Math. Program., Ser. B (2010) 124:255–269
DOI 10.1007/s10107-010-0364-8

FULL LENGTH PAPER

Approximating a class of combinatorial problems with rational objective function

José R. Correa · Cristina G. Fernandes · Yoshiko Wakabayashi

Received: 31 July 2008 / Accepted: 10 August 2009 / Published online: 8 May 2010

Abstract In the late seventies, Megiddo proposed a way to use an algorithm for the problem of minimizing a linear function $a_0 + a_1x_1 + \cdots + a_nx_n$ subject to certain constraints to solve the problem of minimizing a rational function of the form $(a_0+a_1x_1+\cdots+a_nx_n)/(b_0+b_1x_1+\cdots+b_nx_n)$ subject to the same set of constraints, assuming that the denominator is always positive. Using a rather strong assumption, Hashizume et al. extended Megiddo's result to include approximation algorithms. Their assumption essentially asks for the existence of good approximation algorithms for optimization problems with possibly negative coefficients in the (linear) objective function, which is rather unusual for most combinatorial problems. In this paper, we present an alternative extension of Megiddo's result for approximations that avoids this issue and applies to a large class of optimization problems. Specifically, we show that, if there is an α-approximation for the problem of minimizing a nonnegative linear function subject to constraints satisfying a certain *increasing* property then there is an α-approximation ($1/\alpha$-approximation) for the problem of minimizing (maximizing) a

A preliminary version of this paper appeared in the Proceedings of SWAT 2006. Research partially supported by CNPq Prosul Proc. 490333/04-4, Proc. 307011/03–8, and 305702/07–6 (Brazil); ProNEx - FAPESP/CNPq Proc. 2003/09925-5 (Brazil); and CONICYT (Chile), grants Anillo en Redes ACT08 and Fondecyt 1060035.

J. R. Correa (✉)
Department of Industrial Engineering, Universidad de Chile, Santiago, RM, Chile
e-mail: jcorrea@dii.uchile.cl

C. G. Fernandes · Y. Wakabayashi
Department of Computer Science, Universidade de São Paulo, São Paulo, Brazil
e-mail: cris@ime.usp.br

Y. Wakabayashi
e-mail: yw@ime.usp.br

nonnegative rational function subject to the same constraints. Our framework applies to covering problems and network design problems, among others.

Keywords Approximation algorithms · Rational objective · Covering

Mathematics Subject Classification (2000) 68W25 Approximation algorithms · 90C27 Combinatorial optimization

1 Introduction

We address the problem of finding approximate solutions for a class of combinatorial optimization problems with rational objective function. Our starting point is the seminal work of Megiddo [20] who showed that, for a large class of problems, optimizing a rational objective can be done in polynomial time, as long as there is an efficient algorithm to optimize a linear objective. The class of problems we address has a natural motivation in particular in network design problems. Suppose we want to build a network where each link has a construction cost, as well as a profit. The profit could measure some overall social benefit associated with the corresponding link or, for example, could be inversely related to the environmental damage its construction causes. The goal then would be to find a network (satisfying some connectivity requirements) that minimizes the cost-benefit ratio. In general, whenever we are faced with problems where cost-to-profit relations have to be optimized, we are in this situation.

Fractional programming, which deals with the optimization of the ratio of two functions subject to constraints, has attracted attention since the sixties in the continuous optimization community [2,8,12]. Optimizing rational objectives is a particular case of fractional programming in which both the denominator and the numerator of the fraction to be optimized are linear functions. Several combinatorial optimization problems have been studied in this context. For instance, minimum mean-weight cycle [15], minimum or maximum average cost problems [3,6,10,21], minimum cost-to-time ratio problems [5,7,19], minimum cost-reliability problems [16], fractional knapsack problems [1], fractional assignment problems [24], among others [18,23]. A concrete example of this class of problems, that was considered recently, is the one studied by Gubbala and Raghavachari [10]: given a weighted k-connected graph G, find a k-connected spanning subgraph of G with minimum average weight. They looked at the edge and vertex connectivity versions of the problem. Besides proving that these problems are NP-hard, they presented a 3-approximation algorithm for the edge-connectivity version, and an $O(\log k)$-approximation algorithm for the vertex-connectivity version.

Although Megiddo's paper has motivated mostly works on exact algorithms, some works on approximation algorithms have also been carried out. The latter include the work of Hashizume, Fukushima, Katoh and Ibaraki [11] who extended Megiddo's approach to take into account approximation algorithms for a class of optimization problems under some assumptions. The result of Hashizume et al. [11] is of similar flavor to what we do here. They proved that an α-approximation to a combinatorial problem with a rational objective can be derived from an α-approximation for its linear

counterpart. However, they need a rather strong assumption, namely, the existence of a polynomial-time algorithm for the linear version of the combinatorial problem that gives an α-approximate solution even if negative weights are allowed. As they show, such an algorithm exists for the knapsack problem, allowing them to deduce approximation results for the fractional knapsack problem. Note however that this does not hold for most optimization problems, in particular, for the ones we consider here. For instance, for the problems considered by Gubbala and Raghavachari [10] with $k = 2$, there is no constant factor approximation algorithm, unless P = NP, if we allow the weights to be arbitrary. Indeed, if we put weight $-(n-1)$ for a specific edge $e = (u, v)$ and weight 1 for all other edges, then the optimal 2-connected subgraph has weight 0 if and only if there is a Hamiltonian path from u to v, and has weight at least 1 otherwise. This implies that there cannot be an approximation algorithm for the problem if we allow negative weights, unless P = NP. Therefore, the results by Hashizume et al. cannot be applied to those problems (among others).

