3.2.1 Finding preferential solutions

Since the number of solutions is often quite large, three cost criteria were introduced to characterize preferential solutions, as proposed in [1]. It is important to observe that the conclusions drawn from the analysis of the output solutions are far more interesting than the straight application of these cost criteria in real practice.

The first criterion c_1 defines the cost of measuring the flow q_{ij} by the number of roads between i and j, which is expressed by (9).

$$c_1(q_{ij}) = \begin{cases} j-i, & \text{if } i<j \\ j-i+n, & \text{otherwise} \end{cases} \tag{9}$$

For both the remaining criteria c_2 and c_3, the costs are dynamically computed during the search and are determined by the previous choices. In both cases, it is assumed that origin-destination data must be collected at some of the entries and exits, such as by manually recording of plate numbers or by video surveys, being the total cost defined by the number of locations where such OD surveys are carried out. However, c_3 assigns a negligible cost to measuring q_{ij}, when the road j immediately follows i, assuming that such traffic flow can be fully observed in site. For each flow q_{ij} we now have

$$c_2(q_{ij}) = \begin{cases} 0, & \text{if } i \in M_{\mathcal{O}}, j \in M_{\mathcal{D}} \\ 2, & \text{if } i \notin M_{\mathcal{O}}, j \notin M_{\mathcal{D}} \\ 1, & \text{otherwise} \end{cases} \qquad c_3(q_{ij}) = \begin{cases} 0, & \text{if } i \in M_{\mathcal{O}}, j \in M_{\mathcal{D}}, \text{or } j=i+1 \\ 2, & \text{if } i \notin M_{\mathcal{O}}, j \notin M_{\mathcal{D}}, j \neq i+1 \\ 1, & \text{otherwise} \end{cases}$$

being $M_{\mathcal{O}}$ and $M_{\mathcal{D}}$ the set of entries and exits where that type of data has to be collected given the selections already in the set. In case q_{ij} is chosen to be measured and its cost value is not null, the sets $M_{\mathcal{O}}$ and $M_{\mathcal{D}}$ are updated by the program so as to ensure that surveys are located also at entry i and exit j, respectively.

The search procedure was adapted to implement a branch-and-bound strategy, as shown in Fig. 4 for the case of c_1. The main function `explore(int nv, int cvars, int remain)` is first called as `explore(0,0,NVars)`, where `NVars` is the number OD flows, whereas `cvars` and `nv` are the number of columns already selected to the basis and the index of next candidate in the `SortV` array, respectively. The variables are selected in a non-increasing order of cost, as given in `SortV`. The above mentioned conflicts are dealt by `MarkVars`, whose entries have `ELIGIBLE` value at start. Initially, `Vars[i].count` has value `MEASURED`, for all `i`.

By calling `deletemarks(nqij+1)`, the program restores as `ELIGIBLE` the variables that were detected in conflict with `nqij` and returns the overall cost of measuring them. Two complementary cost estimates are defined by `COST(nqij)` and `LOSS(nqij)`, being `CostParc` and `LossParc` used to prune the search space. `COST(nqij)` is $c_1(q_{ij})$ whereas `LOSS(nqij)` is the difference between the maximum value of c_1 and $c_1(q_{ij})$, that is $\max_{ij}(c_1(q_{ij})) - c_1(q_{ij})$.

When entering `explore()`, the global variable `CostParc` contains the sum of costs $c_1(q_{ij})$ for those q_{ij} that should be measured as settled by the previous selections, whereas `LossParc` contains the overall `LOSS` for those that are already in the basis. `MinCij` is a constant, that is defined by $\min_{ij}(c_1(q_{ij}))$, being 1 if c_1 is given by (9). Notice that, using `SortV`, the variables are selected to the basis in non-decreasing order of loss value. Finally, `Rank-cvars` is the number of variables that we need to complete the basis.

```
void explore(int nv, int cvars, int remain)
{ int i, j, costv=0, conflictvar, nqij;

  while (cvars < Rank && remain+cvars >= Rank) {
    if (CostParc + (remain-(Rank-cvars))*MinCij > CostOpt)  break;
    nqij = SortV[nv];  // index of selected variable
    if (LossParc + (Rank-cvars)*LOSS(nqij) > LossOpt) break;
    if (MarkVars[nqij] == ELIGIBLE) {
      if ((conflictvar = incgauss(cvars,nqij)) == nqij) {
        Vars[nqij].count = NOT_MEASURED;
        LossParc += LOSS(nqij);
        explore(nv+1,cvars+1,remain-1);
        CostParc -= deletemarks(nqij+1);
        Vars[nqij].count = MEASURED; remain--;
        costv += COST(nqij);
        LossParc -= LOSS(nqij);
      } else { MarkVars[nqij] = conflictvar+1; remain--; }
      CostParc += COST(nqij);
    }
    nv++;
  }
  if (cvars == Rank) { // adding the costs of the remaining variables
    for(j=nv,i=0; i<remain && CostParc < CostOpt; j++)
      if (MarkVars[SortV[j]] == ELIGIBLE) {
        CostParc += COST(SortV[j]);  costv += COST(SortV[j]);
        i++;
      }
    if (CostParc < CostOpt) { // found a better solution
      save_new_sol(); CostOpt = CostParc; LossOpt = LossParc;
    }
  }
  CostParc -= costv;
}
```

Figure 4: Searching for optimal solutions wrt c_1.

By modelling the cost by c_1, we wanted to abstract the idea that the longer movements get, the more difficult their direct observation is. Since our major motivation was that of finding possible interesting patterns for particular solutions, we decided to drastically increase the penalty for measuring longer movements by defining `COST(nqij)` and `LOSS(nqij)` as follows.

$$\begin{aligned}\texttt{COST(nqij)} &= n^{c_1(q_{ij})-1}\\ \texttt{LOSS(nqij)} &= n^{n-c_1(q_{ij})}\end{aligned}$$

This results in a significant runtime speedup, and quite surprisingly we observed that the program outputs exactly the same (unique) optimal solution as before, for each roundabout. The theoretical explanation for that fact is given in Subsection 4.4.

4 From Program Outputs to Mathematical Properties

Before we move on to present some interesting properties of this model, we recall classical results of Transportation Problems in Linear Programming (e.g., [4]), which were fundamental to find them.

4.1 On Transportation Problems in Linear Programming

The constraints of a well-balanced Transportation Problem in standard form, given by (10)–(11),

$$\sum_{j=1}^{n} x_{ij} = a_i,\ \ a_i > 0,\ \ i = 1,\ldots,m \tag{10}$$

$$\sum_{i=1}^{m} x_{ij} = b_j,\ \ b_j > 0,\ \ j = 1,\ldots,n \tag{11}$$

can be written in matrix form as $\mathbf{Px} = \mathbf{b}$, where $\mathbf{x} = [x_{11},\ldots,x_{1n},\ldots,x_{m1},\ldots,x_{mn}]$ and $\mathbf{b} = [a_1,\ldots,a_m,b_1,\ldots,b_n]$ are column vectors. When $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, the problem is called well-balanced, and in that case it has some non-negative solutions.

