K-Best Routing Strategies in a Labelled Graph: Application to the Design of a Public Transportation Network

J. Maublanc [1] *D. Peyrton* [1] *A. Quilliot* [2]

[1] Certu Lyon

[2] Université BLAISE PASCAL
ISIMA BP 125, Campus des Cézeaux
63173 Aubiere, France

Abstract
We present here models and algorithms for the construction of efficient path systems, robust to possible variations of the characteristics of the network. We propose some interpretations of these models and proceed to numerical experimentations of the related algorithms. We conclude with a discussion of the way those concepts may be applied to the design of a Public Transportation System.

Keywords: Shortest Paths, Network Design, Routing.

1 Introduction

Shortest path problems are among the combinatorial optimization problems which have been the most widely studied during the last thirty years. Applications to stock management, planning, routing and network design have first led to very well-known algorithms designed for explicitly defined networks: Bellman algorithms for acyclic networks [8,17], Dijkstra algorithm for positive networks [2,3], Dantzig algorithm for the general case...[9, 11, 19]. Applications related to robotics and strategic games have next motivated several adaptations of these methods: A* and B* algorithms for very large scale state networks, stochastic models for the case when the effects of some actions or transitions cannot be predicted in a deterministic way [1, 10, 14, 16].

It may occur that we can't expect to get a complete knowledge of the future state of the network at the time when we are required to handle the path search problem.

In such a case, we need to look for several paths, which at the same time are the most efficient possible and also are pairwise independent with respect to the possible future configurations of the network. For instance, routing messages or goods through some telecommunication or transportation network or through some multiprocessor MIMD architecture is usually managed in two steps: the first one consists in prealably computing some convenient candidate paths, and the second one consists, every time the routing process is launched, in picking up [5, 6, 7, 13, 21], while taking into account the current state of the network, the path which seems the most efficient at this time.

Then the underlying problem consists in searching, for a given origin/destination pair (o,d), some path family, which is made with paths from o to d, independent in the sense that their performances under the possible variations of the state of the network are not too narrowly correlated, and which defines at the same time some kind of shortest path family.

It is with this problem which we are going to deal here. We shall first propose some general model, aimed at providing some formalization of the above notion of independence, together with some examples and resolution algorithms. We will discuss the complexity of these algorithms and conclude by focusing on a specific case, related to transportation system design, and by presenting an application.

2 A General Model

2.1 Preliminaries

Let us consider some network G = (X,E). For any arc e in G, we denote by $o(e)$ the origin of e and by $s(e)$ the extremity of e.

A path γ in G is a finite sequence $\gamma = e_1..e_n$ of arcs in E such that for any $i = 1...n-1, o(e_{i+1}) = s(e_i)$. The vertex $o(e_1)$ is called the origin of γ and is denoted by $or(\gamma)$; The vertex $s(e_n)$ is called the extremity of γ and is denoted by $ext(\gamma)$. Such a path γ is said to be elementary if the vertices $o(e_i), i = 1...n$, and $s(e_n)$, are pairwise distinct.

The vertex sequence $o(e_1), ..., o(e_n), s(e_n)$, is called the support of the path γ and is denoted by $Supp(\gamma)$.

In case no two arcs of G connect the same pair of vertices, we consider that γ is completely defined by its support.

If d is a length function which to any arc e in E makes correspond some real number d(e), then we set:

$d*(\gamma) = \sum_{i=1...n} d(ei)$ = d-length of γ.

If x and y are two vertices in the support of γ, such that x is located before y on $Supp(\gamma)$, then we denote by $\gamma_{x,y}$ the subpath of γ whose origin is x and whose extremity is y, and we call it the restriction of γ from x to y.

We denote by P(G) the set of all the elementary paths of G and by P*(G) the set of all the paths of G.

We denote by Nil the empty path, which may be viewed as connecting any vertex to itself.

If γ and γ' are two paths such that $or(\gamma') = ext(\gamma)$, then we denote by $\gamma + \gamma'$ the concatenation of γ and γ'.

For any pair x,y of vertices of G, we denote by $P(G)_{x,y}$ the set of all elementary paths whose origin is x and whose extremity is y.

2.2 Notion of Strategic Triple

The crucial notion behind our problem is the notion of independence. We want to express the fact that several paths are efficient, in the sense that they allow a fast connection between two given vertices, while being independent with relation to the evolution of some set of state parameters.

Practically, two paths will be considered as independent, if their underlying strategies are not the same and if their performance are not tied to each other in a logical way. When talking about strategy, we may think for instance into some traveller which moves from one area to another: its underlying strategy is basically defined by the sequence of transportation modes that he uses. More generally, the arcs of the network G= (X,E) are in most of the cases endowed with some kind of symbolic or numerical information, and the notions of equivalence or domination between strategies must express themselves through some equivalence or partial ordering relations on the word set defined by all the possible concatenations of those informations.

In order to cast these intuitions into a general model, let us consider some network G = (X,E).

We call Strategic Triple defined on G, any triple (R, O, L) where:

- R is an equivalence relation defined on the path set P(G)*;

- O is a partial order relation defined on P(G)*; (O may be read "more efficient than")

- L is a subset of P(G)*;

which is such that:

(H1): O and R are compatible, which means that if γ, γ' and γ'' in P(G)* are such that:

- γ' R γ'';

- γ O γ' (γ' O γ)

then we also have: γ O γ'' (γ'' O γ).

(H2): For any *gamma* in P(G)*, Nil R γ or Nil O γ (Nil O $\vee$ R γ)

(H3): R and O are stable by right side concatenation, which means that if γ, γ' and γ'' in P(G)* satisfy:

- γ' R (O) γ'';

- or(γ) = ext(γ') = ext(γ'');

then we also have: ($\gamma' + \gamma$) R (O) ($\gamma'' + \gamma$).

(H4): There exists a projection operator $\prod$ from P(G)* to L which is such that:

- for any γ in L, $\prod(\gamma) = \gamma$;

- for any γ in P(G)*, or($\prod(\gamma)$) = or(γ) and ext($\prod(\gamma)$) = ext(γ);

- $\prod$ is an homomorphism for both relations R and O $\vee$ R (It conserves both relations);

- if γ is in L, and if u is an arc of G, such that ext(γ) = o(u), then there exists some vertex x in the support of γ such that the arc [x,s(u)] exists and that $\prod(\gamma + \{u\}) = \gamma_{or(\gamma),x} + \{[x, s(u)]\}$.

(H5): Any subpath of some path γ in L is also in L.

If d is a positive length function defined on the arc set of G, we say that d is compatible with the strategic triple (R,O,L) if for any path γ in P(G)*, we have:

$d * (\prod(\gamma)) \leq d * (\gamma)$.

Comments: The equivalence relation R formalizes here the notion of strategy. A strategy is an equivalence class for R. We call Strategy Set associated with G = (X,E) and with the strategic triple (R,O,L) the quotient set P(G)*/R, and we denote it by ST(G,R).

Axiom (H1) means that the partial ordering O defines a partial ordering O* on the Strategy Set ST(G,R).

Axiom (H2) and (H3) mean that any subpath of a given path γ is usually less costly or more fiable than γ.

The subset L of P(G)* expresses the fact that the concept of strategy may be restrained by some syntactical constraints. If for instance we think into a strategy as being some sequence of transportation modes, we will only consider alternated mode sequences, that means sequences which never contain 2 identical consecutive symbols. In such a case, Axiom (H4) ensures that to any path of G will correspond some strategy.

