To appear in *Optimization*
Vol. 00, No. 00, Month 20XX, 1–17

Primal and dual multi-objective linear programming algorithms for linear multiplicative programmes

Lizhen Shao[a,b*] and Matthias Ehrgott[b]

[a]*School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China*; [b]*Department of Management Science, Lancaster University Management School, Bailrigg, Lancaster LA1 4YX, United Kingdom*;

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Multiplicative programming problems (MPPs) are global optimisation problems known to be NP-hard. In this paper, we employ algorithms developed to compute the entire set of nondominated points of multi-objective linear programming (MOLP) problems to solve linear MPPs. First, we improve our own objective space cut and bound algorithm for convex MPPs in the special case of linear MPPs by only solving one linear programme in each iteration instead of two as the previous version indicates. We call this algorithm, which is based on Benson's outer approximation algorithm for MOLP problems, the primal objective space algorithm. Then, based on the dual variant of Benson's algorithm, we propose a dual objective space algorithm for solving linear MPPs. The dual algorithm also requires solving only one linear programme in each iteration. We prove the correctness of the dual algorithm and use computational experiments comparing our MOLP based algorithms to a recent global optimisation algorithm for linear MPPs from the literature as well as two general global solvers to demonstrate the superiority of the new algorithms in terms of computation time. Thus we demonstrate that the use of multi-objective optimisation techniques can be beneficial to solve difficult single objective global optimisation problems.

Keywords: linear multiplicative programming; multi-objective optimisation; approximation algorithm; nondominated point.

1. Introduction

Consider the linear multiplicative programming problem (MPP),

$$(\mathrm{P}_{\mathcal{X}}) \quad \begin{aligned} &\min \phi(x) = \prod_{i=1}^{p} (c_i^T x + d_i) \\ &\text{s.t. } x \in \mathcal{X} = \{x \in \mathbb{R}^n : Ax \geqq b\}. \end{aligned}$$

$\mathcal{X}$ is the feasible set in decision (or variable) space $\mathbb{R}^n$. We assume that $\mathcal{X}$ is nonempty, $c_i^T x + d_i > 0, i = 1, \ldots, p$ for all $x \in \mathcal{X}$ and the minimal value of $c_i^T x + d_i$ over $\mathcal{X}$ exists. Let $P(x) = Cx + d = (c_1^T x + d_1, \ldots, c_p^T x + d_p)^T$. $\mathcal{Y} = \{P(x) : x \in \mathcal{X}\}$ is called the feasible set in objective (or outcome) space $\mathbb{R}^p$. Let $\bar{x}$ be an optimal solution of $(\mathrm{P}_{\mathcal{X}})$ and $\phi(\bar{x})$ the corresponding optimal value. Let $\epsilon > 0$ and let x^* be a feasible solution of $(\mathrm{P}_{\mathcal{X}})$, $x^* \in \mathcal{X}$. Then x^* is called ϵ-optimal solution of $(\mathrm{P}_{\mathcal{X}})$ if $\phi(x^*) \leqq \phi(\bar{x})(1 + \epsilon)$.

Problem $(\mathrm{P}_{\mathcal{X}})$ is a nonconvex global optimisation problem. It is known that problem $(\mathrm{P}_{\mathcal{X}})$ may possess several local minima and is an NP-hard problem, even

*Corresponding author. Email: l.shao@ustb.edu.cn

when $p = 2$ [1]. Problem ($P_{\mathcal{X}}$) has attracted considerable attention in the literature because of practical applications in various fields of study, including economic analysis [2], bond portfolio optimisation [3], VLSI chip design [4], multi-objective optimisation [5] and so on.

A traditional way to solve multiplicative programming problems is to consider the problem in objective space $\mathbb{R}^p$ and most existing algorithms work in objective space. Benson and Boger [6] use a cutting plane method to solve linear MPPs, whereas Kuno [7] and Gao et al. [8] use branch and bound methods. Ryoo and Sahinidis [9] develop algorithms for linear multiplicative and generalized linear multiplicative programs based upon a lower bounding procedure and greedy branching schemes. Kim et al. [10] propose an outcome-space outer approximation algorithm for linear multiplicative programming. Depetrini and Locatelli [11] propose an algorithm for minimizing the product of two affine functions over a polyhedral set. Correspondences between multi-objective programming (MOP) problems and multiplicative programming problems were first pointed out by Benson and Boger [12]. They have shown that an optimal solution of an MPP problem is an efficient solution of a corresponding multi-objective optimisation problem. Thus algorithms for solving MOP problems can be adapted to solve MPP problems.

In earlier work, we have modified the approximation algorithm of [13] for solving convex MOP problems into an objective space cut and bound algorithm to solve convex MPPs. In this paper we pursue this relationship further for the special case of linear MPPs of the form ($P_{\mathcal{X}}$) and multi-objective linear programmes (MOLPs) (P),

$$\begin{aligned} \text{(P)}\ \min & P(x) = (c_1^T x + d_1, \ldots, c_p^T x + d_p)^T \\ & \text{s.t. } x \in \mathcal{X} = \{x \in \mathbb{R}^n : Ax \geqq b\}, \end{aligned}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} = \{P(x) : x \in \mathcal{X}\} \subseteq \mathbb{R}^p$ are, as before, the feasible set in variable space and objective space, respectively. We understand the minimisation here as finding the weakly nondominated points of $P(\mathcal{X})$. The relationship between linear MPPs and MOLPs imply that the nonconvex global optimisation problem ($P_{\mathcal{X}}$) can be solved by solving multi-objective linear programme (P) in objective space (see Section 2). Hence, the question arises whether a multi-objective linear programming approach to solve ($P_{\mathcal{X}}$) can be competitive with single objective nonconvex global optimisation approaches. We show that this is the case.

The main contributions of the paper are: (1) we exploit the linearity of the objective functions to improve the objective space cut and bound algorithm of [14] by only solving one linear programme (LP) in each iteration rather than two as the previous original version requires; (2) we turn the dual variant of Benson's multi-objective linear programming algorithm [15] into a (new) sandwich algorithm for MOLP problems; and (3) based on the sandwich version of the dual Benson MOLP algorithm, we propose a dual objective space algorithm to solve linear MPP problems.

The paper is organised as follows. In Section 2 we first introduce some notation, then we review the relationships between linear MPPs and multi-objective linear programming problems. The proposed primal and dual algorithms for MPPs as well as an illustrative example are presented in Section 3. Numerical results are presented in Section 4. Here, we compare the performance of the primal and dual MOLP based algorithms proposed in Section 3 with a recent global optimisation algorithm for linear MPPs from the literature as well as two global solvers. These numerical results demonstrate the superiority of the new MOLP based algorithms

over general global optimisation approaches for linear MPPs. Finally, we draw some conclusions in Section 5.

2. Multiplicative programming problems and multi-objective optimisation

In this paper we use the notation $y^1 \leq y^2$ to indicate $y^1 \leqq y^2$ but $y^1 \neq y^2$ for $y^1, y^2 \in \mathbb{R}^p$ whereas $y^1 < y^2$ means $y^1_k < y^2_k$ for all $k = 1, \ldots, p$. The k-th unit vector in $\mathbb{R}^p$ is denoted by e^k and a vector of all ones is denoted by e. Given a mapping $P : \mathbb{R}^n \to \mathbb{R}^p$ and a subset $\mathcal{X} \subseteq \mathbb{R}^n$ we write $P(\mathcal{X}) := \{P(x) : x \in \mathcal{X}\}$.

Let $\mathcal{A} \subseteq \mathbb{R}^p$. We denote the interior and the boundary of $\mathcal{A}$ by $\operatorname{int}\mathcal{A}$ and $\operatorname{bd}\mathcal{A}$. Furthermore, suppose $\mathcal{A}$ is a polyhedral convex set, then a convex subset $\mathcal{F} \subseteq \mathcal{A}$ is called a *face* of $\mathcal{A}$ if for all $y^1, y^2 \in \mathcal{A}$ and $\alpha \in (0,1)$ such that $\alpha y^1 + (1-\alpha)y^2 \in \mathcal{F}$ it holds that $y^1, y^2 \in \mathcal{F}$. A face $\mathcal{F}$ of $\mathcal{A}$ is called *proper* if $\emptyset \neq \mathcal{F} \neq \mathcal{A}$. A point $y \in \mathcal{A}$ is called an *extreme point* of $\mathcal{A}$ if $\{y\}$ is a face of $\mathcal{A}$. Extreme points are also called vertices. The set of all vertices of a polyhedral convex set $\mathcal{A}$ is denoted by $\operatorname{vert}\mathcal{A}$.

