Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

Modified interactive Chebyshev algorithm (MICA) for convex multiobjective programming

Mariano Luque [a,*], Francisco Ruiz [a], Ralph E. Steuer [b]

[a] University of Málaga, Calle Ejido 6, 29071 Málaga, Spain
[b] University of Georgia, Terry College of Business, Athens, GA 30602-6253, USA

ARTICLE INFO

Article history:
Received 6 February 2009
Accepted 7 November 2009
Available online 22 November 2009

Keywords:
Multiobjective programming
Interactive procedures
Tchebycheff method
Reference point methods
Aspiration criterion vectors
Reservation levels

ABSTRACT

In this paper, we describe an interactive procedural algorithm for convex multiobjective programming based upon the Tchebycheff method, Wierzbicki's reference point approach, and the procedure of Michalowski and Szapiro. At each iteration, the decision maker (DM) has the option of expressing his or her objective-function aspirations in the form of a reference criterion vector. Also, the DM has the option of expressing minimally acceptable values for each of the objectives in the form of a reservation vector. Based upon this information, a certain region is defined for examination. In addition, a special set of weights is constructed. Then with the weights, the algorithm of this paper is able to generate a group of efficient solutions that provides for an overall view of the current iteration's certain region. By modification of the reference and reservation vectors, one can "steer" the algorithm at each iteration. From a theoretical point of view, we prove that none of the efficient solutions obtained using this scheme impair any reservation value for convex problems. The behavior of the algorithm is illustrated by means of graphical representations and an illustrative numerical example.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

When facing a real decision problem, a decision maker (DM) must often deal with several conflicting objectives. In such cases, the traditional optimization approach, in which a single objective is optimized subject to a given set of constraints, is no longer applicable. Instead, a multiobjective model is to be formulated and solved. Because of the rarity of solutions that optimize all objectives simultaneously, multiobjective programming utilizes efficient solutions. These are solutions from which no objective can be improved without deteriorating at least one of the others. Being "trade-off efficient" in this way, the set of all efficient solutions is precisely the set of all candidates for optimality. But as for which is to be optimal, this is for the DM to decide, and this often involves a contemplative process.

As outlined in Hwang and Masud (1979), procedures for solving multiobjective decision problems can be grouped into three categories depending upon whether preference information is elicited from the DM before, after, or during the solution process. In the "before" category are *a priori* methods. In these methods, after eliciting information from the DM, an optimization problem is solved to compute a solution. A difficulty of a priori methods is that it is hard to know in advance with sufficient accuracy the information required by the optimization problem for it to produce a *final* solution (an optimal solution or a solution close enough to one to qualify in its stead). Also, with these methods, there is the question about being able to recognize a final solution even when confronted with one without knowing more about the efficient set. In the "after" category are *a posteriori* or, as called by Cohon (1985), *generating* methods. In these methods, a comprehensive set of efficient solutions (or in the best case the whole efficient set) is generated and shown to the DM. Then, the DM is to choose his or her most preferred solution from the set. The drawback of these methods is that usually a great number of efficient solutions has to be generated, and it can be extremely hard for the DM to manage all of the information.

In the "during" category are interactive procedures. Interactive procedures are designed to overcome the difficulties encountered in a priori and a posteriori methods. In interactive procedures, phases of information elicitation are interleaved with phases of computation. In the beginning, the information exchanged between the DM and procedure is general, but then becomes more local in character as the procedure continues. In this way, interactive procedures have two main features: (a) they help a DM learn about a problem while solving it, and (b) they put to work iteratively any new insights gained during the solution process to help the DM navigate to a final solution. Prominent interactive procedures include STEM by Benayoun et al. (1971), the Zionts–Wallenius

* Corresponding author.
E-mail addresses: mluque@uma.es (M. Luque), rua@uma.es (F. Ruiz), rsteuer@uga.edu (R.E. Steuer).

