Computers & Operations Research 38 (2011) 505–513

Contents lists available at ScienceDirect

Computers & Operations Research

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

Heuristic algorithms for the general nonlinear separable knapsack problem

Claudia D'Ambrosio, Silvano Martello *

DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

ARTICLE INFO

Available online 3 August 2010

Keywords:
Nonlinear knapsack
Nonconvexity
Separable knapsack
Heuristic
Local search
Mixed integer nonlinear programming

ABSTRACT

We consider the nonlinear knapsack problem with separable nonconvex functions. Depending on the assumption on the integrality of the variables, this problem can be modeled as a nonlinear programming or as a (mixed) integer nonlinear programming problem. In both cases, this class of problems is very difficult to solve, both from a theoretical and a practical viewpoint. We propose a fast heuristic algorithm, and a local search post-optimization procedure. A series of computational comparisons with a heuristic method for general nonconvex mixed integer nonlinear programming and with global optimization methods shows that the proposed algorithms provide high-quality solutions within very short computing times.

1. Introduction

The general nonlinear knapsack problem is

$$\max f(x) \tag{1}$$

$$g(x) \le c \tag{2}$$

$$x \in D \tag{3}$$

where $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, $f(x)$ and $g(x)$ are continuous differentiable functions, c is a nonnegative value, and $D \subseteq \mathbb{R}^n$. In this paper we consider the case where both $f(x)$ and $g(x)$ are separable functions, and D includes bounds and integrality requirements on (part of) the variables. The resulting problem can thus be described, using the knapsack terminology, as follows. Given n items j, each having a *profit* function $f_j(x_j)$ and a *weight* function $g_j(x_j)$, associated with n variables x_j limited by upper bounds u_j ($j \in N = \{1, \ldots, n\}$), determine nonnegative x_j values such that the total weight does not exceed a given *capacity* c and the total produced profit is a maximum. A subset $\overline{N}$ of the x_j variables can be restricted to take integer values. Formally, we consider the *nonlinear knapsack* (NLK) *problem*

$$\max \sum_{j \in N} f_j(x_j) \tag{4}$$

$$\sum_{j \in N} g_j(x_j) \le c \tag{5}$$

$$0 \le x_j \le u_j \quad \forall j \in N = \{1, \ldots, n\} \tag{6}$$

$$x_j \text{ integer} \quad \forall j \in \overline{N} \subseteq N \tag{7}$$

where, as it is common in the knapsack literature, $f_j(x_j)$ and $g_j(x_j)$ are nonlinear nonnegative nondecreasing functions and the unique constraint is the capacity constraint (5). We have no further assumption on functions $f_j(x_j)$ and $g_j(x_j)$, i.e., they can be nonconvex and nonconcave. Cases in which lower bounds l_j (different from zero) on the variable values are also given can be handled by replacing, for $j \in N$, x_j with $y_j + l_j$, so (6) takes the form $0 \le y_j \le u_j - l_j$.

These problems classically arise, e.g., in resource-allocation contexts, where a limited resource of amount c (such as an advertising budget) has to be partitioned for different categories $j \in N$ (such as advertisements for different products). The objective function is the overall expected return, in terms of sales, from all categories. The nonconvexity arises because the return sharply increases with the amount of allocation when the advertisements are noticed by an increasing number of consumers which are more and more motivated to buy it. On the other hand, at some point, saturation occurs, and an increase of advertisement no longer leads to a significant increase in sales. The situation is depicted in Fig. 1. If x_j denotes the amount of advertisement allocated to category j ($j \in N$), $f_j(x_j)$ the expected return in terms of sales of category j, and $g_j(x_j) = x_j$ the corresponding cost for advertising, the resulting optimization problem is then modeled by NLK.

The well-known (linear) knapsack problem (see [10,9]) is the special case of NLK arising when $f_j(x_j)$ and $g_j(x_j)$ are linear integer functions, i.e., $f_j(x_j) = p_j x_j$ and $g_j(x_j) = w_j x_j$ $\forall j \in N$ and $\overline{N} = N$. It follows that NLK is $\mathcal{NP}$−hard.

