Applied Soft Computing 12 (2012) 3526–3538

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

A regularity model-based multiobjective estimation of distribution algorithm with reducing redundant cluster operator

Yong Wang[a,b], Jian Xiang[a,b,*], Zixing Cai[a,b]

[a] School of Information Science and Engineering, Central South University, Changsha 410083, PR China
[b] Hunan Engineering Laboratory for Advanced Control and Intelligent Automation, Changsha 410083, PR China

ARTICLE INFO

Article history:
Received 6 July 2011
Received in revised form 24 May 2012
Accepted 6 June 2012
Available online 20 July 2012

Keywords:
Estimation of distribution algorithm
Multiobjective optimization
Modeling
Reducing redundant cluster operator

ABSTRACT

A regularity model-based multiobjective estimation of distribution algorithm (RM-MEDA) has been proposed for solving continuous multiobjective optimization problems with variable linkages. RM-MEDA is a kind of estimation of distribution algorithms and, therefore, modeling plays a critical role. In RM-MEDA, the population is split into several clusters to build the model. Moreover, the fixed number of clusters is recommended in RM-MEDA when solving different kinds of problems. However, based on our experiments, we find that the number of clusters is problem-dependent and has a significant effect on the performance of RM-MEDA. Motivated by the above observation, in this paper we improve the clustering process and propose a *r*educing *r*edundant *c*luster *o*perator (RRCO) to build more precise model during the evolution. By combining RRCO with RM-MEDA, we present an improved version of RM-MEDA, named IRM-MEDA. In this paper, we also construct four additional continuous multiobjective optimization test instances. The experimental results have shown that IRM-MEDA outperforms RM-MEDA in terms of efficiency and effectiveness. In particular, IRM-MEDA performs on average 31.67% faster than RM-MEDA.

1. Introduction

Many optimization problems involve not one but several objectives which should be optimized simultaneously. This kind of problems is considered as multiobjective optimization problems (MOPs). In this paper, we consider the following continuous MOPs:

$$\text{minimize} \quad \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) \tag{1}$$

where $\vec{x} = (x_1, \ldots, x_n) \in X \subseteq R^n$ is the decision vector, X is the decision space, $\vec{y} \in Y \subseteq R^m$ is the objective vector, and Y is the objective space.

There are some basic definitions in multiobjective optimization, which are introduced as follows.

Definition 1. Given two decision vectors $\vec{a} = (a_1, \ldots, a_n)$ and $\vec{b} = (b_1, \ldots, b_n)$, if $\forall i \in \{1, \ldots, m\}$, $f_i(\vec{a}) \le f_i(\vec{b})$ and $\exists j \in \{1, \ldots, m\}$, $f_j(\vec{a}) < f_j(\vec{b})$, we say $\vec{a}$ Pareto dominates $\vec{b}$, denoted as $\vec{a} \prec \vec{b}$.

Definition 2. A decision vector $\vec{x} \in X$ is called Pareto optimal solution if there does not exist another decision vector $\vec{x}' \in X$ such that $\vec{x}' \prec \vec{x}$.

Definition 3. The *Pareto set* (*PS*) is the set of all the Pareto optimal solutions:

$$PS = \{\vec{x} \in X | \neg \exists \vec{x}' \in X, \vec{x}' \prec \vec{x}\} \tag{2}$$

The solutions in the *PS* are also called nondominated solutions.

Definition 4. The *Pareto front* (*PF*) is the set of the objective vectors of all the Pareto optimal solutions:

$$PF = \{\vec{f}(\vec{x}) | \vec{x} \in PS\} \tag{3}$$

For MOPs, in most cases, we cannot find a single solution to optimize all the objectives at the same time. Therefore, we have to balance them and find a set of optimal tradeoffs, i.e., *Pareto set* (*PS*) in the decision space and *Pareto front* (*PF*) in the objective space, respectively. Since evolutionary algorithms (EAs) deal with a group of candidate solutions simultaneously, it seems to be natural to use EAs for finding a group of Pareto optimal solutions when solving MOPs. Vector evaluation genetic algorithm (VEGA), introduced by Schaffer [1] in 1980s, is the first actual implementation of EAs to solve MOPs. After that, a considerable number of multiobjective evolutionary algorithms (MOEAs) have been proposed due to increasing interest in solving MOPs by EAs.

The development of MOEAs can be briefly divided into three generations [2,3]. In the first generation of MOEAs, Pareto ranking and fitness sharing are the most common techniques adopted by MOEAs. There are some paradigms in this generation, for

* Corresponding author at: School of Information Science and Engineering, Central South University, Changsha 410083, PR China.
E-mail addresses: ywang@csu.edu.cn (Y. Wang), xiangjiancsu@gmail.com (J. Xiang).

1568-4946/$ – see front matter
http://dx.doi.org/10.1016/j.asoc.2012.06.008

example: nondominated sorting genetic algorithm (NSGA), proposed by Srinivas and Deb [4], is based on several layers of classifications of the individuals as suggested by Goldberg [5] and uses crowding distance to maintain the diversity of the population. Niched-Pareto genetic algorithm (NPGA), proposed by Horn et al. [6], employs tournament selection based on Pareto dominance and fitness sharing to keep the diversity. Fonseca and Fleming [7] introduced a multiobjective genetic algorithm (MOGA).

