MULTI-OBJECTIVE STOCHASTIC PATH PLANNING

A Thesis

by

SUMANTRA DASGUPTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2008

Major Subject: Industrial Engineering

MULTI-OBJECTIVE STOCHASTIC PATH PLANNING

A Thesis

by

SUMANTRA DASGUPTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Amarnath Banerjee
Committee Members,	Guy L. Curry
	Lewis Ntaimo
	Faming Liang
Head of Department,	Brett A. Peters

May 2008

Major Subject: Industrial Engineering

ABSTRACT

Multi-Objective Stochastic Path Planning. (May 2008)

Sumantra Dasgupta, B.E., Birla Institute of Technology, Mesra, India;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Amarnath Banerjee

The present research formulates the path planning as an optimization problem with multiple objectives and stochastic edge parameters. The first section introduces different variants of the PP problem and discusses existing solutions to the problem. The next section introduces and solves various versions of the PP model within the scope of this research. The first three versions describe a single entity traveling from a single source to a single destination node. In the first version, the entity has a single objective and abides by multiple constraints. The second version deals with an entity traveling with multiple objectives and multiple constraints. The third version is a modification of the second version where the actual probability distributions of travel times along edges are known. The fourth and final version deals with multiple heterogeneous entities routed from multiple sources (supply nodes) to multiple destinations (demand nodes) along capacitated edges. Each of these formulations is solved by using either exact algorithms or heuristics developed in this research. The performance of each algorithm/heuristic is discussed in the final section. The main contributions of this research are:

1. Provide a framework for analyzing PP in presence of multiple objectives and stochastic edge parameters.
2. Identify candidate constraints where clustering based multi-level programming can be applied to eliminate infeasible edges.
3. Provide an exact O (V.E) algorithm for building redundant shortest paths.
4. Provide an O (V.E+C^2) heuristic for generating Pareto optimal shortest paths in presence of multiple objectives where C is the upper bound for path length. The complexity can be further reduced to O (V.E) by using graphical read-out of the Pareto frontier.
5. Provide a cost structure which can capture multiple key probability distribution parameters of edge variables. This is in contrast with usual techniques which just capture single parameters like the mean or the variance of distributions.
6. Provide a MIP formulation to a multi-commodity transportation problem with multiple decision variables, stochastic demands and uncertain edge/route capacities.
7. Provide an alternate formulation to the classic binary facility selection problem.

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Amarnath Banerjee, and my committee members, Dr. Curry, Dr. Ntaimo and Dr. Liang for their guidance and support throughout the course of this research.

I would also like to thank Dr Amarnath Banerjee and Dr. James A. Wall for the constant monetary support they provided me throughout my MS career.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Thanks to Adwitiya for standing by me and supporting me through thick and thin.

Finally, thanks to my mother and father for their constant love and encouragement.

NOMENCLATURE

E	Set of edges in a graph
G (V, E)	Graph with vertices in set V and edges in set E
MIP	Mixed Integer Programming
NP	Non Polynomial time
O (Q)	Order of Q run-time complexity
OL	Observability Limit
P(event)	Probability of event happening (cdf)
PP	Path Planning
SDP	Stochastic Dynamic Programming
SL	Safety Limit
V	Set of vertices in a graph

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION: THE PROBLEM AND PREVIOUS WORK

Path-planning problems appear in various applications. Robotics, VR walk-through, video-game, terrain-navigation, traffic modeling, routing of data packets in a telecom network, protein transport are just a few areas where path planning is the central problem. This wide application of path planning makes it an intensely researched subject. Various researchers and practitioners have come up with various formulations of the path-planning problem and various ways of solving them.

The basic problem can be stated as follows:

P1. Unconstrained Deterministic Path Planning: Given a graph G(V,E) where V is the set of nodes and E is the set of edges, a set of edge weights c_{ij} , with *i, j* being the end points of edges in E, a single source node *s*, a single destination node *d* and a single entity *e*, what is the minimum-weight path from *s* to *d* for entity *e*?

One can define a family of variants to the basic path-planning problem.

P2. Constrained Deterministic Path Planning: Given a graph G (V, E) where V is the set of nodes and E is the set of edges, a set of edge weights c_{ij}, with *i, j* being the end points of edges in E, a single source node *s*, a single destination node *d*, a single entity *e* and an upper-bound on the total allowed path weight, C, what is the minimum-weight path from *s* to *d* for entity *e*?

P3. Multi-Constrained Deterministic Path Planning: Given a graph G (V, E) where V is the set of nodes and E is the set of edges, a set of edge weights c_{ij} and edge delays d_{ij},

This thesis follows the style of *IEEE Transactions on Systems, Man and Cybernetics.*

with *i, j* being the end points of edges in E, a single source node *s*, a single destination node *d*, a single entity *e*, upper-bounds on the total allowed path weight (C) and on the total allowed path delay (D), what is the minimum-weight path from *s* to *d* for entity *e*? Each of the problems P0, P1 or P2 can have multiple-objective formulations where one needs to enumerate all the non-dominated solutions (in the Edgeworth-Pareto sense) to the path planning problem with more than one objective (e.g. minimize path delay and path cost).

Also, each of the above mentioned problems can have their stochastic equivalents where the edge parameters (e.g. weights and delays) are known only in distribution.

In practical situations, one usually works on networks with multiple sources (or supply nodes), multiple destinations (or demand nodes), multiple commodity flow and capacitated edges with multiple edge parameters. Such problems often fall into the multiple-objective, multiple-constraint framework. Stochasticity can appear in the form of probabilistic edge parameters and probabilistic demands.

Till now, it has been assumed that the graph G (V, E) is known. If G is not known in advance then one has to generate it before any path planning can be done. The graph generating techniques can be broadly classified into two categories:

1. Sampling Based: The search space is randomly sampled and the sampled points are joined to form a graph. e.g. Probabilistic Road Maps (PRM).

2. Tree Based: Edges are grown from the center of the search space to various directions until the whole search space is covered. e.g. Rapidly exploring Random Trees (RRT).

Usually, when a graph is not given, the search space (for path planning) has obstacles embedded in it. One can have a new formulation to the path planning problem.

P4. Obstacle Avoidance based Path Planning: Given a search space *S* with a set of static obstacles *O*, a source *s* and a destination *d*, what is the shortest path that an entity can follow from source to destination without running into obstacles?

This problem can again have variants with mobile obstacles and multiple entities.

The present work aims to create a framework where one can study the path planning problem on an underlying mesh (the graph G is not explicitly given but has to be generated from the mesh). It is assumed that the mesh has a set of static obstacles and that the parameters of each grid on the mesh are known either deterministically, in distribution or as a discrete histogram of scenarios. The majority of the research assumes that the path planning is for a single entity traveling from a given source to a given destination along un-capacitated edges. The final part of the research deals with a generalization of the path planning problem to multiple homogeneous entities routed from multiple sources (supply nodes) to multiple destinations (demand nodes) along capacitated edges.

Previous work in this field can be categorized as follows:

A. Extensions of Dijkstra's Algorithm: A* algorithm [1] was developed by Nilsson in 1980. By adding a simple heuristic to the distance function (an underestimate of the distance of the current point from the destination node), it is better adapted to single source, single destination cases. People have worked on various modifications of A*[2, 3, 4]. D* [2] is a popular modification of A* for a dynamically changing environment.

B. Multi-criteria path planning: Routing under multiple constraints has been shown to be NP-Complete [5]. Various pseudo-polynomial time algorithms and approximate algorithms have been reported for routing under multiple constraints [6, 7]. Genetic Algorithms have been applied to generate the non-dominated Pareto optimal set in multi-criteria path planning [8]. Potential and value-function approaches have been reported for multi-criteria path planning in [9, 10].

C. Stochastic path planning: Interval based path planning has been reported by [11, 12]. It has been proved to be NP-hard in [12]. Distribution based path planning has been reported by [13, 14]. Apart from a few special cases, most distribution based path planning formulations are NP-hard [14]. Mean-Variance formulations have been reported by [8]. Two-stage stochastic programming formulations have been reported by [15]. A Dynamic Programming approach suited for stochastic scenarios (better known as SDP) has been used in [16] to develop optimal policies.

D. Graph Generation: The graph generation techniques can broadly be categorized as point based or tree based. The most well known point-based algorithm is the Probabilistic Road Maps (PRM) [17]. Many further extensions and novel point-based algorithms have been reported in [17]. Rapidly exploring Random Trees (RRT) [17] is a very well known tree based graph building algorithm. Further tree-based algorithms can be found in [17].

The main contributions of this research are:

1. Provide a framework for analyzing PP in presence of multiple objectives and stochastic edge parameters.
2. Identify candidate constraints where clustering based multi-level programming can be applied to eliminate infeasible edges.
3. Provide an exact O (V.E) algorithm for building redundant shortest paths.
4. Provide an O (V.E+C^2) heuristic for generating Pareto optimal shortest paths in presence of multiple objectives where C is the upper bound for path length. The complexity can be further reduced to O (V.E) by using graphical read-out of the Pareto frontier.
5. Provide a cost structure which can capture multiple key probability distribution parameters of edge variables. This is in contrast with usual techniques which just capture single parameters like the mean or the variance of edge distributions.
6. Provide a MIP formulation to a multi-commodity transportation problem with multiple decision variables, stochastic demands and uncertain edge/route capacities.
7. Provide an alternate formulation to the classic binary facility selection problem.

The research is organized as follows. Section 2 describes path planning models and their solutions. Section 2.1 defines the general model for PP to be used in sections 2.1-2.3. It describes the multilevel programming approach for feasibility analysis, an approach for solving the single objective shortest path and an approach for building redundancy into the shortest path solution in case of edge failures. Section 2.2 extends section 2.1 for solving multi-objective shortest path problems. Section 2.3 indicates ways of capturing multiple key probability distribution parameters of edge variables. Section 2.4 extends the concept of a single entity PP between a given source and destination to multiple heterogeneous entities routed from multiple sources to multiple demand nodes. It formulates the problem as a MIP which can be solved using a MIP solver like CPLEX. Section 3 discusses results, conclusions and future work. These sections are followed by reference and appendices. Appendix A provides computer codes for all the algorithms described in the paper. Appendix B provides extensive results from the solution of an instance of the problem described in section 2.4.

2. PATH PLANNING MODELS AND SOLUTIONS

2.1. SINGLE OBJECTIVE, MULTIPLE PROBABILISTIC CONSTRAINTS

2.1.1. PROBLEM FORMULATION

To start with, the path-planning problem is formulated as a single objective multi-constrained problem for a single entity (given source *s* and destination *d*)

$Min_{\text{edge} \in \text{shortest-path}} \sum \text{length (edge)}$

s.t.

