Solving optimal Location of Traffic Count-Posts in CLP(FD)*

Ana Paula Tomás**

DCC-FC & LIACC, Universidade do Porto,
R. do Campo Alegre, 823, 4150-180 Porto, Portugal
apt@ncc.up.pt

Abstract. An application of Constraint Logic Programming (CLP) for finding the minimum number and location of count-posts at urban roundabouts to obtain origin-destination data at minimum cost is given. By finding interesting mathematical properties, we were able to model this problem as a constraint satisfaction problem in finite domains, and use CLP(FD) systems to solve it, with almost no implementation effort and quite fast. The model is biased to allow effective propagation of values and constraints. This application illustrates how an adequate usage of problem-specific knowledge, namely to deduce some global constraints, helps reduce the computational effort. In particular, we needed no advanced search techniques for finding optimal solutions. This may not hold for another novel application of CLP this paper points to, that of sensor location problems (SLP) for traffic networks, which we are investigating. It is also crucial now to find the right model to the problem, but it is not easy to state declaratively that, in the end, some variables should have their value fixed by constraint propagation. If we may overcome this problem, we think that the SLP for networks may be efficiently solved using Constraint Programming, since they have very good features for strong constraint propagation.

1 Motivation and Introduction

Traffic studies require origin-destination (OD) data whose acquisition is difficult and expensive. Important, but often insufficient, data is collected by counting the vehicles that pass at specific points of the traffic network, either manually or automatically. In the past two decades, there has been a considerable amount of research on the difficult problem of OD matrix estimation from (link) traffic counts, with economic issues as a major motivation. Some authors have examined various errors that may arise in the estimation, but with very limited discussion about how to identify the subset of network links where flow information should be collected and used [12]. Two recent publications [3, 12] address the optimization of the number and location of traffic counting points for traffic

* Extended Version of [15]

** This work was supported by funds granted to LIACC through *Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia* and *Programa POSI*.

networks. In [3], a model and heuristic based algorithms are given to solve the minimum-cost Sensor Location Problem (SLP). This problem is that of *finding the minimum number and location of count-posts in order to obtain the complete set of traffic flows on a transport network at minimum cost.*

Traffic studies for urban intersections are far less complex but still of prime importance in traffic engineering. In order to obtain OD data, it is typically necessary to carry out OD surveys, such as manually recording of registration numbers or roadside video surveys, in combination with some number plate tagging scheme. In this work, we present results from our research on the problem of *finding the minimal number of exits and entries where OD surveys must be done, when some user-defined subset of traffic counts can be obtained at a relatively negligible cost (e.g., by direct observation in site).* This can be viewed as a variant of the minimum-cost SLP. The model we propose is that of a constraint satisfaction problem in finite domains (CSP). No prior knowledge of turning movement coefficients is assumed or used in our work, which makes the problem clearly distinct from that in [3], so that we needed another approach.

This paper complements a previous work [13, 14], whose goal was to look for possible patterns for optimal cost solutions, that could eventually be used as practical rules by traffic engineers. The former involved a systematic case analysis of all hypothetical roundabouts with n legs (i.e., where n roads/streets intersect), for increasing values of n, in computer. *Hypothetical* since all strings $R_1 R_2 \dots R_n \in \{\mathrm{E}, \mathrm{D}, \mathrm{S}\}^\star$ were seen as possible roundabouts, with $R_i \in \{\mathrm{E}, \mathrm{D}, \mathrm{S}\}$ indicating whether road i is just an entry (E), just an exit (S) or both an entry and exit (D) road. In this way, we did not care whether some of these strings would ever represent real-world roundabouts. Notice that both E and S refer to one-way streets, whereas D means double-way. Computer programs were developed (in C) to enumerate minimal subsets of OD flows that, when counted and used in conjunction with total counts at entries and exits and cross-sections, allow to deduce the complete OD matrix (q_{ij}) for a given time period.

A major result of [13] is the conclusion that if the cost $c(q_{ij})$ of measuring q_{ij} is known, for each q_{ij}, 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. In fact, in this case the problem is an instance of that of computing a maximum-weight independent subset in a linear matroid which may always be solved by a Greedy Algorithm (see e.g. [4]). This means that, in such case, there exists a polynomial algorithm for computing the optimal solution. Actually, this independence is the linear independence of the columns of the constraint matrix that are related to the selected OD flows, and can be checked in polynomial time by Gaussian elimination, for instance. Another important contribution of [13] was the development of a rather simple method to test for linear independence, for this specific problem, which involves only exact operations. By further exploiting the mathematical properties underlying such method, it is shown in [14] the existence of an exact characterization of the optimal cost for the SLP at roundabouts when the OD flows $q_{i\,i+1}$'s are set a negligible measuring cost.

The work we now present was motivated by the claim [1] that the cost criteria should be more flexible to encompass specific features of the particular roundabout in study. By contrast to [13], the idea is no longer that of finding possible patterns of optimal solutions but rather to solve the minimum-cost SLP for every given real-world roundabout, the relevant data being input by the end-user of the application. The problem structure is further exploited, putting some effort on the formulation of a fairly clean mathematical model of the problem so that, in the end, CLP(FD) systems could be used to solve it. For space reasons, we shall not include experimental results, but just say that problems are solved in fractions of a second. The use of CP not only has led to a drastic reduction in the implementation effort but allows to easily extend the model to cater for other constraints that may be useful in practice.

Although this work seems at first sight too focused, its methodology is of wider interest. The proposed solution for encoding the non-trivial constraint (6) (see section 3) represents a compromise between *generate and test* and *constrain and generate* techniques. It is an example of the tradeoff between efficiency and declarativeness. Significant pruning is achieved by extracting global, although incomplete, information. Furthermore, our approach clearly shows the advantage of exploiting links to well-known problems when facing a new one.

In the following section, we give a formal description of the problem and some of its mathematical properties. Finally, the CSP model and aspects of its implementation in CLP(FD) systems are discussed in section 3.

2 The Problem and Some Background

After we have numbered the n intersecting streets in the way traffic circulates, the roundabout is perfectly identified by a string $R_1 R_2 \dots R_n \in \{\mathrm{E}, \mathrm{D}, \mathrm{S}\}^\star$, as defined above. Let $\mathcal{O} = \{\imath_1, \dots, \imath_e\}$ and $\mathcal{D} = \{\jmath_1, \dots, \jmath_s\}$ be the ordered sets of origins and destinations, $e = |\mathcal{O}|$ and $s = |\mathcal{D}|$. The traffic flow from the entry i to the exit j is denoted by q_{ij}, for $i \in \mathcal{O}$ and $j \in \mathcal{D}$. These flows are related to the total volumes at entries, exits and passing through the cross-sections of the circulatory roadway in frontal alignment with the intersecting roads (respectively, O_i, D_j and F_k) by (1)–(3).

$$\sum_{j \in \mathcal{D}} q_{ij} = O_i, \text{ for } i \in \mathcal{O} \tag{1}$$

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

$$\sum_{i \in \mathcal{O} \setminus \{k\}} \; \sum_{j \in \mathcal{D},\, k \prec j \preceq i} q_{ij} = F_k, \text{ for } 1 \leq k \leq n \tag{3}$$

In (3), $j \in \mathcal{D},\ k \prec j \preceq i$ stands for the exits between road k and road i, being k excluded. Vehicles are assumed to exit the roundabout and, in addition, loops are disallowed (i.e., vehicles do not pass their destination). Other cross-sections may be considered, as those standing strictly between two consecutive roads.

