A Polynomial Algorithm for a Class of 0–1 Fractional Programming Problems Involving Composite Functions, with an Application to Additive Clustering

Pierre Hansen and Christophe Meyer

Abstract We derive conditions on the functions φ, ρ, v and w such that the 0–1 fractional programming problem $\max\limits_{x \in \{0;1\}^n} \frac{\varphi \circ v(x)}{\rho \circ w(x)}$ can be solved in polynomial time by enumerating the breakpoints of the piecewise linear function $\Phi(\lambda) = \max\limits_{x \in \{0;1\}^n} v(x) - \lambda w(x)$ on $[0; +\infty)$. In particular we show that when φ is convex and increasing, ρ is concave, increasing and strictly positive, v and $-w$ are supermodular and either v or w has a monotonicity property, then the 0–1 fractional programming problem can be solved in polynomial time in essentially the same time complexity than to solve the fractional programming problem $\max\limits_{x \in \{0;1\}^n} \frac{v(x)}{w(x)}$, and this even if φ and ρ are non-rational functions provided that it is possible to compare efficiently the value of the objective function at two given points of $\{0; 1\}^n$. We apply this result to show that a 0–1 fractional programming problem arising in additive clustering can be solved in polynomial time.

Keywords 0–1 fractional programming • Submodular function • Polynomial algorithm • Composite functions • Additive clustering

1 Introduction

We consider the following 0–1 composite fractional programming problem

$$(CFP) \qquad \max_{x \in B_n} \frac{\varphi \circ v(x)}{\rho \circ w(x)}$$

P. Hansen (✉) • C. Meyer
GERAD, HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine,
Montréal, Québec, Canada H3T 2A7
e-mail: pierre.hansen@gerad.ca; christophe.meyer@gerad.ca

where $B_n = \{0; 1\}^n$, φ and ρ are functions from $\mathbb{R}$ to $\mathbb{R}$ and v and w are functions from B_n to $\mathbb{R}$.

In order for the problem (CFP) to be well-defined, we must assume that $\rho \circ w(x) \neq 0$ for all $x \in B_n$. Actually we will make a stronger assumption and assume that ρ is of constant sign on the convex hull $conv\,(w\,(B_n))$ of the image of B_n by w (we will see later that there is little hope to obtain a polynomial algorithm to solve the problem (CFP) when $\rho \circ w(x)$ can assume both positive and negative values on B_n). More precisely we assume that:

(C1) ρ is strictly positive on $conv\,(w\,(B_n))$.

Since the aim of this paper is to identify polynomial instances of problem (CFP), a natural assumption is:

(C2) evaluation and comparison of the value of the objective function $\frac{\varphi \circ v}{\rho \circ w}$ can be done in polynomial time for any two points x and x' of B_n.

We also need to assume that v and w are rational functions. By redefining φ and ρ if necessary, we assume that

(C3) v and w take integral values on B_n.

We explore a solution approach for problem (CFP) that consists in two steps: first we reduce problem (CFP) to the problem of computing a set of points $X^+ \subseteq B_n$ that define the slopes of the piecewise linear function $\Phi(\lambda) = \max\limits_{x \in B_n} v(x) - \lambda w(x)$ on $[0; +\infty)$; then we consider the problem of computing in an efficient way the set X^+. We show that the reduction step is valid if one of the following sets of assumptions is satisfied:

(C4)   there exists $x \in B_n$ such that $(\varphi \circ v)(x) \geq 0$;
(C5) φ and ρ are increasing;
(C6) φ and $-\rho$ are convex;

or:

(C4') $(\varphi \circ v)(x) < 0$ for all $x \in B_n$;
(C5') φ and $-\rho$ are increasing;
(C6') φ and ρ are convex.

Actually we will derive a weaker condition than (C6) and (C6'), but this weaker condition is difficult to exploit as it is expressed in terms of the elements of the set X^+. This weaker condition is implied by (C6) and (C6').

In order for our algorithm to run in polynomial time, we must be able to enumerate in polynomial time the breakpoints of the function Φ. The only nontrivial class of functions that we know for which this can be done in polynomial time is related to the concept of supermodularity. Let us introduce this last set of assumptions:

(C7) v and $-w$ are supermodular on B_n;
(C8)   one of the following conditions is satisfied:

(C8a) v or w takes a polynomial number of distinct values on B_n;

(C8b) v and w are both linear;

(C8c) v or w is monotone and the application $x \mapsto (v(x), w(x))$ is weakly bijective on B_n.

The definitions of supermodularity, monotonicity and weak bijection can be found in Sect. 2.1.1. Note that since the opposite of a submodular function is a supermodular function, we could have expressed some of the above assumptions in different equivalent ways. For example, the assumption ("φ is increasing and v is supermodular") is equivalent to the assumption ("φ is decreasing and v is submodular").

Let $T(n)$ be the time to compute the set X^+ and $U(n)$ be the time to evaluate and compare the value of the objective function at two given points x and x' of B_n. The main results of this paper are:

Theorem 1. *If the conditions (C1)–(C8) are satisfied, then problem (CFP) can be solved in polynomial time* $O\big(T(n) + |X^+|U(n)\big)$.

Theorem 2. *If the conditions (C1)–(C3), (C4')–(C6'), (C7) and (C8) are satisfied, then problem (CFP) can be solved in polynomial time* $O\big(T(n) + |X^+|U(n)\big)$.

By polynomial time, we mean a running time that is polynomial in n and in the size of the number $M = \max\left\{ \max_{x \in B_n} |v(x)|, \max_{x \in B_n} |w(x)|, \max_{x \in B_n} |(\varphi \circ v)(x)|, \max_{x \in B_n} |(\rho \circ w)(x)| \right\}$.

The remaining of this paper is organized as follows. In Sect. 2 we collect several definitions, facts and results from the literature that are pertinent for our work: the concept of supermodularity is reviewed in Sect. 2.1; Sect. 2.2 is devoted to the minimum cut problem (with non-negative capacities) and to problems reducible to it. In Sect. 2.3 we review in more detail the fractional programming problem with particular emphasis on the so-called Dinkelbach's algorithm.

Section 3 is the main part of this paper. We start by defining more precisely the new algorithm in Sect. 3.1. In Sect. 3.2 we present an algorithm to compute the set X^+ and identify sufficient conditions on the functions v and w that guarantee that this algorithm runs in polynomial time. In Sect. 3.3 we determine conditions on the functions φ and ρ that guarantee that the set X^+ computed by the breakpoint enumeration algorithm of Sect. 3.2 actually contains at least one optimal solution of problem (CFP). Putting together the results of the two previous subsections, we then prove Theorems 1 and 2 in Sect. 3.4, where we also discuss the complexity time of the resulting algorithms.

In Sect. 4 we show how our method can be used to derive a polynomial algorithm for a problem arising in additive clustering.

Extensions of our results to minimization problems, maximization of product of composite functions and constrained problems are discussed in Sect. 5, before we conclude in Sect. 6.

2 Definitions and Related Works

2.1 Supermodularity

2.1.1 Definitions

A function f is *supermodular* over B_n if

$$f(x \wedge y) + f(x \vee y) \geq f(x) + f(y) \qquad \forall x, y \in B_n \qquad (1)$$

where $x \wedge y$ is the binary vector whose ith component is the minimum between x_i and y_i, and $x \vee y$ is the binary vector whose ith component is the maximum between x_i and y_i.

A function f is *submodular* if $-f$ is supermodular. A function that is both submodular and supermodular is *modular*. Alternate equivalent definitions exist for super- and submodularity, see, e.g., Nemhauser and Wolsey [41].

For any two vectors x and y of $\mathbb{R}^n$ we write $x \leq y$ if and only if $x_i \leq y_i$ for $i = 1, \ldots, n$, and $x < y$ if and only if $x_i \leq y_i$ for $i = 1, \ldots, n$ and $x \neq y$. We define similarly the notations $x \geq y$ and $x > y$. If neither $x \leq y$ nor $x \geq y$ holds we say that x and y are *not comparable*. Following Topkis [54], a function $f(x)$ from a partially ordered set X to $\mathbb{R}$ is *increasing* (resp. *decreasing*) if $x \leq y$ in X implies $f(x) \leq f(y)$ (resp. $f(x) \geq f(y)$). A function f is *monotone* if it is either increasing or decreasing. A function $f(x)$ from a partially ordered set X to $\mathbb{R}$ is *strictly increasing* (resp. *strictly decreasing*) if $x < y$ in X implies $f(x) < f(y)$ (resp. $f(x) > f(y)$). In this paper the set X will be either $B_n = \{0; 1\}^n$ (a partially ordered set that is not totally ordered) or $\mathbb{R}$ (a partially ordered set that is totally ordered). It is common in lattice theory literature (Topkis [54]) to use the terms *isotone* and *antitone* rather than "increasing" and "decreasing" for a partially ordered set that is not a totally ordered set, but the latter are used herein in order to have a more uniform terminology between the partially ordered set B_n and the totally ordered set $\mathbb{R}$. Although we use only functions with value in a totally ordered set ($\mathbb{R}$ or $\mathbb{N}$), in order to be consistent with Topkis [54] we avoid in this paper the use of the terms "nonincreasing" and "nondecreasing"; in particular the terms "increasing," "decreasing," "strictly increasing" and "strictly decreasing" used in this paper correspond to what may be called "nondecreasing," "nonincreasing," "increasing" and "decreasing" elsewhere.

Finally we say that the application $x \mapsto (v(x), w(x))$ is *weakly bijective* if for all $x, x' \in B_n$,

$$(v(x), w(x)) = (v(x'), w(x')) \quad \Rightarrow \quad x = x' \text{ or } x \text{ and } x' \text{ are not comparable.}$$

2.1.2 Mathematical Results

The following result, which states some conditions on two functions such that their composition is supermodular or submodular, is due to Topkis [53].

Proposition 1. *Let g be a function defined on B_n, and f be a function defined on $conv\,(g\,(B_n))$, where $conv\,(g\,(B_n))$ denotes the convex hull of the image of set B_n by g.*

a) *If f is convex and increasing on $conv\,(g\,(B_n))$ and g is supermodular and monotone on B_n, then $f \circ g$ is supermodular on B_n.*

b) *If f is convex and decreasing on $conv\,(g\,(B_n))$ and g is submodular and monotone on B_n, then $f \circ g$ is supermodular on B_n.*

c) *If f is concave and decreasing on $conv\,(g\,(B_n))$ and g is supermodular and monotone on B_n, then $f \circ g$ is submodular on B_n.*

d) *If f is concave and increasing on $conv\,(g\,(B_n))$ and g is submodular and monotone on B_n, then $f \circ g$ is submodular on B_n.*

Proof. We only prove a) since the proof for the other assertions is similar. Let x, y be two elements of B_n. By definition of the operators $\wedge$ and $\vee$, we have $x \wedge y \leq y \leq x \vee y$. Since g is increasing or decreasing, we thus have

$$g(x \wedge y) \leq g(y) \leq g(x \vee y)$$

or

$$g(x \vee y) \leq g(y) \leq g(x \wedge y).$$

In both cases there exists $t \in [0; 1]$ such that

$$g(y) = tg(x \wedge y) + (1 - t)g(x \vee y). \tag{2}$$

On the other hand, since g is supermodular and by (2)

$$g(x) \leq g(x \wedge y) + g(x \vee y) - g(y) = tg(x \vee y) + (1 - t)g(x \wedge y).$$

Since f is increasing it follows that

$$f\left(g(x)\right) \leq f\left(tg(x \vee y) + (1 - t)g(x \wedge y)\right)$$

$$\leq tf\left(g(x \vee y)\right) + (1 - t)f\left(g(x \wedge y)\right)$$

$$= f\left(g(x \vee y)\right) + f\left(g(x \wedge y)\right) - \left(tf\left(g(x \wedge y)\right)\right.$$

$$\left. + (1 - t)f\left(g(x \vee y)\right)\right)$$

$$\leq f\left(g(x \vee y)\right) + f\left(g(x \wedge y)\right) - f\left(g(y)\right)$$

where we used (2) and twice the convexity of f. Hence $f \circ g$ is supermodular on B_n. ∎

2.1.3 Supermodular Maximization

Consider the following problem:

$$\text{Supermodular Function Maximization (SFM)} : \max_{x \in B_n} f(x)$$

where f is a supermodular function defined on B_n. Note that SFM could also stand for "Submodular Function Minimization" as for example in [39]. Since a function f is submodular if and only if $-f$ is supermodular and since the problem of maximizing f is equivalent to the problem of minimizing $-f$, the two interpretations are however largely equivalent regarding complexity.

Grötschel, Lovász, and Schrijver [26] were the first to provide a (weakly) polynomial time algorithm for SFM which uses the ellipsoid algorithm for linear programming. It was later shown by the same authors [27] that the ellipsoid algorithm can be used to construct a strongly polynomial algorithm for SFM that runs in $\tilde{O}\left(n^5\text{EO} + n^7\right)$ time. Here the notation $\tilde{O}\left(f(n)\right)$ hides the logarithmic factors, i.e., stands for $O\left(f(n) \cdot (\log n)^k\right)$ for some fixed k and EO stands for the time needed for one evaluation of the objective function. However, this result was not considered very satisfactory since the ellipsoid algorithm is not very practical and does not give much combinatorial insight [39]. Then nearly simultaneously two quite different combinatorial strongly polynomial algorithms (combinatorial in the sense of not using the ellipsoid algorithm) were proposed by Schrijver [49] and Iwata et al. [36], both building on previous works by Cunningham [14]. A few years later Orlin [42] proposed a fully combinatorial strongly polynomial algorithm, i.e., an algorithm that does not use multiplication or division. Let M be an upper bound on $\max_{x \in B_n} |f(x)|$. According to McCormick [39] the best theoretical complexity bounds are $O\left((n^4\text{EO} + n^5)\log M\right)$ for weakly polynomial algorithms (Iwata [35]), $O\left(n^5\text{EO} + n^6\right)$ for strongly polynomial algorithms (Orlin [42]) and $O\left(n^8\text{EO}\log^2 n\right)$ for fully combinatorial algorithms (Iwata [35]). See McCormick [39] for a survey of these and other algorithms.

