An Analysis of Dinkelbach's Algorithm for 0-1 Fractional Programming Problems

Tomomi MATSUI * Yasufumi SARUWATARI † Maiko SHIGENO ‡

(METR92-14, December 1992)

* Department of Mathematical Engineering and Information Physics
Faculty of Engineering, University of Tokyo
Bunkyo-ku, Tokyo 113, Japan

†Department of Social Sciences
The National Defense Academy
Hashirimizu, Yokosuka, Kanagawa 239, Japan

‡Department of Management Science
Science University of Tokyo
Kagurazaka, Shinjuku-ku, Tokyo 162, Japan

Abstract: The 0-1 fractional programming problem minimizes the fractional objective function $(c_1x_1 + c_2x_2 + \cdots + c_nx_n)/(d_1x_1 + d_2x_2 + \cdots + d_nx_n) = \boldsymbol{cx}/\boldsymbol{dx}$ under the condition that $\boldsymbol{x} = (x_1, \cdots, x_n) \in \Omega \subseteq \{0,1\}^n$, where Ω is the set of feasible solutions. For a fractional programming problem, Dinkelbach developed an algorithm which obtains an optimal solution of the given problem by solving a sequence of subproblems $\mathrm{Q}(\lambda)$, in which the linear objective function $\boldsymbol{cx} - \lambda\boldsymbol{dx}$ is minimized under the same condition $\boldsymbol{x} \in \Omega$. In this paper, we show that Dinkelbach's algorithm solves at most $\mathrm{O}(\log(nM))$ subproblems in the worst case, where $M = \max\{\max_{i=1,2,\cdots,n} |c_i|, \max_{i=1,2,\cdots,n} |d_i|, 1\}$.

1 0-1 Fractional Programming Problems

The problem of maximizing or minimizing the ratio of two linear functions is called a fractional programming problem and/or a hyperbolic programming problem. Fractional programming problems are found in various fields [22]. A frequent example which occurs in economics is finding the most favorable ratio of revenues and allocations subject to restrictions on the availability of goods. In addition, measures of efficiency for a system are represented as ratios such as profit/time, return/risk and cost/time. There are a lot of articles which dealing with analysis and algorithms for this class of problems [3, 5, 12, 20, 24].

There are several classes of fractional programming problems that have been studied extensively. Those are 0-1 fractional programming problems [1], which include minimal ratio spanning tree problems [4], minimum ratio circuit problems [8, 6, 19], fractional knapsack problems [15], and fractional cutting stock problems [10]. In this paper, we study 0-1 fractional programming problems described below.

Let $\boldsymbol{c} = (c_1, c_2, \cdots, c_n)$ and $\boldsymbol{d} = (d_1, d_2, \cdots, d_n)$ be n-dimensional integer vectors. Then a 0-1 fractional programming problem is formulated as:

$$\begin{aligned} \text{FP:} \quad \text{minimize} \quad & \frac{\sum_{i=1}^{n} c_i x_i}{\sum_{i=1}^{n} d_i x_i} = \frac{\boldsymbol{cx}}{\boldsymbol{dx}}, \\ \text{subject to} \quad & \boldsymbol{x} \in \Omega \subseteq \{0,1\}^n, \end{aligned}$$

where $\boldsymbol{x}$ denotes the column vector $(x_1, x_2, \cdots, x_n)^T$. The subset of $0-1$ valued vectors Ω is called *feasible region,* and when a $0-1$ valued vector $\boldsymbol{x}$ is an element of Ω, we say $\boldsymbol{x}$ is a *feasible solution.* Throughout the paper, we assume that for any feasible solution $\boldsymbol{x}$, the value $\boldsymbol{dx}$ is positive. Here we denote $C = \max\{\max_{i=1,\cdots,n} |c_i|, 1\}$ and $D = \max\{\max_{i=1,\cdots,n} |d_i|, 1\}$. Then, it is clear that the optimal value of the problem is in the interval $[-nC, nC]$.