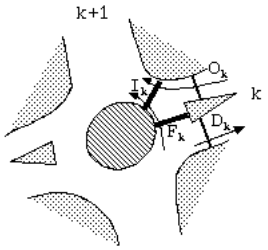


Fig. 1. Cross-sections of the circulatory roadway at a 4 legs roundabout

Clearly, the traffic volume I_k, that passes between roads k and $k+1$, is related to F_k and O_k by $I_k = F_k + O_k$, which means that if we already know O_k and F_k then counting I_k becomes redundant work. We assume that *counting the volumes O_i's, D_j's, F_k's and I_k's is of negligible cost when compared to other measuring tasks.* Thus, a maximal independent set of such counts should be used, but it does not matter which one. At most $|\mathcal{O}| + |\mathcal{S}|$ (that is, $e + s$) total counts are globally non-redundant, the exact number may be found immediately using Property 1 [13].

Property 1. The matrix of the system defined by (1)–(3), in the variables q_{ij}'s, has rank $e + s$ if and only if none of equations in (3) is of the form $F_k = 0$, being $e + s - 1$ otherwise. The rank is $e + s - 1$ if and only if the roundabout is not identifiable by a string $R_1 R_2 \dots R_n$ of the language described by the regular expression $S^\star(D + SE)E^\star$.

We consider *error-free traffic counts*, an usually accepted condition [3, 12]. So, the system that results from replacing such counts in (1)–(3) is consistent. Hence, by a basic property of systems of linear equations (see e.g., [7, 8, 10]), it is equivalent to each of its subsystems, that consists of equations (1)–(2) and one in (3). In the sequel, we consider that the last equation is that of F_1.

Notation. Let $\mathbf{P}'$ be the matrix of such subsystem and $\mathbf{p}'_{ij}$ the column of the variable q_{ij}. Let $\mathbf{P}$ be the matrix of the subsystem (1)–(2) and $\mathbf{p}_{ij}$ be the column of q_{ij}. If we represent the coefficient of q_{ij} in the equation of F_1 by σ_{ij}, it shall be clear that

$$\mathbf{p}'_{ij} = \begin{bmatrix} \mathbf{p}_{ij} \\ \sigma_{ij} \end{bmatrix}$$

This relation is a cornerstone in this work.

Every consistent system of linear equations may be written in solved form. For example, for the roundabout SSEDE[1], $\mathcal{O} = \{3, 4, 5\}$, $\mathcal{D} = \{1, 2, 4\}$ and

[1] SSEDE is interesting because the subsystem is not too large. If it were SESED, it would be a real case (in V. N. Gaia).

$$\begin{cases} q_{31}+q_{32}+q_{34}=O_3 \\ q_{41}+q_{42}+q_{44}=O_4 \\ q_{51}+q_{52}+q_{54}=O_5 \\ q_{31}+q_{41}+q_{51}=D_1 \\ q_{32}+q_{42}+q_{52}=D_2 \\ q_{34}+q_{44}+q_{54}=D_4 \\ q_{32}+q_{42}+q_{44}+q_{52}+q_{54}=F_1 \end{cases} \Leftrightarrow \begin{cases} q_{31}= \quad q_{42} \quad\quad +q_{52}+F_1-O_4+D_1-D_2 \\ q_{32}=-q_{42} \quad\quad -q_{52}+D_2 \\ q_{34}= \quad\quad\quad\quad\quad O_5-F_1+O_4-D_1+O_3 \\ q_{41}=-q_{42}-q_{51}-q_{52}+D_2+O_5-F_1+O_4 \\ q_{44}= \quad\quad q_{51}+q_{52}+F_1-D_2-O_5 \\ q_{54}= \quad\quad -q_{51}-q_{52}+O_5 \end{cases}$$

and, we immediately see that if the values of all the variables in the right-hand side are known (i.e., q_{42}, q_{51} and q_{52}, as well as O_3, O_4, O_5, D_1, D_2 and F_1), the system has a unique solution for the remaining q_{ij}'s.

By generic results about linear systems, it follows that at least $es - \mathit{rank}(\mathbf{P}')$ of the q_{ij}'s shall be known (i.e., counted). And, if $es - \mathit{rank}(\mathbf{P}')$ are known, the values of the remaining $\mathit{rank}(\mathbf{P}')$ variables may be determined, by solving the system with respect to them, if and only if their columns in $\mathbf{P}'$ are linearly independent (make a basis for $\mathbf{P}'$). In the example, $\mathcal{B}' = \{\mathbf{p}'_{31}, \mathbf{p}'_{32}, \mathbf{p}'_{34}, \mathbf{p}'_{41}, \mathbf{p}'_{44}, \mathbf{p}'_{54}\}$ is a basis (i.e., maximal independent set) of the linear subspace spanned by the columns of $\mathbf{P}'$. It is interesting to note that $\mathbf{p}'_{42} = \mathbf{p}'_{41} - \mathbf{p}'_{31} + \mathbf{p}'_{32}$, $\mathbf{p}'_{51} = \mathbf{p}'_{54} - \mathbf{p}'_{44} + \mathbf{p}'_{41}$ and $\mathbf{p}'_{52} = \mathbf{p}'_{54} - \mathbf{p}'_{44} + \mathbf{p}'_{41} - \mathbf{p}'_{31} + \mathbf{p}'_{32}$ and also that for $\mathbf{P}$, $\mathbf{p}_{42} = \mathbf{p}_{41} - \mathbf{p}_{31} + \mathbf{p}_{32}$, $\mathbf{p}_{51} = \mathbf{p}_{54} - \mathbf{p}_{44} + \mathbf{p}_{41}$ and $\mathbf{p}_{52} = \mathbf{p}_{54} - \mathbf{p}_{44} + \mathbf{p}_{41} - \mathbf{p}_{31} + \mathbf{p}_{32}$.

Our problem. To find a basis $\mathcal{B}'$ consisting of $\mathit{rank}(\mathbf{P}')$ independent columns of $\mathbf{P}'$, such that to measure the q_{ij}'s for $\mathbf{p}'_{ij} \notin \mathcal{B}'$, surveys are undertaken at a minimum number of exits and entries. Without further assumptions, the optimal cost is, in general, $(e-1)+(s-1)$, meaning that the registration shall take place at all the entries but one and at all the exits but one. So, our goal is to find whether this cost may be reduced by using much less expensive means (whatever that may mean) to count part of the q_{ij}'s.