P (obstacle avoidance) = 1

$Min_{\text{edge} \in \text{shortest-path}}$ P (safety (edge)) >= SL

$Min_{\text{edge} \in \text{shortest-path}}$ P (observability (edge)) >= OL

As mentioned previously, it is assumed that a mesh for the search space is given. The mesh consists of uniformly sampled grids (with square projections on the horizontal plane) as shown in Fig. 1. Each grid is characterized by:

1. Average height field.
2. Horizontal projection dimensions (same for uniform sampling).
3. Vegetation type. (best: forest, average: tall-grass, worst: barren).

The PP problem is bound to have some constraints which need to fulfilled as the entity moves along the path. The current research concentrates on the constraints mentioned below:

1. Obstacle: If the average height field of a particular grid is above a particular threshold, it is marked as an obstacle grid.

2. Safety: In the scope of this research, safety is assumed to be solely dependent on vegetation type. Forest cover provides maximum safety (SL=1) while barren ground provides no safety (SL=0).
3. Observability: If the average height of the grid is more than the average height of it's neighbors, it's observability factor is improved. Compared to it's effect on safety, vegetation has an inverse effect on observability. The value of the limit, OL, is scaled between 0 and 1.

Caveat: In most cases, either safety or observability is included as a constraint because if they are both active simultaneously, the total search space becomes infeasible.

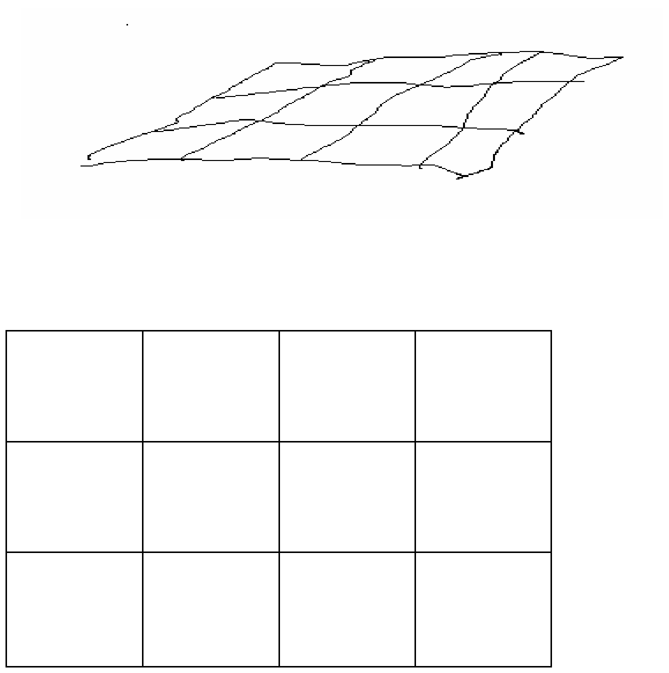


Fig. 1. Mesh model of search space and it's projection on the horizontal plane

2.1.2. SOLUTION

A. Feasibility Analysis

The approach employed in finding the feasible region is similar to multi-level programming.

Level 1: Obstacles are avoided at any cost.

Level 2: The user is given a chance to choose between safety and observability.

At each level, the clustering algorithm is run to cluster out infeasible grids and to form a progressively diminishing feasible region as shown in fig. 2.

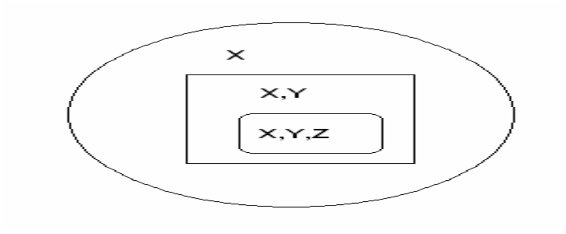


Fig. 2. Feasible region shrinking as the number of constraints increases

B. Optimality Analysis

The procedure begins with a connectivity check between source and destination. If the feasible region doesn't contain both the source and the destination nodes in a continuous stretch, then the problem is infeasible. Assuming that the feasible region has source and the destination nodes in it's interior, a graph is built connecting the grids in the feasible region. For the scope of the current research, graph building is simplified by assuming a simple rule of motion mentioned below.

Rule of Motion: The entity moves from the center of each grid to the center of it's neighboring grids.

Going by this simple rule, one can join the center of each feasible grid to the center of it's immediate feasible neighbors (a maximum of 8 neighbors) and form a graph (not a di-graph). Once the graph is formed, Dijkstra's algorithm is employed to find the shortest path.

Fig. 3 shows a small example of the grid model used for this section. It shows an 8x8 grid with 2 parameters per grid, source node, s and destination node, d. The first parameter is the average height (normalized) of the grid and the second parameter is the type of vegetation in the terrain. To be navigable, the average normalized height is set to 3. The vegetation can be of three types, namely forest (F), grassland (G) and barren (B) with covertness/safety ranging from maximum in F, medium in G and minimum in B.

5,B	5,B	5,B	1,F	1,F	1,F	1,F	1,F *s*
5,B	5,B	1,F	5,B	5,B	1,G	5,B	1,B
5,B	1,F	1,G	1,G	1,G	1,B	5,B	1,B
1,F	1,B	1,G	5,B	5,B	5,B	1,B	1,B
1,F	5,B	1,B	5,B	5,B	1,B	1,B	1,B
1,F	5,B	1,B	1,B	1,B	1,B	1,B	1,B
1,F	1,G	1,B	1,B	1,B	1,B	1,B	1,B
1,F *d*	1,G	1,B	1,B	1,B	1,B	1,B	1,B

Fig.3. Grid model used for PP

2.1.3. BUILDING ROBUST SOLUTIONS

In the previous section, the shortest path is pre-calculated in the sense that the entity knows the shortest path before it sets out on it's journey. While actually following the path, it may so happen that one of the edges included in the shortest path becomes suddenly unavailable. To address such emergency situations, redundant paths can be pre-built into the solution. The rule followed for finding alternate paths is stated below.

Rule for Re-routing: While the entity follows the minimum distance path from source to destination, it has two options at each node:

1. It follows the minimum distance path if the next edge on that path is available.
2. It follows the pre-computed next best path.

The algorithm for building redundancy into the solution (of minimum path) is similar to Dijkstra's algorithm. It is named Redundant Dijkstra because it adds redundancy to the shortest path problem. The pseudo-code (table on page 29) is given below:

Code 1: Redundant_Dijkstra (*G:graph*, *c:edge cost matrix*, *d:destination_node*)

```
1 INITIALIZE-SINGLE-SOURCE ($G$, $d$)
2 $S \leftarrow \emptyset$, $Q \leftarrow V[G]$
3 while $Q \neq \emptyset$
4     do $u \leftarrow$ EXTRACT-MIN ($Q$)
5         $S \leftarrow$ Union ($S$, $\{u\}$)
6         for each vertex $v \in Adj[u]$ //re-compute 2 shortest distances for each neighbor
```

Code 1: Continued

```
7        do RELAX-MINIMUM ($u$, $v$, $c$) and RELAX-SECOND-MINIMUM ($u$, $v$, $c$)
```

The above algorithm has the same order of complexity as Dijkstra's algorithm. It has more storage requirements. It is to be noted that the actual algorithm has been solved with the destination node instead of the source node. This is done so that each node (other than the destination node) has two paths leading to it's neighborhood and subsequently to the destination node. The first path is the optimal path. The second path is a sub optimal path where (in the absence of the first arc of the optimal path) the entity takes the second best arc from it's current node to one of it's neighbors and follows an optimal path subsequently (if there are no further disruptions).

Example 1: This example (Fig. 4) shows the working of Redundant_Dijkstra on a 4x4 grid. Grid 3 is the destination node (d) and the problem has been solved with grid 3 as the source. Every other grid has two arrows pointing into it from it's neighbors. The blue arrow is in the minimum distance path while the black arrow is in the second best path. Following blue arrows from any grid to the destination will derive the minimum distance path and should be availed whenever it is available. Only when the blue arrow edge is disrupted does one take a black arrow edge (but return to blue arrows as soon as possible).

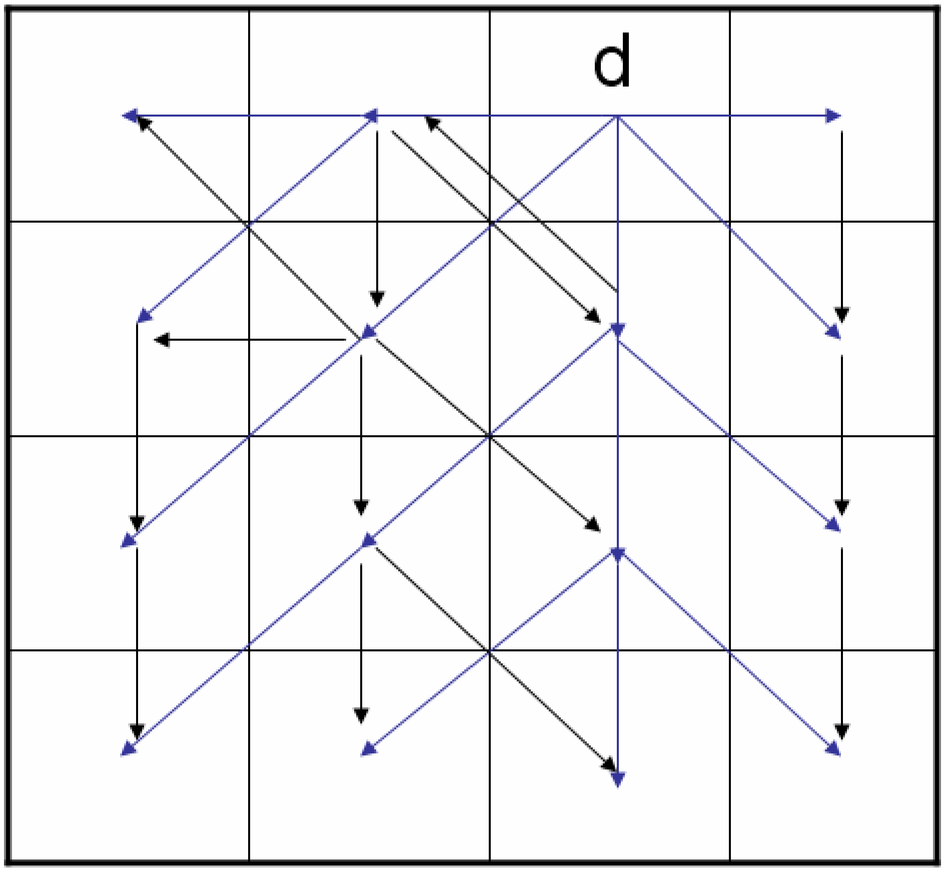


Fig. 4. Redundant paths for Example 1

2.2. MULTIPLE OBJECTIVES, MULTIPLE CONSTRAINTS

2.2.1. PROBLEM FORMULATION

The path-planning problem is now formulated as a bi-objective multi-constrained problem for a single entity (given source *s* and destination *d*)

$Min_{\text{edge} \in \text{shortest-path}} \sum \text{length (edge)}, \sum \text{delay (edge)}$

s.t.

