Journal of Global Optimization **24:** 385–416, 2002.

Global Optimization of $0-1$ Hyperbolic Programs ⋆

MOHIT TAWARMALANI[1], SHABBIR AHMED[2] and NIKOLAOS V. SAHINIDIS*[3]

[1]*Krannert School of Management, Purdue University, West Lafayette, IN 47907-1310, USA.*

[2]*School of Industrial & Systems Engineering, Georgia Institute of Technology, 765 Ferst Drive, Atlanta, GA 30332, USA.*

[3]*Department of Chemical Engineering, The University of Illinois at Urbana-Champaign, 600 South Mathews Avenue, Urbana, IL 61801, USA. e-mail: nikos@uiuc.edu*

Abstract. We develop eight different mixed-integer convex programming reformulations of $0-1$ hyperbolic programs. We obtain analytical results on the relative tightness of these formulations and propose a branch and bound algorithm for $0-1$ hyperbolic programs. The main feature of the algorithm is that it reformulates the problem at every node of the search tree. We demonstrate that this algorithm has a superior convergence behavior than directly solving the relaxation derived at the root node. The algorithm is used to solve a discrete p-choice facility location problem for locating ten restaurants in the city of Edmonton.

Key words: Fractional programming, Convex extensions, Facility location

1. Introduction

The $0-1$ hyperbolic program consists of optimizing the sum of ratios of linear functions of binary variables subject to linear constraints. The problem can be stated as follows:

$$(H) \qquad \begin{aligned} \max \ & \sum_{i=1}^{m} \frac{a_{i0} + a_i^T x}{b_{i0} + b_i^T x} \\ \text{s.t.} \ & Dx \leqslant c \\ & x \in \{0,1\}^n \end{aligned}$$

where $D \in \mathbb{R}^{k\times n}$, $c \in \mathbb{R}^k$, $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}^n$, $a_{i0} \in \mathbb{R}$ and $b_{i0} \in \mathbb{R}$. It is assumed that $(b_{i0} + b_i^T x) > 0$ for any i and any feasible x.

Applications of $0-1$ hyperbolic programs of the above form can be found in many diverse areas including airline crew scheduling [3], cutting stock problems [13], optimization of queries in information retrieval [19], scheduling of common carriers [37], and computer-aided molecular design [36].

An unconstrained single ratio version of (H) was addressed by Hammer and Rodeanu [18] who showed that any local minimum is also a global minimum and

⋆ The research was supported in part by NSF awards DMII 95-02722 and BES 98-73586 to NVS.

developed a simple algorithm for its solution. This algorithm was improved by Robillard [32]. More recently a $O(n)$ algorithm for this problem was proposed by Hansen et al. [19].

Numerous researchers have addressed the constrained single ratio $0-1$ hyperbolic program. This problem is clearly NP-hard since pure $0-1$ integer programming is a special case with $b_i = \mathbf{0}$ and $b_{i0} = 1$ for $i = 1, \ldots, m$. Solution strategies for these problems include linearization [45], branch and bound [1, 32], cutting plane algorithms [15, 16], enumerative methods [3, 14] and approximation algorithms [22]. For certain problems with specially structured constraint set, strongly polynomial algorithms have been developed by Megiddo [27]. A survey of these and other methods for the constrained single ratio $0-1$ hyperbolic program appear in the book by Stancu-Minasian [41].

In contrast to the single ratio problem, the constrained multiple ratio $0-1$ hyperbolic program (H) has received relatively little attention. One such problem consisting of a single cardinality constraint was addressed by Saipe [37]. The author developed a branch and bound method along with several heuristic strategies for this problem. Li [24] and Wu [46] suggested solving the constrained multiple ratio $0-1$ hyperbolic program by reformulation into mixed-integer linear programs and applying standard branch and bound techniques. Optimization of sums of fractional terms over linear polytopes have been addressed in [9, 31]. Solution approaches for these problems have been reviewed by [38, 39].

In [43], we defined a convex/concave extension of a lower/upper semi-continuous function ϕ as a convex/concave function that agrees with ϕ at a pre-specified set of points. In this paper, we construct the convex and concave extensions of fractional functions of $0-1$ variables. As a result, we propose a number of mixed-integer convex programming reformulations of (H). The reformulation schemes are used to develop a branch and bound algorithm for this problem which is then applied to solve a discrete p-choice facility location problem.

The remainder of the paper is organized as follows. In Section 2, we discuss the problem of extending a linear fractional function of binary variables over the unit hypercube. We show that the convex envelope of an associated bilinear function yields a convex extension of the fractional function when the variables are restricted to be binary. These extensions are used to derive a number of reformulation schemes for (H) in Section 3. The proposed reformulations have superior relaxation bounds compared to those in the existing literature [24, 46]. The importance of the bounds on the fractional terms in the reformulations is discussed in Section 5. In Section 6, we develop a branch and bound algorithm for (H). Standard branch and bound solvers may be used to solve the problem once it is reformulated into a mixed-integer program. However, such a blind-folded application of branch and bound exhibits poor convergence. We propose a modified form of the branch and bound algorithm that reformulates the original non-convex program at every node of the branch and bound tree thereby expediting convergence. We specialize the algorithm for the cardinality constrained hyperbolic programs in Section 7. Com-

putational experiments in Section 8 explore the relaxation bounds of the various reformulations and compare the solution effort of the proposed branch and bound algorithm to the commercial integer programming solver CPLEX 6.0 [7] for cardinality constrained hyperbolic programs. Finally, the p-choice facility location problem is described as an application of the cardinality constrained hyperbolic programs. This formulation is used to locate restaurant franchises in the city of Edmonton, Canada. The computational experience indicates that the proposed reformulation strategy along with tightening of the reformulations at each node of the branch and bound tree is extremely effective for solving hyperbolic programs.

2. Extensions of Fractional Functions

We first review some definitions and results from [43] that we use to construct the convex extensions of fractional functions.

DEFINITION 2.1. ([43]). Let C be a convex set and $X \subseteq C$. A convex extension of a function $\phi : X \mapsto \mathbb{R}$ over C is any convex function $\eta : C \mapsto \mathbb{R}$ such that $\eta(x) = \phi(x)$ for all $x \in X$.

DEFINITION 2.2. ([43]). Let C be a convex set and $X \subseteq C$. A concave extension of a function $\phi : X \mapsto \mathbb{R}$ over C is any concave function $\chi : C \mapsto \mathbb{R}$ such that $\chi(x) = \phi(x)$ for all $x \in X$.

THEOREM 2.3. ([43]). *Let $L(x_1, \ldots, x_k)$ be a multilinear function where $x_j \in [x_j^L, x_j^U]$ for $j = 1, \ldots, k$. Define $H^k = \prod_{j=1}^k [x_j^L, x_j^U]$ and let E^k be the set of extreme points of H^k. The convex (concave) envelope of $L(x_1, \ldots, x_k)$ forms the convex (concave) extension of $L : E^k \mapsto \mathbb{R}$ over H^k with the smallest epigraph (hypograph).*

Consider the problem

$$\text{(P)} \quad \max\{\phi(x) \mid x \in X\}.$$

It is clear from the above definitions that $\phi(x) : X \mapsto \mathbb{R}$ can be replaced by its concave extension, $\chi(x) : C \mapsto \mathbb{R}$, if such an extension is possible to construct. Then, $\max\{\chi(x) \mid x \in C\}$, is a convex relaxation of (P).

In this section, we identify the convex and concave extensions of rational terms of the form

$$\text{(LF)} \quad g = \frac{a_0 + \sum_{j \in N} a_j x_j}{b_0 + \sum_{j \in N} b_j x_j}$$

where $x \in \{0, 1\}^n$. Note that the objective in (H) consists of a sum of several terms of the form (LF).

We denote an n-dimensional hypercube by H^n and its extreme points by E^n. Also U^n denotes $[0, 1]^n$ and B^n denotes $\{0, 1\}^n$. The convex envelope of a function ϕ will be denoted by convenv(ϕ).

Let us first consider the quadratic form $\phi(x, f) = fb_0 + f\sum_{j\in N} b_j x_j + a_0 + \sum_{j\in N} a_j x_j$ such that $x \in U^n$ and $f \in [f^l, f^u]$.

THEOREM 2.4. *The convex envelope of* $\phi(x, f) = fb_0 + f\sum_{j\in N} b_j x_j + a_0 + \sum_{j\in N} a_j x_j$ *over* $U^n \times [f^l, f^u]$ *is given by:*

$$\text{convenv}\,(\phi(x, f)) = fb_0 + \sum_{j\in N} \text{convenv}(b_j f x_j) + a_0 + \sum_{j\in N} a_j x_j.$$

Proof. Since $fb_0 + a_0 + \sum_{j\in N} a_j x_j$ is linear, it follows that

$$\text{convenv}\,(\phi(x, f)) = fb_0 + \text{convenv}\left(\sum_{j\in N} b_j f x_j\right) + a_0 + \sum_{j\in N} a_j x_j.$$

Let the epigraph of the convex envelope of $\sum_{j\in N} b_j f x_j$ be Φ, and the epigraph of the convex envelope of $b_j f x_j$ be Φ_j, $j \in N$. Let Φ^l and Φ^l_j be the restrictions of Φ and Φ_j, respectively, when $f = f^l$ and Φ^u and Φ^u_j be the restrictions of Φ and Φ_j, respectively, when $f = f^u$. Consider any point $(x^0, f^0) \in U^n \times [f^l, f^u]$ such that $f^0 = \lambda f^l + (1 - \lambda) f^u$. Since ϕ is linear when x is fixed, Φ is the convexification of Φ^l and Φ^u such that

$$\Phi|_{f=f^0} = \lambda\Phi^l + (1 - \lambda)\Phi^u.$$

However, the restriction of ϕ when $f = f^l$ or $f = f^u$ is linear and hence separable

$$\Phi|_{f=f^0} = \lambda\sum_{j\in N} \Phi^l_j + (1 - \lambda)\sum_{j\in N} \Phi^u_j.$$

By the same token,

$$\Phi_j|_{f=f^0} = \lambda\Phi^l_j + (1 - \lambda)\Phi^u_j.$$

Hence,

$$\Phi = \sum_{j\in N} \Phi_j.$$

In other words,

$$\text{convenv}\,(\phi(x, f)) = fb_0 + \sum_{j\in N} \text{convenv}(b_j f x_j) + a_0 + \sum_{j\in N} a_j x_j. \qquad \square$$

The convex and concave envelopes of bilinear terms of the form $b_j f x_j$ are well known [2, 26]. Using these, it follows readily from Theorem 2.4 that the epigraph of the convex envelope of $\phi(x, f)$ is the following polyhedral set:

$$
\begin{array}{lll}
(CV) \quad z_m = f b_0 + \sum_{j \in N} b_j z_j + a_0 + \sum_{j \in N} a_j x_j & & \\
z_j \geqslant f + f^u x_j - f^u & j \in N, \quad \text{if } b_j > 0 \\
z_j \geqslant f^l x_j & j \in N, \quad \text{if } b_j > 0 \\
z_j \leqslant f^u x_j & j \in N, \quad \text{if } b_j < 0 \\
z_j \leqslant f + f^l x_j - f^l & j \in N, \quad \text{if } b_j < 0
\end{array}
$$

and the hypograph of the concave envelope is:

$$
\begin{array}{lll}
(CC) \quad z_m = f b_0 + \sum_{j \in N} b_j z_j + a_0 + \sum_{j \in N} a_j x_j & & \\
z_j \leqslant f^u x_j & j \in N, \quad \text{if } b_j > 0 \\
z_j \leqslant f + f^l x_j - f^l & j \in N, \quad \text{if } b_j > 0 \\
z_j \geqslant f + f^u x_j - f^u & j \in N, \quad \text{if } b_j < 0 \\
z_j \geqslant f^l x_j & j \in N, \quad \text{if } b_j < 0.
\end{array}
$$

Consider now a general quadratic form $q(x, y) = x^t A y$ where $A = [a_{ij}]$ is an $n \times p$ matrix, $x = [x_1 \ldots, x_n] \in H^n$ and $y = [y_1 \ldots, y_p] \in H^p$.