We consider two ordering cones, namely

$$\mathcal{K} := \mathbb{R}_{\geqq} e^p = \{y \in \mathbb{R}^p : y_1 = \cdots = y_{p-1} = 0, y_p \geqq 0\}$$

and

$$\mathbb{R}^p_{\geqq} = \{x \in \mathbb{R}^p : x_k \geqq 0, k = 1, \ldots, p\}.$$

The set of $\mathcal{K}$-maximal elements of $\mathcal{A}$ is given by

$$\max_{\mathcal{K}}\mathcal{A} := \{y \in \mathcal{A} : (\{y\} + \mathcal{K} \setminus \{0\}) \cap \mathcal{A} = \emptyset\}.$$

The set of (weakly) $\mathbb{R}^p_{\geqq}$-minimal elements of $\mathcal{A}$ (also called the set of (weakly) nondominated points of $\mathcal{A}$) is given by

$$\operatorname{wmin}_{\mathbb{R}^p_{\geqq}}\mathcal{A} := \left\{y \in \mathcal{A} : (\{y\} - \operatorname{int}\mathbb{R}^p_{\geqq}) \cap \mathcal{A} = \emptyset\right\},$$
$$\operatorname{min}_{\mathbb{R}^p_{\geqq}}\mathcal{A} := \left\{y \in \mathcal{A} : (\{y\} - \mathbb{R}^p_{\geqq}) \setminus \{0\} \cap \mathcal{A} = \emptyset\right\}.$$

We also use the notation $\mathcal{A}_{WN} = \operatorname{wmin}_{\mathbb{R}^p_{\geqq}}\mathcal{A}$ and $\mathcal{A}_N = \operatorname{min}_{\mathbb{R}^p_{\geqq}}\mathcal{A}$ to denote the set of (weakly) $\mathbb{R}^p_{\geqq}$-minimal or (weakly) nondominated points of $\mathcal{A}$.

For MOLP (P), i.e., $\operatorname{wmin}_{\mathbb{R}^p_{\geqq}} P(\mathcal{X})$, feasible solutions $x \in \mathcal{X}$ such that $P(x)$ is a (weakly) nondominated element of $P(\mathcal{X})$ are frequently called (weakly) efficient solutions in the literature. It is well known that the image $\mathcal{Y}$ of a nonempty polyhedron $\mathcal{X}$ under a linear map P is also a nonempty polyhedron of dimension $\dim\mathcal{Y} \leqq p$.

Most methods to solve multiplicative programming problem ($\mathrm{P}_{\mathcal{X}}$) consider a formulation of the problem in objective space $\mathbb{R}^p$ and solve it in objective space. For a point $y \in \mathbb{R}^p$, let $T(y) = \prod_{i=1}^{p} y_i$. A direct objective space formulation of ($\mathrm{P}_{\mathcal{X}}$)

is

$$(\mathrm{P}_{\mathcal{Y}}) \qquad \min T(y) \text{ subject to } y \in \mathcal{Y}.$$

It has been proved that any global optimal solution to problem $(\mathrm{P}_{\mathcal{Y}})$ must belong to the nondominated set $\mathcal{Y}_N$ (see, e.g., Proposition 2.1 of [14]). Therefore, an optimal solution of $(\mathrm{P}_{\mathcal{Y}})$ is the same for any objective space formulation $\min\{T(y) : y \in \bar{\mathcal{Y}}\}$ as long as $\mathcal{Y}_N = \bar{\mathcal{Y}}_N$ is guaranteed.

We introduce the following polyhedral convex set $\mathcal{P}$. Let

$$\mathcal{P} = \{y \in \mathbb{R}^p : P(x) \leqq y \text{ for some } x \in \mathcal{X}\},$$

i.e., $\mathcal{P} = \mathcal{Y} + \mathbb{R}^p_{\leqq}$ which we call the extended feasible set in objective space.

THEOREM 2.1 ([13]) *The following statements hold.*

(1) $\mathcal{P} \subseteq \mathbb{R}^p$ *is a nonempty set of dimension* p *and* $\mathcal{P}$ *is* $\mathbb{R}^p_{\leqq}$*-bounded from below.*
(2) $\mathcal{Y}_N = \mathcal{P}_N$.
(3) $\operatorname{vert}\mathcal{P} \subseteq \mathcal{P}_N$.
(4) $\mathcal{P}_{WN} = \operatorname{bd}\mathcal{P}$.

According to Theorem 2.1, $\mathcal{P}$ and $\mathcal{Y}$ have the same nondominated set. Therefore as in [14] we use the following equivalent objective space formulation

$$(\mathrm{P}_{\mathcal{P}}) \qquad \min T(y) = \prod_{i=1}^{p} y_i \text{ subject to } y \in \mathcal{P},$$

and we also have the following result according to [14].

THEOREM 2.2 *Problem* $(\mathrm{P}_{\mathcal{P}})$ *has a global optimal solution in* $\operatorname{vert}\mathcal{P}$.

Because of the properties in Theorem 2.1, e.g., $\mathcal{P}$ always contains an interior point and all the boundary points of $\mathcal{P}$ are weakly nondominated, it is easier to work with $\mathcal{P}$ instead of $\mathcal{Y}$. Thus Benson's outer approximation algorithm [16] and the dual variant of Benson's outer approximation algorithm [15] for solving MOLP (P) try to work with $\mathcal{P}$ to find $\mathcal{Y}_N$. Since our proposed primal and dual algorithms for linear MPPs $(\mathrm{P}_{\mathcal{X}})$ are based on Benson's algorithm and its dual variant, respectively, we work with $\mathcal{P}$ as well.

According to Theorem 2.3 we can easily obtain the optimal solution of $(\mathrm{P}_{\mathcal{X}})$ once we have the optimal solution of $(\mathrm{P}_{\mathcal{P}})$.

THEOREM 2.3 ([14]) *If* $x^* \in \mathcal{X}$ *is an efficient solution of problem MOLP (P),* $y^* = P(x^*) \in \mathcal{Y}$ *is its corresponding nondominated point, and* y^* *is also a global optimal solution of problem* $(\mathrm{P}_{\mathcal{P}})$*, then* x^* *is a global optimal solution of problem* $(\mathrm{P}_{\mathcal{X}})$*. Conversely, if* x^* *is a global optimal solution of problem* $(\mathrm{P}_{\mathcal{X}})$*, then* x^* *is also an efficient solution of problem MOLP (P), and* $y^* = P(x^*)$ *is a global optimal solution of problem* $(\mathrm{P}_{\mathcal{P}})$.

As can be seen from Theorems 2.1 and 2.2, $(\mathrm{P}_{\mathcal{P}})$ always has a global optimal solution at a vertex of $\mathcal{P}$ and any global optimal solution must belong to $\mathcal{Y}_N$. Therefore, multi-objective optimisation algorithms for finding $\mathcal{Y}_N$ can be used to solve multiplicative programming problems.

We use the trivial observation of Proposition 2.4 for determining lower bounds and upper bounds of ($P_{\mathcal{P}}$) in our proposed primal and dual algorithms.

Proposition 2.4 If $\mathcal{O} \supseteq \mathcal{P} \supseteq \mathcal{I}$, then the minimum value of $T(y)$ over $\mathcal{O}$ is a lower bound and the minimum value of $T(y)$ over $\mathcal{I}$ is an upper bound for T over $\mathcal{P}$.

The algorithms for MPPs developed in Section 3 explicitly or implicitly construct polyhedra $\mathcal{O}$ and $\mathcal{I}$ with $\mathcal{O} \supseteq \mathcal{P} \supseteq \mathcal{I}$. Thus the minimum values of $T(y)$ over $\mathcal{O}$ and $\mathcal{I}$ are always achieved at one of the vertices of $\mathcal{O}$ and $\mathcal{I}$ and are upper and lower bounds, respectively, for ($P_{\mathcal{P}}$).

3. Primal and dual objective space algorithms for linear MPPs

Geometric duality [17] defines a dual MOLP (D) for (P).

$$\begin{aligned}(\mathrm{D})\ \max\nolimits_{\mathcal{K}} D(u,\lambda) &= (\lambda_1, ..., \lambda_{p-1}, b^T u + d^T \lambda)^T \\ \text{s.t. } (u,\lambda) \in \mathcal{U} &= \left\{(u,\lambda) \in \mathbb{R}^m \times \mathbb{R}^p : (u,\lambda) \geqq 0, A^T u = C^T \lambda, e^T \lambda = 1\right\}.\end{aligned}$$

The primal problem (P) consists in finding the weakly nondominated points of $P(\mathcal{X})$, the dual problem consists in finding the $\mathcal{K}$-maximal elements of $D(\mathcal{U})$. Let $\mathcal{D} := D(\mathcal{U}) - \mathcal{K}$ be the extended polyhedral objective set of problem (D). Geometric duality provides a relationship between the facial structure of the extended primal and dual feasible sets, i.e., $\mathcal{P}$ and $\mathcal{D}$. Benson [16] proposed an outer approximation algorithm to solve MOLP (P) in objective space. Extensions of Benson's algorithm can be found in [15] and [18]. Based on geometric duality theory, Ehrgott et al. [15] developed a dual variant of Benson's outer approximation algorithm to solve the dual problem (D) in objective space.

In fact, both the primal and dual variant Benson algorithms obtains all the facets and extreme points of $\mathcal{P}$. Thus they can be directly used to solve linear MPPs by evaluating the minimum value of $T(y)$ over $\operatorname{vert}\mathcal{P}$ when the algorithms stop. However, to speed up the calculation and avoid the computation of all vertices of $\mathcal{P}$, we use Proposition 2.4 to compute upper and lower bounds for $T(y)$ and propose the following primal and dual objective space algorithms for linear MPPs. We show the advantages of our proposed algorithms in Section 4.

3.1. *Primal algorithm for solving linear MPPs*

In our previous work, we have extended Benson's outer approximation algorithm for MOLPs to solve convex multi-objective programming problems [13] and further the algorithm has been adapted to be an objective space cut and bound algorithm to solve convex multiplicative programmes [14].

