European Journal of Operational Research 277 (2019) 653–666

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Reducing wall-clock time for the computation of all efficient extreme points in multiple objective linear programming

Craig A. Piercy, Ralph E. Steuer*

Terry College of Business, University of Georgia, Athens, GA 30602, USA

ARTICLE INFO

Article history:
Received 19 May 2018
Accepted 20 February 2019
Available online 25 February 2019

Keywords:
Multiple objective programming
Efficient extreme points
Nondominated vertices
Distributed processing
Criterion cones

ABSTRACT

This paper describes an approach for markedly reducing the time required to obtain all efficient extreme points of a multiple objective linear program (MOLP) with three objectives. The approach is particularly useful when working with such MOLPs possessing large numbers of efficient extreme points. By subdividing the criterion cone into sub-cones, the paper shows how the task of computing all efficient extreme points can be broken down into parts so that the parts can be solved concurrently, thus allowing all efficient extreme points to be computed in much reduced elapsed time. The paper investigates several schemes for conducting this task and reports on a volume of computational experience.

© 2019 Published by Elsevier B.V.

1. Introduction

In this paper, we describe a solution-time reduction idea for use with multiple objective linear programming problems (MOLPs). The approach is designed to be useful for analysts and decision makers simply wishing to save time or who may be working under a time-budget, but wish to be thorough in exploring candidate solutions in large problems. While the approach described here can be applied to MOLPs with more objectives, we have restricted ourselves to the study of MOLPs with three objectives to fully develop the idea and explore its issues at this level. Thus, the material here should provide a strong foundation for extending the study to MOLPs with greater numbers of objectives.

Just as linear programming plays a key role in single criterion optimization, multiple objective linear programming plays a similar role in multiple criteria optimization, an on-going area in multiple criteria decision making (Wallenius et al., 2008). In addition to providing foundational support to multiple criteria optimization (as done in Antunes, Alves, & Climaco, 2016), multiple objective linear programming has a history of being applicable to complex problems in a range of areas from water resources to manpower planning to energy planning and even radiotherapy (Ehrgott & Shao, 2008) when all relationships are linear.

Many methods have been proposed for solving multiple criteria problems. According to Hwang and Masud (1979), the methods can be grouped into three categories based upon when preference information is elicited from the decision maker. These categories are (1) *a priori*, (2) progressive articulation of preferences, and (3) *a posteriori*. In an *a priori* method, preference information is elicited at the very beginning and then an optimization problem is formed with the aim of solving it directly for a "final" solution. A final solution is either an optimal solution or a solution close enough to being optimal to terminate the decision process. In a progressive articulation of preferences method, phases of computation are interleaved with phases of input from the decision maker in a process of sampling from the set of all contenders for optimality (defined shortly). In such methods, a search for a final solution is carried out in a progressively more concentrated fashion. Methods in this category are also known as interactive procedures. A liability of methods from categories (1) and (2) is the fear that important areas of potentially optimal solutions will get missed using them.

In an *a posteriori* method, or sometimes called a *generating* method, a highly representative set of the set of all candidates for optimality (or in the best case, the complete set of such points) is first computed. Once obtained, some strategy for searching through the set is applied until a final solution is obtained. The appeal of this category of methods is that no regions of the set of all candidates for optimality will be missed. However, this thoroughness often comes at a high cost. In other than small problems, it is typically not possible to compute the highly representative sets desired within a reasonable amount of time. The method described in this paper is proposed with the goal of reducing this time obstacle in multiple objective linear programming.

Thus, this paper fits into the literature within the *a posteriori* category as a new way to go about generating highly representative sets of the candidate sets mentioned above for problem

* Corresponding author.
E-mail address: rsteuer@uga.edu (R.E. Steuer).

https://doi.org/10.1016/j.ejor.2019.02.042
0377-2217/© 2019 Published by Elsevier B.V.

sizes that up until now have been out of reach in multiple objective linear programming. One possible downside to the new approach is that, in the short run, the sets generated may be too highly representative. That is, they may result in sets that may inundate users with more points than they may feel they currently have a use for. Nevertheless, being the results of basic research, users will probably soon find ways to use the information given that such sets can now be generated in much reduced time. In fact, one immediate use of the volume of information generated could be in a NAUTILUS-type method such as described in Miettinen, Eskelainen, Ruiz, and Luque (2010), Miettinen, Podkopaev, Ruiz, and Luque (2015) when the problem to be solved is a multiple objective linear programming problem. In this way, a time-budget can be met, or preparation time can at least be conserved, while retaining the benefits of thoroughness provided by an *a posteriori* method.

To get started on the paper, we now define an MOLP in general. In roster-form, an MOLP can be written as

$$\begin{aligned} &\max\{\mathbf{c}^1\mathbf{x} = z_1\} \\ &\quad\vdots \\ &\max\{\mathbf{c}^k\mathbf{x} = z_k\} \\ &\text{s.t.}\ \ \mathbf{x} \in S = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{b} \in \mathbb{R}^m, \mathbf{x} \geq \mathbf{0}\} \end{aligned} \tag{1}$$

or, alternately, in vector-maximum form, as

$$\max\{\mathbf{C}\mathbf{x} = \mathbf{z} \mid \mathbf{x} \in S\}$$

where $\mathbf{C}$ is a $k \times n$ matrix whose rows are the $\mathbf{c}^i$ gradients of the k objectives, $\mathbf{z} \in \mathbb{R}^k$ is a criterion vector, and $S \subset \mathbb{R}^n$ is the feasible region in decision space. In the three-objective version of (1) that we address for application of the solution time reduction idea of this paper, k of course becomes 3, and we further assume that S is bounded and that all three of the resultant MOLP's $\mathbf{c}^i$ gradients of the objectives are linearly independent. We will sometimes refer to the three-objective resultant MOLP as the "original" MOLP because it is the MOLP from which we start in this paper.

Let $Z \subset \mathbb{R}^k$ be the feasible region in criterion space (sometimes called outcome space) where $\mathbf{z} \in Z$ iff there exists an $\mathbf{x} \in S$ such that $\mathbf{z} = \mathbf{C}\mathbf{x}$. In criterion space, a $\bar{\mathbf{z}} \in Z$ is *nondominated* iff there does not exist a $\mathbf{z} \in Z$ such that $\bar{\mathbf{z}} \leq \mathbf{z}$ and $\bar{\mathbf{z}} \neq \mathbf{z}$. In decision space, an $\bar{\mathbf{x}} \in S$ is *efficient* iff $\mathbf{z} = \mathbf{C}\bar{\mathbf{x}}$ is nondominated. We are interested in efficient points because they are *contenders for optimality* in multiple objective programming whereas *inefficient* points are not.

The solution of an MOLP is often characterized by the enumeration of all efficient extreme points of S and many algorithms have been proposed for this purpose. Algorithms for computing all efficient extreme points can be viewed as *decision space* procedures as they pivot among extreme points of S until all efficient extreme points of S are found. Algorithms in this category include those by Evans and Steuer (1973), Zeleny (1974), Isermann (1977), Ecker and Kouada (1978), and Armand and Malivert (1991).

In contrast to the enumeration of all efficient extreme points, the solution of an MOLP is sometimes characterized by the enumeration of all nondominated vertices of Z. Algorithms in this class are viewed as *outcome space* procedures as they mostly strive to pivot among vertices of Z until all nondominated vertices of Z are found. Algorithms that can compute all nondominated vertices include those by Benson and Sun (2000), Benson and Sun (2002), Shao and Ehrgott (2008), Ehrgott, Löhne, and Shao (2012), Rudloff, Ulus, and Vanderbei (2017), and Löhne and Weissing (2017).

The only time there is a difference between the output of the two categories of algorithms is when *collapsing* occurs in which case the output of an algorithm that computes all efficient extreme points subsumes that of an algorithm that only targets the computation of all nondominated vertices. Collapsing, as studied by Dauer (1987, 1993), occurs when two or more efficient extreme points map into the same nondominated vertex of Z or when there exists an efficient extreme point that maps into a criterion vector that is not a vertex of Z. Unfortunately, little is known about the frequency of collapsing and even less is known about how to detect from the outside whether a given MOLP possesses collapsing on the inside. As computing all efficient extreme points is a more general task than computing all nondominated vertices[1], the research of this paper is carried out using an algorithm that computes all efficient extreme points. However, we are inclined to believe that the results of this paper may have applicability to the other category of algorithms, but that is not a part of this research.

Needless to say, the drawback of any algorithm for computing all efficient extreme points is the time to compute them. Thus, the goal of the paper is to alleviate this issue.

The approach described herein involves working with *criterion cones* defined as follows.

Definition 1. Consider any MOLP. The criterion of the MOLP is the convex cone generated by the gradients of the MOLP's objective functions.

That is, the criterion cone of an MOLP is the set of all nonnegative linear combinations of the MOLP's objective function gradients. In general, the smaller the criterion cone, the smaller the number of extreme points efficient with respect to it (Steuer, 1986). With this in mind, we pursue a strategy for computing all efficient extreme points that has three phases. First, we subdivide the criterion cone of the original MOLP into sub-cones. Second, we solve MOLPs associated with the different sub-cones for all of their respective efficient extreme points. Third, we take the union of all of the efficient extreme points computed to obtain the set of all efficient extreme points of the original MOLP.

Because of the way the sub-cones are created, no sub-cone MOLP is dependent upon the output of any other sub-cone MOLP. Thus they are all independent of one another in the sense that all sub-cone MOLPs can be solved concurrently. This allows wall-clock time for computing all efficient extreme points of an MOLP to be markedly reduced.

At this point, it might normally be appropriate to mention more about other research along the same lines as in this paper and compare the strategy and results of this paper with other existing methods. However, this is difficult because we are unaware of any work by other researchers along the lines of this paper regarding the task of computing all efficient extreme points of an MOLP. A possible reason for the absence of such research is that much of multiple objective linear programming is a straightforward extension of single-objective linear programming: the simplex method becomes a multiple objective simplex method, the reduced cost row becomes a reduced cost matrix, alternative optima become efficient points, and so forth. But nowhere in single-objective linear programming is the objective function structure subdivided into parts to save time for solving the whole problem, so the idea is not one that obviously crops up when transiting to multiple objective linear programming.