3 Modeling the Minimum-Cost SLP as a CSP

Praça da República at Porto is a roundabout of type SEESDSE, for which a realistic scenario is as shown below, on the left, as claimed in [1].

$\mathcal{O}\backslash^{\mathcal{D}}$	1	4	5	6
2		⋆	⋆	
3		⋆	⋆	
5				
7	⋆			

	1	4	5	6
2	•	⋆	⋆	•
3	•	⋆	?	
5	•	•		
7	⋆	•	•	•

The positions marked with ⋆ correspond to OD flows that are cheap to obtain. In this particular case, by cheap we mean that they can be fully obtained by direct observation in site. An observer, standing at entry 2, can count q_{24} and q_{25}. The same applies to the pair q_{34} and q_{35}, and to q_{71}. By contrast, for instance, the geometry of the roundabout makes it too difficult to obtain q_{56} by direct

observation, although roads 5 and 6 are consecutive. The solution on the right is one of the eight optimal solutions found by our prototype implementations of the following model in SICStus Prolog [11, 2] and ECLiPSe [5]. It minimizes the number of locations where recording is done: at two entries (3 and 5) and two exits (5 and 6). The flows q_{ij}'s that would not be counted are marked with $\bullet$'s, implying that $\mathbf{p}'_{ij} \in \mathcal{B}'$. The ones that should be obtained by direct observation are marked with $\star$'s. The symbol ? indicates that there are two alternatives to get q_{35}, either by recording or by direct observation.

3.1 The Proposed Model

Given the string $R_1 R_2 \dots R_n$ that identifies the roundabout, let $\mathcal{O}$ and $\mathcal{D}$ be as defined above. Given the set Γ that characterizes the OD flows that may be obtained by some means that the user (likely an engineer) finds preferable to recording, say $\Gamma = \{(i,j) \mid q_{\imath_i \jmath_j} \text{ may be collected by direct observation}\}$, let $\Theta = \{i \mid (i,j) \in \Gamma \text{ for some } j\}$. We note that Θ is the set of rows that have $\star$'s. In the sequel, *sensor* refers to recording means (e.g., somebody registering number-plates or a video recording image) as opposed to *direct observation*.

The decision variables

$E_i \in \{0,1\}$, is 1 iff a sensor is located at entry $\imath_i$, with $1 \le i \le e$
$S_j \in \{0,1\}$, is 1 iff a sensor is located at exit $\jmath_j$, with $1 \le j \le s$
$V_i \in \{0,1\}$, is 1 iff an observer stands at entry $\imath_i$, with $i \in \Theta$
$P_{ij} \in \{0,1\}$, is 1 iff $\mathbf{p}'_{\imath_i \jmath_j}$ would be in $\mathcal{B}'$, with $1 \le i \le e$, $1 \le j \le s$.
$X_{ij} \in \{0,1,-1\}$ if $(i,j) \in \Gamma$. Otherwise, $X_{ij} \in \{0,1\}$, for all i, j.

$$X_{ij} = \begin{cases} 1 \text{ if } q_{\imath_i \jmath_j} \text{ is collected by sensor} \\ -1 \text{ if } (i,j) \in \Gamma \text{ and } q_{\imath_i \jmath_j} \text{ is directly collected at entry } \imath_i \\ 0 \text{ if } q_{\imath_i \jmath_j} \text{ is not collected} \end{cases}$$

The model we propose is biased to allow effective propagation of values and constraints when consistency techniques are applied. This is one of the reasons for choosing $\{0,1,-1\}$ as domain of X_{ij} when $(i,j) \in \Gamma$.

The objective function Minimize $\sum_{i=1}^{e} e_i E_i + \sum_{j=1}^{s} s_j S_j + \sum_{i \in \Theta} o_i V_i$, where e_i, s_j, and o_i denote the defined costs for sensors and observers. The o_i's are assumed relatively insignificant.

The constraints For all $1 \le i \le e$ and $1 \le j \le s$,

1. To record some flow $q_{\imath_i \jmath_j}$, sensors must be located at entry $\imath_i$ and exit $\jmath_j$, that is $E_i + S_j \ge 2X_{ij}$. Notice that, if $E_i + S_j \in \{0,1\}$ then $X_{ij} \in \{0,-1\}$.
2. To obtain $q_{\imath_i \jmath_j}$ by direct observation, there must be an observer at entry $\imath_i$, that is $V_i \ge -X_{ij}$.

3. If $q_{\imath_i\jmath_j}$ is not explicitly collected then $\mathbf{p}'_{\imath_i\jmath_j} \in \mathcal{B}'$. Thus, $X_{ij} = 0 \Leftrightarrow P_{ij} = 1$ or equivalently, $X_{ij} + P_{ij} = 1$, for all $(i,j) \notin \Gamma$ and $X_{ij}P_{ij} = 0 \wedge X_{ij} + P_{ij} \neq 0$ for all $(i,j) \in \Gamma$
4. We adopt the following conventions (that may be also useful during the search for the optimal solution). An observer is placed at entry $\imath_i$ only if it has to count some OD flow $q_{\imath_i\jmath_j}$, with $(i,j) \in \Gamma$.

$$V_i = 1 \Rightarrow \exists_{(i,j)\in\Gamma}\, X_{ij} = -1 \tag{4}$$

A sensor is located at entry $\imath_i$ only if some OD flow $q_{\imath_i\jmath_j}$, with $(i,j) \notin \Gamma$, has to be counted. The same holds for sensors at exits.

$$\sum_{j=1,(i,j)\notin\Gamma}^{s} X_{ij} \geq E_i \qquad \text{and} \qquad \sum_{i=1,(i,j)\notin\Gamma}^{e} X_{ij} \geq S_j \tag{5}$$

5. The set $\mathcal{B}' = \{\mathbf{p}'_{\imath_i\jmath_j} \mid P_{ij} = 1\}$ must be a basis to $\mathbf{P}'$. Equivalently, $\mathcal{B}'$ has cardinality $rank(\mathbf{P}')$, so $\sum_{i=1}^{e}\sum_{j=1}^{s} P_{ij} = rank(\mathbf{P}')$, and $\mathcal{B}'$ is a free set (i.e. its elements are linearly independent). The latter is equivalent to (6), where the variables $\beta_{ij} \in \mathbb{R}$.

$$\sum_{\mathbf{p}'_{\imath_i\jmath_j}\in\mathcal{B}'} \beta_{ij}\mathbf{p}'_{\imath_i\jmath_j} = \mathbf{0} \Rightarrow \beta_{ij} = 0, \text{ for all such } (i,j) \tag{6}$$

3.2 Implementation in CLP

Anyone with basic background on CLP(FD) may see that, with the exception of (4) and (6), these constraints can be straightforwardly encoded in CLP(FD). But, (4) may be implemented by *cardinality* operators — in SICStus Prolog, as `count(-1,XiStr,#>=,Vi)`, where `XiStr` is the list of X_{ij} with $(i,j) \in \Gamma$ — and, in ECLiPSe, as `#(Vi,CtrXi,NSti)`, where `CtrXi` is a list of constraints $X_{ij} = -1$ and `NSti` its length, for $(i,j) \in \Gamma$.