For fractional programming problems, there are many algorithms which solve a sequence of 0-1 integer programming problems with linear objective function described below.

$$\begin{aligned} \text{Q}(\lambda): \quad \text{minimize} \quad & \boldsymbol{cx} - \lambda \boldsymbol{dx} \\ \text{subject to} \quad & \boldsymbol{x} \in \Omega \subseteq \{0,1\}^n. \end{aligned}$$

Denote by $z(\lambda)$ the optimum objective value of Q(λ). Let $\boldsymbol{x}^*$ be an optimal solution of FP and let $\lambda^* = (\boldsymbol{c}\boldsymbol{x}^*)/(\boldsymbol{d}\boldsymbol{x}^*)$ (i.e. the optimum objective value of FP). Then the followings are well-known:

$$z(\lambda) > 0 \quad \text{if and only if} \quad \lambda < \lambda^*,$$
$$z(\lambda) = 0 \quad \text{if and only if} \quad \lambda = \lambda^*,$$
$$z(\lambda) < 0 \quad \text{if and only if} \quad \lambda > \lambda^*.$$

Furthermore, an optimal solution of Q(λ^*) is also optimal to FP (cf. [7, 16, 17]). Thus solving FP is essentially equivalent to finding $\lambda = \lambda^*$ with $z(\lambda) = 0$. For this purpose, the function $z(\lambda)$ possesses good properties with respect to λ : piece-wise linear, concave, strictly decreasing, $z(-nC) \geq 0$, and $z(nC) \leq 0$. From the above properties, it is clear that when we fix the parameter λ, we can check whether the optimal value λ^* is greater than, equivalent to or less than the current value of λ by the optimal objective value of Q(λ).

There are some methods for generating a sequence of parameters converging to λ^*. One is due to the ordinary binary search method [17, 21, 13]. In the case that the objective values of two different feasible solutions are not equivalent, then its difference is greater than or equal to $1/(nD)^2$. It implies that when we apply the binary search method, the optimal value λ^* can be obtained by solving the subproblem Q(λ) at most $\mathrm{O}(\log(\frac{2nC}{1/(nD)^2})) \leq \mathrm{O}(\log(nCD))$ times.

In 1979, Megiddo [18] proposed an ingenious method to generate a sequence of parameters systematically. He showed that if the subproblem Q(λ) is solvable within $\mathrm{O}(p(n))$ comparisons and $\mathrm{O}(q(n))$ additions, then FP is solvable in $\mathrm{O}(p(n)(q(n)+p(n)))$ time.

Another method is similar to the Newton method in theoretical sense and proposed by Isbell and Marlow [14] and Dinkelbach [7] (and called also Dinkelbach's algorithm). This algorithm was discussed in [17, 21, 11] (and possibly by others). A formal description of the algorithm is given in the next section. Schaible [21] showed that for non-linear fractional programming problems, the convergence rate of the binary search method is only linear, however Dinkelbach's algorithm converges superlinearly. In addition, it is said that Dinkelbach's algorithm is efficient and robust in practice (see [13, 23] for example). However, the worst case time complexity of Dinkelbach's algorithm for 0-1 fractional programming prob-

lems is not shown. In this paper, we show that Dinkelbach's algorithm terminates after at most $O(\log(nCD))$ subproblems $Q(\lambda)$ are solved. Remark that this time complexity is equivalent to that of ordinary binary search method. Our result implies that if there exists a polynomial time algorithm for the subproblem $Q(\lambda)$, Dinkelbach's algorithm solves FP also in polynomial time. In addition, even if the subproblem $Q(\lambda)$ is in the class NP-complete or NP-hard, we can obtain an optimal solution to the original FP by solving the subproblem polynomial times.

