BICRITERION PATH PROBLEMS

Pierre HANSEN

Faculté Universitaire Catholique de Mons, Belgium, and
Institut d'Economie Scientifique et de Gestion, Lille, France.

ABSTRACT

Bicriterion path problems in directed graphs are systematically studied. A series of ten problems are defined and their computational complexity examined. Algorithms are provided for some of them, including polynomial algorithms for the MAXMIN-MAXMIN problem and the MINSUM-MAXMIN problem, and a pseudo-polynomial exact algorithm as well as a fully polynomial approximation scheme for the MINSUM-MINSUM problem.

1. INTRODUCTION

Many problems concerning the design and the efficient use of transportation, transmission and communication networks can be expressed as bicriterion path problems. For instance, the choice of a highway route, studied by J.C. Marchet and J. Siskos [12], involves both ecological and economic criteria. Determination of a cost and an index of environmental damage for each feasible highway segment and use of a bicriterion path algorithm (for the MINSUM-MINSUM problem, see below) will yield a set of efficient routes. Such a set constitutes a useful basis for discussion between decision-makers concerned mostly with criteria of the first and of the second kind. When using a transmission or a communication network, both the cost and the reliability of the lines employed are of importance. The selection of the lines forming a path joining two given locations is again a bicriterion path problem of the MINSUM-MINSUM type. Alternately, capacities and reliabilities of the lines may be considered, as done by R.C. Amara, H. Lindgren and M. Pollack [1] and by I.T. Frisch [7]. This yields a MINSUM-MAXMIN type problem, see below. Considering the reliability of a path and the threshold reliability of its lines gives another problem of the same type, see H. Frank and I.T. Frisch [6]. Finally, selection of a route in a transportation network taking capacity and distance or time as criteria also reduces to a problem of that type, see H.M. Moore [14]. Let

us note also that the constrained path problems studied by E. Lawler [11], Y.P. Aneja and K.P.K. Nair [2] and by N. Megiddo [13] are subcases of the MINSUM-MINSUM bicriterion path problem. Many other applications may be thought of.

The problems mentionned above have rarely been considered explicitly as bicriterion path problems; to the author's knowledge, no systematic study of that class of problems has yet been published. The purpose of the present paper is to initiate such a study. A series of ten bicriterion path problems are defined in the next section. Their computational complexity is studied in section 3. Polynomial algorithms are provided in section 4 for two easy problems called MAXMIN-MAXMIN and MINSUM-MAXMIN, to which most of the other easy problems of the list may be reduced. A pseudo-polynomial exact algorithm as well as a fully polynomial approximation scheme are proposed for the MINSUM-MINSUM problem in section 5. Some conclusions are drawn in section 6.

2. BICRITERION PATH PROBLEMS

Let $G = (X,U)$ denote a *directed graph*(1), X its set of *vertices* and U its set of *arcs*. Associate to each arc $(x_j,x_k) \in U$ two non-negative real numbers d_{jk} and c_{jk}. For convenience, d_{jk} and c_{jk} will sometimes be called the *length* and the *cost* of arc (x_j,x_k) in the following. Let L denote a *directed path* $(x_1,x_j,x_k,\ldots,x_m,x_n)$ joining the *initial vertex* x_1 to the *terminal vertex* x_n in G; let $\mathcal{L}$ denote the set of all such paths in G. Further, let L' denote a path L which is *elementary*, i.e. that does not contain twice the same vertex, and let $\mathcal{L}'$ denote the set of all such paths in G.

The following *single criterion path problems* may then be defined : MINSUM. Determine a shortest path between x_1 and x_n in G : $\underset{L\in\mathcal{L}}{\text{Min}} \sum_{(x_j,x_k)\in L} d_{jk}$.

MAXSUM. Determine a longest (elementary) path between x_1 and x_n in G : $\underset{L\in\mathcal{L}}{\text{Max}} \sum_{(x_j,x_k)\in L} d_{jk}$ (or $\underset{L'\in\mathcal{L}'}{\text{Max}} \sum_{(x_j,x_k)\in L'} d_{jk}$).

MINMAX. Determine a path between x_1 and x_n in G for which the length of the longest arc is smallest : $\underset{L\in\mathcal{L}}{\text{Min}} \underset{(x_j,x_k)\in L}{\text{Max}} d_{jk}$.

(1) The graph theoretical terminology used in this paper is that of C. Berge [3] to which the reader is referred for undefined terms.

MAXMIN. Determine a path between x_1 and x_n in G for which the length of the shortest arc is greatest : $\underset{L\in\mathcal{L}}{\text{Max}} \ \underset{(x_j,x_k)\in L}{\text{Min}} \ d_{jk}$.

MINPRODUCT. Determine a path between x_1 and x_n in G for which the product of the arcs lengths is smallest : $\underset{L\in\mathcal{L}}{\text{Min}} \ \underset{(x_j,x_k)\in L}{\Pi} \ d_{jk}$.

MAXPRODUCT. Determine a path between x_1 and x_n in G for which the product of the arcs lengths is greatest : $\underset{L\in\mathcal{L}}{\text{Max}} \ \underset{(x_j,x_k)\in L}{\Pi} \ d_{jk}$.

The MINSUM, MAXSUM, MAXMIN and MAXPRODUCT problems are usually referred to under the names of *shortest* (or *least-cost*) *path problem, longest path problem, maximum capacity path problem* and *maximum reliability path problem* respectively (the $d_{jk} \in [0,1]$ in the last case).