To encode (6) is certainly more tricky and challenging. First, (6) renders the model of MIP (mixed integer programming) type rather than a CSP, so that we would have to combine two different constraint solvers. Also, $\mathcal{B}'$ is in fact a variable in (6). To translate the membership constraint (i.e., $\mathbf{p}'_{\imath_i\jmath_j} \in \mathcal{B}'$), we could use the builtin constraint `element`, in SICStus Prolog. Nevertheless, (6) is too complex to be too often touched during optimization, unless we keep some track of nogoods.

Since we could not overcome these difficulties, we implemented C programs to support the former work, and abandoned CLP [13]. They handle particular cost criteria, computing the optimal bases $\mathcal{B}'$ through a completion procedure based on depth-first search with chronological backtracking. During the search, Gaussian elimination was used in an incremental way to check the elements already selected to $\mathcal{B}'$ for freeness. This also allowed to implement some form of intelligent backtracking. In fact, when some $\mathbf{p}'_{ij}$ is found linearly dependent on the columns already in $\mathcal{B}'$, such information can be propagated upwards along

the search tree. In this way, conflicts that may arise because of choices made near the root of the search tree can be detected. A careful analysis of the program output, gave us a deeper insight on the problem structure.

An important conclusion of [13], is that the condition $\beta_{ij} \in \mathbb{R}$ may be replaced by $\beta_{ij} \in \{0, 1, -1\}$, meaning that the problem is actually a CSP. Indeed, (1)–(2) are the restrictions of a network problem – a Linear Transportation problem (e.g., see [7, 8]) – and, consequently, $\mathbf{P}$ has well-known interesting properties. Based on them, we concluded that when a given $\mathbf{p}'_{ij} \notin \mathcal{B}'$ is a combination of the elements in a free subset $\mathcal{B}'$ of $\mathbf{P}'$, then the unique linear combination giving $\mathbf{p}'_{ij}$ is of the form (7). It might be useful to refer back to the example of SSEDE.

$$\mathbf{p}'_{ij} = \mathbf{p}'_{ij_1} - \mathbf{p}'_{i_1 j_1} + \mathbf{p}'_{i_1 j_2} - \cdots - \mathbf{p}'_{i_k j_k} + \mathbf{p}'_{i_k j} \tag{7}$$

In fact, the combination must mimic the one that defines $\mathbf{p}_{ij}$ as a combination of the related columns, say $\mathcal{B}$, in $\mathbf{P}$. In SSEDE, $\mathbf{p}'_{52} = \mathbf{p}'_{54} - \mathbf{p}'_{44} + \mathbf{p}'_{41} - \mathbf{p}'_{31} + \mathbf{p}'_{32}$, because $\mathbf{p}_{52} = \mathbf{p}_{54} - \mathbf{p}_{44} + \mathbf{p}_{41} - \mathbf{p}_{31} + \mathbf{p}_{32}$ and $\sigma_{52} = \sigma_{54} - \sigma_{44} + \sigma_{41} - \sigma_{31} + \sigma_{32}$.

Property 2. [13] For every subset $\mathcal{B}'$ of $\mathbf{P}'$, if $\mathcal{B}$ is not a free set, then $\mathcal{B}'$ is free if and only if there exists a $\mathbf{p}_{ij}$ such that $\mathcal{B} \setminus \{\mathbf{p}_{ij}\}$ is free and the equality 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 $\sigma_{ij} \neq \sigma_{ij_1} - \sigma_{i_1 j_1} + \sigma_{i_1 j_2} - \cdots - \sigma_{i_k j_k} + \sigma_{i_k j}$).

This property implies a necessary condition for $\mathcal{B}'$ to be free, that is that the columns in $\mathcal{B}$ make at most one cycle, in the sense we describe below.

To analyse cycles in a graph to test for linear dependence If we represent the elements of $\mathbf{P}$ in a tableau and link elements occurring consecutively in (7) by edges, a simple cycle is defined. For example, for *Praça da República* (SEESDSE), $\mathbf{p}'_{21} = \mathbf{p}'_{24} - \mathbf{p}'_{34} + \mathbf{p}'_{31}$ and $\mathbf{p}'_{55} \neq \mathbf{p}'_{56} - \mathbf{p}'_{76} + \mathbf{p}'_{75}$ may be deduced from its tables.

	1		4		5		6
2	$\mathbf{p}_{21}$	$\rightarrow$	$\mathbf{p}_{24}$		$\mathbf{p}_{25}$		$\mathbf{p}_{26}$
	$\uparrow$		$\downarrow$				
3	$\mathbf{p}_{31}$	$\leftarrow$	$\mathbf{p}_{34}$		$\mathbf{p}_{35}$		$\mathbf{p}_{36}$
5	$\mathbf{p}_{51}$		$\mathbf{p}_{54}$		$\mathbf{p}_{55}$	$\rightarrow$	$\mathbf{p}_{56}$
					$\uparrow$		$\downarrow$
7	$\mathbf{p}_{71}$		$\mathbf{p}_{74}$		$\mathbf{p}_{75}$	$\leftarrow$	$\mathbf{p}_{76}$

σ_{ij}	1	4	5	6
2	0	0	0	0
3	0	0	0	0
5	0	1	1	0
7	0	1	1	1

The table on the right contains the coefficients σ_{ij} wrt F_1. The value of the coefficient σ_{ij} of q_{ij} in the equation that describes F_1 is given by:

$$\sigma_{ij} = 0 \text{ if } j = 1 \text{ or } j > i\text{; otherwise } \sigma_{ij} = 1$$

since no vehicle that goes from i to j shall pass in front of road 1, by our hypothesis that it does not pass its destination. The *squares are the simplest cycles*. Four $\mathbf{p}'_{ij}$'s in a square may be independent, though the related $\mathbf{p}_{ij}$'s are not. In general, a subset $\mathcal{B}$ of the columns of $\mathbf{P}$ is free if and only if the graph $G_{\mathcal{B}}$

built in the following way is acyclic. Its 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 closest vertex in the same row (respectively, column), if any exists. A subset $\mathcal{B}$ is a basis of the subspace spanned by the columns of $\mathbf{P}$ if and only if $G_{\mathcal{B}}$ is a tree and contains at least a vertex in each column and in each row of the tableau (e.g., [7, 8]). This result, together with Property 2, supports our implementation of `basis/5`, which we address in section 3.5. This predicate checks whether a set of columns of $\mathbf{P}'$ makes a basis.