As for the papers referenced earlier, despite some of their titles, their focus is on *single-run* algorithms, that is, once they start they don't stop until all efficient extreme points (or nondominated vertices) are enumerated. This is in contrast to this paper which breaks the objective function structure of an MOLP into parts so

[1] To reinforce this point, suppose we are faced with a problem of optimizing a linear function over the efficient set of an MOLP such as studied by Benson (1991); Ecker and Song (1994), and Yamamoto (2002). In this case, an algorithm for computing all efficient extreme points would be more relevant to the problem than an algorithm that is only able to compute all nondominated vertices.

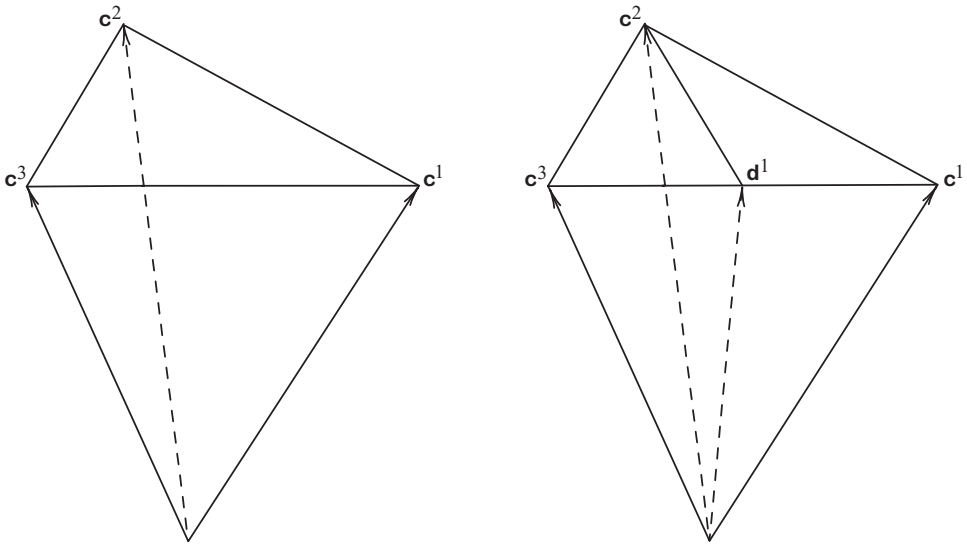

Fig. 1. (*Left*): Criterion cone of a three-objective MOLP. (*Right*): Decomposition of the criterion cone into two subset-cones and a cut-cone.

as to allow the parts to be solved concurrently, which is a lot different.

The paper is organized as follows. In Section 2 we illustrate the subdivision of a criterion cone into sub-cones and suggest different schemes for doing this. In Section 3, we outline the operation of the code used for computing all of the efficient extreme points computed in this paper. In Section 4, we discuss how it is inevitable that some efficient extreme points will be computed more than once when subdividing an MOLP. In Section 5, we describe the random problem generator used to generate the MOLPs of our experiments. In Section 6, we discuss a given scheme/problem-size experiment and the nature of the trials within it. In Section 7, we cross-compare the results of the different scheme experiments conducted, and in Section 8, we conclude the paper with remarks about future directions.

2. Subdividing the criterion cone

Consider an instance of the three-objective MOLPs of interest in this paper and let its criterion cone (the cone generated by its $\mathbf{c}^i$ gradients) be portrayed as in Fig. 1 *Left*. Note that the cone is both polyhedral and of 3 dimensions as a consequence of the MOLP's three $\mathbf{c}^i$ being linearly independent. Also we note that the triangle connecting the head points of the three $\mathbf{c}^i$ gradients is the cross section of the criterion cone upon which lie the head points of all of what we call *composite* gradients, that is, vectors $\lambda^T\mathbf{C}$ that are convex combinations of the problem's $\mathbf{c}^i$.

One way of subdividing the criterion cone of Fig. 1 *Left* into two of what are called "subset-cones," and a consequent number of "cut-cones," is shown in Fig. 1 *Right*. In the illustration, though, there is only one cut-cone as a result of there being only two subset-cones. Employing composite gradient $\mathbf{d}^1$ for the purpose, one subset-cone is generated by $\mathbf{c}^2$, $\mathbf{c}^3$ and $\mathbf{d}^1$ and the other is generated by $\mathbf{c}^1$, $\mathbf{c}^2$ and $\mathbf{d}^1$. This leaves in between the cut-cone generated by $\mathbf{c}^2$ and $\mathbf{d}^1$ for a total of three "sub-cones." With this the case, the three terms of *sub-cone*, *subset-cone*, and *cut-cone* as used in this paper are defined as follows.

Definition 2. Consider the criterion cone of a $k = 3$ MOLP all of whose (three) $\mathbf{c}^i$ are linearly independent. Then any cone that is a subset of the criterion cone is a sub-cone.

Definition 3. Consider a criterion cone as above. In the process of subdividing a criterion cone, any 3-dimensional sub-cone that results is called a subset-cone.

Definition 4. Consider a criterion cone as above. In the process of subdividing a criterion cone, any 2-dimensional sub-cone that results is called a cut-cone.

Note that in the case of Fig. 1 and others to come, the relative interiors of all subset-cones and cut-cones are collectively exhaustive and mutually exclusive with regard to the relative interior of the original criterion cone. That is, their relative interiors form a *partition* of the relative interior of the original criterion cone. In this way, by subdividing a criterion cone into subset-cones and cut-cones, we can obtain the set of all efficient extreme points in a way that can reduce overall solution time. In the current case, instead of solving one MOLP with the full criterion cone of Fig. 1 *Left* to obtain all efficient extreme points, we have the option of solving the two *subset-cone MOLP* s and the one *cut-cone MOLP*

$$\begin{array}{lll} \max\{\mathbf{c}^2\mathbf{x}\} & \max\{\mathbf{c}^1\mathbf{x}\} & \max\{\mathbf{c}^2\mathbf{x}\} \\ \max\{\mathbf{c}^3\mathbf{x}\} & \max\{\mathbf{c}^2\mathbf{x}\} & \max\{\mathbf{d}^1\mathbf{x}\} \\ \max\{\mathbf{d}^1\mathbf{x}\} & \max\{\mathbf{d}^1\mathbf{x}\} & \text{s.t.} \quad \mathbf{x} \in S \\ \text{s.t.} \quad \mathbf{x} \in S & \text{s.t.} \quad \mathbf{x} \in S & \end{array}$$

for all of their respective efficient extreme points where the cut-cone MOLP is the one on the right. Then, by taking the union of the three resulting sets, we have the set of all efficient extreme points of the full MOLP. This follows from Theorem 1.

Theorem 1. *Consider an MOLP subdivided into sub-cones such that the relative interiors of all subset-cones and cut-cones form a partition of the relative interior of the MOLP's original criterion cone. Then by computing all efficient extreme points of all sub-cone MOLPs, all efficient extreme points of the original MOLP are obtained.*

Proof. Let $\mathbf{C}$ be the criterion matrix of the original MOLP, and assume that $\hat{\mathbf{x}} \in S$ is an efficient extreme point of this MOLP. This means that there exists a $\hat{\lambda} > \mathbf{0}$ such that $\hat{\mathbf{x}}$ is a maximizing solution of $\max\{\hat{\lambda}^T\mathbf{C}\mathbf{x} \mid \mathbf{x} \in S\}$, which in turn means that $\hat{\mathbf{x}}$ is a maximizing solution of $\max\{\boldsymbol{\beta}^T\mathbf{x} \mid \mathbf{x} \in S\}$ for some $\boldsymbol{\beta}$ pointing into the criterion cone generated by $\mathbf{C}$ (Steuer, 1986).

Assume that $\hat{\mathbf{x}}$ is not generated as an efficient extreme point of any sub-cone MOLP. This means there exists no $\boldsymbol{\delta}$ pointing into the

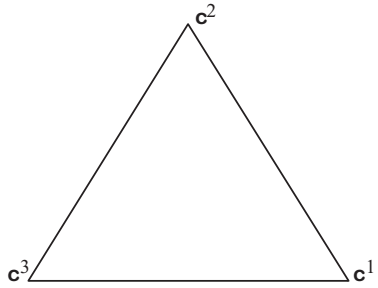

Fig. 2. Cross-section triangle of a criterion cone in the form of an equilateral triangle.

relative interior of a sub-cone such that $\hat{\mathbf{x}}$ is a maximizing solution of $\max\{\delta^T\mathbf{x} \mid \mathbf{x} \in S\}$. But since the union of the relative interiors of all sub-cones is the relative interior of the criterion cone generated by $\mathbf{C}$, this is impossible, thus proving the theorem. □

However, in all but the remotest of cases, it appears that we can obtain all efficient extreme points by just solving a problem's subset-cone MOLPs. Hence we will only talk about solving all subset-cone MOLPs to compute all efficient extreme points of an MOLP until we have a chance to discuss more about cut-cone MOLPs later in the paper.

With time savings being achieved by solving sub-cone MOLPs concurrently, criterion cones may be divided into subset-cones in different ways depending on how composite gradients are employed. Various schemes can be developed that use a single composite gradient in different ways or by employing additional composite gradients. The question then arises as to which schemes might have an edge over others at achieving time savings. In solving the subset-cone MOLPs of the different schemes concurrently, the paper is able to focus on features that play a role in the way different schemes can be effective in reducing elapsed time to compute all efficient extreme points.

As for different schemes for subdividing the criterion cone, Climaco and Antunes (1989) show that a MOLP with any number of objectives can be transformed into a problem with one objective formed by a strictly positive weighted sum of the original problem's objectives. Further, they show that there is a one-to-one correspondence between vectors pointing into the criterion cone and points in Λ where

$$\Lambda = \{\lambda \in \mathbb{R}^k \mid \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1\}.$$

Here Λ is the set of weighting vectors forming the composite gradients mentioned earlier in the section. While this result holds for any number of objectives k, in the case of 3 objectives this allows us to visually represent the cross-section triangle of a criterion cone in the form of an equilateral triangle as in Fig. 2.

In Fig. 3, in equilateral triangular form, we have the set of schemes that we investigate for subdividing a criterion cone in this paper. Note that the way the criterion cone is subdivided in Fig. 1 *Right* corresponds to the subdivided equilateral triangle found as the leftmost item in the top row of Fig. 3. As noted in Fig. 3, this way of subdividing a criterion cone is designated as Scheme 21. We use a two-digit number to identify each scheme. The first digit specifies the number of parts into which the criterion cone is subdivided. The second digit represents the *manner* in which a cone is subdivided. Note the respective commonalities among Schemes 21, 31 and 41 and Schemes 22, 32 and 42. In addition, each scheme is labeled with its edge-length calculation, but these calculations are not used until later in the paper as they are related to cut-cones.