2.3 Minimal Independent Path Families

So, let G = (X,E) be some network, let (R,O,L) be some strategic triple on G and let x,y be two vertices in G.

Two pathes γ and γ' in P(G)* are said to be R-equivalent iff γ R γ'; they are said to be R-independent iff they are not R-equivalent.

We say that γ O-dominate γ' iff γ O γ'; we say that γ and γ' are strongly (O,R)-independent iff they are R-independent and if none of both O-dominates the other.

A family $\Lambda = (\gamma_1 ... \gamma_k)$ of pathes of $P(G)_{x,y}$ will be called a k minimal independent path family from x to y associated with the strategic triple (R,O,L) iff:

- the pathes in Λ are all in L and are pairwise R-independent;

- if i in {1...k}, and γ in $P(G)_{x,y}$ are such that γ O γ_i, then there exists j < i such that γ R γ_j.

Let us suppose now that G is endowed with some real valued length function d, defined on its arc set E and compatible with (R,O,L). A family $\Lambda = (\gamma_1 ... \gamma_k)$ of pathes of $P(G)_{x,y}$ will be said to be a k minimal strongly independent path family from x to y associated with the strategic triple (R,O,L) and with the length function d, iff:

- the pathes in Λ are all in L and are pairwise strongly (O,R)-independent;

- if i = 1...k, and γ in $P(G)_{x,y}$ are such that $d*(\gamma) < d*(\gamma_i)$, then there exists j $\leq$ i such that γ_j O γ or that γ_j R γ and $d*(\gamma_j) < d*(\gamma)$.

One may raise two problems:

Problem 1: Given some network G = (X,E), some strategic triple (R,O,L), some pair x, y of vertices of G, and some integer k, find an associated k minimal independent path family from x to y.

Problem 2: Given some network G = (X,E), some strategic triple (R,O,L), some pair x, y of vertices of G, some length function d defined on E and compatible with (R,O,L), and some integer k, find an associated k minimal strongly independent path family from x to y.

Remark: Both above concepts aim at providing us with a formal framework for the search of a family of independent efficient paths in a network. One may notice, taking into account axioms (H2) and (H3) of item 2.2, that restricting ourselves to elementary paths from x to y in the above definitions doesn't really means the introduction of any additional constraint. Besides, the study of some examples will allow us to notice that any solution of Problem 2 is also a solution of Problem 1 and that solving Problem 2 produces more information than solving Problem 1. For this, most of our work will focus on the study of Problem 2.

3 Examples and Comments

3.1 A Simple Mathematical Example

Let us consider some network G = (X,E), some alphabet A, and let us suppose that any arc in E is endowed with some symbol in A, in such a way that:

- if the arcs [x,y] and [y,z] exist and are both labelled with the same symbol u, then the arc [x,z] exists and is also labelled with u;

Then to any path γ in G corresponds some word $m(\gamma)$ in the word set M(A) defined on A.

We obtain a strategic triple (R,O,L) by setting:

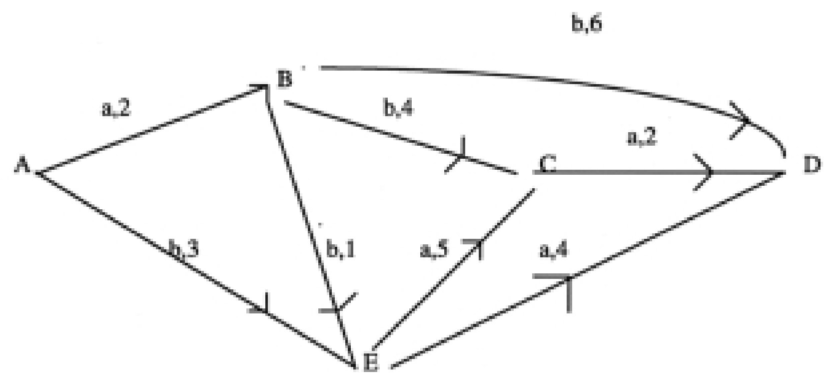
- γ R γ' iff the words $m(\gamma)$ and $m(\gamma')$ are the same;

- γ O γ' iff the word $m(\gamma)$ is a subword of the word $m(\gamma')$;

and by defining L as being the set of all alternated pathes, i.e, all pathes which don't contain two consecutive arcs labelled with the same symbol.

In case some positive length function d is defined on E, we see that d is compatible with (R,O,L) iff for any pair of arcs [x,y] and [y,z] endowed with the same label, we have d([x,z]) $\leq$ d([x,y]) + d([y,z]).

We may consider the following small example:



Vertex set X = { A, B, C, D, E};

Arc set E = { [A,B], [A,E], [B,E], [B,C], [E,C], [C,D], [E,D], [B,D]), those arcs respectively endowed with the labels a,b,b,b, a, a, a, b and with the lengthes 2, 3, 1, 4, 5, 2, 4, 6.

Paths (A,B, C,D) and (A,B,E,D) are here R-equivalent, and are both O-dominated by path (A,E,D).

We also see here how the projection operator $\prod$ works: to the path with support (A,E,C,D) it makes correspond the path with support (A,E,D).

If k = 2, and if x = A, y = D, we see that Problem 1 admits exactly two symetric solutions:

- γ_1 = (A,E,D) and γ_2 = (A,B,D);

- γ_1 = (A,B,D), γ_2 = (A,E,D).

Problem 2 admits exactly one solution: γ_1 = (A,E,D) and γ_2 = (A,B,D).

3.2 An Example Related to Transportation Systems

Let us now look at an example which will be used several times during the rest of the paper.

We consider a network G = (X,E), 2 vertices x_o and y_o in X, and a set U of symbolic variables which provides some labelling of the arcs of G. G is supposed to represent some transportation infrastructure and any symbolic identifier u in U may be considered as the identifier of some speed variable. More precisely, we suppose that if an arc e = [x,y] is given and if the value of U(e) is known, then the time required to go from x to y is given by an expression:

$t(e, U) = \alpha(e) + \lambda(e).U(e)$

where $\alpha(e)$ and $\lambda(e)$ are positive coefficients associated with e.

That means that if γ is some path, made with consecutive arcs $e_1...e_n$, the time $\sum_{i=1..n} t(e_i, w)$ required to run along γ appears as a formal U-expression: $\Lambda * (\gamma) = \Delta(\gamma) + \sum_{u \in U} \Lambda(u, \gamma).u$.

which is linear in the decision vector U.

It may occur that we need to handle the shortest path problem bewteen x_o and y_o without knowing what will be the exact value of the vector U, at the time when the run from x_o to y_o will effectively take place. Such a situation may come for instance from the fact that the effective running decision needs to be a real time decision, which makes impossible full information gathering and shortest path computing, or also from the fact that the shortest path handling process may be part of a global design process aimed the determination of the value of U.

In such a case, we try to get, not a single optimal path, but a whole path family likely to produce, at the time the running decision will be taken, and whatever be then the exact value of w, at least one good strategy.

