Applied Soft Computing 35 (2015) 541–557

Contents lists available at ScienceDirect

Applied Soft Computing

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

A self-adaptive multi-objective harmony search algorithm based on harmony memory variance

Xiangshan Dai, Xiaofang Yuan*, Zhenjun Zhang

College of Electrical and Information Engineering, Hunan University, Changsha 410082, China

ARTICLE INFO

Article history:
Received 5 December 2014
Received in revised form 20 May 2015
Accepted 12 June 2015
Available online 9 July 2015

Keywords:
Multi-objective optimization
Pareto front (PF)
Harmony search (HS)
Self-adaptive parameter setting
Self-adaptive bandwidth

ABSTRACT

Although harmony search (HS) algorithm has shown many advantages in solving global optimization problems, its parameters need to be set by users according to experience and problem characteristics. This causes great difficulties for novice users. In order to overcome this difficulty, a self-adaptive multi-objective harmony search (SAMOHS) algorithm based on harmony memory variance is proposed in this paper. In the SAMOHS algorithm, a modified self-adaptive bandwidth is employed, moreover, the self-adaptive parameter setting based on variation of harmony memory variance is proposed for harmony memory considering rate (HMCR) and pitch adjusting rate (PAR). To solve multi-objective optimization problems (MOPs), the proposed SAMOHS uses non-dominated sorting and truncating procedure to update harmony memory (HM). To demonstrate the effectiveness of the SAMOHS, it is tested with many benchmark problems and applied to solve a practical engineering optimization problem. The experimental results show that the SAMOHS is competitive in convergence performance and diversity performance, compared with other multi-objective evolutionary algorithms (MOEAs). In the experiment, the impact of harmony memory size (HMS) on the performance of SAMOHS is also analyzed.

1. Introduction

The majority of real-world optimization problems consist of multiple objectives which usually conflict with each other, the so called multi-objective optimization problems (MOPs). MOPs generate a set of optimal solutions instead of a single solution, largely known as Pareto optimal set [1]. The efficiency to find multiple solutions of most classical optimization algorithms is low since these optimization algorithms usually achieve only one Pareto optimal solution in one run, therefore, these algorithms have to be run many times to acquire the Pareto optimal set.

In the last few decades, evolutionary algorithms have attracted great attention among researchers to solve MOPs, many multi-objective evolutionary algorithms (MOEAs) have been developed, such as NSGA-II [1], PESA-II [2], SPEA2 [3] and M-PAES [4]. The reason for the fast development of MOEAs is that these algorithms are able to find multiple Pareto optimal solutions in just one single execution. Harmony search (HS) algorithm, inspired by the music improvisation process, is a new kind of meta-heuristic algorithm [5]. The HS algorithm has been applied to solve a wide variety of optimization problems [6–9] as it has several advantages including: (a) HS has fewer control parameters and the initial value setting of decision variables is unnecessary and (b) HS is easy to be implemented and understood.

Recently, there are several attempts to extend the HS to solve MOPs. Literature [10] has used HS to solve multi-objective optimization of time-cost trade-off, while this algorithm has little diversity of non-dominated solutions as it only uses dominated-based comparison and ignores diversity comparison. The multi-objective harmony search algorithm proposed by Sivasubramani et al. [11] is able to converge to Pareto optimal solutions, however it has low convergence rate and needs a large number of iterations. This low convergence rate can be attributed to the use of an improved harmony search (IHS) algorithm [12] whose convergence rate is far from fast. Two detailed proposals for applying basic HS algorithm to solve MOPs have been proposed by Ricart et al. [13], however these algorithms show poor performance in terms of convergence and diversity. Pavelski et al. [14] have investigated four variants of HS algorithm for solving MOPs. However, these four variants cannot generate true and well-distributed Pareto optimal solutions. The poor performance of all aforementioned multi-objective HS algorithms can be attributed to the fact that these algorithms have poor search abilities for MOPs. Moreover, the control parameters of these multi-objective HS algorithms need to be set by users according to experience and problem characteristics. As a result, this causes great burden to new users, and

* Corresponding author. Tel.: +86 13873195923.
E-mail address: yuanxiaofang126@126.com (X. Yuan).

http://dx.doi.org/10.1016/j.asoc.2015.06.027
1568-4946/

hinders the application and development of multi-objective HS algorithm.