Let us now consider Scheme 22. Using a horizontal line instead of a vertical line, this is another way of subdividing a criterion cone into two subset-cones. Employing composite gradients $\mathbf{d}^2$ and $\mathbf{d}^3$ to carry out the scheme, the subset-cone MOLPs of this scheme are

$\max\{\mathbf{c}^2\mathbf{x}\}$	$\max\{\mathbf{c}^1\mathbf{x}\}$
$\max\{\mathbf{d}^2\mathbf{x}\}$	$\max\{\mathbf{c}^3\mathbf{x}\}$
$\max\{\mathbf{d}^3\mathbf{x}\}$	$\max\{\mathbf{d}^2\mathbf{x}\}$
s.t. $\mathbf{x} \in S$	$\max\{\mathbf{d}^3\mathbf{x}\}$
	s.t. $\mathbf{x} \in S$

For the two parts of the Scheme 22 equilateral triangle to be of the same area, the λ of $\mathbf{d}^2$ is $(0, 1-\sqrt{1/2}, \sqrt{1/2})$ and the λ of $\mathbf{d}^3$ is $(\sqrt{1/2}, 1-\sqrt{1/2}, 0)$ where $\sqrt{1/2}$ is the .7071 in the triangle.

The number of objectives of a subset-cone MOLP is given by the number of corner points of the area of the equilateral triangle it is to represent. As for the objective functions themselves, they are given by the gradients and composite gradients that form the corner points of the area of the equilateral triangle being modeled. For Scheme 22, we observe that one of its subset-cone MOLPs has three objectives and the other has four.

Looking at the second row of Fig. 3, we see three schemes for breaking a criterion cone into three parts. For Scheme 31, all three subset-cone MOLPs have three objectives each involving $\mathbf{c}^2, \mathbf{c}^3$ and $\mathbf{d}^1$; $\mathbf{c}^2, \mathbf{d}^1$ and $\mathbf{d}^2$; and $\mathbf{c}^1, \mathbf{c}^2$ and $\mathbf{d}^2$; respectively. For Scheme 32, the λ of $\mathbf{d}^3$ is $(0, 1-\sqrt{1/3}, \sqrt{1/3})$ where $\sqrt{1/3}$ is the .5774 in the triangle, and the λ of $\mathbf{d}^5$ is $(0, 1-\sqrt{2/3}, \sqrt{2/3})$ where $\sqrt{2/3}$ is the .8165 in the triangle. In this scheme, only the subset-cone MOLP of the top part has three objectives while the other two have four objectives each. Scheme 33 is another scheme for subdividing a criterion cone into three parts. Like Scheme 31, all three subset-cone MOLPs of this scheme have three objectives each.

The third row of Fig. 3 shows schemes for subdividing a criterion cone into four parts. Just as Scheme 41 is a continuation of the manner of Schemes 21 and 31, Scheme 42 is a continuation of Schemes 22 and 32. In this scheme the λ for $\mathbf{d}^4$ is $(0, 1-\sqrt{1/4}, \sqrt{1/4})$ where $\sqrt{1/4}$ is the .5000 in the triangle, and the λ for $\mathbf{d}^8$ is $(0, 1-\sqrt{3/4}, \sqrt{3/4})$ where $\sqrt{3/4}$ is the .8660 in the triangle. Scheme 44 is of a different pattern and has all four of its subset-cone MOLPs with three objectives each.

With regard to the edge-length quantities specified below the scheme numbers, they represent the total length of the straight lines in the interior of the triangles where the unit of measure is the edge length of the equilateral triangle. For instance, the total length of the three straight lines in the triangle of Scheme 41 is $.9014 + .8660 + .9014 = 2.6688$. We call the straight lines *cuts* because they are used to "cut" the triangles into parts. Cuts are discussed in more detail in Section 4.

All of the schemes of Fig. 3 are used in the computational tests conducted later in the paper.

3. Code used

In this section we provide an overview of the code, Steuer (2017), that is used for computing all of the efficient extreme points in all of the MOLPs of this paper. Note that ADBASE is a general solver in the sense that it can solve for all efficient extreme points of any MOLP regardless of whether all objective function gradients are linearly independent or not. To help in our discussions. ADBASE derives its operation from the definitions and results of items **(1)** through **(10)** below.

(1) Associated with a basic feasible solution, let $\mathbf{B}$ be an $m \times m$ submatrix of the basic columns of $[\mathbf{A}, \mathbf{I}]_{m\times(n+m)}$ and let $\mathbf{N}$ be the

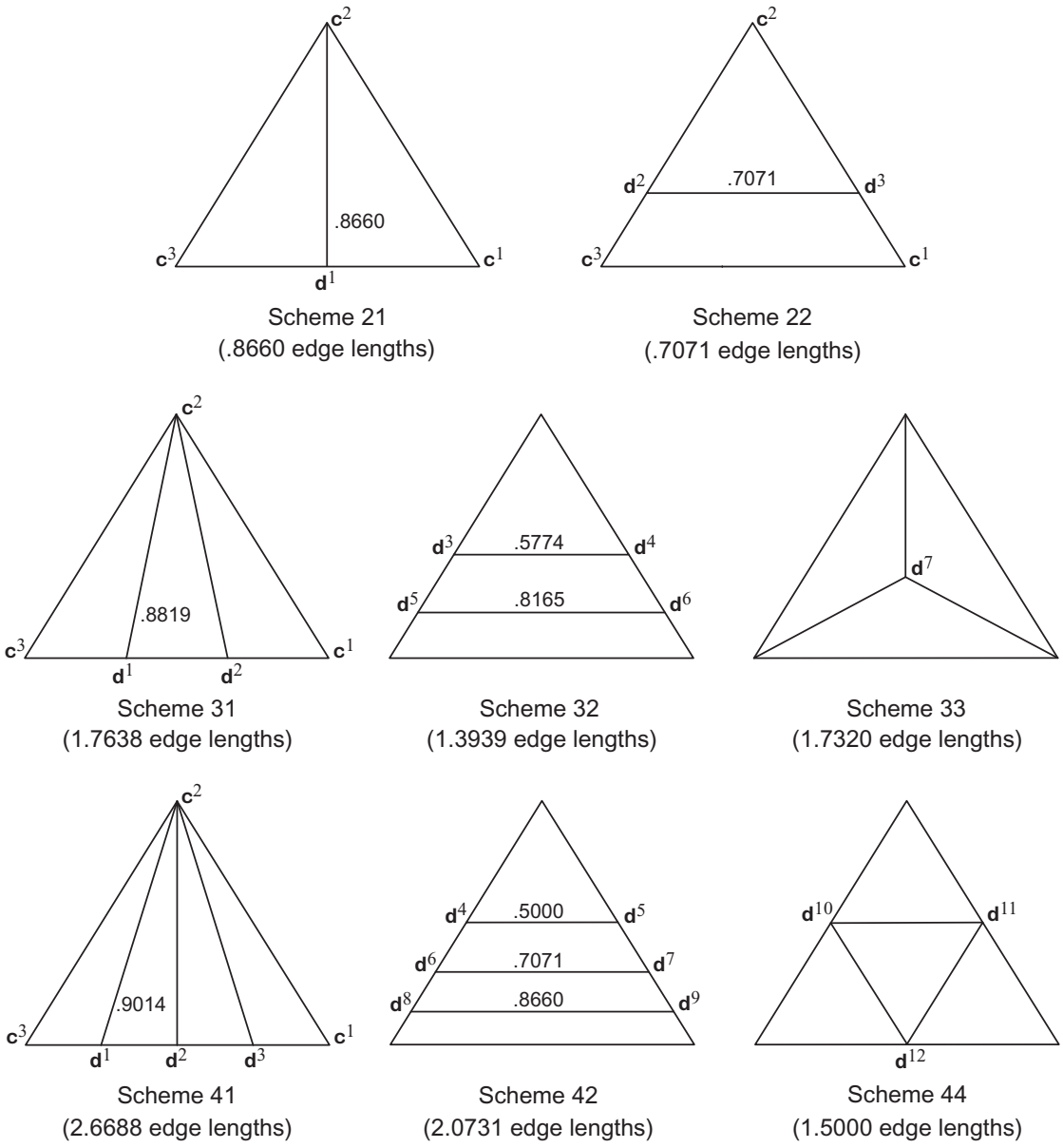

Fig. 3. Schemes for subdividing the criterion cone. For consistency, when subdividing an equilateral triangle into parts, we always do it so the parts are of equal area (this being equivalent to subdividing the cross-section triangle of a criterion cone into parts of equal size). (*Top row*): Schemes for subdividing the (equilateral) triangle into two parts. (*Middle row*): Schemes for subdividing into three parts. (*Bottom row*): Schemes for subdividing into four parts.

$m \times n$ submatrix of nonbasic columns. In the multiple objective simplex tableau for $\mathbf{B}$,

$\mathbf{x}_N$	$\mathbf{x}_B$		
$\mathbf{B}^{-1}\mathbf{N}$	$\mathbf{I}$	$\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{x}_B$
$\mathbf{W}_B$	$\mathbf{0}$	$\mathbf{C}_B\mathbf{B}^{-1}\mathbf{b}$	$\mathbf{z}$

which we will also call the *master problem* tableau, $\mathbf{W}_B$ is the $k \times n$ reduced cost matrix where $\mathbf{W}_B = \mathbf{C}_N - \mathbf{C}_B\mathbf{B}^{-1}\mathbf{N}$; $\mathbf{C}_N$ and $\mathbf{C}_B$ are the nonbasic and basic partitions of $[\mathbf{C}, \mathbf{0}]_{k\times(n+m)}$; and k is the number of objectives.

(2) $\mathbf{B}$ is an efficient basis iff for some $\lambda \in \text{rel}\,\Lambda$, $\lambda^T\mathbf{W}_B \leq \mathbf{0}^T$ where *rel* refers to relative interior.

(3) If $\mathbf{B}$ is an efficient basis, then the $\mathbf{x} \in S$ associated with $\mathbf{B}$ is an efficient extreme point.

(4) It suffices to obtain an initial efficient basis, and hence an initial efficient extreme point, by solving the *weighted-sums* LP

$$\max\{\lambda^T\mathbf{C}\mathbf{x} \mid \mathbf{x} \in S\}$$

for some $\lambda \in \text{rel}\,\Lambda$.