It is not possible to define simple constraints to prevent problematic cycles, that would let us discard (6). But, it is rather simple to avoid problematic squares. In section 3.3 we define constraints to avoid them and in section 3.4 we will see how to compute such constraints efficiently. Furthermore, we argue and describe how some may be replaced by more global constraints with significant enhancements of performance.

3.3 The Square-Free Property

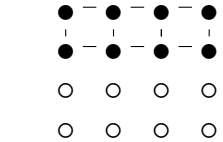
To benefit from consistency-based approaches to solving CSPs [9], we needed to replace (6) by some constraints that permit stronger constraint propagation during search. Moreover, we did not really want to spend much time improving the code, and in particular (6). Three necessary conditions on $\mathcal{B}'$ are implied by the properties $G_{\mathcal{B}}$ must have if (6) holds: $\sum_{j=1}^{s} P_{ij} \geq 1$, $\sum_{i=1}^{e} P_{ij} \geq 1$ and what we call *the Square-Free Property* (SQF, for short). By imposing SQF, we prevent problematic squares. It says that: *if* $\sigma_{\iota_i \jmath_j} - \sigma_{\iota_i \jmath_{j'}} + \sigma_{\iota_{i'} \jmath_{j'}} - \sigma_{\iota_{i'} \jmath_j} = 0$, *then*

$$P_{ij} + P_{ij'} + P_{i'j'} + P_{i'j} < 4 \tag{8}$$

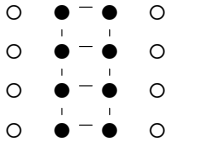
and if, in addition, none of four vertices (of the square) contains a $\star$, *that is* $\{(i,j),(i,j'),(i',j'),(i',j)\} \cap \Gamma = \emptyset$, *then* $E_i + E_{i'} \geq 1$ *and* $S_j + S_{j'} \geq 1$.

The latter are binary constraints and thus they have the right form to achieve good propagation by arc-consistency techniques. Nevertheless, when the examples become larger, too many constraints are generated and the propagation becomes less effective. More global constraints, that provide stronger pruning, may be easily deduced by inspection of the table of the σ_{ij}'s, as we shall see in section 3.4. To illustrate the main idea, we show what happens for SEESDSE when there are no $\star$'s. As before, the entries with $\bullet$ are of $\mathbf{p}'_{ij} \in \mathcal{B}'$ (clearly, in both situations $\mathcal{B}'$ would not be a basis). The analysis of the enlightened columns and rows shows that the optimal cost is $(e-1)+(s-1) = 6$.

	1	4	5	6
2	0	0	0	0
3	0	0	0	0
5	0	1	1	0
7	0	1	1	1

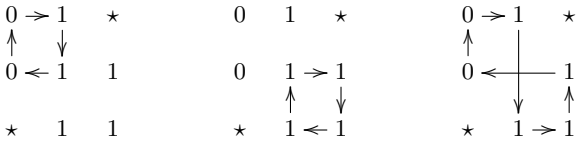


$S_1 + S_2 + S_3 + S_4 \geq 3$
$E_1 + E_2 \geq 1$



$E_1 + E_2 + E_3 + E_4 \geq 3$
$S_2 + S_3 \geq 1$

SQF is not equivalent to (6) in general. Therefore, if we impose SQF instead of (6) there is no guarantee that the computed solution is a basis. Actually, it fails to prune cycles as the one shown below on the right (for SSEDE with $\Gamma = \{(1,3),(3,1)\}$), though it prevents the first two cases.



Therefore we have implemented the following strategy.

> *Enforce all constraints except (6), but add the ones imposed by SQF.*
> *Generate solutions by labeling the decision variables in E, S, V, P.*
> *Check whether P defines a basis.*

The latter two tasks are interleaved when the program is looking for an optimal solution that, in our implementation for SICStus Prolog, is computed by calling

```
minimize(solution(Vars,P,Rank,Ne,Ns,Sigmas),Cost)
```

with `solution/6` as shown below. `Vars` is the list of all the decision variables except the X_{ij}'s (so, we had a `?` in the solution), with the P_{ij}'s in its tail.

```
solution(Vars,P,Rank,Ne,Ns,Sigmas) :-
   labeling([],Vars),   % with no preferential strategy
   basis(P,Rank,Ne,Ns,Sigmas).
```

SQF often provides relevant propagation and significant pruning. Indeed, in real cases, we cannot usually expect an observer to be able to count more than three OD flows even if their destinations are in sight. An extensive simulation for randomly generated roundabouts, with $4 \leq n \leq 10$ and less than four $\star$'s per row, shows that no-good solutions (i.e., for which `basis/5` failed) were found in less than 5% of the cases. We note that the larger the example is, the better SQF may work though the binary constraints on the E_i's and S_j's have to be replaced by more global (non-binary) ones.

3.4 Deducing Global Constraints from SQF

Given two rows i and i' of the tableau, we say that *i and i' are incompatible* if and only if $E_i + E_{i'} \geq 1$ is entailed by the SQF-Property. In a similar way, we define j and j' as incompatible columns if $S_j + S_{j'} \geq 1$ must hold. When applied to two given rows i and i', SQF splits the columns in two sets, say $\mathcal{A}_{(i,i')}$ and $\mathcal{B}_{(i,i')}$, according to whether or not they are *incompatible* with the first one. All columns in the same set are pairwise and globally incompatible.

First we suppose that $\Gamma = \emptyset$ (i.e., no q_{ij} can be directly observed). By successively analyzing pairs of rows, and enforcing SQF-Property, we deduce the following constraints, as well as that $S_j + S_{j'} \geq 1$, for all (j, j').

0 0 0 0	$E_1 + E_2 \geq 1$ $S_1 + S_2 + S_3 + S_4 \geq 3$	$\mathcal{A}_{(1,2)} = \{1,2,3,4\}, \mathcal{B}_{(1,2)} = \emptyset$
0 0 0 0	$E_1 + E_3 \geq 1$ $S_1 + S_4 \geq 1,\ S_2 + S_3 \geq 1$	$\mathcal{A}_{(1,3)} = \{1,4\}, \mathcal{B}_{(1,3)} = \{2,3\}$
0 1 1 0	$E_1 + E_4 \geq 1$ $S_2 + S_3 + S_4 \geq 2$	$\mathcal{A}_{(1,4)} = \{1\}, \mathcal{B}_{(1,4)} = \{2,3,4\}$
0 1 1 1	$E_2 + E_3 \geq 1$ $S_1 + S_4 \geq 1,\ S_2 + S_3 \geq 1$	$\mathcal{A}_{(2,3)} = \{1,4\}, \mathcal{B}_{(2,3)} = \{2,3\}$
	$E_2 + E_4 \geq 1$ $S_2 + S_3 + S_4 \geq 2$	$\mathcal{A}_{(2,4)} = \{1\}, \mathcal{B}_{(2,4)} = \{2,3,4\}$
	$E_3 + E_4 \geq 1$ $S_1 + S_2 + S_3 \geq 2$	$\mathcal{A}_{(3,4)} = \{1,2,3\}, \mathcal{B}_{(3,4)} = \{4\}$

We see that $E_i + E_{i'} \geq 1$, for all i and i'. Our implementation would enforce instead the global constraint $E_1 + E_2 + E_3 + E_4 \geq 3$. To do that, it computes the maximal cliques of these constraints' graph (by finding the *maximal independent sets of the complementary relation*), as we refer below. Just $S_1 + S_2 + S_3 + S_4 \geq 3$ is imposed because it makes the other constraints on the S_j's redundant.