In this paper, we build on Megiddo's work to derive approximation algorithms for the rational version of a class of combinatorial problems that contains many covering and network connectivity problems, including the ones considered by Gubbala and Raghavachari. Furthermore, our approach works for many well-known graph problems such as minimum vertex cover, minimum feedback arc and vertex set, minimum dominating set, minimum multiway cut, and also some general covering problems such as minimum set cover and minimum hitting set. Specifically, we prove that if there is an α-approximation for a problem with a linear objective function with nonnegative coefficients, then there is an α-approximation for its (nonnegative) "rational" version.

1.1 Organization of the paper

In Sect. 2, we introduce the notation, describe the framework in which our result holds, and briefly discuss the approach we take to tackle the rational function approximation. Then, in Sect. 3 we give our algorithm, analyze its running time, and prove its correctness. Although the algorithm we present in Sect. 3 is very efficient in most cases (in particular if the coefficients are not too large), it does not run in strongly polynomial time. Thus, in Sect. 4 we show how to adapt Megiddo's technique to derive a strongly polynomial-time version of our algorithm. We finish with some final remarks in Sect. 5.

2 Preliminaries

2.1 The setting

Our goal is to derive a general approximation technique for a class of NP-hard combinatorial problems with rational objective function of the form $(a_0 + a_1x_1 + \cdots + a_nx_n)/(b_0 + b_1x_1 + \cdots + b_nx_n)$, where the a_i's and b_i's are nonnegative integers and $x_i \in \{0, 1\}$ for each i.

The class of problems to which our framework applies can be described as follows. Let $U = \{e_1, \ldots, e_n\}$ be a finite ground set, and f be a binary (i.e., $\{0, 1\}$-valued) function defined on the subsets of U. We say f is *increasing* if $f(U) = 1$ and $f(S) \leq f(S')$

for all supersets S' of S. Our framework applies to any problem that seeks a set $S \subseteq U$ satisfying $f(S) = 1$ such that its corresponding characteristic vector minimizes a rational objective as above. It is straightforward to verify that this class of problems contains all rational versions of the previously mentioned problems. Moreover, it contains the following class of integer programming problems, with covering constraints,

$$\min \left\{ ax/bx : \; Ax \geq d, \; x \in \{0,1\}^n \right\},$$

where all entries of A, a, b and d are nonnegative rationals. Indeed, in this setting, the ground set will be $U = \{1, \ldots, n\}$ and the binary function $f(S)$ will be 1 if and only if the characteristic vector of S is a feasible solution for $Ax \geq d$. Since the entries of A and d are nonnegative, f is increasing (as long as the problem is not infeasible, that is, as long as $f(U) = 1$).

2.2 A first approach

A well known approach to solve a problem of the form minimize ax/bx subject to $x \in X$, which we employ in this paper, is to guess a parameter c and minimize $ax - c(bx)$. We describe this in detail in the next section. However, an even simpler approach is to repeatedly solve the problem of minimizing ax subject to $x \in X$ and $bx \geq B$, where B is a guess for the denominator. Unfortunately, it is not clear how to make this idea work in general because adding one more linear constraint may turn the linear problem (minimize ax subject to $x \in X$) into a harder problem. For instance, if $X = \{0,1\}^n$ then minimize ax subject to $x \in X$ is a trivial problem. However, minimize ax subject to $x \in X$ and $\sum_{i=1}^n a_i x_i \geq \sum_{i=1}^n a_i/2$ is equivalent to the partition problem [9].

Also in terms of approximability, adding a linear constraint to an integer linear program (IP), we may turn it into a harder problem. An easy example is to consider the standard IP for finding a minimum cost 2-edge connected spanning subgraph in a given connected graph with costs on its edges, which has constant factor approximations. By adding a linear constraint ($\sum_e x_e \leq n$, where n is the number of vertices in the given graph and x_e is the variable associated to edge e), we turn it into the TSP, which is not approximable within any factor unless P = NP. In this case however, the added constraint is not a "covering" one. Nevertheless, we now present an example of a problem that becomes provably harder to approximate when adding a covering constraint. In particular, this (somewhat technical) problem suits the framework of this paper and it is derived from the vertex cover problem. Consider the vertex cover problem in connected graphs of maximum degree 3, which is APX-hard [22]. Denote by G a connected graph whose maximum degree is 3 and by n the number of vertices in G. Note that any minimum vertex cover of G is of size at least $\lceil (n-1)/3 \rceil$ and at most $3n/4$. Consider the usual IP for vertex cover and add an extra variable, say z, to each of the inequalities. Add z also to the objective function with a coefficient of $\lceil (n-1)/3 \rceil$. The resulting IP is easy to solve: an optimal solution is $z = 1$ and all other variables null. Let us show that, if we add the ("covering") constraint $\sum_{u=1}^n x_u \geq \lceil (n-1)/3 \rceil$, where x_u is the variable for vertex u, then there is no PTAS for the resulting IP, unless P = NP. Indeed,

suppose there is such a PTAS. Let G' be obtained from G by hanging a path of length $2n$ to an arbitrary vertex of G. Then the number of vertices of G' is $n' = n + 2n = 3n$. Observe that any vertex cover in G' has size at least $\lceil (n-1)/3 + n \rceil = \lceil (4n-1)/3 \rceil$ and at most $3n/4 + n = 7n/4$. Now consider the IP described above for G'. The coefficient of the variable z in its IP is $\lceil (n'-1)/3 \rceil = \lceil (3n-1)/3 \rceil = \lceil n - 1/3 \rceil = n$. The extra ("covering") constraint for G' is $\sum_{u=1}^{n'} x_u \geq n$. Any solution for this IP with $z = 1$ has value at least $2n$. So, any PTAS will ultimately output a solution with $z = 0$. That is, it will be a PTAS for finding a vertex cover for G'. But, from a PTAS for G', we get a PTAS for G. That is, there would be a PTAS for the vertex cover problem in connected graphs of maximum degree 3.