Note that the MINMAX and MAXMIN problems are equivalent in the sense that substituting lengths $d'_{jk} = \underset{(x_j,x_k)\in U}{\text{Max}} \ d_{jk} - d_{jk}$ for the original lengths d_{jk} transforms one problem into the other. Note also that the results of these problems are invariant under a monotone transformation of the d_{jk}. Taking $d'_{jk} = \log d_{jk}$ transforms into the MINSUM problem the MINPRODUCT problem if all $d_{jk} \in [0,1]$ and the MAXPRODUCT problem if all $d_{jk} \in [1,\infty[$. Transformations to the MAXSUM problem are obtained similarly. It is well known that the MINSUM, MAXMIN (or MINMAX) and MAXPRODUCT with all $d_{jk} \in [0,1]$ problems are *easy*, i.e. can be solved by algorithms requiring in the worst case a number of operations bounded by a polynomial in the length of the problem's characteristics (or, more precisely, in the length of the encoding of the problem's input). The common algebraïc structure underlying the MINSUM, MAXMIN and MINPRODUCT problems - that of a *semiring* - has been studied by many authors, see e.g. A. Wongseelashote [17] and allows to formulate unified polynomial algorithms for them all.

It can be shown that the MAXSUM problem is easy if the path sought for is not required to be elementary, which implies it is easy when G is *circuit-free*. Otherwise this problem is *NP-hard* and little is known about how to solve it (see W.W. Hardgrave and G.L. Nemhauser [10], S.N.N. Pandit [15]).

Let us now define a series of bicriterion path problems by combining pairwise the objective functions considered above. Given a pair of objective functions, let us call a path $L \in \mathcal{L}$ (or $L' \in \mathcal{L}'$) *efficient* if

and only if no other path $L^{*} \in \mathcal{L}$ (or $L^{*'} \in \mathcal{L}'$) has a better value for one criterion and a not worse value for the other one. For instance, a path $L \in \mathcal{L}$ will be efficient for the MINSUM-MINSUM problem, where both criteria are of the first type listed, if and only if there exists no path $L^{*} \in \mathcal{L}$ with smaller length and not larger cost or with not greater length and smaller cost. A path which is not efficient is thus *dominated* by at least one efficient path. Let us say that two efficient paths are *equivalent* if and only if their values agree for both criteria. Further, let us define a *complete set* $S \subset \mathcal{L}$ of efficient paths as a set such that any path $L^{*} \notin S$ is either dominated or equivalent to at least one efficient path $L \in S$. Let us call a complete set S *minimal* if and only if no two of its efficient paths are equivalent. Finally, let us write a general *bicriterion path problem* as follows : *Given a graph* $G = (X,U)$, *the length* d_{jk} *and costs* c_{jk} *of its arcs, an initial vertex* x_1, *a terminal vertex* x_n, *and two criteria, determine a minimal complete set* S *of efficient paths of* G. A list of ten bicriterion path problems is given in table 1; they will be studied in the remainder of the paper.

Problem number	Name	Criterion 1	Criterion 2
1	MAXMIN-MAXMIN	$\max_{L \in \mathcal{L}} \min_{(x_j,x_k) \in L} d_{jk}$	$\max_{L \in \mathcal{L}} \min_{(x_j,x_k) \in L} c_{jk}$
2	MINMAX-MAXMIN	$\min_{L \in \mathcal{L}} \max_{(x_j,x_k) \in L} d_{jk}$	$\max_{L \in \mathcal{L}} \min_{(x_j,x_k) \in L} c_{jk}$
3	MINMAX-MINMAX	$\min_{L \in \mathcal{L}} \max_{(x_j,x_k) \in L} d_{jk}$	$\min_{L \in \mathcal{L}} \max_{(x_j,x_k) \in L} c_{jk}$
4	MINSUM-MAXMIN	$\min_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} d_{jk}$	$\max_{L \in \mathcal{L}} \min_{(x_j,x_k) \in L} c_{jk}$
5	MINSUM-MINMAX	$\min_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} d_{jk}$	$\min_{L \in \mathcal{L}} \max_{(x_j,x_k) \in L} c_{jk}$
6	MINSUM-MINSUM	$\min_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} d_{jk}$	$\min_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} c_{jk}$
7	MAXSUM-MAXMIN	$\max_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} d_{jk}$	$\max_{L \in \mathcal{L}} \min_{(x_j,x_k) \in L} c_{jk}$
8	MAXSUM-MINMAX	$\max_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} d_{jk}$	$\min_{L \in \mathcal{L}} \max_{(x_j,x_k) \in L} c_{jk}$
9	MAXSUM-MINSUM	$\max_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} d_{jk}$	$\min_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} c_{jk}$
10	MAXSUM-MAXSUM	$\max_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} d_{jk}$	$\max_{L \in \mathcal{L}} \sum_{(x_j,x_k) \in L} c_{jk}$

Table 1. Ten bicriterion path problems.

Variants of the problem 7 to 10 are obtained by requiring that the path sought for be elementary, i.e. by considering $L' \in \mathcal{L}'$ instead of $L \in \mathcal{L}$.

3. COMPLEXITY OF BICRITERION PATH PROBLEMS

In this section, the computational complexity of the problems listed in table 1 is examined; for background the reader is referred to the excellent recent book of M. Garey and D.S. Johnson [8].

Using the transformations stated in the previous section it is easy to show that the first three problems are equivalent, as well as the fourth and fifth, and the seventh and eighth. Polynomial algorithms will be provided in the next section for the MAXMIN-MAXMIN and MINSUM-MAXMIN problems; although not done here, a polynomial algorithm could also be obtained for the MAXSUM-MAXMIN problem. So all these problems are easy.

Let us now show that the problems 6, 9 and 10 are, in worst case, intractable (this must not be confused with a proof of NP-hardness; some NP-hard problems could be solved in polynomial time in the very improbable case that $P = NP$).