P (obstacle avoidance) = 1

$Min_{\text{edge} \in \text{shortest-path}}$ P (safety (edge)) >= SL

$Min_{\text{edge} \in \text{shortest-path}}$ P (observability (edge)) >= OL

$\sum_{\text{edge} \in \text{shortest-path}}$ length (edge) <=C

$\sum_{\text{edge} \in \text{shortest-path}}$ delay (edge) <=D

The last two constraints are deterministic constraints. C is the upper bound on path length and D is the upper bound on path delay.

2.2.2. SOLUTION

A. Feasibility Analysis

The probabilistic constraints are dealt with as before (multi-level clustering). The deterministic constraints cannot be dealt with in the same way because they are path constraints. The whole path is either feasible or infeasible w.r.t one or both the deterministic constraints. One cannot separate them into edge constraints. Infact, it has been proven in literature that multi-criteria deterministic path planning is NP-Complete [18]. If we notice carefully, the approach in section 2.1.2 worked because the constraints in section 2.1.1 were path constraints which could be separated into individual edge constraints.

So, the multi-level programming with clustering method addresses the obstacle and safety/observability constraints. The length and delay constraints cannot be addressed in the feasibility analysis. They are addressed later while generating the actual non-dominated solutions (in the Pareto sense) to the bi-objective problem.

B. Optimality Analysis

In order to characterize the optimal solution in presence of multiple objectives, the definition of a non-dominated or Pareto set is necessary.

Definition: Non-dominated solution or Pareto Set: The set of solutions/n-tuples to a multiple-objective (n objectives, n>1) optimization problem having the properties:

1. No single solution is the best in all the objective values.
2. No single solution is worse in all the objective values than any existing solution in the set.

The optimality analysis involves generating a set of non-dominated solutions of [length, delay] tuples from source to destination. While generating the set, the infeasible combinations are discarded (the length and delay constraints are satisfied).

The algorithm used for generating the multiple non-dominating solutions is given below. The formulation can be easily extended to problems with more than two objectives.

Code 2: Non_Dom (*G:graph*, *C: ub on length of path, D: ub on delay of path, D, s:source , d:destination, c:*edge cost matrix, *del:* edge delay matrix)

```
1 INITIALIZE-SINGLE-SOURCE ($G$, $s$)
2 $S \leftarrow \emptyset$, $Q \leftarrow V[G]$
3 while $Q \neq \emptyset$
4     do $u \leftarrow$ EXTRACT-MIN ($Q$) // min is extracted based on shortest path length value
5         $S \leftarrow$ Union($S$,{$u$}) and mark $u$ as fathomed
6         for each vertex $v \in Adj[u]$ which is not fathomed yet
7             add to $v$ all paths coming through $u$; // generates alternative path set for $v$
8             update cost and delay values;
9             mark $u$ as parent for all those paths;
10    for destination node $d$
11            intialize Pareto set P = Ø
12              remove infeasible paths from alternative paths set, A of destination (based -
13             - on C, D)
14             sort A
15            Add non-dominating paths from A to P
```

The above algorithm has an order of complexity of O (V.E+C^2) where V is the number of vertices in the graph, E is the number of edges in the graph and C is the upper bound on length of feasible paths. The algorithm is very similar to Dijkstra's algorithm. Each node stores multiple paths to the source nodes. The paths stored for the destination node (set A) are chosen for investigation from line 10 onward. Line 12 restricts A population only to feasible paths (both length and delay feasible). Line 14 sorts the feasible solutions in ascending order of path length. Before insertion of a new path from set A to the Pareto set P, line 14 compares each feasible path in A with those in P as follows:

1. Both the length and delay values of a feasible path in A are larger than those of a path in Pt: Discard the feasible solution.
2. The length/delay of a feasible path in Ais smaller than that of a path which already exists in P but it’s delay/length value is larger than that of the path in P: Add the feasible solution to the Pareto set.
3. Both the length and delay values of a feasible path in A are smaller than those of a path R which already exists in P: Add the feasible solution to P, discard the path R from P.

The complexity of the algorithm can be brought down to O (VE) if one uses a graphical approach to choose the efficient frontier/Pareto set. In this method, the user chooses the Pareto set manually after the set A has been generated (lines 1-9).

2.3. PROBABILISTIC TRAVEL-TIME DISTRIBUTIONS

2.3.1. PROBLEM FORMULATION

The path-planning problem is now formulated as a bi-objective multi-constrained problem for a single entity (given source *s* and destination *d*).

$Min_{\text{edge} \in \text{shortest-path}} \sum \text{length (edge)}, \sum \text{E (cost (delay (edge)))}$

s.t.

P (obstacle avoidance) = 1

$Min_{\text{edge} \in \text{shortest-path}}$ P (safety (edge)) >= SL

$Min_{\text{edge} \in \text{shortest-path}}$ P (observability (edge)) >= OL

$\sum_{\text{edge} \in \text{shortest-path}}$ length (edge) <=C where C is the upper-bound on path length

Here, the delay parameter of an edge is known only in distribution. It is assumed that the delays of edges are i.i.d. random variables.

Let an edge e have a random travel time Y with density *f*(.), mean μ and variance σ^2. If we define a quadratic travel time cost given by $C(t) = t^2$, we have,

E(*C* (delay for path)) = $\sum_{\text{edge} \in \text{shortest-path}} (\mu^2 + \sigma^2)$

It is to be noted that the cost structure is defined in such a way that the expectation of the path cost is expressed as a sum of the expectations of the individual edge costs, that is, the path cost function is linear (in edge costs). Such a cost function imposes an optimal substructure on the problem which is required for the solution (proposed in the next sub-section) to work.

In the present setting, mean and variance parameters have been chosen to be representative of the underlying probability distribution. Although this is sufficient for most practical situations, one might encounter edge distributions which need higher order moments (besides mean and variance) to describe them with a proper level of accuracy. In that case, one can define a higher order cost structure incorporating enough moments (of the underlying distribution) to model the uncertainty. Thus the model becomes more 'informed' about the uncertainty in edge parameters.

The concept can be easily extended to other probabilistic edge variables whose distributions are i.i.d.

2.3.2. SOLUTION

The solution techniques are same as that used in section 2.2.2 if the deterministic travel times for edges (in section 2.2.1) are replaced by the mean-square + variance of random travel times for edges (in section 2.3.1). So, a deterministic equivalent of the probabilistic travel time is used in which multiple key probability distribution parameters of edge variables are captured.

2.4. MULTIPLE SOURCE/DESTINATION/ENTITY PATH PLANNING

2.4.1. PROBLEM FORMULATION

To better illustrate how the path planning problem might appeal to an industrial engineer, this section extends the single source, single destination, and single entity problems discussed previously to negotiate multiple source nodes, multiple destination nodes and multiple entities (heterogeneous).

A formulation is given for a multi-period production scheduling problem with transportation through capacitated arcs. The multi-objective flavor of the problem is given by multiple decision variables concerning production, inventory management, facility expansion/selection and transportation. Stochasticity is incorporated in the form of uncertainty in arc capacity and demand uncertainty. Uncertainty is captured in a scenario based fashion which implies the existence of a discrete number of scenarios that the demands and arc capacities conform to. This has been done to keep the problem tractable.

The problem parameters are defined below:

Parameters

ORIG - set of origin nodes

DEST- set of destination nodes

PROD – set of finished products to be transported

T – number of stages of production (maybe weeks/months)

S – number of scenarios

*prob{*1..*S}* – probability of scenario in 1..*S* *rate{PROD}* – the rate at which a product in *PROD* can be produced (tons/hr)

inv0{ORIG,PROD} – initial inventory of different products at the origins (unit tons)

exc0{DEST,PROD} – initial consignment stock of different products at different destination locations (unit tons)

*avail{*1..*T}* – usual hours of operation available in each stage t in 1..*T*

*demand{DEST,PROD,*1..*T,*1..*S}* – demand for products at different destinations in various stages (1..*T*) under various scenarios (1..*S*) (unit tons)

*limit{ORIG,*DEST,1..*S}* – the capacity of the arcs connecting the *ORIG* nodes to the *DEST* nodes under different scenarios *(*1..*S)* (unit is in total tonnage shipped)

prodcost{PROD} – cost per ton of product produced

inv{PROD} – cost per ton of product inventoried

trans_cost{ORIG,DEST,PROD} – shipping cost per ton in the arcs connecting the *ORIG* nodes to the *DEST* nodes

expan_cost{ORIG} – cost incurred in expansion of hours of operation at the facilities (cost is given per hour of expansion)

exces_cost{PROD} – cost per ton of product held as consignment stock

Decision variables

*Make{ORIG,PROD,*1..*T,*1..*S}* – tons of each product manufactured at each facility in each stage under each scenario

*Inv{ORIG,PROD,*0..*T,*1..*S}* – tons of each product inventoried at each facility in each stage under each scenario

*Trans{ORIG,DEST,PROD,*1..*T,*1..*S}* – tons of each product shipped from each origin to each demand node in each stage under each scenario

*exces{DEST,PROD,*0..*T,*1..*S}* – tons of each product inventoried at each demand site in each stage under each scenario

*expan{ORIG,*1..*T,*1..*S}* – hours of expansion needed at each facility in each stage under each scenario

The first four set of decision variables are integral.

