Computers & Operations Research 55 (2015) 12–22

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

The robust knapsack problem with queries ☆

Marc Goerigk [a,*,1], Manoj Gupta [c], Jonas Ide [b,2], Anita Schöbel [b], Sandeep Sen [c]

[a] Technische Universität Kaiserslautern, Germany
[b] Georg-August Universität Göttingen, Germany
[c] IIT Delhi, India

ARTICLE INFO

Available online 13 October 2014

Keywords:
Robust optimization
Queries
Robust knapsack problem
Competitiveness

ABSTRACT

We consider robust knapsack problems where item weights are uncertain. We are allowed to query an item to find its exact weight,where the number of such queries is bounded by a given parameter Q. After these queries are made, we need to pack the items robustly, i.e., so that the choice of items is feasible for every remaining possible scenario of item weights.

The central question that we consider is: Which items should be queried in order to gain maximum profit? We introduce the notion of *query competitiveness* for strict robustness to evaluate the quality of an algorithm for this problem, and obtain lower and upper bounds on this competitiveness for interval-based uncertainty. Similar to the study of online algorithms, we study the competitiveness under different frameworks, namely we analyze the worst-case query competitiveness for deterministic algorithms, the expected query competitiveness for randomized algorithms and the average case competitiveness for known distributions of the uncertain input data. We derive theoretical bounds for these different frameworks and evaluate them experimentally. We also extend this approach to Γ-restricted uncertainties introduced by Bertsimas and Sim.

Furthermore, we present heuristic algorithms for the problem. In computational experiments considering both the interval-based and the Γ-restricted uncertainty, we evaluate their empirical performance. While the usage of a Γ-restricted uncertainty improves the nominal performance of a solution (as expected), we find that the query competitiveness gets worse.

1. Introduction

The knapsack problem is one of the classic problems in discrete optimization and has been intensely studied in many variants; [1] gives an overview on the topic. In its basic form, we are given a list I of n items with corresponding weights w_i and profits p_i. The problem is to choose a subset $\mathcal{S} \subseteq I$ whose total weight $\sum_{i \in \mathcal{S}} w_i$ is bounded by a given budget B such that the total profit $\sum_{i \in \mathcal{S}} p_i$ is maximized. A formulation as an integer linear program is

$$\max\left\{ \sum_{i \in [n]} p_i x_i : \sum_{i \in [n]} w_i x_i \leq B, x \in \{0,1\}^n \right\},$$

where we use the notation $[n]:=\{1,\ldots,n\}$. In this paper, we focus on the case where the weight parameters w_i are not known exactly. The only knowledge about them is given by an uncertainty set

$$\mathcal{U} = \mathcal{U}_1 \times \cdots \times \mathcal{U}_n = [\underline{w}_1, \overline{w}_1] \times \ldots \times [\underline{w}_n, \overline{w}_n]$$

that contains all possible parameter realizations. The two most common approaches for uncertain optimization problems are *stochastic optimization* (see e.g. [2] for a general introduction, and Chapter 14 of [1] for the stochastic knapsack problem) and *robust optimization* (see e.g. [3, for a recent survey]). The latter is described in the following.

Considering robust knapsack problems, we have to hedge against the worst case; i.e. we have to be sure that the knapsack remains feasible even if for all items the upper bounds of their intervals are realized. This means we have to solve a knapsack problem with $w_i = \overline{w}_i$ for all $i=1,\ldots,n$. Clearly, this is a rather conservative point of view, sometimes referred to as *strict robustness*, as developed in [4–6].

The robust knapsack problem has been intensively studied in the recent literature. In order to overcome the conservatism of strict robustness, [7,8] considered the robust knapsack problem with only a finite uncertainty set affecting the item profits. Refs. [9,10] introduce new types of uncertainty sets for the item weights, where the number of coefficients that may deviate from

☆Partially supported by Grant SCHO 1140/6-3 within the Indo-German DST-DFG Programme.

* Corresponding author. Tel.: +49 631 205 3878.
E-mail address: goerigk@mathematik.uni-kl.de (M. Goerigk).

[1] Partially supported by grant SCHO 1140/3-2 within the DFG programme Algorithm Engineering.
[2] Partially supported by the DFG RTG 1703 "Resource Efficiency in Corporate Networks".

http://dx.doi.org/10.1016/j.cor.2014.09.010
0305-0548/

their expected value is bounded by Γ, and illustrate this new concept on knapsack problems. A dynamic programming algorithm for the robust knapsack problem with Γ-uncertainty has recently been presented in [11]. An extension of the Γ-approach with a variable budgeted uncertainty and an application to the uncertain knapsack problem have been proposed in [12].

Another possibility to overcome the conservatism of strict robustness with interval-based uncertainty is to apply less conservative robustness concepts such as recovery robustness. For knapsack problems this has been done in [13,14].

It is still an open question how to compare robust and non-robust solutions. The best-known approach is to calculate bounds on the so-called price of robustness, which is the ratio of the best robust objective value to the best non-robust objective value. In [15], this price of robustness is studied for the robust knapsack problem with Γ-type uncertainty sets.

