Project Number 288094

eCOMPASS
eCO-friendly urban Multi-modal route PlAnning Services for mobile uSers

STREP
Funded by EC, INFSO-G4(ICT for Transport) under FP7

eCOMPASS – TR – 014

# Robust optimization in the Presence of Uncertainty

Joachim Buhmann, Matus Mihalak, Rastislav Sramek, Peter Widmayer

January 2013

# Robust Optimization in the Presence of Uncertainty

Joachim M. Buhmann
Department of Computer Science,
ETH Zurich,
Zurich, Switzerland
jbuhmann@inf.ethz.ch

Matúš Mihalák
Department of Computer Science,
ETH Zurich,
Zurich, Switzerland
mmihalak@inf.ethz.ch

Rastislav Šrámek
Department of Computer Science,
ETH Zurich,
Zurich, Switzerland
rsramek@inf.ethz.ch

Peter Widmayer
Department of Computer Science,
ETH Zurich,
Zurich, Switzerland
widmayer@inf.ethz.ch

## ABSTRACT

We study optimization in the presence of uncertainty such as noise in measurements, and advocate a novel approach of tackling it. The main difference to any existing approach is that we do not assume any knowledge about the nature of the uncertainty (such as for instance a probability distribution). Instead, we are given several instances of the same optimization problem as input, and, assuming they are typical w.r.t. the uncertainty, we make use of it in order to compute a solution that is good for the sample instances as well as for future (unknown) typical instances.

We demonstrate our approach for the case of two typical input instances. We first propose a measure of similarity of instances with respect to an objective. This concept allows us to assess whether instances are indeed typical. Based on this concept, we then choose a solution randomly among all solutions that are near-optimum for both instances. We show that the exact notion of near-optimum is intertwined with the proposed measure of similarity. Furthermore, we will show that our measure of similarity also allows us to derive formal statements about the expected quality of the computed solution: If the given instances are not similar, or are too noisy, our approach will detect this. We demonstrate for a few optimization problems and real world data that our approach works well not only in theory, but also in practice.

## Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; I.2.6 [**Artificial Intelligence**]: Learning

## Keywords

Optimization; Uncertainty; Noise; Robustness; Instance similarity

## 1. INTRODUCTION

Inaccuracy and uncertainty of data in input instances of optimization problems are a curse, but a reality. How can we even hope to solve a problem optimally if the input data are noisy? So much for the traditional opinion.

In this paper, we argue that the opposite is often true: Uncertainty can be a blessing that allows us to reach meaningful solutions. In reality, it is mostly an illusion that input data are accurate or that solutions can be implemented accurately. For instance, edge weights of a graph might represent travel time measurements. A shortest path from a start vertex to a target is used to indicate the fastest trip, but might turn out not to be the best choice if the trip is taken in the future, or if the measurements were noisy. For real world problems, this dilemma questions the standard approach of optimizing for a certain objective: Why should it be best to find an optimum solution for an input instance whose data come from measurements that we cannot realistically assume to be exact? And, on the same note: What should the computational goal be in this case?

Viewed from a machine learning perspective, optimization under uncertainty pursues the conflicting goals of optimizing an objective function while keeping the result statistically robust. Optimization must naturally adapt solutions to the last detail, and therefore to fluctuations in the data. That objective is the reason why solutions need mathematical regularization to avoid over-fitting.

In this paper, we advocate an approach that attempts to extract information from data, to eliminate noise, and thereby to reach a typical and robust solution rather than a solution that is geared far too strongly towards a noisy input. Our approach has the following four key properties: (1) It is based on a few input samples only (at least two), (2) it needs no tweaking of parameters, (3) it needs no knowledge on probability distributions of input instances, whether systematic (like travel times that depend on weather) or random, and (4) it provides a self-assessment of how typical and meaningful its solution is. We reach our goal through a notion of *information with respect to an objective*, through a measure that we call the problem-based similarity of instances. Our approach at the same time provides for a purely algorithmic justification of a pioneering information theoretic method [1, 2], as well as for a refinement of this method for optimization problems that exhibit structure in the solution space. This approach reflects the understanding that averaging over fluctuating solutions with near optimal objective function values promises to stabilize the optimization process. Both strategies together, optimization and fluctuation averaging, enhance the predictive value of solutions in the sense that optimized, stable solutions should also yield good objective function values on new problem instances. In this sense, we interpret optimization under uncertainty as a prediction problem.

For a few examples of optimization problems, we demonstrate what it means to use our approach in solving their uncertain versions. It turns out that on one hand, interesting algorithmic problems arise. On the other hand, we get solutions that tend to be only a few percentage points from optimum for real world input instances, and that are better than any competing method that we could imagine.

### Related work.

Noisy inputs received attention in a variety of ways. *Stochastic optimization* [3, 4] investigates the fortunate situation in which a distribution of input data is known, and it mostly aims at the optimum in expectation. *Info-gap decision theory* is peculiar in that it models uncertainty as an information gap rather than a probability [5]. *Robust optimization* [6] explicitly describes the uncertainty that is inherent in the input data. In the discrete version, we get a discrete set of input instances for which the structure of a solution must be chosen (such as a path from start to target vertex), and after this choice, one of the given instances is revealed as the true one. A worst case perspective in which an adversary will reveal the worst possible instance, given the chosen solution, is far too pessimistic for real world situations. *Recoverable robust optimization* [7] pursues this approach one step further and allows the algorithm to modify the proposed solution after the true instance has been revealed. This adaptation enables us to at least generate a feasible solution in all cases, while for robust optimization in general, feasibility cannot be guaranteed. *Optimization for stable inputs* attempts to understand when and how small changes in the input data affect the solution [8, 9, 10, 11, 12, 13]. *Regret minimization and online learning* focuses on finding a good solution in an online setting, revealing one input after another, assuming an arbitrary adversary [14]. Here, for a meaningful analysis, necessarily more instances are needed.