We call the cut and bound algorithm "the primal algorithm". The algorithm constructs a sequence of polyhedra $\mathcal{S}^0 \supseteq \mathcal{S}^1 \supseteq \ldots \supseteq \mathcal{S}^k \supseteq \mathcal{P}$ approximating $\mathcal{P}$ from the outside. In iteration k we will evaluate $T(y)$ at all vertices of $\mathcal{S}^k$. The smallest of these values is a lower bound on the optimal value of $T(y)$ over $\mathcal{P}$. Moreover, at each iteration a boundary point $P(x^k) \in \mathcal{P}$ is computed which provides an upper bound $T(P(x^k))$ on the optimal value of ($P_{\mathcal{P}}$). Combining the lower bound and the upper bound information, the idea of our primal algorithm for MPPs can be described as follows.

Assume that individual minima of the functions $P_i(x) = c_i^T x + d_i$ over $\mathcal{X}$ are attained at x^{0i} for $i = 1, 2, \ldots, p$. Let $y^{0i} = P(x^{0i})$ and let $y^I = (P_1(x^{01}), P_2(x^{02}),$

$\ldots, P_p(x^{0p}))^T$ be the ideal point. Our primal algorithm starts with a polyhedron $\mathcal{S}^0 := y^I + \mathbb{R}^p_{\geqq} \supseteq \mathcal{P}$. The lower bound is initialised with $LB := T(y^I)$. Furthermore the upper bound UB is set to be the minimum of $\prod_{i=1}^p y^{0i}$, $i = 1, \ldots, p$. An approximation error $\epsilon \geqq 0$ needs to be specified. In each iteration, we choose a vertex $s \in \operatorname{vert} \mathcal{S}^k$ with the minimal value $T(s)$ as s^k, compute $(\bar{\mathrm{P}}_1(s^k))$ (see below) and get the optimal solution (x^k, z^k), with the boundary point $P(x^k)$ of $\mathcal{P}$. If $T(P(x^k)) < UB$, then we update UB with $T(P(x^k))$. If the relative difference between the upper bound and the lower bound is less than or equal to the approximation error ϵ, then the algorithm terminates. Otherwise, a hyperplane (cut) separating the vertex s^k from $\mathcal{P}$ is added to the description of $\mathcal{S}^k$. We update the vertex set $\operatorname{vert} \mathcal{S}^k$ and calculate $T(s)$ for each vertex $s \in \operatorname{vert} \mathcal{S}$ and find the minimal value of $T(s)$ among all the vertices of $\mathcal{S}^k$. If this is greater than the current lower bound, then we update the lower bound with $T(s)$. If the relative difference between the upper bound and the lower bound is less than or equal to the approximation error ϵ, then the algorithm terminates. Otherwise the procedure is repeated.

According to [18] and [19], the primal algorithm can be improved when solving linear MPP problems $(\mathrm{P}_{\mathcal{X}})$. Actually only one linear programme (LP) needs to be solved for finding the cut at each iteration step instead of two as the previous cut and bound algorithm indicates. The cut is based on the consideration of the following pair of dual linear programming problems and Theorem 3.1.

$$(\bar{\mathrm{P}}_1(y)) \qquad \min_{(x,z)\in\mathcal{S}(y)} z, \quad \mathcal{S}(y) := \{(x,z) \in \mathbb{R}^n \times \mathbb{R} : Ax \geqq b, Cx + d - ez \leqq y\},$$

$$(\bar{\mathrm{D}}_1(y)) \qquad \max_{(u,\lambda)\in\mathcal{U}} (b^T u - (y-d)^T \lambda), \quad \mathcal{U} = \left\{(u,\lambda) \in \mathbb{R}^m \times \mathbb{R}^p : (u,\lambda) \geqq 0, A^T u = C^T \lambda, e^T \lambda = 1\right\}.$$

THEOREM 3.1 ([13]) *Let $y^k \notin \mathcal{P}$. Let (x^k, z^k) be an optimal solution of linear programme $(\bar{\mathrm{P}}_1(y^k))$, and (u^*, λ^*) be the corresponding dual optimal solution of $(\bar{\mathrm{D}}_1(y))$. Let $\mathcal{H} = \{y \in \mathbb{R}^p : y^T \lambda^* = P(x^k)^T \lambda^*\}$. Then $\mathcal{H}$ is a supporting hyperplane of $\mathcal{P}$ at $P(x^k)$.*

Algorithm 3.2 summarises the steps of the algorithm.

Algorithm 3.2 (Primal algorithm for linear MPPs)

Initialisation.

(i1) Compute an optimal solution x^{0i} of $\min\{P_i(x) : x \in \mathcal{X}\}$ for $i = 1, \ldots, p$. Let $y^I = (P_1(x^{01}), \ldots, P_p(x^{0p}))^T$. Set $\mathcal{S}^0 := y^I + \mathbb{R}^p_{\geqq}$, $\operatorname{vert}\mathcal{S}^0 = \{y^I\}$.

(i2) Set $UB_0 = \min\{T(P(x^{0i})), i = 1, \ldots, p\}$ and $LB_0 := T(y^I)$.

(i3) Choose $\epsilon \geqq 0$ and set $k := 0$.

Iteration steps.

(k1) Choose $s \in \operatorname{vert}\mathcal{S}^k$ with the minimal value $T(s)$ as s^k. Solve LP $(\bar{\mathrm{P}}_1(s^k))$, let (x^k, z^k) be its optimal solution and (u^k, λ^k) be the corresponding dual optimal solution.

(k2) If $T(P(x^k)) < UB_k$, set $UB_k := T(P(x^k))$, $y^u := P(x^k)$, $x^u := x^k$.

(k3) If $UB_k \leqq LB_k(1+\epsilon)$, STOP. Otherwise, go to **(k4)**.

(k4) Set $\mathcal{S}^{k+1} := \mathcal{S}^k \cap \{y \in \mathbb{R}^p : y^T\lambda^k \geqq P(x^k)^T\lambda^k\}$.

(k5) Determine $\operatorname{vert}\mathcal{S}^{k+1}$ and set $LB_k := \min\{T(s) : s \in \operatorname{vert}\mathcal{S}^{k+1}\}$.
If $UB_k \leqq LB_k(1+\epsilon)$, STOP. Otherwise, set $k := k+1$, $UB_k := UB_{k-1}$, $LB_k := LB_{k-1}$ and go to **(k1)**.

Result.

(r1) If $\epsilon = 0$ ($\epsilon > 0$), then x^u is a global optimal (ϵ-optimal) solution of problem $(\mathrm{P}_{\mathcal{X}})$, and y^u is a global optimal (ϵ-optimal) solution of problem $(\mathrm{P}_{\mathcal{P}})$.

THEOREM 3.3 *Algorithm 3.2 terminates after a finite number of iterations. If $\epsilon = 0$ ($\epsilon > 0$), then we have an optimal (ϵ-optimal) solution of problem $(\mathrm{P}_{\mathcal{X}})$ at termination.*

Proof. The finiteness of Algorithm 3.2 is obvious because Benson's algorithm is finite (see, e.g., [15], [16]). When the algorithm terminates, we have $x^u = x^k$, where (x^k, z^k) is an optimal solution of the linear programme $(\bar{\mathrm{P}}_1(s^k))$. Then x^k is a weakly efficient solution of MOLP (P) and the corresponding weakly nondominated point is $P(x^k)$. Since $T(P(x^k))$ is an upper bound value and the relative difference between the upper bound and the lower bound is less than or equal to the approximation error ϵ, if $\epsilon = 0$ ($\epsilon > 0$), then $P(x^k)$ is also an optimal (ϵ-optimal) solution of $(\mathrm{P}_{\mathcal{P}})$. Thus, x^u is an optimal (ϵ-optimal) solution of $(\mathrm{P}_{\mathcal{X}})$. □

Remark 3.4 Algorithm 3.2 explicitly constructs a sequence of polyhedra $\mathcal{S}^k$ with $\mathcal{S}^1 \supseteq \ldots \supseteq \mathcal{S}^k \supseteq \mathcal{P}$ approximating $\mathcal{P}$ from the outside. Therefore according to Proposition 2.4, $LB_k = \min\{T(s) : s \in \operatorname{vert}\mathcal{S}^k\}$ is a sequence of increasing lower bounds on $(\mathrm{P}_{\mathcal{P}})$. Since we know $\operatorname{vert}\mathcal{S}^k$, LB_k is easy to compute.

Remark 3.5 Let $conv(P(x^k))$ be the convex hull of vertices $P(x^i), i = 0^1, \ldots, 0^p, 1, \ldots, k$. Let $\mathcal{I}^k = conv(P(x^k)) + \mathbb{R}^p_{\geqq}$. Algorithm 3.2 implicitly constructs a sequence of polyhedra $\mathcal{I}^k$ with $\mathcal{I}^1 \subseteq \ldots \subseteq \mathcal{I}^k \subseteq \mathcal{P}$ approximating $\mathcal{P}$ from the inside. Therefore according to Proposition 2.4, $UB_k = \min\{T(s) : s \in \operatorname{vert}\mathcal{I}^k\}$ is a sequence of decreasing upper bounds on $(\mathrm{P}_{\mathcal{P}})$. Since we know $\operatorname{vert}\mathcal{I}^k$, UB_k is easy to compute.

