On the Minimization Problem of the Sum of Ratios

Davaajargal Jargalsaikhan[1*], Bayanjargal Darkhijav[2]

[1]*Institute of Mathematics and Digital Technology, Mongolian Academy of Sciences, Ulaanbaatar 13330, Mongolia*
[2]*Department of Applied Mathematics, National University of Mongolia, Ulaanbaatar 14200, Mongolia*

**Corresponding author: davaajargal_j@mas.ac.mn; ORCID:0000-0001-7585-2943*

Article Info: Received: 2022.09.15; Accepted: 2022.10.01; Published: 2022.12.26

Abstract: We consider the problem of minimizing a sums of ratios that belong to a class of global optimization problems. Since the problem is nonconvex, the application of local search algorithms can not always guarantee to find a global solution. It has been shown that problem can be solved by DC programming methods and algorithms. Dinkelbach-type algorithms are more efficient techniques because fractional problems reduce to a scalarized optimization problem. For solving the problem, we apply a generalized Dinkelbach algorithm requires finding the roots of a nonlinear equation. The numerical experiments were conducted on Python Jupyter Notebook for a box constrained set. The problem also has been solved by a gradient descent method and compared with the Dinkelbach algorithm. Numerical results are provided.

Key words: Dinkelbach-type algorithm, gradient descent method

1. Introduction

The problem of optimizing one or several ratios of functions is called a fractional program [3]. In this paper, we consider the following fractional programming problem which consists of the sum of ratio convex functions:

$$\min_{x\in D}\sum_{i=1}^{N}\frac{f_i(x)}{g_i(x)}. \tag{1.1}$$

Where $D \subset R^n$ and $f_i(x),\ g_i(x),\ i = 1, 2, ..., N$ are convex on D.
The sum-of-ratios problem, which is to minimize a sum of several fractional functions subjected to convex constraints, is difficult to solve by traditional optimization methods. Fractional programs with only a single ratio or a maximum of finitely many ratios are fairly well understood. Under suitable conditions, these problems still satisfy some form of generalized convexity, which can be exploited in algorithms for the numerical solution of such problems [4]. On the other hand, fractional programs with sums of ratios are much more difficult and not as well understood Algorithms for classes of sum-of-ratios problems are described in [5-10], and in the review article [11]. Using the Dinkelbach algorithm [2] with vector parameter,the problem is linearized, become the following DC-type form:

$$F(x, \lambda_1, \lambda_2, \ldots, \lambda_N) = \sum_{i=1}^{N} f_i(x) - \sum_{i=1}^{N}\lambda_i g_i(x). \tag{1.2}$$

Where $\lambda_i \subset R^n,\ i = 1, 2, \ldots, N.$

Solution of the problem (1.2) is determined by the root of the equation $F(\lambda_1, \lambda_2, \ldots, \lambda_N) = 0$. We also included Gradient descent (GD) method in our study, it is an iterative first-order optimization algorithm used to find a local minimum of a given objective function. To find a local minimum, the function "steps" in the direction of "the negative" of the gradient. Gradient descent method is widely used in field of deep learning especially neural networks such as DNN (deep neural network) [12], CNN (Convolutional Neural Networks) [13] and others. The goal of this method is find optimal solution when minimizing the differentiable objective function:

$$\theta^* = \arg\min_{\theta \in D} F(\theta) \tag{1.3}$$

and the standard approach is the following sequences

$$\theta_{t+1} = \theta_t - \alpha_t \nabla F(\theta) \tag{1.4}$$

where t is number of iterations.

The estimation of these two methods are performed by Python Jupyter Notebook and results are compared.