In a natural way, we consider here that 2 pathes γ_1 and γ_2 are R-equivalent if both formal expressions $\Lambda * (\gamma_1)$ and $\Lambda * (\gamma_2)$ are the same and that path γ_1 O-dominates γ_2 if $\Lambda * (\gamma_1)$ is never larger that $\Lambda * (\gamma_2)$, whatever be the feasible value of the vector

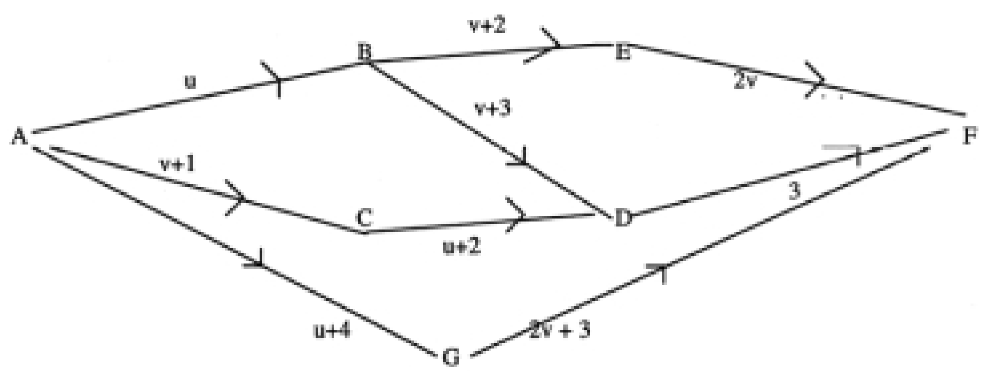
U. We define L as being the set P*(G).

Let us consider for instance the following example:

X = {A,B,C,D,E, F, G};

U = {u, v};

E = {[A,B], [A,C], [B,E], [E,F], [C,D], [D,F], [A,G], [G,F], [B,D]}, those arcs e being respectively endowed by the following linear affine expressions t(e,U): u, v+1, v+2, 2v, u+2, 3, u+4, 2v + 3, v+3.



We see that paths (A,B,D,F) and (A,C,D,F) are R-equivalent and that they both O-dominate path (A,G,F).

Let us suppose k = 2, x_o = A, y_o = F.

Then, Problem 1 admits here 3 solutions, modulo R-equivalence:

First solution: γ_1 = path (A,B,D,F) and γ_2 = path (A,G,F);

Second solution: γ_1 = path (A,B,D,F) and γ_2 = path (A,B,E,F);

Third solution: γ_2 = path (A,B,D,F) and γ_1 = path (A,B,E,F);

Let us suppose now that the mean value of u is 5 and that the mean value of v is 0.5. Thus, we may associate with any arc e in E, some quantity d(e) which is the mean value of t(e,U).

Then we see that the length function d defined this way is compatible with the strategic triple (R,O,L) and that Problem 2 has exactly one solution (modulo R-equivalence):

γ_1 = path (A,B,E, F); γ_2 = path (A,B,D,F).

We also see that if k = 3, then Problem 2 doesn't admit any solution.

We see on this example that Problem 2 is much more constraining than Problem 1, and that answering it is going to provide us with much more information about what will be a good strategy for moving from some origin to some destination, according to the variations of the state of our system.

3.3 An Example Related to Reliability

Once again we consider some network G = (X,E), and some length function d defined on the arc set E of G.

We consider also a set U of independent {0,1} random variables and we suppose that at any instant, every variable in U commands the access to several arcs of E.

Thus to any path γ in G corresponds a U-monomial $m(\gamma)$ which provides the probability that γ may be used at a given time, and which is such that the degree in $m(\gamma)$ of any variable u in U is 0 or 1.

At any time, we may be required to connect two given vertices x_o and y_o with some shortest valid path. Since performing the associated computing under real time constraints may eventually induce some trouble, we prealably extract some path family, which are the shortest possible for the length function d, and which are independent with regard to the way the variables of U behave.

In order to do it, we set:

- L = all the paths in G;

- γ R γ' iff $m(\gamma) = m(\gamma')$;

- γ O γ' iff $m(\gamma) \geq m(\gamma')$ whatever be the values taken by the variables of U.

Thus, we are typically in the situation of looking (Problem 2) for some k minimal strongly independent path family from x_o to y_o associated with the strategic triple (R,O,L) and with the length function d, k being some conveniently choosen integer.

Once this family has been computed, we handle our problem by picking up inside this family, every time it is necessary, the shortest currently valid path.

4 The Main Algorithms

In all this section, we are going to consider some network G = (X,E), some origin/destination pair (x_o, y_o), some strategic triple (R,O,L), some positive length function d defined on the arc set E and compatible with (R,O,L), and some integer k.

We first do some few remarks:

- Because of its compatibility with the equivalence relation R, the partial ordering O turns itself into a partial ordering O* defined on the Strategy Set ST(G,R) = P(G)*/R. A solution of Problem 1 corresponds to some subset S of ST(G,R), such that:

- if s is in S, then any s' such that s' O* s is also in S.

Clearly, such a subset doesn't need to be unique, as we noticed in the previous section.

A solution of Problem 2 corresponds to some subset S of ST(G,R) which is made only with elements which are minimal for the partial ordering O*.

Thus, any solution of Problem 2 is also a solution of Problem 1.

- Let MIN(G,R) be the subset of ST(G,R) defined by all the elements which are minimal for O*.

For any strategy s in ST(G,R), we may define the following quantity:

H(s) = Inf d*(γ), for all the paths γ being in the R-equivalence associated with s;

Thus, solving Problem 2 simply means finding some subset S of MIN(G,R) which yields the k smallest possible values H(s), s in MIN(G,R).

It follows that, if it exists, any solution $\gamma_1 ... \gamma_k$ of Problem 2 is unique in the sense that:

- the sequence of values $d * (\gamma_i), i = 1...k$, is completely determined;

- for any path γ from x_o to y_o such that $d*(\gamma) < Supd*(\gamma_i)$, i = 1..k, and such that the equivalence class of γ is in MIN(P(G)*/R), there must exist exactly one value i such that γ is R-equivalent to γ_i.

These remarks justify the fact that most of the next sections will be devoted to the study of Problem 2.

4.1 Algorithm STRATPATH2 for the Problem 2

In order to present this algorithm, we introduce some additional notations:

- Relation $O_<$: if γ and γ' in P(G)* are such that either γ O γ' or [(γ R γ') and d*(γ)< d*(γ')] then we set γ $O_<$ γ';

- Relation $R_=$: if γ and γ' in P(G)* are such that γ R γ' and d*(γ) = d*(γ') then we set γ R= γ'.

Algorithm STRATPATH2 is designed as an extension of Dijkstra Algorithm for the shortest path problem in positive networks. It is structured into one main loop, in such a way that at every entry into this main loop the situation is the following one:

- For some pairs (x,i), x in X, i in N, an elementary path $\Gamma(x,i)$ from x_o to x has been computed together with some length $\prod(x,i)$. For any such a pair (x,i) with this property, and for any index j in 1...i, the path $\Gamma(x,j)$ and the length $\prod(x,j)$ also exist and the paths $\Gamma(x,j)$, j = 1...i, are strongly (O,R)-independent. For any given vertex x, the sequence of the existing lengthes $\prod(x,i)$ is increasing. Besides, some of these pairs (x,i) have been marked, and if some pair (x,i) is marked, then also is any pair (x,j) with j in 1...i.

Then the following actions are performed:

- Some non marked pair (x,i) is selected, in such a way that $\Gamma(x,i)$ is minimal for the relation $O_<$ among the paths associated with non marked pairs; This pair is called the pivot pair and becomes marked.

