Optimal Paths in Graphs with Stochastic or Multidimensional Weights

Ronald Prescott Loui
Computer Science Department
The University of Rochester
Rochester, NY 14627

TR115
Winter 1982

Abstract

This paper formulates a stochastic and a multidimensional optimal path problem, each as an extension of the shortest path problem. Conditions when existing shortest path methods apply are noted. In each problem instance, a utility function defines preference among candidate paths.

The major result of each formulation is the ability to state explicit conditions for exact solutions using standard methods, and the applicability of well-understood approximation techniques.

In the non-deterministic problem there is a dynamic programming solution when utility functions have constant Arrow-Pratt risk aversion, and an equivalence to the multidimensional problem when utility functions are quadratic, or when they are polynomial and edge weights have Gaussian or Gamma distributions. For the multidimensional problem we modify Dijkstra's 1959 algorithm, and employ dominance in a partial ordering.

Key Phrases: shortest path, stochastic, multidimensional, discrete, combinatorial optimization, operations research, algorithms.

1. Introduction.

The classic problem of the shortest path has never been successfully extended to graphs with non-deterministic or multidimensional weights. When the weights on edges become random variables or vectors, the dynamic programming principle no longer holds for general criteria of optimality; it is no longer generally true that the optimal path consists of optimal subpaths. Since the dynamic programming principle has been the basis for almost all work on optimal path problems, treatments of the extended problems have been unable to exploit previous results.

The major result for the shortest path problem is the so-called "triple operation,"[1] the implementation of the dynamic programming principle that allows combinatorial reductions in the search space. In fact, the reason to study the problem is precisely for its elegant avoidance of combinatorial explosion. In a graph of V vertices, there could be as many as $\lfloor (V-2)!e \rfloor$ paths[2] connecting any two vertices. Yet an algorithm by Dijkstra[9] runs in $O(V^2)$ time and other algorithms have even faster expected behavior (e.g., Spira[29], Bloniarz[6]). Herein lies the essential value of the shortest path problem. An extended study of optimal path problems following the spirit of the shortest path paradigm should not only provide a problem formulation that is consistent, but also describe solutions addressing the combinatorially unwieldy size of the search.

The extended problems deserve study. Applications of the stochastic and multidimensional path problems abound; the model applies to any route-selection problem wherein uncertainty or multiple factors bear on preference. Currently, transshipment and routing problems require tailoring to fit the traditional shortest path model. The assignment of single, constant costs to edges has been necessary, whether it permits an adequate representation or not. This paper explores alternatives to the traditional representation. Furthermore, given the central role the shortest path has played, it seems that its stochastic and

multidimensional variants would aptly serve as examples for other stochastic or multidimensional combinatorial optimization problems.

2.1. Introduction to the Stochastic Problem.

With the notion of preference yet to be determined, the stochastic problem can be stated as:

> Given a weighted digraph $G = (V=\{v_i\}, E=\{e_{ij}\}, \underline{W}=|\tilde{w}(e_{ij})|)$,
> of $|V| = V$ vertices and $|E|$ edges, where weights on edges $\tilde{w}(e_{ij})$
> are independent, real-valued random variables with values in $[0,\infty)$
> and with known distributions F_{ij},
>
> identify the most preferred path λ_{1V}^*, in the set of all the paths
> from vertex v_1 to vertex v_V, L_{1V}.

The weight of a path is also a random variable, $\tilde{w}(\lambda_{1V}) = \sum_{e_{ij}\in\lambda_{1V}} \tilde{w}(e_{ij})$, the sum of the weights on the edges that compose the path.

This problem is identical to the unit demand, unit capacity flow problem with uncertain costs. It is similar to the unsolved probabilistic PERT (longest path) problem. However, unlike the PERT problem where bounds on the weight of the optimal path are sufficient, it is a search problem; the applications of the model require the determination of an actual path in L_{1V}. Though techniques developed for probabilistic PERT (e.g., Shogan[27], Nadas[24]) can be extended to bound the distibution of the least weight of all paths in L_{1V}, these techniques do not solve the selection problem. Moreover, the bounded distribution would not refer to $\tilde{w}(\lambda_{1V})$ for any $\lambda_{1V} \in L_{1V}$; rather, it would describe the behavior of $\tilde{w} \equiv \min_{\lambda_{1V}\in L_{1V}}(\tilde{w}(\lambda_{1V}))$.[3]

Standard works that have encountered the stochastic shortest path problem have avoided it by taking expectations of edge weights and solving the ensuing deterministic problem

(e.g. Dantzig[8], Howard[15], Kleinrock[18]):

$$\lambda_{1V}^{*} \equiv \underset{\lambda_{1V} \in L_{1V}}{\text{argmin}}\ E\{\tilde{w}(\lambda_{1V})\} \qquad \text{(P1a)}$$

However, the identified path can be potentially "risky;" it may have a high probability of realizing a much larger weight than expected. If there were a path of slightly larger expected weight with little probability of realizing very large weights, it might conceivably be preferable. In short, the solution ignores higher moments; it makes no account of "risk."

