Generalized Fractional Programming With User Interaction

Ş. I. Birbil, J. B. G. Frenk and S. Zhang

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2004-033-LIS
Publication	May 2004
Number of pages	27
Email address corresponding author	sibirbil@few.eur.nl
Address	Erasmus Research Institute of Management (ERIM) Rotterdam School of Management / Rotterdam School of Economics Erasmus Universiteit Rotterdam P.O. Box 1738 3000 DR Rotterdam, The Netherlands Phone: +31 10 408 1182 Fax: +31 10 408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

BIBLIOGRAPHIC DATA AND CLASSIFICATIONS		
Abstract	The present paper proposes a new approach to solve generalized fractional programming problems through user interaction. Capitalizing on two alternatives, we review the Dinkelbach-type methods and set forth the main difficulty in applying these methods. In order to cope with this difficulty, we propose an approximation approach that can be controlled by a predetermined parameter. The proposed approach is promising particularly when a decision maker is involved in the solution process and agrees upon finding an effective but nearoptimal value in an efficient manner. The decision maker is asked to decide the parameter and our analysis shows how good is the value found by the approximation corresponding to this parameter. In addition, we present several observations that may be suitable for boosting up the performance of the proposed approach. Finally, we support our discussion through extensive numerical experiments.	
Library of Congress Classification (LCC)	5001-6182	Business
	5201-5982	Business Science
	HB 143	Mathematical Programming
Journal of Economic Literature (JEL)	M	Business Administration and Business Economics
	M 11 R 4	Production Management Transportation Systems
	C 61	Optimization Techniques, programming
European Business Schools Library Group (EBSLG)	85 A	Business General
	260 K 240 B	Logistics Information Systems Management
	255 B	Programming
Gemeenschappelijke Onderwerpsontsluiting (GOO)		
Classification GOO	85.00	Bedrijfskunde, Organisatiekunde: algemeen
	85.34 85.20	Logistiek management Bestuurlijke informatie, informatieverzorging
	31.80	Toepassingen van de wiskunde
Keywords GOO	Bedrijfskunde / Bedrijfseconomie	
	Bedrijfsprocessen, logistiek, management informatiesystemen	
	Wiskundige programmering, gebruikersinterfaces, optimalisatie	
Free keywords	Generalized fractional programming, user interaction, approximation approach, error analysis, performance improvement	

GENERALIZED FRACTIONAL PROGRAMMING WITH USER INTERACTION

Ş. İ. Birbil[†], J. B. G. Frenk[‡] and S. Zhang[♮*]

[†]*Erasmus Research Institute of Management, Erasmus University Rotterdam*
Postbus 1738, 3000 DR Rotterdam, The Netherlands.

[‡]*Econometrics Institute, Erasmus University Rotterdam*
Postbus 1738, 3000 DR Rotterdam, The Netherlands.

[♮]*Systems Engineering and Engineering Management*
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong.

ABSTRACT. The present paper proposes a new approach to solve generalized fractional programming problems through user interaction. Capitalizing on two alternatives, we review the Dinkelbach-type methods and set forth the main difficulty in applying these methods. In order to cope with this difficulty, we propose an approximation approach that can be controlled by a predetermined parameter. The proposed approach is promising particularly when a decision maker is involved in the solution process and agrees upon finding an effective but near-optimal value in an efficient manner. The decision maker is asked to decide the parameter and our analysis shows how good is the value found by the approximation corresponding to this parameter. In addition, we present several observations that may be suitable for boosting up the performance of the proposed approach. Finally, we support our discussion through extensive numerical experiments.

Keywords. Generalized fractional programming, user interaction, approximation approach, error analysis, performance improvement

1 Introduction

Decision makers in different application areas demand effective and easy-to-use solution methods for their optimization problems. In many circumstances, they are ready to welcome a new approach even though this approach may yield a value *close-enough*

[*]Research supported by Hong Kong RGC Earmarked Grants CUHK4233/01E and CUHK4174/03E.

to the actual (or theoretical) optimal value. One way of satisfying this demand is incorporating approximation approaches into well-known solution methods. To our advantage, the precision of an approximation approach depends on predefined parameters. Therefore through user interaction, the tradeoff between the quality of the results and the simplification of the problem can be controlled.

In this paper, we focus on solving a generalized fractional programming (GFP) problem by an approximation approach. GFP problems arise in many real-life applications. One of the earliest example dates back to the time of von Neumann when he used GFP in modeling an expanding economy [von Neumann, J., 1945]. Another important application is within the field of multi-objective programming. In this case, the main goal is to minimize the maximum deviation of the fractional objectives, such as input-output ratios, from their target values [Gugat, M, 1996b, Kornbluth, J.S. and R.E. Steuer, 1981, Jagannathan, R, 1985, Ignizio, J.P., 1976]. Also in numerical analysis, GFP methods are used for solving rational Chebyshev approximations [Barrodale, I., Powell, M.J.D. and F.D.K. Roberts, 1972, Cheney, E.W. and H.L. Loeb, 1961, Gugat, M, 1997, 1996a]. In order to study the theoretical properties of GFP problems, researchers made use of the tools from duality theory, nonlinear programming and numerical analysis. Naturally, the study of the theoretical properties led to the development of various solution methods [Crouzèix, J.P., Ferland, J.A. and S. Schaible, 1985, Crouzèix, J.P., Ferland, J.A. and S. Schaible, 1986, Crouzèix, J.P. and J.A. Ferland, 1991, Barros, A.I., Frenk, J.B.G., Schaible, S. and S. Zhang, 1996b,a, Barros, A, 1998, Gugat, M, 1996b, 1998, Roubi, A, 2000]. The majority of these methods are the variations of the approach suggested in the pioneering work of Crouzèix *et. al.* [1985]. In the literature, these methods are also known as Dinkelbach-type methods [Dinkelbach, W., 1967], and although not immediately clear, it can be shown that these methods are actually cutting plane methods [Barros, A.I. and J.B.G. Frenk, 1995]. Dinkelbach-type methods are based on the idea of solving a sequence of auxiliary problems so that the solutions of the auxiliary problems converge to the solution of the GFP problem. These methods are very powerful, however their performances heavily depend on the effective solution of the auxiliary problems. In general the auxiliary problems are nonsmooth and nonlinear. Therefore, they might pose a major difficulty in solving the GFP problems effectively and easily.

The main idea of this paper is replacing the difficult auxiliary problems in Dinkelbach-type methods with relatively simple approximations. Approximations of these sort yield ε-optimal (close-enough) values to the auxiliary problems. Consequently, we will analyze the error committed by replacing the auxiliary problems with their approximated counterparts. Especially for large size problems, the task of speeding up the algorithms plays an important role. Therefore, we will also propose several approaches that may lead to a performance improvement. During our analysis, we will work with a generic approximation function since the choice of the approximation function does not alter our results. In order to illustrate our idea, we will then give two particular approximation approaches that will be also used in an extensive numerical results section. In terms of our main contribution, we believe that our work provides a useful reference for a decision maker who is willing to solve a GFP problem effectively at the cost of finding inexact but controlled solutions.

2 Generalized Fractional Programming and The Dinkelbach Method

In this section we give an overview on generalized fractional programming and the most common solution approach, namely, the Dinkelbach method. Before proceeding to our subsequent analysis, let us first introduce the generalized fractional programming problem

$$\lambda^* := \inf_{x \in X} \max_{i \in I} \left\{ \frac{f_i(x)}{g_i(x)} \right\} \tag{P}$$

where $X \subseteq \mathbb{R}^n$ is the feasible region, $I := \{1, 2, \cdots, m\}$ a finite index set and for all $i \in I$, $f_i : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ are the problem functions. In addition, if we define the function $p : X \to (-\infty, \infty]$ by

$$p(x) := \max_{i \in I} \left\{ \frac{f_i(x)}{g_i(x)} \right\}, \tag{2.1}$$

then the optimization problem (P) can be written as

$$\lambda^* := \inf_{x \in X} \; p(x). \tag{2.2}$$

In the sequel we denote the optimal solution of problem (P) by x^*.

Introducing now the functions $g_{\min} : \mathbb{R}^n \to \mathbb{R}$ by

$$g_{\min}(x) := \min_{i \in I} \; g_i(x), \tag{2.3}$$

it is necessary to impose the following assumption to avoid pathological instances of problem (P).