Nonlinear knapsack problems have many applications in various fields such as, e.g., portfolio selection, stratified sampling, production planning, resource distribution. We refer the reader to the book by Ibaraki and Katoh [6] and to the survey by Bretthauer and Shetty [2] for extensive reviews.

* Corresponding author.
E-mail addresses: c.dambrosio@unibo.it (C. D'Ambrosio), silvano.martello@unibo.it (S. Martello).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.07.010

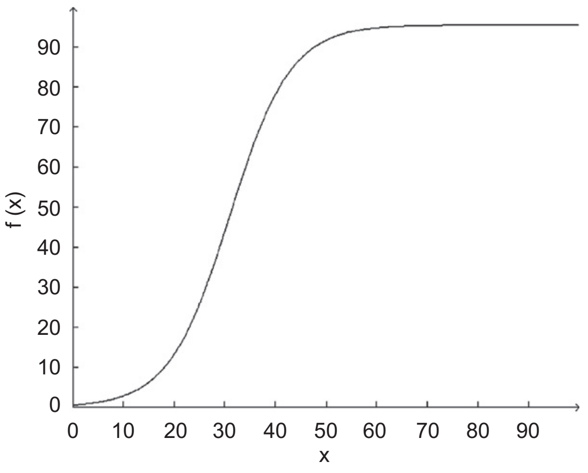

Fig. 1. Example of profit function.

A number of nonlinear separable knapsack problems has been considered in the literature. In most cases the proposed algorithms are specifically tailored to the case of convex or concave functions (see, e.g., the recent article by Zhang and Hua [12], who consider a minimization case in which both the $f_j(x_j)$ and the $g_j(x_j)$ functions are convex). In our model neither convexity nor concavity is assumed, a situation rarely treated in the literature (apart from contributions on the general global optimization, see Horst and Tuy [5]).

A special case, in which $\overline{N}=N$ and the x_j variables can only take a limited set of prefixed feasible values, was considered by Ohtagaki et al. [11], who developed dominance criteria aimed at reducing the number of feasible values, and a specifically tailored greedy algorithm that recursively allocates capacity units to nondominated items and updates the set of dominated items. Another special case, in which $\overline{N}=N$, the $f_j(x_j)$ functions are piecewise linear and the $g_j(x_j)$ functions are linear, was recently considered by Kameshwaran and Narahari [8], who developed polynomial and pseudo-polynomial time approximation algorithms.

In Section 2 we present a constructive heuristic algorithm for NLK based on a discretization of the solution space. In Section 3 we describe post-processing improvement heuristics based on local search procedures that operate on pairs of variables. The resulting approach is experimentally compared with open source nonlinear programming solvers [1,7] which provide heuristic solutions to nonconvex problems. The quality of the solutions is experimentally evaluated by comparison with the upper bounds produced by general global optimization solvers (Couenne [3] and SC-MINLP, developed by D'Ambrosio et al. [4]). The computational results, reported in Section 4, show that the proposed approach provides high-quality solutions within very small CPU times comparing to the other solvers used for the comparisons, even for large-size instances of the problem. Conclusions follow in Section 5.

2. Constructive heuristic

Before giving a detailed statement of the heuristic, we describe its main steps. The algorithm starts by computing profit and weight values over a discretized solution space. Let $\overline{s}$ denote the number of samplings (identical for all functions), so $\delta_j = u_j/\overline{s}$ is the *sampling step*, and define

$$f_{jk}=f_j(k\delta_j) \quad \text{and} \quad g_{jk}=g_j(k\delta_j) \;\; (j\in N; k=1,\dots,\overline{s}). \tag{8}$$