## 2. A BIRD'S EYE VIEW OF OUR APPROACH

### Generator.

We are given two input instances of an optimization problem. The two instances are related in an unknown way, like two travel time measurements of the same set of routes at different times. We will consider them to be generated by some abstract *generator*. Given both generated instances, the goal is to compute a good solution for the next (unknown) generated instance. The generator induces noise that we do not know how to predict or eliminate. Specifically, we do not postulate any characteristics of the noise (such as additive, Gaussian, or independent on different parts of the input instance). We assume that the generated instances have the same structure, and differ only in measured values (e.g., for an optimization problem on graphs, the structure of the graph is the same, but edge weights may differ).

### Optimization problems.

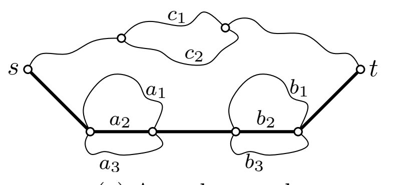
In our setting, an *uncertain minimization problem* is given by $(\mathcal{I}, S, c)$, where $\mathcal{I}$ is the set of all instances (e.g., all edge-weighted graphs of the same topology), $S$ is the set of all feasible solutions of any instance $I \in \mathcal{I}$, i.e., we assume that a solution $s \in S$ is a feasible solution for every instance $I \in \mathcal{I}$ (e.g., all simple paths from vertex $s$ to vertex $t$), and $c(s, I)$ is the cost of solution $s \in S$ in an instance $I \in \mathcal{I}$ (e.g., the sum of the edge-weights in instance $I$ along a path $s$). Note that the set of instances as well as the set of feasible solutions can have infinite size, for instance if we allow arbitrary weights and solutions are points in continuous space. Given an instance $I$ of a (standard) minimization problem, the goal is to find a feasible solution $s^*$ of minimum cost $c(s^*, I)$; we call such a solution a *minimum* of $I$. The goal of our approach for an *uncertain* minimization problem, however, is to find a feasible solution that is close to optimum for a typical instance of the given minimization problem. We define an *uncertain maximization problem* analogously, and by *optimization* we refer to either minimization or maximization. Note that in our definition, we only look at a class of instances of fixed size. Whenever we argue about problem complexity, we will therefore consider the union of problems of all topologies and sizes, as usual.

### Two instances.

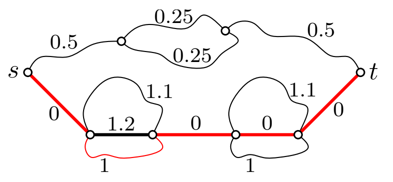
By looking at a single input instance we cannot hope to separate the noise from the information: the input could be fully determined by noise and we would not be able to tell that it is so. We therefore assume that several instances are given as input – these instances are considered to be representative, typical instances of the problem. In this paper we explain our approach for two instances.

### Example.

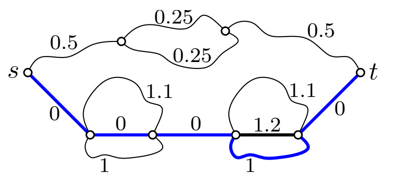
Consider the problem of driving from town $s$ to town $t$ in the road network $G = (V, E)$ of Figure 1a. The path through $a_2$ and $b_2$ is a highway, while the other edges are local roads. The edge weights in Figures 1b and 1c are driving times. Driving on the highway is usually fast. Due to maintenance, however, edges $a_2$ and $b_2$ are prone to traffic jams. A feasible solution is an $s$-$t$ path. In total, there are eleven $s$-$t$ paths (the nine paths using at least partially the highway, and the two paths through local roads $c_1$ and $c_2$). An optimum solution is a path of smallest driving time. In case of no traffic jam the highway is the (only) optimum solution. We focus our investigation on the situation where congestion appears. Figures 1b and 1c are two instances $I_1$ and $I_2$ for two consecutive days where congestion appears



(a) A road network



(b) Instance $I_1$ and its optimum path



(c) Instance $I_2$ and its optimum path

Figure 1: A fastest path from $s$ to $t$

on the first and the second part of the highway, respectively. The question is now how to drive on the third day.

## *Similarity of instances.*

Given two typical instances, we aim at extracting the relevant information and at discarding the irrelevant noise. Since we lack any other knowledge, we look for common features with respect to the goal of close to optimum solutions. We define closeness to optimum by means of an *approximation ratio* $\rho$: A near optimum solution for an instance $I$ is a *$\rho$-approximate solution*, i.e., a solution $s \in \mathcal{S}$ with $\text{cost}(s, I) \leq \rho \cdot \text{cost}(s^*, I)$, where $s^* \in \mathcal{S}$ is an optimum solution for instance $I$. The appropriate $\rho$ is problem and generator dependent and unknown to us. We will call the set of all $\rho$-approximate solutions of an instance $I$ a *$\rho$-approximation set of* $I$, and denote it by $A_\rho(I)$.

Thus, the common features of instances $I_1$ and $I_2$ (with respect to our optimization goal) are feasible solutions that lie in the intersection $A_\rho(I_1) \cap A_\rho(I_2)$ of the respective $\rho$-approximation sets. While one might further investigate the structure of this intersection to capture the information contained in it, we will see that already the number of solutions in this intersection provides a considerable amount of information. Naturally, we expect that for approximation ratio close to 1, the approximation sets, and in particular their intersection, even for unrelated instances, tend to be small, while for approximation ratio far from 1, they tend to be large. When the size of the intersection of both approximation sets is farthest from this expectation, we learn the most from it. Therefore, we choose the value of $\rho$ accordingly. The following definition summarizes this intuition.