The problem is defined below:

The objective function is given below.

minimize Total_Cost:

$$\sum_{s\in 1..S} prob(s)*(\sum_{i\in ORIG}\sum_{p\in PROD}\sum_{t\in 1..T}(prod\cos t(p)*Make(i,p,t,s)+inv\cos t(p)*Inv(i,p,t,s))$$
$$+\sum_{i\in ORIG}\sum_{j\in DEST}\sum_{p\in PROD}\sum_{t\in 1..T} trans\_\cos t(i,j,p)*Trans(i,j,p,t,s)$$
$$+\sum_{i\in ORIG}\sum_{t\in 1..T}\exp an\_\cos t(i)*\exp an(i,t,s)$$
$$+\sum_{j\in DEST}\sum_{p\in PROD}\sum_{t\in 1..T} exces\_\cos t(p)*exces(j,p,t,s))$$

The various constraints are given below:

Constraints on production time: In each stage, the total number of products made at each facility is less than the available work hours plus the expansion needed.

$$\sum_{p\in PROD} Make(i,p,t,s)/rate(p)\le avail(t)+\exp an(i,t,s) \qquad i \text{ in } ORIG,\ t \text{ in } 1..T,\ s \text{ in } 1..S$$

Initial inventory: Initialization for inventory.

$$Inv(i,p,0,s) = inv0(i,p) \qquad i \text{ in } ORIG,\ p \text{ in } PROD,\ s \text{ in } 1..S$$

Initial consignment stock: Initialization for consignment stock.

exces(j,p,0,s) = exc0(j,p) *j* in *DEST*, *p* in *PROD*, *s* in 1..*S*

Production, inventory, shipping balance: In each stage, the total of each product transported from each facility to all the destinations is equal to the total production of that product at the facility minus the change in inventory level for that particular product.

$$\sum_{j \in DEST} Trans(i,j,p,t,s) = Make(i,p,t,s) + Inv(i,p,t,s) - Inv(i,p,t-1,s)$$

i in ORIG, *p* in *PROD, t* in 1..*T*, *s* in 1..*S*

Demand fulfillment: In each stage, the total of each product transported to each demand node is greater than or equal to the demand for that product at the node plus the change in consignment stock level for that particular product.

$$\sum_{i \in ORIG} Trans(i,j,p,t,s) \geq demand(j,p,t,s) + exces(j,p,t,s) - exces(j,p,t-1,s)$$

j in *DEST*, *p* in *PROD, t* in 1..*T*, *s* in 1..*S*

Arc constraints: In each stage, the total tons of products transported are less than the arc limit for each arc.

$$\sum_{p \in PROD} Trans(i,j,p,t,s) \leq \lim it(i,j,s)$$ *i* in *ORIG*, *j* in *DEST, t* in 1..*T*, *s* in 1..*S*

Non-negativity constraints:

Make(*i,p,t,s*), *Inv*(*i,p,t,s*), *exces*(*j,p,t,s*), *Trans*(*i,j,p,t,s*) >=0

i in *ORIG*, *j* in *DEST*, *p* in *PROD, t* in 1..*T*, *s* in 1..*S*

Free variable:

expan(*i,t,s*) *i* in *ORIG*, *t* in 1..*T*, *s* in 1..*S*

Integral constraints:

Make(*i,p,t,s*), *Inv*(*i,p,t,s*), *exces*(*j,p,t,s*), *Trans*(*i,j,p,t,s*)

i in *ORIG*, *j* in *DEST*, *p* in *PROD*, *t* in 1..*T*, *s* in 1..*S*

Non-anticipativity constraints: First stage decisions are same over all scenarios.

Make(*i,p1,s*) = *Make*(*i,p,1,s+1*) *s* in 1..*S*-1

Inv(*i,p1,s*) = *Inv*(*i,p,1,s+1*) *s* in 1..*S*-1

exces(*j,p1,s*) = *exces*(*j,p,1,s+1*) *s* in 1..*S*-1

Trans(*i,j,p1,s*) = *Trans*(*i,j,p,1,s+1*) *s* in 1..*S*-1

expan(*i,1,s*) = *expan*(*i,1,s+1*) *s* in 1..*S*-1

i in *ORIG*, *j* in *DEST*, *p* in *PROD*

2.4.2. SOLUTION

The above problem is formulated as a MIP. Though such problems are NP hard, various MIP solvers exist which can give solutions to such problems. The next section provides an instance of the above problem formulated using AMPL [19] and solved using CPLEX [19].

Notes on the free variable: The expansion variable has been kept free so that it can indicate potential capacity reduction at one or more inefficient facilities at the cost of capacity expansion at one or more efficient units. This is in line with lean manufacturing and it reduces the overall production-transportation-inventory cost. It can also be viewed as a heuristic for the well-known facility selection problem which is usually done using binary decision variables.

3. RESULTS AND CONCLUSIONS

3.1. SINGLE ENTITY

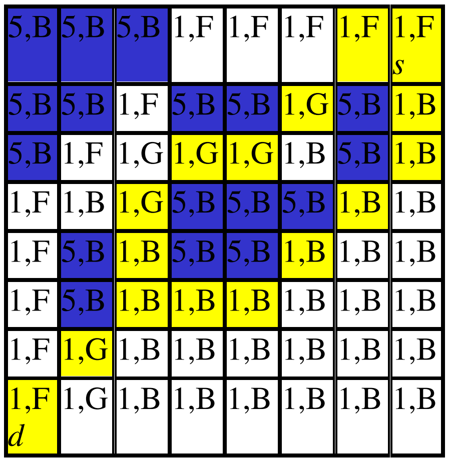
The results given below are based on the theory provided in sections 2.1 to 2.3. The path planning problem is set on a grid based model from which the graph of paths (nodes and edges) is derived as discussed in section 2.1.2.

Fig. 5 shows a small example of the grid model used for this section. It shows an 8x8 grid with 4 parameters per grid, source node, s and destination node, d. The first two parameters are explicitly shown on the grid. They are average height (normalized) of the grid and the type of vegetation in the terrain. To be navigable, the average normalized height is set to 3. The vegetation can be of three types, namely forest (F), grassland (G) and barren (B) with covertness/safety ranging from maximum in F, medium in G and minimum in B. The other two parameters are defined as follows. The entity travels from the center of a grid to the center of one of it's neighboring grids (feasible). The third parameter, the distance traveled (arc length) in between neighboring cells is either 1 unit or $\sqrt{2}$ units depending on whether it is a horizontal/vertical travel or diagonal travel. The fourth parameter, the delay is 3 seconds for diagonal arcs and 1 second for horizontal/vertical arcs.

5,B	5,B	5,B	1,F	1,F	1,F	1,F	1,F *s*
5,B	5,B	1,F	5,B	5,B	1,G	5,B	1,B
5,B	1,F	1,G	1,G	1,G	1,B	5,B	1,B
1,F	1,B	1,G	5,B	5,B	5,B	1,B	1,B
1,F	5,B	1,B	5,B	5,B	1,B	1,B	1,B
1,F	5,B	1,B	1,B	1,B	1,B	1,B	1,B
1,F	1,G	1,B	1,B	1,B	1,B	1,B	1,B
1,F *d*	1,G	1,B	1,B	1,B	1,B	1,B	1,B

Fig. 5. Grid Model

To find the shortest path on this grid, the algorithm in Section 2.1.2 is used. The MATLAB implementation of the code (dijkstra.m) is provided in appendix A. The path given by the algorithm is 8-7-14-21-20-27-35-43-50-57. It has a length of 11.071 units. It is the left yellow path shown in fig. 6.



5,B	5,B	5,B	1,F	1,F	1,F	1,F	1,F *s*
5,B	5,B	1,F	5,B	5,B	1,G	5,B	1,B
5,B	1,F	1,G	1,G	1,G	1,B	5,B	1,B
1,F	1,B	1,G	5,B	5,B	5,B	1,B	1,B
1,F	5,B	1,B	5,B	5,B	1,B	1,B	1,B
1,F	5,B	1,B	1,B	1,B	1,B	1,B	1,B
1,F	1,G	1,B	1,B	1,B	1,B	1,B	1,B
1,F *d*	1,G	1,B	1,B	1,B	1,B	1,B	1,B

Fig. 6. Shortest Path

To find the safest/maximum-covertness path on the grid, the algorithm in Section 2.1.2 is used once again. The path given by the algorithm is 8-7-6-5-4-11-18-25-33-41-49-57. It has a length of 12.243 units. So, this is the path which is the shortest path out of all the safest paths. The path is shown in green in fig. 7.

5,B	5,B	5,B	1,F	1,F	1,F	1,F	1,F *s*
5,B	5,B	1,F	5,B	5,B	1,G	5,B	1,B
5,B	1,F	1,G	1,G	1,G	1,B	5,B	1,B
1,F	1,B	1,G	5,B	5,B	5,B	1,B	1,B
1,F	5,B	1,B	5,B	5,B	1,B	1,B	1,B
1,F	5,B	1,B	1,B	1,B	1,B	1,B	1,B
1,F	1,G	1,B	1,B	1,B	1,B	1,B	1,B
1,F *d*	1,G	1,B	1,B	1,B	1,B	1,B	1,B

Fig. 7. Safest Path

To solve the shortest path problem with redundancy, the algorithm in Section 2.1.3 is used. The MATLAB implementation of the code (redundant_dijkstra.m) is provided in appendix A. The shortest and the redundant paths are shown in Fig. 8. Every feasible grid other than the destination grid has two arrows pointing into it from it's neighbors. The blue arrow belongs to the minimum distance path while the black arrow is in the second best path. Following blue arrows from any grid to the destination will derive the minimum distance path and should be availed whenever possible. Only when the blue arrow edge is disrupted does one take a black arrow edge (but return to blue arrows as soon as possible).

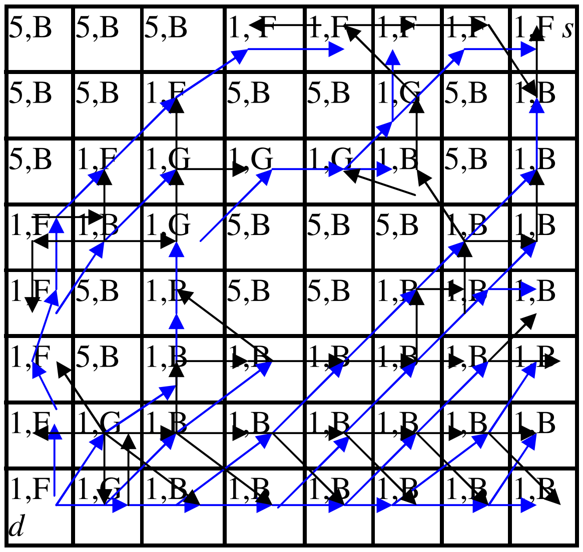

5,B 5,B 5,B 1,F 1,F 1,F 1,F 1,F s
5,B 5,B 1,F 5,B 5,B 1,G 5,B 1,B
5,B 1,F 1,G 1,G 1,G 1,B 5,B 1,B
1,F 1,B 1,G 5,B 5,B 5,B 1,B 1,B
1,F 5,B 1,B 5,B 5,B 1,B 1,B 1,B
1,F 5,B 1,B 1,B 1,B 1,B 1,B 1,B
1,F 1,G 1,B 1,B 1,B 1,B 1,B 1,B
1,F 1,G 1,B 1,B 1,B 1,B 1,B 1,B
d

Fig. 8. Redundant Paths

To solve the joint shortest path problem with length, delay tuples, the algorithm described in section 2.2.2 is used. The MATLAB implementation of the code (multi.m) is provided in appendix A. Fig. 9 shows the distance and delay values of all the feasible paths. The Pareto frontier is given by the lower left diagonal line. It consists of 6 paths with path-length from 11-14 units and delay from 14-19 seconds. The shortest path (11.071 units) found previously has the maximum delay (19 seconds). Table 1, enumerates the lengths and delay values of the paths in the Pareto set.

