A practical efficient fptas for the 0-1 multi-objective knapsack problem

Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten

LAMSADE, Université Paris Dauphine, Place Du Maréchal De Lattre de Tassigny, 75 775 Paris Cedex 16, France. {bazgan,hugot,vdp}@lamsade.dauphine.fr

Abstract. In the present work, we are interested in the practical behavior of a new fptas to solve the approximation version of the 0-1 multi-objective knapsack problem. Nevertheless, our methodology focuses on very general techniques (such as dominance relations in dynamic programming) and thus may be applicable in the implementation of fptas for other problems as well. Extensive numerical experiments on various types of instances establish that our method performs very well both in terms of CPU time and size of solved instances. We point out some reasons for the good practical performance of our algorithm. A comparison with an exact method is also performed.

Keywords: multi-objective knapsack problem, approximation, dynamic programming, dominance relations, combinatorial optimization.

1 Introduction

In multi-objective combinatorial optimization, a major challenge is to generate either the set of efficient solutions, that have the property that no improvement on any objective is possible without sacrificing on at least another objective, or the set of non-dominated criterion vectors corresponding to their image in the criterion space. The reader can refer to [1] about multi-objective combinatorial optimization. However, even for moderately-sized problems, it is usually computationally prohibitive to identify the efficient set for two major reasons. First, the number of efficient solutions can be very large. This occurs notably when solving *intractable* instances of combinatorial multi-objective problems, for which the number of efficient solutions is not polynomial in the size of these instances (see, e.g., [1] about the intractability of multi-objective problems). Second, for most multi-objective problems, deciding whether a given solution is dominated is NP-hard, even if the underlying single-objective problem can be solved in a polynomial time.

To handle these two difficulties, researchers have been interested in developing approximation algorithms with provable guarantee such as fully polynomial approximation schemes (fptas). Indeed, an fptas guarantees to compute, for a given accuracy $\varepsilon > 0$, in a running time that is polynomial both in the size of the input and in $1/\varepsilon$, an $(1+\varepsilon)$-approximation, that is a subset of solutions such that, for each efficient solution, this subset contains a solution that is at most

at a factor $(1+\varepsilon)$ on all objective values. This is made possible since it has been pointed out in [2], that, under certain general assumptions, there always exists an $(1+\varepsilon)$-approximation, with any given accuracy $\varepsilon > 0$, whose size is polynomial both in the size of the instance and in $1/\varepsilon$. Thus using an fptas for solving a multi-objective problem has two main advantages: on the one hand it provides us with an efficient algorithm to compute an approximation with a guaranteed accuracy and on the other hand it computes an approximation of reasonable size. Nevertheless, in this stream, researchers are usually motivated by the theoretical question of proving or disproving the existence of an fptas for a given problem. Thus, practical implementations of fptas are cruelly lacking and most of the schemes proposed in the literature are not effective in practice.

We consider in this paper the 0-1 multi-objective knapsack problem which has been shown to admit an fptas in [3,4,5]. Our perspective, however, is to propose another fptas focusing on its practical behavior. The main idea of our approach, based on dynamic programming, relies on the use of several complementary dominance relations to discard partial solutions. In a previous work [6], such techniques have been proved to be extremely effective to solve the exact version of this problem. Extensive numerical experiments on various types of instances are reported and establish that our method performs very well both in terms of CPU time and size of solved instances (up to 20000 items in less than 1 hour in the bi-objective case). We compare our approach with the exact method of [6], which is the most effective exact method currently known. In our experiments, we point out some reasons for the good practical performance of our algorithm that may be applicable to other fptas. Indeed, since our methodology focuses on very general techniques (such as dominance relations in dynamic programming), it may be applicable in the implementation of fptas for other problems as well.

This paper is organized as follows. In section 2, we review basic concepts about multi-objective optimization and approximation, and formally define the 0-1 multi-objective knapsack problem. Section 3 presents the dynamic programming approach using dominance relations. Section 4 is devoted to the presentation of the dominance relations. Computational experiments and results are described in section 5. Conclusions are provided in a final section.

2 Preliminaries

We first recall that, given $\succsim$ a binary relation defined on a finite set A, $B \subseteq A$ is a *covering (or dominating) set* of A with respect to $\succsim$ if and only if for all $a \in A \backslash B$ there exists $b \in B$ such that $b \succsim a$, and $B \subseteq A$ is an *independent (or stable) set* with respect to $\succsim$ if and only if for all $b, b' \in B,\ b \neq b',\ \text{not}(b \succsim b')$.

2.1 Multi-objective optimization and approximation

Consider a multi-objective optimization problem with p criteria or objectives where X denotes the finite set of feasible solutions. Each solution $x \in X$ is

represented in the criterion space by its corresponding criterion vector $f(x) = (f_1(x), \ldots, f_p(x))$. We assume that each criterion has to be maximized.