It is known that the rank of $\mathbf{P}$ is $m+n-1$, each $m+n-1$ of its rows being linearly independent. Thus, any $m+n-1$ of the constraints (10)–(11) are independent.

Each column of $\mathbf{P}$ contains exactly two 1's. The column $\mathbf{p}_{ij}$, that of the variable x_{ij}, can be written as $\mathbf{p}_{ij} = \mathbf{e}_i + \mathbf{e}_{m+j}$, where the $\mathbf{e}_k$ are the unit vectors of $\mathbb{R}^{m+n}$.

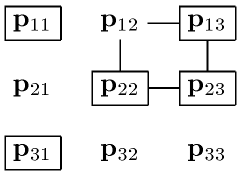
Given a subset $\mathcal{B}$ consisting of $m+n-1$ linearly independent columns of $\mathbf{P}$, so that $\mathcal{B}$ is a basis of the subspace spanned by the columns of $\mathbf{P}$, the linear combination of the basis vectors that gives $\mathbf{p}_{ij}$ is of the form (12)

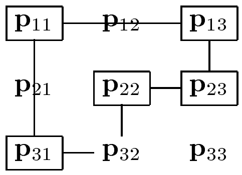
$$\mathbf{p}_{ij} = \mathbf{p}^{B}_{ij_1} - \mathbf{p}^{B}_{i_1j_1} + \mathbf{p}^{B}_{i_1j_2} - \cdots - \mathbf{p}^{B}_{i_kj_k} + \mathbf{p}^{B}_{i_kj} \tag{12}$$

where each vector in the basis occurs at most once and the number of terms is odd.

An interesting fact is that, when the symbols $\mathbf{p}_{ij}$ are written as in the following tables, each column $\mathbf{p}_{ij}$ not in the basis $\mathcal{B}$, and the basis vectors that appear in (12) form a loop, which is necessarily unique for each $\mathbf{p}_{ij}$. The loop, which is called *a simple loop*, consists of horizontal and vertical edges, consecutive edges being orthogonal and no vector being repeated. It can be shown that the columns in a given set are linearly independent if and only if no such a loop can be formed with some columns in the set, as explained in more detail below.

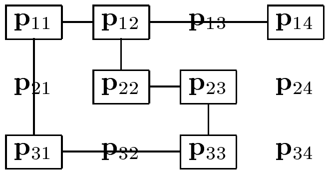
The following example illustrates these concepts. It shows how $\mathbf{p}_{12}$ and $\mathbf{p}_{32}$ are written as a linear combination of the basis vectors, when $\mathcal{B} = \{\mathbf{p}_{11}, \mathbf{p}_{13}, \mathbf{p}_{22}, \mathbf{p}_{23}, \mathbf{p}_{31}\}$ and $m = n = 3$.

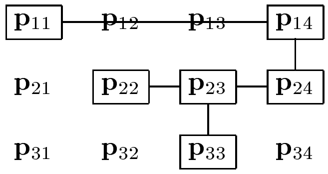




The conclusion is that $\mathbf{p}_{12} = \mathbf{p}_{13} - \mathbf{p}_{23} + \mathbf{p}_{22}$ and $\mathbf{p}_{32} = \mathbf{p}_{31} - \mathbf{p}_{11} + \mathbf{p}_{13} - \mathbf{p}_{23} + \mathbf{p}_{22}$. Notice that $\mathbf{e}_3 + \mathbf{e}_{3+2} = \mathbf{p}_{32} = (\mathbf{e}_3 + \mathbf{e}_{3+1}) - (\mathbf{e}_1 + \mathbf{e}_{3+1}) + (\mathbf{e}_1 + \mathbf{e}_{3+3}) - (\mathbf{e}_2 + \mathbf{e}_{3+3}) + (\mathbf{e}_2 + \mathbf{e}_{3+2})$. Each unit vector that is introduced in the combination, but $\mathbf{e}_3$ and $\mathbf{e}_{3+2}$, is explicitly removed.

Given a subset $\mathcal{B}$ of the columns of $\mathbf{P}$, let us construct a graph G, whose vertices correspond to the elements of $\mathcal{B}$, and the edges are obtained by linking each vertex in a given row (respectively, column) to the vertex in the same row (respectively, column) that is closer to that one in the table, if there is some, as shown in the following examples.





It is known that the columns in $\mathcal{B}$ are linearly independent if and only if the graph G is acyclic. If, in addition, the graph is connected and there is at least a vertex in each column and in each row of the table, then $\mathcal{B}$ is a basis of the subspace spanned by the columns of $\mathbf{P}$.

We shall now see how a simple test for linear independence is deduced by extending these results, in a natural way. This test only involves additions and subtractions.

4.2 Our Simple Test for Linear Independence

In the sequel, we consider that the roads are numerated in the way traffic circulates with $\mathcal{O} = \{\imath_1, \ldots, \imath_e\}$ and $\mathcal{D} = \{\jmath_1, \ldots, \jmath_s\}$, thus $e = |\mathcal{O}|$ and $s = |\mathcal{D}|$. Notice that we shall use both the notations $\imath_k$ (the kth entry) and i_k, being intentionally distinct, and also $\jmath_k$ (the kth exit) and j_k.

Given a roundabout whose constraint system has rank $|\mathcal{O}| + |\mathcal{D}|$, let us consider the subsystem consisting of the constraints (2)-(3) and the one defining F_1. As seen before, this subsystem is equivalent to (2)-(5). Now, let $\mathbf{P}'$ denote the matrix of this subsystem and let $\mathbf{P}$ be the sub-matrix consisting of all the rows of $\mathbf{P}'$ but the last one, which is that of F_1. It is not difficult to see that the columns of $\mathbf{P}'$ are given by (13), being $\mathbf{p}_{\imath_i \jmath_j} = \mathbf{e}_i + \mathbf{e}_{e+j}$.

$$\mathbf{p}'_{\imath_i \jmath_j} = \begin{cases} \mathbf{e}_i + \mathbf{e}_{e+j} + \mathbf{e}_{e+s+1} & \text{if } \imath_i \geq \jmath_j \neq 1 \\ \mathbf{e}_i + \mathbf{e}_{e+j} & \text{if } \imath_i < \jmath_j \text{ or } \imath_i = 1 \text{ or } \jmath_j = 1 \end{cases} \tag{13}$$

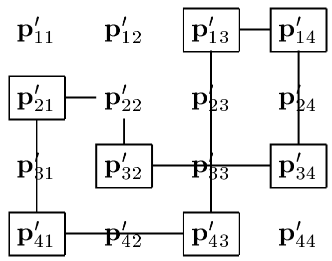
Let the last element of $\mathbf{p}'_{ij}$, that is the coefficient of q_{ij} in the equation defining F_1, be denoted by σ_{ij}. By (13), $\sigma_{ij} = 1$ if $i \geq j \neq 1$, and $\sigma_{ij} = 0$ otherwise. The test for linear independence we propose is based on Propositions 1 and 2, which follow almost directly from the results stated in section 4.1 and the definition of linear independence.