Definition 1 (Problem-based instance similarity). *Let $I_1$ and $I_2$ be two input instances of an uncertain combinatorial optimization problem $\mathcal{P} = (\mathcal{I}, S, c)$. For a given $\rho$, let $A_\rho(I_1)$ and $A_\rho(I_2)$ be $\rho$-approximation sets for $I_1$ and $I_2$. Further, let $es(\rho, a, b)$ be the expected size of the intersection of two $\rho$-approximation sets of sizes $a$ and $b$ of two arbitrary instances $I_a, I_b \in \mathcal{I}$, respectively, where the expectation is over all approximation sets of the respective sizes, using the uniform probability distribution. Then, the expression*

$$U_\rho(I_1, I_2) := \frac{|A_\rho(I_1) \cap A_\rho(I_2)|}{es(\rho, |A_\rho(I_1)|, |A_\rho(I_2)|)} \tag{1}$$

*is the* unexpected similarity of $I_1$ and $I_2$ at value $\rho$ *(with respect to the optimization problem $\mathcal{P}$), and the expression*

$$U(I_1, I_2) := \max_\rho U_\rho(I_1, I_2) \tag{2}$$

*is the* unexpected similarity of $I_1$ and $I_2$ *with respect to the optimization problem $\mathcal{P}$.*

We call $U(I_1, I_2)$ the *unexpected similarity* due to the fact that we normalize the absolute similarity, expressed as the number of solutions in $A_\rho(I_1) \cap A_\rho(I_2)$, with the expected similarity, expressed as the expected size of the intersection. Note that $U(I_1, I_2) \geq 1$ as we can always set $\rho$ so that both approximation sets contain all solutions. The usage of the uniform probability distribution in the definition of the expected intersection follows the Laplace's principle of indifference, as we do not have further information about the generator.

## *Example (continued).*

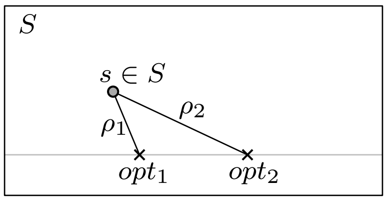
In our examples in Figures 1b and 1c, the optimum paths for the two instances are different, but both have the same travel time, namely 1. For $\rho = 1.1$ we get two paths in $A_\rho(I_1)$, namely the path through $a_3$ and $b_2$ and the path through $a_1$ and $b_2$. Similarly, $A_\rho(I_2)$ contains paths through $a_2$ and $b_3$ and through $a_2$ and $b_1$. These approximation sets have no path in common. For $\rho = 1.2$, only the highway through $a_2$ and $b_2$ is in both $A_\rho(I_1)$ and $A_\rho(I_2)$. Observe that $\rho = 1.2$ is the smallest value for which $A_\rho(I_1) \cap A_\rho(I_2) \neq \emptyset$. This may appear to be a natural choice of $\rho$: Pick the smallest $\rho$ for which $A_\rho(I_1) \cap A_\rho(I_2) \neq \emptyset$. Then, pick a path from the intersection $A_\rho(I_1) \cap A_\rho(I_2)$ – the highway in our case. In our example, this path is risky: It will reach a cost of 2.4 if traffic jams appear in both parts of the highway. Instead, one should also consider to drive on the local roads only (the paths through $c_1$ and $c_2$), resulting in a driving time of 1.25. We will see that our method will choose $\rho = 1.25$ as the approximation ratio that maximizes $U_\rho(I_1, I_2)$.

## *Computing a solution.*

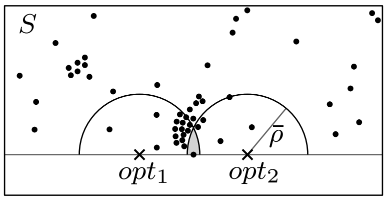
We choose a solution from the intersection of the two approximation sets uniformly at random. In the absence of any knowledge about the generator, this is a natural choice following Laplace's principle of indifference. Naturally, we want to choose $\rho$ so that the chance of getting a good solution is maximized.

## *Choosing $\rho$.*

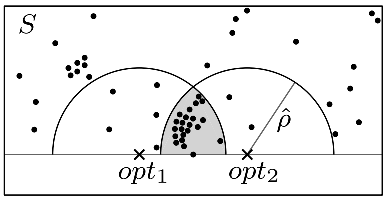
When the two instances are indeed similar, they have many good solutions in common. Some solutions in the intersection $A_\rho(I_1) \cap A_\rho(I_2)$, however, are present simply because $\rho$ is large. Let us call such solutions *expected*. The good solutions are then the ones in addition to the expected ones – the *unexpected* solutions. Now the goal is to find the



(a) Placing a solution $s$



(b) $A_{\bar{\rho}}(I_1) \cap A_{\bar{\rho}}(I_2)$



(c) $A_{\hat{\rho}}(I_1) \cap A_{\hat{\rho}}(I_2)$

Figure 2: Approximation sets for similar instances $I_1$ and $I_2$. (a): We place the respective optimum solutions $opt_1$ and $opt_2$ on a horizontal line at an arbitrary distance. Each remaining point $s \in S$ is then placed above the horizontal line at distance $\rho_1$ from $opt_1$ and $\rho_2$ from $opt_2$, where $\rho_1$ and $\rho_2$ are the corresponding approximation factors that solution $s$ achieves. (b) and (c): Examples of approximation sets $A_\rho(I_i)$, $i = 1, 2$, $\rho = \bar{\rho}, \hat{\rho}$. Warning: distances between solution points other than those to $opt_i$ are meaningless.

value of $\rho$ that will maximize the proportion of good (unexpected) solutions in the intersection, since this maximizes the chances to draw a good solution at random. Figure 2 illustrates these ideas.

We have no direct way of counting the number of unexpected solutions in $A_\rho(I_1) \cap A_\rho(I_2)$, but we can try to compute an expectation of the size of the intersection of any two *feasible approximation sets* $A$ and $B$ of the same sizes as $A_\rho(I_1)$ and $A_\rho(I_2)$, respectively. Here, a feasible approximation set $X$ is a subset of $S$ which is a $\rho'$-approximation set of $I'$ for some instance $I'$ and some value $\rho'$.

Given $\rho$ and the sizes $|A_\rho(I_1)| =: k(\rho)$ and $|A_\rho(I_2)| =: l(\rho)$, let $es(\rho, k(\rho), l(\rho))$ denote the expected size of the intersection of two feasible approximation sets $A$ and $B$ of sizes $k(\rho)$ and $l(\rho)$, respectively. If the intersection of $A_\rho(I_1)$ and $A_\rho(I_2)$ is larger than the expected number $es(\rho, k(\rho), l(\rho))$, then it contains some unexpected solutions.