Remark 3.6 In Algorithm 3.2, the vertices of $\mathcal{S}^k$ are saved in vertex set array $\operatorname{vert}\mathcal{S}^k$ and they are updated at Step **(k5)** after the cut at each iteration. At Step **(k1)**, actually any vertex s of $\mathcal{S}^k$ with $T(s) < UB_k$ can be picked as s^k.

For example, one can pick the first vertex s with $T(s) < UB_k$ in the vertex set array vert $\mathcal{S}^k$ as s^k. However, in our algorithm we always use $s \in \text{vert}\,\mathcal{S}^k$ with the minimal value $T(s)$ as s^k. Thus, the vertex s with the smallest $T(s)$ among all the vertices of $\mathcal{S}$ is always being cut off.

Example 3.7 Consider the linear MPP $\min\left\{\prod_{i=1}^{p} c_i^T x : Ax \geqq b\right\}$, where

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, b = \begin{pmatrix} 8 \\ 6 \\ 8 \\ 1 \\ 1 \end{pmatrix}.$$

We solve it with the primal objective space algorithm. Figure 1 shows $\mathcal{P}$ and the changing of the outer approximation $\mathcal{S}$. In the figure, we use empty triangles to represent upper bound points and filled squares to represent lower bound points at each iteration. As can be seen from the figure, after three cuts, we have $\mathcal{S}^3 = \mathcal{P}$ and the two upper bound points (1,4) and (4,1)coincide with the two lower bound points. The total number of LPs solved is 5 (3 for the cuts and 2 for the ideal point).

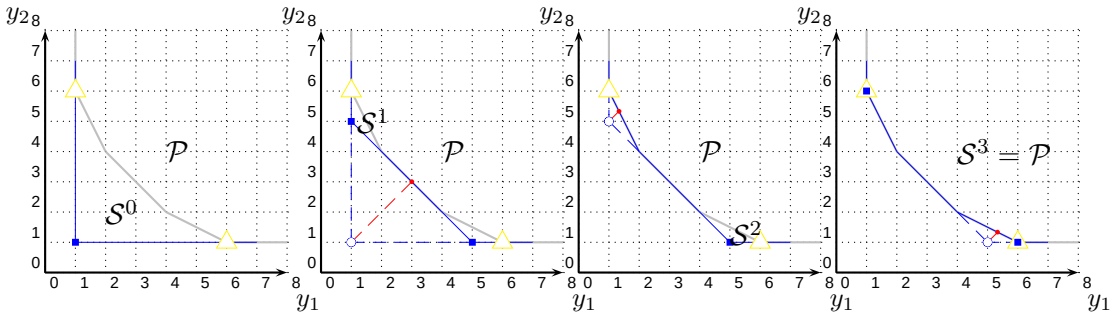

Figure 1. $\mathcal{P}$ and the shrinking of $\mathcal{S}^k$ with iteration k.

3.2. *Dual algorithm for solving linear MPPs*

The idea for solving linear MPPs is to use lower bounds and upper bounds to limit the optimal objective value of $(P_{\mathcal{P}})$. For that purpose outer and inner approximations of $\mathcal{P}$ need to be constructed. In the original dual Benson algorithm, a sequence of outer approximations of the extended feasible set in dual objective space is constructed. These outer approximations can be used to construct a sequence of inner approximations of $\mathcal{P}$ according to duality theory [20]. However, no outer approximations of $\mathcal{P}$ can be constructed from the original dual Benson algorithm. Therefore, we develop a sandwich version of the dual Benson algorithm, and then further adapt it to solve linear MPPs.

3.2.1. Sandwich version of the dual Benson's algorithm

Geometric duality theory [17] states that there is an inclusion reversing one-to-one map Ψ between the set of all proper $\mathcal{K}$-maximal faces of $\mathcal{D}$ and the set of all proper

$\mathbb{R}^p_{\leqq}$-minimal faces of $\mathcal{P}$. The map Ψ is based on the two set-valued maps,

$$\mathcal{H} : \mathbb{R}^p \rightrightarrows \mathbb{R}^p,\ \mathcal{H}(v) := \{y \in \mathbb{R}^p : \varphi(y, v) = 0\},$$
$$\mathcal{H}^* : \mathbb{R}^p \rightrightarrows \mathbb{R}^p,\ \mathcal{H}^*(y) := \{v \in \mathbb{R}^p : \varphi(y, v) = 0\}.$$

where

$$\varphi(y, v) = \sum_{i=1}^{p-1} y_i v_i + y_p \left(1 - \sum_{i=1}^{p-1} v_i\right) - v_p.$$

Based on geometric duality theory, Ehrgott et al. proposed a dual variant of Benson's outer approximation algorithm to solve MOLP (D) and obtain all the facets and extreme points of $\mathcal{P}$ and $\mathcal{D}$, respectively. For details of geometric duality theory and the original dual variant of Benson's algorithm, the reader is referred to [17] and [15], respectively. Here we propose a sandwich version of the dual variant of Benson's algorithm. Our algorithm not only gives a convex polyhedral outer approximation but also a convex polyhedral inner approximation of $\mathcal{D}$ during the iteration steps. By using the function $\varphi(y, v)$, a polyhedral outer approximation and a polyhedral inner approximation of $\mathcal{P}$ can then be easily obtained.

In the course of the algorithm, supporting hyperplanes of $\mathcal{D}$ are constructed. The following linear programming problem plays a key role in constructing hyperplanes.

$$(\bar{\mathrm{P}}_2(v)) \qquad \min_{x \in \mathcal{X}} \lambda(v)^T P(x), \quad \mathcal{X} := \{x \in \mathbb{R}^n : Ax \geqq b\},$$

where $\lambda(v) := \left(v_1, \ldots, v_{p-1}, 1 - \sum_{i=1}^{p-1} v_i\right)^T$.

Proposition 3.8 is critical for finding supporting hyperplanes and it is also the basis for improving the original dual variant of Benson's algorithm with only one LP to be solved at each iteration step.

Proposition 3.8 Let $\bar{v} \in \mathbb{R}^p$ satisfy $\lambda(\bar{v}) \geqq 0$. There exists a solution to $(\bar{\mathrm{P}}_2(\bar{v}))$ and for every solution $\bar{x}$ to $(\bar{\mathrm{P}}_2(\bar{v}))$, $\mathcal{H}^*(P(\bar{x}))$ is a supporting hyperplane of $\mathcal{D}$ containing $v^b = \left(\bar{v}^T \begin{bmatrix} I_{p-1} \\ 0 \end{bmatrix}, \lambda(\bar{v})^T P(\bar{x})\right)$ of $\mathcal{D}$.

Proof. If $\bar{v} \in \max_{\mathcal{K}} \mathcal{D}$, then the proof can be found in [19] and [15]. Since $(\bar{\mathrm{P}}_2(\bar{v}))$ and $(\bar{\mathrm{P}}_2(v^b))$ have the same optimal solutions, we just need to show $v^b \in \max_{\mathcal{K}} \mathcal{D}$. Suppose $(\bar{\mathrm{P}}_2(\bar{v}))$ has an optimal solution denoted by $\bar{x}$, and let $\bar{u}$ be the corresponding dual optimal solution. Strong duality implies $\lambda(\bar{v})^T C\bar{x} = b^T\bar{u}$. Thus, $v^b = \left(\bar{v}^T \begin{bmatrix} I_{p-1} \\ 0 \end{bmatrix}, \lambda(\bar{v})^T P(\bar{x})\right) = \left(\bar{v}^T \begin{bmatrix} I_{p-1} \\ 0 \end{bmatrix}, \lambda(\bar{v})^T (C\bar{x} + d)\right) = \left(\bar{v}^T \begin{bmatrix} I_{p-1} \\ 0 \end{bmatrix}, \lambda(\bar{v})^T d + b^T\bar{u}\right) \in \max_{\mathcal{K}} \mathcal{D}$ holds. □

Our sandwich version of the dual variant of Benson's algorithm is shown in Algorithm 3.9.

Algorithm 3.9 (Sandwich version of the dual variant of Benson's algorithm for MOLPs)

Initialisation.	Compute an optimal solution x^{0i} of $\min\{P_i(x) : x \in \mathcal{X}\}$ for $i = 1, \ldots, p$. Let $y^I = (P_1(x^{01}), \ldots, P_p(x^{0p}))^T$. Set $\mathcal{O}^0 = \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(P(x^{0i}), v) \geqq 0\}$ and compute vert $\mathcal{O}^0$. Set $\mathcal{I}^0 = \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(y^I, v) \geqq 0\}$ and compute vert $\mathcal{I}^0$. Set $k = 0$.
Iteration steps.	
(k1)	If, for each $v \in$ vert $\mathcal{O}^k$, $v \in \mathcal{I}^k$ is satisfied, then stop. Otherwise choose a vertex $v^k \in \mathcal{O}^k \backslash \mathcal{I}^k$.
(k2)	Compute an optimal solution x^k of $(\bar{\mathrm{P}}_2(v^k))$ and let $v^{kb} = \left({v^k}^T \begin{bmatrix} I_{p-1} \\ 0 \end{bmatrix}, \lambda(v^k)^T P(x^k) \right)$.
(k3)	Set vert $\mathcal{I}^{k+1} =$ vert $\mathcal{I}^k \bigcup \{v^{kb}\}$, update the inequality representation of $\mathcal{I}^{k+1}$. If $\lambda(v^k)^T P(x^k) < v_p^k$, set $\mathcal{O}^{k+1} := \mathcal{O}^k \cap \{v \in \mathbb{R}^p : \varphi(P(x^k), v) \geqq 0\}$ else set $\mathcal{O}^{k+1} := \mathcal{O}^k$. Update vert $\mathcal{O}^{k+1}$.
(k4)	Set $k := k + 1$ and go to (k1).