2 Description of Dinkelbach's Algorithm

It is essential to observe that the line

$$z = \boldsymbol{c}\boldsymbol{x}' - \lambda \boldsymbol{d}\boldsymbol{x}',$$

is tangent to the function $z(\lambda)$ at $\lambda = \lambda'$, where $\boldsymbol{x}'$ is an optimal solution of $Q(\lambda')$. Therefore, $-\boldsymbol{d}\boldsymbol{x}'$ is a subgradient of $z(\lambda)$ at λ'. Also it is easy to see that the above line crosses the λ-axis at $\lambda = \boldsymbol{c}\boldsymbol{x}'/\boldsymbol{d}\boldsymbol{x}'$.

Now we describe Dinkelbach's algorithm for FP. Dinkelbach's algorithm generates a sequence of parameters converging to λ^* in the manner as shown by thin lines in Figure 1. When the objective value of the subproblem $Q(\lambda)$ becomes 0, the algorithm terminates.

Dinkelbach's Algorithm

Step 1: Set $\lambda = \lambda^1$ such that $\lambda^* \leq \lambda^1 \leq nC$.

Step 2: Solve $Q(\lambda)$ and obtain an optimal solution $\boldsymbol{x}$.

Step 3: If $z(\lambda) = 0$, then output $\boldsymbol{x}$ and stop. Else, set $\lambda = \boldsymbol{c}\boldsymbol{x}/\boldsymbol{d}\boldsymbol{x}$ and go to Step 2.

As initial λ^1, nC may be used, though better choices are often possible by exploiting the structure of given problems.

3 Analysis of Dinkelbach's Algorithm

In this section, we assume that Dinkelbach's algorithm terminates at kth iteration. Then we obtain a sequence of parameters $(\lambda^1, \lambda^2, \cdots, \lambda^k)$ and a sequence of $0-1$ valued vectors

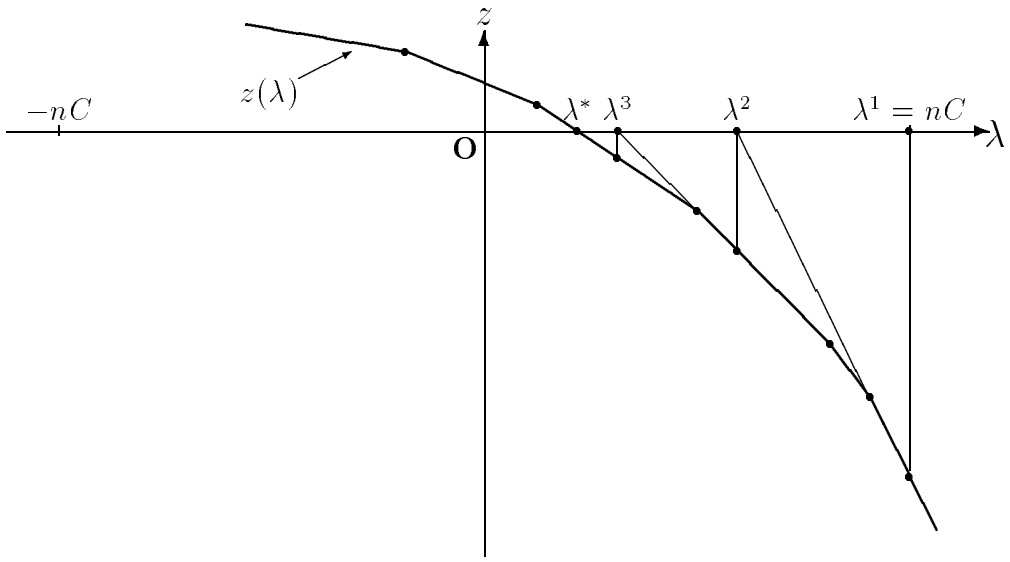

Figure 1: Illustration of the sequence $\{\lambda^r\}$ generated by Dinkelbach's algorithm.