Denoting the number of unexpected solutions in the intersection with $u(\rho)$, we want to find $\rho$ that maximizes $\frac{u(\rho)}{u(\rho)+es(\rho,k(\rho),l(\rho))}$. This is the same as maximizing the ratio

$$\frac{u(\rho) + es(\rho, k(\rho), l(\rho))}{es(\rho, k(\rho), l(\rho))} = \frac{|A_\rho(I_1) \cap A_\rho(I_2)|}{es(\rho, k(\rho), l(\rho))} = U_\rho. \qquad (3)$$

Thus, the value of $\rho$ that maximizes the chances of picking a good solution is the same as the value that defines the unexpected similarity of instances.

*Example (continued).*

Recall that the first non-empty intersection $A_\rho(I_1) \cap A_\rho(I_2)$ appears for $\rho = 1.2$ and contains the path through $a_2$ and $b_2$. The unexpected similarity is $U_{1.2} = \frac{1}{6219/87^2} \approx 1.217$ (one can check this for instance by brute-force enumeration). For $\rho = 1.25$, the approximation sets get larger by the two paths through $c_1$ and $c_2$. The unexpected similarity is then $U_{1.25} = \frac{3}{71307/177^2} \approx 1.318$. Tedious calculations show that this value maximizes $U_\rho$. We therefore pick one of the three $s$-$t$ paths from the intersection $A_{1.25}(I_1) \cap A_{1.25}(I_2)$ uniformly at random, arguably a good choice, which balances (i) the quality and (ii) the robustness, i.e., respectively, the cost of travelling and the risk of travelling too long (if the congestion appears on both parts on the highway).

*Self-assessment.*

If the unexpected similarity is near 1.0, we know that the two instances are not unusually similar. This limit may be because the noise level is very high, or the instances are actually unrelated. On the other hand, if the unexpected similarity is large, then the instances are similar, i.e., the noise level is low and the intersection contains good solutions for a third instance, generated by the same generator. Unexpected similarity is thus at the same time the key to choosing a good solution with the highest probability, and the measure of the expected quality of the chosen solution. Both are rare properties among known generalization and prediction methods.

## Recapitulation of the general approach

For an uncertain optimization problem $(\mathcal{I}, S, c)$, proceed as follows:

1. Determine the domains $\mathcal{F}_k$ of feasible approximation sets of size $k$.
2. Provide a mathematical analysis or an algorithm $ALG_\mathbb{E}$ that computes the expected size of the intersection of two approximation sets of given sizes $k$ and $l$.
3. Provide an algorithm $ALG_\cap$ that computes the size of the intersection $A_\rho(I_1) \cap A_\rho(I_2)$, given $\rho$ and two instances $I_1$ and $I_2$.
4. Find (smallest) $\rho^*$ that maximizes the unexpected similarity $U_\rho$, using $ALG_\mathbb{E}$ and $ALG_\cap$.
5. Provide an algorithm $ALG_{\text{rand}}$ that picks a random solution from the intersection $A_{\rho^*}(I_1) \cap A_{\rho^*}(I_2)$.

In Section 3 we discuss in more detail what it means to follow this approach, and in particular how to determine the expectation in point 2.

## A prototypic example

We expect our method to exceed the performance of other optimization methods, when the set of good solutions that have stable cost over all or most instances is large enough not to be completely hidden in the noise, but there is a certain number of unstable solutions with wildly differing costs which might out-perform the stable solutions with low probability. When the cost of a large number of solutions is very random, a few of them might sometimes eclipse the true, stable solutions. Usual combinatorial optimization methods go straight for the single solution that minimizes the cost, often missing the truly optimal solutions in the process.

In the following we give a concrete example of a very simple instance generator which illustrates such a situation. We consider an uncertain minimization problem $(\mathcal{I}, S, c)$, with a set of solutions $S = G \cup B$, $|G| =: g \ll b := |B|$, and a cost function $c(s, I) \in \{1, \hat{\rho}, 2\}$, $1 < \hat{\rho} < 2$, for every instance $I$ and every solution $s$. An instance $I$ is generated with the following random process. The cost of every solution $s \in G$ is $\hat{\rho}$ (always). The cost of a solution $s \in B$ is 1 with probability $p$ and 2 with probability $(1 - p)$. We set $\hat{\rho}$ and $p$ such that $\hat{\rho}$, the cost of $s \in G$ is smaller than $2 - p$, the expected cost of $s \in B$.

In this setting, the obvious "optimization goal" is to find a solution $s \in G$. How can we achieve this goal, given two instances generated by the above process? We will demonstrate that our approach does well with regard to this goal. To quantify this performance, we compare our approach with two natural competitors: given two instances $I_1$ and $I_2$, (i) the `Average` method averages the costs of every solution and selects the solution $s \in S$ with the smallest average cost; (ii) the `FirstIntersection` method chooses the smallest $\rho$ for which $A_\rho(I_1)$ and $A_\rho(I_2)$ intersect, and selects a solution from the respective intersection (if the intersection contains more than one solution).

Let $b_1, b_2$ be the number of solutions from $B$ with cost 1 in $I_1$ and $I_2$, respectively, and let $b_x$ be the number of solutions that have cost 1 in both $I_1$ and $I_2$. For appropriately set values of $g$, $b$, and $p$ it follows that our approach selects $s \in G$ with higher probability than `Average` and `FirstIntersection`. Roughly speaking, both `Average` and `FirstIntersection` selects $s \in G$ only if $b_x = 0$, which happens with an extremely small probability $(1 - p)^b$. On the other hand, our approach maximizes the unexpected similarity for $\rho = \hat{\rho}$ with high probability and then selects a solution $s \in G$ with probability at least, but usually much more than, $\frac{g}{g+b}$.