(5) If $\mathbf{x} \in S$ is an efficient extreme point, $\mathbf{x}$ has at least one efficient basis.

(6) Let $\mathbf{B}$ be an efficient basis. Then x_j is an *efficient nonbasic variable* iff the *subproblem* LP

$$\begin{aligned} &\max\{\mathbf{e}^T\mathbf{v}\} \\ &s.t. \quad \mathbf{W}_B\mathbf{y} + \mathbf{w}^j\delta + \mathbf{I}\mathbf{v} = \mathbf{0} \\ &\qquad\qquad \mathbf{0} \leq \mathbf{y} \in \mathbb{R}^n \\ &\qquad\qquad 0 \leq \delta \in \mathbb{R} \\ &\qquad\qquad \mathbf{0} \leq \mathbf{v} \in \mathbb{R}^k \end{aligned}$$

has a maximizing objective function value of zero where $\mathbf{w}^j$ is the jth column of $\mathbf{W}_B$ and $\mathbf{e}$ is a vector of ones.

(7) Let x_j be an efficient nonbasic variable. Then any feasible pivot from the x_j column (including any with negative pivot elements) is an *efficient pivot*.
(8) Let **B** be an efficient basis. If basis $\hat{\mathbf{B}}$ is obtainable from **B** by means of an efficient pivot, $\hat{\mathbf{B}}$ is an efficient basis and is said to be *adjacent* to **B**.
(9) Let $\mathbf{W}_B$ be the reduced cost matrix of an efficient basis **B**. By solving the subproblem LP of **(6)** once for each $\mathbf{w}^j$ column of $\mathbf{W}_B$ and then determining all resulting efficient pivots, knowledge about which variables are basic in all efficient bases that are adjacent to **B** is obtained.
(10) If, for a given efficient basis all of item **(9)** is carried out, the extreme point of the basis is noted in a list, and the efficient basis is now said to have been *processed*.

Whereas the master problem tableaus of **(1)** involve m constraint rows, the tableaus of the subproblem LPs of **(6)** have only as many constraint rows as there are rows to the criterion matrix sent to it.

To manage which efficient bases have been processed, and which efficient bases are known to exist via efficient pivots but have not yet been processed, two coded lists are maintained.

LISTB	contains the codes of all known efficient bases
LISTX	contains the codes of the efficient extreme points associated with the processed entries in LISTB

As for the codes in the lists, if the set of indices of the basic variables of a given efficient basis is {7, 5, 1, 3}, the code of this basis is $2^7 + 2^5 + 2^1 + 2^3 = 170$ and it would go in LISTB. If for this basis the set of indices of all nonzero basic variables is {7, 1, 3}, the code of the efficient extreme point of this basis is $2^7 + 2^1 + 2^3 = 138$ and it would go in LISTX. A code is decoded by successively dividing by 2 and noting which remainders equal 1.

In ADBASE, LISTB gets its start from item **(2)** above in which an initial efficient basis is obtained. The code of this basis becomes the first entry in LISTB. Bases in LISTB become processed as follows. All efficient bases adjacent to the efficient basis being processed (current basis) are obtained via **(9)**. Each is checked to determine if it is already in LISTB. Any that are not are added to the bottom of LISTB. Once this is done for all bases adjacent to the current basis, the (efficient) extreme point of the current basis is coded and placed in LISTX. If the extreme point code is not already in LISTX, the coordinates of the extreme point are outputted to a file. After all of this has been completed, the efficient basis is declared *processed* as in **(10)**, and the next basis in LISTB is selected for processing, and so forth.

In the beginning, LISTB grows rapidly, but eventually the processing of unprocessed efficient bases catches up with the end of the list. When this happens, all efficient extreme points have been found and the outputted file now contains the set of all efficient extreme points. Realizing the time that could be involved in checking the code of every efficient basis adjacent to every efficient basis, the entries in LISTB are indexed in a binary tree structure to reduce the time involved in carrying out all of the necessary LISTB checks.

Since the time complexity of doing a LISTB check in a binary tree is $O(p)$ where p is the number of nodes (efficient bases) in the tree, it is fastest to do a given LISTB check when the tree is small. Thus, if we were able to distribute the task so that all LISTB checks could be correctly carried out over trees that are smaller than otherwise would be the case, total LISTB check time can be reduced. This is one of the things achieved when subdividing a criterion cone and is pointed out again in Sections 6 and 7.

4. Efficient extreme points computed more than once

In the process of solving the subset-cone MOLPs of a scheme, there will, however, be efficient extreme points that are computed more than once. The usual case, when an efficient extreme point is computed more than once, is that it is computed in duplicate. But sometimes an efficient extreme point can be computed more than that. It depends upon the scheme and upon the unlikely event of a special situation. Efficient extreme points that are computed more than once lie along what is called in Section 2 a cut. With each straight line in the triangle of a scheme being a cut, they are where two subdivisions share a common straight-line boundary. For instance, in Scheme 21, the vertical line is a cut. In Scheme 32, the two horizontal lines are cuts, and in Scheme 33, the three line segments in the triangle are each cuts. Schemes 41, 42 and 44 all have three cuts each.

To begin illustrating how cuts cause efficient extreme points to be computed more than once, consider the $3 \times 6 \times 9$ MOLP

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9		
Obj			9		5		3			Max	
		8				4		−2	9	Max	
	1			3	−2		9	4	6	Max	
s.t.	6	4	1	1	3	5	6	4	8	≤	766
	−2		8			7		−3		≤	660
	5			1		5			1	≤	397
		1		8			7		6	≤	738
		2			5		6			≤	297
			−1		4			5		≤	344
										All vars ≥ 0	

This MOLP has 18 efficient bases and, as a consequence of the fact that none of the extreme points of the efficient bases is degenerate, the MOLP has 18 efficient extreme points.

Inputting the MOLP to the Trimap capability housed within the iMOLPe package of Climaco and Antunes (1989) and Alves, Antunes, and Climaco (2015), we obtain, after some editing, the triangular representation of the MOLP in Fig. 4 *Left*. The whole triangle of course represents all convex combinations of the $\mathbf{c}^i$. Inside the triangular representation we see regions numbered 1 to 18. They are the $\lambda^T\mathbf{C}$ *gradient indifference* regions, $\lambda \in \Lambda$, for the different efficient extreme points. That is, if $\lambda^T\mathbf{C}$ is a member of a given region, then among the optimal extreme points of the weighted-sums LP

$$\max \{\lambda^T\mathbf{C}\mathbf{x} \mid \mathbf{x} \in S\}$$

will be the efficient extreme point of that given region.

For instance, any gradient from the interior of gradient indifference region 9 causes the weighted-sum LP to generate efficient extreme point 9 as a unique optimal solution. And any gradient, say, from the relative interior of the leftmost boundary of region 9 causes the weighted-sums LP to have both efficient extreme points 9 and 8 as optimal solutions. Let $\mathbf{B}_9$ be the basis of extreme point 9 and $\mathbf{B}_8$ be the basis of extreme point 8. With the λ's of both gradients strictly positive, this then ties in to item **(2)** of ADBASE in that the λ of the first gradient satisfies only $\lambda^T\mathbf{W}_{B_9} \leq \mathbf{0}^T$ whereas the λ of the second gradient satisfies both $\lambda^T\mathbf{W}_{B_9} \leq \mathbf{0}^T$ and $\lambda^T\mathbf{W}_{B_8} \leq \mathbf{0}^T$.

Looking to Fig. 4 *Right*, the vertical line in the figure is there to simulate the cut of Scheme 21. Here the subset-cone MOLP of the left subdivision generates the 11 efficient extreme points of {1, 2, 5, 6, 7, 8, 9, 11, 14, 15, 17}, and the subset-cone MOLP of the right subdivision generates the 12 efficient extreme points of {1, 2, 3, 4, 5, 9, 10, 11, 12, 13, 16, 18}. The intersection of the two sets is {1, 2, 5, 9, 11}. These are the efficient extreme points that are computed in duplicate as they are among those computed by

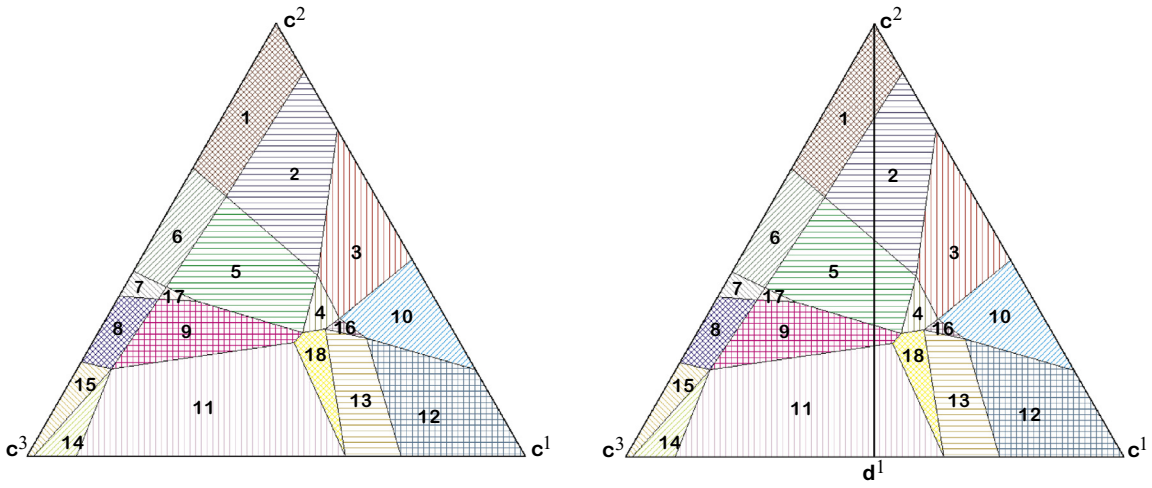

Fig. 4. (*Left*): Trimap produced representation of the $3 \times 6 \times 9$ MOLP showing the gradient indifference regions of each of the MOLP's 18 efficient extreme points. (*Right*): The cut of Scheme 21 superimposed upon the 18 gradient indifference regions.

each subset-cone MOLP. As seen in Fig. 4 *Right*, these are the five efficient extreme points that have the interiors (or more technically, relative interiors) of their gradient indifference regions intersected by the cut. Note that in this problem, without having to compare the batches of extreme points generated by the subset-cone MOLPs, one could solve directly for the duplicates by solving the cut-cone MOLP of

$$\max \{\mathbf{c}^2\mathbf{x}\}$$
$$\max \{\mathbf{d}^1\mathbf{x}\}$$
$$\text{s.t.} \quad \mathbf{x} \in S$$

Since here the number of efficient extreme points of the MOLP is the sum of the numbers of efficient extreme points generated by the subset-cone MOLPs less the number of efficient extreme points generated by the cut-cone MOLP, this gives rise to the idea of a *discrepancy* indicator in the form of

$$D = W_x - \sum_q P_x^q + \sum_r C_x^r \tag{1}$$

where

W_x	number of efficient extreme points of the MOLP
P_x^q	number of efficient extreme points generated by the subset-cone MOLP of the qth subdivision
C_x^r	number of efficient extreme points generated by the cut MOLP of the rth cut

The usefulness of the discrepancy indicator of (1) in our experiments is that it can function as a kind of "check digit" on the efficient extreme points generated in a subdivision process. For example, consider its use on the $3 \times 6 \times 9$ MOLP. With $W_x = 18$, $P_x^{left} = 11$, $P_x^{right} = 12$ and $C_x^1 = 5$, the discrepancy calculation is

$$D = 18 - (11 + 12) + (5) = 0$$

For all schemes in Fig. 3 other than Scheme 33, it is *natural* for D to be zero. After the following discussion about Scheme 33, we will elaborate more on our use of the term "natural".

