08:55 Prague OK655
09:00 Trondheim DY5329 Open
09:30 Jersey AAG022 Closed
09:40 Genoa FR972 Cancelled
09:50 Turin FR464 Cancelled
09:50 Altenburg Leipzig FR8926 Cancelled
09:50 Ancona FR124 Cancelled
09:55 Murcia FR8032 Cancelled
10:00 New York (JFK) MY101 Open
10:00 Palma Mallorca EZY3211 Cancelled
10:30 Limoges FR8776 Cancelled
10:30 Girona Barcelona FR9802 Cancelled
10:35 Alicante AB8010 Open

Robust Optimization with Recovery: Application to Shortest Paths and Airline Scheduling

Niklaus Eggenberg, Transport and Mobility Laboratory, EPFL

Matteo Salani, Transport and Mobility Laboratory, EPFL

Michel Bierlaire, Transport and Mobility Laboratory, EPFL

Conference paper STRC 2007

STRC

7th Swiss Transport Research Conference
Monte Verità / Ascona, September 12. – 14. 2007

Robust Optimization with Recovery: Application to Shortest Paths and Airline Scheduling

Niklaus Eggenberg
TRANSP-OR, EPFL
Lausanne

+41216932432
niklaus.eggenberg@epfl.ch

Matteo Salani
TRANSP-OR, EPFL
Lausanne

+41216932432
matteo.salani@epfl.ch

Michel Bierlaire
TRANSP-OR, EPFL
Lausanne

+41216932537
michel.bierlaire@epfl.ch

Abstract

In this exploratory paper we consider a robust approach to decisional problems subject to uncertain data in which we have an additional knowledge on the strategy (algorithm) used to react to an unforeseen event or recover from a disruption. This is a typical situation in scheduling problems where the decision maker has no a priori knowledge on the probabilistic distribution of such events but he only knows rough information on the event, such as its impact on the schedule. We discuss a general framework to address this situation and its links with other existing methods, we present an illustrative example on the Shortest Path Problem with Interval Data (SPPID) and we discuss a more general application to airline scheduling with recovery.

Keywords

Robust Optimization – Recovery – Recoverable Optimization

1 Introduction

Mathematical modeling is an effective way to solve a wide range of decisional problems. Applications in production, transportation, engineering and finance benefits from quantitative methods developed for mathematical optimization. As the word *model* suggests, we represent the reality through a set of equations and we solve this set of equations in order to take decisions with some quantitative support. Sometimes, some strong assumptions are taken to model decisional problems, because otherwise, they are intractable from a computational point of view. For example objective functions and constraints are assumed to be linear and data is assumed to be completely and deterministically known in advance. Indeed it is impressive to notice how many real life problems can be modeled with accuracy using linear programs. However, data uncertainty is one major issue that might completely invalidate the solution to a decisional problem.

There are many fields where operations research tools are needed and used to help the decision makers, as for example airline scheduling, container transshipment, traffic control, vehicle routing and many others. These tools are useful to solve difficult problems the decision taker is faced with. The common point of all these problems is that the taken decision is carried out in a constantly varying world, and thus the initial plan is rarely fulfilled as planned. There are many works in the literature that try to deal with this uncertainty. There are mainly two approaches: react or modify decisions when data is revealed or anticipate data realization explicitly in the solution. We find in the literature several contributions in these two domains. We refer the reader to Grötschel et al., 2001 and Albers, 2003 for the first, and Kall and Wallace, 1994 and Kouvelis and Yu, 1997 and the references therein for the second type of approach. We refer to them as *Reactive Algorithms* (RA) and *Proactive Algorithms* (PA) respectively.

We study in this paper a general framework to deal with this data uncertainty and illustrate the difference with the existing methods on the Shortest Path Problem with Interval Data, which is a simple but widely studied problem that arises in many transportation applications. We then extend the principle to airline scheduling that is a challenging problem taking more and more importance as the airline transport develops and is faced to bigger and harder scheduling problems than ever.

In section 2 we propose a classification of the different approaches we found in the literature. We then consider in section 3 a general optimization problem and we propose a framework to consider RA and PA together. We state the differences between our framework and stochastic optimization with recourse. We provide the motivation on a simple problem, the Shortest Path Problem with Interval Data (SPPID) in section 4 and we extend the concepts to airline schedule optimization in section 5.

2 Algorithm Classification

Given a general optimization problem P subject to data uncertainty it is common to characterize it as an *uncertainty set* U. A particular *realization*, also called a *scenario*, within this uncertainty set is denoted by $u \in U$. We assume, without loss of generality, that we are faced with an uncertain problem for which we want to minimize some cost function. Let S be the set of feasible solutions to the problem and $c_u(s)$ be the cost of solution $s \in S$ of problem P under scenario $u \in U$.

