Dynamic programming and minimum risk paths

by

Paolo Serafini

University of Udine, Department of Mathematics and Computer Sciences

Via delle Scienze 206, 33100, Udine, Italy, and

CISM, International Center for Mechanical Sciences, Udine, Italy

email: serafini@dimi.uniud.it phone: +390432558442, fax +390432558499

Abstract: This paper addresses the problem of computing minimum risk paths by taking as objective the expected accident cost. The computation is based on a dynamic programming formulation which can be considered an extension of usual dynamic programming models: path costs are recursively computed via functions which are assumed to be monotonic. A large part of the paper is devoted to analyze in detail this formulation and provide some new results. Based on the dynamic programming model a linear programming model is also presented to compute minimum risk paths. This formulation turns out to be useful in solving a biobjective version of the problem, in which also expected travel length is taken into consideration. This leads to define nondominated mixed strategies. Finally it is shown how to extend the basic updating device of dynamic programming in order to enumerate all nondominated paths.

Keywords: routing, risk analysis, dynamic programming, multiple objective programming, hazardous materials.

1. INTRODUCTION

The travelling of hazardous materials has raised the problem of computing paths minimizing not only the length (cost or time) but also the risk of damages caused by accidents. Defining the length of a path and computing a minimal path length is clearly a standard issue. On the contrary there are various ways of defining and computing the "risk" of a path. The paper by Erkut and Verter (1998) reviews in detail the various approaches suggested in the literature and tests them on a real example.

We summarize the relevant facts. Two quantities are typically involved in assessing the risk: the probability of accident occurrence on a certain path edge and the incurred cost in case of accident on that path edge. Although it is not straightforward how to measure the cost, we assume that numbers c_e, related to the accident costs, are available for each path edge e.

Two simple ways of taking care of the risk consider just one of the two quantities. For instance we may consider that accident costs are high on any path edge and consequently we want to minimize the probability of an accident along a path P. If p_e is the probability of an accident on the edge e, we have to compute

minimal length paths with 'length' given by $\sum_{e\in P} -\log(1-p_e)$. Alternatively we may consider that sooner or later an accident will occur by repeated travelling and therefore we may find more appropriate to minimize either $\sum_{e\in P} c_e$ or $\max_{e\in P} c_e$. The two quantities can also be considered together as a bicriterion problem (i.e. considering efficient solutions of minimizing at the same time both the accident probability and the cost).

Furthermore, accident probabilities and costs can be combined to evaluate the expected cost of an accident. This is the so called path risk (Erkut and Verter (1998)). The expected cost on path $P = \{e_1, e_2, \ldots, e_m\}$ is defined as

$$c(P) := p_1\, c_1 + (1-p_1)\, p_2\, c_2 + (1-p_1)\,(1-p_2)\, p_3\, c_3 + (1-p_1)\,(1-p_2)\,(1-p_3)\, p_4\, c_4 + \ldots \qquad (1)$$

where it is explicitly taken into account the fact that once an accident has occurred on path edge e_k no further transportation will take place on the subsequent path edges e_h, $h > k$.

This paper has been first motivated by the attempt to compute paths according to this expression without simplifying it. Erkut and Verter (1998) disregard higher order terms in (1) thus arriving to the following simple linear expression for the path expected cost: $\sum_{e\in P} p_e\, c_e$. They find this approximation both realistic given the small probability values in real life problems and also convenient because (1) is otherwise computed by resorting to a complex nonlinear integer programming problem or to a larger linear integer programming problem.

We want to show in this paper that dynamic programming can be easily used to compute minimal paths according to (1) in polynomial time. Although it can be practical to simplify such a cumbersome expression, we think that it is worthwhile facing the problem directly. The first observation is that (1) can be more compactly written recursively (in backward form)

$$c(P) := p_1\, c_1 + (1-p_1)\, c(P\backslash e_1) \qquad (2)$$

This particular definition of path cost is particularly suited to dynamic programming. However, the dynamic programming formulation we need is slightly more general than the one found in the literature and we need to address in detail this issue in Section 2. As we shall see, there are some subtle points connected to this formulation. There are some strong connections between the path cost definition introduced in this paper and the generalized path algebra introduced by Gondrand and Minoux (1979). In this paper we investigate the link between dynamic programming principles and techniques and a general path cost definition and provide new results.

In Section 3 we apply the dynamic programing techniques to solve (2) and discuss the solution. There are similarities between (2) and Markov decision processes (for a general reference to MDP see for instance Puterman (1994)). The difference is that in MDP paths are random, i.e. they are not known in advance, whereas here paths are deterministic (except for the possible interruption caused by an accident). We will show also a linear programming formulation of the problem.

With respect to the problem of finding the 'best' path for the travelling of hazardous materials it is certainly more interesting to address the problem as a bicriterion one. In Section 4 we consider bicriterion minimal paths and introduce the idea, borrowed from MDP and game theory, of using mixed strategies, that is using alternative paths with random selection (for the concept of mixed strategies refer for instance to Luce and Raiffa (1957)). The linear programming formulation turns out to be naturally suited to this approach. If mixed strategies cannot be considered we present a biobjective version of the basic updating mechanism of dynamic programming algorithms.

2. EXTENDING THE SCOPE OF DYNAMIC PROGRAMMING

Let $G = (N, E)$ be a directed graph (let $n = |N|$ and $m = |E|$) and s a distinguished node of G. We consider directed paths starting from s. This corresponds to a forward dynamic programming model. In case of a backward model we have a distinguished node t and consider directed paths ending in t. We present only one model since the results can be easily reformulated for the other model. Although the problem we consider in this paper requires a backward model we present the results for the forward model because it is slightly more intuitive.

We adopt the following notation: if P is a generic path (starting from s as we always assume in the forward model) and j is a node in P we denote by P_j the restriction of P from s to j. We may also emphasize the fact that a path ends in node i by writing P_i.

The extension of dynamic programming we propose consists in the particular definition of a path cost $V(P)$. For each arc (ij) a function $f_{ij} : \mathbb{R} \to \mathbb{R}$ is defined. The cost of the path P_s consisting only of the source s is defined as $\bar{V}_s$. For all other paths the cost of reaching j from s along the path P is defined by

$$V(P_j) = f_{ij}(V(P_i))$$

with i the node immediately preceding j on P_j. Note that the cost $V(P_j)$ depends (via the function f_{ij}) only on the cost of the path P_i and not on the path itself. Hence a path cost is recursively defined by

$$V(P_s) := \bar{V}_s, \qquad V(P_j) := f_{ij}(V(P_i)) \qquad \forall (i,j) \in P \tag{3}$$

so that the cost of $P : s \to i \to j \ldots \to k \to h$ is given by

$$V(P) := f_{kh}(\ldots(\, f_{ij}(\, f_{si}(\bar{V}_s))))$$

The objective consists in finding for each node j a path P^j minimizing (or maximizing) $V(P^j)$.
In most cases $\bar{V}_s = 0$ and the functions f_{ij} are additive, i.e.

$$f_{ij}(V) = c_{ij} + V \qquad \forall (i,j) \in E \tag{4}$$

This is indeed the case considered in the literature. The functions we consider in this paper are more general although not arbitrary. As we shall see, the functions must exhibit some form of monotonicity. The most important assumption is that the functions are increasing (i.e. $f(a) > f(b)$ for any $a > b$). In case they are only nondecreasing (i.e. $f(a) \geq f(b)$ for any $a > b$) some important properties are retained if they are in addition superlinear (i.e. $f(a) \geq a$) for minimum problems or sublinear (i.e. $f(a) \leq a$) for maximum problems.