From these p criteria, the dominance relation defined on X, denoted by $\underline{\Delta}$, states that a feasible solution x dominates a feasible solution x', $x \underline{\Delta} x'$, if and only if $f_i(x) \geq f_i(x')$ for $i = 1, \ldots, p$. We denote by Δ the asymmetric part of $\underline{\Delta}$. A solution x is *efficient* if and only if there is no other feasible solution $x' \in X$ such that $x' \Delta\ x$, and its corresponding criterion vector is said to be *non-dominated*. The set of non-dominated criterion vectors is denoted by ND. A set of efficient solutions is said to be *reduced* if it contains only one solution corresponding to each non-dominated criterion vector. Observe that $X' \subseteq X$ is a reduced efficient set if and only if it is a covering and independent set with respect to $\underline{\Delta}$.

For any constant $\varepsilon \geq 0$, the relation $\underline{\Delta}_\varepsilon$, called ε-dominance, defined on X, states that for all $x, x' \in X$, $x \underline{\Delta}_\varepsilon x'$ if and only if $f_i(x)(1+\varepsilon) \geq f_i(x')$ for $i = 1, \ldots, p$. For any constant $\varepsilon \geq 0$, an $(1+\varepsilon)$-*approximation* is a covering set of X with respect to $\underline{\Delta}_\varepsilon$. Any $(1+\varepsilon)$-approximation which does not contain solutions that dominate each other, *i.e.* which is independent with respect to $\underline{\Delta}$, is *a reduced* $(1+\varepsilon)$-*approximation*. In the following, for a given reduced $(1+\varepsilon)$-approximation, ND_ε denotes the image in the criterion space of this reduced $(1+\varepsilon)$-approximation.

2.2 The 0-1 multi-objective knapsack problem

An instance of the 0-1 multi-objective knapsack problem consists of an integer capacity $W > 0$ and n items. Each item k has a positive integer weight w^k and p non negative integer profits $v_1^k, \ldots, v_p^k$ $(k = 1, \ldots, n)$. A feasible solution is represented by a vector $x = (x_1, \ldots, x_n)$ of binary decision variables x_k, such that $x_k = 1$ if item k is included in the solution and 0 otherwise, which satisfies the capacity constraint $\sum_{k=1}^{n} w^k x_k \leq W$. The value of a feasible solution $x \in X$ on the ith objective is $f_i(x) = \sum_{k=1}^{n} v_i^k x_k$ $(i = 1, \ldots, p)$. For any instance of this problem, we consider two versions: *the exact version* which aims at determining a reduced efficient set, and *the approximation version* which aims at determining a reduced $(1+\varepsilon)$-approximation.

3 Dynamic Programming for the approximation version

We first describe the sequential process used in Dynamic Programming (DP) and introduce some basic concepts of DP (section 3.1). Then, we present the concept of dominance relations for solving the approximation version by a DP approach (section 3.2).

3.1 Sequential process and basic concepts of DP

The sequential process used in DP consists of n phases. At any phase k we generate the set of states S^k which represents all the feasible solutions made

up of items belonging exclusively to the k first items ($k = 1, \ldots, n$). A state $s^k = (s_1^k, \ldots, s_p^k, s_{p+1}^k) \in S^k$ represents a feasible solution of value s_i^k on the ith objective ($i = 1, \ldots, p$) and of weight s_{p+1}^k. Thus, we have $S^k = S^{k-1} \cup \{(s_1^{k-1} + v_1^k, \ldots, s_p^{k-1} + v_p^k, s_{p+1}^{k-1} + w^k) : s_{p+1}^{k-1} + w^k \leq W, s^{k-1} \in S^{k-1}\}$ for $k = 1, \ldots, n$ where the initial set of states S^0 contains only the state $s^0 = (0, \ldots, 0)$ corresponding to the empty knapsack. In the following, we identify a state and a corresponding feasible solution. Thus, relations defined on X are also valid on S^k, and we have $s^k \underline{\Delta} \tilde{s}^k$ if and only if $s_i^k \geq \tilde{s}_i^k$, $i = 1, \ldots, p$ and $s^k \underline{\Delta}_\varepsilon \tilde{s}^k$ if and only if $s_i^k(1 + \varepsilon) \geq \tilde{s}_i^k$, $i = 1, \ldots, p$

Definition 1 (Completion, extension, restriction). *For any state $s^k \in S^k$ ($k \leq n$), a* completion *of s^k is any, possibly empty, subset $J \subseteq \{k+1, \ldots, n\}$ such that $s_{p+1}^k + \sum_{j \in J} w^j \leq W$. We assume that any state $s^n \in S^n$ admits the empty set as unique completion. A state $s^n \in S^n$ is an* extension *of $s^k \in S^k$ ($k \leq n$) if and only if there exists a completion J of s^k such that $s_i^n = s_i^k + \sum_{j \in J} v_i^j$ for $i = 1, \ldots, p$ and $s_{p+1}^n = s_{p+1}^k + \sum_{j \in J} w^j$. The set of extensions of s^k is denoted by Ext(s^k) ($k \leq n$). Finally, $s^k \in S^k$ ($k \leq n$) is a* restriction *at phase k of state $s^n \in S^n$ if and only if s^n is an extension of s^k.*

3.2 Families of dominance relations in Dynamic Programming

The efficiency of DP depends crucially on the possibility of reducing the set of states at each phase. In the context of the approximation version, a family of dominance relations between states for $\underline{\Delta}_\varepsilon$ is used to discard states at any phase. Each dominance relation of this family is specific to a phase. Indeed, we share out the total error ε between the phases by the mean of an error function and associate to each dominance relation of the family a proportion of this error.