Our first characterization criteria is the nature of this uncertainty set. We distinguish between *Probabilistic Uncertainty Sets* (PUS) and *Non Probabilistic Uncertainty Sets* (NPUS). In PUS, we are given a probability distribution, mapping $u \in U$ into $p(u) \in (0, 1]$, and with $\sum_{u \in U} p(u) = 1$, holding some probabilistic information on the frequency scenario u will occur. Notice that we suppose here the support of U to be discrete for notation simplicity. Recall that for a continuous uncertainty set, on must replace the summation by an integral. In general, in uncertain problems with PUS, the optimal solution to the problem is the one performing best in average over the whole uncertainty set, thus one needs to evaluate the expectation of the cost over the whole uncertainty set.

On the other hand, in NPUS, no probabilistic information is given, we assume to know only the bounds of this uncertainty set, without any frequency indication. Thus, one does not need to evaluate the solution over the whole uncertainty set, but only on the extreme scenarios. The underlying difficulty is to identify these extreme scenarios.

We also distinguish between *Reactive Algorithms* (RA) and *Proactive Algorithms* (PA) as discussed in section 1.

We get four distinct classes, as shown in Table 2.

	Reactive Algorithms (RA)	Proactive Algorithms (PA)
Probabilistic Uncertainty Set (PUS)	Stochastic optimization with Recourse	Stochastic proactive optimization
Non Probabilistic Uncertainty Set (NPUS)	On-Line optimization	Worst-Case Optimization

Table 1: Characterization of the different approaches for an optimization problem under uncertainty.

On-line optimization This class of algorithms is reactive: a new decision is taken according to the revealed data and the previous decisions. Thus, an on-line algorithm usually encodes a decisional strategy rather than a forecast solution. The advantage of these techniques is that this type of situation often occurs in real world. Moreover, they allow to react to any data change in real time. However, it is difficult to measure their performance, as the nature of the scenario is revealed iteratively. The usual way is to compute the *competitivity ratio*, which corresponds to compare the final cost of the obtained solution against the deterministic optimal solution when the scenario that occurred is known. This is clearly an a posteriori performance measure, as one compare the costs once the solution has been computed and carried out. Thus, it is usual to determine bounds on the worst competitivity ratio, which can be tight for some applications (see Albers, 2003). In real world applications this approach performs at acceptable ranges in terms of optimality deviation, but we can usually find scenarios for which the on-line algorithm performs poorly, making it difficult to get any estimates of the costs a priori.

Stochastic optimization with Recourse The main idea of the stochastic optimization with recourse is to include the possibility of taking reactive decisions when a scenario makes the solution unfeasible. The way this is taken into account is to add a constraint violation cost to the overall solution cost. The sum of solution cost and the *expected recourse costs* over the whole uncertainty set has to be minimized (see Kall and Wallace, 1994). The recourse costs are evaluated through experience and the *second-stage problem*, that determines the optimal reaction and its cost to a given one particular scenario, is supposed to be always feasible, i.e. one can always take a recourse decision to make a solution feasible, whatever the solution and the scenario.

We mention here that the definition of stochastic optimization with recourse has slightly different definition according to the applications. In Polychronopoulos and Tsitsiklis, 1996 and Provan, 2003 is given an application of stochastic optimization with recourse to the Shortest Path Problem with Interval Data (SPPID), but in both papers, the technique is presented as a reactive algorithm that encodes a strategy to react to data revealing. We thus classify the method in the reactive class, as it allows reaction and re-optimization after new data is revealed.

Stochastic proactive optimization The aim of stochastic proactive optimization algorithms is to exploit the a priori knowledge about the probabilities of the different scenarios and to compute a solution that has lowest expected cost or that minimizes the probability of high costs over the whole uncertainty set U when carried out without any reaction to any data revealing. This approach implies the evaluation of the expected cost of a solution over the whole uncertainty set, which might be computationally hard.

In both expected cost or high cost probability minimization, the scenarios with low probability have few impact on the optimal solution. If a solution $s \in S$ is unfeasible under a certain scenario $u \in U$, but with a positive probability, it has infinite cost: $c_u(s) = \infty$. Thus, in the expected cost minimization case, if a solution $s \in S$ such that $c_u(s) < \infty$, $\forall u \in U$ exists, then solution s is feasible for every scenario $u \in U$ and the optimal solution also is. On the other hand, high cost probability minimization might lead to a solution with worse potential (higher expected cost), but with the probability of this event being the smallest. See Wallace and Ziemba, 2005, Kall and Wallace, 1994 for details on stochastic algorithms and Laumanns and Zenklusen, 2007 for a high cost probability minimization algorithm.

Stochastic algorithms are useful for both feasibility and cost reducing objectives. However, their main disadvantages are that they need evaluation of the solution on the whole uncertainty set to compute the solution's expected cost, and that they are built on the fact that the expected cost with respect to the random occurrences of the scenarios tends to its mean value according to the probabilistic distribution on the uncertainty set. The stochastic approach is thus useful and a good predictor if we apply the computed solution recursively to many scenarios assuming the probability measure on U remains unchanged, as, in this case, the law of big numbers ensures average costs tends to its expected value. The assumption of the stochastic approach is valid for the case that irreversible structural decisions are made over a long planning horizon or when the decision maker is assumed to be risk neutral (Kouvelis and Yu, 1997), but might predict very badly when the solution is applied only a few scenarios.