The second generation of MOEAs is characterized by the elitism preservation, which usually stores the nondominated individuals into a predefined archive (also called external population). It is necessary to note that incorporating the elitism into MOEAs can facilitate the convergence of the population. Zitzler and Thiele [8] proposed strength Pareto EA (SPEA), which uses an archive to store the nondominated solutions found so far and adopts clustering to prune the archive if the number of nondominated individuals in the archive exceeds a predefined value. Zitzler et al. [9] also proposed an improved version of SPEA, referred as SPEA2. Compared with SPEA, SPEA2 has the following three properties: (1) a new fitness assignment strategy, (2) a density estimation technique, and (3) a novel archive truncation method. Knowles and Corne [10] presented Pareto archive evolutionary strategy (PAES), which uses (1 + 1)-ES to generate offspring. In PAES, the offspring is compared with the parent and the previously archived nondominated individuals for survival. Moreover, PAES divides the objective space into grids, the aim of which is to maintain the diversity of the population. Inspired by PAES, Corne et al. further developed PESA [11] and PESA-II [12]. Deb et al. [13] proposed an improved version of NSGA, called NSGA-II, by incorporating a fast nondominated sorting approach and a crowding-comparison approach.

In the current research, which belongs to the third generation of MOEAs, some new dominance concepts other than traditional Pareto dominance have been introduced. For instance, Laumanns et al. [14] introduced ε-dominance. Hernández-Díaz et al. [15] proposed an adaptive ε-dominance, which is an improvement of the original ε-dominance [14]. Ben Said et al. [16] proposed *r*-dominance for interactive evolutionary multi-criteria decision making. Brockoff and Zitzler [17] proposed a local dominance scheme to reduce objective dimensionality. In addition, some researchers combined traditional weight vector based techniques with EAs to deal with MOPs [18–21]. Recently, Zhang and Li [22] proposed a novel MOEA based on decomposition, called MOEA/D, which converts MOPs into a set of scalar optimization subproblems. Moreover, MOEA/D utilizes the neighbor information to produce offspring and optimize the subproblems simultaneously.

Many attempts have also been made to improve the performance of MOEAs by making use of different kinds of EAs as well as swarm intelligence. For example, Coello Coello et al. [23] incorporated Pareto dominance into particle swarm optimization for solving MOPs. Li and Zhang [24] proposed a new version of MOEA/D [22] based on differential evolution. Igel et al. [25] developed a variant of covariance matrix adaptation evolution strategy (CMA-ES) [26] for multiobjective optimization. Ghoseiria and Nadjari [27] presented an algorithm based on multiobjective ant colony optimization to solve the bi-objective shortest path problem. Jamuna and Swarup [28] proposed a multiobjective biogeography based optimization algorithm to design optimal placement of phasor measurement units. Zhang [29] proposed an immune optimization algorithm for dealing with constrained nonlinear multiobjective optimization problems.

Recently, indicator-based MOEAs have also been actively researched in the community of evolutionary multiobjective optimization [30,31].

It can be induced from the Karush–Kuhn–Tucker condition that the *PS* of a continuous MOP is a $(m-1)$-dimensional piecewise continuous manifold in the decision space [32,33], where m is the number of objectives. Thus, for the continuous biobjective optimization problems (i.e., $m = 2$), the *PS* is a piecewise continuous curve; and for the continuous triobjective optimization problems (i.e., $m = 3$), the *PS* is a piecewise continuous 2-D surface.

Based on the above regularity, Zhang et al. [34] proposed a regularity model-based multiobjective estimation of distribution algorithm, referred as RM-MEDA. As a kind of estimation of distribution algorithms (EDAs) [35], RM-MEDA employs the $(m-1)$-dimensional local principal component analysis ($(m-1)$-D local PCA) [36] to build the model of the *PS* in the decision space. The $(m-1)$-D local PCA is a locally linear approach to nonlinear dimension reduction, which can construct local models, each pertaining to a different disjoint region of the data space. In RM-MEDA, firstly, the $(m-1)$-D local PCA divides the population into K (K is a constant integer) disjoint clusters and computes the central point and principal component of each cluster. Afterward, one model is built based on the corresponding central point and principal component for each cluster. The primary aim of modeling in RM-MEDA is to approximate one of the pieces of the *PS* by making use of the solutions in one cluster. Ideally, if the number of clusters K is equal to the number of the pieces of the *PS*, each piece of the *PS* can be approximated by one cluster. In this case, a precise model may be built and the performance of RM-MEDA may be excellent. However, if the number of clusters K is not equal to the number of the pieces of the *PS*; needless to say, the model is not precise.

Since we have no *priori* knowledge about the number of the pieces of the *PS* for a MOP at hand, it is very difficult to determine a reasonable value for K. Moreover, the setting of K is usually problem-dependent. In particular, based on our experiments, this parameter has a significant effect on the performance of RM-MEDA. Since K is fixed to 5 in RM-MEDA, this setting might not be very effective for different kinds of MOPs. In order to overcome the above drawback of RM-MEDA, we design a *r*educing *r*edundant *c*luster *o*perator (RRCO) to enhance the modeling precision of RM-MDEA. By integrating RRCO with RM-MEDA, IRM-MEDA is derived. Extensive experiments have been conducted to compare IRM-MEDA with its predecessor RM-MEDA on a set of biobjective and triobjective test instances with variable linkages (note that variable linkages reflect the interactions among the variables). The experimental results verify that the efficiency and effectiveness of RM-MEDA can be significantly improved by RRCO.

The rest of the paper is organized as follows. Section 2 briefly reviews RM-MEDA. The drawback of modeling in RM-MEDA is discussed in Section 3. Section 4 presents the details of RRCO. IRM-MEDA is described in Section 5. The experimental results are reported in Section 6. Finally, Section 7 concludes this paper.

2. Review of RM-MEDA

2.1. Framework

During the evolution, RM-MEDA maintains:

- a population P_t of N individuals: $P_t = \{\vec{x}_1, \ldots, \vec{x}_N\}$, where t is the generation number;
- their $\vec{f}$-values : $\vec{f}(\vec{x}_1), \ldots, \vec{f}(\vec{x}_N)$.

RM-MEDA is implemented as follows: