An Algorithm for the Penalized Multiple Choice Knapsack Problem

Elizabeth M. Hilliard\textsuperscript{1} and Amy Greenwald\textsuperscript{2} and Victor Naroditskiy\textsuperscript{3}

Abstract.

We present an algorithm for the penalized multiple choice knapsack problem (PMCKP), a combination of the more common penalized knapsack problem (PKP) and multiple choice knapsack problem (MCKP). Our approach is to convert a PMCKP into a PKP using a previously known transformation between MCKP and KP, and then solve the PKP greedily. For PMCKPs with well-behaved penalty functions, our algorithm is optimal for the linear relaxation of the problem.

1 Introduction

The knapsack problem (KP) is a classic optimization problem. Due to the large number of real-world problems that can be modeled as KPs, the problem comes in many flavors. We focus on a problem variation that combines two previously studied variations: the penalized knapsack (PKP) and the multiple choice knapsack (MCKP) problem.

2 Knapsack Problems, Global Penalty Functions, and Greedy Algorithms

We begin by presenting various knapsack problems, together with greedy algorithms that solve their linear relaxations optimally.

In addition to a problem instance defined by a set of items, each with a weight and value, and a total capacity, our algorithms take as input a metric \( m \), that describes how to evaluate items (e.g., efficiency), a stopping rule \( f \), that indicates when the algorithm should stop taking items and an item-taking rule \( g \), which determines the fraction of the last item considered to take.

Knapsack Problem A KP is defined by a vector of item values, \( v \geq 0 \), a vector of weights, \( w \geq 0 \) and a hard total capacity, \( c \). A solution is a vector \( x \) indicating the amount of each item taken. Thus, the objective is \( \max \_{x} v \cdot x \), subject to \( w \cdot x \leq c \) and each \( x_i \in \{0, 1\} \) (for the discrete problem, in which items are indivisible) or each \( x_i \in [0, 1] \) (for the relaxed problem, \( R(KP) \), in which items may be divisible). (The index \( i \) ranges over items.)

GreedyKP (Alg. 1) takes items in order of efficiency until the knapsack reaches capacity, or there are no more items with positive efficiencies.

Theorem [4]: GreedyKP, with efficiency as the metric \( m \), the hard capacity stopping rule (Alg. 2) as \( f \), and the soft taking rule (Alg. 3) as \( g \), solves \( R(KP) \) optimally.

Algorithm 1 GreedyKP

Input: \( v, w, c, m, f, g \)
Output: \( x \)
\[
\begin{align*}
    x &= 0 \\
    &\text{for all items } i, \text{ CALCMETRIC}(i, v, w, m) \\
    i &= \text{BESTUNTAKENITEMINDEX} \\
    &\text{while } !f(i, x, v, w, c) \text{ and MOREITEMSTOCONSIDER do} \\
    x_i &= 1 \\
    i &= \text{BESTUNTAKENITEMINDEX} \\
    x_i &= g(i, x, v, w, c) \\
    &\text{return } x
\end{align*}
\]

Algorithm 2 HardStoppingRule

Input: \( i, x, v, w, c \)
Output: \{boolean indicating whether to stop taking items or not\}
\[
\begin{align*}
    &\text{return } x \cdot w + w_i > c
\end{align*}
\]

Multiple Choice Knapsack Problem An MCKP is defined similarly to a KP, with an additional constraint over a set of types \( T \), which ensures that only one item \( s \) is taken from each type set, \( T \subset T \). Thus the objective is \( \max _{x} v \cdot x \), subject to \( w \cdot x \leq c \), \( \sum _{s \in T} x_s \leq 1 \), \( \forall T \in T \), and each \( x_i \in \{0, 1\} \), with \( R(MCKP) \) defined analogously.

Theorem [6]: GreedyMCKP, with efficiency as the metric \( m \), the hard capacity stopping rule as \( f \), the soft taking rule as \( g \), solves \( R(MCKP) \) optimally.

[6]’s algorithm (Alg. 5) proceeds in three steps: first it transforms the given instance of MCKP into a KP (Alg. 4); second it solves the ensuing KP optimally using GreedyKP; third it maps the resulting KP solution back into an optimal solution to the MCKP (Alg. 6).

Penalized Knapsack Problems A PKP is defined by \( v \) and \( w \), and a penalty function \( \kappa \). The objective is \( \max _{x} \pi(x) \), where \( \pi(x) \equiv v \cdot x - p(x, v, w) \) subject to each \( x_i \in \{0, 1\} \), with \( R(KP) \) defined analogously. We refer to the penalty functions studied in [2] as \textit{global} because their only input is the knapsack’s total weight \( \kappa \equiv w \cdot x \). In such cases, it suffices to search greedily with efficiency as the metric; however, for non-global penalty functions, other metrics may be more sensible.

Theorem [2]: If the penalty function is global and convex, then the GreedyPKP algorithm, which invokes GreedyKP with efficiency

Algorithm 3 SoftTakingRule

Input: \( i, x, v, c \)
Output: \{fraction of item \( i \) to take\}
\[
\begin{align*}
    &\text{return } (c - x \\
\end{align*}
\]
1. Let \( \max v \cdot x \) denote the total value of the penalty function \( p \) on \( x \). Output: \( \text{decreasing, then} \)

Input: \( v, w, T \)

Algorithm 4 REDUCEMCKPTOKP

Input: \( v, w, T \)
Output: \( v, w \)

SortByWeight(v, w) {Reindex vectors}
for \( T \in T \) do
\((v[T], w[T]) = \text{REMOVELPDOMINATED ITEMS}(v, w)\)
for \( i \in [1, |T| - 1] \) do
\((v[x_i], w[x_i]) = ((v[x_i] + 1) - (v[x_i]), (w[x_i] + 1) - (w[x_i]))\)
return \( v, w \)

Algorithm 5 GREEDYMCKP

Input: \( v, w, T, c, m, f, g \)
Output: \( x \)

\((v', w') = \text{REDUCEMCKPTOKP}(v, w, T)\)
\( x' = \text{GREEDYKPSOL}(v', w', c, m, f, g)\)

return \( \text{CONVERTKPSOLTO MCKPSOL}(x', v', w', v, w) \)

as the metric \( m \), the penalized stopping rule (Alg. 7) as \( f \), and the penalized taking rule (Alg. 8) as \( g \), solves (PKP) optimally.