Assumption 2.1

$$g_{\min}(x) > 0 \textit{ for every } x \in X. \tag{2.4}$$

The main idea of the Dinkelbach method is to provide a solution to the difficult optimization problem (P) by solving a sequence of relatively simple parameterized problems. In this overview section, we focus on two main Dinkelbach-type approaches. The first approach is based on solving constrained subproblems, whereas the second approach aims at reformulating the parameterized subproblems as unconstrained nonlinear optimization problems.

We start with the first approach by defining for every $i \in I$, the functions $\phi_i : \mathbb{R}^{n+1} \to \mathbb{R}$ given by

$$\phi_i(\lambda, x) := f_i(x) - \lambda g_i(x) \tag{2.5}$$

and for every $(\lambda, x) \in \mathbb{R}^{n+1}$, the function

$$\phi(\lambda, x) := \max_{i \in I} \phi_i(\lambda, x). \tag{2.6}$$

This leads to the associated parametric *constrained* optimization problem

$$\phi(\lambda) := \inf_{x \in X} \phi(\lambda, x). \tag{P_λ}$$

After minor modifications of the proofs in [Crouzèix, J.P., Ferland, J.A. and S. Schaible, 1985], one can show the following important results.

Lemma 2.1 $\phi(\lambda) \geq 0$ *if and only if* $-\infty < \lambda \leq \lambda^*$.

Lemma 2.2 *The following conditions are equivalent.*

1. *The optimization problem (P) has an optimal solution.*
2. *The set $\mathcal{V}$ defined by*

   $$\mathcal{V} := \{\lambda \in \mathbb{R} : \text{ Problem } (P_\lambda) \text{ has an optimal solution and } \phi(\lambda) = 0\}$$

   is nonempty.
3. *The set $\mathcal{V}$ is a singleton with $\mathcal{V} = \{\lambda^*\}$.*

We now discuss the unconstrained approach. In this case the feasible region X is given by

$$X := \{x \in \mathbb{R}^n : h_j(x) \leq 0, j \in J\}, \tag{2.7}$$

where $J := \{1, 2, \cdots, p\}$ is a finite index set and $h_j : \mathbb{R}^n \to \mathbb{R}$ for all $j \in J$ with $h(x) := (h_1(x), \cdots, h_p(x))^T$. Clearly, this form of the feasible set is a commonly used form for numerical purposes.

In addition to Assumption 2.1, we also assume that the following condition holds.

Assumption 2.2 *The function $x \to h_j(x)$, $j \in J$ are convex and the set X has a nonempty interior,* i.e., *there exist some $x_0 \in \mathbb{R}^n$ such that*

$$\max_{j \in J} \; h_j(x_0) < 0. \tag{2.8}$$

As before, we start by introducing the parametric function $\psi_i : \mathbb{R}^{n+1} \to \mathbb{R}$ given by

$$\psi_i(\lambda, x) := \begin{cases} f_i(x) - \lambda g_i(x), & \text{for } i \in I \\ h_i(x), & \text{for } i \in J \end{cases} \tag{2.9}$$

and consider for $(\lambda, x) \in \mathbb{R}^{n+1}$, the function

$$\psi(\lambda, x) := \max_{i \in I \cup J} \psi_i(\lambda, x) \tag{2.10}$$

with the parametric *unconstrained* optimization problem

$$\psi(\lambda) := \inf_{x \in \mathbb{R}^n} \psi(\lambda, x). \tag{$\bar{P}_\lambda$}$$

It is shown in [Roubi, A, 2000, Addou, A. and A. Roubi] that the following versions of the previous two lemmas hold.

Lemma 2.3 $\psi(\lambda) \geq 0$ *if and only if* $-\infty < \lambda \leq \lambda^*$.

Lemma 2.4 *The following conditions are equivalent.*

1. *The optimization problem (P) has an optimal solution.*
2. *The set $\mathcal{W}$ defined by*

   $$\mathcal{W} := \{\lambda \in \mathbb{R} : \text{ Problem } (\bar{P}_\lambda) \text{ has an optimal solution and } \psi(\lambda) = 0\}$$

   is nonempty.
3. *The set $\mathcal{W}$ is a singleton with $\mathcal{W} = \{\lambda^*\}$.*

Using the above results, we can discuss the computational procedures for both constrained and unconstrained cases. In order to apply the Dinkelbach-type approach corresponding to the constrained case, we always assume that problem (P) and for any $\lambda \geq \lambda^*$, problems (P_λ) are solvable. The general scheme of the constrained case has now the form given in Algorithm 2.1.

Algorithm 2.1 Constrained Case

```
1. Select $x_0 \in X$ and set $k := 1$.
2. Set $\lambda_k := p(x_{k-1})$.
3. Determine an element $x_k$ of the set $\{x \in X : \phi(\lambda_k, x) < 0\}$. If such an element
   does not exist, then return $(\lambda_k, x_{k-1})$, else set $k := k + 1$ and return to Step 2.
```

As in the constrained case, we assume that problem (P) and for any $\lambda \geq \lambda^*$, problems ($\bar{P}_\lambda$) are solvable. Observe that if $\psi(\lambda_k, x_k) < 0$, then it follows automatically that $h_j(x_k) < 0$ for every $j \in J$ and hence, $x_k \in X$. The algorithm for the unconstrained Dinkelbach-type approach has now the form given in Algorithm 2.2.

Algorithm 2.2 Unconstrained Case

```
1. Select $x_0 \in X$ and set $k := 1$.
2. Set $\lambda_k := p(x_{k-1})$.
3. Determine an element $x_k$ of the set $\{x \in \mathbb{R}^n : \psi(\lambda_k, x) < 0\}$. If such an
   element does not exist, then return $(\lambda_k, x_{k-1})$, else set $k := k + 1$ and return to
   Step 2.
```

The geometrical interpretation of Algorithm 2.1 is illustrated in Figure 2. Clearly, for any $x_k \in \{x \in X : \phi(\lambda, x) < 0\}$ we have

$$\phi(\lambda) \leq \phi(\lambda, x_k). \tag{2.11}$$

Hence, at the k^{th} iteration of Algorithm 2.1, the nonconvex function ϕ is approximated from above by the polyhedral convex function $\lambda \to \phi(\lambda, x_k)$. As a direct consequence of Lemma 2.2, our main goal is to compute λ that satisfi es $\phi(\lambda) = 0$. Instead of this ambitious goal, we solve the approximating equation $\phi(\lambda, x_k) = 0$, which leads to

$$\lambda = \max_{i \in I} \left\{ \frac{f_i(x_k)}{g_i(x_k)} \right\}. \tag{2.12}$$

Notice that this solution is indeed the next iteration in Algorithm 2.1 (Step 2). A similar interpretation can be given for Algorithm 2.2, where the function ψ is approximated by the polyhedral convex function $\lambda \to \psi(\lambda, x_k)$.

In Step 3 of both algorithms we suggest to solve membership problems instead of solving the optimization problems (P_λ) and ($\bar{P}_\lambda$). The reason for this suggestion is given by the following observation. Consider the constrained approach and suppose at

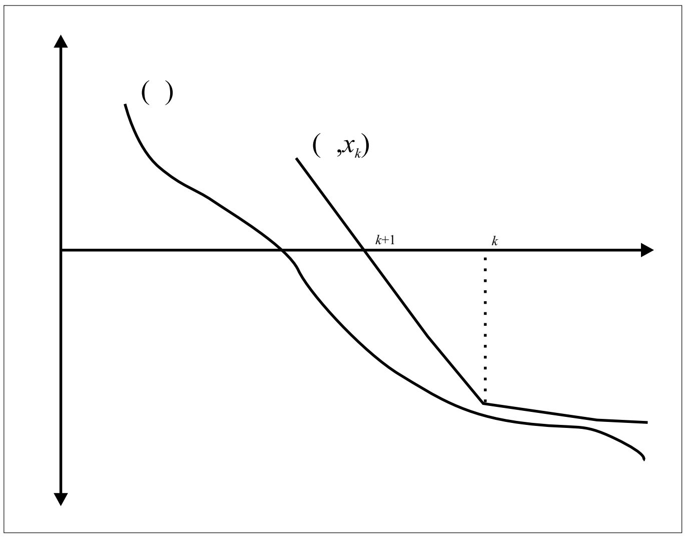

Figure 1: A typical iteration of Algorithm 2.1.

iteration k we compute the optimal solution x_k^* of problem (P_{λ_k}) and at the same time we choose a different vector x_k satisfying $\phi(\lambda_k, x_k) < 0$. It might now happen that the value $\lambda_{k+1} := p(x_k)$ is closer to λ^* than the value λ_{k+1} derived from x_k^*. An instance showing this behavior is given in Example 2.1. This suggests that investing time in solving problem (P_λ) to optimality might not always be the best strategy. On the other hand, to determine whether there exists an element of the set $\{x \in X : \phi(\lambda_k, x) < 0\}$ or decide whether this set is empty, it is most natural to solve problem (P_{λ_k}), and so x_k^* becomes a natural candidate.