Proposition 1 *Let $\mathcal{B}'$ be a subset of the columns of $\mathbf{P}'$ and $\mathcal{B}$ the set of the corresponding columns in $\mathbf{P}$. If some column $\mathbf{p}_{ij}$ is written in an unique way as a combination of the columns in $\mathcal{B}$ by*

$$\mathbf{p}_{ij} = \mathbf{p}_{ij_1} - \mathbf{p}_{i_1 j_1} + \mathbf{p}_{i_1 j_2} - \cdots - \mathbf{p}_{i_k j_k} + \mathbf{p}_{i_k j}$$

then $\mathbf{p}'_{ij}$ is free relatively to $\mathcal{B}'$ if and only if $\sigma_{ij} \neq \sigma_{ij_1} - \sigma_{i_1 j_1} + \sigma_{i_1 j_2} - \cdots - \sigma_{i_k j_k} + \sigma_{i_k j}$.

By analogy to the Transportation problem, let us represent $\mathbf{P}'$ in a tableau form. The technique is illustrated by the following example, that is related to the roundabout identified by the sequence $DDDD$, the columns in framed boxes being those in $\mathcal{B}' = \{\mathbf{p}'_{13}, \mathbf{p}'_{14}, \mathbf{p}'_{21}, \mathbf{p}'_{32}, \mathbf{p}'_{34}, \mathbf{p}'_{41}, \mathbf{p}'_{43}\}$.

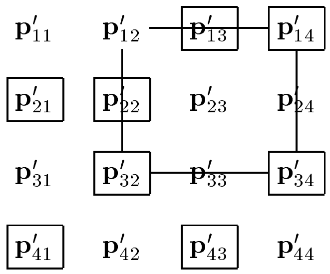


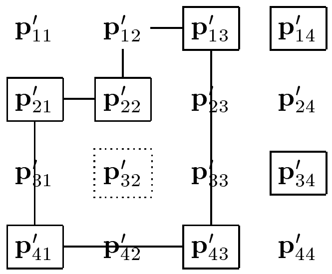
The loop shown in the tableau implies that $\mathbf{p}_{22} = \mathbf{p}_{21} - \mathbf{p}_{41} + \mathbf{p}_{43} - \mathbf{p}_{13} + \mathbf{p}_{14} - \mathbf{p}_{34} + \mathbf{p}_{32}$, there existing no other way of writing $\mathbf{p}_{22}$ as a combination of the columns in $\mathcal{B}$. But

$$\sigma_{22} = 1 \neq 0 - 0 + 1 - 0 + 0 - 0 + 1 = \sigma_{21} - \sigma_{41} + \sigma_{43} - \sigma_{13} + \sigma_{14} - \sigma_{34} + \sigma_{32}$$

so, we deduce that $\mathbf{p}'_{22} \neq \mathbf{p}'_{21} - \mathbf{p}'_{41} + \mathbf{p}'_{43} - \mathbf{p}'_{13} + \mathbf{p}'_{14} - \mathbf{p}'_{34} + \mathbf{p}'_{32}$, and thus $\mathcal{B}' \cup \{\mathbf{p}'_{22}\}$ is free.

Therefore, a free set is obtained if the vector $\mathbf{p}'_{22}$ is included in $\mathcal{B}'$, although the resulting set $\mathcal{B}$ is no longer free. For that reason, to conclude that, for instance $\mathbf{p}'_{12}$, is a linear combination of the columns in the new $\mathcal{B}'$, we have to adapt somehow the former procedure. As shown in the picture below, $\mathbf{p}_{12}$ is given by different combinations of the columns in the new $\mathcal{B}$.





And, although $\mathbf{p}_{12} = \mathbf{p}_{14} - \mathbf{p}_{34} + \mathbf{p}_{32}$, we have $\sigma_{12} = 0 \neq 0 - 0 + 1 = \sigma_{14} - \sigma_{34} + \sigma_{32}$, so that $\mathbf{p}'_{12} \neq \mathbf{p}'_{14} - \mathbf{p}'_{34} + \mathbf{p}'_{32}$. However, if one of the columns $\mathbf{p}_{14}$, $\mathbf{p}_{34}$ and $\mathbf{p}_{32}$ is removed, for instance $\mathbf{p}_{32}$, there remains a unique combination giving $\mathbf{p}_{12}$, namely $\mathbf{p}_{12} = \mathbf{p}_{13} - \mathbf{p}_{43} + \mathbf{p}_{41} - \mathbf{p}_{21} + \mathbf{p}_{22}$. As concerns F_1, we have $0 = 0 - 1 + 0 - 0 + 1$, and thus $\mathbf{p}'_{12} = \mathbf{p}'_{13} - \mathbf{p}'_{43} + \mathbf{p}'_{41} - \mathbf{p}'_{21} + \mathbf{p}'_{22}$, which shows the linear dependence of $\mathbf{p}'_{12}$ relatively to the new $\mathcal{B}'$. This is formalized by Proposition 2.

Proposition 2 *Let $\mathcal{B}'$ and $\mathcal{B}$ be defined as in Proposition 1. If $\mathcal{B}$ is not free, then $\mathcal{B}'$ is free if and only there exists some $\mathbf{p}_{ij}$ such that $\mathcal{B} \setminus \{\mathbf{p}_{ij}\}$ is free and the equality of the form*

$$\mathbf{p}_{ij} = \mathbf{p}_{ij_1} - \mathbf{p}_{i_1 j_1} + \mathbf{p}_{i_1 j_2} - \cdots - \mathbf{p}_{i_k j_k} + \mathbf{p}_{i_k j}$$

that expresses $\mathbf{p}_{ij}$ as a combination of the columns in $\mathcal{B} \setminus \{\mathbf{p}_{ij}\}$, does not hold for the corresponding columns in $\mathcal{B}'$, because $\sigma_{ij} \neq \sigma_{ij_1} - \sigma_{i_1 j_1} + \sigma_{i_1 j_2} - \cdots - \sigma_{i_k j_k} + \sigma_{i_k j}$.