## 3. PROPERTIES OF UNEXPECTED SIMILARITY

In this section we investigate the unexpected similarity. Computing the unexpected similarity is not straightforward. One of the difficulties lies in the computation of the denominator – the expected size of an intersection of two approximation sets. If each set of feasible solutions is a *feasible approximation set*, i.e., $\rho'$-approximation set for some instance $I'$ and some value $\rho'$, we can calculate the expected size of the intersection easily, as the following theorem shows. In the following, we denote by $\mathcal{F}_x$ the set of all feasible approximation sets of size $x$.

THEOREM 1. *Let $P = (\mathcal{I}, S, c)$ be an optimization problem with the property that for any subset $A$ of the set of all feasible solutions $S$ there exists an input instance $I$ and value $\rho'$ such that $A_{\rho'}(I) = A$. Then, the unexpected similarity of two instances $I_1, I_2 \in \mathcal{I}$ at value $\rho$ is*

$$U_\rho = \frac{|S||A_\rho(I_1) \cap A_\rho(I_2)|}{|A_\rho(I_1)||A_\rho(I_2)|}. \tag{4}$$

PROOF. Let $|A_\rho(I_1)| = k$ and $|A_\rho(I_2)| = l$. As every subset $A$ of $S$ is a $\rho'$-approximation set of some instance $I$, we get that $\mathcal{F}_k$ and $\mathcal{F}_l$ are the sets of all subsets of $S$ of size $k$ and $l$, respectively. Therefore, $\mathbb{E}_{A \in \mathcal{F}_k, B \in \mathcal{F}_l} |A \cap B|$ is the expected overlap of two randomly chosen subsets $A$ and $B$ of $S$ of size $k$ and $l$, respectively, which amounts to $l\frac{k}{|S|}$, and the claim follows. □

As an example, the problem of clustering data points [1] satisfies this property.

When not every subset $A \subseteq S$ is a feasible approximation set, the situation is more complicated, and there is no general recipe for how to compute the expected size of the intersection. Still, the right-hand side of Equation (4) is in some cases an upper bound on the unexpected similarity, as we will later argue. Whenever $|A_\rho(I_1)| = |A_\rho(I_2)|$, this expression may be close to the real value of the unexpected similarity.

The following equality will prove to be useful in arguing about the expected size of the intersection of two unrelated feasible approximation sets.

LEMMA 2. *The expected size of the intersection of two approximation sets $A \in \mathcal{F}_k$, $B \in \mathcal{F}_l$ for the uniform probability distribution is*

$$\frac{1}{|\mathcal{F}_k||\mathcal{F}_l|} \sum_{\substack{F_1 \in \mathcal{F}_k \\ F_2 \in \mathcal{F}_l}} |F_1 \cap F_2| =$$

$$\frac{1}{|\mathcal{F}_k||\mathcal{F}_l|} \sum_{s \in S} |\{F \in \mathcal{F}_k | s \in F\}| \cdot |\{F \in \mathcal{F}_l | s \in F\}|.$$

PROOF. It is easy to see that $\sum_{\substack{F_1 \in \mathcal{F}_k \\ F_2 \in \mathcal{F}_l}} |F_1 \cap F_2| = \sum_{s \in S} |\{F \in \mathcal{F}_k | s \in F\}| \cdot |\{F \in \mathcal{F}_l | s \in F\}|$ simply by summing up in two different ways. □

We now show that in some cases the expression in Equation (4) provides an upper bound on $U_\rho$.

THEOREM 3. *Let $P = (\mathcal{I}, S, c)$ be an optimization problem. If $|A_\rho(I_1)| = |A_\rho(I_2)|$ for a given $\rho$, then*

$$U_\rho \leq \frac{|S||A_\rho(I_1) \cap A_\rho(I_2)|}{|A_\rho(I_1)||A_\rho(I_2)|}. \tag{5}$$

PROOF. Define $k := |A_\rho(I_1)| = |A_\rho(I_2)|$. Recall that $\mathcal{F}_k$ is the collection of all feasible approximation sets that consist of $k$ feasible solutions. Substituting $k$ for $|A_\rho(I_1)|$ and $|A_\rho(I_2)|$, we can calculate the expected size of the intersection as:

$$\frac{1}{|\mathcal{F}_k|^2} \sum_{\substack{F_1 \in \mathcal{F}_k \\ F_2 \in \mathcal{F}_k}} |F_1 \cap F_2| \overset{(a)}{=} \frac{1}{|\mathcal{F}_k|^2} \sum_{s \in S} |\{F \in \mathcal{F}_k | s \in F\}|^2$$

$$\overset{(b)}{\geq} \sum_{s \in S} \frac{k^2 |\mathcal{F}_k|^2}{|\mathcal{F}_k|^2 |S|^2} = \frac{k^2}{|S|} = \frac{|A_\rho(I_1)||A_\rho(I_2)|}{|S|} \tag{6}$$

Equality $(a)$ follows from Lemma 2. Inequality $(b)$ is due to the fact that the sum of squares of positive numbers $\sum_i^n a_i^2$ subject to $\sum_i^n a_i = A$ is minimized when for all $i$, $a_i = A/n$. Here, obviously, $\sum_s |\{F \in \mathcal{F}_k | s \in F\}| = k|\mathcal{F}_k|$.

Dividing the expression $|A_\rho(I_1) \cap A_\rho(I_2)|$ by the obtained lower bound on the expected size of an intersection, we obtain the claimed upper bound on the unexpected similarity.

□

From the proof of Theorem 3 we immediately see that Equation (4) holds whenever every feasible solution $s \in S$

is part of the same number of approximation sets $\mathcal{F}_k$. Generalizing this observation, we obtain a lower bound on the unexpected similarity, as follows.

THEOREM 4. *Let $a$ be a constant such that for each feasible solution $s$ of some optimization problem $P = (\mathcal{I}, S, c)$ it holds that $|\{F \in \mathcal{F}_k | s \in F\}| \le ak|\mathcal{F}_k|/|S|$. Then,*

$$U_\rho \ge \frac{|S||A_\rho(I_1) \cap A_\rho(I_2)|}{a|A_\rho(I_1)||A_\rho(I_2)|} \tag{7}$$

PROOF. The proof is similar to the proof of Theorem 3. The sum of contributions of a single feasible solution is maximized when $1/a$ of the feasible solutions are in the maximum number of approximation sets and the rest of the feasible solutions is in no approximation set. □

If instead of the best case situation in the proof of Theorem 3 we consider the worst case, we get a lower bound on $U_\rho$ that is smaller by a factor $|S|$ as compared with the derived upper bound of Theorem 3.