Theorem 1. *The* MINSUM-MINSUM, MAXSUM-MINSUM *and* MAXSUM-MAXSUM *problems are, in worst case, intractable, i.e. require for some problems a number of operations which grows exponentially with these problem's characteristics.*

Proof. It is sufficient to show there exists for each problem a family of graphs for which the number of efficient paths in a minimal complete set grows exponentially with $n = |X|$. As listing these efficient paths requires an exponential number of operations no polynomial behaviour can then be expected in worst case from any possible algorithm. Therefore, consider a family of graphs G with an odd number n of vertices and $\frac{3}{2}(n-1)$ arcs defined as follows : each vertex x_i, with $i = 1, 3, \ldots, n-2$, is the origin of an arc (x_i, x_{i+2}) with $d_{i,i+2} = 2^{\frac{i-1}{2}}$, $c_{i,i+2} = 0$ and of an arc (x_i, x_{i+1}) with $d_{i,i+1} = 0$, $c_{i,i+1} = 2^{\frac{i-1}{2}}$; each vertex x_{i+1} is the origin of an arc (x_{i+1}, x_{i+2}) with $d_{i+1,i+2} = c_{i+1,i+2} = 0$; there are no other arcs (see figure 1). Let $t = 2^{\frac{n-1}{2}}$.

Clearly, any path from x_1 to x_n uses either the arc (x_i, x_{i+2}) or the

arcs (x_i,x_{i+1}) and (x_{i+1},x_{i+2}) but not both, for i = 1, 3, ..., n-2. Therefore a) there are t paths in G; b) the lengths of these paths are equal to the integers from 0 to t-1; c) the costs of these paths are equal to the integers from 0 to t-1; d) the sum of the length and of the cost of any path is equal to t-1. Hence, all paths of G are efficient paths for the MINSUM-MINSUM problem and constitute a minimal complete set. This proves the first part of theorem 1. The same result holds for the MAXSUM-MAXSUM problem and a similar proof is obtained for the MAXSUM-MINSUM problem, after modifying the weights of the arcs (x_i, x_{i+2}) for i = 1, 3,..., n-2 to have $d_{i,i+2} = c_{i,i+2} = 2^{\frac{i-1}{2}}$, all other weights being equal to 0.

Note that the graphs used in this proof are circuit-free; the variants of problems 9 and 10 are therefore also intractable. The variants of the problems 7 and 8 are NP-hard; they contain as subcase the MAXSUM problem for elementary paths which is NP-hard.

The complexity of the *constrained shortest path problem* $\underset{L\in\mathcal{L}}{\text{Min}} \sum_{(x_j,x_k)\in L} d_{jk}$ subject to $\sum_{(x_j,x_k)\in L} c_{jk} \leqslant C$ has been studied by N. Megiddo [13], according to M. Garey and D.S. Johnson [8], and proved to be NP-hard. That problem is a subcase of MINSUM-MINSUM. The following one is a subcase of MAXSUM-MINSUM : $\underset{L\in\mathcal{L}}{\text{Max}} \sum_{(x_j,x_k)\in L} d_{jk}$ subject to $\sum_{(x_j,x_k)\in L} c_{jk} \leqslant C$ (where C is a non-negative real number).

Theorem 2. *The constrained* MAXSUM *problem is* NP-*hard.*

Proof. The KNAPSACK problem, known to be NP-hard, may be written $\text{Max} \sum_{j=2}^{n-1} a_j v_j$ subject to $\sum_{j=2}^{n-1} b_j v_j \leqslant B$ and $v_j \in \{0,1\}$, j = 2, 3,..., n-1. Let us associate to this problem a graph G with n vertices and with arcs between each vertex and all vertices with higher indices. Choose weights $d_{1k} = c_{1k} = 0$ for k = 2, 3,..., n; $d_{jk} = a_j$ and $c_{jk} = b_j$ for j =

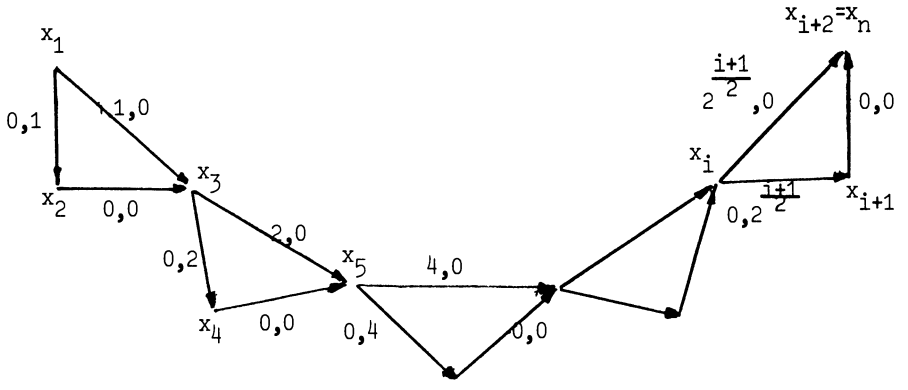

Figure 1. Graph illustrating theorem 1.

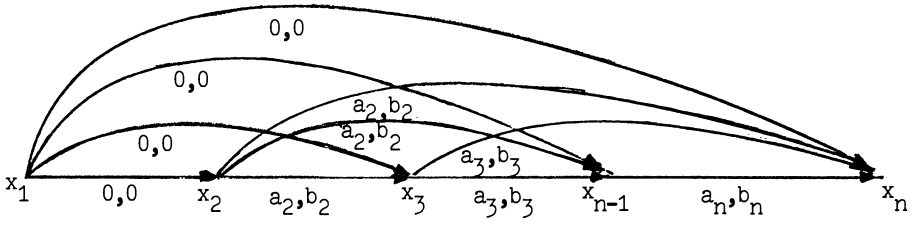

Figure 2. Graph illustrating theorem 2.