2. Methodology

2.1. SLSQP-Sequential Least-Squares Programming

SLSQP is a sequential quadratic programming (SQL) optimization algorithm proposed by Dieter Kraft in 1988 [14]. This is an iterative method for nonlinear optimization problems where objective function and constraints are twice continuously differentiable. Structure algorithm of SQL is the following nonlinear programming problems with minimizing a scalar function:

$$\min p(x) \tag{2.1}$$

subject to general equality and inequality constraints:

$$q_i(x) \geq 0, \ i = 1, 2, \ldots, k. \tag{2.2}$$

$$q_j(x) = 0, \ j = k+1, \ldots, m. \tag{2.3}$$

and to lower and upper bounds on the variables:

$$l_i \leqslant x \leqslant u_i, \ i = 1, 2, \ldots, n. \tag{2.4}$$

This problem (2.1)-(2.4) is can be solved by Lagrangian method

$$L(x, \mu_1, \mu_2) = p(x) - \mu_1 q_i(x) - \mu_2 q_j(x), \ i = 1, 2, \ldots, k., \ j = k+1, \ldots, m. \tag{2.5}$$

where μ_1 and μ_2 are Lagrangian multipliers [15].

SQP methods solve a sequence of optimization subproblems, each of which optimizes a quadratic model of the objective subject to a linearization of the constraints. If the problem is unconstrained, then the method reduces to Newton's method for finding a point where the gradient of the objective vanishes. If the problem has only equality constraints, then the method is equivalent to applying Newton's method to the first-order optimality conditions, or Karush–Kuhn–Tucker conditions, of the problem. At an iterate x_k, a basic sequential quadratic programming algorithm defines an appropriate search direction d_k as a solution to the quadratic programming subproblem:

$$\min_d p(x_k) + \nabla p(x_k)^T d + \frac{1}{2} d^T W_k(x, \mu) d. \tag{2.6}$$

subject to general equality and inequality constraints:

$$q_i(x_k) + \nabla q_i(x_k)^T d \geq 0,\ \ i = 1, 2, \ldots, k. \tag{2.7}$$

$$q_j(x_k) + \nabla q_j(x_k)^T d = 0,\ \ j = k+1, \ldots, m. \tag{2.8}$$

Where $W_k(x, \mu)$ denotes the Hessian of the Lagrangian:

$$W(x, \mu) = \nabla^2_{xx} L(x, \mu). \tag{2.9}$$

Denote by A is the Jacobian matrix, that is

$$A(x)^T = (\nabla q_1(x), \nabla q_2(x), \ldots, \nabla q_m(x)) \tag{2.10}$$

where $q_i(x)$ is i-th component of the vector $q(x)$ [15].

Theorem 2.1. *Suppose that x^* is a solution point of problem(2.1)-(2.4). Assume that the Jacobian A_* of the active constraints at x^* has full rank, that $d^T W_* d \geq 0$ for all $d \neq 0$ such that $A_* d = 0$, and that strict complementary holds. Then if (x_k, μ_k) is sufficiently close to (x^*, μ^*), there is a local solution of the subproblem (2.6)-(2.8),whose active set A_k is the same as the active set A_* of the nonlinear program (2.1)-(2.4) at x^*.*

2.2. Dinkelbach-type algorithm

The algorithms is divided into the following steps:

Step 1: Get initial guess x^0 then $\lambda_i^0 = \frac{f_i(x^0)}{g_i(x^0)},\ \ i = 1, 2, \ldots, N$ and $k = 1$.

Step 2: Solve the following minimization optimization problem

$$\min_{x \in D} F(x, \lambda_1, \lambda_2, \ldots, \lambda_N) = \min_{x \in D} \left(\sum_{i=1}^{N} f_i(x) - \sum_{i=1}^{N} \lambda_i^{k-1} g_i(x) \right). \tag{2.11}$$

Get optimal solution x^k then $\lambda_i^k = \frac{f_i(x^k)}{g_i(x^k)},\ \ i = 1, 2, \ldots, N$.

Step 3: If $F(\lambda_1, \lambda_2, \ldots, \lambda_N) = 0$ the algorithm stops and optimal solution is $\sum_{i=1}^{N} \lambda_i^k$, $i = 1, 2, \ldots, N$. Otherwise $k = k + 1$ and goto Step 2.

Minimization problem (2.11) estimated by SLSQP(Sequential Least-Squares Programming) method in Python Jupyter notebook.

2.3. Gradient descent method algorithm

The algorithm of gradient descent can be outlined as follows:

Step 1: Get initial guess x_0 and precision value ε.