Definition 2 (Error function). *The function $e : \{1, \ldots, n\} \to \mathbb{R}$ is an error function if and only if $\sum_{k=1}^{n} e(k) \leq 1$ and $e(k) \geq 0$, $k = 1, \ldots, n$.*

In this way, families of dominance relations between states for $\underline{\Delta}_\varepsilon$ are defined as follows.

Definition 3 (Families of dominance relations between states for $\underline{\Delta}_\varepsilon$). *For any $\varepsilon \geq 0$ and any error function e, a family of relations D^k on S^k, $k = 1, \ldots, n$, is a family of dominance relations for $\underline{\Delta}_\varepsilon$ if for all $s^k, \tilde{s}^k \in S^k$,*

$$s^k D^k \tilde{s}^k \Rightarrow \forall \tilde{s}^n \in Ext(\tilde{s}^k), \exists s^n \in Ext(s^k), s_i^n (1 + \varepsilon)^{e(k)} \geq \tilde{s}_i^n, \quad i = 1, \ldots, p \quad (1)$$

When $\varepsilon = 0$, Definition 3 collapses to the classical definition of dominance relations used in the context of the exact version: a relation D^k on S^k, $k = 1, \ldots, n$, is a dominance relation for $\underline{\Delta}$ if for all $s^k, \tilde{s}^k \in S^k$,

$$s^k D^k \tilde{s}^k \Rightarrow \forall \tilde{s}^n \in \mathrm{Ext}(\tilde{s}^k), \exists s^n \in \mathrm{Ext}(s^k), s^n \underline{\Delta} \tilde{s}^n \quad (2)$$

Even if dominance relations can be non-transitive, in order to be efficient in the implementation, we consider only transitive dominance relations. We introduce now the way of using families of transitive dominance relations for $\underline{\Delta}_\varepsilon$ in

DP approach (see Algorithm 1). At each phase k, Algorithm 1 generates a subset of states $C^k \subseteq S^k$. This is achieved by first creating from C^{k-1} a temporary subset $T^k \subseteq S^k$. Then, we apply transitive dominance relation D^k to each state of T^k in order to check if it is not dominated by any state already in C^k (in which case it is added to C^k) and if it dominates states already in C^k (which are then removed from C^k).

Algorithm 1: Computing a reduced $(1+\varepsilon)$-approximation

```
1  C^0 ← {(0,...,0)};
2  for k ← 1 to n do
3      T^k ← C^{k-1} ∪ {(s_1^{k-1} + v_1^k, ..., s_p^{k-1} + v_p^k, s_{p+1}^{k-1} + w^k) | s_{p+1}^{k-1} + w^k ≤ W, s^{k-1} ∈ C^{k-1}};
       /* Assume that T^k = {s^{k(1)}, ..., s^{k(r)}}                                   */
4      C^k ← {s^{k(1)}};
5      for i ← 2 to r do
           /* Assume that C^k = {s̃^{k(1)}, ..., s̃^{k(ℓ_i)}}                            */
6          dominated ← false ; dominates ← false ; j ← 1;
7          while j ≤ ℓ_i and not(dominated) and not(dominates) do
8              if s̃^{k(j)} D^k s^{k(i)} then dominated ← true
9              else if s^{k(i)} D^k s̃^{k(j)} then C^k ← C^k \ {s̃^{k(j)}} ; dominates ← true;
10             j ← j + 1;
11         if not(dominated) then
12             while j ≤ ℓ_i do
13                 if s^{k(i)} D^k s̃^{k(j)} then C^k ← C^k \ {s̃^{k(j)}};
14                 j ← j + 1;
15             C^k ← C^k ∪ {s^{k(i)}};
16 return C^n;
```

The following results characterize the set C^k obtained at the end of each phase k and establish the validity of Algorithm 1.

Proposition 1. *For any transitive relation D^k on S^k, the set C^k obtained at the end of phase k in Algorithm 1 is a covering and independent set of T^k with respect to D^k $(k = 1, \ldots, n)$.*

Theorem 1. *For any family of transitive dominance relations $D^1, \ldots, D^n$ for $\underline{\Delta}_\varepsilon$, Algorithm 1 returns C^n a covering set of S^n with respect to $\underline{\Delta}_\varepsilon$. Moreover, if $\underline{\Delta} \subseteq D^n$, C^n is a reduced $(1+\varepsilon)$-approximation.*

Algorithm 1 can be significantly simplified by generating states of $T^k = \{s^{k(1)}, \ldots, s^{k(r)}\}$ according to any topological order based on the asymmetric part of D^k. Thus, we have either $s^{k(i)} D^k s^{k(j)}$ or not$(s^{k(j)} D^k s^{k(i)})$ for all $i < j$ $(1 \leq i,j \leq r)$ and step 9 and loop 12-14 can be omitted.

Remark that when $\varepsilon = 0$, we have $\underline{\Delta}_\varepsilon = \underline{\Delta}$, and thus C^n is a covering set of X with respect to $\underline{\Delta}$. Moreover, in this case, if $\underline{\Delta} \subseteq D^n$, C^n corresponds to a reduced efficient set.