Example 2.1 *Suppose we want to solve*

$$\lambda^* = \inf_{x\in[0,10]} \max\{\frac{-11x+1}{2x+2}, \frac{-7x+2}{4x+1}, \frac{3x-2}{16x+3}\}.$$

Let $x_0 = 1$ *then* $\lambda_1 = p(x_0) = 1/19$. *If we solve problem (*P_{λ_1}*), then we find the optimal solution* $x_1^* = 38/89$ *with the objective function value* $\phi(\lambda_1, x_1^*) = -1.21$. *Moreover, at the next iteration, we compute* $\lambda_2^* := p(x_1^*) = -0.068$. *Instead of the optimal solution, if we select* $x_1 = 30/89$ *yielding* $\phi(\lambda_1, x_1) = -0.48$*, then the corresponding value of* λ *at the next iteration becomes* $\lambda_2 := p(x_1) = -0.1178$*, which is less than* λ_2^*.

This observation may also become useful whenever the optimization routine for solving the auxiliary problems requires a considerable effort. In this case, one may stop as soon as the optimization routine finds a feasible, but not necessarily optimal, point x that solves the membership problem. As Lemma 2.1 and Lemma 2.3 show, from a practical point of view, we only need a decreasing sequence of λ_k values.

3 The Approximation Approach

Notice that in both Algorithm 2.1 and Algorithm 2.2, the membership problems at Step 3 can be solved by reformulating the problems as nonlinear optimization problems, *i.e.*, one needs to solve (approximately, see Example 2.1), in the constrained case, the auxiliary optimization problem

$$\min_{x\in X} \max\{f_1(x) - \lambda g_1(x), \cdots, f_m(x) - \lambda g_m(x)\} \tag{3.1}$$

and in the unconstrained case, the problem

$$\min_{x\in\mathbb{R}^n} \max\{f_1(x) - \lambda g_1(x), \cdots, f_m(x) - \lambda g_m(x), h_1(x), \cdots, h_p(x)\}. \tag{3.2}$$

Although the $\max$ function is a convex increasing function, unfortunately it is nonsmooth. Therefore, we still encounter a nonsmooth objective function in the auxiliary optimization problems. One promising solution to this obstacle is to approximate the $\max$ function by a smooth alternative. Consider now a smooth, convex and increasing function $\Pi_s : \mathbb{R}^s \to \mathbb{R}$ that satisfi es

$$\lim_{\varepsilon\downarrow 0} \varepsilon\Pi_s(\frac{y}{\varepsilon}) = \max\{y_1, \cdots, y_s\} \tag{3.3}$$

and

$$0 \le \varepsilon\Pi_s(\frac{y}{\varepsilon}) - \max\{y_1, \cdots, y_s\} \le \beta_s(\varepsilon) \tag{3.4}$$

for every $y \in \mathbb{R}^s$. As a direct consequence of the above two relations, the function $y \to \varepsilon\Pi_s(\frac{y}{\varepsilon})$ becomes a good candidate for approximating the nonsmooth objective functions of the auxiliary problems. Let us now give two examples that will also be used in the numerical results section.

Example 3.1 *It is easy to check that the entropy-type function*

$$\Pi_s(y) = \log\left(\sum_{i=1}^{s} e^{y_i}\right)$$

is convex, differentiable and increasing [Fang, S.-C., Rajasekera, J. R. and H.-S. J. Tsao, 197]. Moreover, it satisfies relation (3.3) and relation (3.4) with $\beta_s(\varepsilon) = \varepsilon\log s$ *[Ben-Tal, A. and M. Teboulle, 1989].*

Example 3.2 *In some problems, entropy-type function in Example 3.1 may create overflow problems because of the exponential function. Recently Birbil* et. al. *proposed a more stable approximation method that aims at extending two dimensional approximation functions to higher dimensions by recursion [Birbil, Ş.İ., Fang, S.-C., Frenk, J.B.G. and S. Zhang, 2002]. The main idea of the method is based on* *the observation that the higher dimensional* $\max$ *function can be written recursively as follows*

$$\max\{y_1, \cdots, y_s\} = \max\{\max\{y_1, \cdots, y_q\}, \max\{y_{q+1}, \cdots, y_s\}\},$$

where $1 < q < s$*. Following this observation, the method proposed by Birbil* et. al. *provides a twice differentiable convex increasing function* Π_s *that satisfies relation (3.3). In this case, the error term in (3.4) is given by* $\beta_s(\varepsilon) = \frac{\varepsilon}{2}(\log_2(s-1)+1)\Pi_2(0)$*, where* $\Pi_2 : \mathbb{R}^2 \to \mathbb{R}$ *is the base function that is used for approximating the two dimensional* $\max$ *function,* i.e., $\lim_{\varepsilon \downarrow 0} \varepsilon \Pi_2(y/\varepsilon) = \max\{y_1, y_2\}$.

3.1 Constrained Case with Approximation

We fi rst start with the constrained case of the Dinkelbach-type method. Given $\varepsilon > 0$, defi ne the function $\phi_\varepsilon : \mathbb{R} \times X \to \mathbb{R}$ by

$$\phi_\varepsilon(\lambda, x) := \varepsilon \Pi_m \left(\frac{f_1(x) - \lambda g_1(x)}{\varepsilon}, \cdots, \frac{f_m(x) - \lambda g_m(x)}{\varepsilon} \right). \tag{3.5}$$

Hence, instead of problem (3.1), we can consider the smooth optimization problem

$$\phi_\varepsilon(\lambda) := \min_{x \in X} \phi_\varepsilon(\lambda, x). \tag{P_λ^ε}$$

To derive the difference in the optimal objective function values between optimization problems (P_λ) and (P_λ^ε) we observe by relation (3.4) and $x(\lambda)$ belonging to $\arg\min_{x \in X} \phi_\varepsilon(\lambda, x)$ that

$$\phi_\varepsilon(\lambda) - \phi(\lambda) \leq \phi_\varepsilon(\lambda, x(\lambda)) - \phi(\lambda, x(\lambda)) \leq \beta_m(\varepsilon). \tag{3.6}$$

By the same relation, we also obtain that $\phi_\varepsilon(\lambda, x) \geq \phi(\lambda, x)$ for every $x \in X$ and $\lambda \in \mathbb{R}$ and this shows

$$\phi_\varepsilon(\lambda) \geq \phi(\lambda) \tag{3.7}$$

for every $\lambda \in \mathbb{R}$. Hence it follows by relations (3.6) and (3.7) that

$$0 \leq \phi_\varepsilon(\lambda) - \phi(\lambda) \leq \beta_m(\varepsilon) \tag{3.8}$$

for every $\lambda \in \mathbb{R}$. Therefore at the expense of some limited error, we consider the following *approximated* Dinkelbach-type method.

Algorithm 3.1 Constrained Case with Approximation

```
1. Select $x_0 \in X$ and set $k := 1$.
2. Set $\lambda_k := p(x_{k-1})$.
3. Determine an element $x_k$ of the set $\{x \in X : \phi_\varepsilon(\lambda_k, x) < 0\}$. If such an
   element does not exist, then return $(\lambda_k, x_{k-1})$, else set $k := k + 1$ and return to
   Step 2.
```

Before discussing the properties of Algorithm 3.1, we first introduce the nonempty set-valued mappings $S : \mathbb{R} \times X \rightrightarrows I$ and $L : X \rightrightarrows I$ given by

$$S(\lambda, x) := \{i \in I : \phi_i(\lambda, x) = \phi(\lambda, x)\}$$

and

$$L(x) := \{i \in I : \frac{f_i(x)}{g_i(x)} = p(x)\}.$$

It is now easy to derive the following upper and lower bounds on the difference $\lambda_{k+1} - \lambda_k$ for every $k < k_0$, with $k_0 \leq \infty$ being the total number of iterations spent in Algorithm 3.1 before meeting the stopping condition.