2.3 Main result

Our main result states that if f is increasing and there is an α-approximation algorithm for the problem of finding a set $S \subseteq U$ such that $f(S) = 1$ minimizing a linear function with nonnegative coefficients, then there is an α-approximation algorithm to find a set $S \subseteq U$ such that $f(S) = 1$ minimizing a nonnegative rational function. Thus, this framework allows the "rational" version of several problems to inherit the approximation results for their standard versions.

For instance, we can improve upon the results obtained by Gubbala and Raghavachari [10]. They showed a 3-approximation (resp. a $(1 + 2\sqrt{2H_k} + 2H_k)$-approximation) for the problem of finding a minimum average weight k-edge (resp. vertex) connected subgraph, where H_k is the k^{th} harmonic number. Indeed, we obtain a 2-approximation algorithm for the edge-connectivity version, and a $2H_k$-approximation algorithm for the vertex-connectivity version. The former follows by using the algorithm by Khuller and Vishkin [17] for the problem of finding a minimum weight k-edge connected spanning subgraph, while the latter follows by using the $2H_k$-approximation for the problem of finding a k-vertex connected spanning subgraph of minimum weight of Jain, Măndoiu, Vazirani, and Williamson [14]. We can also derive a 2-approximation algorithm for the "rational" version of the more general edge-connectivity problem studied by Jain [13].

The scheme can be adapted for maximizing rational objectives, however the result is not exactly symmetric. What we get in this case is the following. For the same class of problems (given by an increasing property), if we have an α-approximation for *minimizing* a nonnegative linear objective, we obtain a $1/\alpha$-approximation for maximizing a nonnegative rational objective. (This corresponds to applying the scheme above to minimize the inverted fraction.) This asymmetry is somehow expected. Indeed, the maximization of a linear function on domains given by an increasing property is trivial (the ground set is optimum). However it is not obvious how to maximize a rational function on the same domain. For instance, it is trivial to find a set cover of maximum weight (when all weights are nonnegative) but how do we find a set cover of maximum average weight?

The main idea behind our algorithm is to use a transformed cost function which depends on a, b and a certain parameter (which is nothing but a guess of the optimal value), and then search for the parameter that gives a "right" answer. Although this

trick is fairly standard in parametric optimization, we want to avoid negative costs, so we need to "truncate" the costs. Another difficulty here is that we are dealing with approximate solutions. Therefore we need to prove that if the parameter is sufficiently close to the optimal value, then the approximate solution is close as well (up to a certain factor). Unfortunately, because of the truncated costs, we can only prove a one-sided result (Lemma 1), which makes the search part of the algorithm harder. Nevertheless there is a way around this issue with essentially no extra computational effort as we show in Theorem 1 and Theorem 2.

In what follows, if S is a subset of a set U and w is a function that assigns a number to each element of U, we let $w(S)$ denote the sum of w_e for all e in S. Also, given an increasing binary function f, we will say that $S \subseteq U$ is *feasible* if and only if $f(S) = 1$.

3 Approximating rational objectives

Let f be an increasing binary function defined on all subsets of a finite set U. Recall that f is *increasing* if and only if $f(U) = 1$ and $f(B) \leq f(A)$, for all $B \subseteq A \subseteq U$. Also, we will assume that f is *polynomially computable*, i.e., there is a polynomial-time algorithm that, given a subset S of U, computes $f(S)$. We are interested in the following problems:

MINLIN (U, w, f): Given a finite set U, a nonnegative rational w_e for each e in U, and a polynomially computable increasing function $f : 2^U \to \{0, 1\}$ (usually given implicitly), find a subset S of U such that $f(S) = 1$ and $w(S)$ is minimum.

MINRATIONAL (U, a, b, f): Given a finite set U, nonnegative integers a_0, b_0 and a_e and b_e for each e in U, and a polynomially computable increasing function $f : 2^U \to \{0, 1\}$ (usually given implicitly), find a subset S of U such that $f(S) = 1$ and $(a_0 + a(S))/(b_0 + b(S))$ is minimum. (To avoid divisions by zero, we assume that $b_0 + b(S) > 0$ for all feasible S.)

For instance, the problem of, given a weighted k-edge-connected graph G, finding a k-edge-connected spanning subgraph of G with minimum average weight is an example of MINRATIONAL (U, a, b, f). The set U in this case is the set of edges of G and a_e is the weight of edge e in G while $b_e = 1$ for each edge e of G. Of course, $a_0 = b_0 = 0$. The function f is such that $f(S) = 1$ if and only if the subgraph of G induced by the edge set S is spanning and k-edge-connected.