THEOREM 2.5. $\text{convenv}_{H^n \times H^p} q(x, y) = q(x, y)$ *if and only if for every* $a_{ij} \neq 0$ *either* x_i *or* y_j *is at one of its bounds.*

Proof. ($\Rightarrow$) Consider a point (x_0, y_0) such that $x^0 = [x_1^0, \ldots, x_n^0]$ and $y^0 = [y_1^0 \ldots, y_p^0]$. Assume that there exists an $a_{ij} \neq 0$ such that neither x_i^0 nor y_j^0 is at one of its bounds. Then, consider the neighborhood S of (x_0, y_0), $S \subset H^n$ defined as

$$S = \left\{(x, y) \mid x_u = x_u^0 \, \forall u \neq i, \, y_v = y_v^0 \, \forall v \neq j\right\} \cap H^n.$$

On S, $q(x, y) = a_{ij} x_i y_j + c$ for some constant c. Or,

$$q(x, y) = a_{ij} \frac{1}{4}(x_i + y_j)^2 - a_{ij} \frac{1}{4}(x_i - y_j)^2 + c.$$

Depending on the sign of a_{ij}, $q(x, y)$ is strictly concave in either $[1, 1]$ or $[1, -1]$ direction in the subspace spanned by x_i and y_j. Since neither x_i^0 nor y_j^0 is at one of its bounds, both directions are feasible. Therefore, it follows that

$$\text{convenv}_{H^n \times H^p} q(x, y) < q(x, y).$$

($\Leftarrow$) The convex envelope of $q(x, y)$ over $H^n \times H^p$ at (x^0, y^0) is (see [23]),

$$(C) \qquad \text{convenv}_{H^n \times H^p}\, q(x^0, y^0) = \min \sum_{r=1}^{n+p+1} \lambda_r q(x^r, y^r)$$
$$\text{s.t.} \sum_{r=1}^{n+p+1} \lambda_r x^r = x^0$$
$$\sum_{r=1}^{n+p+1} \lambda_r y^r = y^0$$
$$\sum_{r=1}^{n+p+1} \lambda_r = 1$$

where $(x^r, y^r) \in H^n \times H^p$ and $\lambda_r \in (0, 1)$ for all r. Let

$$I = \left\{ i \mid x_i^0 \text{ is at one of its bounds} \right\}$$
$$J = \left\{ j \mid y_j^0 \text{ is at one of its bounds} \right\}.$$

Define F to be the face of $H^n \times H^p$ expressed as

$$F = \left\{ (x, y) \mid x_i = x_i^0 \text{ for } i \in I,\, y_j = y_j^0 \text{ for } j \in J \right\}.$$

If (x^0, y^0) is obtained as a convex combination of points

$$(x^1, y^1), \ldots, (x^{n+p+1}, y^{n+p+1})$$

when the minimum in (C) is attained then, for any $r \in \{1, \ldots, n + p + 1\}$ such that $\lambda_r > 0$, $(x^r, y^r) \in F$. Since for every element $a_{ij} \neq 0$, either x_i^0 or y_j^0 is at its bound, q is linear on F. Hence,

$$q(x^0, y^0) \leqslant \sum_{r=1}^{n+p+1} \lambda_r q(x^r, y^r).$$

Moreover, $\text{convenv}_{H^n \times H^p}\, q(x^0, y^0) \leqslant q(x^0, y^0)$. Hence,

$$\text{convenv}_{H^n \times H^p}\, q(x^0, y^0) = q(x^0, y^0). \qquad \square$$

COROLLARY 2.6. *Consider the quadratic form* $q(x, y) = x^t A y : E^n \times E^p \mapsto \mathbb{R}$. *Then,* $\text{convenv}_{H^n \times H^p}\, q(x, y)$ *is a convex extension of* $q(x, y)$ *over* $H^n \times H^p$.

In fact, it follows from Theorem 2.3 that the $\text{convenv}_{H^n \times H^p}\, q(x, y)$ is the convex extension with the smallest epigraph.

REMARK 2.7. The quadratic form $q(x, y)$ is closed under negation. Hence, Theorem 2.5 implies that the convex and concave envelopes are exact if and only if for every $a_{ij} \neq 0$ either x_i or y_j is at one of its bounds.

Returning to the function $\phi(x, f)$, it is clear from Theorem 2.5 and Remark 2.7 that the convex and concave envelopes of ϕ are exact when $x \in B^n$ and hence form the convex and concave extensions of ϕ restricted to $B^n \times [f^l, f^u]$. From this equivalence, it follows readily that we can replace a term of the form $\phi(x, f)$ appearing in a mathematical program where $x \in B^n$ by the polyhedral sets (CV) and (CC).

Recall now the rational term

$$(LF) \qquad g = \frac{a_0 + \sum_{j \in N} a_j x_j}{b_0 + \sum_{j \in N} b_j x_j}$$

which may be written as

$$(M1) \quad \begin{aligned} f a_0 + f \textstyle\sum_{j \in N} a_j x_j &= g \\ f b_0 + f \textstyle\sum_{j \in N} b_j x_j &= 1 \end{aligned}$$

or as

$$(M2) \quad g b_0 + g \sum_{j \in N} b_j x_j - a_0 - \sum_{j \in N} a_j x_j = 0$$

In either case, the functions generated are of the form of $\phi(x, f)$ and can be replaced by their convex and concave extensions given in (CV) and (CC). This provides us with two different reformulation schemes for hyperbolic programs.

3. Reformulations of the Hyperbolic Program

In the previous section, we have seen that a fractional term of binary variables may be reformulated using linear inequalities in two different ways. Thus, using different combinations of these forms, we can reformulate problem (H) in a number of different ways.

Note that problem (H) consists of a sum of several fractional terms, each in the form of (LF). Using the form (M1) along with the convex and concave envelopes from (CV) and (CC) we get the following reformulation of (H):

$$(R1) \quad \max \sum_{i=1}^{m} f_i a_{i0} + \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} u_{ij}$$

$$\text{s.t.} \quad Dx \leqslant c$$

$$b_{i0} f_i + \sum_{j=1}^{n} b_{ij} u_{ij} = 1 \qquad i = 1, \ldots, m$$

$$u_{ij} \leqslant f_i^u x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \qquad (1)$$

$$u_{ij} \leqslant f_i + f_i^l x_j - f_i^l \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \qquad (2)$$

$$u_{ij} \geqslant f_i + f_i^u x_j - f_i^u \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \qquad (3)$$

$$u_{ij} \geqslant f_i^l x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \qquad (4)$$

$$x \in \{0, 1\}^n$$

where variable f_i is used for the term $1/(b_{i0} + b_i^T x)$ and u_{ij} is used for the term $f_i x_j$, and f_i^l and f_i^u are valid lower and upper bounds for f_i. Note that, along with the integrality requirement, constraints (1) and (2) form the concave extension and constraints (3) and (4) form the convex extension. Li [24] and Wu [46] have suggested special cases of this reformulation where the lower bounds on the fractional terms are assumed to be zero.

REMARK 3.1. If for each $i = 1, 2, \ldots, m$, $b_{i0} + b_i^T x > 0$ on the feasible region of (H), then the constraints

$$f_i \geqslant \frac{1}{b_{i0} + \sum_{j=1}^{n} b_{ij} x_j} \qquad i = 1, \ldots, m \tag{5}$$

are convex. In such a case, the convex extension of $f_i b_{i0} + f_i \sum_{j=1}^{n} b_{ij} x_j$ is implied and may be replaced by the above convex inequalities in *(R1)*.

Following the above remark, a reformulation (R2) is constructed replacing constraints (3) and (4) by the convex nonlinear inequalities (5).

(R1) is a maximization problem, thus, if $a_{i0} + a_i^T x > 0$ for all i, we only need the concave extensions of the fractional terms as far as the optimal solution is concerned. The concave extension of the fractional term is provided by the convex extension of $f_i b_{i0} + f_i \sum_{j=1}^{n} b_{ij} x_j$. Hence, the claim is true:

REMARK 3.2. If $a_{i0} + a_i^T x > 0$ over the feasible region of the mathematical program, then the concave extension of $f_i b_{i0} + f_i \sum_{j=1}^{n} b_{ij} x_j$ can be dropped from *(R1)*.

Next, we employ the form (M2) to derive another reformulation of (H). Since (H) is a maximization problem, it may be reformulated as:

$$\begin{aligned} (H) \qquad \max \ & \sum_{i=1}^{m} g_i \\ \text{s.t.} \ & g_i \leqslant \frac{a_{i0} + \sum_{j=1}^{n} a_{ij} x_j}{b_{i0} + \sum_{j=1}^{n} b_{ij} x_j} \\ & Dx \leqslant c \\ & x \in \{0, 1\}^n \end{aligned} \tag{6}$$

Since $b_{i0} + \sum_{j=1}^{n} b_{ij} x_j > 0$, inequality (6) may be rewritten as

$$g_i b_{i0} + g_i \sum_{j \in N} b_{ij} x_j - a_0 - \sum_{j \in N} a_{ij} x_j \leqslant 0.$$

Hence, replacing $g_i b_{i0} + g_i \sum_{j \in N} b_{ij} x_j$ by the epigraph of its convex extension is adequate for the reformulation and the concave extension is redundant as far as the optimal solution is concerned. We can thus claim:

REMARK 3.3. When *(M2)* is employed to reformulate *(H)*, the concave extension may be dropped in the reformulation.