As pointed out, the common ways to deal with uncertain information are stochastic and robust optimization. In this paper we follow an alternative approach. We use the concept of *queries* which has been discussed in [16,17]. The basic idea is that one can determine the real value of an uncertain parameter at a certain cost in order to use this real value in the optimization step. In the literature this concept is used, e.g., for shortest path problems where the length of the edges is uncertain [18], computing the value of an objective function with uncertain input [19], minimum spanning tree problems with uncertain edge weights [20], and linear programs with uncertain objective coefficients [21]. In our paper, we use queries to enhance an underlying robustness concept as follows: before packing the knapsack we are allowed to successively query a fixed number $Q < n$ of items where every query reveals the true weight of the queried item. We may utilize this information to decide about which items to pack into the knapsack. As before, the knapsack has to remain feasible even if for all items the upper bounds are realized. The question is, which items should be queried such that the profit is maximized?

Queries can be thought of as an option to use an exact (but expensive) measuring instrument to get precise information about the weights of the items before packing them into the knapsack. Apart from physical measurements, queries can also capture other forms of resource allocation constraints. Assume that different projects (or applications) compete for the same budget. Often the resources needed by these projects are only estimates. For some of them, a more precise estimation of the required resources may be obtained by a closer analysis. Since such a closer analysis is expensive, it can only be done for Q of the projects while for all other projects the upper bounds must be used.

Note that this concept does not only apply to knapsack problems but can in fact be applied to all kinds of combinatorial optimization problems. However, we chose knapsack as a starting point of analysis due to its very intuitive problem structure.

Contributions: We introduce robust knapsack problems with queries for interval-based and Γ-uncertainties in which items are queried in order to obtain more information. In order to evaluate such concepts, we introduce the concept of worst-case, expected and average *query competitiveness* for robust knapsack problems. These are motivated by competitive analysis as used in online optimization. We present upper bounds on the worst-case query competitiveness of the robust knapsack problem for interval-based uncertainties under the above frameworks and give examples showing that these bounds are tight. Our randomized strategy yields a better expected competitive ratio. Assuming that the (unknown) weights follow a known probability distribution, e.g., the uniform distribution in the range $[1, k]$, we analyze the average performance of simple heuristics like randomly querying items. Several other heuristics for the robust knapsack problem are evaluated experimentally by analyzing their empirical performance.

Overview: In Section 2, we give a formal definition of the robust knapsack problem with queries and introduce the concept of query competitiveness. Contrary to the price of robustness, we do not compare the objective value of a robust solution to the objective value of a non-robust solution, but instead to the objective value when using the best possible choice of queries.

We present upper and lower bounds on the deterministic query competitiveness in Section 3 and on the expected query competitiveness in Section 4. In Section 5, we study the average case performance for a uniform distribution of the weights and unit profits and compare our theoretical estimates with experimental results. In Section 6 we extend query competitiveness to Γ-uncertainty sets and discuss some results for this type of uncertainty. We propose heuristics for the case of arbitrary profits, and compare their empirical behavior in Section 7. We conclude the paper in Section 8.

2. Problem definition

The *robust knapsack problem with queries* is given as follows:

Let a list I of n items with profits p_i and weight intervals $\mathcal{U}_i = [\underline{w}_i, \overline{w}_i]$ be given. We may perform $Q < n$ *queries* on the items I. Each query returns the real weight $\hat{w}_i \in \mathcal{U}_i$ of the queried item. The goal is to select Q items to be queried such that the optimal objective value of the resulting deterministic knapsack problem with profits p_i and weights

$$w_i := \begin{cases} \hat{w}_i & \text{if item } i \text{ has been queried} \\ \overline{w}_i & \text{if item } i \text{ has not been queried} \end{cases} \tag{1}$$

is maximized.

Example 1. We are given five items with the following data where the real weights $\hat{w}_i$ are not known:

i	1	2	3	4	5
p_i	3	5	3	4	6
$\mathcal{U}_i$	[3, 7]	[4, 6]	[2, 2]	[1, 7]	[6, 7]

The budget is $B = 10$, and we may do $Q = 2$ queries. We decide to query items 1 and 4. The query of item 1 gives a weight of 6, and the query of item 4 yields a weight of 3. Using this additional information, the best choice is to take items 4 and 5 as a solution. This gives an objective value of 10, with an (exact) weight of 3 for item 4 and (estimated) weight of 7 for item 5.

An algorithm for the robust knapsack problem consists of a *query strategy* in order to select the items to be queried. After having the information of the queries, an optimal solution for the resulting deterministic knapsack problem is determined. In this paper we focus on the effect of our queries. Despite it being NP-hard, we assume that we can solve the resulting knapsack problem exactly throughout this work in order to avoid this effect being mixed with other errors. Solution methods to this end, including an FPTAS, have recently been proposed in [22].

We assume that queries are made successively, i.e., we use an adaptive query strategy where the result of one query may be taken into account for the following queries. Note that for the above definition of uncertainty, this is equivalent to performing all queries simultaneously, as item weights are independent.

How can we compare different query strategies? Similar to online optimization we would like to compare the value of the knapsack obtained through our queries with the worst case which is generated by a (malicious) adversary. In order to keep the adversary from assigning a weight of $\underline{w}_i$ to all non-queried items (and pack as many of them as possible into the knapsack) we pose the following