We now describe how an α-approximation algorithm for MINLIN (U, w, f) can be turned into an α-approximation algorithm for MINRATIONAL (U, a, b, f). To this end, let MINWEIGHT$_\alpha(U, w, f)$ denote an α-approximation algorithm for MINLIN (U, w, f). Note that α may depend on the input. It is easy to see that we can assume $\alpha \leq |U|$. Indeed, consider the following algorithm for MINLIN (U, w, f):

TRIVIAL (U, w, f)
1 sort the elements in U in nondecreasing order of w, obtaining $w_{e_1} \leq \cdots \leq w_{e_{|U|}}$

2 find the smallest index i such that $f(\{e_1, \ldots, e_i\}) = 1$
3 **return** $\{e_1, \ldots, e_i\}$

Clearly, $\sum_{j=1}^{i} w_{e_j} \leq i\, w_{e_i} \leq |U|\, w_{e_i}$. Moreover, it is immediate that TRIVIAL finds a feasible set S minimizing $\max\{w_e : e \in S\}$. Thus, any optimal solution to MINLIN(U, w, f) contains an element of U of weight at least w_{e_i}, which implies that TRIVIAL achieves a ratio of at most $|U|$. In summary, we can run both MINWEIGHT$_\alpha$ and TRIVIAL and therefore assume that $\alpha \leq |U|$.

3.1 The algorithm

The core of our transformation is given by the following routine, which we call AUXILIAR$_\alpha$. Observe that it applies the traditional trick used, for instance, by Karp [15], Megiddo [20], and Hashizume et al. [11], adapted to avoid negative entries in the derived weight function.

AUXILIAR$_\alpha$ (U, a, b, f, c)
1 $L \leftarrow \{e : a_e \leq c\, b_e\}$
2 **for** each element e in U **do**
3 $\quad w_e \leftarrow \max\{0, a_e - c\, b_e\}$
4 $S \leftarrow$ MINWEIGHT$_\alpha$ (U, w, f)
5 **return** $S \cup L$.

In what follows, AUXILIAR$_\alpha$ is used to find an α-approximate solution to MINRATIONAL. Algorithm MINFRAC$_\alpha$ consists of two phases: an "approximate" truncated binary search and an extra step needed to assure the ratio α. After the truncated binary search, we guarantee that the algorithm found a feasible solution within ratio α or the search interval contains the optimum value. At the end, the search interval, scaled by α, is small enough (choice of ϵ) to contain at most one possible value of a feasible solution. If the best solution found so far (S_t in line 13) is not within ratio α, the second phase finds a better feasible solution that will be within ratio α.

Let ratio(S) denote the ratio $(a_0 + a(S))/(b_0 + b(S))$ for any feasible set S.

MINFRAC$_\alpha$ (U, a, b, f)
1 $\epsilon \leftarrow 1/((b_0 + b(U))^2\alpha)$ ▷ **First phase**
2 $i \leftarrow 1$
3 $S_0 \leftarrow U$
4 *left* $\leftarrow 0$
5 *right* $\leftarrow$ ratio(U)
6 **while** *right* $-$ *left* $> \epsilon$ **do**
7 $\quad$ *middle* $\leftarrow$ (*left* $+$ *right*)$/2$
8 $\quad S_i \leftarrow$ AUXILIAR$_\alpha$ $(U, a, b, f,$ *middle*$)$
9 $\quad$ **if** ratio$(S_i) \leq \alpha$ *middle*
10 $\quad\quad$ **then** *right* $\leftarrow$ *middle*
11 $\quad\quad$ **else** *left* $\leftarrow$ *middle*
12 $\quad i \leftarrow i + 1$
13 $S_t \leftarrow \text{argmin}\{\text{ratio}(S_j) : 0 \leq j < i\}$
14 $c' \leftarrow$ ratio$(S_t)/\alpha$ ▷ **Second phase**

15 $S' \leftarrow \text{AUXILIAR}_\alpha\ (U, a, b, f, c')$
16 $S \leftarrow \text{argmin}\{\text{ratio}(S_t), \text{ratio}(S')\}$
17 **return** S.

We note that in the previous algorithm α might be in fact $\alpha(|U|)$, if α is a function, not simply a constant.

3.2 Analysis of the running time

Let us show that the above algorithm is polynomial in the size of its input.

Theorem 1 *Algorithm* $\text{MINFRAC}_\alpha(U, a, b, f)$ *runs in polynomial time. More precisely, it runs in time* $O(\log(a_{\max}) + \log(b_{\max}) + \log|U|)$ *times the running time of* $\text{MINWEIGHT}_\alpha(U, w, f)$, *where* $a_{\max} = \max\{a_e : e \in U \cup \{0\}\}$ *and* $b_{\max} = \max\{b_e : e \in U \cup \{0\}\}$.

Proof First observe that the number of iterations of the **while** in line 6 is

$$\begin{aligned}\lceil \log(\text{ratio}(U)/\epsilon))\rceil &= \lceil \log((a_0 + a(U))(b_0 + b(U))\alpha)\rceil \\ &= O(\log(a_0 + a(U)) + \log(b_0 + b(U)) + \log|U|) \\ &= O(\log(a_{\max}) + \log(b_{\max}) + \log|U|).\end{aligned}$$

The second equality above uses the fact that $\alpha \leq |U|$. Now, the most time consuming operation within each iteration of the **while** is the call to AUXILIAR_α. Clearly AUXILIAR_α runs in polynomial time in the size of its input. Moreover, its running time is exactly that of MINWEIGHT_α. Therefore, it is enough to verify that, in each call of AUXILIAR_α at line 8 of MINFRAC_α, the parameter $middle$ has size polynomially bounded by the size of (U, a, b, f). Indeed, in the i^{th} call of AUXILIAR_α, we have that $middle$ is a multiple of $\text{ratio}(U)/2^i$, where $i = O(\log(a_{\max}) + \log(b_{\max}) + \log|U|)$ (as the number of iterations of the **while**). Therefore each AUXILIAR_α call runs in polynomial time in the size of (U, a, b, f). □

Observe that, if we are given a PTAS (resp. FPTAS) for MINLIN, then we have a PTAS (resp. FPTAS) for MINRATIONAL. Unfortunately, if we are given a strongly polynomial algorithm for MINLIN, we only obtain a polynomial algorithm for MINRATIONAL this way. But in Sect. 4 we will show that, under some assumptions, we can get a strongly polynomial algorithm for MINRATIONAL from a strongly polynomial one for MINLIN.