As a consequence of Remark 3.3, it follows that (H) can be reformulated as:

$$
\begin{aligned}
(R3) \quad \max \quad & \sum_{i=1}^{m} g_i \\
\text{s.t.} \quad & Dx \leqslant c \\
& b_{i0} g_i + \sum_{j=1}^{n} b_{ij} v_{ij} = a_{i0} + \sum_{j=1}^{n} a_{ij} x_{ij} \quad i = 1, \ldots, m \\
& v_{ij} \leqslant g_i^u x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} < 0 \qquad (7) \\
& v_{ij} \leqslant g_i + g_i^l x_j - g_i^l \quad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} < 0 \qquad (8) \\
& v_{ij} \geqslant g_i + g_i^u x_j - g_i^u \quad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} > 0 \qquad (9) \\
& v_{ij} \geqslant g_i^l x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} > 0 \qquad (10) \\
& x \in \{0, 1\}^n
\end{aligned}
$$

where variable g_i is used for the term $(a_{i0} + a_i^T x)/(b_{i0} + b_i^T x)$ and v_{ij} is used for the term $g_i x_j$, and g_i^l and g_i^u are valid lower and upper bounds for g_i. Note that, depending upon the sign of b_{ij}, the constraints (7) and (8), and (9) and (10) form the concave extensions.

The reformulations (R1) and (R3) can also be combined to construct the following:

$$
\begin{aligned}
(R4) \quad \max \quad & \sum_{i=1}^{n} g_i \\
\text{s.t.} \quad & Dx \leqslant c \\
& g_i = f_i a_{i0} + \sum_{j=1}^{n} a_{ij} u_{ij} \quad i = 1, \ldots, m \\
& b_{i0} f_i + \sum_{j=1}^{n} b_{ij} u_{ij} = 1 \quad i = 1, \ldots, m \\
& u_{ij} \leqslant f_i^u x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \\
& u_{ij} \leqslant f_i + f_i^l x_j - f_i^l \quad i = 1, \ldots, m;\ j = 1, \ldots, n \\
& u_{ij} \geqslant f_i + f_i^u x_j - f_i^u \quad i = 1, \ldots, m;\ j = 1, \ldots, n \\
& u_{ij} \geqslant f_i^l x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n
\end{aligned}
$$

$$
\begin{aligned}
& b_{i0}g_i + \sum_{j=1}^{n} b_{ij}v_{ij} = a_{i0} + \sum_{j=1}^{n} a_{ij}x_{ij} \quad i = 1, \ldots, m \\
& v_{ij} \leqslant g_i^u x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} < 0 \\
& v_{ij} \leqslant g_i + g_i^l x_j - g_i^l \qquad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} < 0 \\
& v_{ij} \geqslant g_i + g_i^u x_j - g_i^u \qquad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} > 0 \\
& v_{ij} \geqslant g_i^l x_j \qquad i = 1, \ldots, m;\ j = 1, \ldots, n;\ b_{ij} > 0 \\
& x \in \{0, 1\}^n
\end{aligned}
$$

To obtain tighter relaxations of the reformulated problem, we can introduce additional valid inequalities obtained by multiplying the original constraint set of (H) by the variables corresponding to the fractional terms. Specifically, we can introduce the following constraints to (R1):

$$\sum_{j=1}^{n} d_{rj}u_{ij} - f_i^l \sum_{j=1}^{n} d_{rj}x_j \leqslant c_r f_i - c_r f_i^l \quad r = 1, \ldots, k;\ i = 1, \ldots, m \quad (11)$$

$$f_i^u \sum_{j=1}^{n} d_{rj}x_j - \sum_{j=1}^{n} d_{rj}u_{ij} \leqslant c_r f_i^u - c_r f_i \quad r = 1, \ldots, k;\ i = 1, \ldots, m \quad (12)$$

and the following constraints to (R3):

$$\sum_{j=1}^{n} d_{rj}v_{ij} - g_i^l \sum_{j=1}^{n} d_{rj}x_j \leqslant c_r g_i - c_r g_i^l \quad r = 1, \ldots, k;\ i = 1, \ldots, m \quad (13)$$

$$g_i^u \sum_{j=1}^{n} d_{rj}x_j - \sum_{j=1}^{n} d_{rj}v_{ij} \leqslant c_r g_i^u - c_r g_i \quad r = 1, \ldots, k;\ i = 1, \ldots, m \quad (14)$$

where d_{rj} are the elements of the constraint matrix D and c_r are the elements of the right hand side vector c. Let (R5) be obtained by adding constraints (11) and (12) to (R1) and (R7) be obtained by adding constraints (13) and (14) to (R3). It is obvious that (R5) and (R7) are valid reformulations of (H). In (R5) and (R7), we retain the concave as well as the convex extensions since in the presence of constraints (11), (12), (13) or (14) the concave extensions are no longer redundant when the integrality requirements are dropped.

Another reformulation, (R6), is constructed from (R5) by adding the convex non-linear constraints (5). Our final reformulation (R8) is constructed by combining (R5) and (R7) in a similar manner as (R4) is constructed from (R1) and (R3).

Note that no additional binary variables are introduced in the reformulations. However, a large number of continuous variables are introduced. Table 1 compares the size of the various reformulations (R1)–(R8) in terms of m, n and k. The relative

Table 1. Size of the reformulations

Reformulation	Number of continuous variables	Number of constraints	
		Linear	Nonlinear
(R1)	$m + mn$	$k + m + 4mn$	0
(R2)	$m + mn$	$k + m + 2mn$	m
(R3)	$m + mn$	$k + m + 2mn$	0
(R4)	$2(m + mn)$	$k + 3m + 6mn$	0
(R5)	$m + mn$	$k + 2km + m + 4mn$	0
(R6)	$m + mn$	$k + 2km + m + 2mn$	m
(R7)	$m + mn$	$k + 2km + m + 4mn$	0
(R8)	$2(m + mn)$	$k + 4km + 3m + 8mn$	0

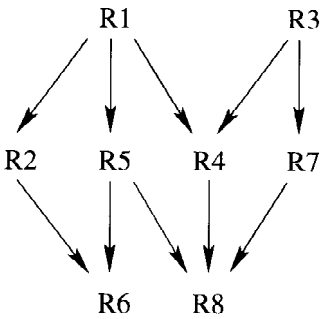

Figure 1. Comparison of reformulation bounds: Ri $\rightarrow$ Rj $\Leftrightarrow$ $V_{\mathrm{LRi}} \geqslant V_{\mathrm{LRj}}$.

importance of these reformulations depends on the strength the bounds offered by their relaxations.

4. Upper Bounding of 0 − 1 Hyperbolic Programs

Relaxing the integrality requirement in the reformulations (R1)–(R8) results in convex upper bounding problems suitable for use within a branch and bound framework. We denote these relaxations by (LR1)–(LR8) respectively. The optimal value of a mathematical program (R) will be denoted by V_{R}. The upper bounds obtained from the reformulations are compared in Figure 1. These relationships are clear from the construction of the reformulations.

As mentioned earlier, the reformulations proposed in [24] and [46] are special cases of (R1). Thus, it is clear from the relationships in Figure 1, that the bounds obtained from the reformulations (R2), (R4), (R5), (R6) and (R8) dominate those obtained from the reformulations in [24] and [46]. As an example of the relative tightness of the bounds from the different reformulations, we consider the

following numerical example from [24]:

$$\begin{aligned} \max \quad & -\frac{1+x_1}{0.5x_1+x_2} - \frac{2-3x_2}{2x_2+x_3+x_4} \\ \text{s.t.} \quad & x_1+x_2+x_3 \geqslant 2 \\ & 2x_1+3x_2-4x_4 \leqslant 2 \\ & 0.5x_1+x_2 > 0 \\ & 2x_2+x_3+x_4 > 0 \\ & x_1,x_2,x_3,x_4 \in \{0,1\} \end{aligned}$$

The reformulation proposed in [24] provides an LP relaxation bound of -0.5. Reformulations (R5) and (R7) provide LP relaxation bounds of -0.6923 and -0.6563, respectively. In all three cases, the reformulations were constructed using bounds on the fractional terms that were determined by solving unconstrained single ratio $0-1$ hyperbolic programs. The optimal solution to the problem is -0.75. Thus, (R5) and (R7) reduce the relaxation gap provided in [24] by 76.9 and 62.4%, respectively. Further computational results on the comparison of the bounds from the various reformulations are presented in Section 8.

Note that the objective function of (H) may be expressed as $\sum_{i=1}^{m} g_i(x)$ where

$$g_i(x) = \frac{a_{i0} + \sum_{j=1}^{n} a_{ij}x_j}{b_{i0} + \sum_{j=1}^{n} b_{ij}x_j}.$$

Let g_i^u $i = 1,\ldots,m$ be the tightest upper bounds available on the following maximization problem:

$$\max\left\{g_i(x) \mid Dx \leqslant c, x \in \{0,1\}^n\right\}.$$

Such bounds may be obtained by solving single ratio hyperbolic programs and will be discussed in the next section. Upper bounds on $g_i(x)$ may also be obtained over the linear relaxation of the constraint set of (H) using the Charnes–Cooper transformation for single ratio fractional programs [6]. Let us denote these bounds by q_i^u, $i = 1,\ldots,m$. We then define $V_{\mathrm{UB}} = \sum_{i=1}^{m} g_i^u$ and $V_{\mathrm{CC}} = \sum_{i=1}^{m} q_i^u$. It is easy to see that V_{UB} and V_{CC} provide valid upper bounds for V_{H}. We shall now compare the reformulation bounds to these simple bounding strategies.

THEOREM 4.1. $V_{\mathrm{H}} \leqslant V_{\mathrm{LR3}} \leqslant V_{\mathrm{UB}}$. *If the extreme points of the feasible region of (H) are binary-valued and* $V_{\mathrm{H}} < V_{\mathrm{UB}}$, *then* $V_{\mathrm{LR3}} < V_{\mathrm{UB}}$.

Proof. Since (LR3) is a valid upper bounding schemes for (H), $V_{\mathrm{H}} \leqslant V_{\mathrm{LR3}}$. Since $g_i \leqslant g_i^u$, it follows that $V_{\mathrm{LR3}} \leqslant V_{\mathrm{UB}}$.

If there is an optimal solution to (LR3) which is integral, then $V_{\mathrm{H}} = V_{\mathrm{LR3}}$ and hence $V_{\mathrm{H}} < V_{\mathrm{UB}}$ implies $V_{\mathrm{LR3}} < V_{\mathrm{UB}}$.