THEOREM 5. *Let $P = (\mathcal{I}, S, c)$ be an optimization problem. Then,*

$$U_\rho \ge \frac{|A_\rho(I_1) \cap A_\rho(I_2)|}{|A_\rho(I_1)||A_\rho(I_2)|} \tag{8}$$

PROOF. We can construct an (artificial) optimization problem where all but one feasible solution can only lie in a single approximation set of some fixed size $k$ and the one last solution is contained in all approximation sets. Then the expected size of the intersection attains the lower bound that results from the most uneven division of feasible solutions into approximation sets. □

The missing factor $|S|$ in the lower bound would for small $|A_\rho(I_1)|$ and $|A_\rho(I_2)|$ often dominate the whole expression and leaving it out makes for an extremely large gap between the upper and the lower bound. This shows that the step of deriving the appropriate specific formula or algorithm to calculate the expected size of the intersection is a necessary component of the approach unless it is possible to show that for a concrete problem, the upper bound is sufficient. One could hope that many typical combinatorial optimization problems have solution spaces which are uniform, i.e., the constant $a$ from Theorem 4 should be low and the upper bound should be reasonably tight.

To the end, we point out an interesting property of our approach. Given two instances $I_1$ and $I_2$, where the (unique) optimum solution of $I_1$ equals the (unique) optimum solution of $I_2$, then assuming the instances are typical, one would pick the common optimum solution as a good robust solution for the optimization problem. This is what our method usually does as well. In the case when every subset $F \subseteq S$ of solutions is a feasible approximation set, it is always the case, because $U_1 \ge U_\rho$ for every $\rho > 1$, as $U_1 = \frac{1}{1 \cdot 1}|S| \ge \frac{\min\{|A_\rho(I_1)|, |A_\rho(I_2)|\}}{|A_\rho(I_1)| \cdot |A_\rho(I_2)|}|S| \ge U_\rho$.

PROPOSITION 6. *Let $I_1$ and $I_2$ be two instances of an uncertain optimization problem such that $s \in S$ is a unique optimum solution in $I_1$ and a unique optimum solution in $I_2$. Then $U_1(I_1, I_2) \ge U_\rho(I_1, I_2)$ for every $\rho \ge 1$. Moreover, our method chooses $s$ as the solution.*

## 4. FINDING A SUB-ARRAY OF MAXIMUM SUM

The classical textbook optimization problem MAXIMUM-SUBARRAYSUM takes an array $(a_1, a_2, \ldots, a_n)$ of $n$ integers and asks for a contiguous sub-array for which the sum of its elements is maximum. A linear time solution of the problem is attributed to Jay Kadane [15]. Despite its general importance, finding the $k$ best sub-arrays has been studied only recently [16, 17, 18]. The optimal algorithm runs in time $O(n + k)$.

### *Uncertain optimization.*

In the uncertain optimization version of this problem, we are given $I_1$ and $I_2$, two integer arrays of size $n$ each. The set of all feasible solutions consists of all the $\frac{n(n+1)}{2} + 1$ sub-arrays, including the empty sub-array whose sum is zero. Given an approximation ratio $\rho$, in order to determine $U_\rho$ we need to compute: (i) the size of the intersection $A_\rho(I_1) \cap A_\rho(I_2)$, and (ii) the expected size of the intersection $A \cap B$ of any two feasible approximation sets $A$ and $B$ of sizes $|A| = |A_\rho(I_1)|$ and $|B| = |A_\rho(I_2)|$.

### *The size of the intersection.*

The size of the intersection $A_\rho(I_1) \cap A_\rho(I_2)$ can be computed as follows. First, note that from a given approximation ratio $\rho$, a threshold $t = \rho \cdot OPT_I$ for the sub-array sum approximation follows immediately from the optimum $OPT_I$ of a given instance $I$. We scan through the array and maintain for scanning position $j$ an auxiliary array $aux_j[i]$ of sub-array sums of elements from position $i$ to $j$. It is not necessary to maintain this explicitly, since at each position in the scan we only need to know how many of the entries of $aux$ are greater than $t$. Therefore, instead of updating all entries in $aux$ when a new position with element $x$ is scanned, we decrease the threshold by $x$. It is then enough to maintain a data structure that supports a query for the number of elements above a threshold, in addition to the insertion of new elements. A balanced search tree with aggregates at inner vertices can service both in time $O(\log n)$. This allows us to compute the size of an approximation set for a given value of $\rho$ in time $O(n \log n)$. The intersection of two approximation sets can be found similarly, by maintaining a 2-dimensional range tree with aggregates at inner vertices, and by querying it with a 2-dimensional threshold at each scan position. Hence, we proved the following theorem.

THEOREM 7. *Let $I_1$ and $I_2$ be two arrays of size $n$. Given $\rho$, the size of the intersection of the $\rho$-approximation sets $A_\rho(I_1)$ and $A_\rho(I_2)$ can be found in time $O(n \log^2 n)$.*

### *The expected size of the intersection.*

Here we face the problem that not every set of sub-arrays is a feasible approximation set, and thus we cannot use Theorem 1. To see this, consider an example with the set of sub-arrays $F = \{(a, c), (b, d)\}$, where $a < b < c < d$ are array positions and $(i, j)$ denotes the sub-array from position $i$ to position $j$. Recall that a sub-array $(i, j)$ is in an approximation set $A_\rho$ if and only if $a_i + \ldots + a_j \ge \rho \cdot \text{opt}$. Thus, $F$ cannot be a feasible approximation set: it would also need to contain one or both of the sub-arrays $(a, d)$ and $(b, c)$.