$(\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^k)$. The concavity of $z(\lambda)$ implies the following inequalities:

$$\boldsymbol{dx}^1 > \boldsymbol{dx}^2 > \cdots > \boldsymbol{dx}^{k-1} \geq \boldsymbol{dx}^k > 0,$$
$$\boldsymbol{cx}^1 + nC\boldsymbol{dx}^1 > \boldsymbol{cx}^2 + nC\boldsymbol{dx}^2 > \cdots > \boldsymbol{cx}^{k-1} + nC\boldsymbol{dx}^{k-1} \geq \boldsymbol{cx}^k + nC\boldsymbol{x}^k \geq 0.$$

Since the function $z(\lambda)$ is strictly decreasing, it is also easy to see that

$$z(\lambda^1) < z(\lambda^2) < \cdots < z(\lambda^k) = 0 \text{ and } \lambda^1 > \lambda^2 > \cdots > \lambda^k.$$

Lemma 1 *If* $0 \geq z(\lambda^r) > -1/nD \quad (2 \leq r \leq k)$, *then* $z(\lambda^r) = 0$.

Proof. Since $\lambda^r = \boldsymbol{cx}^{r-1}/\boldsymbol{dx}^{r-1}$,

$$z(\lambda^r) = \boldsymbol{cx}^r - \lambda^r \boldsymbol{dx}^r = \boldsymbol{cx}^r - \frac{\boldsymbol{cx}^{r-1}\boldsymbol{dx}^r}{\boldsymbol{dx}^{r-1}} = \frac{\boldsymbol{cx}^r\boldsymbol{dx}^{r-1} - \boldsymbol{cx}^{r-1}\boldsymbol{dx}^r}{\boldsymbol{dx}^{r-1}}.$$

Suppose that $z(\lambda^r) < 0$, Then $\boldsymbol{cx}^r\boldsymbol{dx}^{r-1} - \boldsymbol{cx}^{r-1}\boldsymbol{dx}^r \leq -1$. Thus the inequalities $0 < \boldsymbol{dx}^{r-1} \leq nD$ imply that $z(\lambda^r) \leq -1/nD$. It is a contradiction.//

The above lemma comes from the integrality of the weight vectors $\boldsymbol{c}$ and $\boldsymbol{d}$. The lemma implies that if $z(\lambda^r) > -1/nD$, then $z(\lambda^r) = 0$ and thus the algorithm terminates at the rth iteration.

Lemma 2 *If* $0 \leq \boldsymbol{cx}^r + nC\boldsymbol{dx}^r < 1$, *then* $z(\boldsymbol{cx}^r/\boldsymbol{dx}^r) = 0$.