3.3 Analysis of the approximation ratio

First observe that as MINWEIGHT_α returns a subset S of U such that $f(S) = 1$, AUXILIAR_α also returns a subset S of U such that $f(S) = 1$ (so $b_0 + b(S) > 0$). Therefore, MINFRAC_α returns a subset S of U such that $f(S) = 1$, i.e., a feasible solution. Now we focus on the approximation ratio. To this end, we need to establish a key lemma.

Proposition 1 *Let $c \geq 0$ and $w_e = \max\{0, a_e - c\, b_e\}$ for all $e \in U$. Consider the set $L = \{e \in U : a_e \leq c\, b_e\}$ and define the quantity $D = \sum_{e \in L}(a_e - c\, b_e) = a(L) - c\, b(L)$. Then, if $R \subseteq U$,*

$$w(R) \;\leq\; a(R) - c\, b(R) - D.$$

Moreover, if $L \subseteq R$, then equality holds.

Proof We have that,

$$\begin{aligned} w(R) &= a(R \backslash L) - c\, b(R \backslash L) \\ &= a(R) - a(R \cap L) - c\, b(R) + c\, b(R \cap L) \\ &= a(R) - c\, b(R) - (a(R \cap L) - c\, b(R \cap L)) \\ &\leq a(R) - c\, b(R) - D. \end{aligned} \tag{1}$$

The last inequality holds because each term in the sum that defines D is negative or zero. Note also that, if $L \subseteq R$, then equality holds in (1). □

Lemma 1 *Let c^* be the optimal value of* MINRATIONAL(U, a, b, f). *For any $c \geq c^*$, if $\hat{S}$ is the output of* AUXILIAR$_\alpha(U, a, b, f, c)$, *then*

$$\text{ratio}(\hat{S}) \;=\; \frac{a_0 + a(\hat{S})}{b_0 + b(\hat{S})} \;\leq\; \alpha\, c. \tag{2}$$

Moreover, the previous inequality is strict whenever $c > c^*$.

Proof We want to prove (2), which is equivalent to

$$a(\hat{S}) - \alpha\, c\, b(\hat{S}) \;\leq\; \alpha\, c\, b_0 - a_0.$$

Let $L = \{e \in U : a_e \leq c\, b_e\}$ and $D = a(L) - c\, b(L)$. We start showing that $w(\hat{S}) + \alpha\, D \leq \alpha\, c\, b_0 - a_0$, where $w_e = \max\{0, a_e - c\, b_e\}$ as defined in lines 2–3 of the AUXILIAR$_\alpha$ routine. Indeed, since MINWEIGHT$_\alpha(U, w, f)$ is an α-approximation algorithm for MINLIN(U, w, f), we have that $w(\hat{S}) = w(S) \leq \alpha\, \text{opt}(\text{MINLIN}(U, w, f)) \leq \alpha\, w(S^*)$, where S denotes the set defined in line 4 of the AUXILIAR$_\alpha$ routine and S^* denotes an optimum solution for MINRATIONAL(U, a, b, f). This implies that

$$w(\hat{S}) + \alpha\, D \;\leq\; \alpha\, w(S^*) + \alpha\, D \;=\; \alpha(w(S^*) + D).$$

Also, from Proposition 1 we have that $\alpha(w(S^*) + D) \leq \alpha(a(S^*) - c\, b(S^*))$. By noting that $c \geq c^* = \text{ratio}(S^*) = (a_0 + a(S^*))/(b_0 + b(S^*))$, we can conclude that

$$w(\hat{S}) + \alpha\, D \;\leq\; \alpha(a(S^*) - c\, b(S^*)) \;\leq\; \alpha(c\, b_0 - a_0) \;\leq\; \alpha\, c\, b_0 - a_0.$$

Furthermore, the middle inequality is strict if $c > c^*$.

With the previous inequality in hand, we turn to finish the proof. Note that

$$\begin{aligned} a(\hat{S}) - \alpha\, c\, b(\hat{S}) &= w(\hat{S}) + c\, b(\hat{S}) + D - \alpha\, c\, b(\hat{S}) && (3)\\ &= w(\hat{S}) + D + (\alpha - 1)(-c\, b(\hat{S})) \\ &\leq w(\hat{S}) + D + (\alpha - 1)D && (4)\\ &= w(\hat{S}) + \alpha\, D \\ &\leq \alpha\, c\, b_0 - a_0. && (5) \end{aligned}$$

As $\hat{S}$ contains L, equality (3) holds by Proposition 1. Inequality (4) holds because $D = a(L) - c\, b(L) \geq -c\, b(L) \geq -c\, b(\hat{S})$ (again since $\hat{S}$ contains L). Finally, inequality (5) follows by the previous argument, and it is strict if $c > c^*$. The proof is complete. □

Unfortunately, we do not have any guarantee on ratio($\hat{S}$) if $c < c^*$, where $\hat{S}$, c and c^* are as defined in the statement of Lemma 1. Nevertheless, MINFRAC$_\alpha$ gets around this and still provides an α-approximation for MINRATIONAL, as we show next.