Lemma 3.1 *The sequence λ_k, $k < k_0$ generated by Algorithm 3.1 is strictly decreasing and satisfies*

$$\frac{\phi(\lambda_k, x_k)}{\max_{i \in S(\lambda_k, x_k)} g_i(x_k)} \leq \lambda_{k+1} - \lambda_k \leq \frac{\phi(\lambda_k, x_k)}{\min_{i \in L(x_k)} g_i(x_k)}.$$

Proof. For every $k < k_0$ we know by relation (3.4) and (3.5) that $\phi(\lambda_k, x_k) \leq \phi_\varepsilon(\lambda_k, x_k) < 0$, and since $\phi(\lambda_{k+1}, x_k) = 0$ and $\lambda \to \phi(\lambda, x_k)$ is strictly decreasing, this yields $\lambda_{k+1} < \lambda_k$. We now verify the above inequalities. By the definition of λ_{k+1}, it follows for every i^* belonging to $\arg\max_{i \in S(\lambda_k, x_k)} g_i(x_k)$ satisfying $g_{i^*}(x_k) = \max_{i \in S(\lambda_k, x_k)} g_i(x_k)$ that

$$\lambda_{k+1} - \lambda_k \geq \frac{f_{i^*}(x_k) - \lambda_k g_{i^*}(x_k)}{g_{i^*}(x_k)} = \frac{\phi(\lambda_k, x_k)}{\max_{i \in S(\lambda_k, x_k)} g_i(x_k)}.$$

Also for every i_* belonging to $\arg\min_{i \in L(x_k)} g_i(x_k)$ satisfying $g_{i_*}(x_k) = \min_{i \in L(x_k)} g_i(x_k)$, we obtain by using again the definition of λ_{k+1} that

$$\lambda_{k+1} - \lambda_k = \frac{f_{i_*}(x_k) - \lambda_k g_{i_*}(x_k)}{g_{i_*}(x_k)} \leq \frac{\phi(\lambda_k, x_k)}{\min_{i \in L(x_k)} g_i(x_k)},$$

and this completes the proof. □

If at the kth step of Algorithm 3.1, we observe that $\phi_\varepsilon(\lambda_k, x_k) \leq \delta$ for some $\delta < 0$, and it holds that $g^* := \max_{x\in X} \max_{i\in I} g_i(x) < \infty$, then it follows by Lemma 3.1 and relation (3.4) that

$$\lambda_{k+1} - \lambda_k \leq \frac{\phi_\varepsilon(\lambda_k, x_k)}{\min_{i\in L(x_k)} g_i(x_k)} \leq \frac{-\delta}{g^*}. \tag{3.9}$$

Hence in this case the improvement in the objective function value is at least $-\delta/g^*$, and in most cases it can be expected that this is a conservative bound. In addition to the number of iterations, we are also interested in the quality of the solution generated by Algorithm 3.1. This means that one likes to derive an upperbound on the difference $\lambda^* - \lambda_{k_0}$ with k_0 being the iteration at which the stopping condition applies. First we need the following lemma.

Lemma 3.2 *If $g_* := \inf_{x\in X} g_{\min}(x) > 0$, then it follows for every $k < k_0$ that*

$$0 \geq \lambda^* - \lambda_{k+1} \geq \frac{\phi_\varepsilon(\lambda_{k+1}) - \beta_m(\varepsilon)}{g_*}.$$

Proof. It is obvious that $0 \geq \lambda^* - \lambda_{k+1}$ and to prove the reverse inequality, we observe for every $k < k_0$, x^* an optimal solution of optimization problem (P) and i^* belonging to $\arg\max_{i\in S(\lambda_{k+1},x^*)} g_i(x^*)$ that

$$0 \geq \lambda^* - \lambda_{k+1} \geq \frac{f_{i^*}(x^*) - \lambda_{k+1} g_{i^*}(x^*)}{g_{i^*}(x^*)} = \frac{\phi(\lambda_{k+1}, x^*)}{\max_{i\in S(\lambda_{k+1},x^*)} g_i(x^*)}. \tag{3.10}$$

This shows that $\phi(\lambda_{k+1}, x^*) \leq 0$, and since $\max_{i\in S(\lambda_{k+1},x^*)} g_i(x^*) \geq g_*$, by using relation (3.10) we obtain that

$$\lambda^* - \lambda_{k+1} \geq \frac{\phi(\lambda_{k+1}, x^*)}{\max_{i\in S(\lambda_{k+1},x^*)} g_i(x^*)} \geq \frac{\phi(\lambda_{k+1}, x^*)}{g_*}. \tag{3.11}$$

Applying now relation (3.4) shows that

$$\phi(\lambda_{k+1}, x^*) \geq \phi_\varepsilon(\lambda_{k+1}, x^*) - \beta_m(\varepsilon) \geq \phi_\varepsilon(\lambda_{k+1}) - \beta_m(\varepsilon),$$

and by relation (3.11) the desired result follows. □

Notice that the stopping condition in Algorithm 3.1 is satisfi ed if $\phi_\varepsilon(\lambda_{k_0}) \geq 0$ for fi nite k_0. Therefore, in combination with Lemma 3.2, we have

$$\lambda^* - \lambda_{k_0} \geq \frac{-\beta_m(\varepsilon)}{g_*}. \tag{3.12}$$

Finally we observe in this subsection that the optimization problem $(P^{\varepsilon}_{\lambda})$ is a smooth convex optimization problem if $\lambda > 0$, X is a convex set defined by differentiable convex inequalities and the functions f_i, (g_i) for all $i \in I$ are differentiable and convex (concave). In this case, the function $x \to g_{\min}(x)$ is also a concave function and hence, the value g_* can be computed by solving a concave minimization problem. Such a problem is in general NP-hard [Horst, R, Pardalos, P.M and N.V. Thoai, 1995].

3.2 Unconstrained Case with Approximation

Similar to the previous section, we start with introducing an approximation function $\psi_\varepsilon : \mathbb{R}^{n+1} \to \mathbb{R}$ defined by

$$\psi_\varepsilon(\lambda, x) := \varepsilon\Pi_{m+p}\left(\frac{f_1(x) - \lambda g_1(x)}{\varepsilon}, \cdots, \frac{f_m(x) - \lambda g_m(x)}{\varepsilon}, \frac{h_1(x)}{\varepsilon}, \cdots, \frac{h_p(x)}{\varepsilon}\right). \tag{3.13}$$

We can now consider the smooth optimization problem

$$\psi_\varepsilon(\lambda) := \min_{x\in\mathbb{R}^n} \psi_\varepsilon(\lambda, x). \tag{$\bar{P}^{\varepsilon}_{\lambda}$}$$

In this case the counterpart of relation (3.8) becomes

$$0 \le \psi_\varepsilon(\lambda) - \psi(\lambda) \le \beta_{m+p}(\varepsilon) \tag{3.14}$$

for every $\lambda \in \mathbb{R}$. The adapted version of Algorithm 2.2 with the approximation is now given in Algorithm 3.2.

Algorithm 3.2 Unconstrained Case with Approximation

```
1. Select $x_0 \in X$ and set $k := 1$.
2. Set $\lambda_k := p(x_{k-1})$.
3. Determine an element $x_k$ of the set $\{x \in \mathbb{R}^n : \psi_\varepsilon(\lambda_k, x) < 0\}$. If such an
   element does not exist, then return $(\lambda_k, x_{k-1})$, else set $k := k + 1$ and return to
   Step 2.
```

Observe that if $\psi_\varepsilon(\lambda_k, x) < 0$, then $\psi(\lambda_k, x) \le \psi_\varepsilon(\lambda_k, x) < 0$, and hence x belongs to X. As before we are now interested in the difference $\lambda_{k+1} - \lambda_k$ with the sequence of λ_k values generated by Algorithm 3.2. Again, k_0 denotes the total number of iterations spent in Algorithm 3.2 before meeting the stopping condition.