0377-2217/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.11.011

procedure (1976), Wierzbicki's reference point method (1980), interactive goal programming (Spronk (1981) for instance), the Tchebycheff method of Steuer and Choo (1983), Pareto Race by Korhonen and Wallenius (1988), and the bi-reference procedure of Michalowski and Szapiro (1992). Others that can be mentioned are the light beam search method of Jaszkiewicz and Słowinski (1999), Miettinen's NIMBUS (1999), and the normal vector identification approach of Yang and Li (2002).

The real forces behind so many interactive procedures have been the many different types of problems that lend themselves to multiple criteria analysis and the fact, due to the differences in the procedures, that the procedure to use in a given instance is typically application and decision-making style of the user dependent. Consequently, there has been a need for today's many interactive procedures. From the cognitive point of view, interactive procedures basically differ from one another in the way information is asked of the DM at each iteration. Four styles can be differentiated. One asks the DM to specify local tradeoffs or marginal rates of substitution between the objectives. Another asks the DM to select from several solution candidates at each iteration. A third asks the DM to specify target or aspiration levels for the different objectives, and a fourth asks the DM to classify the objectives, for instance, as to which are to be improved, which are permitted to become relaxed, and which are to be held at their current levels on the next iteration. Naturally, researchers have thought of consolidating procedures. There have been attempts to create global formulations such as by Gardiner and Steuer (1994) and Luque et al. (in press), and also to design combined implementations such as by Antunes et al. (1992) and Caballero et al. (2002b). But mostly these have involved *procedure-switching*. By procedure-switching, we mean giving to the user the option to switch to any procedure on any iteration. For example, a user may choose to start with one procedure on the first iteration, switch to another for the second iteration, switch to a third for the third iteration, and so forth. But this significantly increases the cognitive burden when, if anything, we should be going in the opposite direction.

Primarily motivated by the Tchebycheff method of Steuer and Choo (1983), the reference point method of Wierzbicki (1980), and the bi-reference point procedure of Michalowski and Szapiro (1992), we present the modified interactive Chebyshev algorithm (MICA) of this paper.[1] In one sense, MICA is similar to the Tchebycheff method in that it conducts multiple probing and uses "oversampling/filtering" techniques when developing each iteration's group of solutions to be presented to the DM. But MICA departs from the Tchebycheff procedure in the way the neighborhoods of the efficient set that are to be examined at each iteration are defined, shifted, contracted, and sampled. In the Tchebycheff method, the neighborhoods are defined, shifted, and contracted by manipulating subsets in weight space. Unfortunately, being in weight space, these manipulations are not very intuitive, and there is no intention of pursuing them further here. Rather, the neighborhoods to be explored in MICA are designed to be controlled by an iteratively adjustable aspiration criterion vector and an iteratively adjustable reservation vector. In this way, the aspiration and reservation vectors of a given iteration define the "frame" that contains the neighborhood of the efficient set to be explored on that iteration. And by adjusting the vectors, the neighborhoods can be shifted and contracted one iteration to the next in search of a final solution.

With regard to the sampling of the neighborhoods, it is to be pointed out that MICA possesses a special technical feature. The technical feature involves the way in which the weight vectors used to sample the neighborhoods are generated. As shown, they are specially generated to ensure that no reservation level of any objective is ever violated during the sampling process without ever having to include any reservation level in the constraint set of the program used to carry out the sampling operations.

[1] We use the term Chebyshev to stress MICA's relationship to, yet differences from, the Tchebycheff method of 1983.

There is also another item that arises in the paper. It stems from the number of procedures that currently comprise the field of interactive multiobjective programming. Even though the possession of many procedures is generally considered a strength of interactive multiobjective programming, things could well be different in the future with the field ultimately becoming dominated by a much smaller number of procedures, each capturing the power of several of today's procedures without incurring a cognitive burden greater than any of the procedures singly. As we will see, in taking a step in this direction, MICA shows that at least some of this is possible.