Theorem 2 *Let S be the output of algorithm* MINFRAC$_\alpha(U, a, b, f)$ *and S^* be an optimal solution to problem* MINRATIONAL(U, a, b, f). *We have that*

$$\text{ratio}(S) \;\leq\; \alpha\, \text{ratio}(S^*).$$

Proof Let $c^* = \text{ratio}(S^*)$. Suppose that at some iteration, say iteration i, of MINFRAC$_\alpha(U, a, b, f)$ step 10 was executed and at that point we had $middle \leq c^*$. Clearly, in this case we are done as this implies that ratio(S_i) $\leq \alpha\, middle \leq \alpha c^*$, and moreover ratio($S$) $\leq$ ratio(S_j) for all j. Thus, we may assume that whenever step 10 was executed, we had that $middle > c^*$. On the other hand, if at some iteration i step 11 was executed, we had that $middle < c^*$. Otherwise, Lemma 1 would imply that AUXILIAR$_\alpha(U, a, b, f, middle)$ returns a set S_i such that ratio(S_i) $\leq \alpha\, middle$, contradicting the fact that step 11 was executed.

Thus it remains to analyze the case in which, at each iteration, either step 10 is executed and $middle > c^*$, or step 11 is executed and $middle < c^*$. In this case, at step 13 we have that $left \leq c^* \leq right$ and $right - left \leq \epsilon$. Note that this is enough to justify that MINFRAC$_\alpha(U, a, b, f)$ is an $(\alpha + \delta)$-approximation, for some $\delta > 0$. (Indeed, ratio(S_t) $\leq \alpha\, right \leq \alpha(c^* + \epsilon) \leq (\alpha + 1/(b_0 + b(U))^2)c^*$.) Steps 14–16 are what we need to get rid of the additive term δ.

So, suppose that at step 13 we have ratio(S_t) $> \alpha c^*$. By definition we have that c' defined in step 14 is strictly greater than c^*. Then Lemma 1 implies that the set S' defined in step 15 is such that

$$\text{ratio}(S') \;<\; \alpha c' \;=\; \text{ratio}(S_t).$$

On the other hand, consider k, the last iteration of MINFRAC$_\alpha(U, a, b, f)$ at which step 10 was executed (if step 10 was never executed, we let $k = 0$). It is straightforward

to see that

$$\alpha\, left \;\leq\; \alpha c^* \;<\; \text{ratio}(S_t) \;\leq\; \text{ratio}(S_k) \;\leq\; \alpha\, right,$$

where *right* denotes the final value of this variable in the execution of the algorithm (or its value after iteration k).

We conclude by observing that, for any two feasible solutions $F, G \subseteq U$ of different values, we have that $|\text{ratio}(F) - \text{ratio}(G)| > 1/(b_0 + b(U))^2$. Thus,

$$\begin{aligned}
\text{ratio}(S') &< \text{ratio}(S_t) - \frac{1}{(b_0 + b(U))^2} \\
&\leq \alpha\, right - \frac{1}{(b_0 + b(U))^2} \\
&= \alpha(right - \epsilon) \\
&\leq \alpha\, left \\
&\leq \alpha c^*.
\end{aligned}$$

This concludes the proof of the theorem. □

4 Applying Megiddo's technique

In this section we outline how to turn MINFRAC$_\alpha$ into a strongly polynomial-time algorithm using the approach of Megiddo. The reader is referred to the short paper [20] for details behind this approach. We even follow closely the notation of Megiddo's paper, so that it becomes easier to understand our approach.

Recall that Megiddo showed how to derive a polynomial-time algorithm for MINRATIONAL (without the increasing assumption) from a polynomial-time algorithm for MINLIN. Assuming that comparisons and additions are the only operations the algorithm for MINLIN(U, w, f) does on w, Megiddo's scheme leads to a strongly polynomial-time algorithm for MINRATIONAL as long as the algorithm for MINLIN is also strongly polynomial. Algorithm MINFRAC$_\alpha$ in Section 3 shows how to get an α-approximation algorithm for MINRATIONAL from an α-approximation algorithm for MINLIN. Nevertheless, in the description given in Section 3, even if MINWEIGHT$_\alpha(U, w, f)$ runs in strongly polynomial time, MINFRAC$_\alpha(U, a, b, f)$ will not be strongly polynomial. In what follows, we describe how to get a strongly polynomial-time α-approximation algorithm for MINRATIONAL if, as in Megiddo's algorithm, comparisons and additions are the only operations MINWEIGHT$_\alpha(U, w, f)$ does on w. The idea is that of Megiddo with a few adjustments to accommodate the non-negativity of the function w as well as the approximation goal.

We start with a general description of the algorithm. It consists of two phases. In the first phase, we sort the ratios in $\{a_e/b_e : e \in U \cup \{0\}\}$ in nondecreasing order (consider $a_e/b_e = \infty$ if $b_e = 0$) and let $c_1 \leq \cdots \leq c_{n+1}$ be the result of this sorting, where $n = |U|$. Let $c_0 = 0$ and S_j be the output of AUXILIAR$_\alpha(U, a, b, f, c_j)$ for $j = 0, \ldots, n+1$. Also, let k be the largest j such that $\text{ratio}(S_j) > \alpha\, c_j$. Note that