4 Dominance relations

We first present the family of dominance relations for $\underline{\Delta}_\varepsilon$ used in our approach that can provide an fptas in certain cases (section 4.1). Then, we present two

complementary dominance relations for $\underline{\Delta}$ (section 4.2) and give a brief explanation of the way of applying them together with the family of dominance relations for $\underline{\Delta}_\varepsilon$ (section 4.3).

4.1 Family of dominance relations for $\underline{\Delta}_\varepsilon$

To solve the exact version of the 0-1 multi-objective knapsack problem, we showed in a previous work [6] that a powerful dominance relation for $\underline{\Delta}$ is the relation $D^k_{\underline{\Delta}}$ that is a generalization to the multi-objective case of the dominance relation usually attributed to Weingartner and Ness [7]. Relation $D^k_{\underline{\Delta}}$ is defined on S^k for $k = 1, \ldots, n$ by:

$$\text{for all } s^k, \tilde{s}^k \in S^k,\ s^k D^k_{\underline{\Delta}} \tilde{s}^k \Leftrightarrow \begin{cases} s^k \underline{\Delta} \tilde{s}^k & \text{and} \\ s^k_{p+1} \leq \tilde{s}^k_{p+1} \text{ if } k < n \end{cases}$$

To solve the approximation version of the 0-1 multi-objective knapsack problem, we generalize relations $D^k_{\underline{\Delta}}$ $(k = 1, \ldots, n)$ to obtain the family of dominance relations $D^k_{\underline{\Delta}_\varepsilon}$ $(k = 1, \ldots, n)$ for $\underline{\Delta}_\varepsilon$ that is based on a partition of the criterion space into hyper-rectangles. For a constant $\varepsilon > 0$ and an error function e, at each phase k we partition each positive criterion range $[1, U_i]$, where U_i is an upper bound on the value of the feasible solutions on the ith criterion $(i = 1, \ldots, p)$, into disjoints intervals of length $(1+\varepsilon)^{e(k)}$ $(k = 1, \ldots, n)$. When $e(k) = 0$, we obtain the following degenerate intervals: $[1,1], [2,2], \ldots, [U_i, U_i]$ $(i = 1, \ldots, p)$. When $e(k) \neq 0$, we obtain the following intervals: $[1; (1+\varepsilon)^{e(k)}[, [(1+\varepsilon)^{e(k)}; (1+\varepsilon)^{2e(k)}[, \ldots, [(1+\varepsilon)^{(\ell^k_i - 1)e(k)}; (1+\varepsilon)^{\ell^k_i e(k)}[$ where $\ell^k_i = \left\lfloor \frac{\log U_i}{e(k)\log(1+\varepsilon)} \right\rfloor + 1$ $(i = 1, \ldots, p)$. In both cases, we add the interval $[0, 0]$. The number of the interval in which belongs the value of a state s^k on the ith criterion $(i = 1, \ldots, p)$ in this partition is:

$$\mathrm{B}_i(s^k, e(k)) = \begin{cases} s^k_i & \text{if } e(k) = 0 \text{ or } s^k_i = 0 \\ \left\lfloor \frac{\log s^k_i}{e(k)\log(1+\varepsilon)} \right\rfloor + 1 & \text{otherwise} \end{cases}$$

From these partitions, we can define for any $\varepsilon > 0$ and any error function e, relations $D^k_{\underline{\Delta}_\varepsilon}$ on S^k for $k = 1, \ldots, n$ by:

$$\text{for all } s^k, \tilde{s}^k \in S^k,\ s^k D^k_{\underline{\Delta}_\varepsilon} \tilde{s}^k \Leftrightarrow \begin{cases} \mathrm{B}_i(s^k, e(k)) \geq \mathrm{B}_i(\tilde{s}^k, e(k)) \quad i = 1, \ldots, p, & \text{and} \\ s^k_{p+1} \leq \tilde{s}^k_{p+1} \text{ if } k < n \end{cases}$$

The following proposition shows that $D^k_{\underline{\Delta}_\varepsilon}$ is indeed a family of dominance relations for $\underline{\Delta}_\varepsilon$ and gives additional properties of $D^k_{\underline{\Delta}_\varepsilon}$.

Proposition 2. *For any $\varepsilon > 0$ and any error function e, we have:*
(a) $D^k_{\underline{\Delta}_\varepsilon}$, $k = 1, \ldots, n$, is a family of dominance relations for $\underline{\Delta}_\varepsilon$,
(b) for any $k \in \{1, \ldots, n\}$, $D^k_{\underline{\Delta}_\varepsilon}$ is transitive,
(c) for any $k \in \{1, \ldots, n\}$, $D^k_{\underline{\Delta}_\varepsilon} \supseteq D^k_{\underline{\Delta}}$ and $D^k_{\underline{\Delta}_\varepsilon} = D^k_{\underline{\Delta}}$ if $e(k) = 0$.

As a consequence of (c) we have $\underline{\Delta} \subseteq D^n_{\underline{\Delta}_\varepsilon}$ and thus Algorithm 1 using the family of relations $D^k_{\underline{\Delta}_\varepsilon}$, $k = 1, \ldots, n$, computes a reduced $(1+\varepsilon)$-approximation (see Theorem 1). Relation $D^k_{\underline{\Delta}_\varepsilon}$ is a powerful relation since a state can possibly dominate all other states of larger weight. This relation requires at most $p+1$ tests to be established between two states.