Now assume that there does not exist an integral solution to (LR3). Let x^0 be the optimal solution of (LR3). If $V_{\mathrm{LR3}} = V_{\mathrm{UB}}$, then each g_i is at its upper bound. It follows from Theorem 2.5, that (R3) exactly represents the fractional program when each g_i is at its bound. Let F be the smallest face of U^n such that $x^0 \in F$. Then $x^0 \in \mathrm{ri}(F)$. Since, for all i, $g_i(x)$ is an explicit quasi-convex function which achieves its global maximum in $\mathrm{ri}(F)$, each $g_i(x)$ is constant over F (see Theorem 2.3.7 in [41]). Hence, $g_i(x)$, $i = 1, \ldots, m$ attain their optimal value at every $x \in F$. By assumption, the extreme points of F are binary vectors. Hence, there exists an integral optimum of (LR3) thereby contradicting the assumption. □

The above result implies that, if $V_{\mathrm{LR3}} = V_{\mathrm{UB}}$, then $V_{\mathrm{H}} = V_{\mathrm{LR3}} = V_{\mathrm{UB}}$. Moreover, from Figure 1, it also follows that the bounds from (R4), (R7) and (R8) dominate V_{UB}.

PROPOSITION 4.2. $V_{\mathrm{H}} \leqslant V_{\mathrm{LR5}} \leqslant V_{\mathrm{CC}}$.

Proof. Consider the inequalities

$$\sum_{j=1}^{n} \hat{a}_{ij} u_{ij} - \sum_{j=1}^{n} \hat{a}_{ij} x_j f_i^l \leqslant \hat{b}_i (f_i - f_i^l) \qquad i = 1, \ldots, k \tag{15}$$

$$\sum_{j=1}^{n} \hat{a}_{ij} x_j \leqslant \hat{b}_i \qquad i = 1, \ldots, k \tag{16}$$

Since $f_i^l \geqslant 0$, from 16

$$-\sum_{j=1}^{n} \hat{a}_{ij} x_j f_i^l \geqslant -\hat{b}_i f_i^l \qquad i = 1, \ldots, k \tag{17}$$

Then, by 17 and 15, it follows that

$$\sum_{j=1}^{n} \hat{a}_{ij} u_{ij} \leqslant \hat{b}_i f_i \qquad i = 1, \ldots, k.$$

Therefore, the Charnes-Cooper transformation is embedded in (R5) implying that $V_{\mathrm{H}} \leqslant V_{\mathrm{LR5}} \leqslant V_{\mathrm{CC}}$. □

From Figure 1, it follows that the bounds from (R6) and (R8) dominate V_{CC}. The above results will be used in Section 8.3 to show that the proposed reformulations provide better bounds than those in [37] for the cardinality constrained $0 - 1$ hyperbolic program.

5. Bounds on the Rational Terms

Bounds on the fractional terms have a significant effect on the tightness of the relaxation. We show in [42] that the following result holds.

THEOREM 5.1. *Consider the function* $\phi(x, f) = fb_0 + f\sum_{j\in N} b_j x_j - a_0 + \sum_{j\in N} a_j x_j$ *where* $x \in U^n$. *Let* $\alpha(x, f)$ *be the convex envelope of* $\phi(x, f)$ *when* $f \in [f^l, f^u]$ *and* $\alpha_t(x, f)$ *be the convex envelope of* $\phi(x, f)$ *when* $f \in [f^l, f_t^u]$, *where* $f_t^u < f^u$. *Let* $N^+ = \{j \mid b_j > 0\}$ *and* $N^- = \{j \mid b_j < 0\}$. *Let* $x = [x_1, \ldots, x_n]$. *Then,* $\alpha_t(x_0, f_0) > \alpha(x_0, f_0)$, $(x^0, f^0) \in U^n \times [f^l, f_t^u]$ *if and only if*

$$(x^0, f^0) \in \bigcup_{j\in N^+} \left\{(x, f) \mid f + (f_t^u - f^l)x_j - f_t^u > 0,\ x_j < 1\right\} \cup \bigcup_{j\in N^-} \left\{(x, f) \mid f + (f^l - f_t^u)x_j - f^l > 0,\ x_j > 0\right\}$$

The above theorem characterizes the exact non-empty region over which the convex envelope tightens when tighter bounds are used on the fractional terms. It follows that the algorithmic efficiency of any method solving the hyperbolic program depends not only on the reformulation employed but also on the tightness of the bounds on the fractional terms used in the reformulation.

Tight bounds on the fractional term can be obtained by solving constrained single ratio $0-1$ hyperbolic programs. Although such problems are typically difficult, it is possible to develop strongly polynomial algorithms for some specially structured problems. We will rely on an important result due to Megiddo [27] in this context.

THEOREM 5.2. ([27]). *Consider the following two problems:*

$$(A) \quad \max \sum_{j=1}^{n} c_j x_j \quad \text{s.t. } (x_1, \ldots, x_n) \in D$$

$$(B) \quad \max \frac{\sum_{j=1}^{n} a_j x_j}{\sum_{j=1}^{n} b_j x_j} \quad \text{s.t. } (x_1, \ldots, x_n) \in D$$

(assuming the denominator is always positive).

If problem (A) is solvable within $O(p(n))$ *comparisons and* $O(q(n))$ *additions, then problem (B) is solvable in time* $O(p(n)(q(n) + p(n)))$.

The proof of the above theorem is constructive in the sense that, given an algorithm for problem (A), Megiddo provides a recipe for developing an algorithm for problem (B). The recipe exploits the following well-known result:

LEMMA 5.3. ([41]). *Let x^* be an optimal solution to (B) and let*

$$t^* := \frac{a_0 + \sum_{j=1}^{n} a_j x_j^*}{b_0 + \sum_{j=1}^{n} b_j x_j^*}.$$

Define

$$F(t) = \max_{(x_1,\ldots,x_n)\in D} \left(a_0 + \sum_{j=1}^{n} a_j x_j - t \left(b_0 + \sum_{j=1}^{n} b_j x_j \right) \right).$$

Then

(a) $F(t) > 0$ *if and only if* $t < t^*$,

(b) $F(t) = 0$ *if and only if* $t = t^*$,

(c) $F(t) < 0$ *if and only if* $t > t^*$.

The algorithm for solving (B) developed in the proof of Theorem 5.2 determines t^* by parametrically solving $F(t) = 0$ using the algorithm for (A). A comparison is performed by algorithm (A) to choose one out of two possible computation paths. Since $F(t)$ defines a parametric family of problems of type (A), the comparison must decide if a linear function of the form $\alpha + \beta t$ is less than or greater than zero. It follows from Lemma 5.3 that $t^* > \alpha/\beta$ whenever $F(\alpha/\beta) > 0$ and $t^* < \alpha/\beta$ whenever $F(\alpha/\beta) < 0$. Thus, we disregard all values of t greater than α/β if $F(\alpha/\beta) < 0$ and disregard all values of t less that α/β if $F(\alpha/\beta) > 0$. The choice of the appropriate computation path is now unique and can be easily performed.

As mentioned in [27], the above algorithm can be accelerated when the algorithm (A) only performs comparisons of input elements. In this case, the breakpoints of $F(t)$ are searched using the median finding algorithm to locate the linear segment of $F(t)$ containing the optimal t^*. As an example, we apply this accelerated scheme to the unconstrained single ratio $0-1$ hyperbolic program to develop an algorithm with $\Theta(n)$ complexity. The single ratio unconstrained $0-1$ hyperbolic program is formulated as:

$$(P) \quad \max \frac{a_0 + \sum_{j=1}^{n} a_j x_j}{b_0 + \sum_{j=1}^{n} b_j x_j}$$
$$\text{s.t.} \quad x_j \in \{0, 1\} \quad j = 1, \ldots, n$$

where the denominator is assumed to be positive. The function $F(t)$ in Lemma 5.3 reduces to:

$$F(t) = \max_{x_j \in \{0,1\}} \left(a_0 + \sum_{j=1}^{n} a_j x_j - t \left(b_0 + \sum_{j=1}^{n} b_j x_j \right) \right).$$

Meggido's scheme exploits the solution algorithm for the following problem:

$$(L) \qquad \begin{array}{ll} \max & a_0 + \sum_{j=1}^{n} a_j x_j - t b_0 - t \sum_{j=1}^{n} b_j x_j \\ \text{s.t.} & x_j \in \{0, 1\}. \end{array}$$

(L) can be easily solved by setting x_j equal to 1 if $(a_j - t b_j) > 0$ and 0 otherwise. Thus, a_j/b_j for all j are the critical values for t which need to be tested and form the break-points in Megiddo's scheme. Since $F(t)$ is piecewise linear between any two values of a_j/b_j, we search the n breakpoints to locate t^*. We now state a slightly modified version of Megiddo's algorithm for solving (P) that records certain calculations while solving problems of the type (L) to improve the efficiency of the algorithm:

Step 0. $J \leftarrow \{0, \ldots, n\}$.

Step 1. Let $T = \{a_j/b_j \mid j \in J\}$. If $|J| = 2$, then let x_j^* to 0 for all $j \in J \backslash \{0\}$ such that $a_j \leqslant \min\{T\} b_j$, $x_j^* = 1$ for all other $j \in J \backslash \{0\}$ and terminate.

Step 2. Calculate $\hat{t} = \text{median}\{T\}$.

Step 3. Let $J_g = \{j \in J \mid a_j > \hat{t} b_j\} \cup \{0\}$ and $J_1 = \{j \in J \mid a_j \geqslant \hat{t} b_j,\ j \neq 0\}$. Calculate $F(\hat{t}) = \sum_{j \in J_g} a_j - \hat{t} \sum_{j \in J_g} b_j$.

Step 4. If $F(\hat{t}) = 0$, let $x_j^* = 1$ for all $j \in J_g$, $x_j^* = 0$ for all $j \in J \backslash J_g$ and terminate. If $F(\hat{t}) > 0$, let $x_j^* = 0$ for all $j \in J \backslash J_g$, $J \leftarrow J_g$ and return to Step 1. If $F(\hat{t}) < 0$, let $x_j^* = 1$ for all $j \in J_1$, $J \leftarrow J \backslash J_1$, $a_0 \leftarrow a_0 + \sum_{j \in J_1} a_j$, $b_0 \leftarrow b_0 + \sum_{j \in J_1} b_j$, and return to Step 1.