- For any arc [x,y] in E, such that the path $\Gamma(x,i)$ + {[x,y]} is in L and such that no path $\Gamma(y,j)$ satisfies: $\Gamma(y,j)$ $O_<$ ($\Gamma(x,i)$ + {[x,y]})

or

($\Gamma(y,j)$ $R_=$ ($\Gamma(x,i)$ + {[x,y]})

then we clean up the sequence $(\Gamma(y,u), \prod(y,u))$ for u $\geq$ 1 in such a way that:

- it contains the pair ($\Gamma(x,i)$ + {[x,y]}, d*($\Gamma(x,i)$ + {[x,y]}));

- it is ordered through increasing values $\prod(y,u)$;

- the paths $\Gamma(y,u)$, u $\geq$ 1, which appear in this finite sequence remain strongly (O,R)-independent.

The process stops when the pair (y_o,k) is marked or when no pivot pair (x,i) may be selected.

The algorithm STRAT-PATH2 may be formally described as follows:

Algorithm STRAT-PATH2

Input: A network G = (X,E), an origin/destination pair (x_o,y_o), a strategic triple (R, O, L), a length function d defined on E and compatible with (R,O,L) and an integer k.

Output: Failure or a k minimal strongly independent path family from x_o to y_o associated with (R,O,L) and d;

$\Gamma(x_o, 0)$:= Trivial path reduced to the vertex x_o;

Not Stop; No pair (x,i) is marked;

While Not Stop do

Set (Pivot, Index-Pivot) := some non marked pair (x,i) such that $\Gamma(x,i)$ exists and is $O_<$-minimal;

If (Pivot, Index-Pivot) doesnt exist then Stop (failure)

else

if (Pivot, Index-Pivot) = (y_o,k) then Stop (Success: the $\Gamma(y_o,i)$, i = 1..k, are the result)

else

Mark (Pivot, Index-Pivot);

For any arc [x,y] such that $\Gamma(x,i)$ + {[x,y]} is in L and such that no path $\Gamma(y,j)$ satisfies:

$\Gamma(y,j)$ $O_<$ ($\Gamma(x,i)$ + {[x,y]})

or

($\Gamma(y,j)$ $R_=$ ($\Gamma(x,i)$ + {[x,y]})

do

Set Γ := $\Gamma(x,i)$ + {[x,y]};

Remove from the finite list ($\Gamma(y,u)$, $\prod(y,u)$), u $\geq$ 1, all the pairs ($\Gamma(y,u)$, $\prod(y,u)$) such that: Γ $O_<$ $\Gamma(y,u)$;

Insert the pair (Γ, d*(Γ)) in the list ($\Gamma(y,u)$, $\prod(y,u)$), u $\geq$ 1, and adjust the values of the indices u $\geq$ 1 which appear in this list in such a way that:

- the indices u such that $\Gamma(y,u)$ is defined remain consecutive and ordered through increasing values of $\prod(y,v)$ = d*($\Gamma(y,v)$).

Theorem 1: The above algorithm STRAT-PATH2 computes a k minimal strongly independent path family from x_o to y_o associated with (R,O,L) and with the positive length function d, any time such a family exists.

Proof.

We first check by induction the following loop invariant:

- Every time a pair (x,i) is taken as pair (Pivot, Index-Pivot), the pathes $\Gamma(x,j)$, j = 1...i, are elementary paths and form a i minimal strongly independent path family from x_o to x associated with (R,O,L) and d. (I)

We may remark that, since R and O are stable under rigth side concatenation, since Nil (O $\vee$ R) γ for any γ in P*(G), and since d is positive, then any path $\Gamma(x,i)$ is elementary.

Let us suppose the assumption (I) to be wrong, and let us consider the pair (x,i) which corresponds to the first time, during the execution of the algorithm, when (I) above becomes wrong.

That means that there exists some elementary path Γ in L, from x_o to x, such that Γ $O_<$ $\Gamma(x,i)$ and also such that the pathes $\Gamma(x,j)$, j = 1...i-1, and Γ are strongly

(O,R)-independent.

We may suppose that Γ is minimal for $O_<$ with this property.

Then there exists some vertex y # x on Γ and some index j such that:

- the restriction $\Gamma_{x_o,y}$ is the path $\Gamma(y,j)$;

- z being the successor of y on Γ, no index l exists such that $\Gamma_{x_o,z} = \Gamma(z,l)$;

We may assume that, while taking into account the hypothesis on the $O_<$-minimality of Γ, we choosed Γ in such a way that the number of vertices on the subpath $\Gamma_{y,x}$ is the smallest possible.

Since d is positive, since Nil (O $\vee$ R) γ and since O and R are stable under rigth side concatenation, we have $\Gamma(y,j)\ O_<\ \Gamma(x,i)$.

Because of hypothesis (H5), $\Gamma_{x_o,z}$ is in L.

Since no index l exists such that $\Gamma_{x_o,z} = \Gamma(z,l)$, there exists an index u such that $\Gamma(z,u)\ (R_=\ O_<\ )\ \Gamma_{x_o,z}$.

Then, because O and R are stable under right side concatenation and because d is positive, we have:

$\Gamma(z,u) + \Gamma_{z,x}\ (R_=\ O_<\ )\ \Gamma$.

The concatenation $\Gamma(z,u) + \Gamma_{z,x}$ may not be in L. But, because of hypothesis (H4), there exists a path

$\Gamma' = \prod(\Gamma(z,u) + \Gamma_{z,x})$ with origin x_o and extremity x which is in L and which is such that:

- $\Gamma'\ O_<\ \Gamma$ or $\Gamma'\ R_=\ \Gamma$.

- Γ' is the concatenation of some prefix of $\Gamma(z,u)$ and of some path which doesn't contain more arcs than $\Gamma_{z,x}$.

Since $\Gamma(x,1)...\Gamma(x,i-1)$ and Γ are strongly (O,R)-independent, so are $\Gamma(x,1)...\Gamma(x,i-1)$ and Γ'.

If $\Gamma'\ O_<\ \Gamma$, then we get a contradiction on the $O_<$-minimality of Γ.

If Γ' $R_=$ Γ, then Γ' is the concatenation of a prefix P of $\Gamma_{x_o,z}$ and of some path S which doesn't contain more arcs than $\Gamma_{z,x}$. Since P can be writen $\Gamma(z1, i1)$ for some vertex z1 and some index i1, it follows that we get a contradiction on the fact that Γ and y were choosen in order to make the number of vertices in $\Gamma_{y,x}$ minimal.

The fact that the invariant (I) holds implies that if the algorithm succeeds in marking the pair (y_o,k) then what we get is really a k minimal strongly independent path family from x_o to y_o associated with (R,O,L) and d.

In case the algorithm fails in marking (y_o,k), we need to check that no k strongly independent path family from x_o to y_o associated with (R,O,L) exists. In order to do this, we may proceed by exactly the same way as we just did before. We consider the largest index i < k such that (y_o,i) has been marked by the algorithm, we denote it by i_o, and we suppose the existence of some path Γ, minimal for $O_<$, such that the paths $\Gamma(y_o, 1), ..., \Gamma(y_o, i_o)$, Γ form a (i_o+1) minimal strongly (O-R)-independent path family from x_o to y_o. Once again, we consider an arc [x,y] on Γ such that:

- there exists i such that the restriction $\Gamma_{x_o,x} = \Gamma(x, i)$;

- there doesn't exist j such that the restriction $\Gamma_{x_o,y}$ is equal to some path $\Gamma(y, j)$.

We suppose that Γ has also been choosen in such a way that the restriction Γ_{y,y_o} admits the less vertices possible.