Worst-case optimization This approach tends to minimize the maximal possible cost, thus to get an upper cost bound. Similarly to the stochastic algorithm minimizing the expected cost, a worst-case algorithm will find, if it exists, a solution $s \in S$ that is *robust*, i.e. that is feasible and thus with finite cost, for all scenarios $u \in U$. Although the bound on the cost might be very pessimistic, it remains a valid bound, in opposition to the estimated cost of the stochastic solution, where the cost might become dramatically high. Unfortunately, this gain in security translates in a loss of focus on low costs: as we focus on minimizing the upper cost bound in the worst case and we do not consider any probability, we might protect against a scenario that might occur only in extremely rare cases in reality. Moreover, as there is no consideration of better realizations, the solution might have high cost (meaning close to the cost bound) for every scenario, which is in this case a bad property of the robust solution (Kouvelis and Yu, 1997): a solution with a slightly higher worst case cost bound but a much lower best case bound would be much more interesting. In their contribution Bertsimas and Sim, 2004 propose to bound data uncertainty using a box-interval with the additional hypothesis that it is unlikely that all the data changes simultaneously. The approach is similar to the high cost probability minimization but with the objective of worst cost minimization. The authors define a *protection level*, which corresponds to the probability of the solution to be unfeasible.

3 A worst case pro-active method based on a reactive algorithm

We want to focus on a worst-case strategy because we want our solution to be protected against some very nasty scenarios and to bound the costs. The main reason is that determining a valid probabilistic structure of the uncertainty set that matches the nature is extremely difficult and it is a process that usually needs a wide set of observations. However, modeling the uncertainty with stochastic distributions is useful in all situations where it can be done properly (see applications in Wallace and Ziemba, 2005). Moreover, using worst-case measure does not require the evaluation of the solution on the whole uncertainty set but only for the extreme scenarios.

We also want to avoid the reaction process of reactive algorithm, because we don't want the behavior of the solution to be dictated by the nature's realizations: the reason we try to capture some information about the uncertainty is to be able to exploit it as much as possible.

However, we want to keep the modelization of the uncertainty set as simple as possible: we only assume that we know some information about the nature of the scenarios that experience allows us to capture, but we do not try to measure their recurrences.

We are given additional tools: we are able to determine whether and when a solution becomes unfeasible under a certain scenario and we also know the deterministic reactive algorithm, commonly addressed as *recovery algorithm*. It encodes the strategies to recover the unfeasible solution given the disruption point in the scenario. We exploit this knowledge in every scenario with the final goal of finding a solution that has low cost on both the scenarios where it is performed as planned and in the scenarios where some reactive decisions must be taken.

We formalize this concept with the aid of some mathematics. We recall the following notation:

P	the problem to be solved;
$u \in U$	one scenario or realization in the uncertainty set;
$s \in S$	one solution in the set of all possible solutions;
$c_u(s)$	the cost of solution s under scenario u;
c_u^*	the optimal solution to the deterministic problem given scenario u;
$\tilde{c}_u(s)$	the partial cost of solution s under scenario u up to the disruption point;
$c_u^{\mathsf{REC}}(s)$	the additional cost of the recovery algorithm for solution s in scenario u.

$\tilde{c}_u(s)$ is the cost of the solution one has to pay to reach the disruption point that, once a scenario u and a solution s are given, can be evaluated by hypothesis.

We formulate our worst-case optimization problem as follows:

$$(P) \qquad \min_{s \in S} \quad \left\{ \max_{u \in U} \quad \tilde{c}_u(s) + c_u^{\mathsf{REC}}(s) \right\}$$

(P) is indeed worst-case based as it seeks the minimal cost of a solution in the worst possible case. As discussed in section 1, this is a pessimistic objective. Thus, we also want to include some information about more optimistic cases. In fact, in order to try to compensate the paranoiac behavior of worst-case approaches, we add the measure of the best-case approach to the objective function. The point there is that both worst and best cases are extreme scenarios and considering the two extrema simultaneously eventually annihilates the extremum-case effects. We thus focus on the following problem:

$$(P') \qquad \min_{s \in S} \quad \left\{ \max_{u \in U} \quad \tilde{c}_u(s) + c_u^{\mathsf{REC}}(s) \quad + \quad \min_{u' \in U} \quad \tilde{c}_{u'}(s) + c_{u'}^{\mathsf{REC}}(s) \right\}$$

The optimal solution of (P') is the one that minimizes the arithmetical mean between worst and best case over the whole uncertainty set including some reaction costs. Notice that this solution considers the reaction parts in all the scenarios, for the ones that are feasible we have $c_u^{\mathsf{REC}}(s) = 0$. The originality of the above formulation is that we consider the recovery in advance instead of only reacting a posteriori or only trying to find a solution that never needs reaction. In some sense we are planning the solutions which has a low recovery cost in the worst realization; we call it a *recoverable* solution. We refer to this methodology as *recoverable optimization* in the reminder of the paper. In fact, we allow to have additional costs in the worst case, that is no longer simply an unfeasible solution but a solution that is hard (maybe even impossible) to recover, if in the best scenario, this leads to sufficiently savings, which we refer to this as the *potential* of a solution. We thus have a formulation to our uncertain optimization problem P that includes reactive decisions and best-case consideration within a pro-active worst-case framework.