Observe that, even if the authors of [5] do not explicitly mention the use of a family of dominance relations for $\underline{\Delta}_\varepsilon$, their approach could be restated within Algorithm 1 by using the following family of relations D^k_E defined on S^k by:

$$\text{for all } s^k, \tilde{s}^k \in S^k,\ s^k D^k_E \tilde{s}^k \Leftrightarrow \begin{cases} \mathrm{B}_i(s^k, 1/n) = \mathrm{B}_i(\tilde{s}^k, 1/n), & i = 1, \ldots, p, \quad \text{and} \\ s^k_{p+1} \leq \tilde{s}^k_{p+1} \end{cases}$$

Remark that $D^k_E \subseteq D^k_{\underline{\Delta}_\varepsilon}$ for $e(k) = 1/n$ $(k = 1, \ldots, n)$. This relation, which is quite sufficient to establish the existence of an fptas, has two main disadvantages for an efficient implementation. First, it is very poor since it compares only states lying in the same hyper-rectangle. Therefore, even if two states s^k, $\tilde{s}^k$ are such that $s^k D^k_{\underline{\Delta}} \tilde{s}^k$, we keep both of them in C^k provided that they are not in the same hyper-rectangle. Secondly, by applying a constant error of $(1+\varepsilon)^{1/n}$ at each phase, the total error of $1+\varepsilon$ is shared out equitably among all the phases. During the first phases, since the values of the states are small, the hyper-rectangles in which the states belong usually have a length smaller than 1 on all dimensions. In this case, the advantage of the partition is canceled out since only states with same values could be in relation D^k_E. Thus, the error allocated to these phases is wasted.

For a given $\varepsilon > 0$, the running time of Algorithm 1 using relation $D^k_{\underline{\Delta}_\varepsilon}$ depends crucially on the error function e. In order to guarantee that Algorithm 1 is polynomial both in the size of the instance and in $1/\varepsilon$, we have to add some conditions on the error function aiming at limiting the number of phases with an error equals to 0.

Definition 4 (Polynomial error function). *The error function e is a polynomial error function if for $k = 1, \ldots, n$, $e(k) = 1/g(k)$, if k is a multiple of t, 0 otherwise, where t is a strictly positive integer in $O(\log n)$ and where, for any $k = 1, \ldots, n$, $0 < g(k) \leq cn^d$ for some positive fixed constants c, d.*

The following theorem establishes the complexity of Algorithm 1 using the family of dominance relations $D^k_{\underline{\Delta}_\varepsilon}$.

Theorem 2. *For any $\varepsilon > 0$ and any polynomial error function e, Algorithm 1, using the family of dominance relations $D^k_{\underline{\Delta}_\varepsilon}$, is polynomial both in the size of the instance and in $1/\varepsilon$.*

Hence, by Theorems 1 and 2 we have that, for any $\varepsilon > 0$ and any polynomial error function e, Algorithm 1 using the family of dominance relations $D^k_{\underline{\Delta}_\varepsilon}$ is an fptas that produces a reduced $(1+\varepsilon)$-approximation.

4.2 Complementary dominance relations with respect to $\underline{\Delta}$

Since each dominance relation focuses on specific considerations, it is then desirable to make use of complementary dominance relations. Moreover, when deciding to use a dominance relation, a tradeoff must be made between its potential ability of discarding many states and the time it requires to be checked. We present now two other complementary dominance relations for $\underline{\Delta}$. The first one, D_r^k, is very easy to establish and the second one, D_b^k, although more difficult to establish, is considered owing to its complementarity with D_r^k and $D_{\underline{\Delta}_\varepsilon}^k$.

Relation D_r^k is based on the following observation. When the residual capacity associated to a state s^k of phase k is greater than or equal to the sum of the weights of the remaining items (items $k+1, \ldots, n$), the only completion of s^k that can possibly lead to an efficient solution is the full completion $J = \{k+1, \ldots, n\}$. It is then unnecessary to generate extensions of s^k that do not contain all the remaining items. We define thus the dominance relation D_r^k on S^k for $k = 1, \ldots, n$ by:

$$\text{for all } s^k, \tilde{s}^k \in S^k,\ s^k D_r^k \tilde{s}^k \Leftrightarrow \begin{cases} \tilde{s}^k \in S^{k-1}, \\ s^k = (\tilde{s}_1^k + v_1^k, \ldots, \tilde{s}_p^k + v_p^k, \tilde{s}_{p+1}^k + w^k), \\ \tilde{s}_{p+1}^k \leq W - \sum_{j=k}^n w^j \end{cases} \quad \text{and}$$

This dominance relation is quite poor, since at each phase k it can only appear between a state that does not contain item k and its extension that contains item k. Nevertheless, it is very easy to check since, once the residual capacity $W - \sum_{j=k}^n w^j$ is computed, relation D_r^k requires only one test to be established between two states.