In contrast, maximizing a submodular function is an NP-hard problem as it contains, for example, the *maximum cut* problem in a graph. Therefore, the focus of present works on this problem is to develop good approximation algorithms. This paper does not consider the submodular maximization problem; we refer the reader to [18] for a recent reference.

2.1.4 Parametric Supermodular Maximization: The Notion of Monotone Optimal Solutions

Consider the following parametric supermodular function maximization problem:

$$\text{SFM}(\lambda) \quad \max_{x \in B_n} h(x, \lambda)$$

where λ is either a scalar or a vector of parameters, with value in Λ and $h(x,t)$ is supermodular in x for every $\lambda \in \Lambda$. Let S_λ^* be the set of optimal solutions of problem SFM(λ).

We say that problem SFM(λ) has the *Weak Increasing Optimal Solution Property* (respectively, the *Weak Decreasing Optimal Solution Property*) if for any $\lambda' < \lambda'' \in \Lambda$ and any optimal solution x' of SFM(λ') and any optimal solution x'' of SFM(λ'') it holds that $x' \wedge x''$ (resp. $x' \vee x''$) is an optimal solution of SFM(λ') and $x' \vee x''$ (resp. $x' \wedge x''$) is an optimal solution of SFM(λ'').

The Weak Increasing (resp. Decreasing) Optimal Solution Property implies the existence of an optimal solution x' of SFM(λ') and the existence of an optimal solution x'' of SFM(λ'') such that $x' \leq x''$ (resp. $x' \geq x''$). This ordering relation may, however, not be true for any optimal solutions of SFM(λ') and SFM(λ''). This leads to the definition of the *Strong Increasing Optimal Solution Property* and its decreasing counterpart.

We say that problem SFM(λ) has the *Strong Increasing Optimal Solution Property* (respectively, *Strong Decreasing Optimal Solution Property*) if for any $\lambda' < \lambda'' \in \Lambda$, for any optimal solution x' of SFM(λ') and for any optimal solution x'' of SFM(λ'') it holds that $x' \leq x''$ (resp. $x' \geq x''$).

Finally we say that problem SFM(λ) has the *Weak* (respectively, *Strong*) *Optimal Solution Monotonicity Property* if SFM(λ) has either the *Weak* (resp. *Strong*) *Increasing Optimal Solution Property* or the *Weak* (resp. *Strong*) *Decreasing Optimal Solution Property*.

The Weak and Strong Optimal Solution Monotonicity Property turn out to be a very useful property to prove that some algorithms run in polynomial time, see Proposition 3 together with Propositions 2 and 7.

Sufficient conditions on h have been derived by Topkis [53] (see also [54]) for the problem SFM(λ) to have the Weak Increasing Optimal Solution Property or the Strong Increasing Optimal Solution Property. A straightforward adaptation of his results yields also sufficient conditions for the Weak and Strong Decreasing Optimal Solution Property.

Rather than using these general results, which would require to introduce additional notions, we directly state and prove a sufficient condition for the Weak and Strong Optimal Solution Monotonicity Properties in the particular case where $\Lambda = \{\lambda \in \mathbb{R} : \lambda > 0\}$ and

$$h(x, \lambda) = f(x) - \lambda g(x). \tag{3}$$

A slight improvement can be obtained in this case by replacing the strict monotonicity assumption as a sufficient condition for the Strong Optimal Solution Monotonicity Property by the monotonicity assumption plus the weak bijection property. To see that this is indeed an improvement, consider the pair of functions over B_3: $f(x) = x_1$ and $g(x) = x_2 + x_3$. Both functions are monotone but none of them is strictly monotone. On the other hand, the application $x \mapsto (f(x), g(x))$ is weakly bijective since the only nontrivial solution of equation $(f(x), g(x)) = (f(y), g(y))$ is $x = (u, v, 1-v)$, $y = (u, 1-v, v)$ with $u, v \in \{0; 1\}$

and clearly x and y are not comparable. It is not difficult to show that strict monotonicity implies the weak bijection property.

Proposition 2. *Assume that h has the form* (3), *where f and* $-g$ *are supermodular functions on* B_n, *and that* $\Lambda = \{\lambda \in \mathbb{R} : \lambda > 0\}$.

If f or g is monotone, then $SFM(\lambda)$ *has the Weak Optimal Solution Monotonicity Property.*

If f or g is monotone and the application $x \mapsto (f(x), g(x))$ *is weakly bijective, then* $SFM(\lambda)$ *has the Strong Optimal Solution Monotonicity Property.*

Proof. Let $0 < \lambda' < \lambda''$ and let x' (respectively, x'') be a maximizer of $h(x, \lambda')$ (resp., $h(x, \lambda'')$). We prove the result first in the case where g is increasing, then in the case where f is increasing, and finish by saying a few words on how to modify the proof for the two other cases.

Assume that g is increasing. By optimality of x' and x'',

$$f(x') - \lambda'g(x') \geq f(x' \vee x'') - \lambda'g(x' \vee x'') \tag{4}$$

$$f(x'') - \lambda''g(x'') \geq f(x' \wedge x'') - \lambda''g(x' \wedge x''). \tag{5}$$

Summing the two inequalities yields

$$f(x') + f(x'') - f(x' \vee x'') - f(x' \wedge x'')$$

$$\geq (\lambda'' - \lambda') \left(g(x'') - g(x' \wedge x'') \right)$$

$$+ \lambda' \left(g(x') + g(x'') - g(x' \vee x'') - g(x' \wedge x'') \right). \tag{6}$$

The left-hand side of (6) is nonpositive by supermodularity of f while the right-hand side is nonnegative since $\lambda'' > \lambda' \geq 0$ and since g is submodular and increasing (note that $x'' \geq x' \wedge x''$). Hence all inequalities must be satisfied at equality, i.e.,

$$f(x') - \lambda'g(x') = f(x' \vee x'') - \lambda'g(x' \vee x'')$$

$$f(x'') - \lambda''g(x'') = f(x' \wedge x'') - \lambda''g(x' \wedge x'')$$

$$f(x') + f(x'') - f(x' \vee x'') - f(x' \wedge x'') = 0$$

$$g(x'') - g(x' \wedge x'') = 0.$$

The first two equalities show that $y' = x' \vee x''$ is a maximizer of $h(x, \lambda')$ and $y'' = x' \wedge x''$ is a maximizer of $h(x, \lambda'')$, hence that $SFM(\lambda)$ has the Weak Decreasing Optimal Solution Property. From the remaining equalities it follows that $g(x'') = g(y'')$ and $f(x'') = f(y'')$. Since x'' and y'' are comparable, the weak bijection property implies $x'' = y''$. Since $y'' \leq x'$, we conclude that $SFM(\lambda)$ has the Strong Decreasing Optimal Solution Property.

Assume now that f is increasing. By multiplying (4) by $\frac{1}{\lambda'}$ and (5) by $\frac{1}{\lambda''}$ and summing, we obtain

$$g(x' \vee x'') + g(x' \wedge x'') - g(x') - g(x'')$$

$$\geq -\left(\frac{1}{\lambda'} - \frac{1}{\lambda''}\right)\left(f(x' \wedge x'') - f(x'')\right)$$

$$+ \frac{1}{\lambda'}\left(f(x' \vee x'') + f(x' \wedge x'') - f(x') - f(x'')\right). \qquad (7)$$

The left-hand side of (7) is nonpositive by submodularity of g while the right-hand side is nonnegative since $\frac{1}{\lambda'} > \frac{1}{\lambda''} > 0$ and since f is supermodular and increasing. Therefore all inequalities must hold at equality, in particular inequalities (4)–(5). We conclude again that $x' \vee x''$ is a maximizer of $h(x, \lambda')$ and $x' \wedge x''$ is a maximizer of $h(x, \lambda'')$, hence that SFM(λ) has the Weak Decreasing Optimal Solution Property. The Strong Property follows from the monotonicity of f or g and the weak bijection property in the same way than for the case where g is increasing.

If f or g is decreasing we replace inequalities (4)–(5) by

$$f(x') - \lambda' g(x') \geq f(x' \wedge x'') - \lambda' g(x' \wedge x'')$$

$$f(x'') - \lambda'' g(x'') \geq f(x' \vee x'') - \lambda'' g(x' \vee x'').$$

The rest of the proof is similar. In both cases we conclude that SFM(λ) has the Weak or Strong Increasing Optimal Solution Property, depending on whether the weak bijection property holds or not. ∎

2.2 The Minimum Cut Problem

2.2.1 Definition

Let $G = (V, A)$ be a directed graph with vertex set V and arc set A. With each arc $(v_i, v_j) \in A$ we associate a nonnegative number c_{ij}, called the capacity of the arc (v_i, v_j). Given two subsets S and T of V we denote by (S, T) the set of arcs with origin in S and destination in T, that is

$$(S, T) = \{(v_i, v_j) : v_i \in S \text{ and } v_j \in T\}. \qquad (8)$$

Assume that two distinct vertices s and t are given, s being called the *source* and t the *sink*. An (s, t)-*cut*, or more simply a *cut*, is a set $(S, \overline{S})$ (as defined in (8)) with $s \in T$, $t \in \overline{S}$ where $\overline{S} = V \setminus S$ denotes the complement of S. Note that a cut is a set of arcs induced by a set S of nodes. The quantity $c(S, \overline{S}) = \sum_{(v_i, v_j) \in (S, \overline{S})} c_{ij}$ is called the *capacity* of the cut. The *minimum cut problem* consists in determining

the subset S of V that minimizes the capacity $c(S, \overline{S})$. The minimum cut problem can be solved in polynomial time thanks to the max flow–min cut theorem that establishes a strong relation with a linear problem, the *maximum flow problem*, see e.g., Ahuja et al. [1].

2.2.2 The Selection Problem

Hammer and Rudeanu [29] have shown that every function defined on B_n can be written in a unique way as

$$f(x) = \sum_{S \in A} a_S \prod_{i \in S} x_i - \sum_{i=1}^{n} c_i x_i \tag{9}$$

where A is a family of subsets of $\{1, 2, \ldots, n\}$ of size at least 2 and a_S ($S \in A$) and c_j ($j = 1, \ldots, n$) are real numbers. An important special case is obtained by adding the restriction

$$a_S \geq 0, \qquad S \in A. \tag{10}$$

When the restriction (10) holds, the problem of maximizing f given by (9) is called a *selection problem* (Rhys [46], Balinski [3]). It was shown by Rhys and Balinski that the selection problem can be formulated as a minimum cut problem in a network defined as follows. With each product of variables $\prod_{i \in S} x_i$ we associate a vertex v_S and with each variable x_i we associate a vertex v_i ($i = 1, \ldots, n$). There are two more vertices: a source s and a sink t. There is an arc from the source to each vertex v_S with capacity a_S. For each S and for each $i \in S$, there is an arc with infinite capacity from vertex v_S to vertex v_i. Finally for each $i = 1, \ldots, n$ there is an arc from vertex v_i to the sink vertex t with capacity $-c_i$ if $c_i < 0$ or an arc from the source vertex s to the vertex v_i with capacity c_i if $c_i > 0$ (no such arc is needed for vertices v_i such that $c_i = 0$). The network has $n' = |A| + n + 2$ nodes and $m' = |A| + \sum_{S \in A} |S| + n$ arcs.

A network of smaller size exists when the degree of f is ≤ 2 (the degree of f is defined as the largest cardinality of a subset in A). This network has $n + 2$ nodes and $n + |A|$ arcs, see Hammer [34].

It is not difficult to show that the set of functions of degree ≤ 2 that can be written as (9) with the restriction (10) coincides with the set of functions of degree ≤ 2 that are supermodular. This is not true anymore for functions of larger degree as supermodular functions of degree 3 can have negative a_S, see Billionnet and Minoux [5].

2.2.3 The Parametric Minimum Cut Problem

In several applications the capacities of the arcs in a minimum cut problem depend on one or more parameters, and we would like to find a minimum cut for all possible values of the parameters. Gallo et al. [22] have developed an algorithm that solves a special class of parametric minimum cut problem with a single parameter in the same complexity time than what would be necessary to solve the minimum cut problem for a fixed value of the parameter (solving a parametric problem means here to find an optimal solution for all possible values of the parameter). In this special class of parametric minimum cut problem, the capacities of the arcs leaving the source are nondecreasing functions of the (unique) parameter, those of arcs entering the sink are nonincreasing functions of the parameter, and those of all other arcs are constant. The complexity of the Gallo, Grigoriadis and Tarjan algorithm is $O\left(m'n'\log(n'^2/m')\right)$ where n' denotes the number of nodes and m' the number of arcs in the network.

Other classes of the parametric minimum cut problem for which this "all-in-one" property holds have since been identified: see the recent paper by Granot et al. [25] and the references therein.