At each iteration k, the hyperplane given by $\mathcal{H}^*(P(x^k)) = \{v \in \mathbb{R}^p : \varphi(P(x^k), v) = 0\}$ is constructed so that it cuts off a portion of $\mathcal{O}^k$ containing v^k and at the same time, point v^{kb} is added to vert $\mathcal{I}^k$, thus $\mathcal{O}^0 \supseteq \mathcal{O}^1 \supseteq \mathcal{O}^2 \supseteq \ldots \supseteq \mathcal{O}^{k-2} \supseteq \mathcal{O}^{k-1} = \mathcal{D}$ and $\mathcal{I}^0 \subseteq \mathcal{I}^1 \subseteq \mathcal{I}^2 \subseteq \ldots \subseteq \mathcal{I}^{k-2} \subseteq \mathcal{I}^{k-1} = \mathcal{D}$.

When the algorithm terminates, we have the following results.

Proposition 3.10 ([15]) (1) The set of $\mathcal{K}$-maximal vertices of $\mathcal{D}$ is vert $\mathcal{O}^{k-1}$.

(2) The set $\left\{y \in \mathbb{R}^p : \varphi(y, v) \geqq 0 \text{ for all } v \in \text{vert } \mathcal{O}^{k-1}\right\}$ is a nondegenerate inequality representation of $\mathcal{P}$.

(3) All $\mathbb{R}^p_{\geqq}$-minimal (nondominated) vertices of $\mathcal{P}$ are contained in the set $\mathcal{W} := \{P(x^{01}), \ldots, P(x^{0p}), P(x^1), \ldots, P(x^{k-1})\}$.

(4) The set $\{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(y, v) \geqq 0 \text{ for all } y \in \mathcal{W}\}$ is a (possibly degenerate) inequality representation of $\mathcal{D}$.

Proposition 3.10 suggests that we obtain both $\mathcal{P}$ and $\mathcal{D}$ when the algorithm terminates. Moreover, at each iteration using the function φ, vert $\mathcal{O}$ and vert $\mathcal{I}$, an inner approximation and an outer approximation of $\mathcal{P}$ can be obtained. For that purpose, Property 3.11, Definition 3.12, Propositions 3.13 and 3.14 are needed.

Property 3.11 ([20]) *Let us consider special convex polyhedral sets $\mathcal{S} \subseteq \mathbb{R}^p$ with the property that $\mathcal{S} = \mathcal{S} - \mathcal{K}$ and the projection to its first $p-1$ components is the polytope $\{t \in \mathbb{R}^{p-1}, t \geqq 0, \sum_{i=1}^{p-1} t_i \leqq 1\}$.*

Definition 3.12 ([20]) *For a polyhedral convex set $\mathcal{S} \subseteq \mathbb{R}^p$ with Property 3.11, we define $\mathcal{D}(\mathcal{S}) = \{y \in \mathbb{R}^p : \varphi(y, v) \geqq 0,$ for all $v \in$ vert $\mathcal{S}\}$, where $\varphi(y, v) = \sum_{i=1}^{p-1} y_i v_i + y_p \left(1 - \sum_{i=1}^{p-1} v_i\right) - v_p$.*

Proposition 3.13 ([20]) Let $\mathcal{S} \subseteq \mathbb{R}^p$ with Property 3.11. Then $\mathcal{D}(\mathcal{S}) = \mathcal{D}(\mathcal{S}) + \mathbb{R}^p_{\geqq}$.

Proposition 3.14 ([20]) Let $\mathcal{S}^1$ and $\mathcal{S}^0$ be polyhedral convex sets with Property 3.11 and $\mathcal{S}^1 \subseteq \mathcal{S}^0$, then $\mathcal{D}(\mathcal{S}^1) \supseteq \mathcal{D}(\mathcal{S}^0)$.

For the dual variant of Benson's algorithm, Proposition 3.14 indicates that $\mathcal{D}(\mathcal{O}^k)$ enlarges and $\mathcal{D}(\mathcal{I}^k)$ shrinks with the iteration count. When the algorithm terminates at iteration k, $\mathcal{D}(\mathcal{O}^{k-1}) = \mathcal{D}(\mathcal{I}^{k-1}) = \mathcal{D}(\mathcal{D}) = \mathcal{P}$.

We give an example to illustrate the sandwich version of the dual variant of Benson's algorithm and we show how $\mathcal{O}^k$, $\mathcal{I}^k$, $\mathcal{D}(\mathcal{O}^k)$ and $\mathcal{D}(\mathcal{I}^k)$ change with iteration k.

Example 3.15 Consider the MOLP $\min\{Cx : Ax \geqq b\}$, where C, A, b are the same as in Example 3.7. $\mathcal{P}$ and $\mathcal{D}$ are shown in Figure 2. The 5 vertices of $\mathcal{D}$ are $(0,1)$, $(\frac{1}{3},\frac{8}{3})$, $(\frac{1}{2},3)$, $(\frac{2}{3},\frac{8}{3})$, $(1,1)$. Their corresponding facets (supporting hyperplanes) of $\mathcal{P}$ are $y_2 = 1$, $y_1 + 2y_2 = 8$, $y_1 + y_2 = 6$, $2y_1 + y_2 = 8$, $y_1 = 1$. The four vertices of $\mathcal{P}$, $(1,6)$, $(2,4)$, $(4,2)$ and $(6,1)$ correspond to the facets (supporting hyperplanes) of $\mathcal{D}$, $5v_1 + v_2 = 6$,$2v_1 + v_2 = 4$, $-2v_1 + v_2 = 1$ and $-5v_1 + v_2 = 1$, respectively.

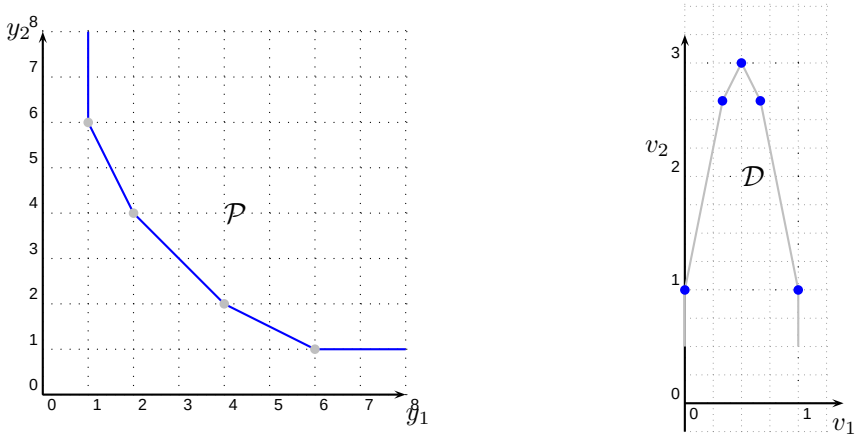

Figure 2. $\mathcal{P}$ and $\mathcal{D}$ for Example 3.15.

Figure 3 shows the change of $\mathcal{O}^k$ and $\mathcal{I}^k$ with each iteration k. As can be seen, with iteration k, $\mathcal{O}^k$ becomes smaller and smaller, $\mathcal{I}^k$ becomes bigger and bigger, until at termination both of them are the same as $\mathcal{D}$. The vertices of $\mathcal{O}^0$ are $(0,1)$, $(\frac{3}{2},\frac{7}{2})$ and $(1,1)$. The first hyperplane cuts off vertex $(\frac{3}{2},\frac{7}{2})$, the vertices of $\mathcal{O}^1$ are $(1,1)$, $(0,1)$, $(\frac{2}{5},3)$, and $(\frac{3}{5},3)$. The second hyperplane cuts off vertex $(\frac{3}{5},3)$, thus the vertices of $\mathcal{O}^2$ are $(0,1)$, $(\frac{2}{5},3)$,$(\frac{1}{2},3)$, $(\frac{2}{3},\frac{8}{3})$ and $(1,1)$. The third hyperplane cuts off vertex $(\frac{2}{5},3)$, thus the vertices of $\mathcal{O}^3$ are $(0,1)$, $(\frac{1}{3},\frac{8}{3})$, $(\frac{1}{2},3)$, $(\frac{2}{3},\frac{8}{3})$ and $(1,1)$. After the third cut, we have $\mathcal{O}^3 = \mathcal{D}$ and the vertices of $\mathcal{I}^3$ are $(0,1)$, $(\frac{2}{5},\frac{14}{5})$,$(\frac{1}{2},3)$,$(\frac{3}{5},\frac{14}{5})$ and $(1,1)$. At the fourth iteration, vertex $(\frac{2}{3},\frac{8}{3})$ is added to $\operatorname{vert}\mathcal{I}^4$ and at the fifth iteration vertex $(\frac{1}{3},\frac{8}{3})$ is added to $\operatorname{vert}\mathcal{I}^5$. Therefore, after five iterations, we have $\mathcal{O}^5 = \mathcal{I}^5 = \mathcal{D}$.