As already said, path cost generalizations have been proposed in the literature. We want to mention the generalized path algebra by Gondrand and Minoux (1979), which extends the additive case to more general algebraic binary operators. In particular they consider a further extension of their model in which arcs are associated to cost functions fulfilling an algebraic property. This property turns out to be a monotonicity property (but not strict monotonicity). There is a strong similarity between this path cost definition and the one adopted in this paper. However, there does not seem to be in the literature a careful investigation of the links between dynamic programming ideas and this kind of path cost generalizations, as we are going to do in this section.

We may wonder whether considering generic (increasing or nondecreasing functions) $f_{ij}(V)$ instead of usual additive functions $V + c_{ij}$ is worth the added mathematical complication. Actually, beside the dynamic programming model we need for the problem addressed in this paper, there may be other practical problems for which we do need generic functions $f_{ij}(V)$. Here we provide a few examples.

Example 1. (see also Halpern and Priess (1974) and Minoux (1976)). A station s must send a message to a station t. The message is routed via a satellite network. Two given satellites can exchange messages only during certain time windows. If a time window is not available the message is stored in the satellite and is sent with a delay. Knowing in advance the time windows for all satellite pairs, which is the fastest route for the message (taking into account only storage delays)? We may model the problem by defining the length $V(P_i)$ of a path P_i as the time needed by the message to reach the node (satellite) i. Then the function $f_{ij}(V)$ is defined as

$$f_{ij}(V) = \begin{cases} V & \text{if } V \text{ is within a time window} \\ W & \text{if } V \text{ is not in a time window and } W > V \text{ is the start of the next available time window} \end{cases}$$

This function is nondecreasing and superlinear. In the cited papers storage time windows are also considered.

∎

Example 2. (see also Cooke and Halsey (1966) and Halpern (1977)). This is an extension of the previous example. Let us suppose that the travelling time of a road depends on the particular time of the day one enters the road. Modeling a road as an arc (i, j) we define the travelling time as $T_{ij}(V)$ with V the time of the day. Hence entering the road at time V means leaving the road at time $V + T_{ij}(V)$. So we define functions $f_{ij}(V) := V + T_{ij}(V)$. These functions are clearly superlinear. We may assume that entering the road earlier means also leaving the road not later, no matter how the traffic condition is. In other

words two identical drivers will never pass each other. With this assumption the functions are nondecreasing (possible stops along the road make the functions non strictly increasing). Dropping the assumption makes the problem more difficult and optimal paths may include cycles (see the cited papers). ∎

Example 3. We have already mentioned that finding a path minimizing the failure probability can be solved by transforming probabilities to their logarithms and solving a normal shortest path problem. However, we can also approach the problem directly by defining $\bar{V}_s = 1$ and $f_{ij}(V) := V \cdot p_{ij}$ (with p_{ij} probability of no failure) and finding a maximum path. The functions f are sublinear and increasing. ∎

As we shall see later, we can efficiently compute the optimal paths of these three examples by a slightly modified Dijkstra algorithm.

Example 4. Bottleneck problems can be framed within dynamic programming by taking functions (for minimum problems) $f_{ij}(V) := \max\{V, c_{ij}\}$, which are nondecreasing and superlinear. An example can be found in the paper by Baker et al. (1983) where n tasks have to be scheduled on one machine. The scheduling of task j is measured by a penalty function $\varphi_j(C_j)$ of the completion time C_j of task j. The functions φ_j are assumed to be nondecreasing. The cost of scheduling is defined as $\max_j \varphi_j(C_j)$. The processing time of task j is p_j. We define a directed layered graph whose nodes correspond to subsets J of $\{1, 2, \ldots, n\}$. In particular the source s corresponds to the empty set and the sink t to $\{1, 2, \ldots, n\}$, and the arcs are defined from any subset J to any subset $J \cup k$, with $k \notin J$. Then each path $s \to t$ corresponds to a permutation of $\{1, 2, \ldots, n\}$. The function f_{ij} associated to the arc (ij) is

$$f_{(J,J\cup k)}(V) = \max\{\, V \,;\, \varphi_k(\sum_{i \in J \cup k} p_i)\}$$

(this definition is justified by the property of the functions φ_j). Although the graph has an exponential number of nodes an optimal path can be found in polynomial time with respect to the number of tasks. ∎

Assuming functions f more general than additive ones requires a review of the dynamic programming concepts. This will cast a new light to known properties and will also provide new results. Toward the goal of finding optimal paths we state the well known optimality principle (however stated as a definition and not as a theorem):

Definition 1: *An optimal path $\hat{P}$ satisfies the optimality principle if, for every node k on $\hat{P}$, $\hat{P}_k$ is optimal among all paths ending in k.* ∎

The optimality principle does not necessarily hold for all possible functions f_{ij}. The following theorem is straightforward:

Theorem 2: *If the functions f_{ij} are increasing the optimality principle holds for any optimal path.* ∎

If the functions are only nondecreasing there may be optimal paths not satisfying the optimality principle. For instance $\bar{V}_s = 0$, $f_{s1}(V) = V + 3$, $f_{s2}(V) = V + 2$, $f_{12}(V) = \min\{V, 1\}$, $f_{21}(V) = 2$. There are four simple paths, namely $P^1 : s \to 2 \to 1$, $P^2 : s \to 1 \to 2$ and their restrictions $P_2^1 : s \to 2$, $P_1^2 : s \to 1$. All other (nonsimple) paths are obtained by looping around the cycle $1 \to 2 \to 1$. We have $V(P^1) = 2$, $V(P^2) = 1$, $V(P_2^1) = 2$, $V(P_1^2) = 3$. Any other (nonsimple) path ending in 2 has cost 1 and any other path ending in 1 has cost 2. Hence there are infinite minimal paths ending in 2 with value 1. For these paths take the restriction consisting of the single arc $(s, 1)$ with value 3 which is not optimal among the paths ending in 1. So no optimal path from s to 2 satisfies the optimality principle. In order to investigate the role of the optimality principle with nondecreasing functions f we need two preliminary lemmas. The first is straightforward and is based on the observation that superlinearity rules out paths with cycles in searching for minimal paths.

Lemma 3: *If the functions f_{ij} are superlinear (sublinear) there exists a minimum (maximum) path to any node k.* ∎

Lemma 4: *If the functions f_{ij} are nondecreasing and superlinear (sublinear), then for each node k in a minimum (maximum) simple path P^j, there exists a minimum (maximum) path P^k not containing the node j.*

Proof: (for minimum problems) The proof is by contradiction. Let us suppose that paths not containing j are not minimal. Then a minimum path P^k contains j and $V(P^k) < V(P_k^j)$. Let us denote $V^j := V(P^j)$, $V^k := V(P^k)$, $V_k := V(P_k^j)$ and $V_j := V(P_j^k)$. So $V^k < V_k$.