But it's not clear how best to account for risk. Frank[12] proposes the following condition of path optimality: For a specified k, consider the path that maximizes the probability of realizing a weight less than k as the optimal path;[4] i.e.,

$$\lambda_{1V}^{*} \equiv \underset{\lambda_{1V} \in L_{1V}}{\text{argmax}}\ [\ \text{Prob}\{\tilde{w}(\lambda_{1V}) < k\}\ ] \qquad \text{(P1b)}$$

Sigal, Pritsker, and Solberg[28] suggest a different condition: optimality entailing the greatest probability of realizing the least weight.

$$\lambda_{1V}^{*} \equiv \underset{\lambda_{1V}^{x} \in L_{1V}}{\text{argmax}}\Big(\prod_{\lambda_{1V}^{y} \in L_{1V}} [\ \text{Prob}\{\tilde{w}(\lambda_{1V}^{x}) \leq \tilde{w}(\lambda_{1V}^{y})\}\ ]\ \Big) \qquad \text{(P1c)}$$

They offer a cutset approach to reducing the number of arguments requisite in the continuous product. However, no solution addressing the combinatorially large size of L_{1V} is known to either problem; both appear to require exhaustive search.

2.2. Other Approaches.

There do exist reasonable criteria of optimality that admit dynamic programming solutions. Given a constant $\alpha \in [0,1]$, for instance, the Hurwicz principle prefers the path that minimizes $\alpha w_L + (1-\alpha) w_U$, where w_L and w_U are, respectively, the lower and upper bounds on the values realizable by each path's weight.

$$\lambda_{1V}^{*} \equiv \underset{\lambda_{1V} \in L_{1V}}{\text{argmin}} [\, \alpha w_L(\lambda_{1V}) + (1-\alpha) w_U(\lambda_{1V}) \,] \qquad \text{(P1d)}$$

where $w_L(\lambda_{1V}) \leq \tilde{w}(\lambda_{1V}) \leq w_U(\lambda_{1V})$.

For $\alpha = 1$, this criterion expresses complete optimism, and for $\alpha = 0$, the criterion is completely pessimistic. λ_{1V}^{*} is obtained by solving the deterministic shortest path problem in the graph $G' = (V, E, \underline{W}')$, where, in the matrix of weights, each element $\tilde{w}(e_{ij})$ has been replaced by $[\, \alpha w_L(e_{ij}) + (1-\alpha) w_U(e_{ij}) \,]$.

Of course, the Hurwicz principle is not widely used. And it does not produce well-defined problems when each edge has a finite probability of realizing an infinite weight (i.e., a chance of "failing"), as in the class of problems studied by Mirchandani[23].

Alternately, there is a linear program that corresponds to the stochastic shortest path problem:

$$\text{maximize} \left(\sum_{i=2}^{V} u_i \right)$$

$$\text{s.t. } u_1 = 0$$

$$u_j - u_i \leq \tilde{w}(e_{ij}) \qquad \text{for } i = 1,\ldots V;\ j = 2,\ldots V;\ i \neq j \ .^{5}$$

The standard programming approach would be to convert each of the $(V-1)^2$ uncertain constraints into chance constraints, requiring compliance with probability at least $1-\beta$. For each edge, e_{ij}, define

$$w_\beta(e_{ij}) : \ \text{Prob}\{\, \tilde{w}(e_{ij}) \geq w_\beta(e_{ij}) \,\} = \beta \, .$$

Now the constraints have the form

$$u_j - u_i \leq w_\beta(e_{ij}) \ .$$

The solution to this new program determines the path

$$\lambda_{1V}^{*} \equiv \underset{\lambda_{1V} \in L_{1V}}{\text{argmin}} \left[\sum_{e_{ij} \in \lambda_{1V}} w_\beta(e_{ij}) \right] , \qquad \text{(P1e)}$$

and this path is just the shortest path in the graph with new weights $w_\beta(e_{ij})$.

In practical instances of the stochastic problem, this approach may suffice. But paths of differing cardinality are being compared at different levels of compliance. The weight of a

path composed of n edges is correctly bounded by its new weight with probability $(1-\beta)^n$; the probability of compliance decreases with n. In this sense, the solution is biased toward paths with many edges.

2.3. The Decision Analytic Formulation.

The expected utility criterion of Bernoulli, von Neumann, and Morgenstern is the prevalent and most comprehensive for preference under uncertainty. The results of this paper suggest that it is also the most useful.

In addition to the weigted digraph, G,

define a utility function u(x), $u:\mathbb{R}^1 \rightarrow \mathbb{R}^1$, monotonically decreasing in x.

The utility of a random variable $\tilde{x}$ is defined to be its expected utility, $u(\tilde{x}) \equiv E_x u(\tilde{x})$. Now formulate the stochastic problem as above with preference implied by utility;

$$\lambda_{1V}{}^x > \lambda_{1V}{}^y \Leftrightarrow u(\tilde{w}(\lambda_{1V}{}^x)) > u(\tilde{w}(\lambda_{1V}{}^y)) \; ;$$

$$\lambda_{1V}{}^* \equiv \underset{\lambda_{1V} \in L_{1V}}{\operatorname{argmax}} [\, u(\tilde{w}(\lambda_{1V})) \,] \qquad \text{(P1)}$$

Note that if all edge-weight densities were to have point support, (P1) with any strictly monotone u(x) would define the traditional shortest path problem.