Proof. Let $\mathbf{B}'$ and $\mathbf{B}$ be the sub-matrices consisting of the columns in $\mathcal{B}'$ and $\mathcal{B}$, respectively. Necessarily, $rank(\mathbf{B}') \leq rank(\mathbf{B}) + 1$ and $\mathcal{B}'$ is free if and only if $rank(\mathbf{B}') = |\mathcal{B}'|$. Thus, $\mathcal{B}'$ is free if and only if $rank(\mathbf{B}) \geq |\mathcal{B}| - 1$, since $|\mathcal{B}'| = |\mathcal{B}|$. By hypothesis, $\mathcal{B}$ is not free, and thus $rank(\mathbf{B}) = |\mathcal{B}| - 1$, which means that there exists some $\mathbf{p}_{ij}$ such that $\mathcal{B} \setminus \{\mathbf{p}_{ij}\}$ is free. From the results of section 4.1, we have that $\mathbf{p}_{ij}$ is given by a linear combination of the columns in $\mathcal{B} \setminus \{\mathbf{p}_{ij}\}$ of the form given above, which is unique. The conclusion now follows from Proposition 1. $\square$

4.3 The Rank of the System Matrix

The experimental results suggest the following characterization of the rank of the system matrix defined by (2)–(5), which renders its computation almost straightforward.

Proposition 3 *The rank r of the system matrix defined by equations (2)–(5) is $|\mathcal{O}|+|\mathcal{D}|$ if and only if, for all k, the equation defining F_k is not $F_k = 0$, being $|\mathcal{O}| + |\mathcal{D}| - 1$ otherwise. Furthermore, if $|\mathcal{O}| > 1$ and $|\mathcal{D}| > 1$ then $r = |\mathcal{O}| + |\mathcal{D}| - 1$ if and only if the string that identifies the roundabout is described by the regular expression $S^\star(SE + D)E^\star$.*

Proof. The rank r is $e + s$ (that is, $|\mathcal{O}| + |\mathcal{D}|$) if and only if the equation defining F_1 is non-redundant wrt (2)–(3). If some F_k is defined by $F_k = 0$, then from (7) we conclude that F_1 is a linear combination of the O_i's and D_j's, and therefore so are all the remaining F_k's. For the proof of the converse implication, we assume, with no loss of generality, that $\imath_e = n$ and $\jmath_1 = 1$ (i.e. road 1 is an exit and road n an entry). By case analysis, now we show that if $e > 1$, $s > 1$ and $\jmath_s > \imath_1$, the flow F_k is free from the O_i's and D_j's, for all k, which implies that $r = e + s$. Our claim is that $\{\mathbf{p}'_{\imath_1\jmath_1}\, \mathbf{p}'_{\imath_1\jmath_2}, \ldots \mathbf{p}'_{\imath_1\jmath_s}, \mathbf{p}'_{\imath_2\jmath_s}, \ldots, \mathbf{p}'_{\imath_e\jmath_s}, \mathbf{p}'_{\imath_e\jmath_1}\}$ is a basis of $\mathbf{P}'$, with $\mathbf{P}'$ and $\mathbf{P}$ defined as above. The four cases to study are: $\jmath_1 = \imath_1 < \jmath_s = \imath_e$, $\jmath_1 = \imath_1 < \jmath_s < \imath_e$, $\jmath_1 < \imath_1 < \jmath_s = \imath_e$ and $\jmath_1 < \imath_1 < \jmath_s < \imath_e$. Using the tableau, we see that $\mathbf{p}_{\imath_1\jmath_1}\, \mathbf{p}_{\imath_1\jmath_2}, \ldots \mathbf{p}_{\imath_1\jmath_s}, \mathbf{p}_{\imath_2\jmath_s}, \ldots, \mathbf{p}_{\imath_e\jmath_s}$ make a basis of $\mathbf{P}$, being $\mathbf{p}_{\imath_e\jmath_1} = \mathbf{p}_{\imath_1\jmath_1} - \mathbf{p}_{\imath_1\jmath_s} + \mathbf{p}_{\imath_e\jmath_s}$. In the four cases, $\sigma_{\imath_e\jmath_1} = 0 \neq 0 - 1 + 0 = \sigma_{\imath_1\jmath_1} - \sigma_{\imath_1\jmath_s} + \sigma_{\imath_e\jmath_s}$. Thus, F_1 is free from the O_i's and D_j's. It remains to prove that when either $e = 1$ or $s = 1$ or $\jmath_s \leq \imath_1$, we have $F_k = 0$ for some k, and therefore $r = e + s - 1$. In fact, when there is a single entry or exit, the flow through the cross-section in frontal alignment with that road is null. When $\jmath_s \leq \imath_1$, we have $F_{\imath_1} = 0$, because $\imath_1$ is the first entry and all the exits are between 1 and $\imath_1$. $\square$

4.4 An Unique Optimal Solution for the First Cost Criterion

The experimental results show the existence of an unique optimal solution for the cost criterion c_1, which has a quite well-defined pattern, as illustrated by the examples in Fig. 5. As before, the

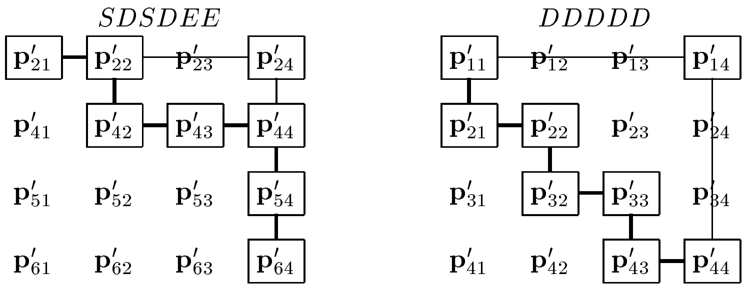


Figure 5: The pattern of the optimal solution for c_1.

columns in framed boxes are those in the basis. To obtain this pattern, $R_1R_2 \ldots R_n$ must satisfy $R_1 \neq E$ (road 1 is an exit) and $R_n \neq S$ (road n is an entry). When that is the case, the optimal solution wrt c_1 may be found using the following algorithm.

> **The algorithm:** Select for each entry i the flow q_{ij} to the exit that is the farthest from i. For each entry, select all the directional flows that precede the one already selected, but without passing the rightmost selected element in the previous row. Finally, select the rightmost element in the first row (northeast corner), if it has not been selected yet.

Fig. 6 exemplifies its application for the roundabout SDSDEE. The northeast corner is selected in the first step if and only if the roundabout is described by $S^\star(SE+D)E^\star$. It can be seen that the variables selected by the algorithm, in each row, are the ones that have the highest costs.