Dominance relation D_b^k is based on the comparison between extensions of a state and an upper bound of all the extensions of another state. In our context, a criterion vector $u = (u_1, \ldots, u_p)$ is an upper bound for a state $s^k \in S^k$ if and only if for all $s^n \in \text{Ext}(s^k)$ we have $u_i \geq s_i^n$, $i = 1, \ldots, p$.

We can derive a general type of dominance relations as follows: considering two states $s^k, \tilde{s}^k \in S^k$, if there exists a completion J of s^k and an upper bound $\tilde{u}$ for $\tilde{s}^k$ such that $s_i^k + \sum_{j \in J} v_i^j \geq \tilde{u}_i$, $i = 1, \ldots, p$, then s^k dominates $\tilde{s}^k$.

This type of dominance relations can be implemented only for specific completions and upper bounds. In our experiments, we just consider two specific completions J' and J'' defined as follows. Let $\mathcal{O}^i$ be an order induced by considering items according to decreasing order of ratios v_i^k/w^k $(i = 1, \ldots, p)$. Let r_i^ℓ be the rank or position of item ℓ in order $\mathcal{O}^i$. Let $\mathcal{O}^{\max}$ be an order according to increasing values of the maximum rank of items in the p orders $\mathcal{O}^i$ $(i = 1, \ldots, p)$ and $\mathcal{O}^{sum}$ be an order according to increasing values of the sum of the ranks of items in the p orders $\mathcal{O}^i$ $(i = 1, \ldots, p)$. After relabeling items $k+1, \ldots, n$ according to $\mathcal{O}^{\max}$, completion J' is obtained by inserting sequentially the remaining items into the solution provided that the capacity constraint is respected. J'' is defined similarly by relabeling items according to $\mathcal{O}^{sum}$. To compute u, we use the upper bound presented in [8, Th 2.2] computed independently for each criterion value.

Finally, we define D_b^k a particular dominance relation of this general type for $k = 1, \ldots, n$ by:

$$\text{for all } s^k, \tilde{s}^k \in S^k,\ s^k D_b^k \tilde{s}^k \Leftrightarrow \begin{cases} s_i^k + \sum_{j \in J'} v_i^j \geq \tilde{u}_i, \quad i = 1, \ldots, p \\ \text{or} \\ s_i^k + \sum_{j \in J''} v_i^j \geq \tilde{u}_i, \quad i = 1, \ldots, p \end{cases}$$

where $\tilde{u} = (\tilde{u}_1, \ldots, \tilde{u}_p)$ is the upper bound, according to [8, Th 2.2] for $\tilde{s}^k$.

D_b^k is harder to check than relations D_r^k, $D_{\underline{\Delta}}^k$ and $D_{\underline{\Delta}_\varepsilon}^k$ since it requires much more tests and state-dependent information.

4.3 Use of multiple dominance relations

In order to be efficient, we will use the dominance relations $D_r^k, D_{\underline{\Delta}_\varepsilon}^k$, and D_b^k at each phase. As underlined in the previous subsection, dominance relations require more or less computational effort to be checked. Moreover, even if they are partly complementary, it often happens that several relations are valid for a same pair of states. It is thus natural to apply first dominance relations which can be checked easily (such as D_r^k and $D_{\underline{\Delta}_\varepsilon}^k$) and then test on a reduced set of states dominance relations requiring a larger computation time (such as D_b^k).

5 Computational experiments and results

5.1 Experimental design

All experiments presented here were performed on a bi-Xeon 3.4GHz with 3072Mb RAM. All algorithms are written in C++. In the bi-objective case ($p = 2$), the following types of instances were considered:

A) Random instances: $v_1^k \in_R [1, 1000]$, $v_2^k \in_R [1, 1000]$ and $w^k \in_R [1, 1000]$

B) Unconflicting instances, where v_1^k is positively correlated with v_2^k: $v_1^k \in_R [111, 1000]$ and $v_2^k \in_R [v_1^k - 100, v_1^k + 100]$, and $w^k \in_R [1, 1000]$

C) Conflicting instances, where v_1^k and v_2^k are negatively correlated: $v_1^k \in_R [1, 1000]$, $v_2^k \in_R [\max\{900 - v_1^k; 1\}, \min\{1100 - v_1^k; 1000\}]$, and $w^k \in_R [1, 1000]$

D) Conflicting instances with correlated weight, where v_1^k and v_2^k are negatively correlated, and w^k is positively correlated with v_1^k and v_2^k: $v_1^k \in_R [1, 1000]$, $v_2^k \in_R [\max\{900 - v_1^k; 1\}, \min\{1100 - v_1^k; 1000\}]$, and $w^k \in_R [v_1^k + v_2^k - 200; v_1^k + v_2^k + 200]$.

where $\in_R [a, b]$ denotes uniformly random generated in $[a, b]$. For all these instances, we set $W = \lfloor 1/2 \sum_{k=1}^n w^k \rfloor$.

Most of the time in the literature, experiments are made on instances of type A. Sometimes, other instances such as those of type B, which were introduced in [9], are studied. However, instances of type B should be viewed as quasi single-criterion instances since they involve two non conflicting criteria. Nevertheless, in a bi-objective context, considering conflicting criteria is a more appropriate

way of modeling real-world situations. For this reason, we introduced instances of types C and D for which criterion values of items are conflicting. In instances of type D, w^k is positively correlated with v_1^k and v_2^k. These instances were introduced in order to verify if positively correlated weight/values instances are harder than uncorrelated weight/values instances as in the single-criterion context [8,10].