The fact that $\Gamma_{x_o,y}$ is not equal to any path $\Gamma(y, j)$ means that there exists some index u such that $\Gamma(y, u)$ exists and satisfies:

- $\Gamma(y, u)$ $O_<$ $\Gamma_{x_o,y}$ or $\Gamma(y, u)$ $R_=$ $\Gamma_{x_o,y}$.

As we previously did, we notice that the concatenation of $\Gamma(y, u)$ and Γ_{y,y_o} may not be in L, but that its projection through $\prod$ provides us with a contradiction with the hypothesises which we made about Γ. □

4.2 Complexity of the Algorithm STRATPATH2

Our main problem comes here from the fact that before getting $\Gamma(y_o, k)$, STRATPATH2 will eventually compute many paths $\Gamma(x, k')$ with k' much larger than k. We can't forecast a bound for k' and we notice that most of these intermediary results won't be really usefull. So, we may, for a given parameter I ≥ k, rewrite STRATPATH2 in such a way that it never keeps into memory any path $\Gamma(x, k')$ with k' > I. Let us denote by STRATPATHBIS(I) the procedure obtained this way. Of course, STRATPATH2 and STRATPATHBIS(I) may yield different results.

So let us set the following definitions:

- Critical-Index(G,x_o,y_o,R,O,L,d) = the smallest value I such that STRATPATH2 and STRATPATHBIS(I) produce the same results from the input defined by the network G = (X,E), the length function d, the origin/destination pair of vertices x_o,y_o, and the strategic triple (R,O,L);

- Index(G,R,O,L) =Sup Critical-Index(G,x_o,y_o,R,O,L,d).

x_o, y_o in X, d positive length function on G

Let us also denote by $\tau(n)$ the complexity of testing that some path with n arcs is in L, and by $\gamma(n)$ the complexity of comparing, for the O and R relationships, 2 paths of P*(G) , with no more than n arcs.

At any iteration of the main loop of STRATPATHBIS(I) some "For" loop is executed. This loop consists in scanning, for the current pair (Pivot, Index-Pivot) pair, the set of the vertices y in X such that [Pivot,y] is in E, and in updating the pathes $\Gamma(y, j)$, j = 1..I.

Thus any execution of the STRAT-PATHBIS(I) process contains at most I. $\sum_{x \in X} d^+G(x) = I.$ |E| executions of this updating process, where $d^+G(x)$ denotes the outer degree in G of the vertex x.

For such a vertex y in X, such that [Pivot,y] is in E, this updating process may induce:

- checking the presence of some path in the set L;

- performing |I| comparizons of 2 paths of L for the O and R relationships.

We deduce that the complexity of STRAT-PATH(I) is no more than: O(I.|E|.(τ(|X|) + I.γ(|X|)).

We may state:

Proposition 1.

The complexity of STRAT-PATH2 is bounded by O(INDEX(G,O,R,L)2.(|E|.(τ(|X|) + γ(|X|)).

Unfortunately, the fact is that INDEX(G,O,R,L) may be arbitrarily large.

Example.

Let us consider some network G = (X,E), given together with a distance function d defined on the arc set E, with an alphabet A and with some A-labelling of the arcs of E as follows:

- A = $\{a_o...a_n\}$

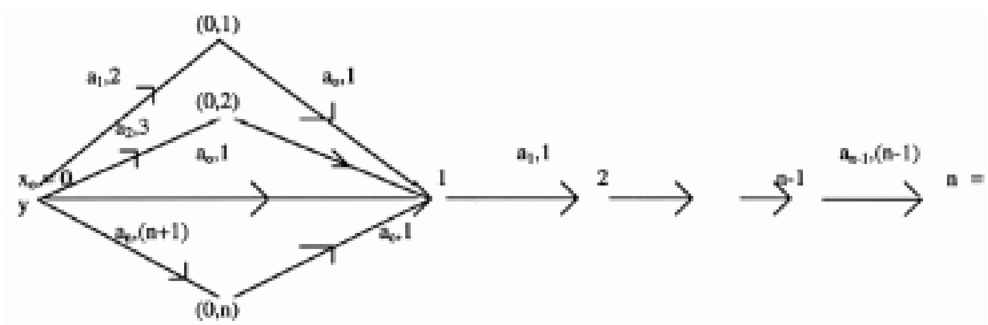
- X = { 0, 1.. n, (0,1)..(0,n)}

- E = { [i,i+1], i = 0..n-1} + { [0, (0,i)], i = 1..n, [(0,i), 1], i = 1..n};

- Any arc [i, i+1] has length 1 and label a_i;

- Any arc [0, (0,i)] has distance 1 + i and label a_i;

- Any arc [(0,i), 1], i = 1..n, has distance 1 and label a_o.



We suppose that: $x_o = 0$; y_o = n.

We consider the strategic triple (R,O,L) defined as follows:

- L = P*(G) = the set of all the possible pathes of G;

- O = the trivial empty relation;

- R: γ R γ' iff any symbol which appears on γ also appears in γ' and conversely.

Then, if k = 2, a solution of Problem2 for the above input is given by the pathes:

(x_o, 1,...,n) and (x_o, (0,n), 1,..,n) with respective lengthes n and 2n + 1.

In order to compute it through the algorithm STRAT-PATH2, we need to compute the path $\Gamma(1, n+1)$.

Still, it is possible in many cases to bound the value of Index(R,O,L).

In order to see how to do it, let us consider some network G = (X,E), some strategic triple (R,O,L) and some length function d defined on E.

We call degree of R in relation to L, denoted by D(G,R,L), the largest integer s such that there exist s+1 pathes $\gamma, \gamma_1 ... \gamma_s$ in G which satisfy:

- the pathes $\gamma_1 ... \gamma_s$ are pairwise R-independent and they share a same origin and a same extremity;

- the origin of γ is the common extremity of the γ_i, i = 1..s;

- for any i, j in 1..s, $\prod(\gamma_i + \gamma)$ R $\prod(\gamma_j + \gamma)$. (recall: $\prod$ is the projection operator of the hypothesis (H4)).

Theorem 2: If O is the empty partial ordering, then Index(G,O,R,L) D(G,R,L).

Proof.

Let us suppose the converse, that means let us suppose that some path $\Gamma(x, i) + \Gamma$ appears in the minimal strategy distinct k-path family from x_o to y_o computed by STRAT-PATH2, with i > k.D(G,R,L).

Obviously we get a contradiction since the pathes $\prod(\Gamma(x, i) + \Gamma)$, i = 1.. k.D(G,R,L) have their length no more than the length of $\Gamma(x, i) + \Gamma$, while it is possible to extract k paths from these k.D(G,R,L) pathes which are pairwise R-independent. □

We may provide some examples of values D(G,R,L):

- Let us suppose that G = (X,E), and (R,O,L) are like in the example of section 3.2. Then D(G,R,L) = 1.

- Let us suppose that G = (X,E) is such that any arc e in E is labelled with some symbol a(e) in an alphabet A, and let us define R and L as follows:

- L = P(G)*;

- γ R γ' iff γ globally involve the same symbols as γ' (independently of their multiplicity);

In such a case we have that D(G,R,L) = $2^{|A|}$.

- Let us consider some network G = (X,E) such that any arc e in E is labelled with some symbol s(e) in an alphabet A, in such a way that: if [x,y] and [y,z] are two arcs which are endowed with a same label s, then there is an arc [x,z] which is also labelled with s.

Let us also suppose that:

- L is the set of the alternated paths of G, that means the paths which don't contain any pair of consecutive arcs endowed with the same label;

- γ R γ' iff the A-words associated with γ and γ' are the same.