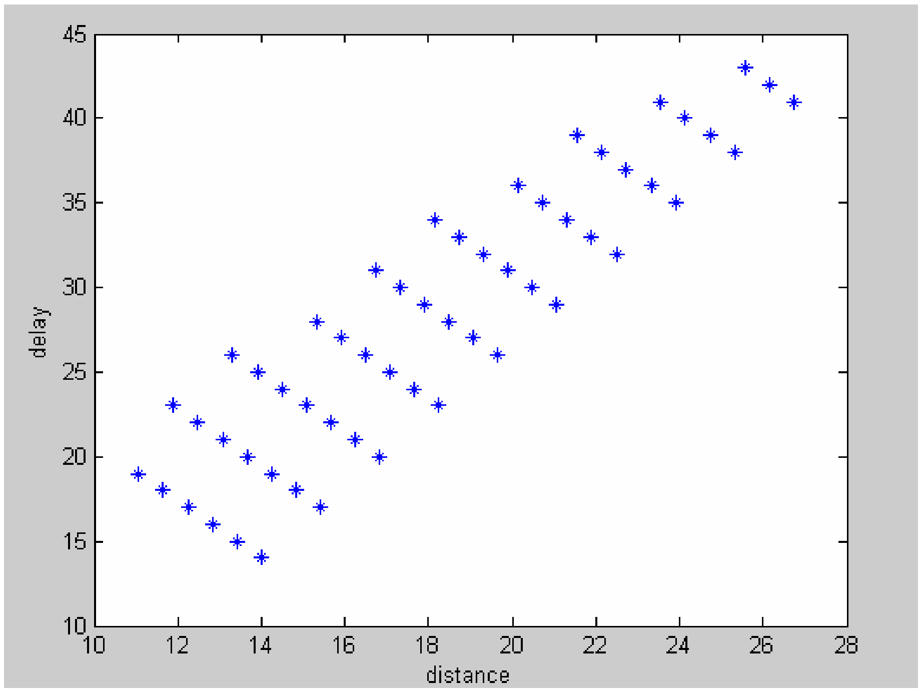


Fig. 9. Delay and distance feasible paths from source to destination

Table 1: The length and delay values of Pareto optimal paths

Length (# of sides of square block used in grid model)	Delay (in sec)
11.07	19
11.67	18.03
12.25	17.01
12.82	15.9
13.42	15.07
14	14.1

3.2. MULTIPLE ENTITY

In this section an instance of the problem formulation given in section 2.4.1 is solved using the well-known MIP solver AMPL. The model and data files used for this section (steelTptstoch1.mod, steelpTstoch1.dat) are given in appendix A.

There are 3 production sites, 7 demand sites and 2 kinds of products to be transported (bands and coils). There are 2 stages of production and there are 2 probabilistic scenarios (which affect the arc-capacities and demands). Each production site is capable of producing both bands and coils and transporting it directly to all demand sites. The production sites might have initial inventory while the demand sites might have initial consignment stock (inventory belonging to supplier stored at demand site). Each production site can operate for a maximum number of hours (different in each stage). The production limit applies uniformly to all products. Each product can be produced at a fixed rate. The cost of production of a particular product is constant. So are the costs of inventorying a particular product at the facility or at the demand sites. The transportation costs (between factories and demand sites) are different for each edge/arc. The arc limits for transportation are also different for each arc under each scenario. The arc limit applies uniformly to all products. The demands are different for each demand node for each period under each scenario. The data has been derived mainly from [18]. Stochastic scenarios have been generated by empirically perturbing data around usual values.

Expansion cost(of unit work hours) at the three facilities have been calculated. Now the decision maker is faced with the devising an optimal production, inventory management expansion and transportation strategy which will bring down the overall cost incurred by the suppliers (all the supply nodes).

The problem is modeled in AMPL and solved using CPLEX. The resulting MIP problem has 324 variables and 287 constraints. There is only one objective which encapsulates the multiple decision variables. The solution has two cases:

CASE A: The expansion variable is free: As noted earlier it can indicate potential capacity reduction at one or more inefficient units at the cost of capacity expansion at one or more efficient units. This is in line with lean manufacturing and it reduces the overall production-transportation-inventory cost. It can also be viewed as a heuristic for the well-known facility location problem which is usually done using binary decision variables.

Detailed results are shown in appendix B. In this section some key figures are discussed which confirm the validity of the model.

Table 2 shows the various amounts of the two products manufactured and inventoried at the 3 facility location. The first stage decisions are the same over all scenarios. So, the decision maker can take an unique decision inspite of multiple probabilistic scenarios.

Table 2: Make, Inv variables for Case A

Facility	Item	Stage	Scenario	Make	Inv
CLEV	bands	0	1		0
CLEV	bands	0	2		0
CLEV	bands	1	1	1250	0
CLEV	bands	1	2	1250	0
CLEV	bands	2	1	2310	0
CLEV	bands	2	2	1100	0
CLEV	coils	0	1		0
CLEV	coils	0	2		0
CLEV	coils	1	1	1150	0
CLEV	coils	1	2	1150	0
CLEV	coils	2	1	450	0
CLEV	coils	2	2	1050	0
GARY	bands	0	1		0
GARY	bands	0	2		0
GARY	bands	1	1	225	0
GARY	bands	1	2	225	0
GARY	bands	2	1	350	0
GARY	bands	2	2	175	0
GARY	coils	0	1		0

Table 2: Continued

Facility	**Item**	**Stage**	**Scenario**	**Make**	**Inv**
GARY	coils	0	2		0
GARY	coils	1	1	3250	0
GARY	coils	1	2	3250	0
GARY	coils	2	1	4300	0
GARY	coils	2	2	3050	0
PITT	bands	0	1		100
PITT	bands	0	2		100
PITT	bands	1	1	525	0
PITT	bands	1	2	525	0
PITT	bands	2	1	50	0
PITT	bands	2	2	555	0
PITT	coils	0	1		0
PITT	coils	0	2		0
PITT	coils	1	1	0	0
PITT	coils	1	2	0	0
PITT	coils	2	1	0	0
PITT	coils	2	2	0	0

Table 3 shows the values of the expansion variable (expan). As expected the expan variable indicates both facility expansion and reduction. Inventory and consignment stock values are mostly zero (except the initial inventory at PITT and some consignment stock at LAN).

Table 3: expan variable for Case A

Facility	**Stage**	**Scenario**	**expan**
CLEV	1	1	-0.5
CLEV	1	2	-0.5
CLEV	2	1	9.8
CLEV	2	2	8
GARY	1	1	9.3
GARY	1	2	9.3
GARY	2	1	27.5
GARY	2	2	17.7
PITT	1	1	-12.4
PITT	1	2	-12.4
PITT	2	1	-4.8
PITT	2	2	-2.2

Table 4 shows total transportation decisions for each arc in stage 1 (under scenarios 1 and 2). Table 5 shows the corresponding arc capacities under the two scenarios. It is to be noted that the arc capacities are fulfilled for both scenarios. *The combined decision is bounded from above by the minimum of the two arc capacities.*

Total product transported in a particular arc < = Min (arc limits for that arc under the two scenarios)

Table 4: Trans variable for Case A. Stage: 1, Scenario: 1 and 2; b-bands; c-coils

	FRA		DET		LAN		WIN		STL		FRE		LAF	
Product	b	c	b	c	b	c	b	c	b	c	b	c	b	c
GARY	0	550	0	550	150	400	75	250	0	500	0	550	0	450
CLEV	350	50	300	200	0	0	0	0	100	450	150	400	350	50
PITT	0	0	0	0	0	0	0	0	550	0	75	0	0	0

Table 5: Arc limits in scenarios 1 and 2

	FRA		DET		LAN		WIN		STL		FRE		LAF	
	1	2	1	2	1	2	1	2	1	2	1	2	1	2
GARY	650	550	650	550	650	550	650	550	750	500	650	550	650	450
CLEV	650	550	650	550	650	550	650	550	650	550	650	550	650	450
PITT	650	550	650	550	650	550	650	550	650	550	650	550	650	450

Table 6 shows actual stage 1 demand fulfillment for bands and coils. It shows the demand and the total of received items per demand site.). It is to be noted that the demands are fulfilled for both scenarios. *The unique derived decision (total of each product transported to a particular demand site) is bounded from below by the maximum of the two demand (in the two scenarios).*

Total of any product transported to a particular demand site > = Max (demands under the two scenarios for that product at that particular site)

Table 6: Demand fulfillment in Case A

Stage:1	**Demand**	**Demand**	**Received**	**Demand**	**Demand**	**Received**
Product	Bands	Bands	Bands	Coils	Coils	Coils
Scenario	1	2	1 or 2	1	2	1 or 2
FRA	300	350	350	500	600	600
DET	300	200	300	750	700	750
LAN	100	110	150	400	300	400
WIN	75	55	75	250	200	250
STL	650	600	650	950	950	950
FRE	225	205	225	850	950	950
LAF	250	350	350	500	400	500

CASE B: The expansion variable is non-negative: It can only indicate potential capacity expansion at one or more efficient units. Tables 7, 8 and 9 show some decision results. Since the expansion variable is non-negative only, the problem loses some degree of freedom (compared with case A). In such a case, the inventory and consignment inventory variables play more active roles as shown by more non-zero table entries for Inv and exces variables.