$$
\begin{array}{cccc}
\mathbf{p}'_{21} & \boxed{\mathbf{p}'_{22}} & \mathbf{p}'_{23} & \mathbf{p}'_{24} \\
\mathbf{p}'_{41} & \mathbf{p}'_{42} & \mathbf{p}'_{43} & \boxed{\mathbf{p}'_{44}} \\
\mathbf{p}'_{51} & \mathbf{p}'_{52} & \mathbf{p}'_{53} & \boxed{\mathbf{p}'_{54}} \\
\mathbf{p}'_{61} & \mathbf{p}'_{62} & \mathbf{p}'_{63} & \boxed{\mathbf{p}'_{64}}
\end{array}
\qquad
\begin{array}{cccc}
\boxed{\mathbf{p}'_{21}} & \boxed{\mathbf{p}'_{22}} & \mathbf{p}'_{23} & \mathbf{p}'_{24} \\
\mathbf{p}'_{41} & \boxed{\mathbf{p}'_{42}} & \boxed{\mathbf{p}'_{43}} & \boxed{\mathbf{p}'_{44}} \\
\mathbf{p}'_{51} & \mathbf{p}'_{52} & \mathbf{p}'_{53} & \boxed{\mathbf{p}'_{54}} \\
\mathbf{p}'_{61} & \mathbf{p}'_{62} & \mathbf{p}'_{63} & \boxed{\mathbf{p}'_{64}}
\end{array}
\qquad
\begin{array}{cccc}
\boxed{\mathbf{p}'_{21}} & \boxed{\mathbf{p}'_{22}} & \mathbf{p}'_{23} & \boxed{\mathbf{p}'_{24}} \\
\mathbf{p}'_{41} & \boxed{\mathbf{p}'_{42}} & \boxed{\mathbf{p}'_{43}} & \boxed{\mathbf{p}'_{44}} \\
\mathbf{p}'_{51} & \mathbf{p}'_{52} & \mathbf{p}'_{53} & \boxed{\mathbf{p}'_{54}} \\
\mathbf{p}'_{61} & \mathbf{p}'_{62} & \mathbf{p}'_{63} & \boxed{\mathbf{p}'_{64}}
\end{array}
$$

Figure 6: Applying the algorithm to the roundabout SDSDEE

A remarkable observation, to the proof of the correction of the algorithm, is that the rightmost element in the first row makes just the continuation of *the stairs* formed by the other chosen variables, as shown in Fig. 7. Based on this stairs shape of the output solution, it is possible to accurately define

$$
\begin{array}{cccccccc}
\boxed{\mathbf{p}'_{21}} & \boxed{\mathbf{p}'_{22}} & \mathbf{p}'_{23} & \mathbf{p}'_{24} & \mathbf{p}'_{21} & \mathbf{p}'_{22} & \mathbf{p}'_{23} & \mathbf{p}'_{24} \\
\mathbf{p}'_{41} & \boxed{\mathbf{p}'_{42}} & \boxed{\mathbf{p}'_{43}} & \boxed{\mathbf{p}'_{44}} & \mathbf{p}'_{41} & \mathbf{p}'_{42} & \mathbf{p}'_{43} & \mathbf{p}'_{44} \\
\mathbf{p}'_{51} & \mathbf{p}'_{52} & \mathbf{p}'_{53} & \boxed{\mathbf{p}'_{54}} & \mathbf{p}'_{51} & \mathbf{p}'_{52} & \mathbf{p}'_{53} & \mathbf{p}'_{54} \\
\mathbf{p}'_{61} & \mathbf{p}'_{62} & \mathbf{p}'_{63} & \boxed{\mathbf{p}'_{64}} & \mathbf{p}'_{61} & \mathbf{p}'_{62} & \mathbf{p}'_{63} & \mathbf{p}'_{64} \\
\mathbf{p}'_{21} & \mathbf{p}'_{22} & \mathbf{p}'_{23} & \boxed{\mathbf{p}'_{24}} & \boxed{\mathbf{p}'_{21}} & \boxed{\mathbf{p}'_{22}} & \mathbf{p}'_{23} & \mathbf{p}'_{24} \\
\mathbf{p}'_{41} & \mathbf{p}'_{42} & \mathbf{p}'_{43} & \mathbf{p}'_{44} & \mathbf{p}'_{41} & \boxed{\mathbf{p}'_{42}} & \boxed{\mathbf{p}'_{43}} & \boxed{\mathbf{p}'_{44}}
\end{array}
$$

Figure 7: Unfolding the optimal solution – a stairs-like shape

the way of writing any given $\mathbf{p}'_{ij}$ as a linear combination of the selected columns, as exemplified in Figures 8 and 9, for the larger hypothetical roundabout SDEESDSSEEED. The following practical

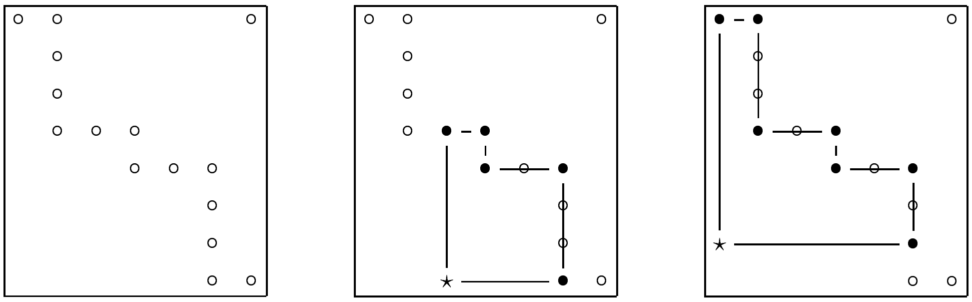


Figure 8: Writing $\mathbf{p}'_{ij}$ when $i > j$.

rule may be applied. First, we find $\mathbf{p}'_{ij}$ under the stairs, which requires their extension whenever $i < j$, as in the example of Fig. 9. Then, starting at $\mathbf{p}'_{ij}$, we move upwards until reaching a selected position, where we start jumping down the stairs to come back to the row of $\mathbf{p}'_{ij}$.

Given the variation of c_1, which in broad terms increases along each of the tableau rows and decreases along each column, it is possible to show that no improvement of the output solution can

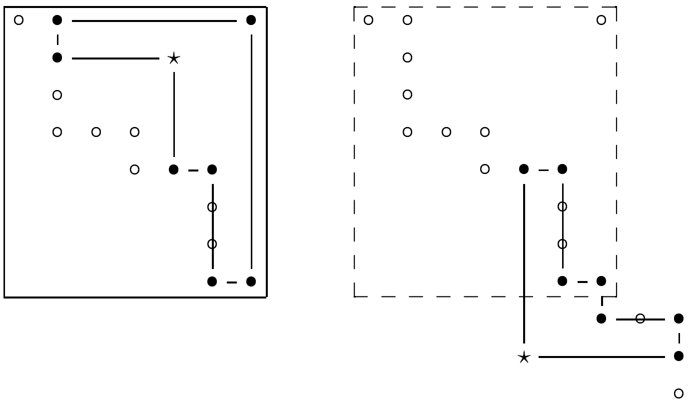


Figure 9: Writing $\mathbf{p}'_{ij}$ when $i < j$. On the right, the uniform view.

be achieved by changing one of the selected columns to a given $\mathbf{p}'_{ij}$. The variables that q_{ij} can replace are the ones associated to the columns in the combination that defines $\mathbf{p}'_{ij}$, whose costs exceed $c_1(q_{ij})$. Actually, for each exit j, we have $c_1(q_{i_\alpha j}) > c_1(q_{i_\beta j})$ if either $i_\alpha < i_\beta < j$ or $j \leq i_\alpha < i_\beta$, being $c_1(q_{i_\alpha j}) < c_1(q_{i_\beta j})$ otherwise. Similarly, for each entry i, if either $j_\alpha < j_\beta \leq i$ or $i < j_\alpha < j_\beta$, then $c_1(q_{ij_\alpha}) < c_1(q_{ij_\beta})$, being $c_1(q_{ij_\alpha}) > c_1(q_{ij_\beta})$, otherwise.