For general u(x) and general F_{ij}, the dynamic programming principle is violated. Consider the two paths that access vertex v_4 from v_1 in the graph in figure 1., $\lambda_{14}{}^x = e_{12}e_{24}$, and $\lambda_{14}{}^y = e_{13}e_{34}$. Random variable $\tilde{w}(\lambda_{14}{}^x)$ has marginal density $f_{12}{}^* f_{24}$, and $\tilde{w}(\lambda_{14}{}^y)$ has density $f_{13}{}^* f_{34}$, as shown in figure 3. Suppose the utility function behaves linearly until just beyond the support of each density function, at which point it turns down sharply (fig. 2.). $\tilde{w}(\lambda_{14}{}^x)$ has a broader density than $\tilde{w}(\lambda_{14}{}^y)$, but the former has a smaller mean. For a

combinatorially reductive search algorithm to reduce the number of candidate paths through v_4 without elaborate knowledge of the graph,

$$\lambda_{14}{}^x > \lambda_{14}{}^y \text{ must imply } \lambda_{14}{}^x e_{45} > \lambda_{14}{}^y e_{45} ,$$

for all admissible densities f_{45} of the weight $\tilde{w}(e_{45})$ on edge e_{45}. I.e., if path λ^x is better than path λ^y at v_4, one may discard λ^y as a candidate subpath of the optimal path in deference to λ^x under the condition that extensions of λ^y never become better than the same extensions of λ^x. In particular, λ^y cannot be preferred to λ^x at v_5.

But if f_{45} has point support, $f_{45} = \delta_c$ (fig. 4.), then the wider spread of $f_{12}{}^* f_{24}{}^* f_{45}$ would cause its expected-utility integral to be smaller than the expected-utility integral of $f_{13}{}^* f_{34}{}^* f_{45}$. Therefore, in this case,

$$\lambda_{14}{}^x > \lambda_{14}{}^y \text{ does not imply } \lambda_{14}{}^x e_{45} > \lambda_{14}{}^y e_{45}.$$

figure 1. A simple graph.

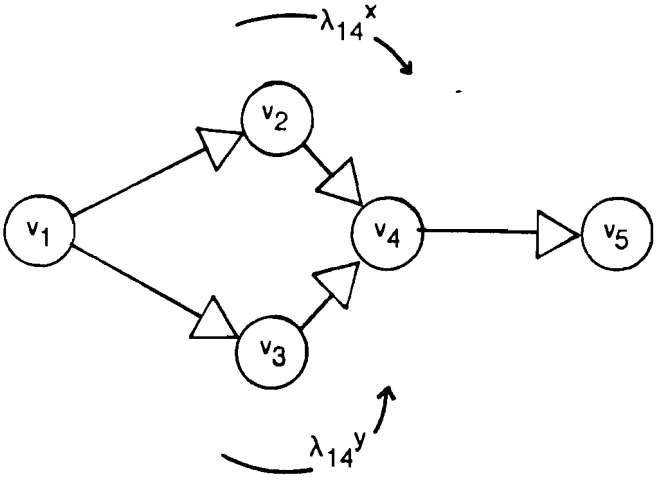


figure 2. A utility function that does not allow a dynamic programming solution for general F_{ij}.

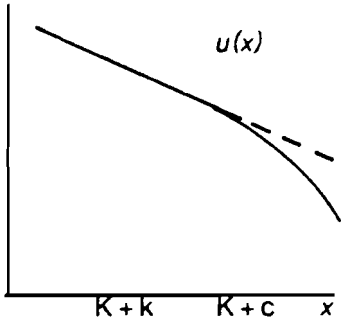


figure 3. Probability densities for the weights of two paths that access v_4.

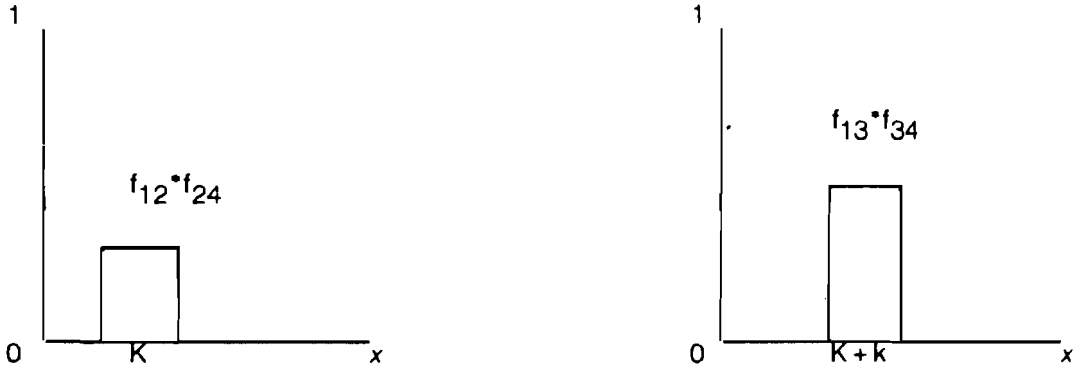
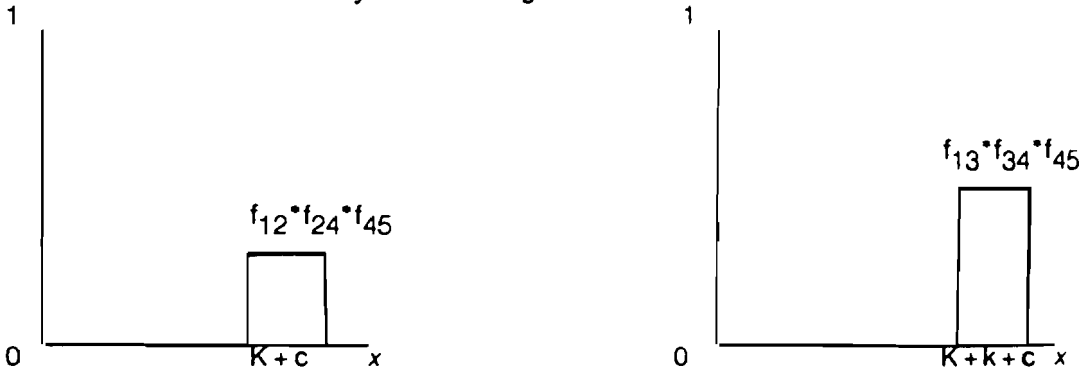


figure 4. Probability density functions for the paths that are extensions of the paths discussed above. Compare these graphs with the utility function in figure 2.