We then obtain the profit-to-weight ratios as

$$r_{jk}=\frac{f_{jk}}{g_{jk}} \quad (j\in N; k=1,\dots,\overline{s}) \tag{9}$$

and we can compute, for each item j, the maximum ratio

$$r_{j,m(j)}=\max_{k=1,\dots,\overline{s}}\{r_{jk}\}. \tag{10}$$

Assume by simplicity that the items are sorted according to nonincreasing $r_{j,m(j)}$ values, so, with respect to the current sampling step, $r_{1,m(1)}$ is the best available ratio and $m(1)\delta_1$ is the value of variable x_1 that produces the best filling of the first $g_{1,m(1)}$ capacity units. Consider now the second best item 2, and its best ratio $r_{2,m(2)}$ $(\le r_{1,m(1)})$, and observe that item 1 could have a better ratio than item 2 also for a higher x_1 value. Specifically, we can find the highest $m'(1)$ value (with $m'(1)\in\{m(1),\dots,\overline{s}\}$) such that $r_{1,m'(1)}\ge r_{2,m(2)}$.

The idea is then to define $x_1=m'(1)\delta_1$, i.e., to use item 1 for the first $g_{1,m'(1)}$ capacity units, then to continue with the residual capacity $\overline{c}=c-g_{1,m'(1)}$. At the second iteration, item 2 (second best available ratio) is used for the next $g_{2,m'(2)}$ capacity units, where $m'(2)$ is analogously defined as the highest value such that $r_{2,m'(2)}\ge r_{3,m(3)}$, and so on.

The method is further improved by refining the search for the best value $m'(j)$ of the current x_j as follows. Once $m'(j)$ has been computed, the sampling step is decreased, say, to $\overline{\delta}_j=\delta_j/\overline{\sigma}$ (where $\overline{\sigma}$ is a prefixed refinement parameter), and new profit-to-weight ratios of item j are computed, for $x_j=m'(j)\delta_j+\overline{\delta}_j\sigma$ $(\sigma=1,2,\dots,\overline{\sigma})$, and compared with $r_{j+1,m(j+1)}$, to obtain a more precise x_j value. Such refining process can be iterated by further decreasing the sampling step.

Note that, whenever an x_j is defined, the ratios $r_{l,m(l)}$ of the unscanned items need to be updated, if the resulting residual capacity is insufficient for their best ratio, i.e., if $g_{l,m(l)}>\overline{c}$. In such a case the first and second best available ratios can change. In addition, by assuming that the current best ratios correspond to items j and $j+1$, when looking for the $m'(j)$ value, the residual capacity has to be taken into account, i.e., $m'(j)$ must be defined as the highest value such that $r_{j,m'(j)}\ge r_{j+1,m(j+1)}$ and $g_{j,m'(j)}\le\overline{c}$.

The algorithm performs a final step when one of the two cases occurs:

1. all items but one have been assigned an x_j value: in this case all the residual capacity is used for the last item, i.e., we find the largest feasible x_n such that $g_n(x_n)$ does not exceed the residual capacity;
2. the residual capacity is insufficient for allocating the minimum sampled weight of any unscanned item, i.e., $\overline{c}<g_{j1}$ for each unscanned j: in this case the best between two options is adopted, namely (let j be the last scanned item):
   (a) the whole residual capacity is used for the first unscanned item, i.e., we find the largest feasible x_{j+1} such that $g_{j+1}(x_{j+1})$ does not exceed the residual capacity;
   (b) x_j is increased as much as possible, i.e., we find the largest feasible x_j such that $g_j(x_j)$ does not exceed the residual capacity. If some capacity still remains, this is used for the first unscanned item, i.e., we find the largest feasible x_{j+1} such that $g_{j+1}(x_{j+1})$ does not exceed the new residual capacity.

In all cases above, some residual capacity can remain, if the updating is limited by the upper bounds. In this case a simple greedy search scans the items of N in any order, and, for each item j, increases the value of x_j as much as possible.

Note that, to satisfy the integrality requirements for items in $\overline{N}$, it is sufficient to only consider integer valued sampling steps δ_j for all $j\in\overline{N}$.

The detailed statement of the method is given in Algorithm 1. As already observed, whenever a new x_j is defined, the ratios $r_{l,m(l)}$ of the unscanned items need in general to be updated and re-sorted. For this reason, in order to achieve a better efficiency, the items are not actually sorted at each iteration, but only the two