Now, for Praça da República, we may deduce the following constraints, and the program imposes $E_1 + E_2 + E_3 \geq 2$.

0 $\star$ $\star$ 0	$E_1 + E_2 \geq 1$	$S_1 + S_4 \geq 1$	$\mathcal{A}_{(1,2)} = \{1,4\},\ \mathcal{B}^{(1,2)} = \emptyset$
0 $\star$ $\star$ 0	$E_1 + E_3 \geq 1$	$S_1 + S_4 \geq 1$	$\mathcal{A}_{(1,3)} = \{1,4\},\ \mathcal{B}_{(1,3)} = \emptyset$
0 1 1 0			$\mathcal{A}_{(1,4)} = \{1\},\ \mathcal{B}_{(1,4)} = \{4\}$
$\star$ 1 1 1	$E_2 + E_3 \geq 1$	$S_1 + S_4 \geq 1$	$\mathcal{A}_{(2,3)} = \{1,4\},\ \mathcal{B}_{(2,3)} = \emptyset$
			$\mathcal{A}_{(2,4)} = \{1\},\ \mathcal{B}_{(2,4)} = \{4\}$
	$E_3 + E_4 \geq 1$	$S_2 + S_3 \geq 1$	$\mathcal{A}_{(3,4)} = \{2,3\},\ \mathcal{B}_{(3,4)} = \{4\}$

The reason for such partitions is that the first column of every tableau contains only 0's and for each row that contains two or more 0's, only 0's occur to the right of the second 0. Furthermore, in each column, there are only 1's below each 1. Thus there are five types of the problematic squares to avoid:

0 0	1 1	0 1	0 0	1 0
0 0	1 1	0 1	1 1	1 0

From this observation, it is possible to design an elegant algorithm for finding the pairs of incompatible columns (respectively rows). Let b_i be the column of the second 0 in row i, if any exists, and Γ_i be the set of columns that contain $\star$'s in the row i. We may conclude that $b_i = \min(s+1, \{j \mid \jmath_j \in \mathcal{D} \text{ and } \jmath_j > \imath_i\})$ and $\Gamma_i = \{j \mid (i,j) \in \Gamma\}$. Then, the following lemmas are of help to implement the SQF-Property. Their proofs follow quite immediately from our previous observation.

Lemma 1. *For all $(i,i'),(j,j')$ such that $1 \leq i < i' \leq e$ and $1 \leq j < j' \leq s$, it holds $\sigma_{\imath_i \jmath_j} - \sigma_{\imath_{i'} \jmath_{j'}} + \sigma_{\imath_{i'} \jmath_{j'}} - \sigma_{\imath_{i'} \jmath_j} = 0$ if and only if either $j, j' \in [1, b_i[\cup [b_{i'}, s]$ or $j, j' \in [b_i, b_{i'}[$. For all (j,j') with $1 \leq j < j' \leq s$, the columns j and j' are incompatible iff $j, j' \in ([1, b_i[\cup [b_{i'}, s]) \setminus (\Gamma_i \cup \Gamma'_i)$ or $j, j' \in [b_i, b_{i'}[\setminus (\Gamma_i \cup \Gamma'_i)$, for some (i,i') such that $1 \leq i < i' \leq e$.*

Now, let $\mathcal{A}_{(i,i')} = ([1, b_i[\cup [b_{i'}, s]) \setminus (\Gamma_i \cup \Gamma'_i)$ and $\mathcal{B}_{(i,i')} = [b_i, b_{i'}[\setminus (\Gamma_i \cup \Gamma'_i)$ be the *two* sets of incompatible columns we obtain by enforcing the SQF-Property wrt rows i and i'. Some examples are shown above.

Lemma 2. *The constraints $S_j + S_{j'} \geq 1$ that SQF-Property may impose are satisfied if instead we have (9) for all (i, i') such that $1 \leq i < i' \leq e$.*

$$\sum_{j \in \mathcal{A}_{(i,i')}} S_j \geq |\mathcal{A}_{(i,i')}| - 1 \quad \textit{and} \quad \sum_{j \in \mathcal{B}_{(i,i')}} S_j \geq |\mathcal{B}_{(i,i')}| - 1 \tag{9}$$

The ones imposed on P are equivalent to $P_{ij} + P_{ij'} + P_{i'j'} + P_{i'j} < 4$, for all $1 \leq i < i' \leq e$ and $1 \leq j < j' \leq s$ such that $j, j' \in [1, b_i[\cup [b_{i'}, s]$ or $j, j' \notin [1, b_i[\cup [b_{i'}, s]$. Two rows i and i', such that $1 \leq i < i' \leq e$, are incompatible if $|\mathcal{A}_{(i,i')}| \geq 2$ or $|\mathcal{B}_{(i,i')}| \geq 2$.

From these lemmas we see that it is possible to design an algorithm for establishing the SQF-Property whose main task is to compute the intervals $[1, b_i[\cup [b_{i'}, s]$, $\mathcal{A}_{(i,i')}$ and $\mathcal{B}_{(i,i')}$, which is not difficult at all. By doing that, we get enough information about sets of incompatible columns and pairs of incompatible rows. Then, to obtain more global constraints on the E_i's, we may consider the complementary relation — defined by the pairs (i, i') of compatible rows — and find all *maximal independent sets with at least two elements* in the undirected graph that represents it. Though this is not a great idea in general, since finding *all* these sets is an NP-complete problem (e.g. [6]), it works quite well in our specific case, because the graphs are small for the small examples and too sparse (i.e., very likely, have no edges) for the largest ones. Using `library(ugraphs)`, in SICStus, we may find these independent sets by calling `findall(Set,independent_set(UGraph,2,Set),LSets)`, for the undirected graph `UGraph` that represents the relation. In the implementation, we start from a complete graph `CGraph` with vertices `1..s`, and subsequently remove the edges that link incompatible columns to yield the compatibility relation for the colums. At the same time, we construct the graph `RGraph` of the compatibility relation for the rows and impose (8). The independent sets are found for both these graphs to deduce the global constraints on the E_i's and S_j's.