2.4 Necessary Conditions on u(x).

The class of utility functions that permits dynamic-programming-directed search is quite restricted. For general F_{ij}, the dynamic programming principle is satisfied if and only if u(x) is affine linear or exponential.

This restriction is easily shown. In fact, it was essentially shown by Howard and Matheson in a slightly different context [16]. Dynamic programming requires, for all density functions f, g, and h, consisting of convolutions of the density functions for edge weights in the graph,

$$[\int u(x)f(x)dx > \int u(x)g(x)dx] \Leftrightarrow [\int u(x)f*h(x)dx > \int u(x)g*h(x)dx] \qquad (1)$$

(integrals taken over all space). If we can invoke Fubini's theorem (e.g., all integrals exist and are finite), then (1) is the same as

$$[\int u(x)f(x)dx > \int u(x)g(x)dx] \Leftrightarrow [\int h(\zeta)\int u(x)f(x-\zeta)dx\,d\zeta > \int h(\zeta)\int u(x)g(x-\zeta)dx\,d\zeta]. \quad (2)$$

With a change of variables, the latter predicate is equivalent to

$$[\int h(\zeta)\int u(x+\zeta)f(x)dx\,d\zeta > \int h(\zeta)\int u(x+\zeta)g(x)dx\,d\zeta].$$

For this to imply and be implied by $[\int u(x)f(x)dx > \int u(x)g(x)dx]$, u's behavior under translation is restricted to $u(x+\zeta) = \gamma(\zeta)u(x) + \varphi(\zeta)$. So if u has its first two derivatives, they have the forms:

$$u'(x) = k_1u(x) + k_2;$$

$$u''(x) = k_1u'(x).$$

Hence, u(x) will have to be affine linear or exponential.

In utility theory, this restriction on u(x) is exactly the restriction to constant Arrow-Pratt absolute risk aversion:

$$-u''(x)/u'(x) \text{ constant.}$$

This exact correspondence of restrictions is fortunate for subtle reasons. "Constant aversion to risk," as an unquantified description of an agent 's propensity to gamble, is exactly the intuitive constraint on the kind of separability required to implement dynamic programming. The Arrow-Pratt measure just happens to be the correct quantification of risk in this application. One can certainly imagine another measure of risk aversion, the constancy of which does not correctly capture the required separability aspect [2]. So previous work on this particular measure of risk aversion (see [26]) can be brought to bear on problems formulated with the Bernoulli-von Neumann-Morgenstern criterion.

Some stochastic problems can be restated as multidimensional problems. These problems will be discussed later in this paper.

3.1. Introduction to the Multidimensional Problem.

Although work has been done on multi-commodity flow with linear objectives and on the multidimensional knapsack, the literature on graphs with multiple edge-weights is impoverished. The only original result in multidimensional path problems is an iterative procedure of Christofides[7] that reduces the problem:

$$\text{find } \lambda_{1V}^{*} \equiv \underset{\lambda_{1V}\in L_{1V}}{\text{argmax}}\ \varphi\Big(\max_{e_{ij}\in\lambda_{1V}} \theta(e_{ij}),\ \eta(\lambda_{1V})\Big) \qquad \text{(P2a)}$$

into the problem of finding

$$\underset{\lambda_{1V}\in L_{1V}}{\text{argmax}}\ \eta(\lambda_{1V}),$$

when φ is monotonic. He applies his procedure to the "greatest expected capacity" problem in $O(V^4)$, where each edge e_{ij} has a capacity $c(e_{ij})$ and a reliability $\rho(e_{ij})$, and one seeks

$$\underset{\lambda_{1V}\in L_{1V}}{\text{argmax}}\ \Big[\ \min_{e_{ij}\in\lambda_{1V}} (c(e_{ij}))\ \cdot \prod_{e_{ij}\in\lambda_{1V}} \rho(e_{ij})\ \Big]\ .$$

Because the result applies only to problems with extremizations over edges in paths, the procedure does not enjoy wide use.

Lawler[19] obtained a result for doubly-weighted *cycles*. He uses existing shortest path methods to devise an ε-approximate algorithm to determine the path with weights of maximal ratio. However, his result generalizes neither to problems of simple paths, nor to problems with different objectives.

We pose a general multidimensional optimal path problem in the same decision-analytic form used above for the stochastic problem (P1). The use of utility functions to define preference has a natural multidimensional generalization. In fact, the modern interest in utility functions has been primarily for their ability to make explicit the trade-offs between various factors in determining preference.

Given a weighted digraph $G = (V, E, \underline{w})$, where weights on edges $w(e_{ij})$ are d-dimensional vectors with each element real non-negative, and a preference function $u(x):\mathbb{R}^d \rightarrow \mathbb{R}^1$, monotonically decreasing in each x_i for fixed x_{-i},

identify the path $\lambda_{1V}^* \in L_{1V}$,

the weight of which, $w(\lambda_{1V}) \equiv \sum_{e_{ij} \in \lambda_{1V}} w(e_{ij})$,

maximizes u;

$$\lambda_{1V}^* \equiv \underset{\lambda_{1V} \in L_{1V}}{\text{argmax}} \left[u(w(\lambda_{1V})) \right] . \qquad \text{(P2)}$$

Note that when d = 1, the problem is identical to the shortest path problem for strictly monotonic u.