By definition we have $V^j = F_{kj}(V_k)$ (with F_{kj} composition of the f_{ij} from k to j along P^j) and $V^k = F_{jk}(V_j)$ (with F_{jk} composition of the f_{ij} from j to k along P^k). By optimality $V^j \leq V_j$ and $V^k \leq V_k$. By superlinearity $V_k \leq F_{kj}(V_k)$ and $V_j \leq F_{jk}(V_j)$. Therefore

$$V^j \leq V_j \leq F_{jk}(V_j) = V^k \leq V_k \leq F_{kj}(V_k) = V^j$$

from which $V^k = V_k$ contradicting the hypothesis. ∎

Theorem 5: *If the functions f_{ij} are nondecreasing and superlinear (sublinear), then there exists, for each node j, a minimum (maximum) path $\hat{P}^j$ for which the optimality principle holds.*

Proof: Given an optimal path $\hat{P}^j$ let k be the node preceding j on $\hat{P}^j$. Let $\hat{P}^k$ be an optimal path not containing j. Such a path exists by the previous lemma. Now redefine $\hat{P}^j$ as the path $\hat{P}^k \cup (k, j)$. Let us now consider the node h preceding k on the new path $\hat{P}^j$. By the lemma there exists an optimal path $\hat{P}^h$ containing neither k nor j. Let us redefine $\hat{P}^j$ as $\hat{P}^h \cup (h, k) \cup (k, j)$. By proceeding recursively there are no repeated nodes by the lemma and so we must reach s. The claimed path is given by the final $\hat{P}^j$. ∎

As is well known the importance of the optimality principle consists in the possibility of exploiting the following recursive equation (generally referred to in the literature as 'Bellman's equation') in the variables $V_1, \ldots, V_n$, which, for minimum problems is

$$V_j = \min_{i \in \delta^-(j)} f_{ij}(V_i) \qquad V_s = \bar{V}_s \tag{5}$$

and for maximum problems

$$V_j = \max_{i \in \delta^-(j)} f_{ij}(V_i) \qquad V_s = \bar{V}_s \tag{6}$$

with $\delta^-(j) := \{i \in N : (i,j) \in E\}$.

A direct consequence of (5) and the monotonicity property is that, for each node i and any cycle C from i, the solutions of the Bellman's equation for minimum problems must satisfy $V_i \leq F_C(V_i)$ (for maximum problems $V_i \geq F_C(V_i)$), where F_C is the composition of the f along the cycle C.

Theorem 6: *If the optimality principle holds for at least one optimal path $\hat{P}^j$, for each node j, then the optimal values $V_j := V(\hat{P}^j)$ satisfy the Bellman's equation.*

Proof: (for minimum problems) Let k be the node preceding j in the optimal path $\hat{P}^j$ for which we assume the optimality principle holds. Then $V_j = f_{kj}(V_k)$ (by the optimality principle). Let i be any predecessor node of j and let $\hat{P}^i$ be the optimal path from s to i with optimal value V_i. By extending $\hat{P}^i$ up to j one has that $f_{ij}(V_i) \geq V_j$ by optimality of V_j and the thesis follows. ∎

It is interesting to note that the Bellman's equation is satisfied by the optimal values even if the optimality principle does not hold (see the previous example). It is enough that the functions are nondecreasing.

Theorem 7: *If the functions f_{ij} are nondecreasing and there exist optimal paths $\hat{P}^j$ for each node j, then the optimal values V_j satisfy the Bellman's equation.*

Proof: (for minimum problems) Let k be node preceding j on $\hat{P}^j$. Then we have $V_k \leq V(\hat{P}^j_k)$ by optimality and, since the functions are nondecreasing, $f_{kj}(V_k) \leq f_{kj}(V(\hat{P}^j_k)) = V_j$. For every predecessor i of j let us consider the paths $\hat{P}^i \cup (i,j)$ whose costs are $f_{ij}(V_i)$. By optimality $f_{ij}(V_i) \geq V_j$ for every i and in particular $f_{kj}(V_k) \geq V_j$. From the preceding inequality $f_{kj}(V_k) = V_j$ and the Bellman's equation is satisfied. ∎

We may wonder whether the Bellman's equation may be satisfied by the inf (sup) of the path costs when optimal paths do not exist and the problem is bounded. This can happen if the graph has cycles and looping around the cycles improves the path cost. For instance the problem of computing minimum risk paths may present such a situation (see next section). We have the following theorem.

Theorem 8: *If the functions f_{ij} are nondecreasing and continuous and the path costs are bounded, then the inf (sup) of the path costs satisfy the Bellman's equation.*

Proof: (for minimum problems) Let $\hat{P}^h_j$ be a sequence of paths $s \to \ldots k \to j$ such that $\lim_{h\to\infty} V(\hat{P}^h_j) = V_j$. There always exists a node k such that a sequence of that type exists. Let $\bar{P}^h_k$ be a sequence of paths $s \to \ldots k$

such that $\lim_{h\to\infty} V(\bar{P}_k^h) = V_k$. By restricting every path $\hat{P}_j^h$ to k, one has $V_k \le V(\hat{P}_{jk}^h)$. Then, since the functions are nondecreasing, $f_{kj}(V_k) \le f_{kj}(V(\hat{P}_{jk}^h)) = V(\hat{P}_j^h)$ for every h. Hence by taking the limit, $f_{kj}(V_k) \le V_j$. Let i be any predecessor node of j and let $\bar{P}_i^h$ be a generic sequence of paths convergng to the value V_i. By extending every path $\bar{P}_i^h$ up to j one has, by optimality of the limiting values, $f_{ij}(V(\bar{P}_i^h)) \ge V_j$. By taking the limit and by continuity one has $f_{ij}(V_i) \ge V_j$ for every i and in particular $V_j = f_{kj}(V_k)$. ∎

The continuity assumption is necessary. Just consider $\bar{V}_s = 0$, $f_{s1}(V) = V + 2$, $f_{12}(V) = \sqrt{V}$, $f_{23}(V) = \sqrt{V}$, $f_{31}(V) = \sqrt{V}$, $f_{24}(V) = V$ if $V > 1$ and $f_{24}(V) = V - 1$ if $V \le 1$. Then $\inf_P V(P_4) = 1$ but the solution of the Bellman's equation is $V_1 = V_2 = V_3 = 1$, $V_4 = 0$.

The previous results show that optimal path values can be computed by solving the Bellman's equation. However, we may wonder whether there are solutions of the Bellman's equation not related to optimal path values. Indeed this may happen. Just consider (with all additive functions!): $\bar{V}_s = 0$, $f_{s1}(V) := V + 3$, $f_{s2}(V) := V + 2$, $f_{s3}(V) := V + 2$, $f_{12}(V) := V$, $f_{23}(V) := V$, $f_{31}(V) := V$, $f_{24}(V) := V + 1$, $f_{34}(V) := V + 1$. There are infinite solutions of the Bellman's equation (minimum), i.e.: $V_s = 0$, $V_1 = V_2 = V_3 = \alpha$, $V_4 = 1 + \alpha$, for any $\alpha \le 2$. Only the values with $\alpha = 2$ correspond to optimal path values. The spurious solutions are due to the cycle $1 \to 2 \to 3 \to 1$ for which the composition of the f_{ij} is the identity.