Step 2: $k = k + 1$ and find gradient of a given function: $s_k = -\nabla F(x_{k-1})$,

where $\nabla F(x) = \sum_{i=1}^{m} \frac{\nabla f_i(x) g_i(x) - \nabla g_i(x) f_i(x)}{g_i^2(x)}$.

Step 3: To choose the value α_k, the consider the following problem

$$\alpha_k = \arg\min |F(x_{k-1} + \alpha_k s_{k-1})|. \tag{2.12}$$

then $x_k = x_{k-1} + \alpha_k s_{k-1}$.

Step 4: If $\|x_k - x_{k-1}\| < \varepsilon$ the algorithm stops and optimal solution is $F(x_k)$. Otherwise goto Step 2.

Consider the quadratic function:

$$r(x) = \frac{1}{2} x' Q x \tag{2.13}$$

where Q is positive and symmetric, and method of steepest descent

$$x_{k+1} = x_k - \alpha_k \nabla r(x_k) \tag{2.14}$$

where the stepsize α_k is chosen according to the minimization rule

$$r(x_k - \alpha_k \nabla r(x_k)) = \min_{\alpha \geq 0} r(x_k - \alpha \nabla r(x_k)). \tag{2.15}$$

Theorem 2.2. [16] *The minimization problem (2.14) holds the following estimates for all k,*

$$r(x_{k+1}) \leq \left(\frac{M - m}{M + m} \right)^2 r\left(x_k\right) \tag{2.16}$$

where M and m are the largest and smallest eigenvalues of Q, respectively.

2.4. Convergence of Dinkelbach-type algorithm

Convergence of Dinkelbach-type algorithm in fractional programming is formulated by [1]. Let be an open set $\Omega \in R^n$ given and functions $f_i(x), g_i(x) : \Omega \to R$, $i = 1, 2, \ldots, m$, that are continuous on Ω, and a closed set $S \subset \Omega$, such that

$$f_i(x) > 0, g_i(x) > 0, \ i = 1, 2, \ldots, m, \quad x \in S. \tag{2.17}$$

Consider the following fractional optimization problem

$$F(x) = \min_x \sum_{i=1}^{m} \frac{f_i(x)}{g_i(x)}, \quad x \in S. \tag{2.18}$$

Together with problem (2.18) we also create the parametric optimization problem:

$$G(x, \alpha) = \min_x \sum_{i=1}^{m} f_i(x) - \alpha_i g_i(x)). \ x \in S \tag{2.19}$$

Further, let introduce function $V(\alpha)$ of the optimal value to problem (2.19) as follows

$$V(\alpha) = \inf_x |G(x, \alpha)| = \inf_x \left\{ \sum_{i=1}^{m} |f_i(x) - \alpha_i g_i(x)| : x \in S \right\} \tag{2.20}$$

In addition, suppose that the following assumptions are fulfilled

- $: V(\alpha) > -\infty, \alpha \in K, where\ K\ is\ a\ convex\ compact\ set\ from\ R^m.$ (2.21a)
- $: \alpha \in K \subset R^m$ *there exists a solution* $z = z(\alpha)$ *to problem* (2.19). (2.21b)

Theorem 2.3. [1] *Suppose that in problem (2.18) the assumptions (2.17), (2.21) are satisfied. In addition, let there exists a vector $\alpha_0 = (\alpha_{01}, \alpha_{02}, \ldots, \alpha_{0m})^T \in K \subset R^m$. Besides,suppose that problem (2.18) in α_0 case,the following equality take place:*

$$V(\alpha_0) = \min_x \{ \sum_{i=1}^{m} |f_i(x) - \alpha_{0i} g_i(x)| : x \in S \} = 0 \tag{2.22}$$

Then, any solution $z = z(\alpha)$ to (2.22) is a solution (2.18), so that $z \in Sol(2.22) \subset Sol(2.18)$

The proof of the this theorem is given in paper [1].

3. The results

In the numerical experiment, we solve the following minimization problem (3.1) considered for N = 2.

$$\min_{x\in D}\left\{\frac{< Cx, x >}{< C^2x, x >}+\frac{< C^2x, x >}{< C^3x, x >}\right\} \tag{3.1}$$

Where $D = \{x \in R^n \mid 1 \leqslant x \leqslant 100\}$ is compact and C_{nxn} is Cesaro matrix [17] which is

$$C=\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & \dots & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n} & \frac{1}{n} & \dots & \frac{1}{n} \end{pmatrix}.$$

The Table 1 shows results of the problem (3.1) using Dinkelbach algorithm up to 100 dimension.