In such a case we have that D(G,R,L) = 2.

Unfortunately, it seems difficult to extend Theorem 2 to cases when the order relation O is non empty.

However, we may remark (verification left to the reader) that:

- If the network G = (X,E) and the strategic triple (R,O,L) satisfy the following implication:

for any vertices x,y, z in G, any paths γ_1, γ_2, γ such that:

or(γ) = ext(γ_1) = ext(γ_2) = y;

or(γ_1) = or(γ_2) = x; ext(γ) = z;

γ_1 and γ_2 are both O-minimal in $P_{x,y}(G)$;

γ_1 + γ is minimal in $P_{x,z}(G)$;

then γ_2 + γ is also minimal in $P_{x,z}(G)$.

then Index(G,R,O,L) = D(G,R,L).

4.3 Algorithm STRATPATH1 for the Problem 1

The basic ideas for solving Problem 1 are the same as those which we just previously presented. They may be summarized as follows:

Algorithm STRAT-PATH1

Input: The network G = (X,E), the origin/destination pair (x_o,y_o), the strategic triple (R, O, L) and the integer k.

Output: Failure or a k minimal independent path family from x_o to y_o associated with the strategic triple (R,O,L).

$\Gamma(x_o, 0)$:= Trivial path reduced to the vertex x_o;

Not Stop; No pair (x,i) is marked;

While Not Stop do

Set (Pivot, Index-Pivot) := some pair (x,i) such that $\Gamma(x, i)$ exists, is not marked, and is minimal for the relation O with these properties;

Mark (Pivot, Index-Pivot);

If (Pivot, Index-Pivot) doesn't exist then Stop (failure)

Else

If (Pivot, Index-Pivot) = (y_o,k) then Stop (Success: the $\Gamma(y_o, i)$, i = 1...k, yield the result)

Else

For any arc [x,y], such that $\Gamma(x, i)$ + {[x,y]} is in L and is not R-equivalent to any path

$\Gamma(y, j)$, j in N do

Set Γ := $\Gamma(x, i)$ + {[x,y]};

Let k(y) be the largest index j such that $\Gamma(y, j)$ exists;

Let uo be the largest index u such that: Not (($\Gamma(x, i)$ + {[x,y]}) O $\Gamma(y, u)$);

For j := k(y) downto u_o+1 do $\Gamma(x, j+1) := \Gamma(x, j)$;

$\Gamma(y, u_o + 1) : = \Gamma$;

Theorem 3: The above algorithm STRAT-PATH1 effectively computes a k minimal independent path family from x_o to y_o associated with the strategic triple (R,O,L), every time that such a family exists.

We leave the proof of this result to the reader, since it is very close to the proof of our next result.

5 Numerical Experiments

We present here two classes of experiments, both related to the example of section 3.2. The first one tests the quantity Index(O,R,L) which was introduced in the previous section 4.2. The second one aims at comparing the results produced by the STRATPATH2 algorithm with the results produced by a stochastic approach of the same problem of the search for independent efficient routing strategies in a network.

So we consider some network G = (X,E), some integer k, 2 vertices x_o and y_o, and we suppose that every arc e in E is endowed with 2 positive coefficients a(e) and l(e) and with some symbol U(e), belonging to some set U of symbolic variables.

Then, to every path $\gamma = \{e_o ... e_n\}$ in G, corresponds some symbolic U-expression:

$\Lambda * (\gamma) = \sum_{i=0...n} \alpha(e_i) + \lambda(e_i).U(e_i)$.

The relations R and O are defined by:

- γ R γ' iff the symbolic U-expressions $\Lambda * (\gamma)$ and $\Lambda * (\gamma')$ are the same;

- γ O γ' iff $\Lambda * (\gamma) < \Lambda * (\gamma')$ whatever be the feasible value taken by U;

L is defined as the set of all the possible pathes on G.

In order to test the quantity Index(O,R,L) on such an input, we use:

- two classes of networks with 100 vertices:

- "dense" networks: the presence of an arc in the arc set is determined through random sorting, with a probability p which may takes values 0.1, 0.2, 0.4;

- “sparse“ networks: the outer degree of any vertex may not to exceed some number q = 4, 6 or 8.

- a symbolic variable set U with 5 or 10 symbols;

- k = 4, 6 or 8, and values for the coefficients $\lambda(e)$ and a(e) randomly sorted between 1 and 5.

- origin/destination pairs x_o,y_o randomly sorted, while distinguishing “close“ pairs (x_o and y_o may be connected by a path with no more than 5 arcs) from the other pairs.

For any test, we compute the value Critical-Index = Critical-Index(G,O,R,L,x_o,y_o) defined in section 4.2.

Then for any sequence S of identically parameted experiments, we get the following quantities:

- Ind-Min = Inf Critical-Index, taken for all the experiments in S;

- Ind-Max =Sup Critical-Index, taken for all the experiments in S;

- Ind-Mean = Mean value of Critical-Index, taken for all the experiments in S;

- Double = Proportion of the experiments which provided a value of Critical-Index larger than 2k.

The results which we get may be summarized according to the following arrays:

“Dense“ and “Close“	Ind-Min	Ind-Max	Ind-Mean	Double
k = 4	4	10	5.4	48/50
k = 8	8	27	13.5	42/50

“Dense“ and “Far“	Ind-Min	Ind-Max	Ind-Mean	Double
k = 4	4	13	6.5	44/50
k = 8	8	34	16	32/50

“Sparse“ and “Close“	Ind-Min	Ind-Max	Ind-Mean	Double
k = 4	4	8	5.2	48/50
k = 8	8	22	10.4	44/50

“Sparse“ and “Far“	Ind-Min	Ind-Max	Ind-Mean	Double
k = 4	4	10	6.2	45/50
k = 8	8	28	12.6	39/50

We notice that:

- in most of the cases, Critical-Index is no more than 2k;

- increases in the "density" of G, or in the distance between origin x_o and destination y_o, or in the coefficient k tend to deteriorate the value of Critical-Index.

Our second class of experiments must be related to our initial discussion of the semantics of the problem (sections 1 and 2). Let us recall that we introduced this model of "strongly independent path family", in order to help us in computing paths which are at the same time efficient, and independent in relation to the possible variations of some state vector. But in the context of the above specific example, we could have tried to reach this goal through an other approach: we could have proceeded by randomly generating a sequence of values for the vector U, and by looking for a shortest path in G for the resulting positive length function. Namely, we could have applied the following GENER-PATH Procedure:

GENER-PATH(N, k):

Input: A network G = (X,E), such that any arc e in E is endowed with some formal linear affine expression t(e,U) = $\alpha(e)$ + $\lambda(e)$.u(e) , where $\alpha(e)$ and $\lambda(e)$ are positive coefficients and where u(e) is some symbolic variable,and 2 vertices x_o, y_o in X;

Output: Some family of strongly (O,R)-independent path;

For i := 1 to N do

Randomly generate a value for the "speed" vector U, according to some distribution σ, with mean value in 1;

Compute a shortest path γ_i from x_o to y_o in G, associated with the distance function which to any arc e in E makes correspond the value of t(e,U);

Extract from the paths computed this way, a set S of strategies (R-equivalence classes) ordered according to the number of representents of these strategies among the pathes γ_i, i = 1..N;

For i = 1 to k, select some path which represent the i th element of S.