2, 3,..., n-1 and $k > j \in \{2, 3,..., n\}$; set C = B (see figure 2). Clearly, there is a 1-1 correspondance between paths from x_1 to x_n in G and Boolean vectors (v_j); moreover to any feasible vector corresponds a path satisfying the constraint and vice-versa. Thus, any polynomial algorithm for the constrained MAXSUM problem would imply the existence of a polynomial algorithm for KNAPSACK and hence that *P = NP*.

4. POLYNOMIAL ALGORITHMS

A path L is efficient for the MAXMIN-MAXMIN problem if and only if *L* contains no path L' for which the length of the shortest arc is greater and the cost of the cheapest arc not smaller, or for which the length of the shortest arc is not smaller and the cost of the cheapest arc is larger. Clearly a minimal complete set can contain at most $m = |U|$ efficient paths. Such a set can be obtained by the following algorithm.

ALGORITHM 1 : MAXMIN-MAXMIN.

a) *Initialisation.*

Read the data. Set $C_{min} = -1$. Proceed to b).

b) *Greatest minimum length path routine.*

b1) Set $T = \{1,2,...,n\}$, $\lambda_1 = \alpha$, $\lambda_j = 0$ for $j = 2, 3,..., n$.

b2) Compute $\overline{\lambda} = \text{Max } \{\lambda_j \mid j \in T\}$. If $\overline{\lambda} = 0$, stop, all the efficient paths sought for having been found. Otherwise select x_k such that $k = \text{Max } \{j \mid j \in T, \lambda_j = \overline{\lambda}\}$. If k = n, go to c). Otherwise proceed to b3).

b3) For all j such that $(x_k,x_j) \in U$, $j \in T$ and $c_{kj} > C_{min}$, if $\lambda_j < \min (\lambda_k,d_{kj})$ set $\lambda_j = \min (\lambda_k,d_{kj})$. Set $T := T \setminus \{k\}$ and go to b2).

c) *Largest minimum cost path routine.*

c1) Set $T = \{1,2,...,n\}$, $\mu_1 = \alpha$, $\mu_j = 0$ for $j = 2,3,...,n$, $p_j = 0$ for $j = 1, 2,..., n$.

c2) Compute $\overline{\mu} = \text{Max } \{\mu_j \mid j \in T\}$; select x_k such that $k = \text{Max } \{j \mid j \in T, \mu_j = \overline{\mu}\}$. If k = n, go to d). Otherwise proceed to c3).

c3) For all j such that $(x_k,x_j) \in U$, $j \in T$, $c_{kj} > C_{min}$, and $d_{kj} \geqslant \lambda_n$ if

$\mu_j < \min(\mu_k, c_{kj})$ set $\mu_j = \min(\mu_k, c_{kj})$ and $p_j = k$. Set $T = T \setminus \{k\}$ and go to c2).

d) *Output of an efficient path.*

Print λ_n, length of the shortest arc, and μ_n, cost of the cheapest arc of the path just found. Set $j = p_n$; recursively print j and set $j = p_j$ until $j = 0$. Set $C_{min} = \mu_n$ and return to b).

Theorem 3. Algorithm 1 yields a minimal complete set of efficient paths for the MAXMIN-MAXMIN *problem in* $O(m^2 \log n)$ *operations.*

Proof. Step b) and step c) are both isomorphic to the "maximum capacity path" version of Dykstra's algorithm [4]; they are applied to partial graphs of G such that all $c_{kj} > C_{min}$ and such that both all $d_{kj} \geq \lambda_n$ and all $c_{kj} > C_{min}$ respectively. When a *heap* is used to store the labels λ_j or μ_j with $j \in T$ an application of that algorithm to G requires $O(m \log n)$ operations in worst case and cannot take more for a partial graph of G. As C_{min} can take at most m distinct values, i.e. all those of the c_{kj}, Dykstra's algorithm must be applied 2 m times at most. Step d) requires $O(n)$ operations and occurs at most m times, so (under the reasonable assumption that $m \geq n$) the number of operations necessitated by algorithm 1 is $O(m^2 \log n)$ in worst case.

Note that other implementations are possible : listing the λ_j or μ_j completely yields an $O(mn^2)$ algorithm; if t denotes the largest of the numbers of distinct values of the c_{kj} and the d_{kj}, storing in a table the indices of the vertices with temporary labels and acceeding to that table through a *binary counting tree* (see [9]) yields an $O(m^2 \log t)$ algorithm.

Example. Algorithm 1 applied to the graph G of figure 3 yields two efficient paths : x_1, x_2, x_3, x_4, x_6 with $\lambda_6 = 4$ and $\mu_6 = 1$ and x_1, x_2, x_3, x_5, x_6, with $\lambda_6 = 2$ and $\mu_6 = 2$. The details of the resolution are summarized in table 2 (starred values correspond to $j \notin T$).