Another interesting example is the following: $\bar{V}_s = 0$, $f_{s1}(V) := V + 3$, $f_{s2}(V) := V + 2$, $f_{s3}(V) := V + 2$, $f_{12}(V) := V^2/4$, $f_{23}(V) := V + \sqrt{V}/2$, $f_{31}(V) := V + 1/2$, $f_{24}(V) := V + 1$, $f_{34}(V) := V + 1$. We see that $F_C(V) := f_{31}(f_{23}(f_{12}(V))) = V^2/4 + V/4 + 1/2$ and that $F_C(V) = V$ for $V = 1$ and $V = 2$. The Bellman's equation is satisfied by three different solutions: $V_s = 0$, $V_1 = 1$, $V_2 = 1/4$, $V_3 = 1/2$, $V_4 = 5/4$, or $V_s = 0$, $V_1 = 2$, $V_2 = 1$, $V_3 = 3/2$, $V_4 = 2$, or $V_s = 0$, $V_1 = 5/2$, $V_2 = 25/16$, $V_3 = 2$, $V_4 = 41/16$. Of the three solutions only the last one corresponds to optimal path values. Also note that $F_C(3/2) < 3/2$. Hence there may be cycles C and values V such that $F_C(V) < V$, and yet the Bellman's equation is feasible (as remarked $F_C(\tilde{V}) < \tilde{V}$ is excluded for the solutions $\bar{V}$ of the Bellman's equation).

In both examples the solutions of the Bellman's equation corresponding to optimal values are the ones with largest values and the spurious solutions are related to the presence of cycles C and values V such that $F_C(V) = V$. By excluding this possibility we get rid of the spurious solution.

Theorem 9: *If $F_C(V) > V$ ($F_C(V) < V$) for all cycles C and the functions f_{ij} are nondecreasing, then the solutions of the Bellman's equation (5) ((6)) correspond to minimum (maximum) path values.*

Proof: (for minimum problems) Let $\tilde{V}$ be a solution of the Bellman's equation. Let $k(j)$ be the node for which the minimum is achieved in the expression

$$\tilde{V}_j = \min_{i:(ij)\in E} f_{ij}(\tilde{V}_i) = f_{k(j)j}(\tilde{V}_{k(j)}) \tag{7}$$

(break ties arbitrarily). Let us define the set of arcs

$$\tilde{E} := \{(k(j), j) : j \in N \setminus s\}$$

We claim that $\tilde{E}$ is a tree rooted in s. Let us suppose there is a cycle C in $\tilde{E}$. If not all arcs in C have the same orientation, then at least one node in C must have two entering arcs, but this case is excluded because in $\tilde{E}$ all nodes different from s have exactly one entering arc and the source has none. Therefore all arcs in C have the same orientation. Then (7) implies $\tilde{V}_h = F_C(\tilde{V}_h)$ with h a node on the cycle, contradicting the hypothesis.

Therefore the sequence of nodes j, $k(j)$, $k(k(j))$, ..., ends in s for each j, implying that $\tilde{E}$ is is a tree rooted in s and $\tilde{V}_j$ is the length of the unique path in $\tilde{E}$ from s to j. We now show that these paths are optimal.

Let P be any path in E. For each arc $(i,j) \in P$ let us assume the induction hypothesis that $V(P_i) \geq \tilde{V}_i$. Then the nondecreasing property and (7) imply

$$V(P_j) = f_{ij}(V(P_i)) \geq f_{ij}(\tilde{V}_i) \geq \tilde{V}_j$$

i.e. the property is true also at node j. Since $V(P_s) = \tilde{V}_s = \bar{V}_s$, the thesis is true for all nodes of the path P implying optimality of the solutions $\tilde{V}_j$. ∎

Lemma 10: *Let $\hat{P}$ be a minimum (maximum) path with values $\hat{V}_i := V(\hat{P}_i)$ and let $\tilde{V}$ be a solution of the Bellman's equation. Then $\tilde{V}_i \leq \hat{V}_i$ ($\tilde{V}_i \geq \hat{V}_i$) for every $i \in \hat{P}$ if the functions f_{ij} are nondecreasing.*

Proof: (for minimum problems) The proof goes by induction. One has $\tilde{V}_s \leq \hat{V}_s$ (in fact $\tilde{V}_s = \hat{V}_s = \bar{V}_s$). Let i and j be two adjacent nodes in $\hat{P}$ and suppose $\tilde{V}_i \leq \hat{V}_i$. Then

$$\tilde{V}_j = \min_{k \in \delta^-(j)} f_{kj}(\tilde{V}_k) \leq f_{ij}(\tilde{V}_i) \leq f_{ij}(\hat{V}_i) = \hat{V}_j$$

∎

The spurious solutions may be eliminated by transforming the Bellman's equation into the following optimization problem (for minimum problems)

$$\begin{aligned} \max \quad & \sum_i V_i \\ & V_j \leq f_{ij}(V_i) \qquad \forall (i,j) \in E \\ & V_s = \bar{V}_s \end{aligned} \tag{8}$$

Theorem 11: *The optimal solutions of (8) satisfy the Bellman's equation and correspond to the minimum path values.*

Proof: Let V_i be a feasible solution in (8). If there exists a node j such that $V_j < f_{ij}(V_i)$ for each $i \in \delta^-(j)$, then there exists another feasible solution V', with $V'_i := V_i$ if $i \neq j$ and $V'_j := \min_{i \in \delta^-(j)} f_{ij}(V_i) > V_j$. Since the functions f are nondecreasing we are guaranteed that V' is feasible, because $V'_k = V_k \leq f_{jk}(V_j) \leq f_{jk}(V'_j)$ for each $(j,k) \in E$. Hence V cannot be optimal and necessarily $\hat{V}_j := \min_{i \in \delta^-(j)} f_{ij}(\hat{V}_i)$ for every optimal solution. Moreover every solution of the Bellman's equation is feasible in (8). From Lemma 10 the optimum is the solution corresponding to the optimal paths. ∎

The monotonicity property of the functions is therefore fundamental in order to compute optimal paths by exploiting the optimality principle. If a problem can be modeled such that the monotonicity assumption is satisfied, the problem is "well behaved" and dynamic programming can be applied. If the monotonicity assumption is weakly satisfied because the functions are only nondecreasing there may be technical problems (like the spurious solutions) but dynamic programming can still be applied. In these cases the situation can be improved by assuming also superlinearity. However, the fundamental property is monotonicity because, if it fails to be true for just one arc, as in the example below, finding an optimal path may be NP-hard (and the Bellman's equation is useless because the optimality principle does not hold).