While not necessary for computation, we see two possible related uses of the discrepancy indicator. One is when an analyst wishes to make one last minute check to make sure the numbers from the experiments are consistent with one another before handing over solutions to a decision maker. The other is when the decision maker himself or herself may want to see the value of D as a form of "insurance" that all possible potential solutions are being produced for examination.

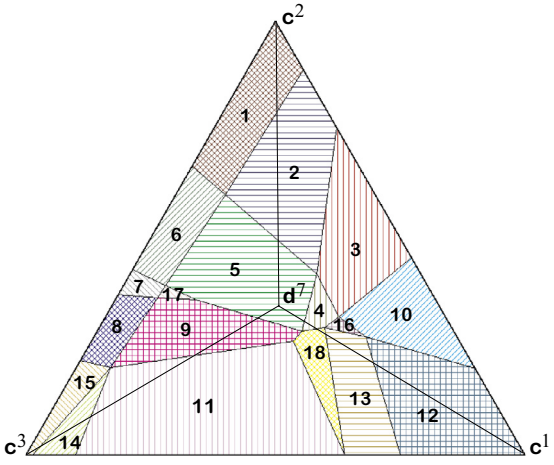

Fig. 5. Trimap produced representation of the $3 \times 6 \times 9$ MOLP showing the cuts of Scheme 33 superimposed upon the problem's 18 gradient indifference regions.

For Scheme 33, it is "natural" for it to have a discrepancy of $D = 1$. To see why, consider again the Trimap produced representation of the $3 \times 6 \times 9$ MOLP, but this time with the cuts of Scheme 33 superimposed upon it as in Fig. 5. Focusing on gradient indifference region 5, note that all three subset-cone MOLPs generate this efficient extreme point, and all three of the cut-cone MOLPs

$$\max \{\mathbf{c}^1\mathbf{x}\} \quad \max \{\mathbf{c}^2\mathbf{x}\} \quad \max \{\mathbf{c}^3\mathbf{x}\}$$
$$\max \{\mathbf{d}^7\mathbf{x}\} \quad \max \{\mathbf{d}^7\mathbf{x}\} \quad \max \{\mathbf{d}^7\mathbf{x}\}$$
$$\text{s.t.} \quad \mathbf{x} \in S \quad \text{s.t.} \quad \mathbf{x} \in S \quad \text{s.t.} \quad \mathbf{x} \in S$$

generate it as well. This then gives us for discrepancy

$$D = 18 - (11 + 10 + 9) + (5 + 3 + 5) = 1$$

A positive discrepancy is caused by too many efficient extreme points being generated by the cut-cone MOLPs. This will typically happen whenever cuts meet at a point in the interior of the equilateral triangle, such as $\mathbf{d}^7$. Thus, in contrast to the "natural" zero discrepancy of all of the other schemes, it is "natural" for Scheme 33 to have a discrepancy result of 1.

The reason we have been saying "natural" is that there can occur possible, but not common, special situations that can cause a scheme's discrepancy to differ from its "natural" value.

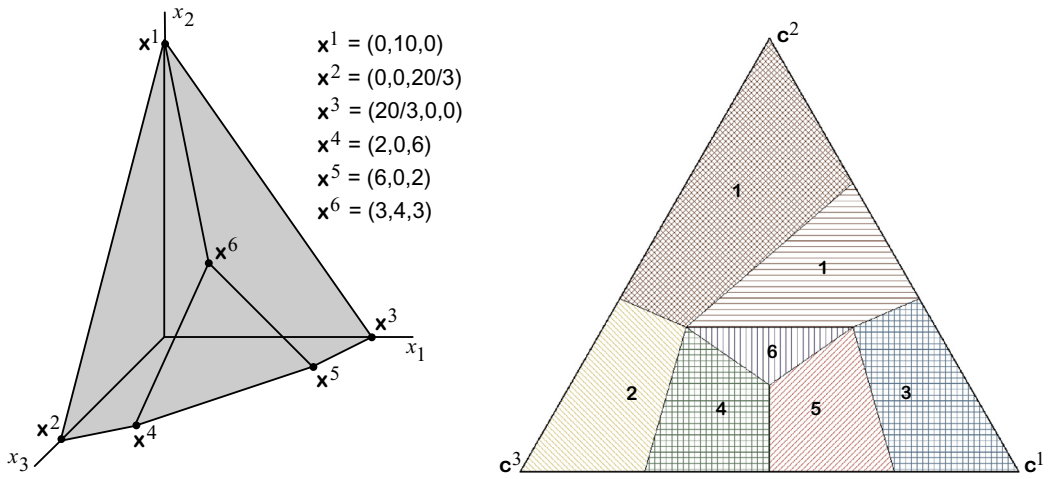

Fig. 6. (*Left*): A Scheme 21 $D = 2$ MOLP in decision space. (*Right*): The MOLP's Trimap produced representation showing how gradient indifference regions 4 and 5 are each tangent to the vertical line cut of Scheme 21. The only reason "1" appears twice in Fig. 6 *Right* is because efficient extreme point $\mathbf{x}^1$ is degenerate in this example.

Consider the following MOLP

	x_1	x_2	x_3		
Obj	1			Max	
		1		Max	
			1	Max	
s.t.	6	12	18	$\leq$	120
	16	8	16	$\leq$	128
	18	12	6	$\leq$	120
	All vars ≥ 0				

whose graph in decision space and Trimap produced representation are in Fig. 6. Imagine the vertical line cut of Scheme 21 superimposed upon the Trimap produced representation in Fig. 6 *Right*. Since the left subset-cone MOLP generates efficient extreme points {1, 2, 4, 6}, the right subset-cone MOLP generates {1, 3, 5, 6}, and the cut-cone MOLP generates {1, 4, 5, 6}, we get for this example

$$D = 6 - (4 + 4) + (4) = 2$$

But $D = 0$ is the "natural" discrepancy for Scheme 21. The cause of $D = 2$ is that, beyond the natural, we get an additional unit of discrepancy for each gradient indifference region that is tangent to a cut. Because gradient indifference regions 4 and 5 are each tangent to the vertical cut, the discrepancy is two more than what would be the natural figure.

There is another special situation that is probably much more rare than the one just described. It is when the gradient indifference region of an efficient extreme point is just a singleton point. An example of such an efficient extreme point is $\mathbf{x}^4$ on page 185 of Steuer (1986). Not only is collapsing in an MOLP required for this to occur, but it is necessary for the collapsing to possess an efficient extreme point whose gradient indifference region is a singleton point, and it is necessary for that singleton point to lie exactly along the relative interior of a cut. It is only if such an extraordinary sequence of events occurs that any cut-cone MOLP must be solved to round out the computation of all efficient extreme points. This is why we have concluded for all practical purposes that the solving of any cut-cone MOLP can be effectively skipped when solving for all efficient extreme points of an MOLP.

Since strong coincidences are required for either special situation to occur, while possible, it is not likely that we will encounter many such situations in testing or practice. In fact, when more or fewer units of discrepancy are encountered than natural, it is probably more likely for the cause to be round-off error rather than the exact occurrence of a special situation. However, it is nice to have the discrepancy measure to flag non-natural discrepancies in the results whatever their cause, just so that we are aware. Of course, one could always solve all cut-cone MOLPs for peace of mind.

5. Random problem generator

To carry out our experiments, a large number of MOLPs with various numbers of constraints and variables are needed. Fortunately, ADBASE is equipped with a random problem generator. To configure the random problem generator, we specify the following quantities:

NUMB	number of MOLPs to be generated
k	number of objectives
m	number of constraints
n	number of variables
[JLC, JUC]	interval from which nonzero integers are selected for the nonzero elements of **C**
JCDEN	percent nonzero density of **C**-matrix
[JLA, JUA]	interval from which nonzero integers are selected for the nonzero elements of **A**
JADEN	percent nonzero density of **A**-matrix
[JLB, JUB]	interval from which nonzero integers are selected for the computation of the elements of **b**

While the **A**-matrix and **b**-vector are randomly generated as described in Steuer and Piercy (2005), the **C**-matrix is generated as follows. First, all of the elements of **C** are populated by randomly selecting uniformly from the nonzero integers contained in the interval [JLC, JUC]. Then, the elements of **C** are randomly selected and converted to zeros, making sure no row or column becomes all zeros, until the percent nonzero density JCDEN is achieved.

With the random problem generator, in all of the experiments of this paper, the following parameters have been set as follows:

NUMB	= 20
[JLC, JUC]	= [−2, 9]
JCDEN	= 75
[JLA, JUA]	= [−8, 8]
JADEN	= 50
[JLB, JUB]	= [50,100]

Using these settings, the sample size for each of our experiments was 20. Also, by setting [JLC, JUC] = [−2, 9] the criterion cones of the MOLPs generated are not especially narrow. This allows for healthy numbers of efficient extreme points to be seen in the experiments.

6. The experiments

In this section, we describe our experimental design and report on the results obtained from our experiments. The experiments are designed to investigate the effectiveness of the eight schemes of Fig. 3. In the testing of each scheme, six problem sizes are used. They are listed in the leftmost column of Table 1. The eight schemes and six problem sizes yield 48 scheme/problem-size combinations, with each combination being an experiment.