Lemma 3.3 *The sequence λ_k, $k < k_0$ generated by Algorithm 3.2 is strictly decreasing and satisfies*

$$\lambda_{k+1} - \lambda_k \le \frac{\psi(\lambda_k, x_k)}{\min_{i\in L(x_k)} g_i(x_k)}.$$

Proof. Redoing the last part of the proof of Lemma 3.1 leads to

$$\lambda_{k+1} - \lambda_k \leq \frac{\phi(\lambda_k, x_k)}{\min_{i \in L(x_k)} g_i(x_k)}$$

and using now $\phi(\lambda_k, x_k) \leq \psi(\lambda_k, x_k)$ yields the desired result. □

In order to give an upperbound on the error $\lambda_k - \lambda^*$ we will use the strong duality theorem for Lagrangean duality. To apply this result we need to impose some well known additional properties on the functions f_i and g_i.

Assumption 3.1 *The following conditions hold:*

1. *The functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $i \in I$ are convex on $\mathbb{R}^n$ and nonnegative on X. Moreover, the functions $g_i : \mathbb{R}^n \to \mathbb{R}$, $i \in I$ are concave on $\mathbb{R}^n$ and nonnegative on X.*
2. *The functions $h_j : \mathbb{R}^n \to \mathbb{R}$, $j \in J$ are convex and there exists some $x_0 \in \mathbb{R}^n$ such that $\alpha_0 := \max_{j \in J} h_j(x_0) < 0$.*

If the functions f_i, $i \in I$ are convex and g_i, $i \in I$ are concave, this does not necessarily imply that the ratio of these two functions is convex. Consequently, the generalized fractional programming problem (P) is not necessarily a convex optimization problem.

Introducing now for every $\alpha_0 < \alpha < 0$ the nonempty perturbed feasible region

$$X(\alpha) := \{x \in \mathbb{R}^n : h_j(x) \leq \alpha, \ \ j \in J\}$$

we defi ne for $\alpha_0 < \alpha \leq 0$ the following perturbed optimization problem

$$v(\alpha) := \min_{x \in X(\alpha)} p(x).$$

By the defi nition of $v(\alpha)$, it is clear that $\lambda^* = v(0)$ and $0 \leq v(\alpha) \leq \lambda^*$ for $\alpha_0 < \alpha < 0$. In the next lemma we give a lower bound on $\lambda^* - v(\alpha)$.

Lemma 3.4 *If $g_* := \inf_{x \in X} g_{\min}(x) > 0$ and Assumption 3.1 holds, then it follows that*

$$\lambda^* - v(\alpha) \geq \frac{\mu_*^\top(\alpha)\alpha}{g_*}$$

with $\mu_*^\top(\alpha)$ the optimal Lagrangean multiplier associated with the convex optimization problem

$$\min_{x \in X(\alpha)} \phi(v(\alpha), x).$$

Proof. By the strong duality theorem (cf.[Bertsekas, D.P., 1995]) we obtain for every $\alpha_0 < \alpha < 0$ that

$$\begin{aligned}\phi(v(\alpha)) &= \min_{x\in X}\phi(v(\alpha),x)\\ &= \max_{\mu\geq 0}\min_{x\in\mathbb{R}^n}\phi(v(\alpha),x)+\mu^\top(h(x)-\alpha)+\mu^\top\alpha.\end{aligned}$$

This shows applying Lemma 2.2 and using again the strong duality theorem that

$$\begin{aligned}\phi(v(\alpha)) &\geq \min_{x\in\mathbb{R}^n}\phi(v(\alpha),x)+\mu_*^\top(\alpha)(h(x)-\alpha)+\mu_*^\top(\alpha)\alpha \\ &= \max_{\mu\geq 0}\min_{x\in\mathbb{R}^n}\phi(v(\alpha),x)+\mu^\top(h(x)-\alpha)\ +\mu_*^\top(\alpha)\alpha\\ &= \min_{x\in X(\alpha)}\phi(v(\alpha),x)+\mu_*^\top(\alpha)\alpha=\mu_*^\top(\alpha)\alpha.\end{aligned} \tag{3.15}$$

By relation (3.15) it follows for every $x \in X$ that there exist some $i \in I$ satisfying

$$f_i(x)-v(\alpha)g_i(x)\geq \mu_*^\top(\alpha)\alpha.$$

and this shows the desired result. □

Using Lemma 3.4 one can now show the following result.

Lemma 3.5 *Let Assumption 3.1 hold and assume $g_* > 0$ and $2\kappa > \alpha_0$ with $\kappa := -\beta_{m+p}(\epsilon)$. If the algorithm stops at iteration k_0, then the vector x_{k_0-1} belongs to X and*

$$0\geq\lambda^*-\lambda_{k_0}\geq\frac{2\kappa\mu_*^\top(2\kappa e)e+\kappa}{g_*}$$

with $\mu_*^\top(2\kappa e)$ the optimal Lagrangean multiplier associated with the convex optimization problem

$$\min_{x\in X(2\kappa e)}\phi(v(2\kappa e),x).$$

Proof. If the algorithm stops at iteration k_0,then clearly x_{k_0-1} belongs to X and so we obtain $0\geq\lambda^*-\lambda_{k_0}$. Moreover, by the stopping rule we know that $\psi_\epsilon(\lambda_{k_0})\geq 0$ and this implies for every $x\in\mathbb{R}^n$ that $\psi_\epsilon(\lambda_{k_0},x)\geq 0$. Applying now relation (3.4) and (3.13) yields for every $x\in\mathbb{R}^n$ that

$$\psi(\lambda_{k_0},x)\geq\kappa \tag{3.16}$$

By relation (3.16) and using the definition of $\psi(\lambda_{k_0},x)$ we obtain for every $x\in X(2\kappa e)$ that

$$\phi(\lambda_{k_0},x)=\psi(\lambda_{k_0},x)\geq\kappa.$$

This implies for every $x\in X(2\kappa)$ that $p(x)-\lambda_{k_0}\geq\kappa g_*^{-1}$,and so it follows that

$$v(2\kappa e)-\lambda_{k_0}\geq\kappa g_*^{-1}. \tag{3.17}$$

Applying now Lemma 3.4 and relation (3.17) yields

$$\begin{aligned}\lambda^* - \lambda_{k_0} &= \lambda^* - v(2\kappa e) + v(2\kappa e) - \lambda_{k_0}\\ &\geq (2\kappa\mu_*^\top(2\kappa e)e + \kappa)g_*^{-1}\end{aligned}$$

and this shows the desired inequality. □

Finally we note for the unconstrained case that if additionally the functions f_i and g_i are convex for all $i \in I$, then the optimization problem ($\bar{P}_\lambda^\varepsilon$) is a smooth unconstrained convex optimization problem as a direct consequence of Assumption 2.2 and Assumption 3.1.

4 Observations for Performance Improvement

Decision makers often face with large size problems and in many cases, the performance of an algorithm worsens when the size of the problem increases. So, improving the performance of an algorithm plays an important role for solving real problems. In this section, we present several observations that may lead to a performance improvement for the approximation approach discussed in the previous section.

4.1 Weaker Stopping Condition

Notice that in the constrained case of the proposed approximation method, the stopping rule, $\phi_\varepsilon(\lambda_k) \geq 0$ in Step 3 of Algorithm 3.1 can be replaced by a weaker condition $\phi_\varepsilon(\lambda_k) \geq -\delta$ for some $\delta > 0$. Implementing this observation may reduce the number of iterations. However, in this case the accuracy of the results may also decrease. The following result provides an upperbound on the number of subproblems to be solved and gives an upperbound on the difference between the optimal objective function value, λ^* and the objective function value computed after applying the weaker stopping condition.

Lemma 4.1 *Suppose $k_0 := \min\{k : \phi_\varepsilon(\lambda_k) \geq -\delta\}$, then it follows that*

$$k_0 \leq \frac{(\lambda_1 - \lambda^*)g^*}{\delta} \text{ and } \lambda_{k_0} - \lambda^* \leq \frac{\delta + \beta_m(\varepsilon)}{g_*}$$

Proof. By using $\lambda_k \geq \lambda^*$ for all $k < k_0$ and relation (3.9), it follows that

$$\lambda^* - \lambda_1 \leq \sum_{k=1}^{k_0-1}(\lambda_{k+1} - \lambda_k) \leq \frac{-\delta k_0}{g^*}.$$

Multiplying both sides by -1 yields the first part. Using now $\phi_\varepsilon(\lambda_k) \geq -\delta$ and Lemma 3.2 shows that

$$\lambda^* - \lambda_{k_0} \geq \frac{-\delta - \beta_m(\varepsilon)}{g_*}.$$

Again multiplying both sides by -1 shows the second part. □

A similar but less exciting discussion can be given for the unconstrained case after replacing the stopping rule, $\psi_\varepsilon(\lambda_k) \geq 0$ in Step 3 of Algorithm 3.2, by the weaker condition $\psi_\varepsilon(\lambda_k) \geq -\delta$ for some $\delta > 0$. Here we state the result without the proof.