Theorem 12: *Finding an optimal path according to (3) is NP-hard for generic functions f_{ij}.*

Proof: Let us consider an instance of the Partition problem with values $a_1, \ldots, a_n$ (let $b := \sum_i a_i/2$) and build the following directed graph $N = \{0, 1, 2, \ldots, n, n+1\}$ and arcs $e_i^0 = (i-1, i)$, $e_i^1 = (i-1, i)$, $i := 1, \ldots, n$, $e_* = (n, n+1)$ (this is actually a multigraph, but the proof can be carried out also on a graph, just insert a node in the arcs e_i^0). Let

$$f_{e_i^0}(V) := V \qquad f_{e_i^1}(V) := V + a_i \qquad f_{e_*}(V) := |V - b|$$

It is immediate to see that there exists a minimal zero cost path if and only if the Partition instance is feasible (each path uses either the arc e_i^1 or e_i^0 and this corresponds to either choosing the i-th number or not, so that the path cost in the node n is equal to the sum of the chosen numbers, and the last arc 'decides' that the best path is the one partitioning best the numbers). ∎

It is immediate to check that the optimality principle does not hold for the example in the proof. Moreover, superlinearity could not play any role in restoring a well behaviour. If we change the cost of the last arc from $|V - b|$ to $|V - b| + b$ the proof remains valid and all functions are superlinear. Just note incidentally that the Partition problem can indeed be solved via Dynamic Programming but with a different formulation, one for which the functions are additive and the optimality principle does hold.

In the backward model we consider directed paths P from a node i to t and the cost of P is defined through functions $g_{ij}(V)$ as

$$V(P^i) = g_{ij}(V(P^j))$$

with P^i the restriction of P from i to t. Hence the recursive computation for the backward model is

$$V(P^t) := \bar{V}^t, \qquad V(P^i) := g_{ij}(V(P^j)) \qquad \forall (i,j) \in P$$

All the previous results can be translated almost verbatim for the backward model. We point out that forward and backward model may represent the same problem only in special cases. By "representing the same problem" we mean that any path $P : s \to t$ can be split into two parts P_k (from s to k) and P^k from k to t and the cost of P can be computed as $V(P_k) + V(P^k)$ with $V(P_k)$ the cost in the forward model, $V(P^k)$

the cost in the backward model, for any k on P. If this is possible then, not surprisingly, the functions f and g are additive with same data, i.e. $f_{ij}(V) = V + c_{ij}$ and $g_{ij}(V) = V + c_{ij}$ (actually we could extend the definition to commutative groups but we prefer not to dwell with such aspects here). This is shown by the following result.

Theorem 13: *Let s and t be two distinguished nodes and let P be a generic path $P : s \to t$. Then the condition $V(P_k) + V(P^k) = V(P_t) = V(P^s)$ is satisfied by any k in P if and only if $f_{ij}(V) = V + c_{ij}$ and $g_{ij}(V) = V + c_{ij}$.*

Proof: Sufficiency is trivial. To show the necessity let us suppose that $V(P_i)+V(P^i) = V(P_j)+V(P^j)$ with (i, j) an arc of P. Then $V(P_i) + g_{ij}(V(P^j)) = f_{ij}(V(P_i)) + V(P^j)$, i.e. $f_{ij}(V(P_i)) = V(P_i) + g_{ij}(V(P^j)) - V(P^j)$. The function f_{ij} cannot depend on the path following node j and so the expression $g_{ij}(V(P^j))-V(P^j)$ must be invariant with respect to P^j and depend only on the arc (i, j), i.e. $g_{ij}(V(P^j)) - V(P^j) =: c'_{ij}$. Similarly $f_{ij}(V(P_i)) - V(P_i) =: c''_{ij}$ and, by comparing the expressions, $c'_{ij} = c''_{ij} =: c_{ij}$. ∎

The Bellman's equation, viewed as a fixed point equation, suggests the algorithm known as the Bellman-Ford algorithm (sometimes also as the Bellman-Ford-Moore algorithm, see Bellman (1958), Ford (1956) and Moore (1957)), where for each arc (i, j) the basic updating assignment $V_j := \min\{V_j; f_{ij}(V_i)\}$ is computed and this assignment sequence is repeated at most n times starting from the initial values $V_s := \bar{V}_s$, $V_j := \infty$. We have:

Theorem 14: *If the functions f_{ij} are increasing the Bellman-Ford algorithm finds all optimal paths or establishes that there are no optimal paths in time $O(n\,m)$.*

Proof: (for minimum problems) Let V_j^k be the value of V_j after the k-th iteration of the algorithm (V_j^0 is the initial value). If $V_j^{k+1} < V_j^k$, then there exists a node i such that $V_j^{k+1} = f_{ij}(V_i^k)$. From the algorithm $V_j^k \le f_{ij}(V_i^{k-1})$ for $k \ge 1$. The three relations imply $f_{ij}(V_i^k) < f_{ij}(V_i^{k-1})$, from which, since the functions are increasing (in fact to be nondecreasing is enough) $V_i^k < V_i^{k-1}$. By applying recursively this reasoning there exists a sequence of nodes $i_0, i_1, \ldots$ such that $V_{i_h}^{k-h+1} < V_{i_h}^{k-h}$. If $k > n$ there must be at least a repeated node in the sequence, say $r := i_{p-q} = i_p$, for which we have

$$V_r^{k-p+q+1} < V_r^{k-p+q} \le V_r^{k-p+1} < V_r^{k-p}$$

The nodes $i_p, i_{p-1}, \ldots, i_{p-q}$ form a directed cycle C. Let us denote by F_C the compostion of the functions f along the cycle. So we have $F_C(V_r) < V_r$. F_C is increasing as well and therefore $F_C(F_C(V_r)) < F_C(V_r) < V_r$. Consequently there exists a sequence of paths $P_r(q)$ each one traversing q times the cycle C and such that $V(P_r(q)) < V(P_r(q-1))$. Therefore there is no optimal solution. However the sequence $V(P_r(q))$ could be bounded and tend to a finite limit.

Therefore if there exists an optimal solution the values V_j cannot be updated after the n-th iteration. The computational complexity statement is proved. However, it remains to prove that the computed values

V_j are indeed optimal. What we have proved is only that they satisfy the Bellman's equation. By using an induction argument let us suppose that at a generic iteration step the values V_j^k satisfy the inequalities $V_j^k \geq \hat{V}_j$ with $\hat{V}_j$ optimal values. Then we have

$$V_j^{k+1} = \min\left\{V_j^k\,;\, f_{ij}(V_i^k)\right\} \geq \min\left\{\hat{V}_j\,;\, f_{ij}(\hat{V}_i)\right\} = \hat{V}_j$$

where we have exploited the fact that the functions are increasing and the optimal values satisfy the Bellman's equation. Hence the inductive hypothesis is true at the next step as well and since it is true at the initial step it must be always true. Then from Lemma 10 the thesis follows. ∎

By weakening the assumption of increasing functions to nondecreasing functions the algorithm might require a higher number of iterations to find an optimal solution. For instance for $f_{st}(V) := \max\{V-1, -3\,K\}$, $f_{t1}(V) := V-1$, $f_{1s}(V) := V-1$, the optimum is found after Kn iterations (a pseudopolynomial value). However, if we assume superlinearity the algorithm works well.