Table 1: The result of problem 3.1.

n	Dinkelbach				
	λ_1	λ_2	$min(\lambda_1+\lambda_2)$	k	time(s)
10	1.014	1.003	2.017	3	1.92
20	1.021	1.004	2.025	3	1.73
30	1.026	1.004	2.03	4	1.74
40	1.029	1.004	2.034	4	1.94
50	1.032	1.004	2.037	4	1.74
60	1.035	1.004	2.04	4	1.73
70	1.037	1.004	2.042	4	1.75
80	1.04	1.004	2.046	4	2.04
90	1.042	1.004	2.047	4	2.09
100	1.044	1.004	2.049	4	2.19

The Table 2 shows comparative results of between Dinkelbach algorithm and Gradient descent method in the minimization problem (3.1).

Table 2: Compared results of problem 3.1.

n	Dinkelbach			Gradient descent		
	min	k	time(s)	min	k	time(s)
10	2.017	3	1.92	1.969	3	2.2
20	2.025	3	1.73	2.074	3	1.86
30	2.03	4	1.74	2.143	3	2.51
40	2.034	4	1.94	2.194	3	1.76
50	2.037	4	1.74	2.234	3	1.73
60	2.04	4	1.73	2.268	3	1.92
70	2.042	4	1.75	2.296	3	1.72
80	2.046	4	2.04	2.321	3	1.64
90	2.047	4	2.09	2.342	3	1.6
100	2.049	4	2.19	2.362	3	1.61

4. Conclusions

We consider the sum-of-ratio fractional minimization problem (1.1) over a box constrained set with Cesaro matrix. We solve a problem (3.1) by two methods: Dinkelbach's algorithm and Gradient descent method. Comparisons of these two methods of the problem were made in up to 100 dimensions and the numerical results were conducted on Python Jupyter notebook.In convergence, these two methods are the same, but the solution of Dinkelbach algorithms appears to be more stable than Gradient descent methods solution when increasing the number of Cesaro matrix dimensions. In future, N=3 or more cases will be considered.

References

[1] T. V. Gruzdeva, and A. S. Strekalovsky, "On solving the sum-of-ratios problem," *Applied Mathematics and Computation*, Vol. 318, pp. 260-269, 2018, doi: https://doi.org/10.1016/j.amc.2017.07.074.

[2] W. Dinkelbach, "On nonlinear fractional programming," *Management Science*, Vol. 13, pp. 492–498, 1967, doi: https://doi.org/10.1287/mnsc.13.7.492.

[3] S. Schaible, and J. Shi, "Fractional Programming:The sum-of ratio case," *Optimization Methods and Software*, Vol.18, No. 2, pp. 219–229, 2003, doi: https://doi.org/10.1080/1055678031000105242.

[4] R. W. Freund, and F. Jarre, "Solving the Sum-of-Ratio Problem by an Interior-Point Method," *Journal of Global Optimization*, Vol. 19, pp. 83-102, 2001, doi: https://doi.org/10.1023/A:1008316327038.

[5] A. Cambini, L. Martein, and S. Schaible "On maximizing a sum of ratios," *Journal of Information and Optimization Sciences*, Vol. 10, pp. 65–79, 1989, doi: https://doi.org/10.1080/02522667.1989.10698952.

[6] D. Z. Chen, O. Daescu, Y. Dai, N. Katoh, X. Wu and J. Xu, "Efficient algorithms and implementations for optimizing the sum of linear fractional functions, with applications," *Technical Report, Department of Computer Science, University of Notre Dame, Notre Dame, IN*, 1998.

[7] J. E. Falk, and S. W. Palocsay, "Optimizing the sum of linear fractional functions, in: C.A. Floudas and P.M. Pardalos (eds.)," *Recent Advances in Global Optimization, Princeton University Press, Princeton, NJ*, pp. 221–258, 1992, doi: https://doi.org/10.1515/9781400862528.221.