Lemma 4.2 *Suppose* $k_0 := \min\{k : \psi_\varepsilon(\lambda_k) \geq -\delta\}$*, then it follows that*

$$k_0 \leq \frac{(\lambda_1 - \lambda^*)g^*}{\delta} \text{ and } \lambda_{k_0} - \lambda^* \leq \frac{-(2\kappa\mu_*^\top(2\kappa e)e + \kappa)}{g_*}.$$

4.2 Normalization

In order to improve the rate of convergence, Crouzèix *et. al.* proposed to replace problem (P_λ) at iteration k by

$$\min_{x\in X} \max_{i\in I} \left\{ \frac{f_i(x) - \lambda g_i(x)}{g_i(x_{k-1})} \right\}. \tag{4.1}$$

Crouzèix *et. al.* have shown that the rate of convergence becomes at least linear with the above modification and they have proved that whenever the sequence $\{x_k\}$ is convergent, the convergence is superlinear [Crouzèix, J.P., Ferland, J.A. and S. Schaible, 1986]. Following our previous results, one way to explain the intuition behind the normalization is as follows. Introduce for every $i \in I$ the functions $\bar{\phi}_i : \mathbb{R}^{2n+1} \to \mathbb{R}$ given by

$$\bar{\phi}_i(\lambda, x, y) := \frac{f_i(x) - \lambda g_i(x)}{g_i(y)}$$

and the function $\bar{\phi} : \mathbb{R}^{2n+1} \to \mathbb{R}$ defined by

$$\bar{\phi}(\lambda, x, y) := \max_{i\in I} \bar{\phi}_i(\lambda, x, y).$$

The parametric optimization problem associated with the above function is now given by

$$\bar{\phi}(\lambda, y) := \min_{x\in X} \bar{\phi}(\lambda, x, y).$$

Moreover, to define the approximation let

$$\bar{\phi}_\varepsilon(\lambda, x, y) := \varepsilon\Pi_m(\frac{\bar{\phi}_1(\lambda, x, y)}{\varepsilon}, \cdots, \frac{\bar{\phi}_m(\lambda, x, y)}{\varepsilon})$$

and consider the approximated parametric optimization problem

$$\bar{\phi}_\varepsilon(\lambda, y) = \min_{x \in X} \bar{\phi}_\varepsilon(\lambda, x, y).$$

The constrained algorithm with normalization as well as approximation is now given in Algorithm 4.1.

Algorithm 4.1 Constrained Case with Normalization and Approximation

```
1. Select $x_0 \in X$ and set $k := 1$.
2. Set $\lambda_k := p(x_{k-1})$.
3. Determine an element $x_k$ of the set $\{x \in X : \bar{\phi}_\varepsilon(\lambda_k, x, x_{k-1}) < 0\}$. If such an
   element does not exist, then return $(\lambda_k, x_{k-1})$, else set $k := k + 1$ and return to
   Step 2.
```

Before discussing the bounds on the improvement $\lambda_{k+1} - \lambda_k$ for this algorithm, we introduce the following set valued mapping $\bar{S} : \mathbb{R} \times X \times X \rightrightarrows \mathbb{R}$ given by

$$\bar{S}(\lambda, x, y) := \{i \in I : \bar{\phi}_i(\lambda, x, y) = \bar{\phi}(\lambda, x, y)\}.$$

Lemma 4.3 *Suppose Algorithm 4.1 stops at iteration k_0. The sequence λ_k, $k < k_0$ generated by Algorithm 4.1 is strictly decreasing and satisfies*

$$\bar{\phi}(\lambda_k, x_k, x_{k-1}) \min_{i \in \bar{S}} \frac{g_i(x_{k-1})}{g_i(x_k)} \leq \lambda_{k+1} - \lambda_k \leq \bar{\phi}(\lambda_k, x_k, x_{k-1}) \max_{i \in L(x_k)} \frac{g_i(x_{k-1})}{g_i(x_k)}$$

where $\bar{S} := \bar{S}(\lambda_k, x_k, x_{k-1})$.

Proof. A straightforward modifi cation of the proof of Lemma 3.1. □

Using similar discussions as in Lemma 3.2 and Lemma 4.1, we can also give results regarding the difference $\lambda^* - \lambda_{k_0}$ and the number of iterations required to satisfy $\bar{\phi}(\lambda_k, x_k, x_{k-1}) > -\delta$. Notice that if x_k is close to x_{k-1}, then the value of $\max_{i \in L(x_k)}(g_i(x_{k-1})/g_i(x_k))$ is approximately equal to 1. Therefore, in certain instances we expect that the convergence of the proposed method will improve considerably. Our numerical experiments in Section 5 supports this observation.

4.3 Additional Remarks

The membership problems in Step 3 of Algorithm 3.1 and Algorithm 3.2 are usually solved by reformulating them as optimization problems. As we have discussed before, the optimal solutions of these optimization problems do not necessarily yield a

λ_k value that is closer to λ^* than a λ_k value computed by a point that satisfies the condition in Step 3 (see Example 2.1). Therefore, at any iteration if the solution method for the optimization problems finds a point that satisfies the condition in Step 3, then the solution method can be terminated and the algorithm can move to the next iteration. Solving the optimization problems in Step 3 constitutes the major computational burden of Algorithm 3.1 and Algorithm 3.2. In this regard, terminating the solution methods before finding the optimal solution may save some computational effort.

An important advantage of Algorithm 3.2 over Algorithm 3.1 is that at each iteration one needs to solve an unconstrained membership problem rather than a constrained membership problem. Usually, solving unconstrained problems is easier than solving the constrained counterparts. However, in Algorithm 3.2 the constraints are also incorporated into the problem and only an approximated function is used. Thus, the solution of the unconstrained problem may be an infeasible point of the original problem. On the other hand, the solution of the constrained membership problem in Algorithm 3.1 is always feasible. In order to balance the tradeoff between the speed and feasibility, at early iterations one can start with solving the unconstrained problems and then switch to solving the constrained problems.

Recall that in Lemma 3.1 and Lemma 3.3, the differences between λ_{k+1} and λ_k are given by

$$\lambda_k - \lambda_{k+1} \geq \frac{-\phi(\lambda_k, x)}{\min_{i \in L(x_k)} g_i(x_k)} \tag{4.2}$$

and

$$\lambda_k - \lambda_{k+1} \geq \frac{-\psi(\lambda_k, x_k)}{\min_{i \in L(x_k)} g_i(x_k)}, \tag{4.3}$$

respectively. Therefore, for faster convergence, one may consider to increase these differences as much as possible. In this case it seems reasonable to solve, in the constrained case, the optimization problem,

$$\max_{x \in X} \frac{-\phi(\lambda_k, x)}{\min_{i \in L(x)} g_i(x)} \tag{4.4}$$

and in the unconstrained case, the following problem should be solved

$$\max_{x \in X} \frac{-\psi(\lambda_k, x)}{\min_{i \in L(x)} g_i(x)}. \tag{4.5}$$

Although this may be an effective way of speeding up the algorithms, it may not be easy to solve problems (4.4) and (4.5), unless there exists a special structure.

5 Numerical Results

We have programmed Algorithm 3.1 and Algorithm 3.2 with MATLAB. The entropy-type function given in Example 3.1 and the recursive function given in Example 3.2 are used for approximating the $\max$ function. In order to implement the recursion, we have used the following two dimensional approximation function

$$\Pi_2(y) = \frac{\sqrt{(y_1 - y_2)^2 + 1} + y_1 + y_2}{2}.$$

which leads to the following approximation function

$$\varepsilon\Pi_2(\frac{y}{\varepsilon}) = \frac{\sqrt{(y_1 - y_2)^2 + \varepsilon^2} + y_1 + y_2}{2}. \tag{5.1}$$

The right hand side of the equation (5.1) is also known as Chen-Harker-Kanzow-Smale function and it has been used frequently as a stable alternative for approximating the two dimensional $\max$ function [Chen, C. and O.L. Mangasarian, 1995].