Remark that different objective functions can be considered. It is clear that one should not use $c_u(s)$ instead of $\tilde{c}_u(s)$, as for unfeasible scenarios, $c_u(s) = \infty$ and thus, the recovery costs are pointless and the formulation reduces to find, if it exists, a solution that has finite cost, i.e. that is robust against all scenarios and thus has always $c_u^{\mathsf{REC}}(s) = 0$.

We also rejected the idea of minimizing the difference between the worst case and the best case:

$$\min_{s \in S} \quad \left\{ \max_{u \in U} \quad c_u(s) + \tilde{c}_u^{\mathsf{REC}}(s) \quad - \quad \min_{u' \in U} \quad \tilde{c}_{u'}(s) + c_{u'}^{\mathsf{REC}}(s) \right\}$$

If at least a solution with finite cost exists, then the optimal solution to this problem will be a recoverable solution for all scenarios, which is the desired property of the solution. On the other hand, the objective leads to the solution with least variability instead of a solution with bounded cost. Suppose there is a unique deterministic solution $s \in S$ that is recoverable, i.e. $\max_{u \in U} \tilde{c}_u^{\mathsf{REC}}(s) = \quad \min_{u' \in U} \tilde{c}_{u'}(s) + c_{u'}^{\mathsf{REC}}(s) < \infty$. Then clearly, it is the optimal solution, but the optimality is independent of the cost itself, which can be arbitrarily big. There might be a solution $s' \in S$ with much better potential, i.e. having lower costs than s in both best and worst cases, but with non-zero variability, which is against scheduler's intuition. Moreover, this approach is contradictory to the fact we want to exploit uncertainty, as it is avoiding, potentially by the mean of big costs, the variability.

Another possibility is to seek for a solution that is closest to the optimal solution in the deterministic case, i.e. minimizing the maximal deviation defined by $\max_{u \in U}\{\tilde{c}_u(s) + c_u^{\mathsf{REC}}(s) - c_u^*\}$. In this case, the goal of robustness is still predominant with respect to cost minimization. Although the objective of lowest optimality deviation in the worst case is interesting, especially as it compares the worst case of a solution against another solution, the approach suffers from the same property than the variance minimization, namely that the approach does no longer focus on a proper cost minimization. In their paper Montemanni and Gambardella, 2004 use the same objective function applied to the shortest path problem with interval data which we use in the next section as an illustrative example. They call the solution to this problem a *robust* shortest path. In the literature, it is also referred to as *minimax regret*, see Averbakh and Lebedev, 2004.

4 Application to Shortest Path Problem with Interval Data

Let us illustrate the different concepts on to the Shortest Path Problem with Interval Data (SP-PID) (Karasan et al., to appear and Montemanni and Gambardella, 2004), which is defined as follows:

Let $G = (V, A)$ be an oriented graph, where V is the set of nodes and A is the set of arcs. There is a unique source node $s \in V$ and a unique sink node $t \in V$. The cost c_{ij} of arc (i, j) is not deterministically known, but lies within an uncertainty interval $[l_{ij}, u_{ij}]$, where $l_{ij} \in [0, \infty)$ and $u_{ij} \in [0, \infty]$ (infinite arc cost means that the arc cannot be traversed). In NPUS, we do not have any further information, in PUS, we are additionally given a probability distribution function for every arc. A scenario $u \in U$ is then a set $\{c_{ij}^u \mid c_{ij}^u \in [l_{ij}, u_{ij}], \forall (i, j) \in A\}$, containing one cost realization for every arc within their respective uncertainty sets. Moreover, we suppose that when a probability measure is given, then $P\{c_{ij} = u_{ij}\} > 0$.

We define here some dynamic properties in order to characterize the behavior of both the online algorithm and the recovery one. The cost realization of an arc is revealed when it's origin is reached. In order to ensure at least one feasible solution for every scenario $u \in U$, we suppose that there exists at least one path such that for each of the arcs of the path $u_{ij} < \infty$. In these conditions, we always find a feasible solution. By consequence, we have at least one robust path, leading to a finite solution in both the robust and the stochastic problems. Moreover, if stuck in a dead-end, we can always, in the worst case, use the whole reversed partial path to get back to the origin and thus always find a feasible solution with the reactive algorithms.

Recovery algorithm If a partial path ends up in a dead end (no more outgoing arcs), we are allowed to use the arc used to reach the dead end in reverse sense, at its highest cost u_{ij} and remove the arc from the network.

As this is an recovery decision taken when traversing the path that we want to avoid, we do not consider the possibility of a reverse arc in the proactive problems.

On-line algorithm Take the arc with least cost leaving from the actual node.

The worst scenario for the on-line algorithm now depends on the additional cost of taking and arc backwards.

Stochastic algorithm with recourse Compute the shortest expected path (including recourse, i.e. turn-back at dead-ends) as soon information is revealed from the actual node to the sink node.