As a first step in setting up the experiments, the ADBASE random problem generator, as parameterized in Section 5, was employed to randomly generate and solve 20 MOLPs for each of the six problem sizes without subdividing the criterion cone. This was done so each scheme could be tested using the same 20 problems of each problem size. By not subdividing the criterion cone for any of the 120 MOLPs, this enables us to obtain individual *baseline* results for the 20 MOLPs of each problem size against which subdivision results can be compared. Average baseline information about the 20 MOLPs of each problem size is found in Table 1. For example, in the 160 × 240 row, where 160 is m (number of constraints) and 240 is n (number of variables), the 20 randomly generated MOLPs of this problem size have on average 87,732 efficient extreme points each, and took on average, using ADBASE, 119.95 seconds each to compute.[2] This comes to 1.3672 seconds per thousand efficient extreme points on average (119.95/87,732).

Commenting further on the entries in the seconds/1000 column of Table 1, they increase with the problem sizes in the table for two reasons. One is because the master problem tableau of **(1)** grows in size with m and n, so pivots in it take longer. The other is that the subproblem LPs of **(6)** take longer to solve because $\mathbf{W}_B$ takes on more columns since the number of master problem nonbasic variables increases with problem size.

With regard to the six batches of 20 randomly generated MOLPs, the procedure for conducting each of the 48 scheme/problem-size experiments is as follows. That is, for a given scheme/problem-size experiment:

1. obtain the number of efficient extreme points and the baseline time to solve for them for each of the 20 MOLPs of the problem-size of the experiment;
2. obtain the numbers of efficient extreme points generated by the subset-cone MOLPs of the scheme and the times to solve for them for each of the 20 MOLPs of the problem size of the experiment;
3. obtain the numbers of efficient extreme points generated by the cut-cone MOLPs of the scheme and the times to solve for them for each of the 20 MOLPs of the problem size of the experiment;

[2] All computer times in this paper are in seconds using ADBASE on an HP desktop with an i7-4790 processor.

Table 1
Average baseline (where baseline means solving without subdividing the criterion cone) results for the 20 MOLPs of each problem-size category.

Problem size	Eff ext pts	Seconds	Seconds/1000
3 × 40 × 60	1,467	0.14	0.0954
80 × 120	13,023	4.48	0.3440
120 × 180	43,372	33.10	0.7632
160 × 240	87,732	119.95	1.3672
200 × 300	162,193	358.16	2.2082
240 × 360	267,100	893.00	3.3433

4. tabulate the efficient extreme point and time results from the above steps to discrepancy check and calculate summary statistics.

With the sample size of each scheme/problem-size experiment being 20, the processing as above of each of the 20 MOLPs of an experiment is a *trial*. Thus, over the 48 experiments, there are 960 trials. Note that the number of subset-cone MOLP and cut-cone MOLP optimizations per trial is a function of the scheme being tested. For instance, each trial of a Scheme 21 experiment involves two subset-cone MOLPs and one cut-cone MOLP, while each trial of a Scheme 44 experiment involves four subset-cone MOLPs and three cut-cone MOLPs.

In addition to D, W_x, P_x^q and C_x^r, let us define the notation of

W_t	time in seconds to solve the MOLP without subdividing the criterion cone
P_t^q	time in seconds to solve the qth subset-cone MOLP
T_x	in percentage terms, the total number of efficient extreme points generated by the subset-cone MOLPs in excess of the efficient extreme points possessed by the MOLP of the trial. Let $t_x = \sum_q P_x^q / W_x$. Then $T_x = t_x \times 100$
T_t	in percentage terms, the total time taken by the subset-cone MOLP process relative to the time taken to solve the MOLP of the trial when not subdividing the criterion cone. Let $t_t = \sum_q P_t^q / W_t$. Then $T_t = t_t \times 100$
E_x	in percentage terms, the maximum percentage of the total number of efficient extreme points generated by any subset-cone MOLP, that is, $E_x = \max_q \{P_x^q / W_x\} \times 100$
E_t	in percentage terms, the elapsed time required to solve all subset-cone MOLPs concurrently relative to the time required to solve the MOLP of the trial without subdividing the criterion cone, that is, $E_t = \max_q \{P_t^q / W_t\} \times 100$.

With this notation, we tabulate experimental results. For example, in Table 2, we have the tabulation of the Scheme 21/120 × 180 problem-size experiment and related statistics. Table 2 is one of 48 such tabulations obtained from our experiments.

Looking into the tabulation of Table 2, let us consider the 5th trial. The MOLP of this trial has 41,158 efficient extreme points and took a baseline time of 31.6 seconds to compute. The two subset-cone MOLPs generated 16,775 and 24,715 efficient extreme points in 12.5 and 18.6 seconds, respectively. The cut-cone MOLP for this trial generated 332 efficient extreme points as seen in column C_x^1. Thus, the discrepancy D for the problem of this trial is zero since $D = 41,158 - (16,775 + 24,715) + (332) = 0$, which is natural for problems of this scheme.

With 332 efficient extreme points computed in duplicate by the scheme, the entry in the T_x column of the trial is 0.81% (332/41,158). This is the percentage of efficient extreme points that were computed in excess by the subset-cone MOLP process. From a cumulative time point of view, it is seen that the sum of times to conduct the two subset-cone MOLP optimizations is 31.1 seconds (12.5 + 18.6). Comparing this cumulative time against the 31.6 seconds required to compute all efficient extreme points in baseline, this is 1.6% less (0.5/31.6). This is reported as 98.4% in the T_t

Table 2
Tabulation and summary statistics for the Scheme 21/120 × 180 problem-size experiment.

	W_x	W_t	P_x^1	P_t^1	P_x^2	P_t^2	C_x^1	D	T_x	T_t	E_x	E_t
1	37,972	29.2	19,033	14.7	19,248	14.0	309	0	0.81	98.4	50.7	50.4
2	41,855	31.7	18,396	13.9	23,749	17.8	290	0	0.69	99.9	56.7	56.2
3	33,604	25.2	17,512	12.9	16,413	11.9	321	0	0.96	98.3	52.1	51.1
4	37,818	29.3	20,395	15.6	17,692	13.7	269	0	0.71	100.1	53.9	53.2
5	41,158	31.6	16,775	12.5	24,715	18.6	332	0	0.81	98.4	60.1	59.0
6	33,604	25.2	17,512	12.9	16,413	11.9	321	0	0.96	98.3	52.1	51.1
7	46,923	36.3	26,246	20.0	21,070	15.7	392	−1	0.84	98.2	55.9	55.1
8	55,724	42.6	27,905	20.7	28,225	21.8	406	0	0.73	99.8	50.7	51.3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
19	42,884	31.8	22,643	17.8	18,955	13.6	359	0	0.84	98.8	56.6	56.0
20	52,694	38.2	26,661	18.7	26,414	18.7	381	0	0.72	97.9	50.6	49.0
Ave	43,372	33.1	22,643	17.0	21,069	15.7	340		0.79	98.6	54.7	53.6
								Min	0.69	97.0	50.6	49.0
								Max	0.96	100.1	60.1	59.0

Table 3
Summary statistics for the Scheme 21 experiments. As in Tables 4–6 that follow, $n = 1.5m$. Also, the rows that span the full width of a table are called "average" rows as they are taken from values in the "Ave" rows of the tabulations as in Table 2.

Sch	m	W_x	W_t	S^1_{1000}	S^2_{1000}	S^3_{1000}	S^4_{1000}	T_x	T_t	E_x	E_t
21	40	1,467	0.14	.0952	.0947	–	–	4.01	103.7	60.0	59.5
							Min	2.83	97.1	53.0	52.5
							Max	5.44	112.5	69.8	70.0
	80	13,023	4.48	.3359	.3341	–	–	1.41	98.7	58.8	57.2
							Min	1.08	96.9	52.3	49.6
							Max	2.25	100.8	70.5	68.7
	120	43,372	33.10	.7474	.7451	–	–	0.79	98.6	54.7	53.6
							Min	0.69	97.0	50.6	49.0
							Max	0.96	100.1	60.1	59.0
	160	87,732	119.95	1.3478	1.3461	–	–	0.55	99.0	55.0	54.5
							Min	0.44	97.9	50.8	50.5
							Max	0.64	100.6	62.0	61.2
	200	162,193	35816	2.1862	2.1666	–	–	0.41	99.0	54.9	54.2
							Min	0.34	96.6	50.3	49.8
							Max	0.48	100.2	61.6	61.2
	240	267,100	893.00	3.2847	3.2616	–	–	0.32	98.5	54.0	53.2
							Min	0.25	93.3	50.3	46.9
							Max	0.39	100.2	61.4	60.4

column of Table 2. Note that in all but one trial, the values in the T_t column are less than 100%. This means that even though extra time is involved in computing the duplicates of column C_x^1, in 19 of the 20 trials of this experiment this extra time is more than offset by the savings accruing from spreading the LISTB checks out over two trees rather than one.

The last two columns of the tabulation provide E_x and E_t values. E_x is the maximum percentage of the total number of efficient extreme points generated by a subset-cone MOLP. For the 5th trial, the second subset-cone MOLP generated the most efficient extreme points. Thus, the E_x value for this trial is 60.1% (24,715/41,158). E_t is, in percentage terms, the time taken to solve all subset-cone MOLPs of a scheme when solving them concurrently relative to the baseline solution time of the MOLP. Since, for the 5th trial, the second subset-cone MOLP took the longest time, E_t for this trial is 59.0% (18.6/31.6). This means that all efficient extreme points of the MOLP of this trial can be computed in 18.6 seconds using Scheme 21 as opposed to 31.6 seconds for the baseline MOLP with no subdivision of the criterion cone. This is a savings of 41.0%. Despite the 41.0%, all of the other trials in the experiment saved more. This results from the fact that this trial exhibits more imbalance in terms of the number of efficient extreme points generated by the subset-cone MOLPs (16,775 vs. 24,715) than is seen in any of the other trials of the experiment.

As for the D column in the tabulation, it contains all zeros except for a −1 for trial 7, and a 1 and a 2 among the trials not shown. Since the 1 and 2 discrepancies could well be the result of special situations such as discussed in Section 4, that is, gradient indifference regions being numerically tangent to a cut, they are not bothersome to us because we can see how they can happen.