PROPOSITION 5.4. *Megiddo's algorithm as applied to the unconstrained single ratio* $0-1$ *hyperbolic program with positive denominator has* $\Theta(n)$ *complexity.*

Proof. The effort in each iteration of the above algorithm consists of determining the median of the current set T of critical t values and solving problem $F(t)$. Note that the cardinality of T and J is halved in each iteration. Thus, the total time spent by the algorithm in finding medians [4] and evaluating $F(t)$ is $n + n/2 + n/4 + \cdots$ or $O(n)$. The total effort required in evaluating $F(t)$ is then $O(n)$. Clearly, the lower bound on the total time required is $\Omega(n)$, and so the result follows. □

The above algorithm developed by applying the constructive scheme of Megiddo is equivalent to the $\Theta(n)$ algorithm of Hansen et al. [20] for the unconstrained single ratio $0-1$ hyperbolic program with a positive denominator.

Saipe [37] developed an $O(np)$ algorithm for single ratio $0-1$ hyperbolic programs with a single cardinality constraint, i.e.

$$D := \left\{ x_j \in \{0,1\},\ j = 1,\ldots,n \mid \sum_{j=1}^{n} x_j = p \right\}.$$

Problem (A) with D defined as above can be solved by first determining the order–p statistic, $\hat{c}$, of the coefficients, c_j, by using the linear median finding algorithm of [4] in $O(n)$ comparisons. Let $J := \{j | c_j > \hat{c}\}$, the optimal solution value of problem (A) is then given by $\sum_{j\in J} c_j + (p - |J|)\hat{c}$, which requires $O(p)$ additions. Thus, by Theorem 5.2, Megiddo's scheme would give rise to an algorithm for problem (B) of $O(np + n^2)$. Saipe [37] takes advantage of the result that the variable x_j corresponding to the index $j = \text{argmax}\{a_i/b_i \mid i = 1,\ldots,m\}$ is 1 in an optimal solution to (P).

For problems with general constraint sets, one can derive the bounds in polynomial time by relaxing the integrality requirements and solving the continuous hyperbolic program by using the Charnes–Cooper linear programming reformulation [6].

6. A Branch and Bound Algorithm

In this section, we develop a branch and bound algorithm for the $0-1$ hyperbolic program. A naive approach to solving (H) would be to apply standard branch and bound techniques to its MILP reformulation. As has been pointed out in Section 5, the gap between the reformulation and its continuous relaxation is extremely sensitive to the quality of the bounds on the fractional terms. Also, since in a branch and bound framework, the feasible space of the subproblems at each node is a subset of the feasible space of the problem at the root node, it is possible to derive tighter bounds on the fractional terms over these smaller feasible spaces. Therefore, in the branch and bound algorithm for (H), we propose to reformulate the problem at every node of the branch and bound tree using the best available bounds. Henceforth, we shall refer to this technique as *node tightening*.

We now present a formal statement of the algorithm. The following notation is used:

k	Iteration number
i	Current node of the branch and bound tree
$\mathcal{L}$	List of active nodes
R_i	Reformulated problem at the ith node
$U(R_i)$	Upper bound on the solution of R_i
U^k	Upper bound on the solution of (H) at the kth iteration
L^k	Lower bound on the solution of (H) at the kth iteration

Initialization:

Set $k = 0$, $L^k = -\infty$.

Construct R_0, the reformulation at the root node, and set $U(R_0) = +\infty$.

Include the root node in the list: $\mathcal{L} = \{0\}$.

Main Step (Iteration k)

Step 1. Termination:

Set the upper bound $U^k := \max_{i \in \mathcal{L}}\{U(R_i)\}$.

Set $\mathcal{L} = \mathcal{L} \backslash \{i\}$ for all i with $U(R_i) \leqslant L^k$.

If $\mathcal{L} = \phi$ stop, the current best solution is optimal. Else, set $k = k + 1$, $U^k = U^{k-1}$ and $L^k = L^{k-1}$.

Step 2. Node Selection:

Select node i from $\mathcal{L}$ according to some node selection rule.

Set $\mathcal{L} = \mathcal{L} \backslash \{i\}$.

Step 3. Node Tightening:

Get tight bounds on the fractional terms.

Reconstruct R_i using these bounds.

Step 4. Upper Bounding:

If the relaxation of R_i is desired to be solved, solve it and obtain $U(R_i)$ and go to Step 5. Else, go to Step 6.

Step 5. Lower Bounding:

If $U(R_i) \leqslant L^k$, go to Step 1.

Use a heuristic method to construct a feasible solution and update L^k. Record the best known solution.

Step 6. Branching:

Use a branching strategy to obtain a set of new problems $R_{i_1}, \ldots, R_{i_q}$ from R_i.

Update node list $\mathcal{L} = \mathcal{L} \cup \{i_1, i_2, \ldots, i_q\}$.

Go to Step 1.

As mentioned earlier, the key difference between the above algorithm and standard branch and bound schemes for global optimization is in the node tightening step. The next two sections present specific implementations of the algorithm.

7. Cardinality Constrained Hyperbolic Programs

Cardinality constrained hyperbolic programs are of the form:

$$(CCH) \qquad \max \sum_{i=1}^{m} \frac{a_{i0} + \sum_{j=1}^{n} a_{ij} x_j}{b_{i0} + \sum_{j=1}^{n} b_{ij} x_j}$$

$$\text{s.t.} \sum_{j=1}^{n} x_j = p$$

$$x_j \in \{0, 1\} \qquad j = 1, \ldots, n.$$

We shall demonstrate the efficacy of the proposed branch and bound algorithm for $0 - 1$ hyperbolic programs through a specific implementation for (CCH). The applications of cardinality constrained hyperbolic program include scheduling common carriers [37] and p-choice facility location which we address in Section 8.3.

In [37], Saipe developed a specialized branch and bound algorithm for (CCH) using the upper bound V_{UB} described in Section 4, where the upper bound for each fractional term is obtained employing Saipe's algorithm for single ratio fractionals as discussed in Section 5. We show next that (LR3)-(LR8) provide better bounds than that used in [37].

COROLLARY 7.1. *$V_{\mathrm{CCH}} \leqslant V_{\mathrm{LR3}} \leqslant V_{\mathrm{UB}}$. If $V_{\mathrm{CCH}} < V_{\mathrm{UB}}$, then $V_{\mathrm{LR3}} < V_{\mathrm{UB}}$.*

Proof. Direct application of Theorem 4.1. □

COROLLARY 7.2. *For* (CCH), *$V_{\mathrm{UB}} = V_{\mathrm{CC}}$ and $V_{\mathrm{CCH}} \leqslant V_{\mathrm{LR5}} \leqslant V_{\mathrm{UB}}$.*

Proof. The set of feasible solutions given by

$$D = \left\{ x_j \in [0, 1],\ j = 1, \ldots, n \mid \sum_{j=1}^{n} x_j = p \right\}$$

is a polytope with integral vertices. Hence, $V_{\mathrm{CC}} = V_{\mathrm{UB}}$. Then, it follows from Proposition 4.2 that $V_{\mathrm{CCH}} \leqslant V_{\mathrm{LR5}} \leqslant V_{\mathrm{UB}}$. □

The above results establish the dominance of bounds obtained from the reformulations (R3) and (R5) over V_{UB}. Recall, from Figure 4, that (R4) and (R7) provide tighter bounds than (R3), and (R6) and (R8) provide tighter bounds than (R5). Thus, the reformulations (R3)–(R8) provide tighter bounds than those used in [37].

Next, we apply the proposed reformulation technique to a numerical instance of (CCH) from the literature. We consider Example 3 of [37] for which $m = 3, n = 6$ and $p = 3$. The values of the parameters a_{ij} and b_{ij} are shown in Table 2. The bounding scheme employed in [37] at the root node provides an upper bound of

Table 2. Data for Example 3 in [37]

	1	2	3	4	5	6
a_{1j}	9	2	3	5	8	7
b_{1j}	6	2	8	2	9	1
a_{2j}	9	8	5	2	1	3
b_{2j}	8	8	7	3	6	1
a_{3j}	9	2	2	4	3	5
b_{3j}	7	2	7	6	4	5

5.119. The problem was eventualy solved to optimality in seven nodes. The optimal objective value is 4.729. Upon applying reformulation (R7) to this example with the bounds on the fractional terms determined by Saipe's algorithm for single ratio problems, we find that the LP relaxation of the reformulation provides the optimal solution.

8. Computational results

In this section, we provide detailed computational results for the cardinality constrained hyperbolic program (CCH). We first compare the bounding strengths of the various reformulation schemes proposed in Section 5. Next, we describe an implementation of the proposed branch and bound strategy to solve this problem to global optimality. Finally, the proposed algorithm is used to solve a facility location problem.

8.1. COMPARISON OF BOUNDS

To compare the bounds obtained from the various reformulations, we solve relaxations for randomly generated instances of (CCH). The coefficients a_{ij} and b_{ij} for problems with different values of m, n and p were generated from a uniform distribution with range [1, 1000]. The reformulations were modeled using GAMS [5] while the LP relaxations were solved using OSL [30] and the NLP relaxations were solved using MINOS [28]. The optimal solution of each problem was obtained by solving the MILP reformulation of type (R8) using OSL [30]. The minimum, average and maximum percentage relaxations gaps and the average CPU times, in seconds, for solving the relaxations are compared in Tables 3–6. The CPU times reported do not include the time spent in deriving bounds on the fractional terms. The averages are computed based upon five instances for each problem type. All computations were carried out on an IBM RS/6000 power PC with 64 MB RAM.

The theoretical dominance of the various relaxation bounds as discussed in Section 4 can be clearly observed in Tables 3 and 8. For some cases the non-linear relaxation of (R2) could not be solved to optimality owing to numerical difficulties with MINOS. This fact results in the discrepancy between the bounds obtained from (R1) and (R2). For these cases, the numbers in parenthesis indicate the number of instances solved. For problem $20-20-12$, the non-linear relaxation of (LR6) could not be solved for any of the five instances. The CPU times reflect the relative sizes of the various reformulations as compared in Table 1. The relaxations (LR2) and (LR6) presumably take more time to optimize due to the presence of non-linearities in these relaxations. From Tables 3–6, it is clear that even though the time taken to solve (LR1)–(LR8) is more than the linear programming relaxation of Li's reformulation [24], the significant improvement in the bounds derived using (LR1)–(LR8) justify their use in a branch and bound algorithm. This is the subject matter of the next subsection.

8.2. PERFORMANCE OF THE PROPOSED ALGORITHM

The proposed branch and bound algorithm was implemented for (CCH) using BARON [35], a general purpose global optimization package. The package employs a branch and reduce optimization strategy, integrating conventional branch and bound with a wide array of domain reduction tools and a combination of branching rules (see [33, 34, 10, 40] for algorithmic details; and [8, 25, 17, 44, 36] for applications). In addition to providing several ready to use modules for certain classes of global optimization problems, BARON is capable of solving any global optimization problem as long as the problem specific upper and lower bounding subroutines are supplied by the user. It is the latter feature of this code we build upon here.