Notice that an overflow easily occurs when the exponential function in Example 3.1 is computed with a very large argument. A well-known technique to handle this potential problem is introducing a constant $z \geq \max\{y_1, \cdots, y_s\}$ and then computing

$$\varepsilon\Pi_s(\frac{y}{\varepsilon}) = \varepsilon \log\left(\sum_{i=1}^{s} e^{\frac{y_i - z}{\varepsilon}}\right) + z. \tag{5.2}$$

In our experiments we have also used this technique.

In Step 3 of both algorithms, we have formulated the membership problems as regular nonlinear optimization problems. To solve these nonlinear programs, we have used the standard constrained and unconstrained MATLAB solvers. The MATLAB functions corresponding to these solvers are called fmincon and fminunc, respectively.

To test the approximation approach and verify our observations on performance improvement, we have compiled different test problems from the literature. In the subsequent tables, the columns (or rows) entitled *Entropy* and *Recursive* denote the results with the entropy-type function and the recursive function, respectively. In all test problems the parameter ε is set to $1.0\text{e}{-5}$ and the absolute value function $u \rightarrow |u|$ is replaced with $u \rightarrow \max\{u, -u\}$.

First, we test both approximation functions when they are used in Algorithm 3.1. To do this, a set of problems are randomly generated as suggested by Barros *et. al.* [Barros, A.I., Frenk, J.B.G., Schaible, S. and S. Zhang, 1996b]. The ratios in these problems consist of quadratic functions in the numerators, $f_i(x) := \frac{1}{2}x^T H_i x + a_i^T x + b_i$ and linear functions in the denominators, $g_i(x) := c_i^T x + d_i$. The parameters of these functions are generated as follows:

- ◇ The Hessians H_i are given by $H_i := L_i U_i L_i^T$ where L_i are unit lower triangular matrices with components uniformly distributed within $[-2.5, 2.5]$ and U_i are positive diagonal matrices with the components uniformly distributed within $[0.1, 1.6]$. In order to generate a positive definite Hessian, the first element of the matrix is set to zero.
- ◇ The components of the vectors a_i and c_i are uniformly distributed within $[-15, 45]$ and $[0, 10]$, respectively.
- ◇ Similarly, the increments b_i and d_i are uniformly distributed within $[-30, 0]$ and $[1, 5]$, respectively.

Moreover, the following feasible set is used for all the test problems:

$$X = \{x \in \mathbb{R}^n \mid \sum_{j=1}^{n} x_j \leq 1, \ \ 0 \leq x_j \leq 1, j = 1, \cdots, n\}.$$

Finally, the components of initial feasible point, x_0 are uniformly distributed within $[0, \frac{1}{n}]$.

Table 1 illustrates the performance of Algorithm 3.1 with both approximation functions. The third and fourth columns give the number of iterations required to solve the problems with entropy-type function and recursive function, respectively. We have not reported the solutions λ_k because with both functions all the problems are solved successfully to optimality. We remark that the problems are randomly generated and therefore, a fair comparison between our results and the results reported in [Barros, A.I., Frenk, J.B.G., Schaible, S. and S. Zhang, 1996b] is not easy. In general, we can say that our method requires less number of iterations than theirs.

When we compare the approximation functions, there is no clear evidence suggesting that one function is better than the other one. In some problems using entropy-type function leads to less number of iterations than the recursive function. However, in other problems using recursive function decreases the number of iterations.

Table 1: The comparison of the two approximation functions

		Entropy	Recursive
n	m	No of Iter.	No of Iter.
5	5	2	2
10	5	9	9
15	5	5	5
20	5	6	5
5	10	4	4
10	10	4	4
15	10	5	6
20	10	4	4
5	15	8	8
10	15	9	9
15	15	8	9
20	15	7	6
5	20	5	3
10	20	11	16
15	20	14	13
20	20	13	9

Next we want to compare Algorithm 3.1 (constrained case) and Algorithm 3.2 (unconstrained case). For this comparison, we have used the following problems given in [Gugat, M, 1996b].

Problem 5.1

$$\min_{x\in X} \max\left\{\frac{4x_1^3+11x_2}{16x_1+4x_2}, \frac{4x_1^2-x_1}{3x_1+x_2}, 0\right\} \tag{5.3}$$

where

$$X = \{x \in \mathbb{R}^2 : x_1+x_2 \geq 1,\ 2x_1+x_2 \leq 4,\ x_1, x_2 \geq 0\} \tag{5.4}$$

and the initial feasible point is selected to be $x_0 = (1,1)^T$.

Problem 5.2

$$\min_{x\in X} \max\left\{\left|\frac{3x_1-2x_2}{4x_1+x_2}\right|, \left|\frac{x_1}{3x_1+x_2}\right|\right\} \tag{5.5}$$

where

$$X = \{x \in \mathbb{R}^2 : x_1+x_2 \geq 1,\ 2x_1+x_2 \leq 4,\ x_1, x_2 \geq 0\}, \tag{5.6}$$

and the initial feasible point is selected to be $x_0 = (1,1)^T$.

Problem 5.3

$$\min_{x\in X}\max_{i=0,\cdots,8}\left|\frac{8^3x_1+i^3x_2-i^3x_3-8^3ix_4}{8^4x_4+8i^3x_3}\right| \tag{5.7}$$

where

$$X=\{x\in\mathbb{R}^4: -1000\le x_1,x_2\le 1000,\ 1\le\frac{i^3x_3+8^3x_4}{8^3}\le 1000,\ i=0,\cdots,8\}, \tag{5.8}$$

and the initial feasible point is selected to be $x_0=(0.5,0,0,1)^T$.

Table 2 shows the results with Algorithm 3.1 (constrained case). The first column denotes the example number. The third column gives the number of iterations required to find the final value of λ_k reported in column four. When we analyze the figures in the table, we see that all the problems are successfully solved with Algorithm 3.1. In Problem 5.1, we have obtained a better value than the value reported in [Gugat, M, 1996b] at the expense of more iterations.

Table 2: The results for problems 5.1, 5.2 and 5.3 with Algorithm 3.1

		No of Iter.	λ_k
Pr. 1	Entropy	24	0.4325
	Recursive	24	0.4325
Pr. 2	Entropy	6	0.1961
	Recursive	7	0.1961
Pr. 3	Entropy	33	0.0742
	Recursive	32	0.0742

Table 3 gives the results with Algorithm 3.2 (unconstrained case). The layout of the table is same as in Table 2. The figures in the last column show that the precision of the results with Algorithm 3.2 is worse than the results with Algorithm 3.1. Particularly for Problem 5.3, both entropy-type function and the recursive function do not converge to the optimal value. As the number of constraints increase, the resulting auxiliary problems in the unconstrained case become more difficult. Nonetheless, it is interesting to note that in Problem 5.1, the number of iterations with the entropy-type functions has increased.

Table 3: The results for problems 5.1, 5.2 and 5.3 with Algorithm 3.2

		No of Iter.	λ_k
Pr. 5.1	Entropy	62	0.4325
	Recursive	5	0.4665
Pr. 5.2	Entropy	6	0.1962
	Recursive	7	0.1962
Pr. 5.3	Entropy	37	0.3283
	Recursive	9	0.1930

The numerical results up to this point have shown that both entropy-type function and recursive function perform very well, however the results found with Algorithm 3.2 are inferior to the results found with Algorithm 3.1. Therefore, we will use only Algorithm 3.1 with both approximation functions in the remaining experiments.

Before we proceed to verify some of our observations for performance improvement, we want to test the approximation functions on a large size problem. In general, the problems in numerical analysis include many fractions. Therefore, we select the following problem given in [Gugat, M, 1998].