3.5 Detecting Problematic Cycles

We finally describe how we defined `basis/5` in a declarative way. Let $\mathcal{B}' = \{\mathbf{p}'_{\imath_i\jmath_j} \mid P_{ij} \text{ is labelled to 1}\}$ and let $\mathcal{N}_{\mathcal{B}'} = \{(i, j) \mid \mathbf{p}_{\imath_i\jmath_j} \in \mathcal{B}'\}$. The main idea is to obtain the largest subset $\mathcal{S} \subseteq \mathcal{B}'$ such that the graph $G_{\mathcal{S}}$ is a tree. We note that if $\mathcal{B}'$ is a basis then either $\mathcal{S} = \mathcal{B}' \setminus \{\mathbf{p}'_{\imath_o\jmath_d}\}$, for one $\mathbf{p}'_{\imath_o\jmath_d} \in \mathcal{B}'$, or `Rank` is $e + s - 1$ and $\mathcal{S} = \mathcal{B}'$. The unification plays a subtle role, enabling to check in an elegant way this necessary condition on $\mathcal{B}'$.

```
basis(P,Rank,Ne,Ns,Sigmas) :-
  find_nodes(P,Bprime,1,Sigmas),
    % uses P to define Bprime as a list [I-[n(J,SigmaIJ)|...]|...]
  remain(LPod,Rank,Ne,Ns),
    % creates LPod as [] if Rank=Ne+Ns-1; if not LPod=[_]
  lstsingletons(1,Ns,Trees), % creates Trees as [[1],...,[Ns]]
  forest(Bprime,Trees,[_],LPod,S),
  cfree(LPod,S,Ne,Ns).
```

Moreover, `forest(Bprime,Trees,[_],LPod,S)` implements a deterministic completion procedure to find $\mathcal{S}$ and $\mathbf{p}'_{\imath_o\jmath_d}$. The possible resulting $\mathbf{p}'_{\imath_o\jmath_d}$ is the first node found to violate the tree-shape. We recall that the argument `LPod` is bounded to `[]` or `[Pod]`, depending on `Rank`. At iteration i, we try to link the chain consisting of the nodes $(i,j) \in \mathcal{N}_{\mathcal{B}'}$ selected in row i. To do that we adopt the idea of the Kruskal's Algorithm for minimum spanning trees [6].

```
forest([],Trees,Trees,_,[]).
forest([I-NodesI|Nodes],Trees,Treesf,LPod,[I-BasisI|Basis]) :-
   deleting(NodesI,Trees,RTrees,LTrees,I,LPod,BasisI),
   flatten(LTrees,NewTree),
   forest(Nodes,[NewTree|RTrees],Treesf,LPod,Basis).
```

In calls to `forest/5`, the first argument is a list of terms `I-NodesI` that represent the selected nodes in row `I`, being `NodesI` a list of terms `n(J,SigmaIJ)`. If two nodes in the same row were to be attached to the same tree, then the predicate `deleting/7` puts the second one in `LPod`. It gives in `LTrees` the list of trees where the nodes in row `I` were attached to, and the remaining trees in `RTrees`.

We refer to the roundabout SSEDE (with $\Gamma = \{(1,3),(3,1)\}$), to illustrate the behavior of `forest/5` and `cfree/4`.

```
?- forest([1-[n(1,0),n(2,1)],2-[n(1,0),n(3,1)],3-[n(2,1),n(3,1)]]
           ,[[1],[2],[3]],[Tree],[Pod],S).

Pod = n(3,3,1),
Tree = [1,2,3],
S = [1-[n(1,0),n(2,1)],2-[n(1,0),n(3,1)],3-[n(2,1)]]

?- cfree([n(3,3,1)],[1-[n(1,0),n(2,1)],2-[n(1,0),n(3,1)],3-[n(2,1)]],3,3).
no
```

If the program call to `forest/5` succeeds then, when `Rank` is $e+s$, we still have to check whether $\mathbf{p}'_{\imath_o\jmath_d}$ (which `Pod` identifies) can be written as a linear combination of the elements in $\mathcal{S}$ (which is represented by the last argument of `forest/6`). This is handled by `cfree(LPod,S,Ne,Ns)`, which trivially succeeds if `LPod=[]`. If not, it first solves the following CSP in finite domains, that has a *unique* solution for the $e+s$ variables β_{ij}'s.

$$\sum_{\mathbf{p}'_{\imath_i\jmath_j} \in \mathcal{S}} \beta_{ij} \mathbf{p}_{\imath_i\jmath_j} = \mathbf{p}_{\imath_o\jmath_d}, \;\; \beta_{ij} \in \{0,1,-1\}$$

```
cfree([],_,_,_).
cfree([Pod],S,Ne,Ns) :- NBetas is Ne+Ns-1,
   length(Betas,NBetas),
   domain(Betas,-1,1),
   constraints_cfree(Ns,S,Pod,Betas,SigmaPod,SigmasS),
   labeling([],Betas), !,  % unique solution
   scalar_product(SigmasS,Betas,#\=,SigmaPod).
```

The constraints may be created during execution, because $\mathbf{p}_{\imath_i\jmath_j} = \mathbf{e}_i + \mathbf{e}_{e+j}$, with $\mathbf{e}_i$ and $\mathbf{e}_{e+j}$ unit vectors of the real space of dimension $e+s$. To conclude that $\mathcal{B}'$ is a basis, the program checks whether $\sum_{\mathbf{p}'_{\imath_i\jmath_j}\in\mathcal{S}} \beta_{ij}\sigma_{\imath_i\jmath_j} \neq \sigma_{\imath_o\jmath_d}$, for the computed β_{ij}'s. The constraint `scalar_product(SigmasS,Betas,#\=,SigmaPod)` is consistent iff $\mathcal{B}'$ is a basis.

If the cost function is defined by $c_s(\sum_{i=1}^{e} E_i + \sum_{j=1}^{s} S_j) + c_o(\sum_{i\in\Theta} V_i)$ then the input for Praça da República would be

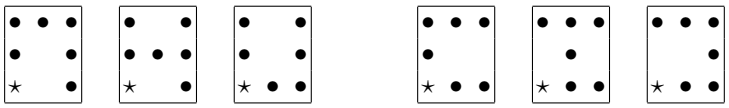
```
'SEESDSE'.
[2-[4,5],3-[4,5],7-[1]].
[10,1].
```

where `[10,1]` is $[c_s, c_o]$ (by our decision), and `[2-[4,5],3-[4,5],7-[1]]` the positions of $\star$'s. The following is an output solution.

```
e=[0,1,1,0]
s=[0,0,1,1]
v=[1,1,0,1]
p=[[1,0,0,1],[1,0,0,0],[1,1,0,0],[0,1,1,1]]
x=[[0,-1,-1,0],[0,-1,_10160,1],[0,0,1,1],[-1,0,0,0]]
```

3.6 Dual Problems

We have seen that this problem has an interesting mathematical structure. Another interesting property is illustrated by the roundabout SSEDE, with $\Gamma = \{(1,3),(3,1)\}$, that was considered in section 3.3. The rank of the system matrix is $e+s=6$, implying that we can avoid counting six of the q_{ij}'s. The optimization problem may be seen as that of placing six $\bullet$'s in the tableau, so as to cover, with $\bullet$'s and $\star$'s, the largest number of columns and rows, while ensuring the linear independence of the $\mathbf{p}'_{ij}$'s with $\bullet$'s. This problem has six optimal solutions.