In the implementation of the proposed algorithm for $0-1$ hyperbolic programs, we have augmented the upper bounding step with a node tightening step. For (CCH), tight bounds on the fractional terms were obtained by solving single ratio $0-1$ hyperbolic programs using Saipe's algorithm. These bounds were used in turn to construct tight reformulations of the problem at every node of the branch and bound tree. The continuous relaxation of the reformulation was then solved to obtain the upper bound. The lower bounding was carried out by heuristically solving CCH using the Genetic Algorithm (GA) described in [21]. In our implementation, the GA was applied to the original formulation (H) which permits easy representation of the solutions in the form of binary vectors. In addition to fitness-based updation of the solution population, at each iteration of the branch and bound algorithm, the best solution to the relaxation was rounded to construct a feasible solution to (CCH) and appended to the solution population maintained by the GA. Introduction of these solutions considerably improved the performance of the GA heuristic. After a pre-specified number (100) of children generation, the GA was terminated and the best solution from the population was used to update the lower

Table 3. Comparison of various bounds ($m = 5$)

Problem	Li [24]				R1				R2			
m-n-p	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
5-10-3	25.02	61.57	91.34	0.22	0.00	5.03	11.85	0.34	0.00	5.03	11.85	0.28
5-10-5	61.45	92.25	138.82	0.24	2.30	4.34	6.55	0.33	2.08	4.30	6.55	0.40
5-10-7	69.70	118.20	177.85	0.25	0.03	1.28	3.67	0.33	0.03	1.28	3.67	0.36
5-20-8	60.01	155.06	298.81	0.46	3.53	6.66	13.33	0.80	3.51	6.65	13.33	1.19
5-20-10	59.77	165.75	302.13	0.46	2.29	4.81	7.61	0.81	2.29	4.81	7.61	1.16
5-20-12	63.46	172.50	299.45	0.49	1.53	2.97	4.39	0.84	1.53	2.96	4.39	1.11
	R3				R4				R5			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
5-10-3	0.43	14.01	24.48	0.32	0.00	4.75	10.62	0.87	0.00	2.83	5.24	0.37
5-10-5	2.83	9.68	14.08	0.37	1.09	3.47	5.76	0.98	0.00	2.65	6.55	0.38
5-10-7	2.70	3.66	5.76	0.38	0.02	1.12	3.35	0.97	0.00	0.82	2.37	0.38
5-20-8	11.20	20.46	34.67	0.92	3.22	6.60	13.33	2.67	1.43	5.25	11.67	0.96
5-20-10	8.77	13.96	22.77	0.91	2.29	4.77	7.61	2.98	1.45	3.63	7.03	0.93
5-20-12	5.85	8.31	12.67	0.94	1.53	2.92	4.39	2.78	1.17	2.19	3.11	0.94
	R6				R7				R8			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
5-10-3	0.00	2.83	5.24	0.35	0.00	2.69	4.96	0.36	0.00	1.87	3.21	0.97
5-10-5	0.00	2.65	6.55	0.43	0.00	2.31	4.25	0.39	0.00	1.67	3.08	1.00
5-10-7	0.00	0.81	2.37	0.45	0.04	0.67	1.69	0.37	0.00	0.34	0.92	1.01
5-20-8	1.43	5.25	11.67	1.97	1.30	4.96	11.08	0.99	0.82	4.23	10.30	3.04
5-20-10	1.45	3.63	7.03	2.02	1.30	3.47	7.01	0.97	0.90	2.96	6.18	3.15
5-20-12	1.17	2.19	3.11	1.92	0.89	1.70	2.52	0.94	0.67	1.43	2.27	3.11

bound in the branch and bound algorithm. The GA served to provide good solutions early on in the branch and bound tree. However, no substantial deterioration of the algorithm was observed without this heuristic lower bounding scheme. In this implementation, the conventional integer programming scheme of branching on the variables taking fractional values was used.

Comparing the average gap and the solution times of the various reformulations for (CCH) in Tables 3 and 4, it appears that for these problems, reformulations (R5) and (R7) are the most effective. We chose to use the reformulation (R7) within our branch and bound strategy. Consequently, the upper bounding was carried out by solving the LP relaxation of (R7) by CPLEX 6.0 [7].

To demonstrate the advantages of node tightening, the proposed algorithm has been compared to solving the MIP reformulation (R7) derived at the root node

Table 4. Comparison of various bounds ($m = 10$)

Problem	Li [24]				R1				R2			
m-n-p	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
10-10-3	62.13	90.27	151.22	0.44	0.64	7.70	14.69	0.77	0.59	7.67	14.60	1.30
10-10-5	68.47	109.22	170.35	0.45	2.96	4.01	5.59	0.82	2.94	4.00	5.59	1.19
10-10-7	88.64	134.67	185.09	0.45	0.73	1.27	1.57	0.78	0.68	1.22	1.55	1.02
10-20-8	117.95	137.90	169.09	0.93	8.08	14.76	22.60	2.69	8.08	14.76	22.60	7.99
10-20-10	116.83	142.34	164.42	0.99	5.38	10.01	16.29	2.73	5.38	10.01	16.29	9.10
10-20-12	114.88	146.84	174.72	0.96	4.01	5.38	8.95	2.62	4.00	5.55	8.95	5.52
	R3				R4				R5			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
10-10-3	12.42	24.28	37.02	0.90	0.30	7.39	14.21	2.46	0.00	5.53	11.60	0.91
10-10-5	12.02	14.56	16.24	1.01	2.95	4.00	5.58	2.45	0.44	1.91	4.12	0.95
10-10-7	3.01	3.89	5.30	0.90	0.59	1.22	1.48	2.72	0.37	0.64	1.06	0.85
10-20-8	24.75	33.60	44.20	2.87	8.07	14.74	22.57	10.11	7.04	13.57	20.48	2.78
10-20-10	15.78	21.46	28.94	2.95	5.19	9.93	16.29	11.23	4.06	8.95	14.86	3.02
10-20-12	9.80	12.08	15.86	2.94	3.74	5.24	8.95	11.30	2.55	4.51	7.54	2.78
	R6				R7				R8			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
10-10-3	0.00	5.53	11.59	1.52	0.00	5.24	11.72	1.00	0.00	4.05	9.15	2.61
10-10-5	0.44	1.91	4.12	1.48	0.03	1.35	3.33	1.24	0.00	1.20	3.01	2.97
10-10-7	0.33	0.60	1.05	1.46	0.08	0.63	1.16	0.84	0.07	0.38	0.90	2.88
10-20-8	7.04	13.57	20.48	8.69	6.80	13.12	20.11	3.22	6.08	12.22	18.96	11.27
10-20-10	4.06	8.95	14.86	10.82	4.08	8.46	14.50	3.19	3.57	7.76	13.56	12.10
10-20-12	2.55	4.51	7.54	8.70	2.41	4.27	7.56	3.10	2.13	3.85	6.85	11.27

using the commercial MIP solver of CPLEX 6.0 [7]. The comparison was carried out on a set of random test problems generated similarly as in Section 8.1. All computations were carried out on an IBM RS/6000 power PC with 64MB RAM.

The results of the comparison are presented in Tables 7–10. For each of the five instances of the same problem size, the tables show the total number of nodes explored, the node at which the optimal solution was found, the maximum number of nodes held in memory during the search and the total CPU seconds required for both algorithms. In all cases, the proposed algorithm required fewer nodes than CPLEX. The differences were much greater for problems with a larger number of binary variables. In several cases, CPLEX was unable to solve the problems to optimality within 1000000 iterations. As the previous section has demonstrated that Li's formulation [24] provides weaker bounds than (R7), using Li's formu-

Table 5. Comparison of various bounds ($m = 15$)

Problem	Li [24]				R1				R2			
m-n-p	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
15-10-3	96.23	191.06	214.77	1.74	8.58	8.83	9.82	4.93	8.58	8.83	9.81	19.59
15-10-5	109.39	109.95	110.08	0.74	6.77	6.94	6.98	1.50	6.77	6.94	6.98	3.43
15-10-7	93.85	153.04	324.63	0.73	0.55	2.31	6.77	1.40	0.52	2.30	6.77	2.55
15-20-8	140.33	167.02	205.17	1.69	16.27	23.00	31.28	5.59	22.52	26.90	31.28	$34.77^{(2)}$
15-20-10	132.53	166.85	211.20	1.80	10.18	13.83	18.87	6.70	10.18	10.34	10.49	$33.19^{(2)}$
15-20-12	130.00	167.63	214.77	2.05	3.38	6.47	8.58	5.66	3.38	6.08	8.58	$18.21^{(4)}$
	R3				R4				R5			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
15-10-3	19.18	22.73	36.93	4.72	8.58	8.80	9.68	21.49	6.84	7.57	7.75	4.63
15-10-5	16.23	16.40	16.44	1.70	6.76	6.93	6.98	5.74	4.98	5.27	5.34	1.52
15-10-7	2.19	6.25	16.23	1.77	0.21	2.22	6.76	6.15	0.00	1.30	4.98	1.61
15-20-8	37.92	44.19	55.47	5.82	16.26	22.98	31.28	22.50	13.78	19.89	27.19	5.33
15-20-10	21.76	27.25	34.34	6.12	10.18	13.83	18.87	27.75	8.50	12.21	16.74	5.85
15-20-12	10.49	14.87	19.18	5.90	3.38	6.47	8.58	26.41	2.84	5.67	7.75	5.48
	R6				R7				R8			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
15-10-3	6.84	7.57	7.75	23.22	7.75	8.42	8.58	6.19	6.14	7.43	7.75	26.01
15-10-5	4.98	5.27	5.34	3.88	4.71	4.96	5.02	1.98	4.52	4.59	4.60	6.54
15-10-7	0.00	1.30	4.98	3.58	0.00	1.30	4.71	1.66	0.00	1.09	4.52	6.67
15-20-8	13.78	19.89	27.19	20.67	15.26	20.04	26.40	6.82	13.29	18.87	25.68	25.75
15-20-10	8.50	12.20	16.74	25.08	8.53	12.25	16.58	6.99	8.24	11.69	15.90	27.50
15-20-12	2.84	5.66	7.75	24.84	2.67	5.87	8.58	6.25	2.22	5.21	7.29	26.88

lation instead of (R7) in these computations would result in an even larger tree. Therefore, we do not report the time it takes to solve the hyperbolic program to optimality using Li's reformulation. The results of Tables 7–10 clearly demonstrate the importance of node tightening in obtaining tighter bounds for the problems and hence faster convergence.