2.3 Single-Ratio Fractional Programming

Problem (CFP) is a 0–1 (single-ratio) fractional programming problem. In general the *(single-ratio) fractional programming* problem is defined as

$$\text{(FP)} \qquad \max_{x \in S} \quad \frac{F(x)}{G(x)} \qquad (11)$$

where F and G are real valued functions on a subset S of $\mathbb{R}^n$ and $G(x) > 0$ for all $x \in S$.

The single-ratio fractional programming problem has received considerable attention from the continuous optimization community since the 1960s [10, 16]. According to Frenk and Schaible [19], many of the results on this topic were already presented in the first monograph on fractional programming published in 1978 by Schaible [47]. The focus has since shifted to problems involving multiple ratios, where one, for example, seeks to maximize the sum of several ratios, or maximize the minimum value of several ratios. Other monographs on fractional programming are Craven [13] and Stancu-Minasian [51], see also [20, 48, 52].

The discrete version of the problem also received considerable interests. When $S = \{0; 1\}^n$, the research focused on the case where F and G are polynomials: see, for example, Hansen et al. [31] for the linear case, Hochbaum [33] and the references therein for the quadratic case, Picard and Queyranne [43], Gallo et al. [22] and Chang [9] for polynomials of larger degree.

When constraints are allowed, the functions appearing in the ratio are generally assumed to be linear. Problems that have been considered include the *minimum ratio spanning-tree problem*, the *maximum profit-to-time ratio cycle problem*, the *minimum mean cycle problem*, the *maximum mean cut problem* and the *fractional 0–1 knapsack problem*: see [45] for references to these problems. See also Correa et al. [12], Ursulenko [55].

2.3.1   The Parametric Approach

Almost every solution method developed for fractional programming since the seminal work of Dinkelbach [16] introduces the following auxiliary problem:

$$\text{FPaux}(\lambda) \quad \max_{x \in S} h_\lambda(x) = F(x) - \lambda G(x).$$

λ can be viewed as a "guess" for the optimal value $\frac{F(x^*)}{G(x^*)}$ of problem (FP): if λ is smaller than $\frac{F(x^*)}{G(x^*)}$, the optimal value of the auxiliary problem FPaux(λ) will be positive and its optimal solution will provide a feasible solution with objective value larger than λ; if, on the other hand, λ is larger than $\frac{F(x^*)}{G(x^*)}$, the optimal value of the auxiliary problem will be negative.

We present below Dinkelbach's algorithm for fractional programming. For variants of it, see, e.g., Radzik [45]. Note in particular the proximity of this method with the Newton method for finding roots of polynomial.

DINKELBACH'S ALGORITHM

Step 0. Select some $x^0 \in S$. Compute $\lambda_0 = \frac{F(x^0)}{G(x^0)}$. Set $k = 0$.

Step 1. Solve the auxiliary problem FPaux(λ_k). Let x^{k+1} be an optimal solution.

Step 2. If $h_{\lambda_k}(x^{k+1}) = 0$, stop: $x^* = x^k$. Otherwise let $\lambda_{k+1} = \frac{F(x^{k+1})}{G(x^{k+1})}$, replace k by $k + 1$ and go to Step 1.

The complexity of Dinkelbach's algorithm is determined by the number of iterations and by the complexity of solving the auxiliary problem FPaux(λ) for a given λ. A property that is very useful to derive polynomial algorithms for the fractional programming problem is the supermodularity (see Sect. 2.1).

Consider the 0–1 unconstrained case, i.e., the case where $S = B_n$ and assume that we know that the optimal value of problem (FP) is positive, i.e., that there exists at least one $\tilde{x} \in S$ such that $F(\tilde{x}) \geq 0$. Then we can restrict our attention to $\lambda \geq 0$. If the function $h_\lambda(x)$ is supermodular in x for any $\lambda \geq 0$ then the auxiliary problem FPaux(λ) can generally be solved in polynomial time by one of the algorithms mentioned in Sect. 2.1.3 for SFM. Moreover if FPaux(λ) has the Strong Optimal Solution Monotonicity Property (see Sect. 2.1.4), then the number of iterations in Dinkelbach's algorithm is bounded by n. More precisely we have:

Proposition 3. *Assume that $S = B_n$, that F and G are rational functions that can be evaluated at a given point in polynomial time, that F and $-G$ are supermodular on B_n, that the optimal value of problem (FP) is positive, that either F or G is monotone and that the application $x \mapsto (F(x), G(x))$ is weakly bijective. Then problem (FP) can be solved in polynomial time.*

3 A New Algorithm

This section is the main part of our paper. We start by defining precisely our algorithm in Sect. 3.1. In Sect. 3.2 we characterize the breakpoint vertex set, present an algorithm to compute it and derive conditions on v and w such that this algorithm is polynomial. In Sect. 3.3 we derive conditions on functions φ and ρ that guarantees that our algorithm correctly finds an optimal solution of problem (CFP). Theorems 1 and 2 are proved in Sect. 3.4.

3.1 Description

Let us introduce the function

$$L_\lambda(x) = v(x) - \lambda w(x)$$

and the parametric problem

$$\text{PARAM}(\lambda) \quad \Phi(\lambda) = \max_{x \in B_n} L_\lambda(x).$$

We will denote by $\hat{x}(\lambda)$ an optimal solution of problem PARAM(λ).

Note that the function $L_\lambda(x)$ coincides with the function $h(x, \lambda)$ that would be considered when solving the problem $\max_{x \in B_n} \frac{v(x)}{w(x)}$ by Dinkelbach's algorithm.

It is well-known that $\Phi(\lambda) = \max_{x \in B_n} L_\lambda(x)$ is a convex piecewise linear function on $\mathbb{R}$ (see, e.g., Nemhauser and Wolsey [41, Corollary 6.4]). Let $\mu_1 > \mu_2 > \cdots > \mu_q$ denote the breakpoints of $\Phi(\lambda)$ and let $X = \{x^0, \ldots, x^q\}$ be a subset of B_n such that

$$\Phi(\lambda) = \begin{cases} v(x^q) - \lambda w(x^q), \ \lambda \in (-\infty, \mu_q] \\ v(x^k) - \lambda w(x^k), \ \lambda \in [\mu_{k+1}, \mu_k] & \text{for } k = 1, \ldots, q-1 \\ v(x^0) - \lambda w(x^0), \ \lambda \in [\mu_1, +\infty) \end{cases} \quad (12)$$

with

$$w(x^{k-1}) < w(x^k) \qquad k = 1, \dots, q. \tag{13}$$

By continuity of Φ we have easily

$$\mu_k = \frac{v(x^k) - v(x^{k-1})}{w(x^k) - w(x^{k-1})}, \qquad k = 1, \dots, q. \tag{14}$$

We will consider subsets of X. Given an interval I of $\mathbb{R}$, we define the set $X_I \subseteq X$ as the set of points x^k needed to define $\Phi(\lambda)$ on the interval I via the formula (12). In particular, $X = X_{(-\infty, +\infty)}$. The set X_I will be called the *breakpoint vertex set* for the function $\Phi(\lambda)$ on interval I. We will be more particularly interested in the set $X_{[0, +\infty)}$, that we will denote more concisely by X^+.

We propose the following algorithm for problem (CFP):

ALGORITHM HM_CFP

Step 1. Construct the set X^+.

Step 2. Compute $x^* = \arg \max_{x \in X^+} \dfrac{(\varphi \circ v)(x)}{(\rho \circ w)(x)}$.

In Sect. 3.2 we study the properties of the set X_I and present an algorithm for its computation. We then determine sufficient conditions on v and w for this algorithm to run in polynomial time in the particular case where $I = [0; +\infty)$. One of the properties identified in Sect. 3.2 is used in Sect. 3.3 to derive conditions on the functions φ and ρ that guarantee the correctness of the algorithm HM_CFP. The results of the previous subsections are used in Sect. 3.4 to identify classes of problems (CFP) that can be solved in polynomial time.

3.2 Computing the Breakpoint Vertex Set

In Sect. 3.2.1 we derive a certain number of properties of the breakpoints and of the breakpoint vertex set, that will be used both to show the correctness of the Eisner and Severance algorithm presented in the next subsection and to derive a sufficient condition on the functions φ and ρ for the set X^+ to contain at least one optimal solution of problem (CFP) in Sect. 3.3. In Sect. 3.2.2 we present the Eisner and Severance algorithm to compute the breakpoint vertex set X_I on a given interval I with at most $2N(I)$ evaluations of the function $\Phi(\lambda)$, where $N(I)$ is the number of breakpoints of Φ on the interval I. Finally in Sect. 3.2.3 we derive conditions on functions v and w that guarantee that the Eisner and Severance algorithm runs in polynomial time when $I = [0; +\infty)$.

3.2.1 Properties

In this section we give some properties of the breakpoint vertex set X introduced in Sect. 3.1. It must be noted that the results in this subsection and in the next one are completely general: no assumption is made on the functions v and w other than being defined on B_n.

We first observe that the set X (and therefore the set X_I for a given interval I) may not be unique if there exists $x', x'' \in B_n$ with $x' \neq x''$ such that $(v(x'), w(x')) = (v(x''), w(x''))$. However both for the purpose of defining the function $\Phi(\lambda)$ and for the algorithm HM_CFP, the two points x' and x'' are completely equivalent. By a slight abuse of language we will continue to write "the set X" (or "the set X_I") in the sequel of this paper.

We now state without proof three easy lemmas.

Lemma 1. *If $\lambda' < \lambda''$ then $-w(\hat{x}(\lambda')) \leq -w(\hat{x}(\lambda''))$.*
Moreover equality holds if and only if $L_{\lambda'}(\hat{x}(\lambda'')) = \Phi(\lambda')$ and $L_{\lambda''}(\hat{x}(\lambda')) = \Phi(\lambda'')$ and in that case we have also $v(\hat{x}(\lambda')) = v(\hat{x}(\lambda''))$.

Lemma 2. *Let $\lambda' < \lambda''$ and assume that $\tilde{x}$ is an optimal solution of both problems PARAM(λ') and PARAM(λ''). Then $\Phi(\lambda)$ is linear on $[\lambda', \lambda'']$.*

Lemma 3. $w(x^0) = \min\limits_{x \in B_n} w(x)$. *Moreover if there exists more than one optimal solution, x^0 is one of them that maximizes $v(x)$.*
$w(x^q) = \max\limits_{x \in B_n} w(x)$. *Moreover if there exists more than one optimal solution, x^q is one of them that maximizes $v(x)$.*

The next result will be used to establish sufficient conditions on the functions φ and ρ for the set X^+ to contain an optimal solution of problem (CFP).

Proposition 4. *It holds:*

$$\frac{v(x) - v(x^{k-1})}{w(x) - w(x^{k-1})} \leq \frac{v(x^k) - v(x)}{w(x^k) - w(x)} \qquad \forall x \in B_n : w(x^{k-1}) < w(x) < w(x^k).$$

Proof. By definition of the breakpoints and of the x^k we have

$$v(x^k) - \mu_k w(x^k) \geq v(x) - \mu_k w(x) \qquad \forall x \in B_n.$$

In particular for all $x \in B_n$ such that $w(x^{k-1}) < w(x) < w(x^k)$:

$$v(x) - \mu_k w(x) \leq v(x^k) - \mu_k w(x^k)$$

$$\Rightarrow \frac{v(x^k) - v(x^{k-1})}{w(x^k) - w(x^{k-1})} = \mu_k \leq \frac{v(x^k) - v(x)}{w(x^k) - w(x)}$$

$$\Rightarrow \frac{\left(v(x^k) - v(x)\right) + \left(v(x) - v(x^{k-1})\right)}{\left(w(x^k) - w(x)\right) + \left(w(x) - w(x^{k-1})\right)} \leq \frac{\left(v(x^k) - v(x)\right)}{\left(w(x^k) - w(x)\right)}$$

where we used (14). Since each term delimited by a pair of parentheses in the denominators is strictly positive, a simple manipulation gives the announced inequality. ∎

We terminate by pointing out that another characterization of the breakpoints can be found in Gallo and Simeone [21], as well as a different approach to compute the breakpoint vertex set that requires to solve a constrained version of problem PARAM(λ).

3.2.2 The Eisner and Severance Algorithm

In this section we present an algorithm to compute the breakpoint vertex set X_I of $\Phi(\lambda)$ on a (finite) interval $I = [\underline{\lambda}, \overline{\lambda}]$, that works for any functions v and w and that requires at most $2N$ solutions of the problem PARAM(λ), where N is the number of breakpoints of $\Phi(\lambda)$ in the interval I. This algorithm was originally proposed by Eisner and Severance [17], see also Gusfield [28].

We first give an informal description of the algorithm. Basically the algorithm partitions the given interval I into subintervals $[\lambda_j, \lambda_{j+1}]$ for $j \in J$. With each λ_j we associate a point $\hat{x}(\lambda_j)$ of B_n that is an optimal solution of problem PARAM(λ_j). The algorithm stops when it can be shown that $\Phi(\lambda)$ is linear on every interval of the partition. Note that by Lemma 2 a sufficient condition for $\Phi(\lambda)$ to be linear on the interval $[\lambda_j, \lambda_{j+1}]$ is that $\hat{x}(\lambda_j)$ is also an optimal solution of problem PARAM(λ_{j+1}) or $\hat{x}(\lambda_{j+1})$ is also an optimal solution of problem PARAM(λ_j). If it is not possible to show that Φ is linear on all intervals of the current partition, we select an interval on which Φ is not known to be linear and subdivide it.