In the technical report [19] we derive an asymptotic substitute for the exact expected intersection size, given the sizes of the feasible approximation sets, as expressed in the following theorem.

THEOREM 8. *The expected size of an intersection of two randomly picked feasible approximation sets of sizes $k$ and $l$ on two arrays of length $n$ each equals*

$$\frac{4k^2l^2 - 2kl}{2k + 2l - 3} \cdot \frac{1}{n^2} + O(n^{-3}). \quad (9)$$

### *An experimental evaluation with real world data.*

Our problem has a natural interpretation when array elements are price differences, such as day-to-day stock price fluctuations: A maximum sub-array is then a single best interval of buying, keeping and selling a stock (where instead of the price difference, we naturally took the ratio of stock prices, calculated as maximum sub-array sum for logarithms of factors). We thus based our experiments on the historical daily prices of a variety of stock indices (BEL-20, Dow Jones, Hang Seng, Nikkei, AEX, CAC-40, DAX, Eurotop100, FTSE100, JSX, Nasdaq, AS30, RTSIndex, and SMI) from 1999 to 2010, adjusted for inflation [20].

In an attempt to assess how well our approach works in comparison with potential competing approaches, we came up with a number of algorithms that, when provided with two instances, return a (hopefully good) sub-array. We then evaluated the sum of this predicted sub-array on a third instance. The first algorithm, denoted by RANDOM, picks a random sub-array and is used simply to illustrate that the results of following algorithms are not caused by excessive growth of the markets. The less naïve algorithm PICKONE randomly chooses one of the input instances and returns an optimum solution for this instance. Algorithm AVERAGE averages the two instances day by day and returns an optimum sub-array for the averaged instance. Algorithm UPPERBOUND maximizes the upper bound on $U_\rho$ given by Theorem 3 and picks a random solution from the generated intersection. Algorithm FIRSTINTERSECTION finds the smallest $\rho$ that results in a non-empty approximation set intersection and returns a random solution from it (in case there is more than one). Algorithm ASYMPTOTIC is similar to UPPERBOUND, but uses the asymptotic estimate given by Theorem 8 for the expectation in $U_\rho$ to choose the correct approximation ratio. Finally, algorithm OPTIMALOFFLINE returns a sub-array that maximizes the average gain over all instances of the given year – the best one one can possibly get.

We considered each year as a separate class of instances, and we used each pair of input instances as an input to each algorithm. We then evaluated the output of the algorithm on all (different) instances from that year. Table 1(a) shows the average loss (over all years) of each method when compared to the algorithm OPTIMALOFFLINE, where $\rho$ was determined by checking values with a step size of 0.01. Algorithm ASYMPTOTIC performs best here at 87% of the optimal off-line algorithm. In Table 1(b) we used only those pairs of instances that had unexpected similarity higher than 5.0, and we evaluated those on all third instances of the same year. This choice corresponds to the idea that instance pairs of low unexpected similarity have little predictive capability. We chose 5.0 so that roughly 50% of all triples of instances were left after the filtering (6048 out of 12012). Choosing higher threshold would decrease the number of chosen triples and thus the relevancy of our results. The performance of ASYMPTOTIC increased to 96% of the optimum off-line solution. Note that all other algorithms (except RANDOM) benefit from the instance similarity calculated by our method as well. It is, however, that the performance of ASYMPTOTIC increases the most.

| Algorithm | loss vs. opt | % of opt |
|---|---|---|
| RANDOM | 24.14 | 4.4% |
| PICKONE | 4.21 | 83.3% |
| AVERAGE | 4.02 | 84.0% |
| UPPERBOUND | 4.26 | 83.1% |
| FIRSTINTERSECTION | 3.44 | 86.3% |
| ASYMPTOTIC | 3.33 | 86.8% |
| OPTIMALOFFLINE | 0.0 | 100.0% |

(a) Instances with arbitrary unexpected similarity

| Algorithm | loss vs. opt | % of opt |
|---|---|---|
| RANDOM | 27.79 | 4.4% |
| PICKONE | 2.22 | 92.4% |
| AVERAGE | 2.13 | 92.7% |
| UPPERBOUND | 1.40 | 95.2% |
| FIRSTINTERSECTION | 1.31 | 95.5% |
| ASYMPTOTIC | 1.16 | 96.0% |
| OPTIMALOFFLINE | 0.0 | 100.0% |

(b) Instances with unexpected similarity larger than 5.0

Table 1: Comparison of optimization methods

## 5. CONCLUSION

### Further experiments

We have demonstrated on two optimization problems that our proposed optimization methodology performs well – on the available data it performed the best among the natural algorithmic competitors. We have performed few more experiments and reported about them in the technical report [19], where we treat two more problems for our optimization method – the shortest path problem and the minimum spanning tree problem. Obviously, the framework requires more experiments with real data.

### Variations of the framework

Our proposed optimization methodology in the presence of noise should be understood as a framework that is open to variations. In the following, we point out some of the most immediate questions that need to be addressed.

#### *How exactly should we choose approximation sets?*

We should study optimizing the size of the intersection for a variety of choices of approximation sets. It may be useful to pick approximation sets parametrized by their size $k$ instead of quality $\rho$. We chose to optimize $\rho$ because this leads to some pleasant properties. For instance, a feasible solution belongs to the approximation set only because of its quality, regardless of other solutions. But it might be worth giving up pleasant properties for others, and for computational ease. Furthermore, we chose a single value of $\rho$ for both instances. It might be worth to study what happens for two independent approximation factors $\rho_1$ and $\rho_2$, or two independent sizes $k_1$ and $k_2$.

In our normalization, we took all topologically different approximation sets to be equally likely. We might, however, have expectations about the generated instances. For shortest paths in a traffic network, for instance, we might expect edge weights to lie in some fixed range. Expectations like these might be factored into the expected size of the approximation set intersection.

### *Are all solutions in the intersection created equal?*

Our method expects all solutions in the best approximation set intersection to be equally desirable (e.g., equally good for a third, unknown instance). In some cases, it might be useful to choose the solution based on some problem specific criterion.