For each type of instances and each value of n presented in this study, 10 different instances were generated. In the following, we denote by pTn a p criteria instance of type T with n items. For example, 2A100 denotes a bi-objective instance of type A with 100 items.

In the experiment, we give also, in some tables, the results obtained in [6] by using relations D_r^k, $D_{\underline{\Delta}}^k$ and D_b^k aiming at solving the exact version of the 0-1 multi-objective knapsack problem. These results are denoted by *exact method*.

In the experiment, the approximation method, respectively the exact method, computes only ND_ε, respectively ND. Standard bookkeeping techniques, not considered here, may be used to produce associated solutions.

5.2 Results

We first try to determine the best error function to use in relation $D_{\underline{\Delta}_\varepsilon}^k$. In Table 1 we compare the CPU time in seconds and the size of ND_ε obtained for these three polynomial error functions:

- $e_1(k) = 1/(\lfloor n/t \rfloor)$ if k is a multiple of t, 0 otherwise
- $e_2(k) = \frac{2k/t}{\lfloor n/t \rfloor(\lfloor n/t \rfloor+1)}$ if k is a multiple of t, 0 otherwise
- $e_3(k) = \frac{6(k/t)^2}{\lfloor n/t \rfloor(\lfloor n/t \rfloor+1)(2\lfloor n/t \rfloor+1)}$ if k is a multiple of t, 0 otherwise

where t, which is a strictly positive integer in $O(\log n)$, expresses the frequency of the application of the error. Table 1 shows clearly that the error function has a significant impact on the CPU time and that error function e_2 is significantly better for all types of instances. Thus, in the following, we will use only error function e_2. Observe also that the size of ND_ε is always extremely smaller than the number of non-dominated criterion vectors.

Table 1. Impact of different error functions in our approach

type	avg. time in s.			avg. $\lvert ND_\varepsilon \rvert$			exact method [6]	
	e_1	e_2	e_3	e_1	e_2	e_3	avg t. in s.	avg. $\lvert ND \rvert$
2A-400	51.775	34.347	50.022	332.1	199.8	134.3	307.093	4631.8
2B-1000	0.238	0.180	0.299	1.0	1.0	1.0	8.812	157.0
2C-300	74.308	50.265	68.974	615.3	326.4	227.5	373.097	1130.7
2D-150	65.144	47.398	67.758	703.0	384.3	263.1	265.058	3418.5

$\varepsilon = 0.1$ and frequency $t = 1$

Second, we show the impact of the frequency t in the error function e_2. Table 2 shows that our approach is always faster by setting the frequency $t = \lfloor \log n \rfloor$. Observe that the cardinality of ND_ε is inversely proportional to the frequency t. For example the increase of a factor 3 of the frequency (from $t = \lfloor \log n \rfloor$ to $\lfloor 3 \log n \rfloor$) leads to a decrease of about a factor 3 of the size of ND_ε.

Table 2. Impact of the frequency in the error function e_2

	avg. time in s.				avg. $\lvert ND_\varepsilon \rvert$				exact method [6]	
type	$t=1$	$\lfloor \log n \rfloor$	$\lfloor 2 \log n \rfloor$	$\lfloor 3 \log n \rfloor$	$t=1$	$\lfloor \log n \rfloor$	$\lfloor 2 \log n \rfloor$	$\lfloor 3 \log n \rfloor$	avg. t. in s.	avg. $\lvert ND \rvert$
2A-400	34.347	4.536	5.441	7.664	199.8	31.3	16.1	11.9	307.093	4631.8
2B-1000	0.180	0.120	0.406	1.009	1.0	1.3	1.1	1.0	8.812	157.0
2C-300	50.265	7.511	8.084	11.618	326.4	53.6	27.3	18.8	373.097	1130.7
2D-150	47.398	10.874	11.935	16.156	384.3	70.1	35.8	26.4	265.058	3418.5

$\varepsilon = 0.1$ and error function e_2

Lastly, we present, in Table 3, the performance of our approach on large size instances. The largest instances solved here are those of type B with 20000 items and the instances with the largest reduced $(1+\varepsilon)$-approximation are those of type D with 900 items. Observe that the average maximum cardinality of C^k, which is a good indicator of the memory storage needed to solve the instances, can be very huge. This explains why we can only solve instances of type D up to 900 items.