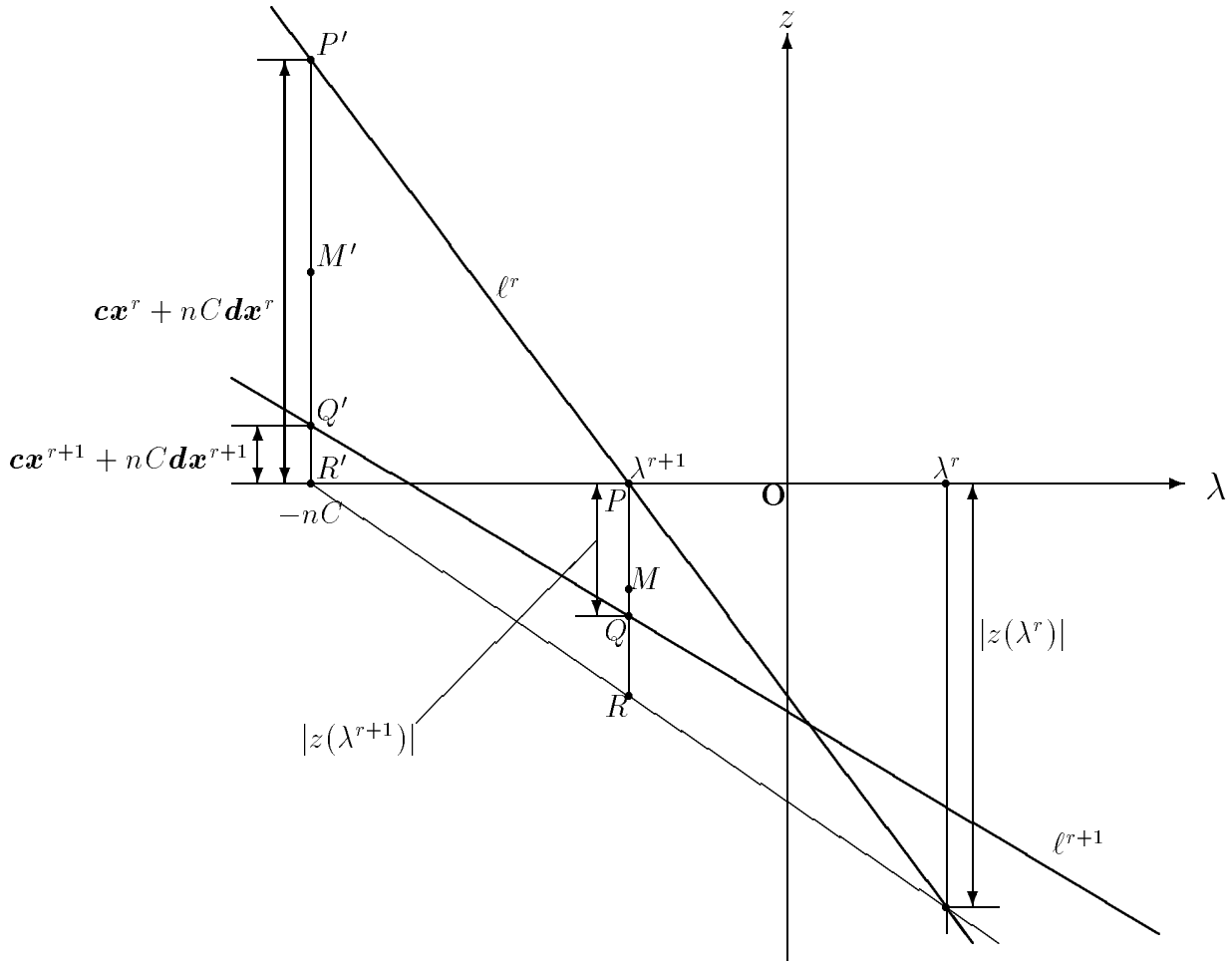

Figure 2: Illustration of Case (i) of the proof of Lemma 3.

Proof. Since the value $\boldsymbol{cx}^r + nC\boldsymbol{dx}^r$ is integer, if $0 \leq \boldsymbol{cx}^r + nC\boldsymbol{dx}^r < 1$, then $\boldsymbol{cx}^r + nC\boldsymbol{dx}^r = 0$ and $\boldsymbol{cx}^r/\boldsymbol{dx}^r = -nC$. From the fact that the optimal value $\lambda^* \geq -nC$, $\boldsymbol{x}^r$ is an optimal solution to FP and $\lambda^* = \boldsymbol{cx}^r/\boldsymbol{dx}^r = -nC$. Then it is clear that $z(\boldsymbol{cx}^r/\boldsymbol{dx}^r) = z(\lambda^*) = 0$. //

The above lemma shows that if the value $\boldsymbol{cx}^r + nC\boldsymbol{dx}^r < 1$, then the algorithm terminates at rth or $r+1$st iteration.

Now we give the main lemma.

Lemma 3 *If* $1 \leq r \leq k-1$, *then either* $|z(\lambda^{r+1})| \leq (1/2)|z(\lambda^r)|$ *or* $\boldsymbol{cx}^{r+1} + nC\boldsymbol{dx}^{r+1} \leq (1/2)(\boldsymbol{cx}^r + nC\boldsymbol{dx}^r)$ *is satisfied.*

Proof. If $\lambda^r + nC \leq 0$, then $\lambda^r = \lambda^* = -nC$ and it implies that $z(\lambda^r) = (1/2)z(\lambda^{r+1}) = 0$. Now assume that $\lambda^r + nC > 0$. Two cases arise.