3.2. Induction via Dynamic Programming and via Dominance.

For affine linear u, $u(x) = c \cdot x^t + k$, and lexicographic u, there exists a dynamic programming solution to the problem. However, the problem does not admit such a solution in general. For instance, let $u(x) = -x_1x_2$; $w(\lambda_{14}^x) = (1,10)$; $w(\lambda_{14}^y) = (3,4)$; and $w(e_{45}) = (5,5)$. At v_4, λ_{14}^x is preferred to λ_{14}^y; $u(1,10) > u(3,4)$. But its extension to v_5 is inferior, $u(6,15) < u(8,9)$.

Nevertheless, the search for λ_{1V}^* need not require the consideration of all paths; one may perform a limited induction via dominance. The paths that access a vertex are in general partially ordered. In the problem (P2),

if $(w(\lambda_{1n}^x))_i \leq (w(\lambda_{1n}^y))_i$, $\forall i$, and $(w(\lambda_{1n}^x))_j < (w(\lambda_{1n}^y))_j$ for at least one j,

then λ^x dominates λ^y, $\lambda^x > \lambda^y$.[6]

This relation is asymmetric and incomplete, so some paths remain incomparable:

$$(\lambda^x \not> \lambda^y \text{ and } \lambda^y \not> \lambda^x) \Rightarrow \lambda^x \sim \lambda^y.$$

For those paths that are comparable, $\lambda^x > \lambda^y$ guarantees that $u(w(\lambda^x\lambda^z)) > u(w(\lambda^y\lambda^z))$ for all λ^z, since u is monotonic.

Dynamic programming works when paths can be well-ordered. At each vertex, all but the unique [7] maximal element are discarded, and only extensions of this "optimal subpath" need subsequently be considered. When the paths are partially ordered, only dominated paths can be discarded; the remaining paths, the maxima, must all be considered for extension. The search for λ_{1V}^* reduces not to finding optimal subpaths, but to finding all maxima that could be subpaths.

One could determine these maxima, these sets of undominated paths associated with each vertex, by successive approximation (the original Ford-Moore-Bellman idea for shortest paths). For at most V – 1 iterations, current members of the sets are extended, possibly causing modifications in the sets at other vertices. Such a strategy could be efficient in acyclic graphs. But in cyclic graphs, some paths that have been extended will later be dominated. A properly designed algorithm can avoid this unnecessary work, and the sorting and iterating associated with it, at the cost of maintaining a much larger sorting structure.

Dijkstra's fundamental contribution[8] to the shortest path problem was a rule that properly ordered the construction of optimal subpaths. The optimal subpaths, λ_{1n}^*, n = 1, . . ., V (one for each vertex in the graph), shall be constructed in order of increasing weight. If this rule is obeyed, then (i) no optimal subpath to any vertex need be reconsidered once constructed, and (ii) the construction of each such path need only consider the extensions of optimal subpaths already constructed.

The multidimensional analogue of this rule serves as a basis for an algorithm solving (P2).

V sets, one for each vertex, contain the maxima that have been discovered. Once a path is entered into a vertex-maxima set, it cannot be removed; entry into the set is permanent. A heap is maintained, containing some of the extensions of all the discovered maxima. A given path can be found in this heap if (i) it is an extension of a discovered maximum, (ii) the set of maxima at the vertex it accesses did not contain a maximum that dominated it at the time it was placed in the heap, and (iii) it has not yet been removed from the heap.

When a path is to be selected from the heap, an arbitrary choice is made between those paths that are not dominated by any other path in the heap. There are several ways to structure the heap such that the root is guaranteed not to be so dominated: e.g., lexicographically and decreasing in each of the d elements of path weight, or decreasing in the sum of the elements.

The selected path is compared with the maxima already discovered for the vertex it accesses. If it is dominated by one of these maxima, it is discarded. Otherwise, it must be a maximum, so it is entered into the vertex's set of maxima, and its extensions are considered for entry into the heap. Paths are selected and processed in this way until the heap empties.

When no paths remain in the heap, the set of maxima for v_V is searched for the path with weights that maximize u(x).

Algorithm (following Dijkstra).

1. H is a heap of paths structured on the weights of these paths, either

   a. lexicographically and decreasing in each element, or

   b. on $-\sum_{i=1}^{d}(w(\lambda))_i$, the negative sum of all the elements in the vector.

   Initially, H contains only λ_{11}, the weight of which is 0.

2. $S_1, \ldots, S_V$ are sets associated with vertices $v_1, \ldots, v_V$ respectively. These sets contain the maximal paths to each vertex, as they are discovered. Initially, all sets are empty.