$k < n+1$. (Indeed, $S_{n+1} = U$ and ratio$(U) \leq c_{n+1}$.) For c in the interval $[c_k \,..\, c_{k+1}]$, the function w defined in lines 2–3 of the AUXILIAR$_\alpha$ routine is linear. The idea now is to follow Megiddo's strategy starting from interval $[c_k \,..\, c_{k+1}]$. Below, we show the resulting algorithm in pseudo-code. The description of the second phase follows the one of algorithm B of Megiddo [20, p. 416].

```
MEGIDDO_APPROX_α (U, a, b, f)
    ▷ First phase
 1  n ← |U|
 2  c_0 ← 0
 3  S_0 ← U
 4  let c_1 ≤ ... ≤ c_{n+1} be the ratios in {a_e/b_e : e ∈ U ∪ {0}} sorted
    nondecreasingly
 5  k ← 0
 6  for j ← 1 to n + 1 do          ▷ This could be a binary search instead.
 7      S_j ← AUXILIAR_α(U, a, b, f, c_j)
 8      if ratio(S_j) > α c_j
 9          then k ← j
    ▷ Second phase
10  i ← n + 2
11  left ← c_k
12  right ← c_{k+1}
13  for each element e in U do      ▷ Observe that w is linear in [left .. right].
14      w_e(c) ← max{0, a_e − c b_e} for c in [left .. right]
15  finished ← FALSE
16  while not finished do
17      follow MINWEIGHT_α(U, w(c), f) for all c in [left .. right] simultaneo-
        usly, from the start or from the recent point of resumption, to the
        next comparison
18      if there are no more comparisons and MINWEIGHT_α(U, w(c), f) terminates
19          then finished ← TRUE
20          else let g_1(c) and g_2(c) be the functions to be compared over
                [left .. right]
21              if there is a unique solution of g_1(c) = g_2(c) in [left .. right]
22                  then let c' be the unique solution of g_1(c) = g_2(c) in
                        [left .. right]
23                       S_i ← AUXILIAR_α(U, a, b, f, c')
24                       if ratio(S_i) ≤ α c'
25                           then right ← c'
26                           else left ← c'
27                       i ← i + 1
28              resume algorithm MINWEIGHT_α(U, w(c), f)
30  S ← argmin{ratio(S_j) : 0 ≤ j < i}
31  return S.
```

Observe that we could also have used the first 12 steps of the above algorithm in the beginning of MINFRAC$_\alpha$ to initialize the variables *left* and *right* in lines 4 and 5 of

MINFRAC$_\alpha$ (with the values c_k and c_{k+1}). Then we would apply the "truncated" binary search as before. The worst-case running time of MINFRAC$_\alpha$ with this modification however remains the same.

4.1 Analysis of the approximation ratio

By definition, k is such that ratio$(S_k) > \alpha\, c_k$ and ratio$(S_{k+1}) \leq \alpha\, c_{k+1}$. Lemma 1 assures that $c_k < c^*$. Also, if $c_{k+1} \leq c^*$, then S_{k+1} achieves the desired ratio. Indeed,

$$\text{ratio}(S_{k+1}) \;\leq\; \alpha\, c_{k+1} \;\leq\; \alpha\, c^*.$$

Thus the set S defined in line 30, which is the output of the algorithm, also achieves the desired ratio and we are done in this case. Therefore we may assume that $c_k < c^* < c_{k+1}$. In this case, we need to argue that the set S of line 30 also achieves the ratio α.

We prove next that the following invariant holds at the beginning of each iteration i of the **while** loop in line 16: $c^* \in [\mathit{left}\,..\,\mathit{right}]$ or ratio$(S_j) \leq \alpha\, c^*$ for some j in $\{0, \ldots, i-1\}$. As argued above, this invariant holds prior to the first iteration of the **while** loop in line 16. Suppose it holds at the beginning of an arbitrary iteration where *finished* is FALSE. Let us show that it then holds at the end of this iteration. If, in the beginning of the iteration, ratio$(S_j) \leq \alpha\, c^*$ for some j in $\{0, \ldots, i-1\}$, then there is nothing to prove. Therefore, assume $c^* \in [\mathit{left}\,..\,\mathit{right}]$. If lines 22–27 are not executed in this iteration, also there is nothing to prove, because in this case neither *left* nor *right* changed and thus, clearly, $c^* \in [\mathit{left}\,..\,\mathit{right}]$ in the end of the iteration. Hence, assume that lines 22–23–24–{25 or 26}–27 are executed. If line 26 is executed, then Lemma 1 assures that $c' < c^*$ and the invariant holds at the end of this iteration. If line 25 is executed and $c' < c^*$, then ratio$(S_i) \leq \alpha\, c' < \alpha\, c^*$, and the invariant clearly holds at the end of this iteration. On the other hand, if line 25 is executed and $c^* \leq c'$, also the invariant holds at the end of this iteration because then $\mathit{right} = c'$ and thus $c^* \in [\mathit{left}\,..\,\mathit{right}]$. This completes the proof of the invariant.

The approximation ratio of the algorithm is an easy consequence of this invariant. After the last iteration of the **while** loop in line 16, by the invariant, we have that ratio$(S_j) \leq \alpha\, c^*$ for some j in $\{0, \ldots, i-1\}$ and therefore ratio$(S) \leq \alpha\, c^*$, or c^* as well as the last value of c' are in $[\mathit{left}\,..\,\mathit{right}]$ and the set S_{i-1}, which is the output of AUXILIAR$_\alpha(U, a, b, f, c')$, is exactly the same as the output of AUXILIAR$_\alpha(U, a, b, f, c^*)$. Indeed, all comparisons result in the same way in MINWEIGHT$_\alpha(U, w(c'), f)$ and MINWEIGHT$_\alpha(U, w(c^*), f)$, and the set L defined in line 1 of AUXILIAR$_\alpha$ is the same for both calls. Thus Lemma 1 assures that ratio$(S_{i-1}) \leq \alpha\, c^*$ and consequently ratio$(S) \leq \alpha\, c^*$.