We now explain the subdivision process. Let $[\lambda', \lambda'']$ be an interval to be subdivided, and let $\hat{x}' = \hat{x}(\lambda')$ and $\hat{x}'' = \hat{x}(\lambda'')$ be the optimal solutions associated with the bounds of the interval. We assume that $\hat{x}'$ is not an optimal solution of problem PARAM(λ'') and $\hat{x}''$ is not an optimal solution of problem PARAM(λ') since otherwise Φ would be linear on the interval, which therefore would not have been selected for subdivision. In particular $w(\hat{x}') > w(\hat{x}'')$ by Lemma 1. Define

$$\tilde{\lambda} = \frac{v(\hat{x}') - v(\hat{x}'')}{w(\hat{x}') - w(\hat{x}'')}. \tag{15}$$

We argue that $\tilde{\lambda} \in (\lambda', \lambda'')$. Indeed

$$\tilde{\lambda} - \lambda' = \frac{\Big(v(\hat{x}') - \lambda' w(\hat{x}')\Big) - \Big(v(\hat{x}'') - \lambda' w(\hat{x}'')\Big)}{w(\hat{x}') - w(\hat{x}'')} > 0$$

Step 1 (initialization)   : Assume that $I = [\underline{\lambda}, \overline{\lambda}]$. Solve $PARAM(\underline{\lambda})$, obtaining an optimal solution $\hat{x}(\underline{\lambda})$ and an optimal value $\Phi(\underline{\lambda})$. Solve $PARAM(\overline{\lambda})$, obtaining an optimal solution $\hat{x}(\overline{\lambda})$ and an optimal value $\Phi(\overline{\lambda})$. Set $\mathcal{L} = \mathcal{NL} = X_I = \emptyset$ and $New = \{[\underline{\lambda}, \overline{\lambda}]\}$.

Step 2 (linearity test)   : For each interval $I' = [\lambda', \lambda'']$ in New, do the following:
      If $L_{\lambda''}(\hat{x}(\lambda')) = \Phi(\lambda'')$, add $\hat{x}(\lambda')$ to X_I and add I' to $\mathcal{L}$.
      If $L_{\lambda'}(\hat{x}(\lambda'')) = \Phi(\lambda')$, add $\hat{x}(\lambda'')$ to X_I and add I' to $\mathcal{L}$.
      If none of the above two cases occur, add I' to $\mathcal{NL}$.
    Set New to $\emptyset$.

Step 3 (optimality test)   : If $\mathcal{NL} = \emptyset$, stop and return X_I.

Step 4 (subdivision)   : Select an interval $I' = [\lambda', \lambda'']$ in $\mathcal{NL}$, compute $\tilde{\lambda}$ by the formula (15), solve $PARAM(\tilde{\lambda})$, obtaining an optimal solution $\hat{x}(\tilde{\lambda})$ and an optimal value $\Phi(\tilde{\lambda})$. Set $New = \{[\lambda', \tilde{\lambda}], [\tilde{\lambda}, \lambda'']\}$ and $\mathcal{NL} = \mathcal{NL} \setminus \{I'\}$. Return to Step 2.
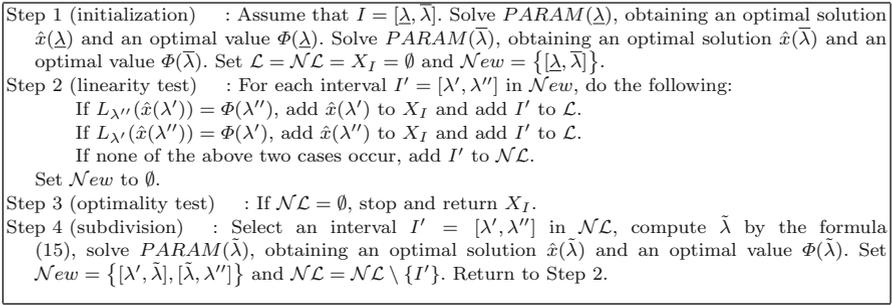
Fig. 1 The Eisner and Severance algorithm for computing X_I

since the numerator is strictly positive by optimality of $\hat{x}'$ and non-optimality of $\hat{x}''$ to problem PARAM(λ'). We show in a similar way that $\tilde{\lambda} - \lambda'' < 0$. The subdivision process consists in replacing the interval $[\lambda', \lambda'']$ by the two intervals $[\lambda', \tilde{\lambda}]$ and $[\tilde{\lambda}, \lambda'']$.

The algorithm will maintain three sets: $\mathcal{L}$ is the subset of intervals of the current partition on which Φ was shown to be linear; $\mathcal{NL}$ is the subset of intervals of the current partition on which Φ is not known to be linear and New is the set of new intervals generated during the last iteration. At Step 3, in every iteration, the intervals in $\mathcal{L} \cup \mathcal{NL}$ form a partition of the given interval I. A formal description of the algorithm is given in Fig. 1.

Proposition 5. *Let N be the number of breakpoints of $\Phi(\lambda)$ in interval I, including the lower and upper bounds of I. The above algorithm is correct and terminates after solving at most $2N - 1$ problems* PARAM(λ).

Proof. Consider a point $\hat{x}(t)$ generated by the algorithm, where t is a bound of an interval. There are two possibilities for $\hat{x}(t)$:

- t coincides with a breakpoint. Obviously this case can happen at most N times.
- t lies in the interior of an interval defined by two consecutive breakpoints. To fix the idea assume that $t \in (\mu_k, \mu_{k-1})$ for some k. Then $\hat{x}(t)$ defines the linear piece on $[\mu_k, \mu_{k-1}]$, i.e, $\Phi(\lambda) = v(\hat{x}(t)) - \lambda w(\hat{x}(t))$ for all $\lambda \in [\mu_k, \mu_{k-1}]$. We claim that at most one such point will be generated by the algorithm for each piece of the piecewise linear function $\Phi(\lambda)$. To show this we will prove that if there is another point $\hat{x}(t')$ generated by the algorithm that defines the same linear function, then t' must be a breakpoint. Assume that the values of λ considered by the algorithm are $\lambda_1 < \lambda_2 < \cdots < \lambda_r$. Since we have $w(\hat{x}(\lambda_j)) \geq w(\hat{x}(\lambda_{j+1}))$ for $j = 1, \ldots, r - 1$ by Lemma 1 we can assume that $t = \lambda_j$ and $t' = \lambda_{j+1}$ for some j, or the converse. Assume furthermore that λ_j was generated by the algorithm before λ_{j+1} (if this is not the case, we simply invert the roles of λ_j and λ_{j+1}). Then λ_{j+1} corresponds to the $\tilde{\lambda}$ of formula (15) for an interval $[\lambda_j, \lambda'']$, i.e.,

$$\lambda_{j+1} = \frac{v(\hat{x}(\lambda'')) - v(\hat{x}(\lambda_j))}{w(\hat{x}(\lambda'')) - w(\hat{x}(\lambda_j))} = \frac{v(\hat{x}(\lambda'')) - v(\hat{x}(\lambda_{j+1}))}{w(\hat{x}(\lambda'')) - w(\hat{x}(\lambda_{j+1}))}$$

where we used the fact that $v(\hat{x}(\lambda_j)) = v(\hat{x}(\lambda_{j+1}))$ and $w(\hat{x}(\lambda_j)) = w(\hat{x}(\lambda_{j+1}))$ as the two points $\hat{x}(\lambda_j))$ and $\hat{x}(\lambda_{j+1})$ define the same piece of linear function. We then have

$$v(\hat{x}(\lambda_{j+1})) - \lambda_{j+1}w(\hat{x}(\lambda_{j+1})) = v(\hat{x}(\lambda'')) - \lambda_{j+1}w(\hat{x}(\lambda''))$$

which shows that $\hat{x}(\lambda'')$ is an optimal solution of problem PARAM(λ_{j+1}). Hence Φ is linear on $[\lambda_{j+1}, \lambda'']$ by Lemma 2, more precisely $\Phi(\lambda) = v(\hat{x}(\lambda'')) - \lambda w(\hat{x}(\lambda''))$ for $\lambda \in [\lambda_{j+1}, \lambda'']$. Since the interval $[\lambda_j, \lambda'']$ was subdivided, $\hat{x}(\lambda'')$ is not an optimal solution of problem PARAM(λ_j), hence we have $w(\hat{x}(\lambda_j)) > w(\hat{x}(\lambda''))$ by Lemma 1. This shows that the two pieces of linear functions are different on the two intervals $[\lambda_j, \lambda_{j+1}]$ and $[\lambda_{j+1}, \lambda'']$. We therefore conclude that λ_{j+1} is a breakpoint. By the monotonicity of the slopes, there can be no λ_ℓ for $\ell > j+1$ that defines the same linear part than λ_j. We can therefore conclude that the number of generated λ_j that lies strictly between two consecutive breakpoints is bounded by $N - 1$.

Therefore the algorithm generates at most $2N - 1$ points, in particular it is finite. Since the algorithm can only stop when $\mathcal{L}$ contains a partition of the given interval I, we conclude that the algorithm is correct. ∎

3.2.3 Complexity

Two conditions must be met for the Eisner and Severance algorithm to compute the set X_I in polynomial time: the number of breakpoints N of $\Phi(\lambda)$ on interval I (or equivalently the size of X_I) must be polynomial in n, and the problem PARAM(λ) must be solvable in polynomial time for fixed λ in I.

In this section we assume that $I = [0; +\infty)$. Note that the upper bound of this interval is not finite as assumed in Sect. 3.2.2, which raises two additional difficulties: we have to find a finite upper bound $\bar{\lambda}$ such that running the Eisner and Severance algorithm on $[0; \bar{\lambda}]$ gives a description of $\Phi(\lambda)$ on the larger interval $[0; +\infty)$, and we have to show that the size of $\bar{\lambda}$ remains polynomial in the size of the data. By (12) and (14), $\bar{\lambda}$ should be chosen such that

$$\bar{\lambda} > \mu_1 = \frac{v(x^1) - v(x^0)}{w(x^1) - w(x^0)}.$$

Let us show that

$$\bar{\lambda} = 1 + \left(\max_{x \in B_n} v(x)\right) - v(\tilde{x})$$

where $\tilde{x}$ is an optimal solution of problem $\min\limits_{x \in B_n} w(x)$, is a valid choice. Since x^0 is an optimal solution of problem $\min\limits_{x \in B_n} w(x)$ that maximizes $v(x)$ by Lemma 3, we have $v(x^0) \geq v(\tilde{x})$. Since $w(x^1) > w(x^0)$ and w takes integral values on B_n we have then $\mu_1 \leq v(x^1) - v(x^0) \leq \left(\max\limits_{x \in B_n} v(x) \right) - v(\tilde{x})$, hence $\overline{\lambda} > \mu_1$. It follows from Sect. 2.1.3 that $\max\limits_{x \in B_n} v(x)$ and $\tilde{x}$ (and therefore $\overline{\lambda}$) can be computed in polynomial time if v is supermodular and w is submodular. Moreover the size of $\overline{\lambda}$ is polynomial in the size of $\max\limits_{x \in B_n} |v(x)|$.

We now consider the condition that problem PARAM(λ) must be solvable in polynomial time for fixed $\lambda \geq 0$. We know of only one sufficiently large class of functions that can be maximized over B_n in polynomial time: it is the class of supermodular functions, see Sect. 2.1. The function $L_\lambda(x)$ is supermodular in x for all $\lambda \geq 0$ if and only if v is supermodular and w is submodular on B_n.

Proposition 6. *If the functions v and $-w$ are supermodular, then the problem* PARAM(λ) *can be solved in polynomial time for any fixed positive λ.*

Proof. Use one of the SFM algorithms mentioned in Sect. 2.1.3. ∎

The other necessary condition for the Eisner and Severance algorithm to run in polynomial time is that the set X^+ is of polynomial size. This condition is satisfied in the following cases:

- When the function v or w takes a polynomial number of distinct values on X. Indeed by (13) the sequence $\{w(x^k)\}$ is strictly increasing; and since we restrict ourselves to $\lambda \geq 0$ and by (14), this is also true for the sequence $\{v(x^k)\}$. Thus the number of breakpoints (and hence the size of X^+) is bounded by the number of distinct values taken by v (or w). Examples of such functions are functions that depend on at most $O(\log n)$ variables; $\sum\limits_{j \in J} x_j$ for some subset J of $\{1, 2, \ldots, n\}$; or combination of a fixed number of the above functions.
- When v and w are both linear functions. Indeed it was shown by Hansen et al. [31] that the number of breakpoints is bounded by $n + 1$.
- When PARAM(λ) has the Strong Optimal Solution Monotonicity Property:

    Proposition 7. *Assume that the problem* PARAM(λ) *has the Strong Optimal Solution Monotonicity Property for $\lambda \geq 0$. Then $|X^+| \leq n + 1$.*

    Proof. Let $0 \leq \lambda_1 < \lambda_2 \cdots < \lambda_r$ be the breakpoints generated by the algorithm, sorted in increasing order (i.e., we do not consider here the λ generated by the algorithm that are strictly between two breakpoints). Define $\alpha_i = \frac{\lambda_i + \lambda_{i+1}}{2}$ for $i = 1, \ldots, r - 1$, so that each α_i is in the interior of an interval defined by two consecutive breakpoints. If PARAM(λ) has the Strong Optimal Solution Monotonicity Property for $\lambda \geq 0$, then either $\hat{x}(\alpha_1) \leq \hat{x}(\alpha_2) \leq \cdots \leq \hat{x}(\alpha_{r-1})$ or

$\hat{x}(\alpha_1) \geq \hat{x}(\alpha_2) \geq \cdots \geq \hat{x}(\alpha_{r-1})$. Since the $\hat{x}(\alpha_i)$ are all distinct as they define the slopes of the different pieces of the piecewise linear function, we conclude that $r \leq n$. ∎

We finally get the following sufficient condition for the algorithm described in Sect. 3.2.2 to run in polynomial time.