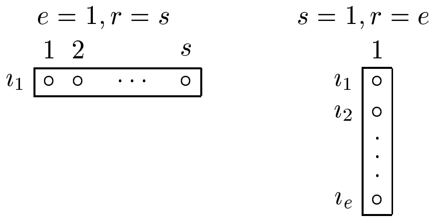
Notice that being each solution a basis of the subspace generated by the columns of $\mathbf{P}'$, it is sufficient to justify that no improvement of the output solution can be achieved by exchanging one of the columns in $\mathcal{B}'$ for any given $\mathbf{p}'_{ij} \notin \mathcal{B}'$.

A formal proof of the correction of the algorithm is presented below. Although the proof is a bit too long, it is simply the result of applying these ideas, while doing a formal case analysis.

Proposition 4 *The algorithm determines the unique optimal solution wrt the function c_1, provided the roundabout is identified by a string $R_1 \ldots R_n$ such that $R_1 \neq E$ and $R_n \neq S$.*

Proof. We use the same notations as in the proof of Proposition 3, where $e = |\mathcal{O}|$, $s = |\mathcal{D}|$, the numbers of the entry roads are $\imath_1, \ldots, \imath_e$ and $\jmath_1, \ldots, \jmath_s$ are those of the exits, with $\imath_e = n$ and $\jmath_1 = 1$.

As we noted in Subsection 4.3, when $e = s = 1$, the system has a unique solution, which is completely determined by the values of the O_i's and D_j's. It can easily be checked that in these cases, the algorithm correctly selects all the variables q_{ij}. The unique solution is necessarily optimal.



In the rest of the proof, we analyse the remaining situations, where $e > 1$ and $s > 1$. We are going to show that

1. the algorithm selects a basis $\mathcal{B}'$ of the subspace generated by the columns of $\mathbf{P}'$, which is actually the mathematical interpretation of the solution;

2. the computed solution is the unique optimal solution;

The columns $\mathbf{p}'_{ij}$ that are selected in Steps 1 and 2, are linearly independent, since the corresponding $\mathbf{p}_{ij}$ make a basis of the subspace generated by the columns of $\mathbf{P}$. This is clearly seen by constructing the graph defined in Subsection 4.1, which has a stairs-like shape. In addition, we conclude that the number of variables that are selected in these two steps is $e+s-1$. To see that $\mathcal{B}'$ is a basis, it remains to show that the northeast corner (i.e., the rightmost element in the first row) is selected in Step 3 if and only if $r = e + s$, being the corresponding column $\mathbf{p}_{\imath_1 \jmath_s}$ free from the ones previously chosen. Actually, from Proposition 3, we know that the rank $r = e + s - 1$ iff $\jmath_s \leq \imath_1$, which is equivalent to the last exit $\jmath_s$ being the farthest exit from the first entry $\imath_1$. This means that, the northeast corner is selected in Step 1 iff $\jmath_s \leq \imath_1$, and because it cannot be selected in Step 2, we deduce that it is selected in Step 3 iff $r = e + s$. Now, we use the techniques of Section 4.2 to conclude that, in this latter case, $\mathcal{B}'$ is a basis. As in Section 4.2, let $\mathcal{B}$ be the set of columns of $\mathbf{P}$ corresponding to the columns in $\mathcal{B}'$. There exists an unique combination of the columns in $\mathcal{B} \setminus \{\mathbf{p}_{\imath_1 \jmath_s}\}$ that gives $\mathbf{p}_{\imath_1 \jmath_s}$, which is of the form (14).

$$\mathbf{p}_{\imath_1 \jmath_s} = \mathbf{p}_{\imath_1 j_1} - \mathbf{p}_{i_1 j_1} + \mathbf{p}_{i_1 j_2} - \cdots + \mathbf{p}_{i_k \jmath_s} \tag{14}$$

We are going to see that (14) does not hold for the corresponding $\mathbf{p}'_{ij}$, and thus, $\mathcal{B}'$ is free, by Proposition 2. In fact, being $\imath_1 < \jmath_s$, the traffic flow from $\imath_1$ to $\jmath_s$ does not pass through the cross-section in front of road 1. This means that the coefficient $\sigma_{\imath_1 \jmath_s}$, of $q_{\imath_1 \jmath_s}$ in the equation defining F_1, is 0. As concerns the variables q_{ij}, that correspond to the columns $\mathbf{p}_{ij}$ in the right-hand side of (14), all such coefficients are 1, with the possible exception of those of $q_{\imath_1 j_1}$ and $q_{i_1 j_1}$, which are 0 iff $j_1 = 1$. That results from our hypothesis on the numeration of the roads and from the way in which the variables are selected in Steps 1 and 2, which ensure that $i \geq j$ for such $\mathbf{p}_{ij}$. Clearly, when $i \geq j \neq 1$, the traffic flow from i to j passes frontally to the road 1. Having the right-hand side of (14) an odd number of terms, which is at least 3, we have $\sigma_{\imath_1 \jmath_s} \neq \sigma_{\imath_1 j_1} - \sigma_{i_1 j_1} + \sigma_{i_1 j_2} - \cdots + \sigma_{i_k \jmath_s}$ since neither $0 = 1 = 1 - 1 + 1 - \cdots + 1$ nor $0 = 1 = 0 - 0 + 1 - \cdots + 1$, and consequently (14) does not hold for the $\mathbf{p}'_{ij}$'s.

In the rest of the proof we show that the output solution is the unique optimal solution. Being each solution a basis of the subspace generated by the columns of $\mathbf{P}'$, it is sufficient to justify that no improvement of the output solution can be achieved by exchanging one of the columns in $\mathcal{B}'$ for any given $\mathbf{p}'_{ab} \notin \mathcal{B}'$.

Our hypothesis on the numeration of the roads implies that either $\imath_1 = \jmath_1 = 1$ or $\imath_1 \neq 1 = \jmath_1$. Therefore, when $\imath_1 = 1$, the unique selections in the first row are $\mathbf{p}'_{11}$ and $\mathbf{p}'_{1\jmath_s}$, whereas in the first column we have $\mathbf{p}'_{11}$ and at least $\mathbf{p}'_{\imath_2 1}$. The relevant aspects of the possible cases are presented in Fig. 10, the circles denoting the selections.