Problem 5.4 *Let*

$$B = \{(i/100, j/100) : i, j \in \{0, \cdots, 100\}\}.$$

For $x \in \mathbb{R}^6$, $t \in B$, define

$$V(x,t) = x_1 + x_2t_1 + x_3t_2 + x_4t_1t_2,$$
$$W(x,t) = x_6 + x_5t_1.$$

Define

$$X = \{x \in \mathbb{R}^6 : 1 \le W(x,t) \le 10^5, \text{ for all } t \in B\}$$

and

$$F : B \to \mathbb{R}, \qquad F(t) = t_2 \exp(-100t_1).$$

Consider the problem

$$\min_{x \in X} \max_{t \in B} \left| \frac{F(t) - V(x,t)}{W(x,t)} \right|.$$

Let the initial feasible point be $x_0 = (1,0,0,0,0,1)^T$.

The fi gures in Table 4 show that the problem is solved within a few iterations by using both approximation functions. Moreover, the number of iterations and the optimal value are much better than the results reported by Gugat [Gugat, M, 1998].

Table 4: The results for Problem 5.4

	No of Iter.	λ_k
Entropy	6	0.0018
Recursive	5	0.0018

Recall that the results reported in Table 2 have shown that all problems can be solved by using both approximation functions. However, solving the problems with approximation functions require more iterations than the methods suggested in the literature [Gugat, M, 1996b]. Therefore, we have decided to use the same set of problems in order to validate our observations for performance improvement.

Table 5 gives the results with Algorithm 3.1 after we implement several performance improvement observations. The first and second columns give the problem numbers and approximation functions, respectively. The acronym "Reg." in the third column stands for *regular* and the solutions in the corresponding row are found by Algorithm 3.1 without implementing any performance improvement observations. These solutions are same as in Table 2. The acronym "Weak." denotes that in the corresponding row the figures are found by implementing the observation about a weaker stopping condition as discussed in Section 4. In this case, the parameter δ is set to $1.0e - 2$. Similarly, the results in the row "Norm." are found by implementing the normalization observation Algorithm 4.1. The fourth columns shows the number of iterations required to find λ_k given in the last column. The fifth column shows the value of ϕ_ε after the algorithm terminates.

When we analyze the figures in Table 5, we see that using a weaker stopping condition decreases the number of iterations. Moreover, the accuracy of the results are slightly affected by this modification. On the other hand, implementing the observation about normalization decreases the number of iterations drastically and, as expected, without sacrificing too much of the quality of the solutions.

Table 5: The results for problems 5.1, 5.2 and 5.3 with Algorithm 3.1 after performance improvements

			No of Iter.	$\phi_\varepsilon(\lambda_k, x_k)$	λ_k
Pr. 5.1	Entropy	Reg.	24	0.0000	0.4325
		Weak.	11	-0.0006	0.4341
		Norm.	3	0.0000	0.4325
	Recursive	Reg.	24	0.0000	0.4325
		Weak.	12	-0.0007	0.4345
		Norm.	3	0.0000	0.4325
Pr. 5.2	Entropy	Reg	6	0.0000	0.1962
		Weak.	3	-0.0005	0.1978
		Norm.	3	0.0000	0.1962
	Recursive	Reg.	7	0.0000	0.1962
		Weak.	3	-0.0004	0.1973
		Norm.	3	0.0000	0.1962
Pr. 5.3	Entropy	Reg.	33	0.0000	0.0742
		Weak.	18	-0.0008	0.0742
		Norm.	8	0.0000	0.0742
	Recursive	Reg.	32	0.0000	0.0742
		Weak.	14	-0.0007	0.0742
		Norm.	7	0.0000	0.0742

6 Conclusion

We have shown that GFP problems can be solved efficiently by applying a smoothing approximation approach within typical Dinkelbach-type methods. Moreover, we have discussed several observations that can be used for improving the performance of the algorithms. The approximation approach yields near-optimal solutions, but fortunately the closeness of the solution to the actual optimal solution can be controlled by a predetermined parameter. In this respect, through decision makers' interaction the analyst can cope with the tradeoff between efficiency and precision. Our numerical results have also supported that the proposed approach is indeed effective for solving GFP problems.

Though, we strive for solving various problems from the literature, there exist many real-life problems that have to be tackled with efficient methods. Moreover, we did not test some of our performance improvement observations, neither did we consider possible combinations of these observations. These issues remain in our agenda as a possible further research direction.

References

Addou, A. and A. Roubi. Linearization method for generalized fractional programming. To appear in Journal of Global Optimization.

Barrodale, I., Powell, M.J.D. and F.D.K. Roberts. The differential correction algorithm for rational l_∞-approximation. *SIAM Journal on Numerical Analysis*, 21:493–504, 1972.

Barros, A. *Discrete and Fractional Programming Techniques for Location Models*, volume 3 of *Combinatorial Optimization*. Kluwer Academic Publishers, Dordrecht, 1998.

Barros, A.I. and J.B.G. Frenk. Generalized fractional programming and cutting plane algorithms. *Journal of Optimization Theory and Applications*, 87(1):103–120, 1995.

Barros, A.I., Frenk, J.B.G., Schaible, S. and S. Zhang. A new algorithm for generalized fractional programs. *Mathematical Programming*, 72:147–175, 1996a.

Barros, A.I., Frenk, J.B.G., Schaible, S. and S. Zhang. Using duality to solve generalized fractional programming problems. *Journal of Global Optimization*, 8:139–170, 1996b.

Ben-Tal, A. and M. Teboulle. A smoothing technique for nondifferentiable optimization problems. In Dolecki, editor, *Optimization*, Lectures notes in Mathematics 1405, pages 1–11, New York, 1989. Springer Verlag.

Bertsekas, D.P. *Nonlinear Programming*. Athena Scientifi c, Belmont, Massachusets, 1995.

Birbil, Ş.İ., Fang, S.-C., Frenk, J.B.G. and S. Zhang. Recursive approximation of the high dimensional max function. *Erasmus Research Institute of Management Report Series, ERS-2003-003-LIS, Erasmus University Rotterdam, The Netherlands*, 2002.

Chen, C. and O.L. Mangasarian. Smoothing methods for convex inequalities and linear complementarity problems. *Mathematical Programming*, 71:51–69, 1995.

Cheney, E.W. and H.L. Loeb. Two new algorithms for rational approximation. *Numerische Mathematik*, 3:72–75, 1961.

Crouzèix, J.P. and J.A. Ferland. Algorithms for generalized fractional programming. *Mathematical Programming*, 52:191–207, 1991.

Crouzèix, J.P., Ferland, J.A. and S. Schaible. An algorithm for generalized fractional programs. *Journal of Optimization Theory and Applications*, 47(1):35–49, 1985.

Crouzèix, J.P., Ferland, J.A. and S. Schaible. A note on an algorithm for generalized fractional programs. *Journal of Optimization Theory and Applications*, 50:183–187, 1986.

Dinkelbach, W. On nonlinear fractional programming. *Management Science*, 13(7): 492–498, 1967.

Fang, S.-C., Rajasekera, J. R. and H.-S. J. Tsao. *Entropy Optimization and Mathematical Programming*. Kluwer Academic Publishers, Boston/London/Dordrecht, 197.

Gugat, M. An algorithm for Chebyshev approximation by rationals with constrained denominators. *Constructive Approximation*, 12:197–221, 1996a.

Gugat, M. A fast algorithm for a class of generalized fractional programming problems. *Management Science*, 42:1493–1499, 1996b.

Gugat, M. *Fractional Semi-Infinite Programming*. PhD thesis, Universität Trier, Aachen, Germany, 1997.

Gugat, M. Prox regularization methods for generalized fractional programming. *Journal of Optimization Theory and Applications*, 99:991–722, 1998.

Horst, R, Pardalos, P.M and N.V. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1995.

Ignizio, J.P. *Goal programming and extensions*. Lexington, Massachusets, 1976.

Jagannathan, R. An algorithm for a class of nonconvex programming problems with nonlinear fractional objectives. *Management Science*, 31:847–851, 1985.

Kornbluth, J.S. and R.E. Steuer. Multiple objective linear fractional programming. *Management Science*, 27:1024–1039, 1981.

Roubi, A. Method of centers for generalized fractional programming. *Journal of Optimization Theory and Applications*, 107:123–143, 2000.

von Neumann, J. A model of general economic eqilibrium. *Review Economic Studies*, 13:1–9, 1945.

[*] A complete overview of the ERIM Report Series Research in Management:
https://ep.eur.nl/handle/1765/1

ERIM Research Programs:
LIS Business Processes, Logistics and Information Systems
ORG Organizing for Performance
MKT Marketing
F&A Finance and Accounting
STR Strategy and Entrepreneurship