Penalized Multiple Choice Knapsack Problem A PMCKP is defined by \( v, w, T, \) and a penalty function, \( p \). The objective is \( \max_x \pi(x) \) subject to \( \sum_{x \in T} x_s \leq 1, \forall T \in T \) and each \( x_i \in \{0, 1\} \), with R(PMCKP) defined analogously.

Theorem 1. If the penalty function is global, monotonic, non-increasing, and convex, then the GreedyPMCKP algorithm, which invokes GreedyMCKP with efficiency as the metric \( m \), the penalized stopping rule as \( f \), and the penalized taking rule as \( g \), solves R(PMCKP) optimally.

Lemma 1. Let \( x^* \) denote an optimal solution to R(PMCKP) with penalty function \( p \), and let \( \kappa^* \) and \( \pi^* \) denote the total weight and total value of \( x^* \), respectively. If \( p \) is global, monotonic, and non-decreasing, then \( x^* \) is also an optimal solution to the corresponding R(MCKP) with capacity \( \kappa^* \). Furthermore, \( v \cdot x^* = \pi^* + p(\kappa^*) \).

Proof. Suppose not: i.e., suppose \( x^* \) is not an optimal solution to the corresponding R(MCKP) with capacity \( \kappa^* \). Instead, suppose \( x \) is optimal, with total weight \( \kappa \) and total value \( \pi \). Then \( v \cdot x > v \cdot x^* \) and \( \kappa \leq \kappa^* \). Now, because the penalty function is global, monotonic, and non-decreasing, \( p(\kappa) \leq p(\kappa^*) \). But then \( \pi = v \cdot x - p(\kappa) \geq v \cdot x - p(\kappa^*) > v \cdot x^* - p(\kappa^*) = \pi^* \). But this is a contradiction, since \( x^* \) is optimal.

Proof of Theorem 1. The proof relies on two observations:

1. Let \( x \) denote an optimal solution to R(MCKP), and let \( \kappa^* \) and \( \pi^* \) denote the total weight and total value of \( x \). \( x \) is an optimal solution to R(MCKP) with capacity \( \kappa^* \).

Algorithm 6 CONVERTKPSOLTO MCKPSOL

Input: \( x', v', w', v, w \)
Output: \( x \)
\( x = 0 \)
for \( T \in T \) do
\((v'[T], w'[T]) = (v'[T], x'[T])\)
\( v'[T] = \text{CONVERTKPSOLTO MCKPSOL}(x', v', w', v, w) \)
return \( x \)

Algorithm 7 PENALIZEDSTOPPINGRULE

Input: \( i, x, v, w, p \)
Output: \( \text{\{boolean indicating whether or not to stop taking items\}} \)

\( x' = x + e^i \) \( \{e^i \text{ is a vector of 0s, except the i\text{th entry is a 1}\} \}

return \( \pi(x') < \pi(x) \)

Algorithm 8 PENALIZEDTAKINGRULE

Input: \( i, x, v, w \)
Output: \( \text{\{fraction of} \ x_i \text{\ to take\} \}

\( \text{return} \ \arg \max v \cdot (x + oe^i) - p(x + oe^i, v, w) \)

solution to R(MCKP) with capacity \( \kappa^* \). The value of the optimal solution to R(MCKP) is \( \pi^* + p(\kappa^*) \). (Lemma 1.)

2. Let \( y \) denote a feasible solution to R(KP), and let \( \kappa^y \) and \( \pi^y \) denote the total weight and total value of \( y \). \( y \) is a feasible solution to R(KP) with total value \( \pi^y - p(\kappa^y) \).

Consider an instance of R(PMCKP). Let \( x \) denote an optimal solution to this problem, with total weight \( \kappa^* \) and total value \( \pi^* \). Consider, as well, a corresponding instance of R(KP) constructed via Zemel’s transformation. Let \( y \) denote an optimal solution to this problem, with total value \( \pi^y \).

We claim that \( \pi^y \geq \pi^* \). Suppose not: i.e., suppose \( \pi^y < \pi^* \). By Fact 1, \( x \) is an optimal solution to R(MCKP) with total weight \( \kappa^* \) and total value \( \pi^* + p(\kappa^*) \). By Zemel’s theorem, \( x \) can be converted into a solution, \( y' \), to R(KP) with capacity \( \kappa^* \), such that the total value of \( y' \) is \( \pi^* + p(\kappa^*) \). Finally, by Fact 2, \( y' \) is also a feasible solution to R(KP) with total value \( \pi^y \). This is a contradiction, because \( y \) was assumed to be an optimal solution to R(KP).

3 Conclusions and Future Work

PMCKP was originally proposed as a model of bidding in ad auctions [1]—specifically, in the context of the annual Trading Agent Competition [3]. Indeed, one of the top-scoring TAC AA agents [5] solved PMCKP using GreedyPMCKP as a subroutine inside a search over capacities, but as the space of possible capacities is enormous, it is conceivable that GreedyPMCKP or a variant could perform better. In future work, we plan to investigate the performance of GreedyPMCKP in an ad auctions context.

4 Acknowledgments

This work was supported by NSF Grant #RI-1217761.

REFERENCES