The change of $\mathcal{D}(\mathcal{O}^k)$ and $\mathcal{D}(\mathcal{I}^k)$ after each iteration k can be seen in Figure 4. The calculation of $\mathcal{D}(\mathcal{S}^k)$ ($\mathcal{S}^k = \mathcal{O}^k$ or $\mathcal{I}^k$) is according to the definition $\mathcal{D}(\mathcal{S}^k) = \{y \in \mathbb{R}^p : \varphi(y,v) \geqq 0 \text{ for all } v \in \operatorname{vert}\mathcal{S}^k\}$. For example, $\mathcal{D}(\mathcal{O}^0) = \{y \in \mathbb{R}^p : \varphi(y,v) \geqq 0 \text{ for } v = (0,1),(1,1)\}$, i.e., $\mathcal{D}(\mathcal{O}^0) = \{y_1 \geqq 1\} \bigcap \{y_2 \geqq 1\}$. In contrast to the shrinking of $\mathcal{O}^k$ and the enlargement of $\mathcal{I}^k$, $\mathcal{D}(\mathcal{O}^k)$ enlarges and $\mathcal{D}(\mathcal{I}^k)$ shrinks with iteration k. When the sandwich version of the dual Benson algorithm terminates, $\mathcal{O}^5 = \mathcal{I}^5 = \mathcal{D}$ and $\mathcal{D}(\mathcal{O}^5) = \mathcal{D}(\mathcal{I}^5) = \mathcal{D}(\mathcal{D}) = \mathcal{P}$.

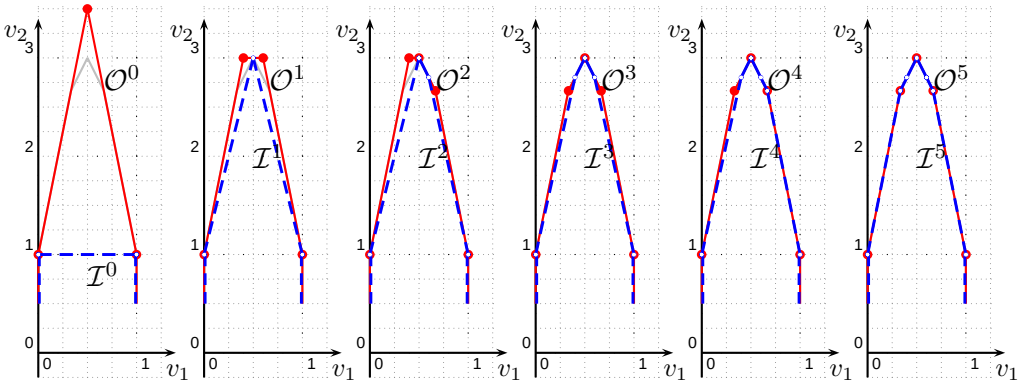

Figure 3. The shrinking of $\mathcal{O}^k$ and the enlargement of $\mathcal{I}^k$ with iteration k.

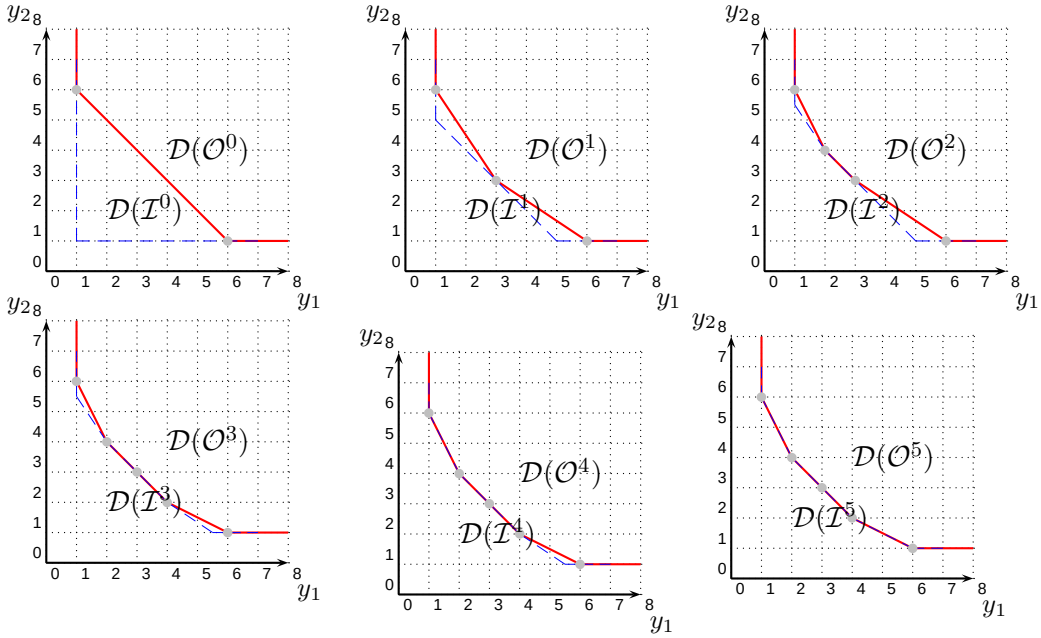

Figure 4. The enlargement of $\mathcal{D}(\mathcal{O}^k)$ and the shrinking of $\mathcal{D}(\mathcal{I}^k)$ with iteration k.

THEOREM 3.16 *The sandwich version of the dual variant of Benson's algorithm is finite.*

Proof. The point $v^{kb} \in \mathcal{D}$ computed in iteration k belongs to $\mathcal{O}^k$. If $v^{kb} \in \operatorname{int} \mathcal{O}^k$, then we have $\mathcal{O}^{k+1} := \mathcal{O}^k \cap \{v \in \mathbb{R}^p : \varphi(P(x^k), v) \geqq 0\}$ and, by Proposition 3.8, we know that $\mathcal{F} := \{v \in \mathcal{D} : \varphi(P(x^k), v) = 0\}$ is a face of $\mathcal{D}$ with $v^{kb} \in \mathcal{F}$, where $\mathcal{F} \subseteq \operatorname{bd} \mathcal{O}^k$. If $v^{kb} \in \operatorname{bd} \mathcal{O}^k$, then $v^{kb} \in \operatorname{bd} \mathcal{D}$. Since $\mathcal{D}$ is polyhedral, it has a finite number of faces, hence the algorithm is finite. □

3.2.2. Dual algorithm for linear MPPs

Now we explain how to adapt the sandwich version of the dual Benson algorithm to solve linear multiplicative programmes $(P_{\mathcal{X}})$. The idea of the dual algorithm for MPPs can be described as follows. The algorithm constructs a sequence of polyhedra $\mathcal{O}^k$ and $\mathcal{I}^k$ sandwiching $\mathcal{D}$. Therefore, $D(\mathcal{O}^k)$ and $D(\mathcal{I}^k)$ are the inner approximation and outer approximation of $\mathcal{P}$, respectively. In iteration k we evaluate $T(y)$ at all vertices of $D(\mathcal{O}^k)$ and $D(\mathcal{I}^k)$. The smallest of $D(\mathcal{O}^k)$ is the upper bound, and the smallest of $D(\mathcal{I}^k)$ is the lower bound. The algorithm improves the lower bound and upper bound by iteratively constructing a cutting plane that cuts off some vertex of $\mathcal{O}^k$ not in $\mathcal{D}$.

Algorithm 3.17 (Dual algorithm for linear MPPs)

Initialisation.

(i1) Compute an optimal solution x^{0i} of $\min\{P_i(x) : x \in \mathcal{X}\}$ for $i = 1, \ldots, p$. Let $y^I = (P_1(x^{01}), \ldots, P_p(x^{0p}))^T$.
Set $\mathcal{O}^0 = \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(P(x^{0i}), v) \geqq 0\}$ and compute vert $\mathcal{O}^0$.
Set $\mathcal{I}^0 = \{v \in \mathbb{R}^p : \lambda(v) \geqq 0, \varphi(y^I, v) \geqq 0\}$ and compute vert $\mathcal{I}^0$.

(i2) Let $x^u = \arg\min\{T(P(x^{0i})), i = 1, \ldots, p\}$. Set $UB_0 = T(P(x^u))$, $y^u = P(x^u)$. Set $LB_0 := T(y^I)$, $y^l = y^I$.

(i3) Choose $\epsilon \geqq 0$ and set $k := 0$.

Iteration steps.

(k1) Choose a vertex v^k of $\mathcal{O}^k$ with $v^k \notin \{v \in \mathbb{R}^p : \varphi(y^l, v) \geqq 0\}$. Solve $(\bar{\mathrm{P}}_2(v^k))$, let x^k be the optimal solution. Set vert $\mathcal{I}^{k+1}$ = vert $\mathcal{I}^k \bigcup \{(v_k^T \begin{bmatrix} I_{p-1} \\ 0 \end{bmatrix}, \lambda(v^k)^T P(x^k))\}$, update the inequality representation of $\mathcal{I}^{k+1}$.

(k2) If $T(P(x^k)) < UB_k$, set $UB_k := T(P(x^k))$, $y^u := P(x^k)$, $x^u := x^k$.

(k3) If $UB_k \leqq (1+\epsilon)LB_k$, STOP. Otherwise, go to **(k4)**.

(k4) Set $\mathcal{O}^{k+1} := \mathcal{O}^k \cap \{v \in \mathbb{R}^p : \varphi(P(x^k), v) \geqq 0\}$, update vert $\mathcal{O}^{k+1}$.