The objective is to find the cheapest possible path, which is determined by the type of algorithm that is used.

We consider the example presented in Figure 1, where the uncertainty intervals of every arc are given. We suppose, without any further details, that the probability distributions on the arc cost intervals for the PUS are symmetric and independent. This implies that the mean of an arc cost equals $\overline{c}_{ij} = \frac{l_{ij}+u_{ij}}{2}$.

We show in Table 3 the resulting costs and the average costs for the different approaches applied to a representative sample of realizations given as the cost vectors in Table 2. Recall that for the on-line algorithm and the stochastic algorithm with recourse, the outgoing arc costs are revealed every time a node is reached and a new decision is taken accordingly.

Note that the shortest path in the best scenario (I1) is $\{s, a, b, t\}$ with cost 11, but it is also the path having highest cost in the worst scenario (I2), with cost 42.

The optimal path for the stochastic method is $\{s, e, f, t\}$, with an expected cost of 27. The robust path, minimizing the worst case realization, is paths $\{s, d, t\}$, with upper cost bound being 33.

With the on-line strategy, when a is reached, i.e. when $c_{sa} < c_{sd}$ and $c_{sa} < c_{se}$, arc (a, b) is always chosen next, as for every possible scenario, $c_{ab} \leq c_{ac}$. Moreover, when $c_{sa} > c_{se}$, the

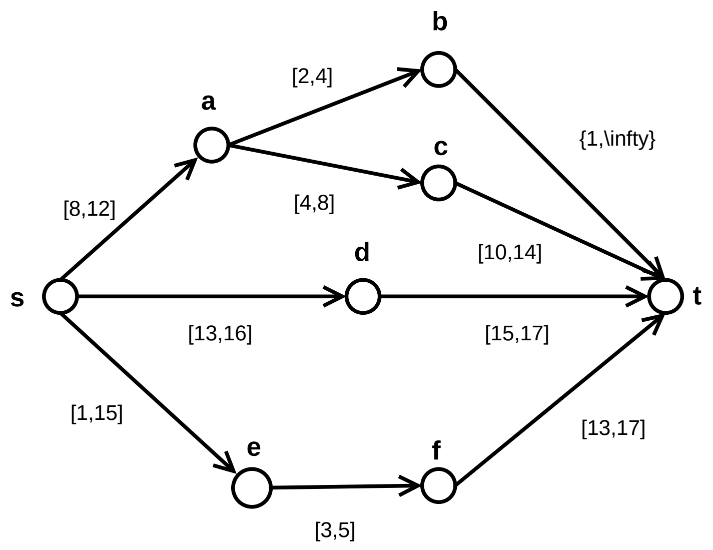

Figure 1: Example of a shortest path with interval data. Arc (b, t) has finite support, taking either value 1 or ∞.

I1	$\{8, 2, 1, 4, 10, 13, 15, 1, 3, 13\}$	every arc is at its lower bound
I2	$\{12, 4, \infty, 8, 14, 17, 19, 8, 5, 17\}$	every arc is at its upper bound
I3	$\{10, 3, 1, 6, 12, 15, 16, 8, 4, 15\}$	every arc takes mean value, (b, t) at lower bound
I4	$\{10, 3, \infty, 6, 12, 15, 16, 8, 4, 15\}$	every arc takes mean value, (b, t) at upper bound
I5	$\{11, 3, 1, 7, 13, 14, 15, 10, 3, 13\}$	
I6	$\{11, 3, \infty, 7, 13, 14, 15, 10, 3, 13\}$	
I7	$\{8, 2, 1, 4, 10, 13, 16, 3, 5, 17\}$	
I8	$\{8, 2, \infty, 4, 10, 13, 16, 3, 5, 17\}$	

Table 2: A sample of scenarios given by the cost vector $\{c_{sa}, c_{ab}, c_{bt}, c_{ac}, c_{ct}, c_{ed}, c_{dt}, c_{se}, c_{ef}, c_{ft}\}$.

on-line strategy leads to the same solution than the stochastic proactive one.

With the stochastic algorithm with recourse, when a is reached, the next taken arc is then either (c, t) with expected cost of $c_{ac} + 12$ or (b, t) with expected cost $\frac{(c_{ab}+1)+c^{\mathsf{REC}}}{2}$, where c^{REC} is the cost of the recourse path $\{a, b, a, c, t\}$ in the case $c_{bt} = \infty$, i.e. $c_{ab} + u_{ab} + c_{ac} + 12$.

Thus, path $\{a, b, t\}$ is chosen if $c_{ac} + 9 \geq 2c_{ab}$, which is always the case. Therefore, the optimal path of the stochastic algorithm with recourse is either $\{s, a, b, t\}$ or $\{s, e, f, t\}$, depending on the realization of c_{sa} and c_{se}: if $c_{sa} < c_{se} + 4.5$ then the chosen path is $\{s, a, b, t\}$, otherwise it is path $\{s, e, f, t\}$.