Case (i) At first we consider the case that $z(\lambda^{r+1})(\lambda^r + nC) \leq \frac{z(\lambda^r)}{2}(\lambda^{r+1} + nC)$. This condition is illustrated in Figure 2. In the figure, the line $z = \boldsymbol{cx}^r - \lambda \boldsymbol{dx}^r$ is denoted by ℓ^r. Here we will use the notations in Figure 2. Let M be the midpoint of the segment $\overline{PR}$. Then the point M has the co-ordinates $(\lambda^{r+1}, \frac{z(\lambda^r)(\lambda^{r+1} + nC)}{2(\lambda^r + nC)})$. Thus the condition $z(\lambda^{r+1})(\lambda^r + nC) \leq \frac{z(\lambda^r)}{2}(\lambda^{r+1} + nC)$ implies that the point $Q = (\lambda^{r+1}, z(\lambda^{r+1}))$ lies on the segment $\overline{MR}$. Under this condition, we show that the inequality $\boldsymbol{cx}^{r+1} + nC\boldsymbol{dx}^{r+1} \leq (1/2)(\boldsymbol{cx}^r + nC\boldsymbol{dx}^r)$ holds. It means that when the line ℓ^{r+1} and the segment $\overline{MR}$ cross each other, the line ℓ^{r+1} also intersects with the segment $\overline{M'R'}$, where M' is the midpoint of the segment $\overline{P'R'}$. Now we prove the inequality.

$$
\begin{aligned}
&(\lambda^r - \lambda^{r+1})(\boldsymbol{cx}^{r+1} + nC\boldsymbol{dx}^{r+1}) \\
&= (\boldsymbol{cx}^{r+1} - \lambda^{r+1}\boldsymbol{dx}^{r+1})(\lambda^r + nC) - (\boldsymbol{cx}^{r+1} - \lambda^r \boldsymbol{dx}^{r+1})(\lambda^{r+1} + nC) \\
&= z(\lambda^{r+1})(\lambda^r + nC) - (\boldsymbol{cx}^{r+1} - \lambda^r \boldsymbol{dx}^{r+1})(\lambda^{r+1} + nC) \\
&\leq \frac{z(\lambda^r)}{2}(\lambda^{r+1} + nC) - (\boldsymbol{cx}^r - \lambda^r \boldsymbol{dx}^r)(\lambda^{r+1} + nC) \\
&= -(1/2)(\boldsymbol{cx}^r - \lambda^r \boldsymbol{dx}^r)(\lambda^{r+1} + nC) = -(1/2)(\frac{\boldsymbol{cx}^r}{\boldsymbol{dx}^r} - \lambda^r)\boldsymbol{dx}^r(\frac{\boldsymbol{cx}^r}{\boldsymbol{dx}^r} + nC) \\
&= -(1/2)(\lambda^{r+1} - \lambda^r)(\boldsymbol{cx}^r + nC\boldsymbol{dx}^r) = (1/2)(\lambda^r - \lambda^{r+1})(\boldsymbol{cx}^r + nC\boldsymbol{dx}^r)
\end{aligned}
$$

Since $\lambda^r > \lambda^{r+1}$, the inequality $\boldsymbol{cx}^{r+1} + nC\boldsymbol{dx}^{r+1} \leq (1/2)(\boldsymbol{cx}^r + nC\boldsymbol{dx}^r)$ has shown.
Case(ii) Next, consider the case that $z(\lambda^{r+1})(\lambda^r + nC) > \frac{z(\lambda^r)}{2}(\lambda^{r+1} + nC)$.