### *Will more input lead to better results?*

We studied two input instances because they are the minimum number necessary to distinguish information from noise. The extension to multiple instances is not immediately obvious. In general, there are two approaches when we have more than two instances at hand: (i) looking at the intersection of all instances and its expectation, or (ii) looking at all intersections between all pairs, triples, etc. of the instances. While the second approach extracts the maximum amount of information from the instances, it seems it could lead to over-fitting and would be extremely computationally difficult.

### *Can we find efficient algorithmic solutions?*

Even though it appears that our approach leads to high-quality solutions, it does not lead to efficient algorithms easily. Quite the opposite: It generates a family of algorithmic problems, namely to compute the intersection size of approximation sets, and to pick a solution at random from this set. Ideally, we would want to avoid the explicit computation of the set, and get hold of a random solution from it more directly. For some problems, such as the MAXIMUM-SUBARRAYSUM, the intersection can be found efficiently. In many cases, though, the sizes of the approximation sets are exponential. The problem of exact counting is often computationally hard, even if the underlying problems themselves are solvable in polynomial time. A way out may be to resort to approximation algorithms for these counting and enumeration problems.

### *Problem-based similarity beyond optimization*

An interesting side-result that we did not focus on in this paper is the expressiveness of instance similarity $U$. For example, consider the problem of computing a shortest path between vertices $s$ and $t$ in a graph $G$. Having two instances $I_1$ and $I_2$ of this problem, one may attempt to measure the similarity of these instances using the usual statistical tools – e.g., the correlation coefficient. If the instances differ "a lot" only in the edge-weights of very "heavy" edges that are never used in any nearly-shortest path, then the unexpected similarity of these two instances will regard these instances to be very similar, whereas the correlation coefficient will tell the opposite. At the same time, if the computational goal was a maximum matching, the unexpected similarity would regard the two instances as significantly different. This example highlights the need for a measure of *similarity of instances with respect to a computational goal*. We therefore believe that the unexpected similarity may prove to be useful also outside of the proposed framework. For example, Table 1 suggests that inputs that are very similar, measured according to the *unexpected similarity*, yield good results not only for our, but also for other optimization methods.

## And finally

In spite of the need to study all the mentioned variations (and more), we believe that the proposed approach has the potential to provide a solid theoretical foundation of practical value for optimization in the presence of unknown noise.

## 6. ACKNOWLEDGMENTS

This work has been partially supported by the Swiss National Science Foundation (SNF) under the grant number 200021_138117/1, and by the EU FP7/2007-2013, under grant agreement no. 288094 (project eCOMPASS).

## 7. REFERENCES

[1] Joachim M. Buhmann. Information theoretic model validation for clustering. In *IEEE International Symposium on Information Theory (ISIT)*, 2010.

[2] Joachim M. Buhmann. Context sensitive information: Model validation by information theory. In *Proceedings of the Third Mexican Conference on Pattern Recognition (MCPR)*, volume 6718 of *LNCS*, pages 12–21. Springer, 2011.

[3] J.J. Schneider and S. Kirkpatrick. *Stochastic Optimization.* Springer, 2007.

[4] Peter Kall and János Mayer. *Stochastic Linear Programming: Models, Theory, and Computation.* Springer Verlag, 2005.

[5] Y. Ben-Haim. *Info-gap decision theory: decisions under severe uncertainty.* Academic, 2006.

[6] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization.* Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

[7] Christian Liebchen, Marco E. Lübbecke, Rolf H. Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2009.

[8] Yonatan Bilu and Nathan Linial. Are stable instances easy? In Andrew Chi-Chih Yao, editor, *Proceedings of the First Symposium on Innovations in Computer Science (ICS)*, pages 332–341. Tsinghua University Press, 2010.

[9] Davide Bilò, Michael Gatto, Luciano Gualà, Guido Proietti, and Peter Widmayer. Stability of networks in stretchable graphs. In Shay Kutten and Janez Zerovnik, editors, *SIROCCO*, volume 5869 of *Lecture Notes in Computer Science*, pages 100–112. Springer, 2009.

[10] Michael Gatto and Peter Widmayer. On robust online scheduling algorithms. *J. Scheduling*, 14(2):141–156, 2011.

[11] Matúš Mihalák, Marcel Schöngens, Rastislav Šrámek, and Peter Widmayer. On the complexity of the metric TSP under stability considerations. In Ivana Černá, Tibor Gyimóthy, Juraj Hromkovič, Keith G. Jeffery, Rastislav Kráľovič, Marko Vukolic, and Stefan Wolf, editors, *SOFSEM*, volume 6543 of *Lecture Notes in Computer Science*, pages 382–393. Springer, 2011.

[12] Yonatan Bilu, Amit Daniely, Nati Linial, and Michael Saks. On the practically interesting instances of MAXCUT. *CoRR*, abs/1205.4893, 2012.

[13] Amit Daniely, Nati Linial, and Michael Saks. Clustering is difficult only when it does not matter. *CoRR*, abs/1205.4891, 2012.

[14] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[15] Jon Bentley. Programming pearls: algorithm design techniques. *Commun. ACM*, 27:865–873, September 1984.

[16] Fredrik Bengtsson and Jingsen Chen. Efficient algorithms for $k$ maximum sums. *Algorithmica*, 46(1):27–41, 2006.

[17] Sung Eun Bae and Tadao Takaoka. Improved algorithms for the k-maximum subarray problem. *The Computer Journal*, 49(3):358–374, 2006.

[18] Chih-Huai Cheng, Kuan-Yu Chen, Wen-Chin Tien, and Kun-Mao Chao. Improved algorithms for the maximum-sums problems. *Theoretical Computer Science*, 362(1-3):162–170, 2006.

[19] Joachim M. Buhmann, Matúš Mihalák, Rastislav Šrámek, and Peter Widmayer. Optimization in the presence of uncertainty. Technical Report 755, Institute of Theoretical Computer Science, ETH Zurich, 2012. Available online at `http://www.inf.ethz.ch/research/disstechreps/techreports`.

[20] Market rates online. `http://www.marketratesonline.com`.