Proposition 8. *If the functions v and $-w$ are supermodular and one of the following properties is satisfied:*

- *v or w takes a polynomial number of distinct values on B_n;*
- *v and w are both linear;*
- *v or w is monotone and the application $x \mapsto (v(x), w(x))$ is weakly bijective;*

then the Eisner and Severance algorithm computes the set X^+ in polynomial time.

Proof. Follows from Propositions 2, 6 and 7. ∎

3.3 Correctness of the New Algorithm

Let $X = \{x^0, x^1, \ldots, x^q\}$ and X^+ be the sets of points of B_n defined in Sect. 3.1. Let S^* be the set of optimal solutions of problem (CFP). Since $\min\limits_{x \in B_n} w(x) = w(x^0) < w(x^1) < \cdots < w(x^q) = \max\limits_{x \in B_n} w(x)$ by Lemma 3 and (12), for any $x^* \in S^*$ there must exist $k \in \{1, 2, \ldots, q\}$ such that $w(x^{k-1}) \leq w(x^*) \leq w(x^k)$.

The next result considers the case where $w(x^*)$ coincides with the bound of an interval $[w(x^{k-1}), w(x^k)]$, while Proposition 10 considers the case where $w(x^*)$ lies strictly in such an interval. We will assume in this section that φ and ρ are increasing.

Proposition 9. *Assume that φ is increasing and ρ is strictly positive, and let x^* be an optimal solution of problem (CFP). For any $k = 0, \ldots, q$ we have the implication*

$$w(x^*) = w(x^k) \quad \Rightarrow \quad x^k \text{ is an optimal solution of problem (CFP)}.$$

Proof. Assume that $w(x^*) = w(x^k)$ for some k. Observe first that $v(x^k) \geq v(x^*)$. Indeed, when $k = 0$, this follows from Lemma 3; when $k \geq 1$, $\Phi(\mu_k) = v(x^k) - \mu_k w(x^k)$ by (12), hence $v(x^k) - \mu_k w(x^k) \geq v(x^*) - \mu_k w(x^*)$. Since $w(x^*) = w(x^k)$, we conclude that $v(x^k) \geq v(x^*)$.

Now since φ is increasing and since $\rho(w(x))$ is strictly positive for all $x \in B_n$, we have easily

$$\frac{\varphi(v(x^k))}{\rho(w(x^k))} \geq \frac{\varphi(v(x^*))}{\rho(w(x^*))}$$

which shows that x^k is an optimal solution of problem (CFP). ∎

Proposition 10. *Assume that φ and ρ are increasing, that ρ is strictly positive and that $\max\limits_{x \in B_n} (\varphi \circ v)(x) \geq 0$ and let x^* be an optimal solution of problem (CFP). For any $k = 1, \ldots, q$ we have the implication*

$$w(x^{k-1}) < w(x^*) < w(x^k) \quad \Rightarrow \quad \begin{cases} v(x^{k-1}) < v(x^*) < v(x^k) \\ \text{or} \\ x^{k-1} \text{is an optimal solution of problem (CFP).} \end{cases}$$

Proof. Assume that $w(x^{k-1}) < w(x^*) < w(x^k)$ holds. Since ρ is increasing, we have

$$\rho(w(x^{k-1})) \leq \rho(w(x^*))$$

or, using the fact that ρ is strictly positive,

$$\frac{1}{\rho(w(x^{k-1}))} \geq \frac{1}{\rho(w(x^*))}. \tag{16}$$

We now show that $v(x^*) \leq v(x^{k-1})$ implies that x^{k-1} is an optimal solution of problem (CFP). Assume that $v(x^*) \leq v(x^{k-1})$. The fact that φ is increasing and the assumption on the sign of the optimal value imply that $0 \leq \varphi(v(x^*)) \leq \varphi(v(x^{k-1}))$. Combining with (16) yields $\frac{\varphi(v(x^{k-1}))}{\rho(w(x^{k-1}))} \geq \frac{\varphi(v(x^*))}{\rho(w(x^*))}$, which shows that x^{k-1} is an optimal solution of problem (CFP). We have thus concluded that either $v(x^*) > v(x^{k-1})$ or x^{k-1} is an optimal solution of problem (CFP).

In the following we assume that the former is true. Since $\Phi(\mu_k) = v(x^{k-1}) - \mu_k w(x^{k-1})$ by (12),

$$v(x^{k-1}) - \mu_k w(x^{k-1}) \geq v(x^*) - \mu_k w(x^*)$$
$$\Rightarrow \mu_k \left(w(x^*) - w(x^{k-1}) \right) \geq v(x^*) - v(x^{k-1}) > 0.$$

Since $w(x^*) > w(x^{k-1})$ we conclude that $\mu_k > 0$. Now since we have also $\Phi(\mu_k) = v(x^k) - \mu_k w(x^k)$,

$$v(x^k) - \mu_k w(x^k) \geq v(x^*) - \mu_k w(x^*)$$
$$\Rightarrow v(x^*) - v(x^k) \leq \mu_k \left(w(x^*) - w(x^k) \right) < 0.$$

Hence $v(x^{k-1}) < v(x^*) < v(x^k)$. ∎

Propositions 9 and 10 leave open the possibility that $x^* = x^k$ for some k such that $\mu_k < 0$. The next result shows that if that happens, then at least one x^ℓ with ℓ such that $\mu_\ell > 0$ is also an optimal solution of problem (CFP).

Proposition 11. *Assume that φ and ρ are increasing, that ρ is strictly positive and that $\max\limits_{x \in B_n} (\varphi \circ v)(x) \geq 0$. If $S^* \cap X \neq \emptyset$ then $S^* \cap X^+ \neq \emptyset$.*

Proof. Recall that by definition $\mu_1 > \mu_2 > \cdots > \mu_q$. If $\mu_q \geq 0$ we are done, so assume that $\mu_q < 0$. Define r to be such that $\mu_{r-1} \geq 0 > \mu_r$. Then we have $\mu_k < 0$ for all $k = r, \ldots, q$. Since $w(x^k) > w(x^{k-1})$ for all k and by (14), it follows that

$$v(x^k) < v(x^{k-1}), \qquad k = r, \ldots, q. \tag{17}$$

Now assume that x^t is an optimal solution of problem (CFP) with $r \leq t \leq q$. We will show that x^t is also an optimal solution of problem (CFP). Since the sequence $\{w(x^k)\}$ is strictly increasing and by (17)

$$v(x^t) < v(x^r)$$
$$w(x^t) > w(x^r).$$

Since φ and ρ are increasing and ρ is strictly positive we get

$$(\varphi \circ v)(x^t) \leq (\varphi \circ v)(x^r)$$

$$0 < \frac{1}{(\rho \circ w)(x^t)} \leq \frac{1}{(\rho \circ w)(x^r)}.$$

Hence, since $(\varphi \circ v)(x^t) \geq 0$,

$$\frac{(\varphi \circ v)(x^t)}{(\rho \circ w)(x^t)} \leq \frac{(\varphi \circ v)(x^r)}{(\rho \circ w)(x^r)}.$$

Therefore x^r is also an optimal solution of problem (CFP), and x^r belongs to X^+ as it defines $\Phi(t)$ over the interval $[0; \mu_{r-1}]$. ∎

The next result establishes a sufficient condition on φ and ρ for the existence of an optimal solution of problem (CFP) in the set X^+.

Proposition 12. *Assume that φ and ρ are increasing and that $\max\limits_{x \in B_n} \varphi(v(x)) \geq 0$. A sufficient condition for $S^* \cap X^+ \neq \emptyset$ is*

$$\left(\frac{t - v(x^{k-1})}{\varphi(t) - \varphi(v(x^{k-1}))} \right) \left(\frac{\rho(u) - \rho(w(x^{k-1}))}{u - w(x^{k-1})} \right) \left(\frac{\varphi(v(x^k)) - \varphi(t)}{v(x^k) - t} \right)$$

$$\times \left(\frac{w(x^k) - u}{\rho(w(x^k)) - \rho(u)} \right) \geq 1$$

$$\forall t : v(x^{k-1}) < t < v(x^k), \quad \forall u : w(x^{k-1}) < u < w(x^k) \tag{18}$$

Proof. We will assume that $S^* \cap X^+ = \emptyset$ and exhibit a couple (t, u) that violates (18).

Let $x^* \in S^*$. By Propositions 9 and 11, $w(x^*) \neq w(x^k)$ for all $k = 0, 1, \ldots, q$. Therefore since $w(x^0) = \min\limits_{x \in B_n} w(x)$ and $w(x^q) = \max\limits_{x \in B_n} w(x)$ by Lemma 3, there must exist some k such that $w(x^*) \in (w(x^{k-1}), w(x^k))$.

To simplify the notations, let $v^* = v(x^*)$, $w^* = w(x^*)$, $v_\ell = v(x^\ell)$ and $w_\ell = w(x^\ell)$ for $\ell \in \{k-1, k\}$. Since $x^{k-1} \notin S^*$ and by Propositions 10 and 11 we have

$$v_{k-1} < v^* < v_k. \tag{19}$$

By optimality of x^* and since $x^{k-1}, x^k \notin S^*$,

$$\frac{\varphi(v^*)}{\rho(w^*)} > \frac{\varphi(v_{k-1})}{\rho(w_{k-1})} \tag{20}$$

$$\frac{\varphi(v^*)}{\rho(w^*)} > \frac{\varphi(v_k)}{\rho(w_k)}. \tag{21}$$

Now by (21)

$$\varphi(v_k) - \varphi(v^*) < \frac{\varphi(v^*)}{\rho(w^*)} \left(\rho(w_k) - \rho(w^*) \right). \tag{22}$$

Since $v_k > v^*$ and φ is increasing, we cannot have $\varphi(v^*) = 0$ hence it follows from the assumptions that $\varphi(v^*) > 0$. Therefore the fact that φ is increasing implies that $\rho(w_k) - \rho(w^*) > 0$. Inequality (22) can then be written:

$$\frac{\varphi(v^*)}{\rho(w^*)} > \frac{\varphi(v_k) - \varphi(v^*)}{\rho(w_k) - \rho(w^*)}. \tag{23}$$

Similarly (20) yields

$$\varphi(v^*) - \varphi(v_{k-1}) > \frac{\varphi(v^*)}{\rho(w^*)} \left(\rho(w^*) - \rho(w_{k-1}) \right). \tag{24}$$

Since $w^* > w_{k-1}$, ρ is increasing and $\varphi(v^*) > 0$, it follows that $\varphi(v^*) - \varphi(v_{k-1}) > 0$. Hence (24) can be written

$$\frac{\rho(w^*)}{\varphi(v^*)} > \frac{\rho(w^*) - \rho(w_{k-1})}{\varphi(v^*) - \varphi(v_{k-1})}. \tag{25}$$

By Proposition 4

$$\frac{v^* - v_{k-1}}{w^* - w_{k-1}} \leq \frac{v_k - v^*}{w_k - w^*}$$

$$\Rightarrow 1 \geq \left(\frac{v^* - v_{k-1}}{w^* - w_{k-1}} \right) \left(\frac{w_k - w^*}{v_k - v^*} \right). \tag{26}$$

Since each factor in inequalities (23), (25) and (26) is nonnegative, we can multiply these two inequalities memberwise, hence

$$\left(\frac{v^* - v_{k-1}}{\varphi(v^*) - \varphi(v_{k-1})}\right)\left(\frac{\rho(w^*) - \rho(w_{k-1})}{w^* - w_{k-1}}\right)\left(\frac{\varphi(v_k) - \varphi(v^*)}{v_k - v^*}\right)\left(\frac{w_k - w^*}{\rho(w_k) - \rho(w^*)}\right) < 1.$$

We have shown that the assumption $S^* \cap X^+ = \emptyset$ implies that (18) is violated for $t = v^*$ and $u = w^*$. Hence (18) implies that $S^* \cap X^+ \neq \emptyset$.   ∎

In order to derive a simpler condition on φ and ρ that implies the sufficient condition of Proposition 12, we need the following Lemma.

Lemma 4. *Let h be a convex function over an interval $[a, b]$. Then*

$$\frac{h(t) - h(a)}{t - a} \leq \frac{h(b) - h(t)}{b - t} \qquad \forall t : a < t < b. \tag{27}$$

Proposition 13. *Assume that φ and ρ are increasing. If in addition φ is convex and ρ is concave, then the sufficient condition (18) for $S^* \cap X^+ \neq \emptyset$ is satisfied.*

Proof. Let (t, u) such that $v(x^{k-1}) < t < v(x^k)$ and $w(x^{k-1}) < u < w(x^k)$. If φ is convex and ρ is concave, we have by Lemma 4

$$\frac{\varphi(t) - \varphi(v(x^{k-1}))}{t - v(x^{k-1})} \leq \frac{\varphi(v(x^k)) - \varphi(t)}{v(x^k) - t} \tag{28}$$

$$\frac{\rho(u) - \rho(w(x^{k-1}))}{u - w(x^{k-1})} \geq \frac{\rho(w(x^k)) - \rho(u)}{w(x^k) - u}. \tag{29}$$

Since φ and ρ are increasing, each ratio in these two inequalities is strictly positive. We then easily derive (18).   ∎

To conclude this section we give an example that shows that the assumption of convexity is not necessary for (27) to be satisfied in Lemma 4. This in particular implies that the sufficient condition of Proposition 12 can be satisfied even when φ is not convex and ρ is not concave.