(k5) Compute vert $D(\mathcal{I}^{k+1})$. Let s^k be the optimal solution of $\min\{T(s) : s \in \text{vert}\, \mathcal{D}(\mathcal{I}^{k+1})\}$. Set $LB_k := T(s^k)$, $y^l = s^k$.
If $UB_k \leqq (1+\epsilon)LB_k$, STOP. Otherwise, set $k := k+1$, $UB_k = UB_{k-1}$, $LB_k = LB_{k-1}$ and go to **(k1)**.

Result.

(r1) If $\epsilon = 0$ ($\epsilon > 0$), then x^u is a global optimal (ϵ-optimal) solution of problem $(\mathrm{P}_\mathcal{X})$, and y^u is a global optimal (ϵ-optimal) solution of problem $(\mathrm{P}_\mathcal{P})$.

THEOREM 3.18 *Algorithm 3.17 terminates after a finite number of iterations. When $\epsilon = 0$ ($\epsilon > 0$), we have an optimal (ϵ-optimal) solution to problem $(\mathrm{P}_\mathcal{X})$ at termination.*

Proof. The algorithm is finite due to the finiteness of the sandwich version of the dual variant of Benson's algorithm. When the algorithm terminates, we have $x^u = x^k$, where x^k is an optimal solution of the linear programme $(\bar{\mathrm{P}}_2(v^k))$. x^k is a weakly efficient point of $\mathcal{P}$ and the corresponding point $P(x^k) \in \mathcal{P}_{WN}$. Since $T(P(x^k))$ is an upper bound value and the relative difference between the upper bound and the lower bound is less than or equal to the approximation error ϵ, if $\epsilon = 0$ ($\epsilon > 0$), then $T(P(x^k))$ is an optimal (ϵ-optimal) solution of $(\mathrm{P}_\mathcal{P})$. Thus, x^u is an optimal (ϵ-optimal) solution of $(\mathrm{P}_\mathcal{X})$. □

Remark 3.19 Algorithm 3.17 constructs a sequence of polyhedra $\mathcal{I}^k$ with $\mathcal{I}^0 \supseteq \ldots \mathcal{I}^k \supseteq \mathcal{D}$ approximating $\mathcal{D}$ from the inside. Furthermore it explicitly constructs $D(\mathcal{I}^k)$ with $D(\mathcal{I}^0) \supseteq \ldots \supseteq D(\mathcal{I}^k) \supseteq \mathcal{P}$ approximating $\mathcal{P}$ from the outside. Therefore according to Proposition 2.4, $LB_k = \min\{T(s) : s \in \text{vert}\, D(\mathcal{I}^k)\}$ is a sequence of increasing lower bounds on $(\mathrm{P}_\mathcal{P})$. Since the vertices of $\mathcal{I}^k$ are known, it is easy to obtain the facets of $D(\mathcal{I}^k)$ using the function φ. However to calculate LB_k, extra vertex enumeration is needed to find vert $D(\mathcal{I}^k)$.

Table 1. vert $\mathcal{O}^k$, vert $\mathcal{I}^k$, vert $D(\mathcal{O}^k)$, vert $D(\mathcal{I}^k)$, LB_k, UB_k.

k	Set	Vertices	Set	Vertices	LB_k	UB_k
0	vert $\mathcal{O}^k$	(0,1),(1,1),($\frac{1}{2}$,$\frac{7}{2}$)	vert $\mathcal{I}^k$	(0,1),(1,1)	1	6
	vert $\mathcal{D}(\mathcal{O}^k)$	$(1,6),(6,1)$	vert $\mathcal{D}(\mathcal{I}^k)$	$(1,1)$		
1	vert $\mathcal{O}^k$	(0,1),(1,1),($\frac{2}{5}$,3),($\frac{3}{5}$,3)	vert $\mathcal{I}^k$	(0,1),(1,1),($\frac{1}{2}$,3)	5	6
	vert $\mathcal{D}(\mathcal{O}^k)$	$(1,6),(3,3),(6,1)$	vert $\mathcal{D}(\mathcal{I}^k)$	$(1,5),(5,1)$		
2	vert $\mathcal{O}^k$	(0,1),(1,1)($\frac{2}{5}$,3)($\frac{1}{2}$,3),($\frac{2}{3}$,$\frac{8}{3}$)	vert $\mathcal{I}^k$	(0,1),($\frac{1}{2}$,3),($\frac{3}{5}$,$\frac{14}{5}$),(1,1)	5	6
	vert $\mathcal{D}(\mathcal{O}^k)$	(1,6),(2,4),(3,3),(6,1)	vert $\mathcal{D}(\mathcal{I}^k)$	(1,$\frac{11}{2}$),(2,4),(5,1)		
3	vert $\mathcal{O}^k$	(0,1)($\frac{1}{3}$,$\frac{8}{3}$),($\frac{1}{2}$,3),($\frac{2}{3}$,$\frac{8}{3}$),(1,1)	vert $\mathcal{I}^k$	(0,1),($\frac{2}{5}$,$\frac{14}{5}$),($\frac{1}{2}$,3),($\frac{3}{5}$,$\frac{14}{5}$),(1,1)	$\frac{11}{2}$	6
	vert $\mathcal{D}(\mathcal{O}^k)$	(1,6),(2,4),(4,2),(6,1)	vert $\mathcal{D}(\mathcal{I}^k)$	(1,$\frac{11}{2}$),(2,4),(4,2),($\frac{11}{2}$,1)		
4	vert $\mathcal{O}^k$	(0,1)($\frac{1}{3}$,$\frac{8}{3}$),($\frac{1}{2}$,3),($\frac{2}{3}$,$\frac{8}{3}$),(1,1)	vert $\mathcal{I}^k$	(0,1),($\frac{2}{5}$,$\frac{14}{5}$),($\frac{1}{2}$,3),($\frac{2}{3}$,$\frac{8}{3}$),(1,1)	$\frac{11}{2}$	6
	vert $\mathcal{D}(\mathcal{O}^k)$	(1,6),(2,4),,(4,2),(6,1)	vert $\mathcal{D}(\mathcal{I}^k)$	(1,6),(2,4),(4,2),($\frac{11}{2}$,1)		
5	vert $\mathcal{O}^k$	(0,1)($\frac{1}{3}$,$\frac{8}{3}$),($\frac{1}{2}$,3),($\frac{2}{3}$,$\frac{8}{3}$),(1,1)	vert $\mathcal{I}^k$	(0,1),($\frac{1}{3}$,$\frac{8}{3}$),($\frac{1}{2}$,3),($\frac{2}{3}$,$\frac{8}{3}$),(1,1)	6	6
	vert $\mathcal{D}(\mathcal{O}^k)$	(1,6),(2,4),(4,2),(6,1)	vert $\mathcal{D}(\mathcal{I}^k)$	(1,6),(2,4),(4,2),(6,1)		

Remark 3.20 Let $conv(P(x^k))$ be the convex hull of vertices $P(x^i), i = 0^1, \ldots, 0^p, 1, \ldots, k$. Let $\mathcal{D}(\mathcal{O}^k) = conv(P(x^k)) + \mathbb{R}^p_{\leqq}$. Algorithm 3.17 constructs a sequence of polyhedra $\mathcal{O}^k$ with $\mathcal{O}^0 \supseteq \ldots \mathcal{O}^k \supseteq \mathcal{D}$ approximating $\mathcal{D}$ from the outside. It also implicitly constructs a sequence of polyhedra $\mathcal{D}(\mathcal{O}^k)$ with $\mathcal{D}(\mathcal{O}^0) \subseteq \ldots \subseteq \mathcal{D}(\mathcal{O}^k) \subseteq \mathcal{P}$ approximating $\mathcal{P}$ from the inside. Therefore according to Proposition 2.4, $UB_k = \min\{T(s) : s \in \text{vert}\,\mathcal{D}(\mathcal{O}^k)\}$ is a sequence of decreasing upper bounds on ($\mathrm{P}_{\mathcal{P}}$). Since we know vert $\mathcal{D}(\mathcal{O}^k)$, UB_k is easy to compute.

Remark 3.21 In Algorithm 3.17, the vertices of $\mathcal{O}^k$ are saved in vertex set array vert $\mathcal{O}^k$ and they are updated at Step **(k4)** after the cut at each iteration. At Step **(k1)**, actually any vertex v of $\mathcal{O}^k$ with $v \notin \mathcal{I}^k$ can be picked as v^k. For example, one can pick the first vertex v in the vertex set array vert $\mathcal{O}^k$ with $v \notin \mathcal{I}^k$. However, in our algorithm, we always pick a vertex v^k of $\mathcal{O}^k$ with $v^k \notin \{v \in \mathbb{R}^p : \varphi(y^l, v) \geqq 0\}$. Since y^l is one of the vertices of $\mathcal{D}(\mathcal{I}^k)$ with the smallest $T(y)$, we actually try to improve the lower bound at every iteration.

Next, we use Example 3.7 to illustrate Algorithm 3.17. The initialisation and the iteration steps are the same as in Example 3.15. We list the vertex set of $\mathcal{O}^k$, $\mathcal{I}^k$, $D(\mathcal{O}^k)$, $D(\mathcal{I}^k)$ and the lower bound and upper bound at each iteration in Table 1. The total number of LPs solved is 7.