However, the −1 discrepancy is of some concern because it means that an efficient extreme point was missed, possibly because its gradient indifference region is a singleton point, but probably because its gradient indifference region is very small and because numerical conditions inside the code at the time the extreme point should have been calculated were such that the code did not see it. After all, with tens of thousands of gradient indifference regions, the smallest can be very small. Given that only a small number of units of negative discrepancy appear over the rest of the experiments, and since tuning is involved in any mathematical programming algorithm, we are comfortable with the position that our code is running as well as can be expected for the tasks of this paper.

The last three rows of Table 2 present summary statistics for the experiment in the form of averages for all columns except D as well as minimum and maximum values for the columns of T_x, T_t, E_x and E_t. Looking at the "Min" row in Table 2, it is interesting to note that the minimum value of E_t is 49.0%. This is from the 20th trial. It results from the trial's cumulative run time being 97.9% of baseline and the fact that the subset-cone MOLPs are well balanced (26,661 vs. 26,414 efficient extreme points). With this causing the

Table 4
Summary statistics for the Scheme 22 experiments.

Sch	m	W_x	W_t	S^1_{1000}	S^2_{1000}	S^3_{1000}	S^4_{1000}	T_x	T_t	E_x	E_t
22	40	1,467	0.14	.0996	.1193	–	–	3.85	119.3	59.7	60.3
							Min	2.68	111.4	51.8	57.1
							Max	5.39	108.7	59.5	62.7
	80	13,023	4.48	.3405	.4047	–	–	1.35	108.7	59.5	62.7
							Min	0.98	105.2	51.1	56.2
							Max	1.75	112.8	70.9	81.1
	120	43,372	33.10	.7595	.8874	–	–	0.73	108.4	55.6	59.8
							Min	0.62	105.5	50.9	55.4
							Max	0.85	111.8	63.7	68.1
	160	87,732	119.95	1.3946	1.6136	–	–	0.50	110.5	54.6	60.3
							Min	0.37	107.7	50.2	54.5
							Max	0.60	134.3	59.4	71.6
	200	162,193	358.16	2.1905	2.6139	–	–	0.37	108.2	55.8	59.2
							Min	0.30	104.8	50.2	54.4
							Max	0.48	113.7	64.7	65.2
	240	267,100	893.00	3.3223	3.8817	–	–	0.28	107.5	55.8	59.5
							Min	0.21	104.1	50.5	52.6
							Max	0.33	111.7	64.3	70.9

Table 5
Summary statistics for the Scheme 31 experiments.

Sch	m	W_x	W_t	S^1_{1000}	S^2_{1000}	S^3_{1000}	S^4_{1000}	T_x	T_t	E_x	E_t
31	40	1,467	0.14	.1002	.0991	.1013	–	8.55	114.6	48.1	49.7
							Min	6.55	107.6	41.2	44.2
							Max	11.03	125.0	54.0	56.3
	80	13,023	4.48	.3238	.3308	.3274	–	2.78	97.8	44.5	42.7
							Min	2.30	95.3	37.3	35.0
							Max	4.13	99.3	51.0	50.4
	120	43,372	33.10	.7320	.7370	.7272	–	1.55	97.5	43.2	41.1
							Min	1.30	95.7	38.2	35.9
							Max	1.83	99.1	50.5	48.5
	160	87,732	119.95	1.3251	1.3376	1.3251	–	1.07	98.3	41.9	41.0
							Min	0.93	97.0	36.8	36.4
							Max	1.27	99.6	48.7	47.8
	200	162,193	358.16	2.1469	2.1851	2.1456	–	0.81	98.7	41.9	41.5
							Min	0.69	94.3	36.3	34.2
							Max	0.94	102.2	50.5	50.8
	240	267,100	893.00	3.2639	3.2914	3.3106	–	0.63	99.0	41.7	41.1
							Min	0.51	94.8	36.2	35.6
							Max	0.80	102.8	47.7	47.4

2.1% time savings to be split nearly 50/50, this is how the E_t result of 49.0% is obtained. Looking at the "Max" row, no problem out of the 20 trials generated more than 0.96% of all efficient extreme points in excess and no trial had an elapsed time savings of less than 41% (a result of the 59.0% figure of trial 5 discussed earlier). Moreover, in only one trial was the subdivision process, as reflected in the T_t column, not faster than solving the MOLP of the trial without subdividing the criterion cone.

7. Discussion

In this section, we study Tables 3–6 which report results from the experiments conducted on Schemes 21, 22, 31 and 41, respectively. Actually, these four tables suffice for all eight schemes as they do a very good job in bringing out all of the points to be learned from the experiments of the paper. This is explained as we go through the section.

In the tables, columns W_x and W_t repeat baseline information from Table 1 about numbers of efficient extreme points and the times to compute them. This is done to facilitate comparisons with the results reported on in the columns to the right. Note that in the tables we make use of the additional notation of

S^q_{1000}	time taken to solve the qth subset-cone MOLP measured in seconds per thousand efficient extreme points generated

to report on results in seconds/1000 efficient extreme points stemming from the subset-cone MOLPs employed by the different schemes. Let us now see what can be observed.

1. A good beginning is to compare Scheme 21 with Scheme 22. For this, consider Tables 3 and 4. Note that while both of Scheme 21's subset-cone MOLPs have three objectives, the second subset-cone MOLP of Scheme 22 has four. This makes a difference. While the values in the S^1_{1000} column of Table 3, S^2_{1000} column of Table 3, and S^1_{1000} of Table 4 are relatively the same, the values in the S^2_{1000} column of Table 4, the column of the four-objective subset-cone MOLPs, are noticeably different, being 17–20% larger than their counterparts over all cases. Thus, that much more work is required to generate an efficient extreme point in a four-objective subset-cone MOLP than in a three. This is logical because, for a four-objective subset-cone MOLP, the subproblem LP in step **(6)** of Section 3 has one more constraint. This causes the time results of Table 4 to fall behind those of Table 3 rendering Scheme 22 noncompetitive relative to Scheme 21. By the same token, Scheme 32 is noncompetitive relative to Scheme 31, and Scheme 42 is noncompetitive relative to Scheme 41. This is why tables for Schemes 32 and 42 are not shown.

2. Once again consider the 49.0% E_t result discussed at the end of the previous section. It shows up in the "Min" row of the 120×180 problem-size experiments of Table 3 just as it should. Note, though, that there are three other figures in the 40's in the

Table 6
Summary statistics for the Scheme 41 experiments.

Sch	m	W_x	W_t	S^1_{1000}	S^2_{1000}	S^3_{1000}	S^4_{1000}	T_x	T_t	E_x	E_t
41	40	1,467	0.14	.1001	.0978	.0987	.0989	12.29	117.1	38.9	39.7
							Min	9.28	105.5	34.0	34.3
							Max	14.43	129.6	43.8	44.6
	80	13,023	4.48	.3149	.3247	.3237	.3181	4.21	97.3	36.2	34.1
							Min	3.42	95.4	29.4	28.9
							Max	6.56	99.6	42.8	39.9
	120	43,372	33.10	.7194	.7292	.7299	.7077	2.31	97.0	32.6	31.0
							Min	1.93	95.0	28.3	26.3
							Max	2.59	98.4	38.1	36.3
	160	87,732	119.95	1.3029	1.3187	1.3195	1.2962	1.60	97.4	33.1	32.0
							Min	1.38	96.2	27.7	26.6
							Max	1.81	98.5	38.4	37.8
	200	162,193	358.16	2.1237	2.1631	2.1435	2.1031	1.20	97.9	32.7	31.9
							Min	0.97	97.1	29.4	29.0
							Max	1.40	99.7	37.5	37.3
	240	267,100	893.00	3.2228	3.2554	3.2589	3.2026	0.93	97.8	32.5	31.6
							Min	0.81	95.6	28.8	27.1
							Max	1.10	100.1	36.9	35.8

Table 7
Relationships between the total cut length of a scheme and the percentage of extra efficient extreme points generated.

Scheme	Total cut length	Ave T_x (ave percentage of excess efficient extreme points)					
		40×60	80×120	120×180	160×240	200×300	240×360
21	0.8660	4.0118	1.4143	0.7890	0.5467	0.4082	0.3150
22	0.7071	3.8479	1.3519	0.7290	0.5049	0.3691	0.2775
31	1.7638	8.5454	2.7769	1.5475	1.0672	0.8127	0.6261
32	1.3939	7.4081	2.5900	1.3431	0.9483	0.6932	0.5405
33	1.7320	7.7665	2.6923	1.4230	0.9951	0.7398	0.5768
41	2.6688	12.2919	4.2116	2.3055	1.5955	1.2007	0.9349
42	2.0731	10.7305	3.7796	1.9877	1.4085	1.0301	0.7947
44	1.5000	8.2328	2.9014	1.5204	1.0923	0.8066	0.6194
Slope		0.2124	0.6141	1.1579	1.7154	2.2089	2.8677
t-value		12.0694	9.8998	12.7855	10.0026	12.2476	13.6505

E_t column of Table 3, too. Thus, a time savings of over 50% is not a totally unusual phenomenon even when an MOLP is broken down into only two parts.

3. In comparing the results of Scheme 33 against Scheme 31, nothing new comes up other than for the fact that the Scheme 33 manner of subdividing a criterion cone has the disadvantage of only being good for certain numbers of subset-cones when subdividing. While three is possible, there is no way to use this manner to subdivide a criterion cone into two parts, nor four or five, as there is with the manner of Scheme 31. The next possibility with Scheme 33 is only nine, which is outside the scope of this paper. Hence, with there being no point, we do not show a table for Scheme 33. For the same reasons we do not show a table for Scheme 44 as its manner of subdividing, again, has the disadvantage of being limited to only certain numbers of parts, with nine being its next as well (when using three rows of triangles instead of two).

4. Over all of the experiments, in percentage terms, the cumulative time taken to solve all subset-cone MOLPs of an MOLP is typically less than when solving the MOLP without subdividing the criterion cone. Other than for the 40×60 problem-size experiments across all tables and for all of the Scheme 22 experiments in Table 4, this is evident by the extent to which the values are below 100% in the T_t columns in all but a few of the "Max" cases in Tables 3, 5 and 6. This is due to the spreading out of the LISTB checks to the subset-cone MOLPs where the trees need not be as large.