With the recoverable algorithm we compute a path prior to any cost revealing but considering the recovery in case a dead-end is encountered. In this case, the optimal path is $\{s, a, b, t\}$, with potential cost (sum of the best and worst case scenarios) of $11 + 42 = 53$. Path $\{s, a, c, t\}$ has cost 56, path $\{s, d, t\}$ a cost of 61 and path $\{s, e, f, t\}$ a cost of 54. Remark that for the paths

Method	I1	I2	I3	I4	I5	I6	I7	I8	Average
On-Line	17	42	27	27	26	26	25	25	26.875
Recourse	17	42	14	35	15	38	25	25	26.375
Stochastic	17	37	27	27	26	26	25	25	26.250
Robust	28	33	31	31	29	29	29	29	29.875
Recoverable	11	42	14	35	15	38	11	28	24.250

Table 3: Cost of the different methods for different scenarios and average cost over the considered scenarios.

were no dead-end is met in the worst case, the cost of the path is simply twice it's mean cost. This is due to the fact the distributions are assumed to be symmetric. Thus, when no recourse is needed in any of the scenarios and the distributions are symmetric, then the recoverable path will be the same than the proactive stochastic shortest path.

This simple example illustrates the differences of the approaches. We see how the realization of the first arc determines how good (or bad) behave the reactive algorithms.

Note that in most of the cases, arc (s, d) has lowest cost, which explains why the on-line algorithm mainly follows the path $\{s, e, f, t\}$ and thus leads to the same results than the proactive stochastic solution.

Moreover, the stochastic solution and the robust path have fairly low variance on this sample. The stochastic path is often the shortest path when $c_{ab} = \infty$, although in its worst scenario the proactive stochastic path has cost 37, as even when arc (b, t) is untraversable, we can get a cost of 28 for the recovered path $\{s, a, b, a, c, t\}$. This shows that the possibility of arc (b, t) to become untraversable affects highly the proactive stochastic solution.

The robust path, on the other hand, is by definition the shortest path in the worst scenario (I2), but it has always a high cost, which translates in a significant higher average cost over the sample of instances we used: from 11% up to 23% higher than the other methods.

The path leading to the highest cost is path $\{s, a, b, t\}$, with cost 42. This is because when $c_{bt} = \infty$ one must pay the recourse fee of 4 and then follow the non-optimal path $\{a, c, t\}$, as there is no alternative. However, in more optimistic scenarios, it is the path leading to the cheapest solutions. The recoverable path has thus higher variability than the stochastic or the robust ones, but according to our potential measure (sum of worst and best case), it is the most interesting.

The difference of the stochastic approach with recourse and the recoverable one is relevant in instances I7 and I8: due to the known arc costs $c_{sa} = 8$ and $c_{se} = 3$, with the stochastic algorithm with recourse, path $\{s, e, f, t\}$ has better potential in average, with a total expected cost of 22 against 22.5 for path $\{s, a, b, t\}$. We see that the cost of path $\{s, a, b, t\}$ highly depends on the cost of arc c_{bt}. If $c_{bt} = \infty$ (I8), then indeed path $\{s, e, f, t\}$ has always lower cost, but in this case the saving is only of 3. In the scenario where $c_{bt} = 1$ (I7), we see however that the cost difference is significantly higher, path $\{s, a, b, t\}$ leading to a save of 14, which is more than 50% less than path $\{s, e, f, t\}$. This better potential is precisely the reason the recoverable path is path $\{s, a, b, t\}$.

Note that if we consider the stochastic method with recourse in a proactive way, i.e. minimizing the sum of expected path length plus expected recovery costs over all scenarios without recomputing a solution at every node, then we get the same solution than the recoverable path: knowing the recourse function (or recovery algorithm), the expected cots including recourse expectation of path $\{s, a, b, t\}$ is 24.5, which is clearly lower than the expected cost of path $\{s, e, f, t\}$, which has least expected cost of 27.

The differences of the presented approaches a clearly shown through this example. In a more general case, we see that both proactive stochastic and robust methods will find the shortest path in a modified graph where all the potentially untraversable arcs are removed (for the stochastic cas this holds as long as $P\{c_{ij} = u_{ij}\} > 0$ holds). The reactive algorithms, on the other hand, have unpredictable behavior. The solution is guided by the realization of the arc costs, which is a property we want to avoid. The recoverable solution shows to be both the best and the worst path according to the situation, but outperforms the other methods on the presented instances as it does over the whole uncertainty set.

5 Applying Recoverable Optimization to Airline Scheduling

The example of the previous section shows how to apply recoverable optimization on a simple problem where the recovery costs can be computed easily.

In more general problems though, the recovery algorithm usually becomes a hard problem itself. Indeed, when we formulate the recoverable problem as in (P'), the evaluation of the terms $c_u^{\mathsf{REC}}(s)$ implies the evaluation of a recovery problem given a solution s and the scenario u. Moreover, we have to determine at which point the solution becomes unfeasible: as a proactive scheduler we are able to evaluate whether a given solution is feasible for a given scenario and, in the latter, when the feasibility is lost and what the costs are up to this disruption point.