Example 1. Consider the following piecewise linear function defined on the interval $[a, b] = [0; 5]$.

$$h(t) = \begin{cases} t & \text{if } t \in [0; 2] \\ 6t - 10 & \text{if } t \in [2; 3] \\ 2t + 2 & \text{if } t \in [3; 5]. \end{cases}$$

The graph of this function is represented in Fig. 2. Clearly $h(t)$ is not convex. It can be verified that for any position of the point T on the curve, the slope of the segment AT is smaller than the slope of the segment TB. This is exactly what is expressed by (27).
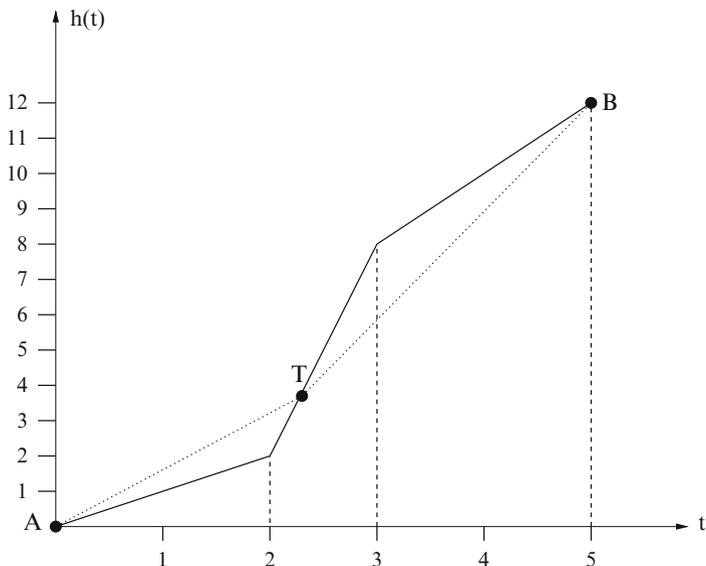
Fig. 2 Geometrical interpretation of the inequality of Lemma 4

3.4 Polynomial Solvable Instances

In this section we prove Theorems 1 and 2.

Proof of Theorem 1   Follows from Propositions 8, 12 and 13.   ∎

Proof of Theorem 2   Assume that the condition (C4') is satisfied, i.e., $(\varphi \circ v)(x) < 0$ for all $x \in B_n$. We observe that

$$\max_{x \in B_n} \frac{(\varphi \circ v)(x)}{(\rho \circ w)(x)} \Leftrightarrow \min_{x \in B_n} \frac{-(\varphi \circ v)(x)}{(\rho \circ w)(x)} \Leftrightarrow \max_{x \in B_n} \frac{(\rho \circ w)(x)}{-(\varphi \circ v)(x)}$$

$$\Leftrightarrow \max_{x \in B_n} \frac{(\varphi' \circ v')(x)}{(\rho' \circ w')(x)}$$

with $\varphi'(t) = \rho(-t)$, $\rho'(t) = -\varphi(-t)$ for $t \in \mathbb{R}$ and $v'(x) = -w(x)$ and $w'(x) = -v(x)$ for all $x \in B_n$. The result then follows from Theorem 1 applied to the problem $\max_{x \in B_n} \frac{(\varphi' \circ v')(x)}{(\rho' \circ w')(x)}$ and by converting the conditions on φ', ρ', v' and w' to conditions on φ, ρ, v and w.   ∎

The equivalent of Proposition 12 for the instances satisfying condition (C4') is:

Proposition 14. *Assume that φ is increasing, that ρ is decreasing and that $\max_{x \in B_n} \varphi(v(x)) < 0$. A sufficient condition for $S^* \cap X^+ \neq \emptyset$ is*

$$\left(\frac{t - v(x^{k-1})}{\varphi(t) - \varphi(v(x^{k-1}))}\right)\left(\frac{\rho(u) - \rho(w(x^{k-1}))}{u - w(x^{k-1})}\right)\left(\frac{\varphi(v(x^k)) - \varphi(t)}{v(x^k) - t}\right)\left(\frac{w(x^k) - u}{\rho(w(x^k)) - \rho(u)}\right) \geq 1$$

$$\forall t : v(x^{k-1}) < t < v(x^k), \quad \forall u : w(x^{k-1}) < u < w(x^k). \tag{30}$$

In particular, the inequality (30) coincides with (18).

We terminate with some remarks:

- In view of Proposition 5, it is natural to assume that the time $T(n)$ to compute the set X^+ is of the order of $|X^+|V(n)$ where $V(n)$ is the time needed to solve the problem PARAM(λ) for some given λ. Actually this can often be done in the order of $V(n)$ by using a different algorithm, see, for example, Sect. 2.2.3.
- The complexity of our algorithm is essentially determined by the computation of the set X^+, which depends only upon the functions v and w. The lowest complexities will be obtained when v and w can be represented by low degree multilinear polynomials. For example, if v and w are both linear, it is possible to compute the set X^+ in $O(n \log n)$ by representing implicitly the elements of X^+. If v and $-w$ are quadratic supermodular functions and some monotonicity property holds, the algorithm of Gallo, Grigoriadis and Tarjan can compute the set X^+ in $O(n^6)$, see Sects. 2.2.2 and 2.2.3.
- Conversely, since the functions φ and ρ are used only to identify the best point in X^+, they can have less attractive properties, for example—they could be non-rational as illustrated by the additive clustering problem, see Sect. 4.3.
- If an instance (φ, ρ, v, w) satisfies the assumptions of Theorem 1, one must in particular have that φ is increasing (condition (C5)) and convex (condition (C6)) and v is supermodular (condition (C7)). By part a) of Proposition 1, only the absence of the monotonicity property for v prevents us from concluding that $\varphi \circ v$ is supermodular (and monotone) on B_n. Similarly, by part d) of Proposition 1, only the absence of the monotonicity property for w prevents us from concluding that $\rho \circ w$ is submodular (and monotone) on B_n. If both v and w were monotone, the assumptions of Theorem 1 are thus close to allow a polynomial algorithm for (CFP) via the direct use of Dinkelbach's algorithm: we would need in addition that φ and ρ are rational functions, and a kind of strict monotonicity for either $\varphi \circ v$ or $\rho \circ w$ but these additional assumptions are relatively minor with respect to those of Theorem 1. One can even notice that monotonicity of v and w is present in the assumption of Theorem 1 (condition (C8c)). This suggests that our new class of polynomially solvable instances extends only marginally the known class of polynomially solvable instances of the fractional programming problem.

  This is not exactly true, because our results do not require either v or w to be monotone: condition (C8) could be satisfied through either (C8a) or (C8b). And even if condition (C8c) is satisfied, only one of the functions v or w need to be monotone. In other words $\varphi \circ v$ and/or $-\rho \circ w$ might not have the supermodularity property when the assumptions of Theorem 1 or 2 are satisfied, in which case we do not know how to solve efficiently the auxiliary problem that arises in Dinkelbach's method.

4 Application to an Additive Clustering Problem

In this section we show how the results obtained up to now can be used to derive a polynomial algorithm for a problem arising in additive clustering. We start by introducing additive clustering in Sect. 4.1. In Sect. 4.2 we reformulate a particular problem arising in this area as a 0–1 fractional programming problem. An $O(n^5)$ algorithm to solve this later problem is then described in Sect. 4.3.

4.1 Additive Clustering

The additive clustering (ADCLUS) model has been introduced by Shepard and Arabie [50], Arabie and Carroll [2] in the context of cognitive modeling.[1] This model assumes that the similarity between two objects, measured by a nonnegative number, is additively caused by the properties (also called *features*) that these two objects share. With each property we can associate a cluster, which contains all objects that have this property. Furthermore with each cluster we associate a positive weight representing the importance of the corresponding property. The similarity predicted by the model for a pair of objects is then defined as the sum of the weights of the clusters to which both objects belong (note that clusters can overlap). An ADCLUS model is characterized by a set of clusters, together with their weights. Given a similarity matrix obtained typically by some experiments, the additive clustering problem consists in constructing a model that explains as much as possible of the given similarity matrix, under the restriction that the model's complexity is limited (if we do not restrict the complexity of the model, we can reconstruct perfectly the similarity matrix with $O(n^2)$ clusters, see Shepard and Arabie [50, p. 98]). We will assume here that the complexity of the model is measured by the number of clusters, see, e.g., Lee and Navarro [38] and references therein for more elaborated measures of the complexity. In other words, limiting the complexity of the model amounts to setting an upper bound on the number of clusters used to construct the approximate similarity matrix. Many authors have developed algorithms to fit this model (or variants or generalizations of it) with this definition of the complexity, see, e.g., [4, 7, 11, 15, 37, 40].

The mathematical formulation of the additive clustering problem with *m* clusters is the following:

$$\text{ADCLUS}(m) \quad \min \ f(x, w) = \sum_{i<j} \left(s_{ij} - \sum_{k=1}^{m} w_k x_i^k x_j^k \right)^2$$

---

[1] A similar model was developed independently and at the same time in the former USSR; see Mirkin [40] and the references therein.

$$\text{s.t.} \quad w_k \geq 0 \qquad k = 1, \ldots, m$$
$$x_i^k \in \{0, 1\} \qquad k = 1, \ldots, m; \ i = 1, \ldots, n$$

where $S = (s_{ij})$ is a $n \times n$ symmetrical nonnegative matrix.

4.2 The Additive Clustering Problem with One Cluster

In an attempt to assess the complexity of problem ADCLUS(m) we studied the version with one cluster:

$$\text{ADCLUS(1)} \quad \min \ f(x, w) = \sum_{i<j} \left(s_{ij} - w x_i x_j \right)^2$$

$$\text{s.t.} \quad w \geq 0$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, n.$$

Note that the cluster must have at least two elements in order to define a non-null reconstructed matrix. This motivates the introduction of the set T:

$$T = \left\{ x \in B_n : \sum_{i=1}^{n} x_i \geq 2 \right\}. \tag{31}$$

We now reformulate problem ADCLUS(1) as a 0–1 fractional programming problem.

Proposition 15. *Problem ADCLUS(1) is equivalent to*

$$\text{ADCLUS}'(1) \qquad \max_{x \in T} \ g(x) = \frac{\left(\sum_{i<j} s_{ij} x_i x_j \right)^2}{\left(\sum_{i=1}^{n} x_i \right) \left(\sum_{i=1}^{n} x_i - 1 \right)}.$$

In particular if x^* is an optimal solution of problem ADCLUS$'$(1) *then* (x^*, w^*) *is an optimal solution of problem ADCLUS(1) with*

$$w^* = \frac{\sum_{i<j} s_{ij} x_i^* x_j^*}{\sum_{i<j} x_i^* x_j^*}.$$

Proof. See Hansen et al. [32].   ∎

4.3   A Polynomial Algorithm

We first explain how a straightforward application of the results of this paper lead to an $O(n^5)$ algorithm to solve problem ADCLUS$'(1)$. Then we show that with a little additional effort an $O(n^4)$ algorithm can be obtained.

Problem ADCLUS$'(1)$ is not a (CFP) problem because its feasible set is a strict subset of B_n. However note that problem ADCLUS$'(1)$ can be reduced to a polynomial number of problems (CFP) of size $n-2$, each problem being obtained from ADCLUS$'(1)$ by fixing two variables to 1. By renumbering the variables if necessary, the general form of such a problem is

$$\text{ADCLUS}'_2(1) \quad \max_{x \in B_{n-2}} \tilde{g}_2(x) = \frac{\left(\sum\limits_{i<j} \tilde{s}_{ij} x_i x_j + \sum\limits_{i=1}^{n-2} \tilde{s}_{ii} x_i + \tilde{c} \right)^2}{\left(\sum\limits_{i=1}^{n-2} x_i + 2 \right) \left(\sum\limits_{i=1}^{n-2} x_i + 1 \right)}$$

where $\tilde{S}$ is a $(n-2) \times (n-2)$ symmetrical matrix with nonnegative entries and $\tilde{c}$ is a nonnegative constant. Clearly solving problem ADCLUS$'(1)$ in polynomial time is equivalent to solving problem ADCLUS$'_2(1)$ in polynomial time.

Unfortunately problem ADCLUS$'_2(1)$ does not satisfy the assumptions of Theorem 1 or 2, so we consider instead the problem:

$$\text{ADCLUS}''_2(1) \quad \max_{x \in B_{n-2}} \tilde{h}_2(x) = \frac{\sum\limits_{i<j} \tilde{s}_{ij} x_i x_j + \sum\limits_{i=1}^{n-2} \tilde{s}_{ii} x_i + \tilde{c}}{\sqrt{\left(\sum\limits_{i=1}^{n-2} x_i + 2 \right) \left(\sum\limits_{i=1}^{n-2} x_i + 1 \right)}}$$

Since the matrix S is assumed to be nonnegative, the numerator $\sum\limits_{i<j} \tilde{s}_{ij} x_i x_j + \sum\limits_{i=1}^{n-2} \tilde{s}_{ii} x_i + \tilde{c}$ is nonnegative for all $x \in B_{n-2}$, hence problem ADCLUS$'_2(1)$ is equivalent to problem ADCLUS$''_2(1)$. Now problem ADCLUS$''_2(1)$ is a problem (CFP) with $\varphi = \tilde{\varphi}$, $\rho = \tilde{\rho}$, $v = \tilde{v}$ and $w = \tilde{w}$ where

$$\tilde{\varphi}(t) = t$$

$$\tilde{\rho}(t) = \sqrt{(t+1)(t+2)}$$

$$\tilde{v}(x) = \sum\limits_{i<j} \tilde{s}_{ij} x_i x_j + \sum\limits_{i=1}^{n-2} \tilde{s}_{ii} x_i + \tilde{c}$$

$$\tilde{w}(x) = \sum\limits_{i=1}^{n-2} x_i$$

and it can be verified that $(\tilde{\varphi}, \tilde{\rho}, \tilde{v}, \tilde{w})$ satisfies the conditions of Theorem 1.