5 Final remarks

There is a way to convert the scheme proposed in this paper so that it applies to optimizing rational objectives for problems described by a *decreasing* property. This class of problems would for example include most packing problems. The scheme

would depend on the existence of an approximation algorithm for the maximization of linear objectives for this class of problems. However, for this class, the scheme of Hashizume et al. [11] already implies the same results. In other words, avoiding negative coefficients in the linear objective function is not necessary for this class of problems: elements with negative weight can be ignored without any loss.

A natural question is whether one can get rid of the requirement that f is an increasing binary function. Unfortunately the answer is likely to be no. Indeed, first observe that many hard maximization problems have an easy minimization variant, e.g. many packing problems such as the maximum independent set problem. However, even if the minimization variant is easy, their rational version is as hard as the original maximization problem. So the natural extension of our result for arbitrary functions f does not hold unless P = NP.

Recently, a related class of problems appeared in the literature. In these problems, the objective is to minimize (or maximize) the sum of rational functions [4,25]. Maybe one can apply an idea similar to the one proposed here to derive approximation algorithms for this class of problems based on approximation algorithms for their linear counterpart.

Acknowledgments The authors thank Andreas S. Schulz for many useful comments and suggestions.

References

1. Billionnet, A.: Approximation algorithms for fractional knapsack problems. Oper. Res. Lett. **30**(5), 336–342 (2002)
2. Bitran, G.R., Magnanti, T.L.: Duality and sensitivity analysis for fractional programs. Oper. Res. **24**, 675–699 (1976)
3. Carlson, J., Eppstein, D.: The weighted maximum-mean subtree and other bicriterion subtree problems. In: ACM Computing Research Repository, cs.CG/0503023. (2005)
4. Chen, D.Z., Daescu, O., Dai, Y., Katoh, N., Wu, X., Xu, J.: Efficient algorithms and implementations for optimizing the sum of linear fractional functions, with applications. J. Comb. Optim. **9**, 69–90 (2005)
5. Dantzig, G.B., Blattner, W., Rao, M.R.: Finding a cycle in a graph with minimum cost to time ratio with applications to a ship routing problem. In: Rosensthiel, P. (ed.) Theory of Graphs: International Symposium, pp. 77–84. Dunod, Paris (1967)
6. Dasdan, A., Gupta, R.K.: Faster maximum and minimum mean cycle algorithms for system-performance analysis. IEEE Trans. Comp. Aid. Des. Integr. Circ. Syst. **17**(10), 889–899 (1998)
7. Dasdan, A., Irani, S.S., Gupta, R.K.: Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems. In: Proceedings of the 36th ACM/IEEE Conference on Design Automation, pp. 37–42 (1999)
8. Dinkelbach, W.: On nonlinear fractional programming. Manag. Sci. **13**, 492–498 (1967)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP—Completeness. W.H. Freeman and Co., New York (1979)
10. Gubbala, P., Raghavachari, B.: Finding k—connected subgraphs with minimum average weight. In: Proceedings of the 6th Latin American Theoretical Informatics Symposium (LATIN), Lecture Notes in Computer Science, vol. 2976, pp. 212–221. Springer, Berlin (2004)
11. Hashizume, S., Fukushima, M., Katoh, N., Ibaraki, T.: Approximation algorithms for combinatorial fractional programming problems. Math. Program. **37**, 255–267 (1987)
12. Jagannathan, R.: On some properties of programming problems in parametric form pertaining to fractional programming. Manag. Sci. **12**, 609–615 (1966)
13. Jain, K.: A factor 2 approximation algorithm for the generalized Steiner network problem. Combinatorica **21**(1), 39–60 (2001)

14. Jain, K., Mădoiu, I., Vazirani, V.V., Williamson, D.P.: A primal-dual schema based approximation algorithm for the element connectivity problem. J. Algorithms **45**(1), 1–15 (2002)
15. Karp, R.M.: A characterization of the minimum cycle mean in a digraph. Discrete Math. **23**(3), 309–311 (1978)
16. Katoh, N.: A fully polynomial-time approximation scheme for minimum cost-reliability ratio problems. Discrete Appl. Math. **35**(2), 143–155 (1992)
17. Khuller, S., Vishkin, U.: Biconnectivity approximations and graph carvings. J. Assoc. Comput. Mach. **41**(2), 214–235 (1994)
18. Klau, G., Ljubi, I. Mutzel, P. Pferschy, U., Weiskircher R.: The fractional prize collecting Steiner tree problem on trees. In: Proceedings of the 11th European Symposium on Algorithms (ESA 2003), Lecture Notes in Computer Science, vol. 2832, pp. 691–702. Springer, Berlin (2003)
19. Lawler, E.L.: Optimal cycles in doubly weighted directed linear graphs. In: Rosensthiel, P. (ed.) Theory of Graphs: International Symposium, pp. 209–214. Dunod, Paris (1967)
20. Megiddo, N.: Combinatorial optimization with rational objective functions. Math. Oper. Res. **4**(4), 414–424 (1979)
21. Orlin, J.B., Ahuja, R.K.: New scaling algorithms for the assignment and minimum mean cycle problems. Math. Program. **54**(1), 41–56 (1992)
22. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. J. Comput. Syst. Sci **43**, 425–440 (1991)
23. Radzik T. Newton's method for fractional combinatorial optimization. In: Proceeings of 33rd Annual Symposium on Foundations of Computer Science, pp. 659–669 (1992)
24. Shigeno, M., Saruwatari, Y., Matsui, T.: An algorithm for fractional assignment problems. Discrete Appl. Math. **56**(2–3), 333–343 (1995)
25. Skiscim, C.C., Palocsay, S.W.: The complexity of minimum ratio spanning tree problems. J. Glob. Optim. **30**, 335–346 (2004)