Table 3. Results of our approach on large size instances

type	n	time in s.			$\lvert ND_\varepsilon \rvert$			avg.
		min	avg.	max	min	avg.	max	$\max_k\{\lvert C^k \rvert\}$
A	100	0.024	0.042	0.072	6	10.1	15	2456.7
	1000	88.121	94.050	103.402	65	68.5	76	321327.6
	2000	896.640	1030.813	1398.060	111	123.9	132	1489132.4
	2500	1635.230	1917.072	2081.410	127	138.6	147	2585169.9
B	1000	0.084	0.118	0.144	1	1.3	2	5596.4
	10000	245.572	269.731	318.808	2	3.3	4	1160906.4
	20000	2424.700	2816.606	3166.580	4	5.3	7	5424849.6
C	100	0.140	0.210	0.316	21	25.3	32	9964.2
	1000	378.135	419.595	471.269	139	150.2	162	923939.4
	2000	3679.770	4296.847	4749.160	255	272.0	285	4256900.6
D	100	1.948	2.356	2.828	50	52.9	57	93507.9
	500	605.837	640.026	681.286	185	196.4	203	3034228.2
	900	4154.610	4689.373	5177.200	297	313.0	329	10276196.8

$\varepsilon = 0.1$, error function e_2 and frequency $t = \lfloor \log n \rfloor$

5.3 Comparison with an exact method

The results of a comparative study between the exact method presented in [6] and our approximation method using relations D_r^k, $D_{\underline{\Delta}_\varepsilon}^k$, and D_b^k are presented in Table 4. We have selected the method presented in [6] since, as shown in this paper, it is the most effective method currently known.

The two methods have been compared on the same instances and the same computer. Table 4 presents results for instances of type A, B, C, and D for increasing size of n for instances that can be solved by the exact method. We give for each series the "average error" that refers to the measured error *a posteriori* which is the smallest value of ε such that the returned set is indeed a reduced $(1+\varepsilon)$-approximation.

Considering the CPU time, the approximation method, of course, is always faster than the exact method (up to more than 600 times faster for instances $2B4000$). Observe that, although the cardinality of ND_ε is very small with regard to the cardinality of ND, the quality of the reduced $(1+\varepsilon)$-approximation is very good since for an error *a priori* $\varepsilon = 0.1$, the error *a posteriori* varies from 0.0023 to 0.0183.

Table 4. Comparison between the exact method presented in [6] and the approximation method

type	n	exact method [6]		approximation method		
		avg. t. in s.	avg. $\lvert ND \rvert$	avg. t. in s.	avg. $\lvert ND_\varepsilon \rvert$	avg. error
A	100	0.328	159.3	0.042 (÷ 8)	10.1 (÷ 16)	0.0159
	700	5447.921	4814.8	32.275 (÷ 169)	49.5 (÷ 97)	0.0060
B	1000	8.812	157.0	0.118 (÷ 75)	1.3 (÷ 121)	0.0041
	4000	6773.264	1542.3	11.220 (÷ 604)	1.8 (÷ 857)	0.0023
C	100	2.869	558.2	0.210 (÷ 14)	25.3 (÷ 22)	0.0178
	500	4547.978	7112.1	44.368 (÷ 103)	88.3 (÷ 81)	0.0064
D	100	40.866	1765.4	2.356 (÷ 17)	52.9 (÷ 33)	0.0183
	250	3383.545	8154.7	62.970 (÷ 54)	110.9 (÷ 74)	0.0098

Approximation: $\varepsilon = 0.1$, e_2 and frequency $t = \lfloor \log n \rfloor$
The decrease factors of the avg. CPU time and of the size of the returned set, corresponding respectively to avg. t. in s. of exact method / avg. t. in s. of approximation method and $|ND|/|ND_\varepsilon|$, are given into brackets

6 Conclusions

The purpose of this work was to design a practically efficient fptas, based on a dynamic programming algorithm, for solving the approximation version of the 0-1 multi-objective knapsack problem. We showed indeed that by using several complementary dominance relations, and sharing the error appropriately among the phases, we obtain an fptas which is experimentally extremely efficient.

While we focused in this paper on the 0-1 multi-objective knapsack problem, we could envisage in future research to apply dominance relations based on similar ideas to the approximation version of other multi-objective problems which admit dynamic programming formulations, such as the multi-objective shortest path problem or multi-objective scheduling problems.

References

1. Ehrgott, M.: Multicriteria optimization. LNEMS 491. Springer, Berlin (2005)
2. Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: IEEE Symposium on Foundations of Computer Science. (2000) 86–92
3. Safer, H.M., Orlin, J.B.: Fast approximation schemes for multi-criteria combinatorial optimization. Working Paper 3756-95, Sloan School (1995)
4. Safer, H.M., Orlin, J.B.: Fast approximation schemes for multi-criteria flow, knapsack, and scheduling problems. Working Paper 3757-95, Sloan School (1995)
5. Erlebach, T., Kellerer, H., Pferschy, U.: Approximating multiobjective knapsack problems. Management Science **48**(12) (2002) 1603–1612
6. Bazgan, C., Hugot, H., Vanderpooten, D.: An efficient implementation for the 0-1 multi-objective knapsack problem. In: 6th Workshop on Experimental Algorithms, WEA'07, LNCS. Volume 4525. (2007) 406–419
7. Weignartner, H., Ness, D.: Methods for the solution of the multi-dimensional 0/1 knapsack problem. Operations Research **15**(1) (1967) 83–103
8. Martello, S., Toth, P.: Knapsack Problems. Wiley, New York (1990)
9. Captivo, M.E., Climaco, J., Figueira, J., Martins, E., Santos, J.L.: Solving bicriteria 0-1 knapsack problems using a labeling algorithm. Computers and Operations Research **30**(12) (2003) 1865–1886
10. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)