Obviously, the paths generated this way form a strongly (O,R)-independent path family. This family may not be minimal. Conversely, given some path γ in a minimal independent path family, there may not exist any value of U such that γ is a shortest path for the length function t(.,U). For instance one may easily see that a strongly

(O,R)-independent 3 path family may be made with 3 paths γ_1, γ_2, γ_3 such that for any value of U, the following inequality holds:

$$\Lambda * (\gamma_1, U) + \Lambda * (\gamma_2, U) < 2.\Lambda * (\gamma_3, U),$$

and then check that such a relation forbids γ_3 from being a shortest path for any well-choosen value of U.

Still, one intuitively feels that both procedures GENER-PATH and STRATPATH2 aim at the same goal. In order to compare them, we use the same tests as in the previous part (with N = 100), while looking, for any test, to the number SHARE of elements shared by the lists produced by GENER-PATH and by STRAT-PATH2. Then, for any sequence S of identically parameted experiments, we compute the following quantities:

- Share-Min = Inf SHARE, taken for all the experiments in S;

- Share-Max =Sup SHARE, taken for all the experiments in S;

- Share-Mean = Mean value of SHARE, taken for all the experiments in S;

- Share = Proportion of the experiments which provided a value of SHARE equal to k.

Then the following arrays summarize our results:

"Dense" and "Close"	Share-Min	Share-Max	Share-Mean	Share
k = 4	2	4	3.2	15/50
k = 8	5	8	6.5	8/50

"Dense" and "Far"	Share-Min	Share-Max	Share-Mean	Share
k = 4	2	4	2.9	12/50
k = 8	5	8	5.8	7/50

"Sparse" and "Close"	Share-Min	Share-Max	Share-Mean	Share
k = 4	3	4	3.5	25/50
k = 8	5	8	6.8	10/50

"Sparse" and "Far"	Share-Min	Share-Max	Share-Mean	Share
k = 4	2	4	3	20/50
k = 8	5	8	6	8/50

So we see that in most of the cases, STRAT-PATH2 and GENER-PATH yield results very close to each other.

6 Application to a General Transportation Problem

As part of a collaboration between LIMOS and the National Center for Urban Transportation Research (CERTU-LYON), we were asked to propose some computer aided methods for the design of public transportation systems, while taking into account the time elasticities of the demands.

This problem, known to be difficult (see [4, 6, 13]), may be modelized with some network G = (X,E), which represents, in an aggregated way, some urban transit infrastructure. Any arc e= [x,y] in G is endowed with:

- a label s(e), which denotes the transportation mode (bus,...) associated with e;

- a length d(e), which denotes the time required for a s(e)-connection from x to y;

- a cost c(e).

Of course, several arcs, corresponding to different modes, may connect a same origin to a same extremity.

A special mode “ by walk ”, is denoted by 0 and is such that the partial network induced from G by all the arcs e with label s(e) = 0 is strongly connected. Intuitively, one may think into the arcs with label 0 as corresponding to existing streets, while the other arcs represent virtual connections, related to various transportation modes, which may eventually be proposed to the public.

Then we define a Route, associated with transportation mode s ? 0, as being some pair (γ, w), where γ is a circuit of G whose all arcs have label s and where w > 0 is the mean frequence, taken for some given standard time, of the s-vehicles on γ.

We define a Transportation System as being some family (Γ,w) = $\{(\gamma_1, w_1)...(\gamma_n, w_n)\}$ of routes.

Such a Transportation System has a cost, which is roughly evaluated as an expression:

Cost (Γ,w) = $\sum_{i=1..n} Cost(\Gamma_i, w_i) = \sum_{i=1..n} \sum_{e \in \gamma_i} c(e).w_i$.

If (Γ,w) is such a Transportation System and if x_o and y_o are two vertices in G, then we may evaluate through some shortest path computation the time T(Γ,w,x,y) necessary to go from x_o to y_o for some user who decides to use the service offered by the system (Γ,w). This time will take into account the expected waiting time $1/2w_i$ induced every time the user takes a new route (γ_i, w_i), i = 1...n.

Let us explain in detail the way this shortest path computation may be performed:

- we denote, for any vertices x and y in G, by $\prod(x,y)$ the shortest path distance from x to y defined by the length function d on the partial network induced by the arcs of G which have label 0.

- we denote, for any index i = 1...n, and any pair of vertices x,y in γ_i, by $\prod_i(x,y)$ the shortest path distance from x to y on the circuit γ_i.

- then we build the auxiliary labelled network H(G,Γ,w) = (X*,E*) as follows:

- X* = {(x,0), x in X} ∪ ($\cup_{i=1..n}$ { (x,i), x in γ_i})
- E* = {[(x,0), (y,0)], for any x,y in X} (arcs with label 0 and length $\prod(x_o, y_o)$)
∪
{ [(x,0), (y,i)], i in 1..n, x in X, y in γ_i} (arcs with label 0 and length $\prod(x_o, x) + 1/2w_i$)
∪
{ [(x,i), (y,j)], i in 1..n, j = 1..n, j? i, for x in γ_i, y in γ_j} (arcs with label i and length $\prod_i(x,y) + 1/2w_j$)
∪
{ [(x,i), (y,0)], x,y in γ_i, i in 1..n} (arcs with label i and length $\prod_i(x,y)$).

Computing T(Γ,w,x,y) comes by computing a shortest path from (x,0) to (y,0) in H(G,Γ,w). One easily checks that this search may be restricted to alternated paths, i.e to paths with no consecutive arcs with the same label.

Therefore, our main problem consists in designing some Transportation System (Γ, w) while taking into account the following criterions:

- the cost of the system must be the smallest possible (Economical criterion);

- the system must be used by the largest number of users possible, and must allow to connect any pair of vertices in the network G in the fastest way possible (Service criterion);

and the following constraints:

- the users always adopt shortest path strategies (Wardrop Principle) [20];

- the frequency of any route Γ_i must be at least equal to some threshold and such that any demand for a trip along this route can be satisfied.

The demands for the service are expressed inside this model by some family of origin/destination pairs (o_j, d_j), j = 1..m, and for any such a pair, by a coefficient

D_j*. Then the effective demand for going from o_j to d_j through some transportation system (Γ,w) comes as follows:

$D_j(\Gamma, w) = D_j * .\Phi(\prod(o_j, d_j), T(\Gamma, w, o_j, d_j))$.

where Φ is some function from R^{+2} to the interval [0,1], which reflects the elasticities of the user's demands with respect to the connection times offered by the system. $\Phi(u, t)$ is increasing in u and decreasing in t.

A formalization of this problem comes as follows:

Problem DESIGN:

Find a Transportation System (Γ,w) which minimizes:

Cost(Γ,w) - $\sum_{j \in 1..m} pj.D_j(\Gamma, w) + k.\sum_{j \in 1..m} D_j * .T(\Gamma, w, o_j, d_j)$.

with the constraints:

- If wi # 0 then $w_i \geq$ Z where Z is some fixed threshold;

- For any arc e in E and any route Γ_i which contains e, $w_i \geq \sum_{j \in 1..m} D_j(\Gamma, w)$ where I(e,i)(w) denotes the set of pairs (o_j,d_j) such that the shortest path associated with T(Γ,w,o_j,d_j) involves the arc e of the route γ_i.

N.B: p = (p_j , j in 1..m), is here a price vector, and k is some multicriterion coefficient.

This problem is difficult to handle, mainly since the expression of both the constraints and the objective function involves shortest paths sub-problems. In case we adopt a local improvement approach to manage it, we need to solve these subproblems every time the current pair (Γ,w) is modified.