3. a. **While** (H is not empty) **do begin**

   b. **remove the root of** H, λ_{1n}; // note that H is structured to guarantee // // that $(\nexists \lambda \in H)(\lambda > \lambda_{1n})$. //

   c. **if** $(\nexists \lambda \in S_n)(\lambda > \lambda_{1n})$ **then begin** // If this path is not yet dominated at the //

   d. $S_n \leftarrow S_n \cup \{\lambda_{1n}\}$; // vertex it accesses, it must be a maximum. //

   e. **if** $(n \neq V)$ **then begin**

   f. **for each** $(m)((m \geq 2)\&(e_{nm} \in E))$ **do begin** // Now consider its extensions //

   g. **if** $(\nexists \lambda \in S_m)(\lambda > \lambda_{1n}e_{nm})$ **then begin** // if any extension is not yet dominated //

   h. **insert** $\lambda_{1n}e_{nm}$ **into** H // at the vertex it accesses, then enter it // // for later consideration. //

   end

   end

   end

   end

   end

4. Determine the path in S_V with the largest utility. // λ_{1V}^{*} is now $\underset{\lambda \in S_V}{\text{argmax}}\,[u(w(\lambda))]$. //.∎

3.3. Correctness and Analysis of the Algorithm.

We establish the correctness of the algorithm informally, as a consequence of some observations. The goal is to show that S_V contains all the maxima in L_{1V}, and contains only those maxima.

1. When a path is removed from the heap, no path still in the heap dominates it. Paths subsequently added to the heap are either extensions of this removed path, or extensions of paths still in the heap. Therefore, if the removed path is not dominated by existing maxima at the vertex it accesses, then it will never be dominated, so it is also a maximum. This path is added to the proper vertex's set of discovered maxima. Vertex sets are augmented only in this way, so sets S_i contain no paths that are not maxima.

2. All extensions of all maxima are added to the heap unless (i) the extension is from v_V, or (ii) the extension is into v_1 (in which case it is always dominated by λ_{11}), or (iii) the extension is dominated by one of the maxima already discovered at the vertex into which it is incident at the time it is considered. Since all maxima in the graph are extensions of other maxima in the graph, and no maximum at v_V is an extension of a path through v_V (non-negativity precludes possible benefits from cycling), all maxima in L_{1V} enter the heap. Step (3) runs until H is exhausted, so all maxima in L_{1V} must be in S_V at step (4).

Let U be an upper bound on the number of maxima in a vertex set,

$$U \geq \max_{i=1,\ldots V} [\,|S_i|\,] \; .$$

Let H be an upper bound on the largest size of the heap. $H < V^2U$, because even if all V extensions of all VU maxima were in the heap at once, clearly $|S_1| < U$, and v_V contributes no extensions. But H is $O(V^2U)$. Steps (3d) and (3g) in the algorithm are dominant; total heaping time will be only O(HlogH) comparisons of O(d). If the S sets are maintained as range trees, following Bentley and Friedman[4] and Lueker[20], (3d) requires $O(U \log^{d-1} U)$ simple

comparisons for each heap element, and (3g) takes $O(V \log^d U)$ simple comparisons per heap element. So the worst-case behavior is $O(V^2U^2\log^{d-1}U + V^3U\log^d U)$. This is a pessimistic bound; reductions from omitted edges are combinatorial.

It is *possible* to construct complete digraphs weighted in suficient dimension in which no paths are dominated, $U = O(V^V)$. However, without more *a priori* knowledge of u, the search problem (P2) is itself trivially bounded by $\Omega(U)$ from below.

An estimate of the expected behavior would be most useful. Bentley et. al.[5] determined the expected number of maxima in a set, $O((\log|S|)^{d-1})$, when rankings are independent in each dimension and set elements are independent. However, when set elements arise from combinations of a limited number of edges, the problem is unsolved. There is nevertheless computational evidence that U is usually small; the algorithm's average behavior is superb [21].

The algorithm does not at first resemble Dijkstra's algorithm. If d = 1 and one establishes a tie-breaking rule for paths of the same weight, then the S_i are restricted in cardinality to zero and one. These correspond to "permanent labels" in the original algorithm. The temporary labels are kept, essentially sorted, in the heap. Though it is true that outdated "temporary labels" rise to the top of our heap, while Dijkstra simply discards them, line (3c) is conceptually a test for discarding. Unfortunately, retaining these outdated labels in the heap hinders the remove and insert steps (3b and 3h). But any implementation of Dijkstra's algorithm must either do the same thing, or replace the old temporary labels with new ones repeatedly. In $\mathbb{R}^1$, the latter alternative's replacement operation is O(V). Dijkstra actually exploits this fact (and the fact that loop 3f is also O(V)) to obtain his coveted $O(V^2)$ complexity. However in $\mathbb{R}^d$, such a replacement is $O(|H|)$. Such are the advantages of $\mathbb{R}^1$.

It is not clear that this Dijkstra-based algorithm actually runs more quickly than a multidimensional Ford-Moore-Bellman algorithm. Some papers cast doubts on the superiority of Dijkstra's algorithm even in $\mathbb{R}^1$ (e.g., Golden[13]). In the worst case, a Ford-Moore-Bellman

algorithm in $\mathbb{R}^d$ as outilned below would have time complexity $O(V^3U\log^{d-1}VU)$.

Algorithm (following Ford-Moore-Bellman).

1. $S_1^t, \ldots, S_V^t$ are sets associated with vertices $v_1, \ldots, v_V$ respectively. These sets are, at any time, approximations to the sets that contain the maximal paths to each vertex.

2. $G_1^t, \ldots, G_V^t$ are intermediate, augmented sets.

3. $t \leftarrow 1$.

   $S_i^0 \leftarrow \{\}$ for all $i \neq 1$. $S_1^0 \dashv \{\lambda_{11}\}$.

   $S_i^{-1} \leftarrow$ **undefined** for all i.