$$
|z(\lambda^{r+1})| = -z(\lambda^{r+1}) < \frac{-z(\lambda^r)}{2}\frac{(nC + \lambda^{r+1})}{(nC + \lambda^r)} = \frac{|z(\lambda^r)|}{2}(1 - \frac{\lambda^r - \lambda^{r+1}}{nC + \lambda^r}) \leq \frac{1}{2}|z(\lambda^r)|.//
$$

Note that both of the values $|z(\lambda)|$ and $\boldsymbol{cx} + nC\boldsymbol{dx}$ are nonincreasing throughout the iterations. At the first iteration, the value $|z(\lambda)|$ is less than or equal to $2n^2CD$. From Lemma 1, it is clear that if there are $\mathrm{O}(\log(\frac{2n^2CD}{1/nD})) \leq \mathrm{O}(\log(nCD))$ iterations, each of which decreases the value $z(\lambda)$ by at least 50%, then the algorithm terminates and an optimal solution is obtained. In the same way, Lemma 2 implies that the number of iterations which decreases the value $\boldsymbol{cx} + nC\boldsymbol{dx}$ by at least 50% is $\mathrm{O}(\log(2n^2CD)) \leq \mathrm{O}(\log(nCD))$ in the worst case. Lemma 3 shows that each iteration decreases either $|z(\lambda)|$ of $\boldsymbol{cx} + nC\boldsymbol{dx}$ by at least 50%. Thus, the number of overall iterations is bounded by $\mathrm{O}(\log(nCD))$.

Theorem 4 *The number of iterations of Dinkelbach's algorithm is* $O(\log(nCD)) \leq O(\log(nM))$ *in the worst case, where* $M = \max\{C, D\}$.

The above theorem shows that the number of iterations of Dinkelbach's algorithm is $\mathrm{O}(\log(nCD))$ in the worst case. It implies that, if there exists a strongly polynomial time algorithm for $\mathrm{Q}(\lambda)$, Dinkelbach's algorithm solves FP in polynomial time. However, when we solve subproblems by a polynomial time algorithm, we need to estimate the input size of the coefficients of the objective function of $\mathrm{Q}(\lambda)$. In the next section, we discuss this argument by considering minimum ratio spanning tree problems and fractional assignment problems.

4 Discussions

Chandrasekaran [4] proposed an algorithm for minimum ratio spanning tree problems, which is based on the binary search method. Dinkelbach's algorithm solves this problem in $\mathrm{O}(T(v, e)\log(vCD))$ time, where v is the number of nodes, e is the number of edges of the given graph and $T(v, e)$ denotes the time complexity of a strongly polynomial time algorithm for ordinary minimum spanning tree problems. It is easy to expand Chandrasekaran's algorithm into general matroid programming problems with fractional objective function. In this case, the number of break points of the function $z(\lambda)$ is at most $n(n-1)/2$ (see [4]). Thus, when the feasible region Ω is the set of characteristic vectors of bases of a matroid, Dinkelbach's algorithm terminates after $\mathrm{O}(\min\{n^2, \log(nCD)\})$ iterations.

For the assignment problem, many algorithms have been developed. Perhaps the most famous algorithm is Hungarian method and its worst case time complexity is $\mathrm{O}(v(v\log v + e))$ [9]. By employing Hungarian method, Dinkelbach's algorithm solves the fractional assignment problem in $\mathrm{O}(v(v\log v + e)\log(vCD))$ time. In [19], Orlin and Ahuja proposed an $\mathrm{O}(\sqrt{v}e\log(vW))$ time algorithm for the assignment problem and it is said that their algorithm is competitive with the existing strongly polynomial algorithm (see [2] also). In their algorithm, it is assumed that the edge weights are integer and W denotes the maximum of the absolute value of edge weights. To incorporate their algorithm into Dinkelbach's algorithm, we need to replace the objective function of the subproblem $\mathrm{Q}(\lambda)$ solved at the

rth iteration by

$$(\boldsymbol{d}\boldsymbol{x}^{r-1})\boldsymbol{c}\boldsymbol{x} - (\boldsymbol{c}\boldsymbol{x}^{r-1})\boldsymbol{d}\boldsymbol{x},$$