The heuristic scheme which we propose in order to deal with the problem DESIGN comes as follows:

Initialize Γ; Not Stop;

While Not Stop do

Solve the above problem DESIGN, while considering that Γ is fixed; (*)

Update Stop; If Not Stop then generate an other circuit γ and insert it into Γ; (**)

Instruction (**) is crucial and can be implemented in several way. Two approaches may be tried:

- the first one consists in first defining, through some aggregation mechanism, some extended demand function between any pair (x,y) of vertices of G, and next in looking for a pair (x,y) which maximizes this demand. This approach yields diameter like routes, i.e routes which connect some origin to some destination through some kind of shortest path.

- The second one consists, for any circuit γ of G, in introducing an approximation V(γ,Γ,w) of the surplus of demand which would be created by the insertion of γ, (with some standard frequency 1 or + 8), into the system (Γ,w). This approximation is such that computing it doesn't require computing any (o_i,d_i)-shortest path. Then, some hill-climbing scheme may be applied, based upon homotopy neighbourhood (see [15]), which enables us to make appear routes with non predetermined structures (radial, circular, ...).

Implementation of the (*) instruction: application of the strongly (O-R) independent path family concept.

Implementing instruction (*), i.e solving the problem DESIGN(Γ) obtained from DESIGN while considering that the route set Γ is fixed, remains difficult, also because of the shortest path assumption. In order to avoid dealing with these shortest paths, we apply the previously described techniques. So, for the current circuit family Γ, for some standard value of w (w = 1), for some integer k (k = 5, k = 7 or k = 9), and for any pair (o_j, d_j), j = 1..m, we prealably compute some k minimal strongly independent path family $\gamma_j = \{\gamma_j$,l, l = 1..k$\}$ from o_j to d_j on the network H(G,Γ,w), with the strategic triple (R,O,L) defined as follows:

- L is the set of the alternated paths with no more than 4 arcs (3 changes of routes);

- γ R γ' iff the formal expressions $\Lambda * (\gamma, w)$ and $\Lambda * (\gamma', w)$ which provides the expressions of the length of γ and γ' as functions of w are the same;

- γ O γ' iff $\Lambda * (\gamma, w) < \Lambda * (\gamma', w)$ whatever be the values of w.

We also impose the presence in this family of the unimodal shortest path "by walk" between o_j and d_j.

Remark:Imposing a restriction on the number of arcs in any path in L keep us from getting the hypothesises of Theorem 1. Still the algorithm STRATPATH2 may be used as an heuristic, and allows us to get pathes which are really pairwise independent in the sense that they scarcely use the same routes.

For any pair (j,l), j = 1..m, l = 1..k, we denote by $T_{j,l}(w)$ the time required to connect oj to dj through the path $\gamma_{j,l}$, computed for the current value of w.

Then DESIGN(Γ) can be rewriten as follows:

Problem DESIGN(G):

Find w $\geq$ 0 in order to minimize:

$\sum_{i=1..n} Cost(\Gamma_i, w_i) - \sum_{j=1..m} p_j.D_j(\Gamma, w) + \sum_{j=1..m} D_j * .Inf_{l=1..k} T_{j,l}(w)$.

with the constraints:

- If wi # 0 then wi $\geq$ Z;

- For any arc e in E and any route Γ_i which contains e, $w_i \geq \sum_{j \in I(e,i)(w)} D_j(\Gamma, w)$, where I(e,i)(w) denotes the set of pairs (o_j,d_j) such that the path $\gamma_{j,l*(j,w)}(w)$ associated with the index l*(j,w) which minimizes $T_{j,l}(w)$, l = 1..k, contains e.

At the end, the algorithmic scheme applied to solve DESIGN(Γ) comes as follows:

Resolution method for DESIGN(Γ):

Not Stop; For any j = 1..m do select l(j) in 1..k(j);

While Not Stop do

Not Stop1;

While Not Stop1 do

Stop1;

Solve (through Lagrangean relaxation) DESIGN(Γ) while imposing, for any j = 1..m, the transportation demand from o_j to d_j to be routed through the path γ_j,l(j);

Let w be the current result produced by this resolution process;

For j = 1..m do

if $T_{j,l}(j)(w)$ # $T_{j,l*(j,w)}(w) = Inf_{l=1..k(j)} T_{j,l}(w)$, then set l(j) := l*(j,w) and Not Stop1;

Search j in 1..m and l1 in 1..k such that replacing l(j) by l1 allows to make decrease some w_i, i = 1..n and thus to make decrease the quantity (path redirection):

$\sum_{i \in 1..n} Cost(w_i) + \sum_{j \in 1..m, l=1..k} D_j * .T_{j,l(j)}(w)$

If such a pair (j, l1) exists then set l(j) := l1 else Stop;

References

[1]. F.BENDALI, A.QUILLIOT: "Ré seaux stochastiques"; RAIRO-RO 24, 2, p 167-190 (1990).

[2].T.H.CORMEN, C.H.LEISERSON, R.L.RIVEST: "Introduction to algorithms"; MIT Press, Cambridge, Mass (1980).

[3]. E.DIJKSTRA: "A note with two problems in connection with graphs"; Numerische Mathematik I, p 269-271 (1959).

[4].R.DIONNE , M.FLORIAN: "Exact and approximate algorithms for optimal network design"; Network 9, p 37-59, (1979).

[5].A.FARLEY: "Minimum broadcast networks"; Networks 10, p 59-70 (1980).

[6].M.FLORIAN: "No linear cost models in transportation analysis"; Math Prog Study 26, p 167-196, (1986).

[7].P.FRAIGNIAUD, E.LAZARD: "Methods and problems of communication in usual networks"; Disc Applied Maths 53, p 79-133 (1994).

[8].M.GONDRAN, M.MINOUX: "Graphes et algorithmes"; Ed Eyrolles (1979).

[9].R.M.KARP, J.B.ORLIN: "Parametric shortest path algorithms with application to cyclic staffing"; Disc Applied Math 3, p 37-45 (1981).

[10].J.LAURIERE: "Intelligence Artificielle"; Eyrolles (1987).

[11].E.LAWLER: "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem"; Management Science 18, 7, p 401-405 (1972).

[12].E.MINIEKA: “Optimization algorithms for networks and graphs“; Marcel Dekker Inc (1978).

[13].M.MINOUX: “Network synthesis and optimum network design problems: models, solution methods and applications“; Network 19, p 313-360 (1989).

[14]. N.NILSSON: “Problem solving methods in A.I“; Mac Graw Hill (1971).

[15]. A.QUILLIOT: “A retraction problem in graph theory“; Disc Math 54, p 61-71 (1985).

[16]. A.QUILLIOT: “Algorithmes de cheminements pour des ré seaux d'actions à effets non dé terministes“; Matematicas Aplicadas 12, p 25-44 (1991).

[17]. M.SAKAROVITCH: “Chemins, flots, ordonnancements dans les ré seaux“; Hermann, Paris (1984).

[18].M.SAKAROVITCH: “The k shortest routes and k-shortest chains in a graph“; Report ORC 66-32, Operation Research Center, University of California, Berkeley, (1966).

[19].D.R.SHIER: “On algorithms for finding the k shortest pathes in a network“; Networks 9, 3, p 195-214 (1979).

[20].J.WARDROP: “Some theoretical aspects of road traffic research“; Proc of the Institute of Civil Engineering, II, 1, p 325-378, (1952).

[21].B.YAGED: “Minimum cost routing for dynamic network models“; Network 3, p 315-331, (1973).