Table 7: Make, Inv variables for Case B

Facility	**Item**	**Stage**	**Scenario**	**Make**	**Inv**
CLEV	bands	0	1		0
CLEV	bands	0	2		0
CLEV	bands	1	1	2071	702
CLEV	bands	1	2	2071	702
CLEV	bands	2	1	515	0
CLEV	bands	2	2	0	0
CLEV	coils	0	1		0
CLEV	coils	0	2		0
CLEV	coils	1	1	650	0
CLEV	coils	1	2	650	0
CLEV	coils	2	1	339	0
CLEV	coils	2	2	700	0

Table 7: Continued

Facility	**Item**	**Stage**	**Scenario**	**Make**	**Inv**
GARY	bands	0	1		0
GARY	bands	0	2		0
GARY	bands	1	1	325	0
GARY	bands	1	2	325	0
GARY	bands	2	1	313	0
GARY	bands	2	2	175	0
GARY	coils	0	1		0
GARY	coils	0	2		0
GARY	coils	1	1	2600	0
GARY	coils	1	2	2600	0
GARY	coils	2	1	3800	0
GARY	coils	2	2	2631	0
PITT	bands	0	1		100
PITT	bands	0	2		100
PITT	bands	1	1	1258	528
PITT	bands	1	2	1258	528
PITT	bands	2	1	226	0
PITT	bands	2	2	0	0
PITT	coils	0	1		0

Table 7: Continued

Facility	**Item**	**Stage**	**Scenario**	**Make**	**Inv**
PITT	coils	0	2		0
PITT	coils	1	1	1219	69
PITT	coils	1	2	1219	69
PITT	coils	2	1	541	0
PITT	coils	2	2	700	0

Table 8: expan variable for Case B

Facility	**Stage**	**Scenario**	**expan**
CLEV	1	1	0
CLEV	1	2	0
CLEV	2	1	0
CLEV	2	2	0
GARY	1	1	5.2
GARY	1	2	5.2
GARY	2	1	23.7
GARY	2	2	14.7
PITT	1	1	0
PITT	1	2	0

Table 8: Continued

Facility	**Stage**	**Scenario**	**expan**
PITT	2	1	0
PITT	2	2	0

Table 9: exces variable for Case B

Item: bands **Stage**	0	1	2
DET	0	50	0
FRA	0	119	0
FRE	0	205	0
LAF	0	50	0
LAN	0	40	0
STL	0	0	0
WIN	0	0	0

So, the results in this section show:

1. The decision maker can take unique decisions and still maintain feasibility under multiple probabilistic scenarios affecting arc capacities and demands. The

decisions are automatically bounded by the worst limit of relevant stochastic variables (upper/lower) over all scenarios.

2. Expansion when used as a free variable can points towards possible production expansion/reduction decisions at the supply facilities, which will reduce the overall cost of the manufacturer. This depends on the costs of expansion.
3. Expansion can serve as substitute for inventorying. This again depends on how the expansion costs compare with the inventory management costs.

3.3. CONCLUSIONS

The present research provides a framework for multi-objective stochastic path planning and describes various algorithms for addressing such problems. The solution is initiated by reformulating the multi-objective problem as a multi-constrained problem. The feasible region of the multi-constrained problem is derived using clustering based multi-level programming. If the multi-constrained formulation has a single objective then the solution is a version of the single source shortest path problem over the feasible set. If the multi-constrained problem has more than one objective, then the non-dominated set of objectives (Pareto set) is derived. Robustness of PP is considered by generating alternate paths in case of arc failures. It is also shown how higher order cost structures can be used to incorporate key parameters of the probability distribution (of path parameters) into the multi-objective framework.

The single entity, single source, single destination path planning problem is extended to multiple entities routed from multiple supply nodes to multiple demand nodes. If the supply nodes are production nodes, production and inventory management (both at supplier and demand sites) become important decisions along with transportation decisions. The problem is further complicated by considering potential facility expansion/reduction decisions. An MIP formulation of such a problem is proposed and solved using AMPL/CPLEX. Multiple decisions are incorporated into a single objective by considering the combined cost of such decisions. Stochasticity (mainly of arc capacities and demands) is incorporated into the model using discrete scenarios.

The main contributions of this research are:

1. Provide a framework for analyzing PP in presence of multiple objectives and stochastic edge parameters.
2. Identify candidate constraints where clustering based multi-level programming can be applied to eliminate infeasible edges.
3. Provide an exact O (V.E) algorithm for building redundant shortest paths.
4. Provide an O (V.E+C^2) heuristic for generating Pareto optimal shortest paths in presence of multiple objectives where C is the upper bound for path length. The complexity can be further reduced to O (V.E) by using graphical read-out of the Pareto frontier.
5. Provide a cost structure which can capture multiple key probability distribution parameters of edge variables. This is in contrast with usual techniques which just capture single parameters like the mean or the variance of edge distributions.

6. Provide a MIP formulation to a multi-commodity transportation problem with multiple decision variables, stochastic demands and uncertain edge/route capacities.
7. Provide an alternate formulation to the classic binary facility selection problem and demonstrate that if cost structures are conducive, selective facility expansion can be a cheaper alternative to inventory management (both at supplier and demand locations).

Future work will aim at developing better algorithms (in matters of running time-complexity) for path planning along with better simulation modeling and visualization of path following.

REFERENCES

1. N. J. Nilsson, *Principles of Artificial Intelligence*, San Francisco, CA: Morgan Kaufmann Publishers Inc., 1980, pp. 72-88.
2. A. Stentz, “Optimal and efficient path planning for partially-known environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 3310-3317.
3. A. Stentz, “The focused D* algorithm for real-time replanning,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, Quebec, Canada, Aug. 1995, pp.1652-1659.
4. D. Ferguson and A. Stentz, “The delayed D* algorithm for efficient path replanning,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 2045-2050.
5. Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, Sept. 1996.
6. R. Hassin, “Approximation schemes for the restricted shortest path problem,” *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36-42, Feb. 1992.
7. J. .M. Jaffe, “Algorithms for finding paths with multiple constraints,” *Networks*, vol. 14, no. 1, pp. 95-116, 1984.
8. Z. Ji, A. Chen and K. Subprasom, “Finding multi-objective paths in stochastic networks: a simulation-based genetic algorithm approach,” in *Proceedings of the*

2004 Congress of Evolutionary Computation, Portland, Oregon, USA, June 2004, vol. 1, pp. 174-180.

9. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol.5, no. 1, pp. 90-98, 1986.
10. J. Barraquand, B. Langlois and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224-241, Mar/Apr 1992.
11. O.E. Karasan, M.C. Pinar and H. Yaman, "The robust shortest path problem with interval data," Department of Industrial Engineering, Bilkent University, Ankara, Turkey, Tech. Rep., Aug. 2002.
12. P. Zielinski, "The computational complexity of the relative robust shortest path problem with interval data," *European Journal of Operational Research*, vol. 558, pp. 570-576, 2004.
13. E. Nikolova, J. Kelner, M Brand and M. Mitzenmacher, "Stochastic shortest paths via quasi-convex maximization," in *Proceedings of 2006 European Symposium of Algorithms (ESA'06)*, Zurich, Switzerland, Sept. 2006.
14. E. Nikolova, M. Brand and D. Karger, "Optimal route planning under uncertainty," in *Proceedings of 2006 International Conference on Automated Planning & Scheduling (ICAPS 2006)*, Lake District, England, June 2006.
15. G. Barbarosoglu and Y. Arda, "A two-stage stochastic programming framework for transportation planning in disaster response," *Journal of the Operational Research Society*, vol. 55, no. 1, pp. 43-53, Jan. 2004.

16. M..P. Wellman, K. Larson, M. Ford and P. R. Wurman, "Path planning under time-dependent uncertainty," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, Quebec, Canada, Aug. 1995, pp. 532-539.

17. S. M. LaValle, *Planning Algorithms.* 1st Ed, New York, NY: Cambridge University Press, 2006.

18. A. Puri and S. Tripakis, "Algorithms for routing with multiple constraints," Tech. Rep. No. UCB/ERL M01/7, 2001.

19. R. Fourer, D.M. Gay and B.W. Kernighan, *AMPL, A Modeling language for Mathematical Programming.* 2nd Ed, Toronto, Canada: Thomson Books/Cole, 2003.

APPENDIX A

MATLAB code for dijkstra.m

```
%dijkstra
%define a set of nodes/grids
%movement---center to center of grid
%figure out some easy way to generate node lists
%assume uniformly sampled square grids
x_grids=8;
y_grids=8;
nodes=zeros(y_grids*x_grids,2);
for i=1:y_grids
    for j=1:x_grids
        index=j+x_grids*(i-1);
        nodes(index,1)=i-1;
        nodes(index,2)=j-1;
    end
end
h=[5 5 5 1 1 1 1 1;
   5 5 1 5 5 1 5 1;
   5 1 1 1 1 1 5 1;
   1 1 1 5 5 5 1 1;
   1 5 1 5 5 1 1 1;
   1 5 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1];
%nodes=[0 0; 0 1; 0 2; 1 0; 1 1; 1 2; 2 0; 2 1; 2 2];
n=size(nodes,1);
%give source
s=8;
dest=57;
%define feasible grids
feasible=ones(1,n);
infeasible=zeros(1,n);
%do feasibility analysis
% %for now use random
for i=1:n
%     r=rand(1,1);
     if h(ceil(i/8),rem(i,8))>3
         feasible(i)=0;
     end
 end
```

```
% %if source is infeasible problem is unsolvable
% %so make source grid feasible
% feasible(s)=1;
% infeasible=ones(1,n)-feasible;
% %define edges and associated cost
%define fathomed node set
fathomed=zeros(1,n); %initially none of the nodes are visited
%define parent node
parent=zeros(1,n);
%define min-priority set
min_pr_q=[1:n];
%define distance matrix
dist=zeros(1,n);
%define cost-matrix
cost_m=zeros(n,n);
%initialize min-priority queue
dist(1:n)=inf;
dist(s)=0;
%main loop begins
for i=1:n
   %pop min unfathomed and feasible node
   temp=dist+100000*fathomed+100000*infeasible;
   [a b]=min(temp);
   %mark as visited
   fathomed(b)=1;
   %relax neighbors and update parent
   for j=1:n
      %check only unfathomed and feasible nodes
      if fathomed(j)==0 & feasible(j)==1
         %check if neighbors
         d=((nodes(b,1)-nodes(j,1))^2+(nodes(b,2)-nodes(j,2))^2)^0.5;
         % 1 neighbor
         if d==1
            if dist(j)>dist(b)+d
               dist(j)=dist(b)+d;
               parent(j)=b;%the path from source to j goes through b
            end
         end
         % 2 neighbor
         if d==2^0.5
            if dist(j)>dist(b)+d
               dist(j)=dist(b)+d;
               parent(j)=b;%the path from source to j goes through b
            end
            end
        end
    end
end
%trace path; maybe modify to A*
%diagram
```