In view of the shortcomings of above-mentioned multi-objective HS algorithms, a self-adaptive multi-objective harmony search (SAMOHS) algorithm based on harmony memory (HM) variance is proposed in this paper. The variation of the population variance has great influence on the explorative ability of the HS algorithm [15], and the variance of HM reflects the diversity of HM to some extent [16]. Inspired by these researches, a novel self-adaptive mechanism based on the variation of the HM variance is employed in SAMOHS. The proposed self-adaptive mechanism has three main improvements, they are: (a) each decision variable has its own control parameters, which are updated adaptively during the search process; (b) a self-adaptive parameter setting based on variation of HM variance is employed for harmony memory considering rate (*HMCR*) and pitch adjusting rate (*PAR*); (c) a modified self-adaptive bandwidth (*bw*) is proposed for HS. By using the self-adaptive mechanism, the SAMOHS has better adaptability and robustness. In addition, the global and local search abilities of SAMOHS are improved, and the burden of parameters setting is alleviated. For solving MOPs, a non-dominated sorting [1] and truncating procedure [3] are utilized to update the HM effectively and to preserve the diversity of non-dominated solutions in HM.

The rest of this paper is organized as follows. In Section 2, basic concepts of MOPs are briefly presented. Section 3 describes the basic HS algorithm. In Section 4, the motivations and framework of the proposed SAMOHS algorithm are presented. The experimental results are given in Section 5. Finally, the conclusion is given in Section 6.

2. Concepts of MOPs

In this section, basic concepts on multi-objective optimization relevant to this paper are presented. The mathematical model of MOPs is formulated as [17]:

$$\begin{cases} \text{minimize} & \boldsymbol{f}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})\}^T \\ \text{subject to} & \boldsymbol{x} \subset S \subset R^n \end{cases} \quad (1)$$

with $k \geq 2$ conflicting objective functions $f_i : S \longrightarrow R$. Here $\boldsymbol{f}(\boldsymbol{x})$ denotes the vector of objective function value to be optimized, and the decision vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ belongs to the search space S defined usually with constraint functions.

Consider two decision vectors $\boldsymbol{g} = (g_1, g_2, \ldots, g_n)^T \in S$ and $\boldsymbol{h} = (h_1, h_2, \ldots, h_n)^T \in S$ for MOPs in Eq. (1), the relation between $\boldsymbol{g}$ and $\boldsymbol{h}$ can be mathematically described as [18]:

$$\begin{cases} \boldsymbol{g} \prec\prec \boldsymbol{h}, & \text{if} \quad \forall i \in \{1, 2, \ldots, k\}, g_i < h_i \\ \boldsymbol{g} \prec \boldsymbol{h}, & \text{if} \quad \forall i \in \{1, 2, \ldots, k\}, g_i \leq h_i \cap \exists i \in \{1, 2, \ldots, k\}, g_i < h_i \\ \boldsymbol{g} \nprec \boldsymbol{h}, & \text{if} \quad \exists i \in \{1, 2, \ldots, k\}, g_i > h_i \end{cases} \quad (2)$$

where $\prec\prec, \prec$ and $\nprec$ denote strong dominance, dominance and not dominated, respectively.