8.3. *p*-CHOICE FACILITY LOCATION

We describe, as an application of (CCH), the p-choice facility location problem. In facility location, the attractiveness of a particular site is often measured in terms of the market share associated with it. In p-choice location models [11, 12], the market share of a location is measured in terms of the ratio of the utility of that

Table 6. Comparison of various bounds ($m = 20$)

Problem	Li [24]				R1				R2			
m-n-p	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
20-10-3	79.81	85.89	91.19	0.97	0.30	7.37	13.47	2.08	0.25	7.33	13.40	4.77
20-10-5	92.94	125.77	203.86	0.91	7.56	8.59	9.51	2.56	7.56	8.58	9.50	6.77
20-10-7	95.48	140.33	223.93	1.07	1.82	2.15	2.81	2.27	1.76	2.11	2.73	4.20
20-20-8	135.65	204.11	320.16	2.67	14.67	23.19	27.86	10.36	14.67	19.94	25.21	58.74[(2)]
20-20-10	142.09	206.42	327.37	2.57	10.57	16.21	20.63	11.98	10.57	10.57	10.57	48.86[(1)]
20-20-12	147.12	207.51	331.69	2.96	6.14	8.51	9.90	10.86	9.49	9.49	9.49	32.72[(1)]
	R3				R4				R5			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
20-10-3	19.13	27.52	36.75	2.30	0.30	7.26	12.97	8.01	0.00	5.58	12.14	2.46
20-10-5	16.06	17.18	19.72	2.68	7.56	8.50	9.50	10.17	5.70	6.73	7.67	2.63
20-10-7	4.15	4.95	5.79	2.55	1.71	2.07	2.74	10.24	0.97	1.41	1.86	2.48
20-20-8	35.80	46.34	51.98	10.00	14.67	23.17	27.86	44.06	12.20	19.82	24.88	9.03
20-20-10	23.46	29.41	32.97	9.80	10.57	16.21	20.63	47.78	9.03	14.07	18.45	10.22
20-20-12	13.66	16.70	18.45	10.15	6.14	8.51	9.90	48.96	5.14	7.47	8.84	11.11
	R6				R7				R8			
	min	ave	max	CPU s	min	ave	max	CPU s	min	ave	max	CPU s
20-10-3	0.00	5.57	12.08	6.05	0.00	5.80	11.50	2.72	0.00	4.72	10.20	9.97
20-10-5	5.70	6.73	7.67	8.29	4.65	6.51	7.69	2.94	4.12	5.82	6.63	11.75
20-10-7	0.96	1.41	1.86	6.27	1.08	1.41	1.93	2.52	0.62	1.04	1.43	12.72
20-20-8	12.20	19.82	24.88	34.19	12.17	20.32	25.17	11.73	11.36	19.09	23.58	50.56
20-20-10	9.03	14.07	18.45	42.85	8.86	14.31	18.44	12.40	8.34	13.85	18.21	54.64
20-20-12	—	—	—	—	5.05	7.65	9.27	12.81	4.71	7.06	8.55	54.15

location to the sum of the utilities of all available locations to the consumers. Consider the problem of locating p facilities in n possible locations to service m customer locations with the objective of maximizing market share. Let U_{ij} denote the utility of location j to the customers at i, d_i be the demand at customer location i and w_j be a preferential weight for a particular location j. Then, the problem of determining the set of facility locations S to maximize the weighted market share

Table 7. Computational results for CCH ($m = 5$)

Problem		CPLEX 6.0				BARON			
m-n-p	No.	Ntot	Nopt	Nmem	CPU s	Ntot	Nopt	Nmem	CPU s
5-10-3	1	1	1	1	0.12	1	1	1	0.20
5-10-3	2	16	7	13	0.28	7	1	3	0.70
5-10-3	3	18	5	17	0.23	3	1	2	0.40
5-10-3	4	12	8	11	0.20	3	1	2	0.50
5-10-3	5	1	1	1	0.14	1	1	1	0.20
5-10-5	1	1	1	1	0.16	1	1	1	0.20
5-10-5	2	10	6	5	0.23	5	1	2	0.60
5-10-5	3	8	6	7	0.20	5	1	2	0.50
5-10-5	4	17	9	17	0.23	7	1	3	0.60
5-10-5	5	1	1	1	0.16	1	1	1	0.20
5-10-7	1	3	3	3	0.17	1	1	1	0.30
5-10-7	2	9	7	9	0.22	5	1	2	0.50
5-10-7	3	7	3	2	0.20	3	2	2	0.50
5-10-7	4	5	5	5	0.14	1	1	1	0.30
5-10-7	5	9	6	8	0.22	3	1	2	0.50

can be formulated as follows [21]:

$$
\begin{aligned}
(P) \qquad \max \quad & \sum_{j \in S} w_j \sum_{i=1}^{m} \frac{U_{ij}}{\sum_{k \in S} U_{ik}} d_i \\
\text{s.t.} \quad & |S| = p \\
& S \subseteq \{1, 2, \ldots, n\}
\end{aligned}
$$

Introducing binary variables x_j equal to 1 if location j is chosen and 0 otherwise, the formulation becomes:

$$
\begin{aligned}
(P) \qquad \max \quad & \sum_{i=1}^{m} \frac{\sum_{j=1}^{n} U_{ij} w_j d_i x_j}{\sum_{j=1}^{n} U_{ij} x_j} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_j = p \\
& x_j \in \{0, 1\} \qquad j = 1, 2, \ldots, n
\end{aligned}
$$

Table 8. Computational results for CCH ($m = 5$)

Problem		CPLEX 6.0				BARON			
m-n-p	No.	Ntot	Nopt	Nmem	CPU s	Ntot	Nopt	Nmem	CPU s
5-20-8	1	127	83	125	2.89	15	1	5	2.40
5-20-8	2	24	20	22	0.66	5	4	2	1.10
5-20-8	3	29	22	25	1.05	5	1	2	1.20
5-20-8	4	15	12	13	0.53	3	2	2	1.00
5-20-8	5	73	34	60	1.43	21	1	4	2.90
5-20-10	1	103	73	83	2.91	21	1	6	3.10
5-20-10	2	29	17	28	0.77	3	1	2	1.10
5-20-10	3	58	45	46	1.40	15	12	4	2.60
5-20-10	4	23	12	22	0.67	5	4	2	1.20
5-20-10	5	41	36	41	1.13	7	1	2	1.50
5-20-12	1	42	37	40	1.21	9	1	2	1.60
5-20-12	2	17	11	15	0.75	5	4	2	1.00
5-20-12	3	36	29	36	1.18	7	6	3	1.50
5-20-12	4	17	13	17	0.79	5	1	2	1.20
5-20-12	5	40	19	39	1.07	11	1	4	1.90
5-50-23	1	45949	30293	30595	2185	574	145	108	222
5-50-23	2	702881	187494	344341	28247	7715	10	1401	2988
5-50-23	3	40042	32275	37739	2536	667	10	130	241
5-50-23	4	—	—	—	—	6938	5164	1457	2488
5-50-23	5	615930	179449	179471	31977	2790	622	448	940
5-50-25	1	69187	33848	54399	3020	975	10	169	374
5-50-25	2	506225	125633	184451	19372	6959	6728	1230	2534
5-50-25	3	32846	24686	32001	1905	877	19	150	329
5-50-25	4	—	—	—	—	4349	28	714	1609
5-50-25	5	—	—	—	—	5261	1788	946	1943
5-50-27	1	36609	18897	36609	1622	641	19	112	233.10
5-50-27	2	203792	44385	192151	7328	4813	37	816	1702
5-50-27	3	23602	15465	20149	1343	637	1	115	251
5-50-27	4	—	—	—	—	5079	3420	893	1803
5-50-27	5	—	—	—	—	11811	19	1909	3704

Table 9. Computational results for CCH ($m = 10$)

Problem		CPLEX 6.0				BARON			
m-n-p	No.	Ntot	Nopt	Nmem	CPU s	Ntot	Nopt	Nmem	CPU s
10-10-3	1	9	6	8	0.55	5	1	2	1.20
10-10-3	2	1	1	1	0.34	1	1	1	0.50
10-10-3	3	19	10	18	0.94	7	1	2	1.60
10-10-3	4	15	8	15	0.88	7	1	3	1.50
10-10-3	5	2	2	2	0.37	1	1	1	0.60
10-10-5	1	8	5	6	0.60	5	1	2	1.40
10-10-5	2	16	12	15	0.95	3	1	2	0.80
10-10-5	3	35	21	35	1.42	7	1	2	1.50
10-10-5	4	11	7	10	0.72	5	1	2	1.30
10-10-5	5	9	7	8	0.62	3	1	2	0.90
10-10-7	1	24	18	21	0.73	7	1	3	1.40
10-10-7	2	9	7	8	0.58	1	1	1	0.60
10-10-7	3	14	5	4	0.57	3	1	2	1.10
10-10-7	4	6	5	6	0.38	1	1	1	0.70
10-10-7	5	8	6	7	0.57	5	1	2	1.20
10-20-8	1	1521	534	1518	104.91	234	231	54	70.50
10-20-8	2	616	310	597	58.28	85	54	17	31.30
10-20-8	3	60	44	55	7.38	13	1	4	5.90
10-20-8	4	663	389	636	54.06	99	1	19	35.00
10-20-8	5	145	133	144	14.66	17	1	4	6.90
10-20-10	1	1804	558	735	121.68	247	1	49	75.80
10-20-10	2	185	83	84	16.51	39	1	11	14.80
10-20-10	3	87	74	86	10.04	13	1	4	5.50
10-20-10	4	452	219	451	39.09	61	1	13	24.30
10-20-10	5	216	168	204	20.99	21	14	6	8.90
10-20-12	1	611	200	611	45.29	79	24	15	31.30
10-20-12	2	59	34	35	6.82	17	1	6	7.00
10-20-12	3	79	45	79	6.86	19	1	4	7.10
10-20-12	4	104	65	101	10.67	27	1	6	10.80
10-20-12	5	96	42	95	7.42	31	31	7	11.20

Table 10. Computational results for CCH ($m = 20$)