Theorem 15: *If the functions f_{ij} are nondecreasing and superlinear (sublinear) the Bellman-Ford algorithm finds minimum (maximum) paths $s \to i$, for each i, in time $O(n\,m)$.*

Proof: (for minimum) The proof goes as in the previous theorem up to the inequality $F_C(V_r) < V_r$. But in this case the superlinearity assumption rules out the inequality. ∎

However, in the hypothesis of the previous theorem, the Bellman-Ford algorithm is computationally too expensive and we may instead resort to the faster Dijkstra algorithm (Dijkstra (1959)).

Theorem 16: *If the functions f_{ij} are nondecreasing and superlinear (sublinear) the Dijkstra algorithm finds minimum (maximum) paths $s \to i$, for each i, in time $O(n^2)$ or $O(m \log n)$.*

Proof: (for minimum) We just sketch the proof assuming the Dijkstra algorithm is well known. At a generic iteration a set S of nodes ($s \in S$) is known and values V_i, $i \in N$, are computed such that for $i \in S$ the values V_i are the optimal path values from s to i and for $i \notin S$ the values V_i are the optimal path values of paths from s to i which use only nodes in S (beside i). The node k achieving the minimum V_i for $i \notin S$ is added to S and the values V_i, $i \notin S$, are updated. Since $V_k \leq V_i$, $i \notin S$, and the functions are nondecreasing $f_{ik}(V_k) \leq f_{ik}(V_i)$. Since the functions are superlinear $V_k \leq f_{ik}(V_k)$ and therefore

$$V_k \leq f_{ik}(V_k) \leq f_{ik}(V_i)$$

i.e. the value V_k is the final optimal path value for k and the correctness of the algorithm follows. The statement on the complexity is the usual one and depends on the implementation. ∎

For the sake of completeness we state the following known and straigthforward result.

Theorem 17: *If the graph is acyclic the Bellman's equation can be solved in $O(m)$ time.* ∎

3. MINIMUM EXPECTED COST PATHS

According to the results of the previous section a path of minimum expected cost can be computed via the backward Bellman-Ford algorithm with functions (derived from the expression (2))

$$g_e(V) := p_e\, c_e + (1 - p_e)\, V$$

and initial value $\bar{V}^t = 0$. These functions are increasing, so the optimality principle holds. However, since they are not superlinear, there may be no optimal solutions. For any cycle C the composition g_C along the cycle is a function of the type

$$g_C(V) = K + V \prod_{e \in C} (1 - p_e)$$

with $K > 0$. Hence there is always a value $\bar{V}$ such that

$$V < g_C(V) \quad \text{if} \quad V < \bar{V}, \qquad \bar{V} = g_C(\bar{V}), \qquad V > g_C(V) \quad \text{if} \quad V > \bar{V}$$

If there are paths $P^i : i \to t$, $i \in C$, such that $V(P^i) > \bar{V}$, then the values V^i are updated as $V^i := g_C(V^i)$ and converge at linear rate to $\bar{V}$. So there are no optimal solutions because it is more convenient for a path to fold infinitely often on C, where the expected cost is (with probability one within the infinite time horizon) $g_C(\bar{V}) = \bar{V}$ and better than the expected cost $V(P^i)$. The value $\bar{V}$ is close but not equal to the average cost on the circuit

$$\frac{\sum_{e \in C} p_e\, c_e}{\sum_{e \in C} p_e}$$

Indeed the expression $g_C(\bar{V}) = \bar{V}$ made explicit is

$$\sum_{i=1}^{k} p_i\, c_i \prod_{j=1}^{i-1} (1 - p_j) + \bar{V} \prod_{j=1}^{k} (1 - p_j) = \bar{V}$$

having numbered $1, \ldots, k$, the arcs of the circuit (conventionally $\prod_{j \in \emptyset} (1 - p_j) = 1$). Hence

$$\bar{V} = \frac{\sum_{i=1}^{k} p_i\, c_i \prod_{j=1}^{i-1} (1 - p_j)}{1 - \prod_{j=1}^{k} (1 - p_j)} \tag{9}$$

Let us write

$$\Phi_i := 1 - \prod_{j=1}^{i} (1 - p_j) = 1 - \prod_{j=1}^{i-1} (1 - p_j) + p_i \prod_{j=1}^{i-1} (1 - p_j) = \Phi_{i-1} + p_i \prod_{j=1}^{i-1} (1 - p_j), \quad \text{with} \quad \Phi_0 = 0$$

from which we see that $\Phi_k = \sum_{i=1}^{k} \bar{p}_i$ with $\bar{p}_i := p_i \prod_{j=1}^{i-1} (1 - p_j)$. Hence

$$\bar{V} = \frac{\sum_{i=1}^{k} \bar{p}_i\, c_i}{\sum_{i=1}^{k} \bar{p}_i} \tag{10}$$

For small probability values $\bar{p}_i \approx p_i$ and therefore in this case

$$\bar{V} \approx \frac{\sum_{e \in C} p_e\, c_e}{\sum_{e \in C} p_e}$$

The expression (10) can be viewed as a conditional risk, since it represents the expected cost along the circuit given that the accident has occurred. Indeed by repeating for ever the circuit the accident will occurr with probability one.

Note that (10) is not symmetric in the circuit data, in the sense that the values $\bar{p}_i$ depend on the node from which the path enters the cycle. As apparent from the definition of $\bar{p}$, it is more biased toward the first arcs of the cycle (this is very similar to the bias in infinite horizon average cost Markov decision processes). Moreover, the values $\bar{V}$ are different on the various nodes of the circuit. Therefore "optimal paths" will move to the node of the cycle with smallest $\bar{V}$ value and loop for ever in the cycle. The intuitive explanation of this behaviour is clear: if the last part of a trip has a high expected cost (say we have to cross a mine field), the best thing to do is to never get there!

As apparent from (10) the value $\bar{V}$ is a convex combination of costs. Then this anomalous behaviour cannot happen if the probability values are sufficiently low. In this case we normally expect that $V(P^i) < \bar{V}$ and the Bellman-Ford algorithm can be implemented without problems.

Alternatively optimal paths can be computed via linear programming as

$$\begin{aligned} \max \quad & V^s \\ & V^i - (1-p_{ij})\, V^j \le p_{ij}\, c_{ij} \qquad \forall (i,j) \in E \\ & V^t = 0 \end{aligned} \tag{11}$$

with dual

$$\begin{aligned} \min \quad & \sum_{ij} p_{ij}\, c_{ij}\, x_{ij} \\ & \sum_{j \in \delta^+(s)} x_{sj} = 1 \\ & \sum_{j \in \delta^+(k)} x_{kj} - \sum_{i \in \delta^-(k)} (1-p_{ik})\, x_{ik} = 0 \qquad k \ne s, \quad k \ne t \\ & - \sum_{i \in \delta^-(t)} (1-p_{it})\, x_{it} + y = 0 \\ & x_{ij} \ge 0 \end{aligned} \tag{12}$$

As can be immediately seen from the constraints in (12), x_{ij} represents the probability of entering the arc (ij). The last equation is actually redundant, but it is useful because the quantity

$$y = \sum_{i \in \delta^-(t)} (1-p_{it})\, x_{it}$$

is the probability of reaching the final destination. Let us just note that if we change in (12) the objective function into max y, the primal of this new problem is

$$\begin{aligned} \min \quad & V^s \\ & V^i \ge (1-p_{ij})\, V^j \qquad \forall (i,j) \in E \\ & V^t = 1 \end{aligned}$$

with V^i the maximum probability of reaching the final destination starting from i. This is the linear programming formulation of the problem shown in the Example 3 in backward form.