4.a. **Until** $(t = V-1)$ or $(S_i^t = S_i^{t-1}, \forall i)$ **do begin** // until nothing changes or // // theoretical limit reached //

   b. **for each** i **do begin**

   c. $G_i^t \leftarrow S_i^t \cup \{\lambda_{1j} e_{ji} : \lambda_{1j} \in S_j^t\}$; // augment the set with // // extensions from each of the other sets //

   d. $S_i^{t+1} \leftarrow \{\lambda_{1i} \in G_i^t : \nexists \lambda_{1}'_i \in G_i^t \text{ s.t. } \lambda_{1}'_i > \lambda_{1i}\}$; // keep only the maxima in this new set //

   end

   e. $t \leftarrow t + 1$;

end

5. Determine the path in S_V^t with the largest utility.

// λ_{1V}^* is now $\underset{\lambda \in S_V^t}{\text{argmax}}\,[u(w(\lambda))]$. //∎

Step (4d) is $O((VU+U)\log^{d-1}(VU+U))$ because of the insertion time for each set's range tree.

By changing the way sets are updated, checking for and removing dominance with every addition to every set, one can produce an $O(dV^3U^2)$ algorithm, which would match the Ford-Moore-Bellman performance when $d = 1$, $U = 1$.

4.1. Stochastic Problems Reducible to Multidimensional Problems.

When u(x) is polynomial of degree d, the expected utility of a random variable is an affine linear function of the random variable's first d moments. Also, given any n independent random variables, $\tilde{w}_i$, the first moment, m_1, and the second central moment, μ_2, of $\Sigma\tilde{w}_i$ are, respectively, the sums of the first moments and second central moments of each $\tilde{w}_i$. Consequently, two types of stochastic problems can be exactly represented as multidimensional problems:

type 1. u is quadratic, $u(x) = a_0 + a_1x + a_2x^2$.

Since u is monotonically decreasing, so must $a_1x + a_2x^2$ and a_2x^2 also be monotonically decreasing. Expected utility integrals with u depend only on the random variable's first two moments:

$$\int u(x)f(x)dx = a_0 + a_1m_1 + a_2m_2^2$$
$$= a_0 + a_1m_1 + a_2m_1^2 + a_2\mu_2.$$

Construct the function $u':\mathbb{R}^2\rightarrow\mathbb{R}^1$,

$$u'(x_1,x_2) = a_0 + a_1x_1 + a_2x_1^2 + a_2x_2.$$

u' is monotonically decreasing in x_2 for fixed x_1, and monotonically decreasing in x_1 for fixed x_2. Define a multidimensional problem on u',G', where $G' = (V, E, \underline{W}')$, and weights are now

$$w'(e_{ij}) = (E[\tilde{w}(e_{ij})],E[(\tilde{w}(e_{ij}))^2]),$$

the means and variances of the previous $\tilde{w}(e_{ij})$. The solution, $\lambda_{1V}'^{*}$ identifies the same path as the solution to the original problem.

type 2. All weights on edges have densities of a class that is closed under convolutions, and uniquely determined by the first two moments.

Examples include the Poisson, Binomial, Gaussian, and Gamma classes. Since all higher moments are calculable from m_1 and m_2 (or m_1 and μ_2), and the distributions of weights on paths all belong to the same class, a polynomial utility function in the stochastic problem can be transformed into a function of two variables, m_1 and m_2. If this new function $u':\mathbb{R}^2 \rightarrow \mathbb{R}^1$ is monotonically decreasing, one can define an equivalent multidimensional problem on u', G', where weights in G' are as above,

$$w'(e_{ij}) = (E[\tilde{w}(e_{ij})], E[(\tilde{w}(e_{ij}))^2]).$$

The ensuing multidimensional problem can be solved using the techniques of the previous section.

This reducibility of some stochastic problems significantly increases the usefulness of the Bernoulli-von Neumann-Morgenstern formulation of the problem. The two most important distributions in applications are *potentially* exactly soluble: Gamma class which models output from a queue (communications networks, shortest time through queueing networks), and the Gaussian class, which models the introduction of noise (measurement error or unexplained variation).

4.2. Heuristics.

An additional virtue of the decision-analytic formulations (P1) and (P2) is that both allow heuristics that are well-understood in operations research. Approximation strategies and heuristics have been studied more closely in [21]; an attempt is made only to suggest the possibilities here.

In the stochastic problem, one can often appeal to central limit effects, even when

distributions are from various classes . When routing through networks with uniform delays (e.g., a pedestrian choosing between paths containing traffic lights), the distributions of weights on paths in L_{1V} can reasonably be modeled as Gaussian. Also, when there is a strong correlation between path cardinality and path weight, or between expected path weight and utility of path weight, the next-best path algorithm of Dreyfus-Hoffman-Pavley[10] can be employed to obtain results of high confidence (probability of having located $\lambda_{1V}^{\bullet}$) quickly. This is useful when edge-weight distributions are similar in support and mean.

In the multidimensional problem, several different gradient-dependent approximations produce polynomial-time approximation algorithms. For instance, given an estimate of $w(\lambda_{1V}^{\bullet})$, u(x) can be locally linearized. An iterative algorithm based on this linearization can thus perform dynamic programming at each step. Rounding the weights or discarding maxima that are numerically very similar are other effective ways to reduce the solution time of the dominance algorithm.

4.3. Perspectives.

In 1964, Klee described a delightful soluton to the shortest path problem. Construct a physical model of the graph with pieces of string having lengths proportional to edge-weights. Then pull tightly on the origin and destination vertices. The shortest path will appear tense: the chain that limits further pulling. Alas, there appear to be no such physical models with higher or stochastic dimensions.

Acknowledgements.