Problem		CPLEX 6.0				BARON			
m-n-p	No.	Ntot	Nopt	Nmem	CPU s	Ntot	Nopt	Nmem	CPU s
20-20-8	2	2373	770	1057	536	267	228	55	246
20-20-8	3	2430	783	739	524	229	1	50	242
20-20-8	4	598	160	150	167	103	32	26	98
20-20-8	5	1072	441	1004	304	117	116	28	120
20-20-10	1	3080	1177	1575	782	299	248	60	297
20-20-10	2	1668	813	1658	489	177	76	36	170
20-20-10	3	6988	2332	3431	1415	565	1	111	533
20-20-10	4	670	201	181	176	93	1	22	85
20-20-10	5	1293	671	1192	400	125	1	31	128
20-20-12	1	1456	476	508	383	189	1	35	197
20-20-12	2	518	266	285	164	99	94	16	95
20-20-12	3	2381	780	2326	552	251	30	46	246
20-20-12	4	349	151	330	113	45	34	10	49
20-20-12	5	666	277	632	207	85	1	18	83
20-30-12	1	178111	67041	86196	55702	3689	1	715	5548
20-30-12	2	56521	19253	33484	20289	1241	1	229	2073
20-30-12	3	68910	20357	13104	24720	1727	1	357	2344
20-30-12	4	248326	75427	117556	76551	4643	1	905	7593
20-30-12	5	41233	13527	41233	14523	1302	995	280	2178
20-30-14	1	173519	59322	91570	56777	3841	3252	765	5510
20-30-14	2	59084	21858	57927	23099	1636	1613	317	2361
20-30-14	3	47991	20725	47076	22763	1135	1	216	1811
20-30-14	4	141853	62496	72827	53159	2489	2278	473	4127
20-30-14	5	92518	36266	44205	33678	2351	1	455	3476
20-30-16	1	104255	44747	99248	41081	3187	271	611	4547
20-30-16	2	27097	8781	10772	11057	1033	1	193	1493
20-30-16	3	22456	9307	18964	10175	585	520	102	1087
20-30-16	4	55943	22708	29132	22709	1501	1	265	2521
20-30-16	5	38832	16381	36904	17047	1353	1	256	1912

The proposed algorithm was applied to the problem of locating a set of 10 restaurant franchises in the city of Edmonton, Canada, using the formulation (P). The city of Edmonton is divided into 886 enumeration areas (EA), or zones for Federal Census purposes. Using geographic information system (GIS) data, the centroids of each of these zones and their inter-spatial distances were calculated. 100 of the most populated EA centroids were chosen as demand points and 58 of these as candidate locations for restaurants. The criterion for choosing these sites was the existence of at least three other restaurants at the location. For each of the demand points, the total expenditure on existing restaurants was assumed to represent the demand for that location. The utility of a prospective location to a customer at a particular demand point was estimated using a multiplicative competition interaction criteria [29] as a function of distance, accessibility, and a number of attractiveness factors. The preferential weight of a location was estimated using the index of retail saturation [11] which is a function of population and per capita expenditure. Detailed description of the data for this problem is presented in [21].

Using the above data, the resulting p-choice model consists of a sum of 100 fractional terms, 58 binary variables and a cardinality requirement of 10. Using reformulation (R7) the resulting MIP has 5900 continuous variables, 58 binary variables, and 23501 constraints. The problem was solved to global optimality using the proposed algorithm on a single processor of an IBM RS/6000 SP2 with 512MB RAM in 5.5 hours. Ninety-seven percent of the CPU time was consumed by CPLEX 6.0 in solving the linear programming relaxations. In particular, 10% of the total CPU time was spent in solving the LP relaxation at the root node. Global optimality was proven in a total of 79 nodes with at most nine nodes stored in memory. The optimal solution was located at the tenth node.

References

1. Agrawal, S.C.. (1977), An alternative method of integer solutions to linear fractional functionals by a Bbranch and bound technique. *Z. Angew. Math. Mech.* 57: 52–53.
2. Al-Khayyal, F.A. and Falk, J.E. (1983), Jointly constrained biconvex programming. *Mathematics of Operations Research* 8: 273–286.
3. Arora, S.R., K. Swarup, K. and Puri, M.C. (1977), The set covering problem with linear fractional functional. *Indian Journal of Pure and Applied Mathematics* 8: 578–588.
4. Blum, M., Floyd, R.W., Pratt, V., Rivest R.L. and Tarjan, R.E. (1973), Time bounds for selection. *Journal of Computer and System Sciences* 7: 448–461.
5. Brook, A., Kendrick, D. and Meeraus, A. (1988), *GAMS–A User's Guide*. Scientific Press, Redwood City, CA.
6. Charnes, A. and Cooper, W.W. (1962), Programming with linear fractional functionals. *Naval Research Logistics Quarterly* 9: 181–186.
7. CPLEX. (1997), CPLEX 6.0 User's Manual. ILOG CPLEX Division, Incline Village, NV.
8. Dorneich, M.C. and Sahinidis, N.V. (1995), Global optimization algorithms for chip layout and compaction. *Engineering Optimization* 25: 131–154.
9. Falk, J.E. and Polocsay, S.W. (1994), Image space analysis of generalized fractional programs. *Journal of Global Optimization* 4: 63–88.

10. Ghildyal, V. and Sahinidis, N.V. (2001), Solving global optimization problems with BARON. In: Migdalas, A., Pardalos, P., Varbrand, P. and Holmqvist, K. (eds.), *From Local to Global Optimization. A Workshop on the Occasion of the 70th Birthday of Professor Hoang Tuy*, Kluwer Academic Publishers, Boston, MA.
11. Ghosh, A. and McLafferty, S. (1987), *Location Strategies for Retail and Service Firms*. Lexington Books, Massachusetts.
12. Ghosh, A., McLafferty, S. and Craig, S. (1995), Multifacility retail networks. In: Drezner Z. (ed.), *Facility Location: A Survey of Applications and Methods*, Springer, New York, pp. 301–330.
13. Gilmore, P.C. and Gomory, R.E. (1963), A linear programming approach to the cutting stock problem – Part II. *Operations Research* 11: 52–53.
14. Granot, D. and Granot, F. (1976), On solving fractional $(0-1)$ programs by implicit enumeration. *INFOR* 14: 241–249.
15. Granot, D. and Granot, F. (1977), On integer and mixed integer fractional programming problems. *Annals of Discrete Mathematics* 1: 221–231.
16. Grunspan, M. and Thomas, M.E. (1973), Hyperbolic integer programming. *Naval Research Logistics Quarterly* 20: 341–356.
17. Gutierrez, R.A. and Sahinidis, N.V. (1996), A branch-and-bound approach for machine selection in just-in-time manufacturing systems. *International J. Production Research* 34: 797–818.
18. Hammer, P.L. and Rudeanu, S. (1968), *Boolean Methods in Operations Research and Related Areas*. Springer, New York.
19. Hansen, P., de Aragao, M.V.P. and Ribeiro, C.C. (1991), Hyperbolic $0-1$ programming and query optimization in information retrieval. *Mathematical Programming* 52: 255–263.
20. Hansen, P., Jaumard, B. and Mathon, V. (1993), Constrained nonlinear 0-1 programming. *ORSA Journal of Computing* 5: 87–119.
21. Haque, M.A. and Ahmed, S. (1998), *p*-Choice facility location in discrete space. *in preparation*.
22. Hashizume, S., Fukushima, M., Katoh, N. and Ibaraki, T. (1987), Approximation algorithms for combinatorial fractional programming problems. *Mathematical Programming* 37: 255–267.
23. Hiriart-Urruty, J. and Lemaréchal, C. (1993), *Convex Analysis and Minimization Algorithms I*. Springer, Berlin.
24. Li, H. (1994), A global approach for general $0-1$ fractional programming. *European Journal of Operational Research* 73: 590–596.
25. Liu, M.L., Sahinidis N.V. and Shectman, J.P. (1996), Planning of chemical process networks via global concave minimization. In: Grossmann I.E. (ed.), *Global Optimization in Engineering Design*. Kluwer Academic Publishers, Boston, MA. Chapt. 7, pp. 195–230.
26. McCormick, G.P. (1982), *Nonlinear Programming: Theory, Algorithms and Applications*. John Wiley and Sons, New York.
27. Megiddo, N. (1979), Combinatorial optimization with rational objective functions. *Mathematics of Operations Research* 4: 414–424.
28. Murtagh, B.A. and Saunders, M.A. (1995), *MINOS 5.4 User's Guide*. Technical Report SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, CA.
29. Nakanishi, M. and Cooper, L.G. (1974), Parameter estimate for multiplicative interactive choice models: least squares approach. *Journal of Marketing Research* 11: 303–311.
30. OSL. (1995, Optimization subroutine library guide and reference release 2.1. International Business Machines Corporation, Kingston, NY, fifth edition.
31. Quesada, I. and Grossmann, I.E. (1995), A global optimization algorithm for linear fractional and bilinear programs. *Journal of Global Optimization* 6: 39–76.

32. Robillard, P. (1971), (0, 1) Hyperbolic programming problems. *Naval Research Logistics Quarterly* 18: 47–57.
33. Ryoo, H.S. and Sahinidis, N.V. (1995), Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering* 19: 551–566.
34. Ryoo, H.S. and Sahinidis, N.V. (1996), A branch-and-reduce approach to global optimization. *Journal of Global Optimization* 8: 107–139.
35. Sahinidis, N.V. (1996), BARON: A general purpose global optimization software package. *Journal of Global Optimization* 8: 201–205.
36. Sahinidis, N.V. and Tawarmalani, M. (2000), Applications of global optimization to process and molecular design. *Computers & Chemical Engineering* 24: 2157–2169.
37. Saipe, A.L. (1975), Solving a (0, 1) hyperbolic program by branch and bound. *Naval Research Logistics Quarterly* 22: 497–515.
38. Schaible, S. (1995), Fractional Programming. In: Horst, R. and Pardalos, P.M. (eds.) *Handbook of Global Optimization*. Kluwer Academic Publishers, Norwell, Massachusetts. pp. 495–608.
39. Schaible, S. (1996), Fractional programming with sums of ratios. working paper 96-04, A.G. Anderson Graduate School of Management, University of California, Riverside.
40. Shectman, J.P. and Sahinidis, N.V. (1998), A finite algorithm for global minimization of separable concave programs. *Journal of Global Optimization* 12: 1–36.
41. Stancu-Minasian, I.M. (1997), *Fractional Programming*. Kluwer Academic Publishers, Dordrecht.
42. Tawarmalani, M., Ahmed S. and Sahinidis, N.V. (submitted 2001), Product disaggregation in global optimation and an application to rational programs *Optimization and Engineering*.
43. Tawarmalani, M. and N.V. Sahinidis, N.V. (accepted 2002), Convex extensions and convex envelopes of l.s.c. functions. *Mathematical Programming*.
44. VanAntwerp, J.G., Braatz, R.D. and Sahinidis, N.V. (1999), Globally optimal robust control. *Journal of Process Control* pp. 375–383.
45. Williams, H.P. (1974), Experiments in the formulation of integer programming problems. *Mathematical Programming Study* 2: 180–197.
46. Wu, T. (1997), A Note on a global approach for general 0-1 fractional programming. *European Journal of Operational Research* 101 220–223.