This leads to the study of scenario characterization, where one tries to identify in a deterministic way depending on the uncertainty set and the recovery algorithm, which scenarios lead to the best and to the worst cases respectively. As to recover from unfeasible solution is costly, it usually makes sense that, in the best realization, $c_u^{\mathsf{REC}}(s) = 0$. We thus are left with the problem of characterizing the scenario leading to the worst possible recoverable solution.

This holds for airline scheduling as well as form many other scheduling problems. The reason we develop the concept on airline scheduling is because we recently addressed a recovery algorithm for the Airplane Recovery Problem (ARP) (Bierlaire et al., 2007).

Airline scheduling is a complex and challenging optimization problem. The usual approach in practice is to divide the problem into several smaller subproblems which are solved iteratively according to their due dates. The first problem to solve is the route choice problem, when airline managers determine the legs to be flown, which is usually done 6 to 12 month in advance. Then, routes must be affected to the planes and this is done in two stages: first a fleet (i.e. a type of plane) is associated to a set of flights, and then the routes for every single plane are computed, which is done 2-3 months in advance. Finally the crew pairing and the crew roistering problems are solved to affect crews to flights. Airline schedules are usually computed with the aim of minimizing operational costs but often unpredicted events, called *disruptions*, make the schedule unfeasible and some recovery decisions must be taken in order to get back to the initial schedule. We recently addressed the recovery problem for the ARP in Bierlaire et al., 2007 and introduced a column generation based algorithm that solves the ARP. The underlying pricing problem is a dynamic programming algorithm that computes elementary resource constrained shortest paths in a so called *recovery network* generated for every plane.

These networks encode all feasible routes for one single plane. We then use a dynamic programming algorithm based on Decremental State Space Relaxation (DSSR) algorithm in Righini and Salani, 2005 to compute the solution to each pricing problem.

We want to extend the concept of recoverable solution presented in section 3 to the airline scheduling problem having the knowledge on the recovery algorithm for the ARP. We thus want to find a schedule for planes, i.e. a successions of flights and maintenances for every plane, such that whatever the scenario of a given uncertainty set, the solution either remains feasible or is recoverable (using the mentioned recovery algorithm) at limited costs.

There are three underlying difficulties. The first one is to determine when a schedule becomes unfeasible given a scenario and compute associated partial costs, the second being, of course, to solve the underlying recovery problem. The last difficulty is to characterize which scenario is the worst for a given schedule s.

To answer the feasibility question is not trivial. One suggestion is to perform a feasibility test where only delays are allowed. If some rule on these delays is not violated, then we consider the solution as feasible, but we add the corresponding delay costs as the recovery costs.

The worst scenario characterization is much more difficult as it is highly dependent on both the structure of the uncertainty set and the recovery algorithm itself, which makes a general characterization impossible. Unfortunately, due to the complex formulation of the recovery algorithm in the airline scheduling case, the problem $\max_{u \in U} \quad c_u(s) + c_u^{\mathsf{REC}}(s)$ given a schedule $s \in S$ is highly non-linear as, of course, the recovery decisions depend on the scenario, and thus the variables of the underlying problem are both the recovery decisions and the scenario coefficients.

One solution to solve this problem is to evaluate the recovery cost for every scenario $u \in U$, which implies firstly that the uncertainty set has finite support, and that we need $\prod | U || S |$ computations of an NP-hard problem. This is of course not affordable. We thus want to look at alternative ways to cope with this problem.

One idea is to sample the scenarios according to some properties that we know being hard for the recovery algorithm, and solve the recovery problem only for a selection of scenarios. However, one has to be careful the way the sample is chosen as when protecting against only the worst case, we might get a solution that performs worse in a scenario that was not considered than the one we are protected against.

The extension on the above principle is to apply it earlier, when determining the uncertainty set. Instead of sampling a given set, one might try to structure the uncertainty set in order to make the computation easier. This can be done through bounding on total amount of scenarios for example or by bounding the worst case as did Bertsimas and Sim, 2004. However, the same care on the characterization has to be taken than for the sampling mentioned previously.

However, as the information needed to determine the optimal schedule s is only its partial operation costs $\tilde{c}_u(s)$ and its recovery cost $c_u^{\mathsf{REC}}(s)$, and not the nature of the recovery decisions themselves, we try to estimate these costs with a simpler algorithm. Although we must be careful not to consider too elementary estimations. Indeed, in this framework we are trying to exploit the nature of the recovery algorithm in order to build a less costly solution in case recovery is needed. Approximating this information is equivalent to approximate the final cost under a given scenario and as this is what we want to minimize, if the approximation is bad, the final solution might be much more costly than expected in reality.

For example, we discard greedy measures that might be related to schedule feasibility, as $c_u^{\mathsf{REC}}(s) = C$ (with C a constant) when s is unfeasible for scenario u and $c_u^{\mathsf{REC}}(s) = 0$ otherwise. The reason is that for this kind of measure, the optimal solution to the initial problem tends to find the solution remaining feasible to the most possible scenarios without any information about the true recovery costs, which turns out to be the robust solution. We thus loose the information about the recoverability that justifies our approach.