4. COMPUTING PARETO OPTIMA

In this section we suppose that along with the objective of minimizing the environmental impact cost we wish at the same time to minimize the travel distance. Hence we need to find nondominated (i.e. Pareto optimal) paths with respect to two objectives. The model (12) cannot be used because the path lengths cannot be expressed through the x_{ij} variables which represent path probabilities rather than paths.

We shall see later that a direct approach to the problem of finding Pareto optima is indeed possible. However, we want to investigate now if we can still make use of (12) after possibly changing our approach to the problem. If d_{ij} is the length of the edge (ij), the quantity $\sum_{(ij)\in E} d_{ij}\, x_{ij}$ is the expected path length. Are we allowed to take into account expected lenghts instead of actual lengths? On one hand we expect, optimistically, to finish each travel without accidents and we are therefore inclined to consider actual lengths. On the other hand we do consider the possibility of an accident when we use expected costs to decide the path to be taken. So considering expected path lenghts is consistent with the modelling of expected path costs.

Hence, given a path P, its expected cost $c(P)$ is computed from (1) and its expected distance is similarly computed as

$$d(P) := d_1 + (1-p_1)\, d_2 + (1-p_1)\,(1-p_2)\, d_3 + \ldots = d_1 + (1-p_1)\, d(P\backslash e_1) \tag{13}$$

Let K be the convex hull in $\mathbb{R}^2$ of the sets $\{(u_1, u_2) : u_1 \geq c(P),\, u_2 \geq d(P)\}$. Given two paths P and P', P is said to dominate P' if $c(P) \leq c(P')$ and $d(P) < d(P')$ or $c(P) < c(P')$ and $d(P) \leq d(P')$. The Pareto optima are by definition those paths which are not dominated by any other path. The Pareto optima can be scanned by solving $\min_P c(P)$ subject to the constraint $d(P) \leq D$ for different values of D. Among the Pareto optima found this way there are those which do not lie on the boundary of K.

It is tempting to add to (12) a constraint of the type $\sum_{(ij)\in E} d_{ij}\, x_{ij} \leq D$ and change D in order to explore alternative Pareto optima. It should be clear however that exploring the Pareto set this way is different from minimizing $c(P)$ subject to the constraint $d(P) \leq D$. The linear programming model

$$
\begin{array}{lll}
\min & \displaystyle\sum_{ij} p_{ij}\, c_{ij}\, x_{ij} & \\
& \displaystyle\sum_{j\in\delta^+(s)} x_{sj} & = 1 \\
& \displaystyle\sum_{j\in\delta^+(k)} x_{kj} \quad - \sum_{i\in\delta^-(k)} (1-p_{ik})\, x_{ik} & = 0 \qquad k\neq s, \quad k\neq t \\
& \qquad\qquad - \displaystyle\sum_{i\in\delta^-(t)} (1-p_{it})\, x_{it} + y & = 0 \\
& \displaystyle\sum_{ij} d_{ij}\, x_{ij} & \leq D \\
& x_{ij} \geq 0 &
\end{array}
\tag{14}
$$

computes only Pareto optima on the boundary of K and their convex combinations as is clear from its dual where arc costs are linear combinations of expected cost and distance (compare with (11)):

$$
\begin{array}{ll}
\max & V^s - D\, W \\
& V^i - (1-p_{ij})\, V^j - d_{ij}\, W \leq p_{ij}\, c_{ij} \qquad \forall (i,j)\in E \\
& V^t = 0
\end{array}
\tag{15}
$$

Solving (14) does not necessarily yield solutions x_{ij} which are positive only along a path and therefore identify uniquely a path. There can be nodes with two or more outgoing arcs with positive x_{ij} values and the fact that a unique path cannot be identified seems a drawback. However, if we agree that paths can be chosen randomly, we may decide the actual path by selecting the outgoing arc from a node with probability proportional to x_{ij}. The expected cost and the expected distance of this random path are indeed $\sum_{ij} p_{ij}\, c_{ij}\, x_{ij}$ and $\sum_{ij} d_{ij}\, x_{ij}$ respectively. The idea of selecting a solution randomly and considering its expected values is very similar to the concept of mixed strategies in game theory (see Luce and Raiffa (1957)) and also to Markov decision processes with constraints across the states (see Puterman (1994)).

Mixed solutions of this type can dominate pure Pareto optima. Let us consider a simple example. There are four nodes $(s, 1, 2, t)$ and arcs $(s,1)$, $(1,t)$, $(s,2)$, $(2,t)$, (s,t). So there are three paths from s to t, namely $P^1 = s \to 1 \to t$, $P^2 = s \to 2 \to t$ and $P^3 = s \to t$. The data are

	d	c	p
$(s,1)$	20	10	0.1
$(1,t)$	20	10	0.1
$(s,2)$	10	20	0.1
$(2,t)$	10	20	0.1
(s,t)	30	30	0.1

from which we directly compute $d(P^1) = 38$, $c(P^1) = 1.9$, $d(P^2) = 19$, $c(P^2) = 3.8$, $d(P^3) = 30$, $c(P^3) = 3$. All three paths are Pareto optima and $(d(P^3), c(P^3))$ is inside K. If we solve (14) setting $D := 28.5$ we find the solution

$$
x_{s1} = 0.5, \qquad x_{1t} = 0.45, \qquad x_{s2} = 0.5, \qquad x_{2t} = 0.45, \qquad x_{st} = 0, \qquad y = 0.81
$$

with expected cost 2.85 and expected distance obviously 28.5. This solution dominates P^3. We observe incidentally that pure solutions giving raise to a mixed solution are not necessarily arc disjoint (in the example just add one arc in front of the three parallel paths).

We may also consider that if the transportation of hazardous materials has to be repeated in time we might be more interested in finding sets of alternative paths rather than just one path and changing path each time according to some rule. Changing paths may be perceived by the public opinion as less dangerous than using always the same path because it spreads and decreases the risk of accident on a larger area. Then instead of selecting randomly arcs we may think of switching paths in accordance with the x_{ij} values. For instance the solution of the previous example can be implemented by switching between P^1 and P^2 for each travel.

The possibility of having explicit the value y can be used to control also the probability of reaching destination without accidents. For instance we might add the constraint $y \geq q$ to (14). If we do so for the previous example and set $y \geq 0.85$ we obtain

$$x_{s1} = 0.242, \qquad x_{1t} = 0.218, \qquad x_{s2} = 0.313, \qquad x_{2t} = 0.281, \qquad x_{st} = 0.445, \qquad y = 0.85 \qquad (16)$$

with expected cost 2.98 and expected distance 28.5. Here the paths P_1, P_2 and P_3 are selected with probabilities 0.242, 0.313, 0.445 respectively.