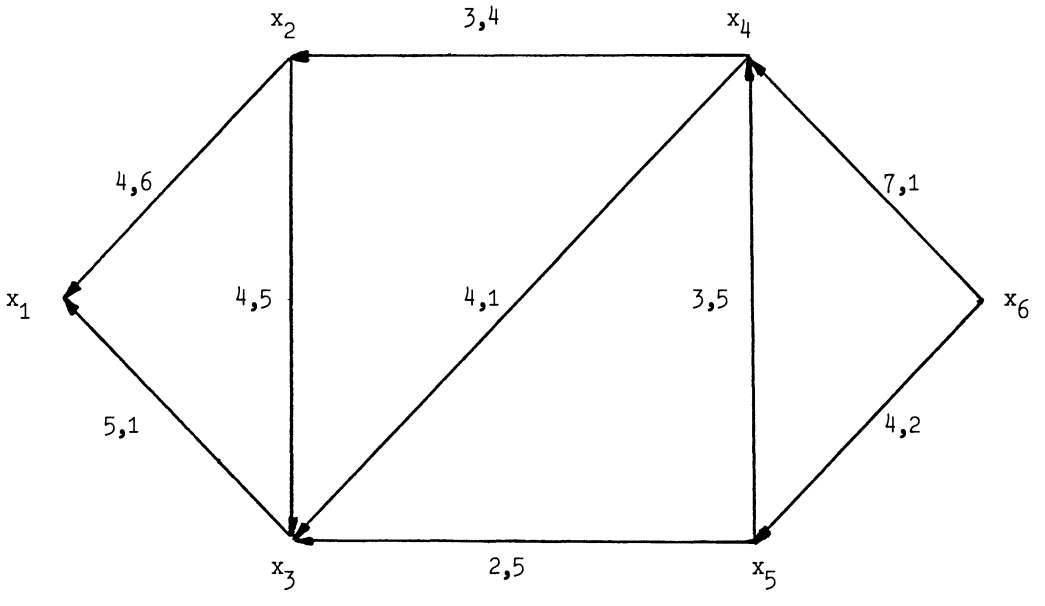

Figure 3. Example.

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
α^*	0	0	0	0	0
α^*	4	5^*	0	0	0
α^*	4	5^*	4^*	2	0
α^*	4	5^*	4^*	2	$\underline{4}^*$
α^*	0	0	0	0	0
α^*	4^*	0	0	0	0
α^*	4^*	4^*	3	0	0
α^*	4^*	4^*	3^*	2	0
α^*	4^*	4^*	3^*	2^*	2
α^*	4^*	4^*	3^*	2^*	$\underline{2}^*$
α^*	0	0	0	0	0
α^*	4^*	0	0	0	0
α^*	4^*	4^*	3	0	0
α^*	4^*	4^*	3^*	2	0
α^*	4^*	4^*	3^*	2^*	0
α^*	4^*	4^*	3^*	2^*	0^*

μ_1	μ_2	μ_3	μ_4	μ_5	μ_6
α^*	0	0	0	0	0
α^*	6^*	1	0	0	0
α^*	6^*	5^*	0	0	0
α^*	6^*	5^*	1^*	0	0
α^*	6^*	5^*	1^*	0	$\underline{1}^*$
α^*	0	0	0	0	0
α^*	6^*	0	0	0	0
α^*	6^*	5^*	4	0	0
α^*	6^*	5^*	4	5^*	0
α^*	6^*	5^*	4^*	5^*	2
α^*	6^*	5^*	4^*	5^*	$\underline{2}^*$

p_1	p_2	p_3	p_4	p_5	p_6
0	0	0	0	0	0
0	1	1	0	0	0
0	1	2	0	0	0
0	1	2	3	0	0
0	1	2	3	0	4
0	0	0	0	0	0
0	1	0	0	0	0
0	1	2	2	0	0
0	1	2	2	3	0
0	1	2	2	3	5
0	1	2	2	3	5

Table 2. Resolution of the MAXMIN-MAXMIN example.

A path L is efficient for the MINSUM-MAXMIN problem if and only if L contains no path L' for which the length is smaller and the cost of the cheapest arc is not smaller or for which the length is not greater and the cost of the cheapest arc is larger. Again, a minimal complete set contains at most m efficient paths. The following algorithm yields such a set and exploits the structure of the MINSUM-MAXMIN problem to avoid recomputing some labels when it is not compulsory.

ALGORITHM 2 : MINSUM-MAXMIN.

a) *Initialisation.*

Read the data. Set $T = N = \{1,2,\ldots,n\}$; $\lambda_1 = 0, \mu_1 = \alpha$, $\lambda_j = \alpha$ and $\mu_j = 0$ for $j = 2,3,\ldots, n$; $p_j = 0$ for $j = 1,2,\ldots, n$; $C_{min} = -1$. Proceed to b).

b) *Selection of the vertex with smallest label and test for ending.*

Compute $\underline{\lambda} = \text{Min } \{\lambda_j \mid j \in T\}$; if $\underline{\lambda} = \alpha$ stop, all the efficient paths sought for having been found. Otherwise compute $\overline{\mu} = \text{Max } \{\mu_j \mid j \in T, \lambda_j = \underline{\lambda}\}$ and select the vertex x_k such that $k = \text{Max } \{j \mid j \in T, \lambda_j = \underline{\lambda}, \mu_j = \overline{\mu}\}$. If $k = n$ go to d); otherwise set $T := T \setminus \{k\}$ proceed to c).

c) *Computation of new labels.*

For all j such that $(x_k,x_j) \in U$, $j \in T$ and $c_{kj} > C_{min}$

- if $\lambda_j > \lambda_k + d_{kj}$ set $\lambda_j = \lambda_k + d_{kj}$, $\mu_j = \min (\mu_k,c_{kj})$ and $p_j = k$;
- if $\lambda_j = \lambda_k + d_{kj}$ and $\mu_j < \min (\mu_k,c_{kj})$ set $\mu_j = \min (\mu_k,c_{kj})$ and $p_j = k$.