A decision vector $\boldsymbol{x}^\star \in S$ for this problem is Pareto optimal solution if there is no $\boldsymbol{x} \in S$ satisfies with $\boldsymbol{f}(\boldsymbol{x}) \prec \boldsymbol{f}(\boldsymbol{x}^\star)$. An objective vector $\boldsymbol{z}^\star = \boldsymbol{f}(\boldsymbol{x}^\star)$ is called Pareto optimal if corresponding vector $\boldsymbol{x}^\star$ is Pareto optimal solution. The set of all Pareto optimal solutions $\boldsymbol{x}^\star \in S$ is called Pareto optimal set, so the Pareto optimal set is also a set of non-dominated solutions. For a given MOP and its corresponding Pareto optimal set $P^\star$, the optimal Pareto front (PF) is defined as $PF^\star = \{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in P^\star\}$.

The main purpose of multi-objective optimization is to obtain the Pareto optimal set and the optimal PF. Recently, an exact algorithm in relation to a multi-objective maintenance problem, able to describe the entire PF in a very short computational time, has been developed [19]. However, in most cases, finding the optimal PF that contains all these points is not realistic [20]. Therefore, it is necessary to find an approximated PF that contains points as evenly spread and as close as possible with respect to the optimal PF in case of without any further information. In order to acquire the approximated PF, the common method is to calculate the feasible domain S and its corresponding $\boldsymbol{f}(S)$ by applying specific algorithms.

3. Basic HS algorithm

HS algorithm is a simple but powerful memory-based stochastic search technique for solving global optimization problems. In the basic HS algorithm, each solution is expressed by a n-dimension real vector and typically named a “harmony” [5]. An initial population of harmony vectors stored in a HM is randomly generated. A new harmony is then improvised based on all harmonies stored in the HM by applying an improvisation scheme. Afterwards, the newly generated harmony is compared with the worst harmony in HM and replaces the worst one if it has a better fitness value. The algorithm repeats until a predefined stopping condition is met.

A typical optimization problem is formulated as follows:

$$\text{Minimize} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad x_j \in [LB_j, UB_j], \quad j = 1, 2, \ldots, n \quad (3)$$

where $f(\boldsymbol{x})$ is the objective function; $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is the set of n-dimension decision variable; LB_j and UB_j denote the lower and upper limits for x_j, respectively.

The main procedure of the basic HS algorithm for optimization problem in Eq. (3) is described as follows:

Step 1: Initialize the algorithm parameters. The following parameters of HS algorithm are set: harmony memory size (*HMS*), *HMCR*, *bw*, *PAR*, and the number of improvisations (*NI*).

Step 2: Initialize the harmony memory. In this step, the HM matrix is filled with *HMS* randomly generated harmony vectors:

$$HM = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_{HMS} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,n} \\ \vdots & \vdots & \ldots & \vdots & \ldots & \vdots \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,n} \\ \vdots & \vdots & \ldots & \vdots & \ldots & \vdots \\ x_{HMS,1} & x_{HMS,2} & \cdots & x_{HMS,j} & \cdots & x_{HMS,n} \end{bmatrix} \quad (4)$$

Step 3: Improvise a new harmony. A new harmony vector $X_{new} = (x_{new,1}, x_{new,2}, \ldots, x_{new,n})$ is generated by applying three rules: (a) memory consideration, (b) pitch adjustment and (c) random selection. Generating a new harmony is typically named “improvisation” [21]. The improvisation procedure works as follows:

Algorithm 1. The procedure of improvisation in HS

for $j = 1, \ldots, n$ **do**
  if $rand() \leq HMCR$ **then**
    $x_{new,j} \in \{x_{1,j}, x_{2,j}, \ldots, x_{HMS,j}\}$//memory consideration
    if $rand() \leq PAR$ **then**
    $x_{new,j} = x_{new,j} \pm rand() \times bw$//pitch adjustment
  end if
  else
  $x_{new,j} = LB_j + rand() \times (UB_j - LB_j)$//random selection
  end if
end for

where *rand*() is a random number generated from an uniform distribution of [0,1].

Step 4: Update the harmony memory. If the newly generated harmony X_{new} is better than the worst harmony stored in the HM, judged by their objective function values, the HM will be updated.