[8] H. Konno, and T. Kuno, "Generalized linear multiplicative and fractional programming," *Annals of Operations Research*, Vol. 25, pp. 147–161, 1990, doi: https://doi.org/10.1007/BF02283691.

[9] H. Konno, and H. Yamashita, "Minimization of the sum and the product of several linear fractional functions over a polytope," *Manuscript. Department of Industrial Engineering and Management, Tokyo Institute of Technology, Tokyo, Japan*, 1998.

[10] K. Ritter, "A parametric method for solving certain nonconcave maximization problems," *J.Computer and System Sciences*, Vol. 1, pp. 44–54, 1967, doi: https://doi.org/10.1021/ed044p54.

[11] S. Schaible, "Fractional programming with sums of ratios," *Working Paper Series, Graduate School of Management, University of California, Riverside, CA*, no. 96-04, 1996.

[12] A. E. Mouatasim, " Fast gradient descent algorithm for image classification with neural networks," *Signal, Image and Video Processing*, Vol. 14, pp. 1565–1572, 2020, doi: https://doi.org/10.1007/s11760-020-01696-2.

[13] N. Cui, "Applying Gradient Descent in Convolutional Neural Networks," *Journal of Physics: Conference Series*, Vol. 1004: 012027, 2018, doi: https://doi.org/10.1088/1742-6596/1004/1/012027.

[14] D. Kraft, "A software package for sequential quadratic programming," *DFVLR-FB 88-28*, 1988.

[15] J. Nocedal, and S. J. Wright, "Numerical Optimizationg," *Springer*, ISBN: 978-0-387-30303-1, 2006.

[16] D. P. Bertsekas, "Nonlinear Programming," *Athena Scientific Press, Second edition*, 1999.

[17] R. Enkhbat, D. Tsedenbayar, and E. Enkhtsolmon, "Extremal Properties of the Cesaro Operator," *Mongolian Mathematical Journal*, Vol. 23, no. 1, pp. 7-15, 2021.

Бутархай Програмчлалын Нийлбэр Хэлбэрийн Бодлогыг Минимумчлах нь

Жаргалсайханы Даваажаргал[1*], Дархижавын Баянжаргал[2]

[1] *Шинжлэх Ухааны Академи, Математик, Тоон Технологийн Хүрээлэн,Улаанбаатар 13330, Монгол улс*
[2] *Монгол Улсын Их Сургууль, Хэрэглээний шинжлэх ухаан, инженерчлэлийн сургууль, Улаанбаатар, 14200, Монгол улс*

**Холбоо барих зохиогч: davaajargal_j@mas.ac.mn; ORCID:0000-0001-7585-2943*

Өгүүллийн мэдээлэл: Хүлээн авсан: 2022.09.15; Зөвшөөрөгдсөн: 2022.10.01; Нийтлэгдсэн: 2022.12.26

Хураангуй: Энэхүү судалгаандаа бид глобал оптимизацийн ангилалд багтах бутархай программчлалын минимумчлах бодлогыг авч үзсэн болно. Бодлого нь ерөнхий тохиолдолд гүдгэр биш тул локал хайлтын аргаар бодоход үргэлж глобал шийд олдохгүй. Энэ төрлийн бодлогыг DC програмчлалын аргаар шийдэж болохыг харуулсан. Динкельбах алгоритм нь бутархай программчлалын бодлогыг энгийн оптимизацийн бодлогод шилжүүлдэг тул илүү үр дүнтэй арга юм. Иймд энэ бодлогыг бодоход Динкельбах алгоритм ашиглан олон хувьсагчтай шугаман бус тэгшитгэлийн шийдийг олох арга руу шилжүүлэх боломжтой байдаг. Тооцооллыг Python Jupyter Notebook дээр 100 хүртэлх хэмжээсийн хувьд хийсэн ба үүний зэрэгцээ градиент бууралтын арга дээр үр дүнгийн туршилт хийж, Динкельбах алгоритмтай харьцуулсан болно.

Түлхүүр үгс: Динкелбах алгоритм, градиент бууралтын арга