Return to b).

d) *Output of an efficient path.*

Print λ_n and μ_n, length and cost of the cheapest arc of the path just found. Set $j = p_n$, recursively print j and set $j = p_j$ until $j = 0$. Proceed to e).

e) *Suppression of arcs and updating of labels.*

Set $C_{min} = \mu_n$. Let $N_1 = \{j \mid j \in N, \lambda_j < \alpha, \mu_j \leqslant C_{min}\}$; set $T := T \cup N_1$. For each $j \in N_1$ let

$P_j = \{k \mid \{x_k,x_j\} \in U, k \in N \setminus T, c_{kj} > C_{min}\}$

- if $P_j = \emptyset$ set $\lambda_j = \infty$, $\mu_j = 0$ and $p_j = 0$.
- otherwise set $\lambda_j = \underset{k \in P_j}{\text{Min}} \; (\lambda_k + d_{kj})$

$$\mu_j = \underset{k \in P_j \mid \lambda_j = \lambda_k + d_{kj}}{\text{Max}} \; (\min (\mu_k, c_{kj}))$$

and $p_j = \text{Max} \{k \mid k \in P_j, \lambda_j = \lambda_k + d_{kj}, \mu_j = \min (\mu_k, c_{kj})\}$.
Return to b).

Theorem 4. *Algorithm* 2 *yields a minimal complete set of efficient paths for the* MINSUM-MAXMIN *problem in* $O(m^2 \log n)$ *operations.*

Proof. For the correctness of the algorithm, the λ_n and μ_n obtained at step d) must correspond to efficient paths. Let us first assume the labels retained at step e) are correct; the λ_j are obtained at step c) and compared at step b) following the rules of Dykstra's algorithm. Therefore, they correspond to the shortest lengths between x_1 and the other vertices x_j of G (or of the partial graph of G whose arcs are such that $c_{kj} > C_{min}$) each time step d) is attained. As any shortest path between x_1 and x_n in G contains only shortest subpaths between x_1 and any intermediairy vertex x_j, the values of the μ_j can be computed along with those of the λ_j[(2)], if ties are taken into account; their correctness is easily shown by induction on $|N-T|$.

To show the labels retained in step e) are correct, note that if $\mu_j > C_{min}$ there exists a path of length λ_j in the partial graph obtained after that step, as no arc of the shortest path from x_1 to x_j is deleted. Clearly, the new partial graph cannot contain a path from x_1 to x_j of length $< \lambda_j$. The rules for computing the temporary labels of the vertices x_j with $j \in N_1$ from those of their predecessors x_k with $k \in N \setminus T$ are easily checked.

The analysis of the number of operations required is similar to that of theorem 3 and therefore omitted here.

(2) Note that an equivalent property does not hold for the MAXMIN-MAXMIN problem.

Other algorithms for problems equivalent to MINSUM-MAXMIN or particular cases thereof are due to I.T. Frisch [6], [7] and to H.M. Moore [14]. In order to obtain a reliability constrained maximum capacity path, Frisch considers partial graphs with arcs such that c_{kj} is larger than a threshold progressively lowered until a path satisfying the constraint is obtained. The gaps in the sequence of the μ_n corresponding to efficient paths are thus not exploited. The algorithm of Moore is similar to algorithm 2 in that it alternates the computation of shortest paths and of maximum capacity paths in the shortest paths subgraphs, recomputing all labels after each iteration.

Example. The details of the application of algorithm 2 to the graph of figure 3 are given in table 3. Two efficient paths are found : x_1, x_3,x_5,x_6 with $\lambda_6 = 11$, $\mu_6 = 1$ and x_1, x_2, x_3, x_5, x_6 with $\lambda_6 = 14$ and $\mu_6 = 2$.

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	p_1	p_2	p_3	p_4	p_5	p_6
0^*	α	α	α	α	α	α	0	0	0	0	0	0	0	0	0	0	0
0^*	4^*	5	α	α	α	α	6	1	0	0	0	0	1	1	0	0	0
0^*	4^*	5^*	7	α	α	α	6	1	4	0	0	0	1	1	2	0	0
0^*	4^*	5^*	7	7^*	α	α	6	1	4	1	0	0	1	1	2	3	0
0^*	4^*	5^*	7^*	7^*	11	α	6	1	4	1	1	0	1	1	2	3	5
0^*	4^*	5^*	7^*	7^*	$\underline{11}^*$	α	6	1	4	1	$\underline{1}$	0	1	1	2	3	5
0^*	4^*	8^*	7^*	0	0	α	6	5	4	0	0	0	1	2	2	0	0
0^*	4^*	8^*	7^*	10^*	0	α	6	5	4	5	0	0	1	2	2	3	0
0^*	4^*	8^*	7^*	10^*	$\underline{14}^*$	α	6	5	4	5	$\underline{2}$	0	1	2	2	3	5
0^*	4^*	8^*	7^*	10^*	0^*	α	6	5	4	5	0	0	1	2	2	3	0

Table 3. Resolution of the MINSUM-MAXMIN example.

5. A FULLY POLYNOMIAL APPROXIMATION SCHEME

As shown by theorem 1, the MINSUM-MINSUM problem is, in worst case, intractable. The following algorithm allows to solve it, but may, for some problems, require an extremely large number of operations. Lists of efficient pairs of values λ_j^i, μ_j^i, i = 1, 2,..., are associated to the vertices x_j of G. Two pointers p_j^i and q_j^i are also used to recompose the efficient paths found; p_j^i = k index of the vertex from which x_j has been labelled and $q_j^i = \lambda_k^{i'}$ value associated to x_k for the labelling of x_j. R_j denotes the set of indices i of the undominated temporary labels associated with x_j; T denotes the set of indices of the vertices for which $R_j \neq \emptyset$:V denotes the set of indices for which $j \in T$ or R_j *was* $\neq \emptyset$.