The optimal cost is thus given as $(e-1)+(s-2)+1^\star$ and $(e-2)+(s-1)+1^\star$, where $1^\star$ means that one $q_{i\,i+1}$ must be obtained by direct observation. In fact, $S_1+S_2 \geq 1$, $E_1+E_2 \geq 1$ $S_2+S_3 \geq 1$ and $E_2+E_3 \geq 1$ are enforced by SQF. Moreover, the cycle we showed, in section 3.3, makes it impossible to cover two exits and two entries at the same time. The symmetry between entries and exits that the cost expressions illustrate is due to the following property: *for every given roundabout $r = R_1R_2\ldots R_n$, let $\Psi(r)$ denote the string that we obtain if we exchange E with S in r and read the resulting expression from right to left. The roundabouts r and $\Psi(r)$ yield exactly the same system of equations, if q^r_{ij} is replaced by $q^{\Psi(r)}_{n-j+1\,n-i+1}$, and the constraints are re-ordered. The optimal cost wrt r is $(e_r-n_1)+(s_r-n_2)+n_3^\star$ iff the optimal cost for $\Psi(r)$ is given by*

$(e_{\Psi(r)} - n_2) + (s_{\Psi(r)} - n_1) + n_3^\star$, *where* $e_r = s_{\Psi(r)}$ *and* $s_r = e_{\Psi(r)}$ *stand for the numbers of entries and exits.*

Because $\Psi(\text{SSEDE}) = \text{SSEDE}$, there are two alternative expressions for the optimal cost.

This was quite significant to achieve a complete characterization of the optimal solutions when we considered that all and only the flows $q_{i\,i+1}$'s could be obtained by less expensive means. Here, $i+1$ refers to the road that immediately follows road i, in the roundabout, for all $i \in \mathcal{O}$.

By a case-based approach, using the test for freeness and SQF to identify the relevant cases, we were able to exactly characterize the optimal cost for each type of roundabout $R_1 R_2 \ldots R_n$. There are relatively few cases to analyse. All the strings in a subset closed under string rotation represent the same roundabout, and there is also this kind duality between entries and exits. Another crucial factor is that when both $e \geq 5$ and $s \geq 5$, the optimal cost is $(e-1) + (s-1)$, by SQF, since there is at most one $q_{i\,i+1}$ in each line of the tableau.

Each case is nothing but a class of roundabouts with exactly the same optimal solutions, being possible to describe their patterns by some *regular expression*. Naturally, the closure properties of string rotation extend to the possible identifying expressions. In broad terms, classes are distinguished by four major features, namely the numbers of entries and of exits (respectively, e and s), the number of directional flows $q_{i\,i+1}$ and the location of the $\mathbf{p}'_{i\,i+1}$'s in the tableau.

Since, only the cases where either $e < 5$ or $s < 5$ remained to study, by designing some *deterministic finite state automata* we trivially achieved a first coarse description of the classes. Further details may be found in [14].

4 Further Research and Conclusion

This paper illustrates that it may be not straightforward to model real world problems so as to solve them using CLP. Often it is required some deep knowledge about the mathematical structure of the problem to be able to achieve strong constraint propagation. We are also studying whether Constraint Programming frameworks may be successfully applied to sensor location problems (SLP) for Transportation Networks, which seem to have a more complex structure.

Two recent publications [3, 12] address the optimization of the number and location of traffic counting points for traffic networks. While in [12], the emphasis is put on the quality of the estimation, examining some heuristics for locating count-posts, in [3], a model and heuristic based algorithms are given for solving the minimum-cost SLP. Traffic networks may be represented by directed graphs. For the internal nodes (as opposed to the so called *centroids*) the inflow equals the outflow. In [3] it is assumed that the turning probabilities at junctions (i.e., nodes) are known, which implies that if we know the flow q_{ij} (from node i to node j) then q_{ik} becomes known for every k adjacent to i. This suggests that constraint programming may be useful to efficiently solve the problem of finding the nodes where the sensors will collect data. When a node has a sensor, then all its incident flows (inflows and outflows) become known. The difficulty is how

to declaratively state (and translate) that each q_{ij} may be obtained either by direct observation (by a sensor at either node i or node j) or by knowledge (i.e., constraints) propagation.

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, Portugal, 2000.
2. Carlsson M., Ottosson G., Carlson B.: An Open-Ended Finite Domain Constraint Solver. In *Proceedings of PLILP'97*, LNCS 1292, 191-206, Springer-Verlag, 1997.
3. Bianco L., Confessore G., and Reverberi P.: A Network Based Model for Traffic Sensor Location with Implications on O/D Matrix Estimates. *J. Transportation Science*, 35(1), 50–60, 2001.
4. Cook W., Cunningham W., Pulleyblank W., and Schrijver A.: *Combinatorial Optimization*. John Wiley & Sons, 1998, Chapter 8.
5. ECLiPSe User Manual Version 5.1.2, IC-PARC, Inperial College, London, 2001.
6. Cormen H., Leiserson C., Rivest R.: *Introduction to Algorithms*, MIT Press, 1990.
7. Hadley G.: *Linear Programming*. Addison-Wesley, 1969.
8. Hillier S. H., Lieberman G. J.: *Introduction to Operations Research*. McGraw-Hill, 6th Ed, 1995.
9. Marriott K., and Stuckey P.: *Programming with Constraints – An Introduction*, The MIT Press, 1998.
10. Mansfield L. E.: *Linear Algebra with Geometric Applications*, Marcel Dekker, 1976.
11. SICStus Prolog User Manual Release 3.8.6, SICS, Sweden, 2001.
12. Yang H., and Zhou J.: Optimal Traffic Counting Locations for Origin-Destination Matrix Estimation. *J. Transportation Research*, 32B(2), 109–126, 1998.
13. Tomás A. P., Andrade M., and Pires da Costa A.: On optimal location of traffic counts in roundabouts, Internal Report DCC-2-2001, DCC-FC & LIACC, University of Porto, 2001. (Presented at CSOR'01, Porto, 12/2001)
    `www.ncc.up.pt/fcup/DCC/Pubs/treports.html`
14. Tomás A. P.: A Note on Sensor Location for Traffic Counting at Roundabouts – Solutions for a Particular Cost Function, Internal Report DCC-3-2001, DCC-FC & LIACC, University of Porto, 2001.
15. Tomás A. P.: Solving Optimal Location of Traffic Counting Points at Urban Intersections in CLP(FD). In C. A. Coello Coello et al (Eds.), *Proc. of MICAI'2002: Advances in Artificial Intelligence*, LNAI 2313, Springer-Verlag, 242–251, 2002.