Another approach is to define more schedule-based measures that help to predict the performance of the recovery algorithm. For example, we measure the structure of the network associated to one schedule in terms of number of plane crossings (at a same airport and the same time) and the average grounding time for the planes. The first indicator helps measuring the number of possible airplane swappings that are possible. The more there are, the better for the recovery algorithm, as it considers plane swappings. The second indicator captures the density of the schedule and thus estimates the un-activity gaps in the schedule that are useful to absorb delays.

By doing so, we are in fact approximating the recovery costs through auxiliary measures that are easy to compute. With this approach we are limiting the complexity of the scenario-based evaluation of $\tilde{c}_u(s)$ and $c_u^{\mathsf{REC}}(s)$ in order to keep the problem tractable. We thus replace the minimization of the partial and recovery costs in (P') by the maximization of the mentioned auxiliary objectives. Thus, we get rid of the NP-hard recovery problem to evaluate $c_u^{\mathsf{REC}}(s)$ by introducing some secondary objectives. The advantage of the multi-objective approach is its computational tractability compared to the recoverable one. Moreover, this approach leads to the generation of a set of Pareto optimal solutions rather than a unique one. The disadvantage is that we do not capture the information of the recovery algorithm and the uncertainty set explicitly. Thus it is hard to exploit well the given information in an implicit way.

Finally, we can use the network-based measures directly on the recovery networks used to solve the ARP in Bierlaire et al., 2007 which might lead to more explicit measures of recoverability, and still has the advantages of the multi-objective approach.

We see that the main challenge of the recoverable approach applied to more complicated problems such as the airline scheduling problem is its computational complexity. The proposed approaches here are only preliminary hints of possible research directions we want to explore.

6 Conclusions

In this exploratory paper we first give a classification of the existing methods to address decisional problems subject to uncertainty. This motivates the definition of the recoverable approach we address to attack these kind of problems with a non probabilistic proactive methodology that is based on the knowledge of the reaction strategy in case a disruption occurs. This allows to compute a solution that is robust for a subset of scenarios but that we know to be recoverable at low costs for the remaining scenarios, which is the originality of the methodology.

We give a comparative illustration of our methodology compared to the existing methods with an application to the shortest path problem with interval data, and then give a preliminary set of directions to explore for the application of the methodology on more complicated problems, in particular to the recoverable airline scheduling problem.

We plan to explore more deeply the field both from theoretical and practical point of view. We intend to review more carefully the aspects of stochastic programming and robust optimization and compare our findings with what has been done so far. We intend to validate the approach with a practical application to airline optimization: we have the access to real world data, we have a recovery algorithm and we have a set of auxiliary measures for recoverability. Thus, we can optimize the schedule considering the exact recovery costs and the auxiliary measures given a set of disruption scenarios.

References

Albers, S. (2003). Online algorithms: A survey, *Mathematical Programming* **97**: 3–26. Invited paper at ISMP 2003.

Averbakh, I. and Lebedev, V. (2004). Interval data minmax regret network optimization problems, *Discrete Appl. Math.* **138**(3): 289–301.

Bertsimas, D. and Sim, M. (2004). The price of robustness, *Operations Research* **52**: 35–53.

Bierlaire, M., Eggenberg, N. and Salani, M. (2007). Airline disruptions: aircraft recovery with maintenance constraints, *Proceedings of the Proceedings of the Sixth Triennial Symposium on Transportation Analysis*, Phuket, Thailand.

Grötschel, M., Krumke, S. O. and Rambau, J. (eds) (2001). *Online Optimization of Large Scale Systems*, Springer.

Kall, P. and Wallace, S. (eds) (1994). *Stochastic Programming*, John Wiley & Sons, New York, N.Y.

Karasan, O., Pinar, M. and Yaman, H. (to appear). The robust shortest path problem with interval data, *Computers and Operations Research* .

Kouvelis, P. and Yu, G. (1997). *Robust discrete optimization and its applications*, Kluwer Academic, Dordrecht.

Laumanns, M. and Zenklusen, R. (2007). Estimation of small s-t reliabilities in acyclic networks, *Technical report*, ETH Zurich, Institute for Operations Research, ETH Zurich, Institute for Operations Research, 8092 Zurich, Switzerland.

Montemanni, R. and Gambardella, L. M. (2004). An exact algorithm for the robust shortest path problem with interval data, *Computers and Operations Research* **31**(10): 1667–1680.

Polychronopoulos, G. H. and Tsitsiklis, J. N. (1996). Stochastic shortest path problems with recourse, *Networks* **27**(2): 133–143.

Provan, J. S. (2003). A polynomial-time algorithm to find shortest paths with recourse, *Networks* **41**(2): 115–125.

Righini, G. and Salani, M. (2005). New dynamic programming algorithms for the resource constrained shortest path problem, *Technical Report 69*, Università degli Studi di Milano. accepted for publication on Networks.

Wallace, S. W. and Ziemba, W. T. (2005). *Applications of Stochastic Programming (Mps-Siam Series on Optimization) (Mps-Saimseries on Optimization)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.