The corresponding problem PARAM(λ) can be reformulated as a parametric minimum cut problem in a network with n vertices and $O(n^2)$ arcs, which can be solved by the Gallo, Grigoriadis and Gallo algorithm, see Sect. 2.2.3. Hence the time needed to compute the set X^+ is $T(n) = O(n^3)$. Step 2 of the HM_CFP algorithm consists in identifying the best feasible point among the points computed in Step 1. In order to avoid problems with the square-root function, we evaluate $\left(\tilde{h}_2(x)\right)^2$ instead of $\tilde{h}_2(x)$. An evaluation costs $O(n^2)$ time, hence the complexity of Step 2 is in $O(n^3)$. Thus the overall complexity for solving ADCLUS$'_2$(1) is $O(n^3)$. Since we need to solve $O(n^2)$ of such problems to solve problem ADCLUS(1), the complexity of this latter problem is $O(n^5)$. Hence we have shown:

Proposition 16. *There exists an $O(n^5)$ algorithm to solve problem ADCLUS(1).*

Proposition 16 suffices to show that problem ADCLUS(1) can be solved in polynomial time. The question is now whether we can lower the order of the complexity. A little attention shows that we can obtain an $O(n^4)$ algorithm by working with problems obtained by fixing one variable to 1 rather than two. The resulting problems are problems (CFP) with B_n replaced by $B_n \setminus \{(0)\}$. By looking at the proofs, we observe that the analysis made for the unconstrained case remains valid, hence yielding an $O(n^4)$ algorithm. We believe that this complexity can still be improved but let this be for further research. Let us only note that working directly with problem ADCLUS$'$(1) by setting $\rho(t) = \sqrt{t(t-1)}$ does not work, as shown by the following example.

Example 2. Let $n = 8$ and consider the following instance of ADCLUS$'$(1) where we maximize the square-root of the original objective function:

$$\max \quad \frac{18x_1x_5 + 15x_1x_6 + 8x_2x_7 + 16x_3x_5 + 22x_4x_6 + 15x_6x_8}{\sqrt{\left(\sum_{i=1}^{8} x_i\right)\left(\sum_{i=1}^{8} x_i - 1\right)}}$$

$$\text{s.t.} \quad \sum_{i=1}^{8} x_i \geq 2$$

$$x_i \in \{0; 1\} \qquad i = 1, \ldots, 8.$$

Then

$$\Phi(\lambda) = \max_{x \in B_8} \left\{18x_1x_5 + 15x_1x_6 + 8x_2x_7 + 16x_3x_5 + 22x_4x_6 + 15x_6x_8 - \lambda\left(\sum_{i=1}^{8} x_i\right)\right\}.$$

Applying the Eisner and Severance algorithm yields

$$\Phi(\lambda) = \begin{cases} 94 - 8\lambda & 0 \le \lambda \le 4 \\ 86 - 6\lambda & 4 \le \lambda \le 14.33 \\ 0 & 14.33 \le \lambda \end{cases}$$

with $X^+ = \{(11111111), (10111101), (00000000)\}$. The best point among these three can easily be shown to be (10111101) with a value for the (squared-rooted) objective function equal approximately to 11.5962. However the optimal solution of problem ADCLUS$'(1)$ is $x^* = (10011100)$ with an (square-rooted) objective value of approximately 15.8771.

As observed in Sect. 3.4, all instances of problem (CFP) solvable by our method do not satisfy the property that $\varphi \circ v$ and $-\rho \circ w$ are monotone supermodular functions, but it turns out that this is true for problem ADCLUS$''_2(1)$. Thus it seems that Dinkelbach's algorithm would be polynomial too. In fact the only assumption of Proposition 3 that is not satisfied is that $x \mapsto \sqrt{\left(\sum_{i=1}^{n-2} x_i + 2\right)\left(\sum_{i=1}^{n-2} x_i + 1\right)}$ is a rational function. We now discuss why the non-rationality of this function is a serious difficulty if we want to show that Dinkelbach's algorithm is polynomial. Recall that Dinkelbach's algorithm requires the solution of the following auxiliary problem

$$\text{FPaux}_{1/2}(\lambda) \quad \max_{x \in B_{n-2}} h_{\lambda,1/2}(x) = \sum_{i<j} \tilde{s}_{ij} x_i x_j + \sum_{i=1}^{n-2} \tilde{s}_{ii} x_i + \tilde{c} - \lambda \sqrt{\left(\sum_{i=1}^{n-2} x_i + 2\right)\left(\sum_{i=1}^{n-2} x_i + 1\right)}$$

for some λ. This auxiliary problem is solved typically by one of the SFM algorithms mentioned in Sect. 2.1.3. Such an algorithm requires from time to time the evaluation of the objective function $h_{\lambda,1/2}$ at some points of B_{n-2}. The question that arises is what is the precision needed for λ and for the evaluation of the objective function value at a point of B_{n-2} in order to guarantee that the solution returned by the SFM algorithm is indeed optimal (note that from the point of view of the correctness of Dinkelbach's algorithm, the optimality of the solution returned by the SFM algorithm is required only at the last iteration; however, if the SFM algorithm returns non-optimal solutions at other iterations, the number of iterations might not anymore be polynomial)? Let us approach this question differently. It is easy to see that the λ_k are of the form $b\sqrt{c}$ where b is a rational and c is an integer of the form $(t + 1)(t + 2)$ with $t \in \{0, \dots, n\}$, hence the value of the objective function $h_{\lambda,1/2}(x)$ can be written as $a' + b'\sqrt{c'}$ where c' is a square-free integer less than $n^2(n + 1)^2$, in particular the objective function value can be outputed exactly by specifying the triple (a', b', c'). Now these values are likely to be added, subtracted and compared together by the SFM algorithm (if we restrict ourselves to a fully strongly combinatorial algorithm we do not have to care about multiplication and division). To represent exactly a sum of such numbers, we can introduce a basis that spans the set of numbers $\bigcup_{1 \le c' \le n^2(n+1)^2} \{\sqrt{c'}\}$. The question is

now: what is the complexity of comparing two numbers written in this basis? Up to now, no polynomial algorithm is known for this comparison problem [6, 44]. It is not impossible that a finer analysis, taking into account what operations exactly are done on these numbers by the SFM algorithm as well as the structure of the numbers forming the basis, would yield a polynomial algorithm by this approach, but this might not be easy. Even if it is possible, the best complexity we could hope for the problem ADCLUS(1) by applying Dinkelbach's algorithm in conjunction with a generic SFM algorithm is $O(n^9)$ if we use Orlin's strongly polynomial algorithm (that uses multiplication and division) or $O(n^{12} \log^2 n)$ if we use Iwata's fully strongly combinatorial algorithm, which is much higher than the $O(n^5)$ algorithm described in this section.

5 Discussions: Limitations and Extensions

In the previous section we have shown that the problem (CFP) can be solved in polynomial time if the functions φ, ρ, v and w satisfy the conditions of Theorem 1 or 2. In this section we discuss various extensions (or impossibility of them) of these polynomial solvable classes. Using NP-hardness results, we start by arguing in Sect. 5.1 that some of the assumptions of Theorem 1 or 2 can hardly be relaxed. We then discuss the extension of our results to minimization problems (Sect. 5.2), maximization of product of two functions (Sect. 5.3) and constrained problems (Sect. 5.4).

5.1 Limitations

In this section we show that unless $NP = P$ we generally cannot hope to solve problem (CFP) in polynomial time if we modify one assumption while keeping the other assumptions unchanged.

- It is not possible to replace the assumption "v is supermodular" by "v is submodular", while keeping all others assumptions unchanged: indeed by choosing $\varphi(t) = t$ and $\rho(t) = 1$, the problem (CFP) would become equivalent to maximizing a submodular function, which is known to be NP-hard. A similar restriction holds for the assumption "w is submodular".
- By the equivalence of the assumptions "φ is increasing and v is supermodular" and "φ is decreasing and v is submodular" (see Sect. 1), it follows that it is not possible to replace the assumption "φ is increasing" by "φ is decreasing". A similar statement could be made for the function ρ.
- It is not possible to remove the assumption that $\rho(t) > 0$ for all t. This follows from the following result of Hansen et al. [31]:

Proposition 17. *The problem*

$$\max_{x \in B_n} \quad \frac{a_0 + \sum\limits_{j=1}^{n} a_j x_j}{b_0 + \sum\limits_{j=1}^{n} b_j x_j}$$

is NP-hard unless the denominator is of the same sign for all $x \in B_n$.

(If the denominator is of the same sign for all $x \in B_n$, the above problem can be solved in linear time as shown by Hansen et al. [31]).

- The following observation involves two assumptions: it is not possible to replace the assumption "φ is convex" by the assumption "φ is concave", while at the same time removing the assumption that φ is increasing. To show this, we need to introduce the following well-known NP-hard problem SUBSET SUM (Garey and Johnson [23]):

Input: n positive integers $s_1, s_2, \ldots, s_n$; an integer S.
Question: does there exist a subset I of the index set $\{1, 2, \ldots, n\}$ such that $\sum_{i \in I} s_i = S$?

We define $\varphi(t) = -|t|$, $\rho(t) = 1$ and $v(x) = \sum_{i \in I} s_i x_i - S$. Clearly the answer to SUBSET-SUM is yes if and only if the maximum of problem (CFP) is 0. The function φ is concave for all t and is increasing for $t < 0$ and decreasing for $t \geq 0$. A similar observation can be made for function ρ.

5.2 Minimization Problems

Consider the minimization version of problem (CFP):

$$(\text{CFPmin}) \qquad \min_{x \in B_n} \frac{(\varphi \circ v)(x)}{(\rho \circ w)(x)}.$$

If $\varphi \circ v$ is strictly positive on B_n, we can use the equivalence

$$\min_{x \in B_n} \frac{(\varphi \circ v)(x)}{(\rho \circ w)(x)} \qquad \Leftrightarrow \qquad \max_{x \in B_n} \frac{(\rho \circ w)(x)}{(\varphi \circ v)(x)} \qquad (32)$$

to derive sufficient conditions on (φ, ρ, v, w) for polynomial solvability of problem (CFPmin) from Theorem 1. However if $\varphi \circ v$ can take positive and negative values on B_n, equivalence (32) is not anymore true.

A similar analysis to the one done for the maximization problem results in the polynomial solvable classes described by Fig. 3. Figure 3 must be read as follows:

1. $(\rho \circ w)(x) > 0$ for $x \in B_n$;
2. it is possible to evaluate the objective function in polynomial time;
3. v and w take integral values on B_n;
4. v and $-w$ are submodular;
5. one of the following conditions is satisfied:

   - v or w takes a polynomial number of distinct values on B_n;
   - v and w are both linear;
   - v or w is monotone and the application $x \mapsto (v(x), w(x))$ is weakly bijective;

6. $\min\limits_{x \in B_n} (\varphi \circ v)(x) \le 0$;	9. $(\varphi \circ v)(x) > 0$ for all $x \in B_n$;
7. φ and $-\rho$ are increasing;	10. φ and ρ are increasing;
8. $-\varphi$ and $-\rho$ are convex;	11. $-\varphi$ and ρ are convex.

Fig. 3 Description of the polynomial solvable classes for the minimization problem

1. $(\varphi_1 \circ v_1)(x) > 0$ for $x \in B_n$;
2. it is possible to evaluate the objective function in polynomial time;
3. v_1 and v_2 take integral values on B_n;
4. v_1 and v_2 are supermodular;
5. one of the following conditions is satisfied:

   - v_1 or v_2 takes a polynomial number of distinct values on B_n;
   - v_1 and v_2 are both linear;
   - v_1 or v_2 is monotone and the application $x \mapsto (v_1(x), v_2(x))$ is weakly bijective;

6. $\max\limits_{x \in B_n} (\varphi_2 \circ v_2)(x) \ge 0$;	9. $(\varphi_2 \circ v_2)(x) < 0$ for all $x \in B_n$;
7. φ_1 and φ_2 are increasing;	10. φ_1 and $-\varphi_2$ are increasing;
8. φ_1 and φ_2 are convex;	11. φ_1 and $-\varphi_2$ are convex.

Fig. 4 Description of the polynomial solvable classes for the problem of product maximization

if an instance of problem (CFPmin) satisfies the conditions 1–8 or satisfies the conditions 1–5 and 9–11, then the instance can be solved in polynomial time.

5.3 Maximization of the Product of Two Composed Functions

Consider the function $\sigma(t) = \frac{1}{\rho(-t)}$: if ρ is increasing and concave, function σ is increasing and convex. Problem (CFP) can then be reformulated as the maximization of the product of two functions:

$$\max_{x \in B_n} \quad \left((\varphi_1 \circ v_1)(x) \right) \left((\varphi_2 \circ v_2)(x) \right). \tag{33}$$

Figure 4 expresses the conditions of Theorems 1 and 2 in this new setting. Note that since (φ_1, v_1) and (φ_2, v_2) play a symmetrical role in (33), the first assumption in Fig. 4 could be replaced by $(\varphi_2 \circ v_2)(x) > 0$ for $x \in B_n$. However assumptions 6, 9, 10 and 11 should be modified accordingly.