MATLAB code for redundant_dijkstra.m

```
%dijkstra
%define a set of nodes/grids
%movement---center to center of grid
%figure out some easy way to generate node lists
%assume uniformly sampled square grids
x_grids=8;
y_grids=8;
nodes=zeros(y_grids*x_grids,2);
for i=1:y_grids
    for j=1:x_grids
        index=j+x_grids*(i-1);
        nodes(index,1)=i-1;
        nodes(index,2)=j-1;
    end
end
%nodes=[0 0; 0 1; 0 2; 1 0; 1 1; 1 2; 2 0; 2 1; 2 2];
n=size(nodes,1);
h=[5 5 5 1 1 1 1 1;
   5 5 1 5 5 1 5 1;
   5 1 1 1 1 1 5 1;
   1 1 1 5 5 5 1 1;
   1 5 1 5 5 1 1 1;
   1 5 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1];
%give source
s=8;
%give destination
dest=57;
%make destination as source and run redundant Dijkstra
sou=s;
s=dest;
%define feasible grids
feasible=ones(1,n);
```

```
infeasible=zeros(1,n);
%do feasibility analysis
%for now use random
for i=1:n
%    r=rand(1,1);
     if h(ceil(i/8),rem(i,8))>3
         feasible(i)=0;
     end
 end
% for i=1:n
%    r=rand(1,1);
%    if r<0.5
%        feasible(i)=0;
%    end
% end
% %if source is infeasible problem is unsolvable
% %so make source grid feasible
% feasible(s)=1;
% infeasible=ones(1,n)-feasible;
%define edges and associated cost
%define fathomed node set
fathomed=zeros(1,n); %initially none of the nodes are visited
%define parent node
parent=zeros(2,n);
%define min-priority set
min_pr_q=[1:n];
%define distance matrix
dist=zeros(2,n);
%define cost-matrix
cost_m=zeros(n,n);
%initialize min-priority queue
dist(1:2,1:n)=inf;
dist(1:2,s)=0;
%main loop begins
for i=1:n
    %pop min unfathomed and feasible node
    temp=dist(1,1:n)+100000*fathomed+100000*infeasible;
    [a b]=min(temp);
    %mark as visited
    fathomed(b)=1;
    %relax neighbors and update parent
    for j=1:n
        %check only unfathomed and feasible nodes
        %if fathomed(j)==0 & feasible(j)==1
        if feasible(j)==1
            %check if neighbors
            d=((nodes(b,1)-nodes(j,1))^2+(nodes(b,2)-nodes(j,2))^2)^0.5;
            % 1 neighbor
            if d==1
                if dist(1,j)>dist(1,b)+d
                    dist(2,j)=dist(1,j);
                    dist(1,j)=dist(1,b)+d;
                    parent(2,j)=parent(1,j);
                    parent(1,j)=b;
%                   parent(j)=b;%the path from source to j goes through b
                elseif dist(2,j)>dist(1,b)+d
                    %dist(2,j)=dist(1,j);
                    dist(2,j)=dist(1,b)+d;
                    %parent(2,j)=parent(1,j);
                    parent(2,j)=b;
%                   parent(j)=b;%the path from source to j goes through b
                end
            end
            % 2 neighbor
            if d==2^0.5
                if dist(1,j)>dist(1,b)+d
                    dist(2,j)=dist(1,j);
                    dist(1,j)=dist(1,b)+d;
                    parent(2,j)=parent(1,j);
                    parent(1,j)=b;
%                   parent(j)=b;%the path from source to j goes through b
                elseif dist(2,j)>dist(1,b)+d
                    %dist(2,j)=dist(1,j);
                    dist(2,j)=dist(1,b)+d;
                    %parent(2,j)=parent(1,j);
                    parent(2,j)=b;
%                   parent(j)=b;%the path from source to j goes through b
                end
            end
        end
    end
end
%trace path; maybe modify to A*
%diagram
```

MATLAB code for multi.m

```
%dijkstra
%define a set of nodes/grids
%movement---center to center of grid
%figure out some easy way to generate node lists
%assume uniformly sampled square grids
x_grids=8;
y_grids=8;
nodes=zeros(y_grids*x_grids,2);
for i=1:y_grids
    for j=1:x_grids
        index=j+x_grids*(i-1);
        nodes(index,1)=i-1;
        nodes(index,2)=j-1;
    end
end
h=[5 5 5 1 1 1 1 1;
   5 5 1 5 5 1 5 1;
   5 1 1 1 1 1 5 1;
   1 1 1 5 5 5 1 1;
   1 5 1 5 5 1 1 1;
   1 5 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1;
   1 1 1 1 1 1 1 1];
%nodes=[0 0; 0 1; 0 2; 1 0; 1 1; 1 2; 2 0; 2 1; 2 2];
pareto=zeros(50,3);
n=size(nodes,1);
%give source
s=8;
dest=57;
%define feasible grids
feasible=ones(1,n);
infeasible=zeros(1,n);
for i=1:n
%     r=rand(1,1);
     if h(ceil(i/8),rem(i,8))>3
         feasible(i)=0;
     end
 end
%do feasibility analysis
% %for now use random
% for i=1:n
%     r=rand(1,1);
%     if r<0.5
%         feasible(i)=0;
%     end
% end
% %if source is infeasible problem is unsolvable
% %so make source grid feasible
% feasible(s)=1;
% infeasible=ones(1,n)-feasible;
% %define edges and associated cost
%define fathomed node set
fathomed=zeros(1,n); %initially none of the nodes are visited
%define parent node
parent=zeros(100,n);
%define min-priority set
min_pr_q=[1:n];
%define distance matrix
dist=zeros(100,n);
delay=zeros(100,n);
nneb=zeros(1,n);
%define cost-matrix
cost_m=zeros(n,n);
%initialize min-priority queue
dist(1:100,1:n)=inf;
delay(1:100,1:n)=inf;
dist(1:100,s)=0;
delay(1:100,s)=0;
cou=zeros(1,n);
cou(s)=1;
%main loop begins
for i=1:n
   %pop min unfathomed and feasible node
   %temp=dist+100000*fathomed+100000*infeasible;
   temp=delay(1,1:n)+100000*fathomed+100000*infeasible;
   [a b]=min(temp);
   %mark as visited
   fathomed(b)=1;
   %relax neighbors and update parent
   for j=1:n
      %check only unfathomed and feasible nodes
      if feasible(j)==1 & fathomed(j)==0
         %check if neighbors
         d=((nodes(b,1)-nodes(j,1))^2+(nodes(b,2)-nodes(j,2))^2)^0.5;
         % 1 neighbor
```

```
          if d==1
             %if dist(1,j)>dist(1,b)+d
                de=1;
                delay(cou(j)+1:cou(j)+cou(b),j)=delay(1:cou(b), b)+de;
                dist(cou(j)+1:cou(j)+cou(b),j)=dist(1:cou(b), b)+d;
                parent(cou(j)+1:cou(j)+cou(b),j)=b;%the path from source to j goes through
b
                cou(j)=cou(j)+cou(b);
                % end
          end
          % 2 neighbor
          if d==2^0.5
             %if dist(1,j)>dist(1,b)+d
                de=3;
                delay(cou(j)+1:cou(j)+cou(b),j)=delay(1:cou(b), b)+de;
                dist(cou(j)+1:cou(j)+cou(b),j)=dist(1:cou(b), b)+d;
                parent(cou(j)+1:cou(j)+cou(b),j)=b;%the path from source to j goes through
b
                cou(j)=cou(j)+cou(b);
                %end
          end
       end
   end
end
```

AMPL model file (steetTptstoch1.mod)

```
#given set of origin and destination nodes
#given products
#given a transportation network with arc capacity
#given demands
#given stages
#given scenarios and their probabilities
#given initial inventory
#given hours available at each facility
#given rate of production
#given production,inventory,transportation,expansion cost

#decide how much to manufacture
#decide how much to inventory
#decide how much to transport
#decide how much to expand/reduce (hours available)
```

```
set ORIG;     # origins
set DEST;     # destinations
set PROD;     # products

param T > 0;  # number of weeks/stages
param S > 0;  # number of scenarios

param prob {1..S} >=0;
param rate {PROD} > 0;          # tons per hour produced
param inv0 {ORIG,PROD} >= 0;    # initial inventory
param exc0 {DEST,PROD} >=0;     # initial consignment stock
param avail {1..T} >= 0;        # hours available in week
param demand {DEST,PROD,1..T,1..S} >= 0;  # demand in destination per week
param limit {ORIG,DEST,1..S} >=0; #arc capacity

param prodcost {PROD} >= 0;     # cost per ton produced
param invcost {PROD} >= 0;      # carrying cost/ton of inventory

#param revenue {PROD,1..T,1..S} >= 0; # revenue per ton sold
param trans_cost {ORIG,DEST,PROD} >= 0;  # shipping cost/ton
#param shortcost {PROD} >= 0;     #shortage cost/ton of inventory
param expan_cost {ORIG} >= 0;     #possible expansion at each origin
param exces_cost {PROD} >= 0;

var Make {ORIG,PROD,1..T,1..S} >= 0 integer;      # tons produced
var Inv {ORIG,PROD,0..T,1..S} >= 0 integer;       # tons inventoried
var Trans {ORIG,DEST,PROD, t in 1..T,1..S} >= 0 integer; # tons transported
var expan {ORIG,1..T,1..S}>=0; #expansion
var exces {DEST,PROD,0..T,1..S} >=0 integer; #consignment stock

minimize Total_Cost:
 sum {s in 1..S} prob[s] *
   (sum {i in ORIG,p in PROD, t in 1..T} ( prodcost[p]*Make[i,p,t,s] +
     invcost[p]*Inv[i,p,t,s])+sum {i in ORIG, j in DEST, p in PROD, t in 1..T}
                         trans_cost[i,j,p] * Trans[i,j,p,t,s]+
                 sum {i in ORIG, t in 1..T} expan[i,t,s]*expan_cost[i]+
                 sum {j in DEST, p in PROD, t in 1..T} exces[j,p,t,s]*exces_cost[p]);


subject to Time {i in ORIG, t in 1..T, s in 1..S}:
   sum {p in PROD} (1/rate[p]) * Make[i,p,t,s] <= avail[t]+expan[i,t,s];

             # Total of hours used by all products
             # may not exceed hours available, in each week
```

```
subject to Init_Inv {i in ORIG, p in PROD, s in 1..S}:  Inv[i,p,0,s] = inv0[i,p];

subject to Init_con_stock {j in DEST, p in PROD, s in 1..S}: exces[j,p,0,s] = exc0[j,p];

          # Initial inventory must equal given value
subject to Balance {i in ORIG, p in PROD, t in 1..T, s in 1..S}:
  sum {j in DEST} Trans[i,j,p,t,s] = Make[i,p,t,s]+Inv[i,p,t-1,s]-Inv[i,p,t,s];
#balance of products

subject to Demandd {j in DEST, p in PROD, t in 1..T, s in 1..S}:
  sum {i in ORIG} Trans[i,j,p,t,s] >= -exces[j,p,t-1,s]+exces[j,p,t,s]+demand [j,p,t,s];
#demand fulfilment

subject to Multi {i in ORIG, j in DEST, t in 1..T, s in 1..S}:
  sum {p in PROD} Trans[i,j,p,t,s] <= limit[i,j,s]; #arc capacity, assumed constant here


#subject to Balance {p in PROD, t in 1..T, s in 1..S}:
#   Make[p,t,s] + Inv[p,t-1,s] = Sell[p,t,s] + Inv[p,t,s];

          # Tons produced and taken from inventory
          # must equal tons sold and put into inventory
#nonanticipativity constraints: the first week's decision is same over all the scenarios

subject to Make_na {i in ORIG, p in PROD, s in 1..S-1}:
  Make[i,p,1,s] = Make[i,p,1,s+1];

subject to Inv_na {i in ORIG, p in PROD, s in 1..S-1}:
  Inv[i,p,1,s] = Inv[i,p,1,s+1];

subject to Trans_na {i in ORIG, j in DEST, p in PROD, s in 1..S-1}:
  Trans[i,j,p,1,s] = Trans[i,j,p,1,s+1];

subject to Expand_na {i in ORIG, s in 1..S-1}:
  expan[i,1,s] = expan[i,1,s+1];

subject to Exces_na {j in DEST, p in PROD, s in 1..S-1}:
  exces[j,p,1,s] = exces[j,p,1,s+1];


#display {s in 1..S}
  #sum {p in PROD, t in 1..T} (revenue[p,t,s]*Sell[p,t,s] -
    #prodcost[p]*Make[p,t,s] - invcost[p]*Inv[p,t,s]);
```