Comments by John Reif and John Pratt provided direction. Other persons contributing comments were Hiroyuki Watanabe, Arthur Berger, Peter Gacs, Gary Peterson, Rajan Suri, and Myron Fiering. This work was based largely on an undergraduate thesis supervised by A. W. Shum at Harvard.

The entire research was supported by Michael Loui. Thanks also to Marcy Singer.

Notes.

[1] Perhaps introduced by Christofides. It refers to the assignment step in all Ford/Dijkstra-like algorithms where $\text{label}(v_j) \leftarrow \min_{\forall i}[w_{ij} + \text{label}(v_i)]$.

Discussed in [1], [7], [10], [19], and [22].

[2] The e here is the base of the natural logarithm. Brackets indicate the greatest integer function. From the identity:

$$\sum_{i=0}^{V-2} \binom{V-2}{i} i! = \lfloor (V-2)!e \rfloor .$$

[3] Shogan's bounding technique[27] has time complexity greater than $O(c^V)$ for some fixed c, and Nadas[24] requires more than $O(|L_{1V}|)$.

[4] Note that if this condition is required to hold for all k, the graph requires the unlikely existence of a least element under stochastic inequality (see Nadas[24]) to give a well-defined solution for any given problem.

[5] Readers familiar with labeling algorithms for the shortest path problem will recognize the

u_i's as labels.

[6] Here, one could define a complete tie-breaking rule lexicographic with the ordering on weights to select a unique maximum from each subset of paths with exactly equivalent weight. All such selections produce combinatorial improvement, and this improvement can be significant in heuristics that force such equivalence through approximation.

[7] When several paths of identical weight access the same vertex, dynamic programming solutions choose the "best" using alternate criteria.

[1] Aho, A.V., Hopcorft, J.E., and Ullman, J.D. *Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, 1974.

[2] Arrow, K. *Essays in the Theory of Risk Bearing*, Markham, Chicago, 1970.

[3] Bentley, J.L. "Multidimensional Divide-and-Conquer," *Communications of the ACM* 23, 1980.

[4] ----------, and Friedman, J.H. "A Survey of Algorithms and Data Structures for Range Searching," Stanford Linear Acceleration Center SLAC PUB-2189, 1978.

[5] ----------, Kung, H.T., Shkolnick, M., and Thompson, C.D. "On the Average Number of Maxima in a Set of Vectors and Applications," *Journal of the ACM* 25, 1978.

[6] Bloniarz, P.A., Meyer, A.R., and Fischer, M.J. "Some Observations on Spira's Shortest Path Algorithm," SUNY Albany Computer Science TR.79-6, 1979.

[7] Christofides, N. *Graph Theory: An Algorithmic Approach*, Academic Press, New York, 1975.

[8] Dantzig, G.B. *Linear Programming and Extensions*, Princeton University Press, Princeton, 1963.

[9] Dijkstra, E.W. "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik* 1, 1959.

[10] Dreyfus, S.E. "An Appraisal of Some Shortest-Path Algorithms," *Operations Research* 17, 1969.

[11] Fishburn, P.C. "Utility Theory," *Management Science* 14, 1968.

[12] Frank, H. "Shortest Paths in Probabilistic Graphs," *Operations Research* 17, 1969.

[13] Golden, B. "Shortest Path Algorithms: A Comparison," *Operations Research* 24, 1976.

[14] Horowitz, E., and Sahni, S. *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, 1978.

[15] Howard, R.A. *Dynamic Probabilistic Systems, v.II*, Wiley and Sons, New York, 1971.

[16] ----------, and Matheson, J. "Risk-Sensitive Markov Decision Processes," *Management Science* 18, 1972.

[17] Klee, V. "String Algorithm for Shortest Path in Directed Networks," *Operations Research* 12, 1964.

[18] Kleinrock, L. *Queueing Systems, v.II*, Wiley and Sons, New York, 1976.

[19] Lawler, E. *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, and Winston, New York, 1976.

[20] Lueker, G.S. "A Data Structure for Orthogonal Range Queries," *Proceedings of the 19th Annual Symposium of Foundations of Computer Science, IEEE*, 1978.

[21] Loui, R.P. "A Graph's Best Path: The Shortest Path Problem in Mulliply and Stochastically Weighted Graphs," unpublished A.B. thesis, Division of Applied Sciences, Harvard University, Cambridge, 1982.

[22] Mahr, B. "A Bird's Eye View to Path Problems," in Noltmeier, *Graphtheoretic Concepts in Computer Science*, Springer-Verlag, Berlin, 1980.

[23] Mirchandani, P. "Shortest Distance and Reliability of Probabilistic Networks," *Computers and Operations Research* 3, 1976.

[24] Nadas, A. "Probabilistic PERT," *IBM Journal of Research and Development* 23, 1979.

[25] Pratt, J.W. "Risk Aversion in the Small and in the Large," *Econometrica* 32, 1964.

[26] ----------. "Outline of Aspects of Risk Aversion and Indirect Utility," unpublished working paper, Harvard Business School,1981.

[27] Shogan, A.W. "Bounding Distributions for a Stochastic PERT Network," *Networks* 7, 1967.

[28] Sigal, C.E., Pritsker, A.A.B., and Solberg, J.J. "The Stochastic Shortest Route Problem," *Operations Research* 28, 1980.

[29] Spira, P.M. "A New Algorithm for Finding All Shortest Paths in a Graph of Positive Arcs in Average Time $O(n^2 \log^2 n)$," *SIAM Journal on Computing* 2, 1973.

[30] Van Slyke, R. "Stochastic Aspects of Networks," *Networks* 5, 1975.