We mention that the case where φ_1 and φ_2 are the identity functions and v_1 and v_2 are linear functions was studied by Hammer et al. [30].

5.4 Constrained Problems

In this last subsection we consider the constrained problem obtained from (CFP) by replacing the set B_n by a strict subset $T \subset B_n$. It can be verified that the sufficient condition of Proposition 12 remains valid. As soon as submodularity is involved, however, we usually need that T is a *sublattice* of B_n. A *sublattice* of B_n is a set T such that the following implication holds

$$x, y \in T \Rightarrow x \vee y \in T \text{ and } x \wedge y \in T.$$

Most of the algorithms for supermodular maximization can be modified to optimize over a sublattice T without increase in the complexity, see McCormick [39]. When T is not a sublattice but can be expressed as the union of a polynomial number of sublattices, then it is possible to solve the constrained problem in polynomial time by running a supermodular maximization algorithm on each sublattice of the union and take the best answer. This is, for example, the case when $T = B_n \setminus \{(0), (1)\}$. In that case maximizing over T can be done via $O(n)$ calls to a SFM algorithm. See again McCormick [39] and also Goemans and Ramakrishnan [24].

When $T = T_{\rho_1,\rho_2}$ with $T_{\rho_1,\rho_2} = \{x \in B_n : \rho_1 \leq \sum_{i=1}^{n} x_i \leq n - \rho_2\}$ where ρ_1 and ρ_2 are fixed integers, it is possible to solve the constrained version of problem (CFP) in polynomial time by reducing it to $\binom{n}{\rho_1}\binom{n}{\rho_2}$ unconstrained submodular maximization problems with $n - \rho_1 - \rho_2$ variables, obtained by considering all possible ways to fix ρ_1 variables to 1 and ρ_2 variables to 0. We gave an illustration of this technique for $\rho_1 = 2$ and $\rho_2 = 0$ in Sect. 4.3. Note that B_n is equal to $T_{0,0}$.

6 Conclusion

We have presented a class of 0–1 fractional programming problems that are solvable in polynomial time. A nice particularity of the algorithm is that the candidate solution set is defined by only two of the four functions defining the objective function, which allow for low complexity if these two (supermodular) functions have a low degree representation (linear, quadratic, etc.). On the other hand, the two other functions may be more complicated, possibly non-rational, provided that their value can be evaluated and compared in polynomial time.

A lot of work remains to be done. On the practical side, it would be of course interesting to find real application problems where this approach can yield a polynomial algorithm. The additive clustering problem used to illustrate this method is a potential candidate but much work is needed to pass from 1 cluster (as illustrated in this paper) to m clusters. More generally, the question of the complexity of this problem for fixed m or when m is part of the input remains open.

On the theoretical side, several questions seem to worth of further study.

- The simplest problem (CFP) that is not fully understood occurs when φ and ρ are the identity functions, v is a quadratic supermodular function and w is a linear function, strictly positive on B_n. If neither v nor w is monotone, and each function takes more than a polynomial number of distinct values on B_n, there is no guarantee that the function Φ will have a polynomial number of breakpoints. Is it possible to either prove that the number of breakpoints will always be polynomial or to construct an example with a super-polynomial number of breakpoints? Carstensen [8], building on a result from Zadeh [56], proves that for any n there exists a parametric minimum cut problem on a graph G_n with $2n + 2$ nodes and $n^2 + n + 2$ arcs that has an exponential number of breakpoints. Unfortunately this network has some arcs with negative capacity, and thus does not seem to be usable to answer the above question.
- Except when one of the functions v or w takes a polynomial number of distinct values, the size of the set X^+ is always $O(n)$. This is mostly related to what we called the Monotone Optimal Solution Property. Does there exist problems where the size is larger, for example, $O(n \log n)$ or $O(n^2)$?
- We have shown in Sect. 5.3 that problem (CFP) can be reformulated as the maximization of the product of two functions of supermodular functions. Can we identify nontrivial polynomially solvable classes of the maximization of the product of p functions of supermodular functions, with $p > 2$?
- Can we find a relation between Dinkelbach's algorithm applied to problem (CFP) and Dinkelbach's algorithm applied to problem $\max\limits_{x \in B_n} \frac{v(x)}{w(x)}$? More precisely, given a guess $\lambda_0 = \frac{(\varphi \circ v)(x^0)}{(\rho \circ w)(x^0)}$ for the optimal value of problem (CFP), can we deduce a λ_0' such that solving problem PARAM(λ_0') will yield an optimal solution $x^1 \in B_n$ that is guaranteed to satisfy $\frac{(\varphi \circ v)(x^1)}{(\rho \circ w)(x^1)} > \lambda_0$ when some termination criteria is not satisfied? One of the difficulties in attacking this question is that the functions v and w can be defined up to an additive constant, resulting in an infinite family of problems PARAM(λ).

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs (1993)
2. Arabie, P., Carroll, J.D.: MAPCLUS: a mathematical programming approach to fitting the ADCLUS model. Psychometrika **45**(2), 211–235 (1980)
3. Balinski, M.L.: On a selection problem. Manag. Sci. **17**, 230–231 (1970)
4. Berge, J.M.F.T., Kiers, H.A.L.: A comparison of two methods for fitting the INDCLUS model. J. Classif. **22**(2), 273–286 (2005)
5. Billionnet, A., Minoux, M.: Maximizing a supermodular pseudoboolean function: a polynomial algorithm for supermodular cubic functions. Discrete Appl. Math. **12**, 1–11 (1985)
6. Blömer, J.: Computing sums of radicals in polynomial time. In: 32nd Annual Symposium on Foundations of Computer Science, San Juan, PR, 1991, pp. 670–677. IEEE Computer Society Press, Los Alamitos (1991)

7. Carroll, J.D., Arabie, P.: INDCLUS: an individual differences generalization of the ADCLUS model and the MAPCLUS algorithm. Psychometrika **48**, 157–169 (1983)
8. Carstensen, P.J.: Complexity of some parametric integer and network programming problems. Math. Program. **26**(1), 64–75 (1983)
9. Chang, C.-T.: On the polynomial mixed 0-1 fractional programming problems. Eur. J. Oper. Res. **131**(1), 224–227 (2001)
10. Charnes, A., Cooper, W.W.: Programming with linear fractional functionals. Naval Res. Logist. Q. **9**, 181–186 (1962)
11. Chaturvedi, A., Carroll, J.D.: An alternating combinatorial optimization approach to fitting the INDCLUS and generalized INDCLUS models. J. Classif. **11**, 155–170 (1994)
12. Correa, J.R., Fernandes, C.G., Wakabayashi, Y.: Approximating a class of combinatorial problems with rational objective function. Math. Program. **124**(1–2, Ser. B), 255–269 (2010)
13. Craven, B.D.: Fractional Programming. Helderman Verlag, Berlin (1988)
14. Cunningham, W.H.: On submodular function minimization. Combinatorica **5**(3), 185–192 (1985)
15. Desarbo, W.S.: GENNCLUS: new models for general nonhierarchical clustering analysis. Psychometrika **47**(4), 449–475 (1982)
16. Dinkelbach, W.: On nonlinear fractional programming. Manag. Sci. **13**, 492–498 (1967)
17. Eisner, M.J., Severance, D.G.: Mathematical techniques for efficient record segmentation in large shared databases. J. Assoc. Comput. Mach. **23**(4), 619–635 (1976)
18. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. SIAM J. Comput. **40**(4), 1133–1153 (2011)
19. Frenk, J.B.G., Schaible, S.: Fractional programming. In: Handbook of Generalized Convexity and Generalized Monotonicity. Nonconvex Optimization and Its Applications, vol. 76, pp. 335–386. Springer, New York (2005)
20. Frenk, J.B.G., Schaible, S.: Fractional programming. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, pp. 1080–1091. Springer, Berlin (2009)
21. Gallo, G., Simeone, B.: On the supermodular knapsack problem. Math. Program. **45**(2, Ser. B), 295–309 (1989)
22. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**(1), 30–55 (1989)
23. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman, San Francisco (1979)
24. Goemans, M.X., Ramakrishnan, V.S.: Minimizing submodular functions over families of sets. Combinatorica **15**(4), 499–513 (1995)
25. Granot, F., McCormick, S.T., Queyranne, M., Tardella, F.: Structural and algorithmic properties for parametric minimum cuts. Math. Program. **135**(1–2, Ser. A), 337–367 (2012)
26. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**(2), 169–197 (1981)
27. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Algorithms and Combinatorics: Study and Research Texts, vol. 2. Springer, Berlin (1988)
28. Gusfield, D.M.: Sensitivity analysis for combinatorial optimization. Ph.D. thesis, University of California, Berkeley (1980)
29. Hammer, P.L., Rudeanu, S.: Boolean Methods in Operations Research and Related Areas. Springer, Berlin (1968)
30. Hammer, P.L., Hansen, P., Pardalos, P.M., Rader, D.J.: Maximizing the product of two linear functions in $0 - 1$ variables. Optimization **51**(3), 511–537 (2002)
31. Hansen, P., Poggi de Aragão, M.V., Ribeiro, C.C.: Hyperbolic 0-1 programming and query optimization in information retrieval. Math. Program. **52**(2, Ser. B), 255–263 (1991)
32. Hansen, P., Jaumard, B., Meyer, C.: Exact sequential algorithms for additive clustering. Technical Report G-2000-06, GERAD (March 2000)
33. Hochbaum, D.S.: Polynomial time algorithms for ratio regions and a variant of normalized cut. IEEE Trans. Pattern Anal. Mach. Intell. **32**(5), 889–898 (2010)

34. Ivănescu (Hammer), P.L.: Some network flow problems solved with pseudo-Boolean programming. Oper. Res. **13**, 388–399 (1965)
35. Iwata, S.: A faster scaling algorithm for minimizing submodular functions. SIAM J. Comput. **32**(4), 833–840 (2003)
36. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial strongly polynomial algorithm for minimizing submodular functions. J. ACM **48**(4), 761–777 (2001)
37. Kiers, H.A.L.: A modification of the SINDCLUS algorithm for fitting the ADCLUS and INDCLUS. J. Classif. **14**(2), 297–310 (1997)
38. Lee, M., Navarro, D.: Minimum description length and psychological clustering models. In: Grunwald, P., Myung, I., Pitt, M. (eds.) Advances in Minimum Description Length Theory and Applications. Neural Information Processing Series MIT Press, pp. 355–384 (2005). https://mitpress.mit.edu/books/advances-minimum-description-length
39. McCormick, S.T.: Chapter 7. Submodular function minimization. In: Aardal, K., Nemhauser, G.L., Weismantel, R. (eds.) Handbook on Discrete Optimization, pp. 321–391. Elsevier, Amsterdam (2005). Version 3a (2008). Available at http://people.commerce.ubc.ca/faculty/mccormick/sfmchap8a.pdf
40. Mirkin, B.G.: Additive clustering and qualitative factor analysis methods for similarity matrice. J. Classif. **4**, 7–31 (1987). Erratum, J. Classif. **6**, 271–272 (1989)
41. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988)
42. Orlin, J.B.: A faster strongly polynomial time algorithm for submodular function minimization. Math. Program. **118**(2, Ser. A), 237–251 (2009)
43. Picard, J.C., Queyranne, M.: A network flow solution to some nonlinear $0 - 1$ programming problems, with applications to graph theory. Networks **12**, 141–159 (1982)
44. Qian, J., Wang, C.A.: How much precision is needed to compare two sums of square roots of integers? Inf. Process. Lett. **100**(5), 194–198 (2006)
45. Radzik, T.: Fractional combinatorial optimization. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, pp. 1077–1080. Springer, Berlin (2009)
46. Rhys, J.M.W.: A selection problem of shared fixed costs and network flows. Manag. Sci. **17**, 200–207 (1970)
47. Schaible, S.: Analyse und Anwendungen von Quotientenprogrammen, ein Beitrag zur Planung mit Hilfe der nichtlinearen Programmierung. Mathematical Systems in Economics, vol. 42. Verlag Anton Hain, Königstein/Ts. (1978)
48. Schaible, S., Shi, J.: Recent developments in fractional programming: single-ratio and max-min case. In: Nonlinear Analysis and Convex Analysis, pp. 493–506. Yokohama Publishers, Yokohama (2004)
49. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theory Ser. B **80**(2), 346–355 (2000)
50. Shepard, R.N., Arabie, P.: Additive clustering: representation of similarities as combinations of discrete overlapping properties. Psychol. Rev. **86**(2), 87–123 (1979)
51. Stancu-Minasian, I.M.: Fractional Programming: Theory, Methods, and Applications. Kluwer, Dordrecht (1997)
52. Stancu-Minasian, I.M.: A sixth bibliography of fractional programming. Optimization **55**(4), 405–428 (2006)
53. Topkis, D.M.: Minimizing a submodular function on a lattice. Oper. Res. **26**(2), 305–321 (1978)
54. Topkis, D.M.: Supermodularity and Complementarity. Frontiers of Economic Research. Princeton University Press, Princeton (1998)
55. Ursulenko, O.: Exact methods in fractional combinatorial optimization. ProQuest LLC, Ann Arbor, MI. Ph.D. thesis, Texas A&M University (2009)
56. Zadeh, N.: A bad network problem for the simplex method and other minimum cost flow algorithms. Math. Program. **5**, 255–266 (1973)

Clusters, Orders, and Trees: Methods and Applications
In Honor of Boris Mirkin's 70th Birthday
Aleskerov, F.T.; Goldengorin, B.; Pardalos, P. (Eds.)
2014, XI, 404 p. 80 illus., 28 illus. in color., Hardcover