Furthermore, once we have accepted the idea that alternative paths may be used, we may consider important also to keep low, uniformly for each arc e, some measure of risk. For instance we might impose $x_e \leq K$, meaning that each arc will be used not more than a certain percentage of times. Alternatively we may either impose $p_e\, x_e \leq K$, lowering the accident probability on each arc, or $p_e\, c_e\, x_e \leq K$, reducing the expected cost of damage on each arc. In all cases the effect will be of spreading the travelling on more arcs (however at higher global cost and distance).

If on the contrary we want to consider paths (and not path probabilities) and their actual lengths, we have to take another approach. The problem of computing all nondominated paths for bicriterion problems has been addressed by Hansen (1980). As noted by Hansen (1980) computing all nondominated paths is difficult in general for the simple reason that there can be exponentially many nondominated solutions. However, as shown in Serafini (1987), even finding one nondominated biobjective path is NP-hard (in the sense of finding solutions not worse than stated goals). The problem can be solved in pseudopolynomial time (as proposed by Hansen (1980) and Serafini (1987)) by making D copies of the graph (with $D := n \max d_e$) with nodes labeled as (i,k) (k-th copy of node i) and arcs $(i,k) \rightarrow (j, k - d_{ij})$ for each original arc (i,j) (backward model). Then the min expected cost paths are computed from (s,k) to $(t,0)$, (one run of a dynamic programming algorithm) and the nondominated solutions are found among these paths. Since the multiple graph is acyclic the algorithm has complexity $O(m\, D)$ and therefore is pretty fast if the edge distances can be coarsely discretized.

Alternatively we may think of using the fundamental updating step of the dynamic programming ap-

proach and adapting it to the bicriterion case. Let us suppose that in each node i a list of temporary nondominated values are stored. This list consists of pairs of values $(V_1^i(k), V_2^i(k))$ with $V_j^i(k)$ the value of the objective j of the k-th path $i \to t$ in the list). The list can be sorted in order of increasing values $V_1^i(k)$, what implies that the values $V_2^i(k)$ are sorted as well in reverse order (otherwise they would be dominated). Updating this list means merging it with the list $(g_{ij}^1(V_1^j(h)), g_{ij}^2(V_2^j(h)))$ and discarding the dominated paths. This can be done with time linear in the list lengths (just one scan).

We have to be careful that a path can be present in both lists. If we discard also solutions which share the same objective values of a previously found solution then repeated solutions are automatically eliminated. One might argue that this way we loose track of interesting solutions. However, this point of view is consistent with the usual situation in single objective optimization where only one solution is produced among possibly many optimal solutions.

The complexity of this approach depends on the number of nondominated solutions. Let S be an upper bound on the number of nondominated solutions for each node i. Then, since we use the Bellman-Ford algorithm, the complexity is $O(n\,m\,S)$. We may note that $S \leq D$ and therefore this approach is recommended with respect to the previous one only if $S \ll D$ or it is not possible to discretize the distance values.

We must also reconstruct the paths at the end of the algorithm. Simple pointers to the nodes are not enough. We must also point to the element in the list. Therefore the list has more information and in node i we have to store four values for the k-th path of the list, namely $(V_1^i(k), V_2^i(k), p^i(k), q^i(k))$ where $p^i(k)$ is the successor node of i of the k-th path in the list of node i and $q^i(k)$ is the element of the list in node $p^i(k)$ corresponding to the k-th path. The h-th nondominated path $s \to t$ can therefore be reconstructed as

$$P := \{s\};\ r := s;\ k := h;\ \textbf{while}\ r \neq p^r(k)\ \textbf{do}\ \{r := p^r(k); k := q^r(k);\ P := P \cup r\}$$

The algorithm is initialized with empty lists except node t where we have only one element, namely $(\bar{V}_1^t, \bar{V}_2^t, t, 1)$ (never updated unless there are improving cycles).

5. CONCLUSION

We have addressed two main issues. The first is an extension of dynamic programming needed to model a larger class of problems. The study of the link between the dynamic programming principles and the path cost generalization is novel and casts a new light on the way dynamic programming works. These ideas have been applied to the problem of finding the minimum expected cost of an accident for the travelling of hazardous materials. It has been shown that there could be no optimal solution with high accident probability values. In case of existence of optimal solutions a linear programming formulation, derived from the dynamic programming model, can be fruitfully used to model also a biobjective version of the problem. This formulation has suggested the implementation of mixed strategies in the selection of optimal paths. If mixed strategies cannot be considered a special dynamic biobjective algorithm has been proposed.

6. REFERENCES

Baker, K.R., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1983), "Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints", *Operations Research*, **31**, 381-386.

Bellman, R. (1958), "On a routing problem", *Quarterly Applied Mathematics*, **16**, 87-90.

Cooke, K.L., and E. Halsey (1966), "The shortest route through a network with time dependent internodal transit times", *Journal of Mathematical Analysis and Applications*, **14**, 493-498.

Dijkstra, E.W. (1959), "A note on two problems in connexion with graphs", *Numerische Mathematik*, **1**, 269-271.

Erkut, E., and V. Verter (1998), "Modeling of transport risk for hazardous materials", *Operations Research*, **48**, 624-642.

Ford, L.R. (1956), "Network flow theory", Technical report P-923, The Rand Corporation.

Gondran, M., and M. Minoux (1979), *Graphes et algorithmes*, Editions Eyrolles, Paris, France.

Halpern, J. (1977), "The shortest route with time dependent length of edges and limited delay possibilites in nodes", *Zeitschrift für Operations Research*, **21**, 117-124.

Halpern, J., and I. Priess (1974), "Shortest paths with time constraints on movement and parking", *Networks*, **4**, 241-253.

Hansen, P. (1980), "Bicriterion path problems", in *Multiple Criteria Decision Making: Theory and Applications*, G. Fandel and T. Gol (eds), Lecture Notes in Economics and Mathematical Systems, 177, Springer, Berlin.

Luce, R.D., and H. Raiffa (1957), *Games and decisions: introduction and critical survey*, Wiley, New York, NY.

Minoux, M. (1976), "Structures algébriques généralisées des problèmes de cheminement dans les graphes: Théorèmes, algorithmes at applications", *R.A.I.R.O. - Recherche opérationnelle*, **10**, 33-62.

Moore, E.F. (1957), "The shortest path through a maze", in *Proceedings of the International Symposium on the Theory of Switching (1957)*, Harvard University Press 1959.

Puterman, M.L. (1994), *Markov decision processes: discrete stochastic dynamic programming*, Wiley, New York, NY.

Serafini, P. (1987), "Some considerations about computational complexity for multi objective combinatorial problems", in *Recent advances and historical developments of vector optimization*, W. Krabs e J. Jahn eds, p. 222-232, Springer Lecture Notes in Economics and Mathematical Systems.