5. Concerning efficient extreme points computed more than once, there is a connection to total cut length. For example, we see this when comparing the total cut lengths given in Fig. 3 for Schemes 21, 22, 31 and 41 with the figures given in the "average" rows of the T_x columns in Tables 3–6, where T_x is as defined earlier. Putting these observations to statistical tests, we have Table 7.

6. To illustrate the table, consider the 240×360 column with respect to the Scheme 21, 22, 31 and 41 rows. For example, the entries 0.3150, 0.2775, 0.6261 and 0.9349 come from the rounded figures of 0.32, 0.28, 0.63 and 0.93 in the bottommost "average" rows of Tables 3–6, respectively. Regressing the different problem-size columns onto total cut length, we have the very high t-values in the bottom row of the table, indicating a very strong positive relationship between the total cut length of a scheme and the percentage of extra efficient extreme points generated by the scheme. Thus, with increased cut lengths so strongly leading to the computation of increased numbers of extra efficient extreme points, it can be safely concluded that the shorter the total cut length of a scheme, all other things equal, the more computationally efficient the method.

7. As for units of discrepancy, there were no units of unnatural discrepancies in any of the 16 experiments involving problem sizes of 40×60 and 80×120. But in problems sizes 120×180 and larger, units occasionally arose. As for units of negative unnatural discrepancy, using ADBASE they only appear to be about a once in every quarter million efficient extreme points phenomenon.

8. Overall, the manner of subdividing a criterion cone shown by Schemes 21, 31 and 41 is arguably the most efficient and versatile of the manners studied in this paper. Considering only the MOLP problem sizes of 120×180 and larger with this manner of subdividing a criterion cone, it is informative to form Table 8. What the first line of this table says is that of the 80 largest MOLPs employed in the experiments, none of the 80 experienced time

Table 8
Time savings guide covering the 80 largest MOLPs when using the manner of Schemes 21, 31 and 41 for subdividing the criterion cone.

	Worst time savings (%)	Average time savings (%)
2 parts	38.8	46.1
3 parts	49.2	58.3
4 parts	62.2	69.6

savings of less than 38.8% when the criterion cone was broken down into two parts according to the manner of Scheme 21, with the average time savings being 46.1% for this case. The 38.8 figure is evidenced by scanning the bottom two-thirds of the E_t column of Table 3 and finding that 61.2 is the largest number in it.

What the second line says is that of the 80 largest MOLPs employed in the experiments, none of them experienced time savings of less than 49.2% when the criterion cone was broken down into three parts according to the manner of Scheme 31, with the average time savings for this case being 58.3%. The 49.2 figure is evidenced by scanning the bottom two-thirds of the E_t column of Table 5 and finding that 50.8 is the largest number in it.

What the third line says is that of the 80 largest MOLPs, none of them experienced time savings of less than 62.2% when subdividing the criterion cone into four parts according to Scheme 41, with the average time savings for this case being 69.6%.

8. Future directions

By subdividing the criterion cone, this paper investigates how the times required to compute all efficient extreme points of a multiple objective linear program can be substantially reduced. For instance, with Table 8 again as a guide, consider a $3 \times 240 \times 360$ MOLP. With such an MOLP, which would otherwise take on average about 893 seconds for the task, we can anticipate that it would take on average (a) about 481 seconds (.539*893) should the criterion cone be broken into two parts, (b) about 372 seconds (.417*893) should the criterion cone be broken into three parts, and (c) about 271 seconds (.304*893) should the criterion cone be broken into four parts, resulting in elapsed time savings of 412, 521 and 622 seconds on average, respectively.

While the code used in the experiments of this paper was ADBASE, the code is immaterial as any code for computing all efficient extreme points should experience the same percentage savings. This is because no changes are made to the code. The only changes that are made are in how formulations sent to the code are constructed.

The strategy of the paper is designed to be especially useful in situations where one's time-budget for computing all efficient extreme points is not enough to compute them as normally (that is, in a single run). Then by comparing the time that it would normally take to compute all efficient extreme points with Table 8 and the time in one's budget, one should then be able to develop a good idea about how many parts into which to break the criterion cone.

As for future research along the lines of this paper, breaking three-objective MOLPs into many more parts, as well as MOLPs with four and five objectives need to be investigated. Also, the issue of collapsing needs to be further explored as this affects the degrees to which decision space algorithms and outcome space algorithms can be made interchangeable. Furthermore, although unproven at this point as there has not been enough time, there is the prospect that this paper may well benefit certain algorithms in the area of integer multiple objective linear programming. By this we are referring to branch-and-cut algorithms such as developed by Abbas and Chaabane (2006) and Chaabane, Brahmi, and Ramdan (2012) where the goal is to optimize a linear function over the integer efficient set of an MOLP. In these algorithms, special constraints are added at each iteration to exclude non-optimal integer efficient points from getting in the way of further progress. But the cost of this, unless an optimal integer efficient point is found early, is a build-up of constraints that will eventually cause the implementation to become untenable. But by using the methods described herein, it should be possible to break the integer MOLP into parts so as to avoid any serious constraint bottlenecks from occurring. As another possibility, consider the algorithm by Kirlik and Sayin (2014) for computing all nondominated criterion vectors of a multiple objective integer programming problem. By utilizing the techniques of the paper, it seems that it should be possible to reduce the wall-clock times of many of the run times reported in that paper. There may be other examples.

Acknowledgments

The authors wish to thank three anonymous referees for their helpful comments.

References

Abbas, M., & Chaabane, D. (2006). Optimizing a linear function over an integer efficient set. *European Journal of Operational Research, 174*(3), 1140–1161.
Alves, M. J., Antunes, C. H., & Climaco, J. (2015). Interactive MOLP explorer – a graphical-based computational tool for teaching and decision support in multi-objective linear programming models. *Computer Applications of Engineering Education, 23*(2), 314–326.
Antunes, C. H., Alves, M. J., & Climaco, J. (2016). *Multiobjective linear and integer programming*. Spinger.
Armand, P., & Malivert, C. (1991). Determination of the efficient set in multiobjective linear programming. *Journal of Optimization Theory and Applications, 70*(3), 467–489.
Benson, H. P. (1991). An all-linear programming relaxation algorithm for optimizing over the efficient set. *Journal of Global Optimization, 1*(1), 83–104.
Benson, H. P., & Sun, E. (2000). Outcome space partition of the weight set in multiobjective linear programming. *Journal of Optimization Theory and Applications, 105*(1), 17–36.
Benson, H. P., & Sun, E. (2002). A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program. *European Journal of Operational Research, 139*(1), 26–41.
Chaabane, D., Brahmi, B., & Ramdan, Z. (2012). The augmented weighted Tchebychev norm for optimizing a linear function over an integer efficient set of a multicriteria linear program. *International Transactions in Operations Research, 19*(4), 531–545.
Climaco, J. C. M., & Antunes, C. H. (1989). Implementation of a user friendly software package – a guided tour of TRIMAP. *Mathematical and Computer Modelling, 12*(10–11), 1299–1309.
Dauer, J. P. (1987). Analysis of the objective space in multiple objective programming. *Journal of Mathematical Analysis and Applications, 126*(2), 579–593.
Dauer, J. P. (1993). On degeneracy and collapsing in the construction of the set of objective values in a multiple objective linear program. *Annals of Operations Research, 46–47*(2), 279–292.
Ecker, J. G., & Kouada, I. A. (1978). Finding all efficient extreme points for multiple objective linear programs. *Mathematical Programming, 14*(1), 249–261.
Ecker, J. G., & Song, J. H. (1994). Optimizing a linear function over the efficient set. *Journal of Optimization Theory and Applications, 83*(3), 541–563.
Ehrgott, M., Löhne, A., & Shao, L. (2012). A dual variant of Benson's "outer approximation algorithm" for multiple objective linear programming. *Journal of Global Optimization, 52*(4), 757–778.
Ehrgott, M., & Shao, L. (2008). Approximately solving multiobjective linear programmes in objective space and an application in radiotherapy treatment planning. *Mathematical Methods of Operational Research, 68*(2), 257–276.
Evans, J. P., & Steuer, R. E. (1973). A revised simplex method for multiple objective programs. *Mathematical Programming, 5*(1), 54–72.
Hwang, C. L., & Masud, A. S. M. (1979). Multiple objective decision making – methods and applications: A state-of-the-art survey. *Lecture Notes in Economics and Mathematical Systems*: L164. Berlin: Springer Verlag.
Isermann, H. (1977). The enumeration of the set of all efficient solutions for a linear multiple objective program. *Operational Research Quarterly, 28*(3), 711–725.
Kirlik, G., & Sayin, S. (2014). A new algorithm for generating all nondominated solutions for multiobjective discrete optimization problems. *European Journal of Operational Research, 232*(3), 479–488.
Löhne, A., & Weissing, B. (2017). The vector linear program solver Bensolve – notes on theoretical background. *European Journal of Operational Research, 260*(3), 807–813.
Miettinen, K., Eskelainen, P., Ruiz, F., & Luque, M. (2010). Nautilus method: an interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research, 206*(2), 426–434.

Miettinen, K., Podkopaev, D., Ruiz, F., & Luque, M. (2015). A new preference handling technique for interactive multiobjective optimization without trading-off. *Journal of Global Optimization, 4*(63), 633–652.

Rudloff, B., Ulus, F., & Vanderbei, R. (2017). A parametric simplex algorithm for linear vector optimization problems. *Mathematical Programming, 163*(1–2), 213–242.

Shao, L., & Ehrgott, M. (2008). Approximating the nondominated set of an MOLP by approximately solving its dual problem. *Mathematical Methods of Operations Research, 68*(3), 469–492.

Steuer, R. E. (2017). *ADBASE: A self-contained multiple objective linear programming solver for all efficient extreme points and unbounded efficient edges*. Terry College of Business, University of Georgia, Athens, Georgia. Version 17.1

Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation, and application*. New York: John Wiley & Sons.

Steuer, R. E., & Piercy, C. A. (2005). A regression study of the number of efficient extreme points in multiple objective linear programming. *European Journal of Operational Research, 162*(2), 484–496.

Wallenius, J., Dyer, J. S., Fishburn, P. C., Steuer, R. E., Zionts, S., & Deb, K. (2008). Multiple criteria decision making, multiattribute utility analysis: Recent accomplishments and what lies ahead. *Management Science, 54*(7), 1336–1349.

Yamamoto, Y. (2002). Optimization over the efficient set: overview. *Journal of Global Optimization, 22*(1), 285–317.

Zeleny, M. (1974). *Linear multi-objective programming*. New York: Springer Verlag.