ALGORITHM 3 : MINSUM-MINSUM.

a) *Initialisation.*

Read the data. Set $p_1^1 = q_1^1 = \lambda_1^1 = \mu_1^1 = 0$; T = {1}, V = {1}, R_1 = {1}, $R_j = \emptyset$ for j = 2, 3,..., n. Proceed to b).

b) *Selection of the vertex with smallest label and test for ending.*

If $T = \emptyset$, end, all efficient apth sought for having been found. Otherwise compute $\underline{\lambda} = \text{Min}\{\lambda_j^i, j \in T, i \in R_j\}$ and select x_k sucht that $k = \text{Max}\{j \mid \lambda_j^{i'} = \underline{\lambda}, j \in T, i' \in R_j\}$. Delete i' from R_k and k from T if $R_k \neq \emptyset$. If k = n, go to d). Otherwise proceed to c).

c) *Computation of new labels.*

For each j such that $(x_k, x_j) \in U$ consider the quadruplet $(p_j^i = k, q_j^i = \lambda_k^i, \lambda_j^i = \lambda_k^{i'} + d_{kj}, \mu_j^i = \mu_k^{i'} + c_{kj})$

- if $j \notin V$ introduce the quadruplet in the list of x_j, set R_j = {1}, $T := T \cup \{j\}$, $V := V \cup \{j\}$;
- if $j \in V$ and $j \notin T$ compare the quadruplet with the undominated quadruplets in the list of x_j; if it is dominated erase it; otherwise add it to the list; choose for i the first value not yet used in R_j; set R_j = {i}, $T := T \cup \{j\}$.
- if $j \in V$ and $j \in T$, compare the quadruplet with the undominated un-

selected, quadruplets of x_j; if some of them are dominated by the new quadruplet, erase them and delete their index from R_j. Then compare the new quadruplet with all the undominated quadruplets of x_j and if it is dominated erase it; otherwise add it to the list; choose for i the first value not yet used in R_j; set $R_j := R_j \cup \{i\}$. Return to b).

d) *Output of an efficient path.*

Print λ_n^i and μ_n^i, length and cost of the efficient path just found. Use the p_j^i and q_j^i to recompose backwards the list of vertices of that path. Return to b).

While algorithm 3 is not polynomial, it is *pseudo-polynomial*, i.e. its number of operations is bounded by a polynomial in the problem's characteristics and the magnitude of the data (assumed to be integers). Let $D = \max_{(x_k,x_j)\in U} d_{jk}$ and assume (without loss of generality) that $D \leqslant C = \max_{(x_k,x_j)\in U} c_{kj}$.

Theorem 5. *Algorithm 3 allows to solve the* MINSUM-MINSUM *problem in* $O(nmD \log(nD))$ *operations.*

Proof. Let us first show that the algorithm yields a complete set of efficient paths : this follows from the facts that any efficient path from x_1 to x_n contains only efficient subpaths from x_1 to any intermediary vertex x_j, quadruplets are only eliminated when they are dominated, i.e. when they do not correspond to efficient subpaths, all quadruplets obtainable from each selected quadruplet are examined, and as quadruplets are selected in order of increasing values of $\underline{\lambda}$ all selected quadruplets correspond to efficient subpaths or paths.

Step a) requires O(n) operations. Step b) requires O (log n) operations if the unselected quadruplets with smallest λ_j^i for each j are stored in a heap; as the minimum number of efficient paths ending in x_j is bounded by nD (bound on the length of the longest elementary path in G), step

b) occurs at most n^2D times and therefore requires $O(n^2D \log n)$ operations in all in worst case. Step c) requires $O(\log nD)$ operations to check dominance for each new quadruplet, if the quadruplets are ranked and stored in a *balanced tree;* it also requires $O(\log n)$ operations to update the heap in which the λ_j^i are stored; the number of new quadruplets is bounded by nmD so step c) requires $O(nmD \log (nD))$ operations. Step d) requires $O(n)$ operations and occurs nD times at most, i.e. requires $O(n^2D)$ operations in all. Thus, assuming $m \geq n$, the algorithm requires $O(nmD \log(nD))$ operations in worst case.

Another algorithm for the MINSUM-MINSUM problem, which generalizes the algorithm of Ford [5] is due to Ph. Vincke [16].

Example. The details of the application of algorithm 3 to the example of figure 3 are given in table 4. Selected quadruplets are starred. Two efficient paths are found : x_1, x_3, x_5, x_6 with $\lambda_6 = 11$ and $\mu_6 = 8$ and x_1, x_3, x_4, x_6 with $\lambda_6 = 16$ and $\mu_6 = 3$.

1	2	3	4	5	6
(0,0,0,0)*	(1,0,4,6)*	(1,0,5,1)			
(0,0,0,0)*	(1,0,4,6)*	(1,0,5,1)*	(2,4,7,10)		
		(2,4,8,11)			
(0,0,0,0)*	(1,0,4,6)*	(1,0,5,1)*	(2,4,7,10)	(3,5,7,6)*	
			(3,5,9,2)		
(0,0,0,0)*	(1,0,4,6)*	(1,0,5,1)*	(2,4,7,10)*	(3,5,7,6)*	(5,7,11,8)
			(3,5,9,2)		
			(5,7,10,11)		
(0,0,0,0)*	(1,0,4,6)*	(1,0,5,1)*	(2,4,7,10)*	(3,5,7,6)*	(5,7,11,8)
			(3,5,9,2)*		(4,7,14,11)
(0,0,0,0)*	(1,0,4,6)*	(1,0,5,1)*	(2,4,7,10)*	(3,5,7,6)*	(5,7,11,8)*
			(3,5,9,2)*		(4,9,16,3)*

Table 4. Resolution of the MINSUM-MINSUM example.