where $\boldsymbol{x}^{r-1}$ denotes the optimal solution of the subproblem obtained at the $r-1$st iteration. Thus, in each iteration, Dinkelbach's algorithm solves an assignment problem such that the absolute values of edge weights are less than or equal to $2v^2CD$. It implies that the worst case time bound of Dinkelbach's algorithm for fractional assignment problems is

$$\mathrm{O}(\sqrt{v}e(\log(2v^3CD))(\log(eCD))) \leq \mathrm{O}(\sqrt{v}e(\log(vCD))^2).$$

We remark that when there exists a polynomial time algorithm for a 0-1 integer programming problem with linear objective function, we can solve a 0-1 fractional programming problem also in polynomial time by using the objective function described above.

Acknowledgment

The authors would like to thank Prof. Hirabayashi of the Science University of Tokyo for his constant encouragement and discussions.

References

[1] Y. Anzai. On integer fractional programming. *J. Operations Research Soc. of Japan*, 17 pp.49–66, 1974.

[2] D. P. Bertsekas and J. Eckstein. Dual coordinate step methods for linear network flow problems. *Mathematical Programming*, 42 pp.203–243, 1988.

[3] G.R. Bitran. Experiments with linear fractional problems. *Naval Research Logistics Quarterly*, 26 pp.689–693, 1979.

[4] R. Chandrasekaran. Minimum ratio spanning trees. *Networks*, 7 pp.335–342, 1977.

[5] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9 pp.181–186, 1962.

[6] Y. Chengen and J. Dayong. A primal-dual algorithm for the minimum average weighted length circuit problem. *Networks*, 21 pp.705–712, 1991.

[7] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13 pp.492–498, 1967.

[8] B. Fox. Finding minimal cost-time ratio circuits. *Operations Research*, 17 pp.546–551, 1969.

[9] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization problems. *J. ACM*, 34 pp.596–615, 1987.

[10] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problems - part II. *Operations Research*, 11 pp.863–888, 1963.

[11] M. Grunspan and M. E. Thomas. Hyperbolic integer programming. *Naval Research Logistics Quarterly*, 20 pp.341–356, 1973.

[12] S. Hashizume, M. Fukushima, N. Katoh, and T. Ibaraki. Approximation algorithm for combinatorial fractional programming problems. *Mathematical Programming*, 37 pp.255–267, 1987.

[13] T. Ibaraki. Parametric approaches to fractional programs. *Mathematical Programming*, 26 pp.345–362, 1983.

[14] J. R. Isbell and W. H. Marlow. Attrition games. *Naval Research Logistics Quarterly*, 3 pp.71–93, 1956.

[15] H. Ishii, T. Ibaraki, and H. Mine. Fractional knapsack problems. *Mathematical Programming*, 13 pp.255–271, 1976.

[16] R. Jagannathan. On some properties of programming problems in parametric form pertaining to fractional programming. *Management Science*, 12 pp.609–615, 1966.

[17] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, 1976.

[18] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4 pp.414–424, 1979.

[19] J. B. Orlin and R. K. Ahuja. New scaling algorithm for the assignment and minimum mean cycle problems. *Mathematical Programming*, 54 pp.41–56, 1992.

[20] P. Robillard. (0,1) hyperbolic programming problems. *Naval Research Logistics Quarterly*, 18 pp.47–57, 1971.

[21] S. Schaible. Fractional programming II: on Dinkelbach's algorithm. *Management Science*, 22 pp.868–873, 1976.

[22] S. Schaible. Fractional programming: applications and algorithms. *Europ. J. Operational Research*, 7 pp.111–120, 1981.

[23] S. Schaible and T. Ibaraki. Fractional programming. *Europ. J. Operational Research*, 12 pp.325–338, 1983.

[24] M. Sniedovich. Analysis of a class of fractional programming problems. *Mathematical Programming*, 43 pp.329–347, 1989.