The paper is organized as follows. Section 2 sets forth the problem to be addressed and reviews some background concepts. Because MICA draws upon features from the Tchebycheff method, Wierzbicki's aspiration criterion vector method, and the bi-reference procedure of Michalowski and Szapiro, these procedures are overviewed in Section 3. The basic philosophy of MICA is outlined in Section 4 along with details about how weight vectors can be generated so not to impair any reservation levels in the sampling process. A step-by-step description of MICA is given in Section 5. An example illustrating the operation of MICA comprises Section 6, and Section 7 brings the paper to a close with concluding remarks. A proof of the main theorem of the paper is given in Appendix A.

2. Formulation and background concepts

MICA is designed for the solution of the (convex) multiobjective problem

$$\begin{array}{ll} \max & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \\ \text{subject to } \mathbf{x} \in X & \end{array} \tag{1}$$

in which all f_i are continuous and (of course) concave, and $X \subset \mathbb{R}^n$, the feasible region in *decision space*, is closed, bounded and (of course) convex. Being a multiobjective problem, there is also $Z \subset \mathbb{R}^k$, the feasible region in *criterion space*, where $Z = \{\mathbf{z} = \mathbf{f}(\mathbf{x}) | \mathbf{x} \in X\}$. In decision space, $\bar{\mathbf{x}} \in X$ is *efficient* if and only if there does not exist another $\mathbf{x} \in X$ such that $\mathbf{f}(\mathbf{x}) \geqslant \mathbf{f}(\bar{\mathbf{x}})$ and $\mathbf{f}(\mathbf{x}) \neq \mathbf{f}(\bar{\mathbf{x}})$. Then, in criterion space, criterion vector $\tilde{\mathbf{z}} \in Z$ is *nondominated* if and only if there exists an $\tilde{\mathbf{x}} \in X$ such that $\tilde{\mathbf{z}} = \mathbf{f}(\tilde{\mathbf{x}})$ and $\tilde{\mathbf{x}}$ is efficient. The set of all efficient points is called the *efficient set* and is designated E. The set of all nondominated criterion vectors is called the *nondominated set*. Also, in decision space, $\bar{\mathbf{x}} \in X$ is *weakly* efficient if and only if there does not exist another $\mathbf{x} \in X$ such that $\mathbf{f}(\mathbf{x}) > \mathbf{f}(\bar{\mathbf{x}})$. Then, in criterion space, criterion vector $\tilde{\mathbf{z}} \in Z$ is *weakly* nondominated if and only if there exists an $\tilde{\mathbf{x}} \in X$ such that $\tilde{\mathbf{z}} = \mathbf{f}(\tilde{\mathbf{x}})$ and $\tilde{\mathbf{x}}$ is weakly efficient. Note that the set of all weakly efficient points subsumes all efficient points.

Ideal and *nadir* criterion vectors, $\mathbf{z}^*$ and $\mathbf{z}^{nad}$, whose components are given by

$$z_i^* = \max_{\mathbf{x} \in E} f_i(\mathbf{x}) \quad (i = 1, \ldots, k) \tag{2}$$

$$z_i^{nad} = \min_{\mathbf{x} \in E} f_i(\mathbf{x}) \quad (i = 1, \ldots, k) \tag{3}$$

are often of interest in multiobjective programming. One use of them, should $\mathbf{z}^{nad}$ be available, would be to form the intervals $[z_i^{nad}, z_i^*]$, $1 \leqslant i \leqslant k$, so as to frame a problem in the sense that no optimal solution will have any component outside its specified interval. Unfortunately, in many problems, nadir criterion vectors with all components known to be correct are difficult to obtain. While there is now the special algorithm by Alves and Costa (2009) for exactly computing nadir criterion values in multiobjec-

Download English Version:

https://daneshyari.com/en/article/481965

Download Persian Version:

https://daneshyari.com/article/481965

Daneshyari.com