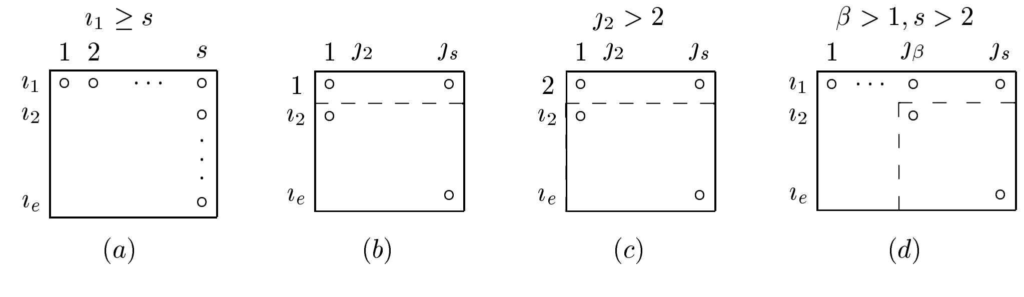


Figure 10: The cases when $e, s > 1$ (incomplete for the dashed part).

It can be shown that the unique combination of the columns in the basis $\mathcal{B}'$ that gives $\mathbf{p}'_{ab}$ is, as illustrated in Figures 8 and 9, of the form (15)

$$\mathbf{p}'_{ab} = \underbrace{\mathbf{p}'_{i_1 b}}_{\substack{\text{last in} \\ \text{col. } b}} - \underbrace{\mathbf{p}'_{i_1 j_1}}_{\substack{\text{last in} \\ \text{row } i_1}} + \underbrace{\mathbf{p}'_{i_2 j_1}}_{\substack{\text{last in} \\ \text{col. } j_1}} - \cdots + \underbrace{\mathbf{p}'_{a j_k}}_{\substack{\text{(last?) in} \\ \text{col. } j_k \text{ and} \\ \text{row } a}} \tag{15}$$

where *the last*, stands for the last one found when moving either downwards or to the right in the possibly augmented (c.f. Figures 7 and 9) tableau, and $\mathbf{p}'_{aj_k}$ is also the first element found in row a to the right of $\mathbf{p}'_{ab}$. The fact that the columns in the right-hand side of (15) do exist in $\mathcal{B}'$, follows as a consequence of the stairs-like shape of the output solution. From the results stated in Section 4.1, we may conclude that (15) holds wrt the columns of $\mathbf{P}$ involved. To conclude that the combination is correct, it remains to check (15) wrt the elements in the last row of $\mathbf{P}'$. In other words, to check that (16) holds.

$$\sigma_{ab} = \sigma_{i_1 b} - \sigma_{i_1 j_1} + \sigma_{i_2 j_1} - \cdots + \sigma_{a j_k} \tag{16}$$

By case analysis, we find that either $\sigma_{ab} = 0$ and there is a single sequence of 0's in the right-hand side of length one or three, or $\sigma_{ab} = 1$ and there is no 0 in the right-hand side. As a result, the combination is as defined by (15), since the number of terms in the rhs is odd. In view of the variation of the cost function c_1, it follows from (15) that the columns in $\mathcal{B}'$ that $\mathbf{p}'_{ab}$ can replace, have a strictly larger cost than $c_1(q_{ab})$, which implies that the output solution is the unique optimal solution.

We present now the above mentioned case analysis in detail, the possible cases being as sketched in Fig. 10. An important remark is that $\sigma_{\imath_1 \jmath_s} = 0$ whereas, for all $\mathbf{p}'_{ij} \in \mathcal{B}' \setminus \{\mathbf{p}_{\imath_1 \jmath_s}\}$, such that $i \neq 1$ and $j \neq 1$, we have $\sigma_{ij} = 1$, since necessarily $i \geq j$.

When $a > b$, the two possible forms of (16) are $0 = 0 - 1 + 1 - \cdots + 1$ and $1 = 1 - 1 + 1 - \cdots + 1$, for $b = 1$ and $b \neq 1$, respectively. In particular, in case (a), where $r = e + s - 1$, the combination is $\mathbf{p}_{ab} = \mathbf{p}_{\imath_1 b} - \mathbf{p}_{\imath_1 s} + \mathbf{p}_{as}$ and, being $a \geq \imath_2 > 1$, we have $\sigma_{ab} = \sigma_{\imath_1 b} - \sigma_{\imath_1 s} + \sigma_{as}$, which is $0 = 0 - 1 + 1$ when $b = 1$ and $1 = 1 - 1 + 1$, otherwise.

When $a < b$, the coefficient $\sigma_{ab} = 0$ and $\mathbf{p}'_{\imath_1 \jmath_s}$ occurs in the right-hand side of (15), since that element is the last selection in column $\jmath_s$ (for the augmented tableau). Moreover, if $a = \imath_1$ then $\mathbf{p}'_{\imath_1 \jmath_s}$ is the last term in the rhs of (15), being (16) of the form $0 = 1 - 1 + \cdots - 1 + 0$. When $a > \imath_1$, the rhs contains the sequence $\pm(\mathbf{p}'_{\imath_1 \jmath_s} - \mathbf{p}'_{\imath_1 \jmath_\beta} + \mathbf{p}'_{i \jmath_\beta})$, for some $i \geq \imath_2$, where $\jmath_\beta$ is the farthest exit from $\imath_1$. In cases (b) and (c), $\jmath_\beta = 1$, so that $\pm(\sigma_{\imath_1 \jmath_s} - \sigma_{\imath_1 \jmath_\beta} + \sigma'_{i \jmath_\beta}) = \pm(0 - 0 + 0)$, while in (d), it is $\pm(0 - 1 + 1)$, all the remaining coefficients in (16) are 1. □

4.4.1 The Greedy Character of the Optimal Solution

The unexpected existence of an exact characterization of the optimal solution, for which the selected variables in each row have higher cost than the remaining ones, led us to wonder whether the solution was a greedy one. We found that this problem is actually an instance of that of computing a maximum-weight independent subset in a linear matroid, which may be solved by the following Greedy Algorithm (see e.g. [3]).

> **The greedy algorithm.** For any given cost function c, an optimal basis $\mathcal{B}'$ is obtained if the columns of $\mathbf{P}'$ are selected in non-decreasing order of cost to form the basis. The column that is considered at each step is included in $\mathcal{B}'$, unless it is linearly dependent on those selected before.

This explains the fact that the program output precisely the same optimal solution when we assigned higher penalties for measuring longer movements (c.f. Subsection 3.2.1). This result is undoubtedly more important than the characterization of the optimal solution wrt criterion c_1. It describes a polynomial algorithm that returns such a solution whenever the cost value $c(q_{ij})$ is a constant known at start. Note that both the criteria c_2 and c_3 do not satisfy this condition on the cost function.

Nevertheless, the characterization of the optimal solution, not only has put into evidence its greedy character, but led us to develop a simple method for checking linear independence. As we shall see, this method allows a better understanding of the output solutions wrt c_2 and c_3.

4.5 The Optimal Solutions wrt Criteria c_2 and c_3

As we mentioned above, both criteria c_2 and c_3 aim at modelling the cost of conducting number plate surveys at some of the entries and exits in order to obtain OD data.