AMPL data file (steetTptstoch1.dat)

```
data;
param S := 2;
param T := 2;

set PROD := bands coils;
set ORIG := GARY CLEV PITT ;
set DEST := FRA DET LAN WIN STL FRE LAF ;

param prob := 1 0.45 2 0.55; # 3 0.2
#param prob := 1 0.0001 2 0.0001 3 0.9998;

param avail :=  1 15  2 5  ; #3 32  4 40 ;

param rate :=  bands 200   coils 140 ;
param inv0 :  bands   coils :=
      GARY   0      0
      CLEV   0      0
      PITT   100      0 ;

param exc0 :  bands   coils :=
      FRA    0      0
      DET    0      0
      LAN    0      0
      WIN    0      0
      STL    0      0
      FRE    0      0
      LAF    0      0 ;

param prodcost :=  bands 10    coils  11 ;
param invcost  :=  bands  1.5  coils   3 ;
param exces_cost := bands 1.5  coils   3 ;
#param shortcost := bands  5    coils   6 ;

#param limit default 1000 ;

param limit :=
 [*,*,1]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
    GARY   650  650  650  650  750  650  650
    CLEV   650  650  650  650  650  650  650
    PITT   650  650  650  650  650  650  650

 [*,*,2]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
   GARY   550  550  550  550  500  550  450
   CLEV   550  550  550  550  550  550  450
   PITT   550  550  550  550  550  550  450 ;

param expan_cost := GARY 10000 CLEV 20000 PITT 30000 ;

param demand :=

 [*,*,1,1] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
        bands    300   300   100    75   650   225   250
        coils    500   750   400   250   950   850   500

 [*,*,2,1] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
        bands    500   500   150   150   700   350   400
        coils    500   750   600   500   750   950   700

 #[*,*,3,1] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
 #       bands    300   300   100    75   650   225   250
 #       coils    500   750   400   250   950   850   500

 #[*,*,4,1] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
 #       bands    300   300   100    75   650   225   250
 #       coils    500   750   400   250   950   850   500

 [*,*,1,2] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
        bands    350   200   110    55   600   205   350
        coils    600   700   300   200   950   950   400

 [*,*,2,2] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
        bands    350   200   110    55   600   205   350
        coils    600   700   300   200   950   950   400  ;

 #[*,*,3,2] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
 #       bands    350   200   110    55   600   205   350
 #       coils    600   700   300   200   950   950   400

 #[*,*,4,2] (tr):   FRA   DET   LAN   WIN   STL   FRE   LAF :=
 #       bands    350   200   110    55   600   205   350
 #       coils    600   700   300   200   950   950   400


#param revenue:  1   2   3 :=
#      bands 1  25  23  21
#      bands 2  26  24  27
#       bands 3  27  25  33
#       bands 4  27  25  35
#       coils 1  30  30  30
#       coils 2  35  33  32
#       coils 3  37  35  33
#       coils 4  39  36  33 ;

param trans_cost :=

 [*,*,bands]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
      GARY    30   10    8   10   11   71    6
      CLEV    22    7   10    7   21   82   13
      PITT    19   11   12   10   25   83   15

 [*,*,coils]:  FRA  DET  LAN  WIN  STL  FRE  LAF :=
      GARY    39   14   11   14   16   82    8
      CLEV    27    9   12    9   26   95   17
      PITT    24   14   17   13   28   99   20 ;
```

APPENDIX B

Results from CASE A in section 3.2

```
184 iterations, objective 543236.6071
ampl: display Make, Inv

:                    Make          Inv        :=
CLEV bands 0 1          .           0
CLEV bands 0 2          .           0
CLEV bands 1 1   1250               0
CLEV bands 1 2   1250               0
CLEV bands 2 1   2310               0
CLEV bands 2 2   1100               0
CLEV coils 0 1          .           0
CLEV coils 0 2          .           0
CLEV coils 1 1   1150               0
CLEV coils 1 2   1150               0
CLEV coils 2 1    450               0
CLEV coils 2 2   1050               0
GARY bands 0 1          .           0
GARY bands 0 2          .           0
GARY bands 1 1    225               0
GARY bands 1 2    225               0
GARY bands 2 1    350               0
GARY bands 2 2    175               0
GARY coils 0 1          .           0
GARY coils 0 2          .           0
GARY coils 1 1   3250               0
GARY coils 1 2   3250               0
GARY coils 2 1   4300               0
GARY coils 2 2   3050               0
PITT bands 0 1          .         100
PITT bands 0 2          .         100
PITT bands 1 1    525               0
PITT bands 1 2    525               0
PITT bands 2 1     50               0
PITT bands 2 2    555               0
PITT coils 0 1          .           0
PITT coils 0 2          .           0
PITT coils 1 1    0                 0
PITT coils 1 2    0                 0
PITT coils 2 1     0                  0
PITT coils 2 2     0                  0
ampl: display exces;
exces [*,bands,*,1]
:    0      1       2    :=
DET    0    0          0
FRA    0    0          0
FRE    0    0          0
LAF    0    0          0
LAN    0   40          0
STL    0    0          0
WIN    0    0          0

 [*,bands,*,2]
:    0      1       2    :=
DET   0    0          0
FRA   0    0          0
FRE   0    0          0
LAF   0   0           0
LAN   0   40          0
STL   0    0          0
WIN   0    0          0

 [*,coils,*,1]
:    0      1      2    :=
DET   0   0         0
FRA   0   0         0
FRE   0    0        0
LAF   0    0        0
LAN   0   0         0
STL   0   0         0
WIN   0   0         0

 [*,coils,*,2]
:    0      1      2    :=
DET   0   0         0
FRA   0   0         0
FRE   0    0        0
LAF   0   0         0
LAN   0   0         0
STL   0   0         0
WIN   0   0         0
;
```

```
ampl: display expan;
expan :=
CLEV  1 1   -0.535714
CLEV  1 2   -0.535714
CLEV  2 1    9.76429
CLEV  2 2    8
GARY 1 1     9.33929
GARY 1 2     9.33929
GARY 2 1    27.4643
GARY 2 2    17.6607
PITT   1 1   -12.375
PITT   1 2   -12.375
PITT   2 1   -4.75
PITT   2 2   -2.225
;

ampl: display Trans;
Trans [*,*,bands,1,1] (tr)
:   CLEV      GARY      PITT    :=
DET   300    0            0
FRA   350    0            0
FRE   150     0            75
LAF   350     0            0
LAN     0     150          0
STL    100     0            550
WIN      0     75           0

 [*,*,bands,1,2] (tr)
:       CLEV         GARY      PITT    :=
DET   300           0            0
FRA   350           0            0
FRE    150           0           75
LAF    350           0           0
LAN      0           150         0
STL     100           0           550
WIN       0           75          0

 [*,*,bands,2,1] (tr)
:   CLEV      GARY     PITT    :=
DET   500   0             0
FRA   350   150           0
FRE   350   0              0
LAF   400   0              0
LAN    60   50             0
```

```
STL   650    0                 50
WIN     0    150              0
 [*,*,bands,2,2] (tr)
:    CLEV       GARY      PITT    :=
DET   200       0            0
FRA   350       0            0
FRE   150        0           55
LAF   300       50          0
LAN     0       70           0
STL   100        0          500
WIN     0       55           0

 [*,*,coils,1,1] (tr)
:    CLEV  GARY PITT    :=
DET   200     550   0
FRA    50     550   0
FRE   400    550   0
LAF    50     450   0
LAN     0     400   0
STL   450    500   0
WIN     0     250   0

 [*,*,coils,1,2] (tr)
:    CLEV  GARY PITT    :=
DET   200     550   0
FRA    50     550   0
FRE   400    550   0
LAF    50     450   0
LAN     0     400   0
STL   450    500   0
WIN     0     250   0

 [*,*,coils,2,1] (tr)
:    CLEV  GARY PITT    :=
DET   100   650   0
FRA     0    500   0
FRE   300    650   0
LAF    50     650   0
LAN     0     600   0
STL     0     750   0
WIN     0     500   0

 [*,*,coils,2,2] (tr)
:    CLEV  GARY PITT    :=
DET   150   550   0
FRA    50   550   0
FRE   400   550   0
LAF     0   400   0
LAN     0   300   0
STL   450   500   0
WIN     0   200   0
;
```

VITA

Sumantra Dasgupta received his Bachelor of Engineering degree in electrical & electronics engineering from Birla Institute of Technology, Mesra, India in August 2000. He entered the electrical engineering program at Texas A&M University in August 2000 and received his Master of Science degree in May 2003. His research interests include large scale optimization, simulation based optimization, optimal & robust control, design of control systems for manufacturing, signal processing, data-mining, AI and visualization.

Mr. Dasgupta may be reached at 8B, Burnpur Road, 1st Floor, PO-Chelidanga, Asansol South, Dist: Burdwan, W.B., India 713304, ph: 011913463261551(India)/ 9792291390(US). His email is sumantrad@gmail.com.