4. Numerical results

In Section 1 we have briefly outlined the literature on algorithms to solve multiplicative programming problems. In this section we compare our primal and dual algorithms presented in Section 3 with our original primal algorithm [14], a recent algorithm for linear MPP, i.e., Gao et al. [8], and two general purpose global optimisation solvers SCIP [21] and BARON [22, 23]. Of course, as we explained on page 5 both the primal and dual variant Benson algorithms for MOLPs can be used to solve linear MPPs directly. We carried out preliminary tests comparing Algorithms 3.2 and 3.17 to the application of the primal and dual variants of Benson's algorithms to solve linear MPPs. Since in all but the smallest instances, Algorithms 3.2 and 3.17 were much faster, we do not report results on the primal and dual variants of Benson's algorithm.

We have implemented the first four algorithms in Matlab 7.3 using MOSEK (`http://www.mosek.com`) as linear programming and nonlinear programming solver. Tests are run on a PC with 2.5GHz and 4.0 GB RAM and the codes including Matlab and MOSEK are run in a single thread mode. At Step **(k5)** of

Table 2. Results for (T), "-" means the problems cannot be solved within 20 minutes.

p	(m,n)	Primal	Dual	Original Primal	Gao et al.	SCIP	BARON
2	(20,30)	0.10	0.11	0.15	0.13	0.18	0.07
2	(50,30)	0.14	0.11	0.19	0.15	0.26	0.07
2	(100,60)	0.20	0.15	0.34	0.20	0.49	0.16
3	(50,30)	0.34	0.29	0.52	0.42	0.54	0.63
3	(60,40)	0.35	0.32	0.58	0.53	0.57	4.89
3	(100,60)	0.68	0.58	0.96	1.25	1.10	17.69
4	(60,40)	2.09	2.26	3.54	5.37	34.00	-
4	(100,60)	7.98	7.94	13.06	14.45	-	-
5	(100,60)	24.17	29.38	49.65	67.98	-	-
6	(100,60)	243.34	259.46	391.31	595.45	-	-

the primal algorithm and Step **(k4)** of the dual algorithm, the method of [24] for on-line vertex enumeration by adjacency lists was used to calculate a vertex representation from the inequality representation of $\mathcal{S}^{k+1}$ or $\mathcal{O}^{k+1}$. For all four algorithms, we use the same approximation error to make the results comparable. Moreover, we solve the test instances with SCIP and BARON. For SCIP, we have downloaded the binaries of SCIP-3.1.0 from http://scip.zib.de/, and run the tests on the same PC as the first four algorithms while for BARON the tests are run on NEOS Server (http://www.neos-server.org/neos).

We apply the four algorithms and SCIP and BARON solvers to randomly generated examples of multiplicative programming problems. The following subclass of $(\mathrm{P}_{\mathcal{X}})$ is considered.

$$\begin{aligned} &\min \prod_{i=1}^{p} {c^i}^T x \\ &\text{s.t. } Ax \geqq b, u \geqq x \geqq l, \end{aligned}$$

where $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ and $c^i \in \mathbb{R}^n$ are constant matrices with entries pseudo-randomly generated in the interval [0,10]. Moreover, we set $0 = l \in \mathbb{R}^n$ and $(100, \ldots, 100) = u \in \mathbb{R}^n$ to make the instances bounded.

Ten examples for selected combinations of m (number of constraints), n (number of variables) and p (number of objectives) were solved. For the first four algorithms, the approximation error was fixed at $\epsilon = 0.01$. Average values of the CPU time in seconds are listed in Table 2.

It can be seen from Table 2, that our primal and dual algorithms behave similarly. Our primal algorithm is always superior to the original primal algorithm because it solves only one LP at each iteration instead of two using the idea of [18]. When p is equal to two, the difference between our proposed primal and dual algorithms and Gao et al.'s algorithm is quite small and all three algorithms behave well. Because Gao et al.'s algorithm [8] needs to solve a nonlinear programming problem instead of a linear programming problem at each iteration, it requires more computation time compared to our algorithms. Therefore, if p is greater than or equal to three, our algorithms clearly outperform Gao et al.'s algorithm [8].

Compared to SCIP and BARON, when p is equal to two, BARON is faster than our algorithms. However as p increases, the computational performance of both SCIP and BARON becomes worse. Especially when p is equal to 4, BARON cannot solve any of the problems within 20 minutes. For $m = 60$ and $n = 40$ SCIP needs 34 seconds while our algorithms only use less than 3 seconds; for $m = 100$ and $n = 60$ SCIP cannot solve any of the problems within 20 minutes while our algorithms only use less than 8 seconds. For p equal to 5 and 6, both SCIP and BARON cannot solve any of the problems within 20 minutes, while our proposed

primal and dual algorithms can solve each of the problems within 5 minutes.

Table 2 suggests that the most critical factor influencing computation time is the number p of functions that make up the product $\phi(x)$. As p increases the computation time increases and more dramatically so for higher values of p than smaller ones. For example, for the problems with $(m, n) = (100, 60)$ shown in Table 2, as p increases from 5 to 6, the computation times for our primal and dual algorithms increase tenfold.

5. Conclusion

In this paper, we have proposed primal and dual objective space algorithms derived from objective space algorithms for multi-objective linear programming to solve linear multiplicative programming problems. The numerical results suggest that our algorithms are superior to recent global optimisation algorithms for linear MPPs as well as general global optimisation solvers. Therefore we have demonstrated the viability of a multi-objective linear programming approach to linear MPPs compared to single objective nonconvex global optimisation approaches. This may inspire researchers to investigate the use of multi-objective methods for other hard single objective optimisation methods. Moreover, our algorithms are easy to implement. Further research is needed to investigate the trade-off between solution quality and computational effort, in particular concerning the growth of computation time as p increases.

References

[1] Matsui T. NP-hardness of linear multiplicative programming and related problems. Journal of Global Optimization. 1996;9:113–119.
[2] Henderson JM, Quandt RE. Microeconomic Theory. McGraw-Hill, New York; 1971.
[3] Konno H, Kuno T. Generalized linear multiplicative and fractional programming. Annals of Operations Research. 1990;25:147–162.
[4] Maling K, Mueller SH, Heller WR. On finding most optimal optional rectangular package plans. In: Proceedings of the 19th Design Automation Conference; 1982. p. 663–670.
[5] Geoffrion A. Solving bicriterion mathematical programs. Operations Research. 1967; 15:39–54.
[6] Benson H, Boger GM. Outcome-space cutting-plane algorithm for linear multiplicative programming. Journal of Optimization Theory and Applications. 2000;104:301–322.
[7] Kuno T. A finite branch-and-bound algorithm for linear multiplicative programming. Computational Optimization and Applications. 2001;20:119–135.
[8] Gao Y, Xu C, Yang Y. An outcome-space finite algorithm for solving linear multiplicative programming. Applied Mathematics and Computation. 2006;179:494–505.
[9] Ryoo HS, Sahindis NV. Global optimization of multiplicative programs. Journal of Global Optimization. 2003;26:387–418.
[10] Kim NTB, Trang NTL, Yen TTH. Outcome-space outer approximation algorithm for linear multiplicative programming. East West Journal of Mathematics. 2007;9:81–98.
[11] Depetrini D, Locatelli M. A FPTAS for a class of linear multiplicative problems. Computational Optimization and Applications. 2009;44:275–288.
[12] Benson H, Boger GM. Multiplicative programming problems: analysis and efficient point search heuristic. Journal of Optimization Theory and Applications. 1997;94:487–510.
[13] Ehrgott M, Shao L, Schöbel A. An approximation algorithm for convex multi-objective programming problems. Journal of Global Optimization. 2011;50:397–416.

[14] Shao L, Ehrgott M. An objective space cut and bound algorithm for convex multiplicative programmes. Journal of Global Optimization. 2014;58:711–728; dOI:10.1007/s10898-013-0102-x.
[15] Ehrgott M, Löhne A, Shao L. A dual variant of Benson's "outer approximation algorithm" for multiple objective linear programming. Journal of Global Optimization. 2012;52:757–778.
[16] Benson H. An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem. Journal of Global Optimization. 1998;13:1–24.
[17] Heyde F, Löhne A. Geometric duality in multi-objective linear programming. SIAM J Optim. 2008;19(2):836–845.
[18] Hamel AH, Löhne A, Rudloff B. Benson type algorithms for linear vector optimization and applications. Journal of Global Optimization. 2014;59:811–836.
[19] Löhne A. Vector optimization with infimum and supremum. Springer; 2011.
[20] Shao L, Ehrgott M. Approximating the nondominated set of an MOLP by approximately solving its dual problem. Mathematical Methods of Operations Research. 2008; 68:469–492.
[21] Achterberg T. SCIP: Solving constraint integer programs. Mathematical Programming Computation. 2009;1(1):1–41; http://mpc.zib.de/index.php/MPC/article/view/4.
[22] Sahinidis NV. BARON 12.1.0: Global Optimization of Mixed-Integer Nonlinear Programs, *User's Manual*. 2013.
[23] Tawarmalani M, Sahinidis NV. A polyhedral branch-and-cut approach to global optimization. Mathematical Programming. 2005;103:225–249.
[24] Chen PC, Hansen P, Jaumard B. On-line and off-line vertex enumeration by adjacency lists. Operations Research Letters. 1991;10:403–409.