Algorithm 3 can be transformed into a *fully polynomial approximation scheme* if approximate solutions only are sought; it is then required that the number of operations be bounded by a polynomial in the problem's characteristics and $1/\varepsilon$ where ε is the percentage or error allowed.

ALGORITHM 4 : APPROXIMATE MINSUM-MINSUM.

a) *Initialisation.*

Read the data. Compute the length λ_{min} of the shortest path and the cost λ_{min} of the cheapest path between x_1 and x_n in G by Dykstra's algorithm; let λ_{max} denote the length of the path corresponding to μ_{min}. Rank the arcs of G in order of increasing lengths. Let d_{kj} denote the length of the shortest arc such that $n\, d_{kj} \geqslant \lambda_{min}$.

b) *Scaling and use of algorithm 3.*

Express the length d_{kj} of all arcs in the unit $\frac{\varepsilon d_{kj}}{n}$ and keep the integer parts of these values. Apply algorithm 3 to the graph so obtained, seeking only efficient paths with a length $\leqslant n\, d_{kj}$.

If $n\, d_{kj} \geqslant \lambda_{max}$, stop, all approximate efficient paths sought for having been found; otherwise consider the next value of d_{kj} and iterate step b).

Theorem 6. *Algorithm* 4 *provides an* ε*-approximate solution to the* MINSUM-MINSUM *problem in* $0\,(m^2 \frac{n^2}{\varepsilon} \log (\frac{n^2}{\varepsilon}))$ *operations.*

Proof. Let d_{kj} denote the length of the longest arc in an elementary path between x_1 and x_n; then the length of this path belongs to $[d_{kj}, (n-1)\, d_{kj}]$ and it contains between 1 and n-1 arcs. So, if the arc's lengths are expressed in the unit $\frac{\varepsilon d_{kj}}{n}$ and restricted to their integer part, the error in the path length will not be greater than $\frac{\varepsilon d_{kj}}{n} \cdot n = \varepsilon d_{kj}$ in absolute value, or to ε in relative value. When in step b) efficient paths of length at most $n d_{kj}$ are thought for, there can be at most $\frac{n^2}{\varepsilon_2}$ of them. Then, the application of algorithm 3 requires $0\,(m \frac{n^2}{\varepsilon} \log (\frac{n^2}{\varepsilon}))$ operations. As there are m arcs the number of operations of

algorithm 4 is bounded by $O\left(m^2 \frac{n^2}{\varepsilon} \log\left(\frac{n^2}{\varepsilon}\right)\right)$.

6. CONCLUSIONS

Bicriterion path problems have many actual and potential applications. Among the various problems of that category it appears that problems involving one or two criteria of the MAXMIN or MINMAX type are easy while those involving two criteria of the MINSUM or MAXSUM type are, in worst case, intractable. Polynomial algorithms are easily devised for the first class of bicriterion problems, while for the second class only pseudo-polynomial algorithms or fully polynomial approximation schemes may be obtained.

Among many questions worth future research are the obtention of algorithms for problems with MAXSUM type objective (easy when the paths sought for are not required to be elementary and aparently quite difficult otherwise) the study of particular cases such as constrained path problems and the examination of the effect of additionnal requirements on G such as planarity, or limitations on the degrees of the vertices.

REFERENCES

[1] Amara, R.C., Lindgren, H. and M. Pollack, "Link Error Control and Network Route Selection", IRE *Trans. Commun. Systems*, CS-9, 328-334, 1961.

[2] Aneja, Y.P. and K.P.K. Nair, "The Constrained Shortest Path Problem", *Naval Research Logistics Quarterly*, 25, 549-555, 1978.

[3] Berge, C., *"Graphes et hypergraphes"*, Paris : Dunod, 1970, English translation Amsterdam : North-Holland, 1973.

[4] Dykstra, E.W., "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, 1, 269-271, 1959.

[5] Ford, L.R. Jr., Network Flow Theory, The Rand Corporation, Paper P-923, July 1956.

[6] Frank, H. and I.T. Frisch, *"Communication, Transmission and Transportation Networks"*, Reading, Massachusetts : Addison - Wesley, 1971.

[7] Frisch, I.T. "Optimum Routes in Communication Systems with Channel Capacities and Channel Reliabilities", IEEE *Trans. Commun. Systems* CS-11, 241-244, 1963.

[8] Garey, M. and D.S. Johnson, *"Computers and Intractability : A Guide to the Theory of NP-Completeness"*, San Francisco : Freeman, 1979.

[9] Hansen, P., "An O(m log D) Algorithm for Shortest Paths", Discrete Applied Mathematics (forthcoming).

[10] Hardgrave, W.W. and G.L. Nemhauser, "On the Relation Between the Traveling Salesman and the Longest Path Problems", *Operations Research*, 10, 647-657 (1962).

[11] Lawler, E., *"Combinatorial Optimization, Network and Matroïds"*, New York : Holt, Rinehart and Winston, 1976.

[12] Marchet, J.C. and J. Siskos, "Aide à la décision en matière d'environnement : Application au choix de tracé autoroutier", Rapport LAMSADE, 23-1979, Université de Paris Dauphine.

[13] Megiddo, N. p. 214 in [7].

[14] Moore, H.M., "On the Fastest Route for Convoy-Type Traffic in Flow-rate Constrained Networks", *Transportation Science*, 10, 113-124, 1976.

[15] Pandit, S.N.N., "Some Observations on the Longest Path Problem", *Operations Research*, 12, 361-364, 1964.

[16] Vincke, Ph., "Problèmes Multicritères", *Cahiers du Centre d'Etudes de Recherche Opérationnelle*, 16, 425-439, 1974.

[17] Wongseelashote, A., "Semirings and Path Spaces", *Discrete Mathematics*, 26, 55-78, 1979.