The total cost is given by the number of points where such data is collected. Thus, for a given basis $\mathcal{B}'$ to be optimal, the columns $\mathbf{p}'_{ij} \in \mathcal{B}'$ must fill a maximum number of rows and columns of the tableau, so that no recording must be done at the corresponding entries and exits. By Proposition 2, the graph associated to $\mathcal{B}$, as defined in Subsection 4.1, can have at most a cycle when $\mathcal{B}'$ is free, implying that it cannot have cycles as those illustrated in Fig. 11, where each $\circ$ denotes a $\mathbf{p}'_{ij} \in \mathcal{B}'$.

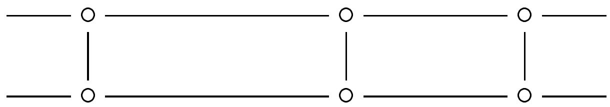


Figure 11: Example of linear dependency.

Hence, a necessary condition for $\mathcal{B}'$ to be a basis is that, when $e \geq 3$, the columns $\mathbf{p}'_{ij} \in \mathcal{B}'$ do not fill two or more columns of the tableau, neither do fill two or more rows when $s \geq 3$, whichever the basis $\mathcal{B}'$ is. In other words, when there are at least three entries and three exits, the optimal cost wrt c_2 is given by $(e-1)+(s-1)$, which translates the fact that one must record data at all the entries but one and at all the exits but one.

It can also be seen that when $e = 2$ and $s \geq 3$, that cost is $(e-1)+(s-2)$ if $r = e+s$, which means that we have to collect information in one exit less. Still, when $r = e+s-1$ the cost is given by $(e-1)+(s-1)$. Similarly, when $e \geq 3$ and $s = 2$, the optimal cost is $(e-2)+(s-1)$ if $r = e+s$, being $(e-1)+(s-1)$ otherwise. Optimal solutions are shown in Fig. 12. When either $e = 1$ or $s = 1$ or $e = s = 2$ and $r = 4$, the variables q_{ij} are fully determined by the O_i's and D_j's.

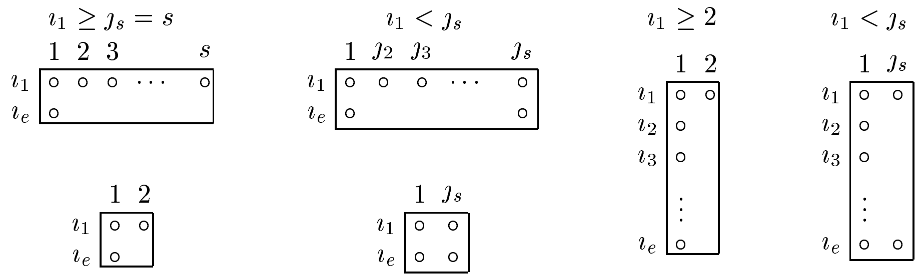


Figure 12: Examples of optimal solutions wrt c_2.

4.5.1 The Case of Criterion c_3

To the cost criterion c_3, we envisage a reduction in the number of locations where recording must be done, since we now suppose that the volumes $q_{i\,i+1}$ may be fully obtained by direct observation. By contrast to the previous criteria, it is almost impossible to abstract the form of the solutions from the program results. Nevertheless, by reasoning about the possible locations of the $\mathbf{p}'_{ij}$'s in the tableau to get bases, it is still possible to exactly characterize the optimal cost wrt c_3 for all the roundabouts, as shown in [7]. Optimal costs are there tabulated and examples of optimal solutions given for classes of roundabouts identified by *regular expressions* (see e.g. [5]). Distinguishing features are the numbers of entries, exits and flows $q_{i\,i+1}$, as well as the relative places of the $\mathbf{p}'_{i\,i+1}$'s in the tableau. For example, $(e-1)+(s-3)+3^\star$ is shown to be the optimal cost for the roundabouts described by $S^{k_1}(D+SE)S^{k_2}(D+SE)S^{k_3}(D+SE)$, with $k_1+k_2+k_3 \geq 2$, where $3^\star$ means that its three traffic flows $q_{i\,i+1}$ should be directly observed in site.

An interesting remark is that if, for any given roundabout $R_1R_2\ldots R_n$, we interchange E's with S's and read the resulting expression from right to left, we find a roundabout that is modelled by exactly the same system of equations, up to renaming and reordering of variables and constraints.

Hence, the optimal costs for both the cases $e \geq 5$ and $s \leq e \leq 4$ can be deduced from those obtained for $s \geq 5$ and $e \leq s \leq 4$, respectively, and reciprocally.

5 Conclusions

Results are given from our research on the problem of finding the OD trip matrix for roundabouts by performing a minimum number of traffic counts at minimum cost. The analysis has focused on the situations when counting vehicles at entries, exits and cross-sections inside the roundabout is seen as preferential. Of some practical interest is the conclusion that the number of non-redundant constraints in the given mathematical model is typically $|\mathcal{O}|+|\mathcal{D}|$, being $|\mathcal{O}|+|\mathcal{D}|-1$ when the traffic flow through one of such cross-sections is null. This means that there may exist a lot of possible hypotheses for the selection of the OD flows that we need to count so as to completely solve the problem. Nevertheless, this work led us to conclude that if a cost is given to measuring each OD flow individually, the overall cost is minimized if the OD flows that should not be measured are selected in non-decreasing order of cost to form an independent set. We gave a rather simple method for checking such independence, which has also the advantage of not involving floating-point operations. Three particular cost functions were proposed for a systematic study of hypothetical roundabouts on computer, and we have concluded that their optimal solutions are well-characterized.

Nonetheless, as claimed in [1], for real-world applications, it would be important to consider more flexible cost criteria in order to take into account specific features of the actual roundabout in study.

References

[1] Andrade M.: *Métodos e Técnicas de Recolha de Dados de Tráfego – Algoritmo para a Definição da Matriz Origem-Destino.* Msc. Thesis, Faculty of Engineering, University of Porto, 2000.

[2] Bianco L., Confessore G., and Reverberi P.: Optimal Location of Traffic Counting Points for Transport Network Control. In *Proceedings of IFAC'97.* Extended version to appear in *J. Transportation Science.*

[3] Cook W., Cunningham W., Pulleyblank W., and Schrijver A.: *Combinatorial Optimization.* John Wiley & Sons, 1998.

[4] Hadley G.: *Linear Programming.* Addison-Wesley, 1969.

[5] Hopcroft J. E., Ullman J. D.: *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[6] Marriott K., and Stuckey P.: *Programming with Constraints – An Introduction*, The MIT Press, 1998.

[7] Tomás, A. P.: A Note on Sensor Location for Traffic Counting at Roundabouts — Solutions for a Particular Cost Function. Internal Report DCC-2001-3, DCC-FC & LIACC, University of Porto, 2001.

[8] Yang H., and Zhou J.: Optimal Traffic Counting Locations for Origin-Destination Matrix Estimation. *J. Transportation Research*, 32B(2), 109–126, 1998.