软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(7):1571−1588 [doi: 10.3724/SP.J.1001.2013.04311]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

记忆增强的动态多目标分解进化算法*

刘 敏[1,2,3], 曾文华[3,4]

[1](漳州师范学院 计算机科学与工程系,福建 漳州 363000)

[2](厦门大学 智能科学与技术系,福建 厦门 361005)

[3](福建省仿脑智能系统重点实验室(厦门大学),福建 厦门 361005)

[4](厦门大学 软件学院,福建 厦门 361005)

通讯作者: 曾文华, E-mail: whzeng64@163.com, http://www.xmu.edu.cn

摘 要: 现实世界中的一些多目标优化问题经常受动态环境影响而不断发生变化,要求优化算法不断地及时跟踪时变的 Pareto 最优解集.提出了一种记忆增强的动态多目标分解进化算法.将动态多目标优化问题分解为若干个动态单目标优化子问题并同时优化这些子问题,以便快速逼近 Pareto 最优解集.给出了一个改进的环境变化检测算子,以便更好地检测环境变化.设计了一种基于子问题的串式记忆方法,利用过去类似环境下搜索到的最优解来有效地响应新的环境变化.在 8 个标准的测试问题上,将新算法与其他 3 种记忆增强的动态进化多目标优化算法进行了实验比较.结果表明,新算法比其他 3 种算法具有更快的运行速度、更强的记忆能力与鲁棒性能,并且新算法所获得的解集还具有更好的收敛性与分布性.

关键词: 进化计算;多目标优化;动态环境;记忆方法;分解

中图法分类号: TP183 **文献标识码**: A

中文引用格式: 刘敏,曾文华.记忆增强的动态多目标分解进化算法.软件学报,2013,24(7):1571−1588. http://www.jos.org.cn/1000-9825/4311.htm

英文引用格式: Liu M, Zeng WH. Memory enhanced dynamic multi-objective evolutionary algorithm based on decomposition. Ruan Jian Xue Bao/Journal of Software, 2013,24(7):1571−1588 (in Chinese). http://www.jos.org.cn/1000-9825/4311.htm

Memory Enhanced Dynamic Multi-Objective Evolutionary Algorithm Based on Decomposition

LIU Min[1,2,3], ZENG Wen-Hua[3,4]

[1](Department of Computer Science and Engineering, Zhangzhou Normal University, Zhangzhou 363000, China)

[2](Cognitive Science Department, Xiamen University, Xiamen 361005, China)

[3](Fujian Provincial Key Laboratory of Brain-Like Intelligent Systems (Xiamen University), Xiamen 361005, China)

[4](Software School, Xiamen University, Xiamen 361005, China)

Corresponding author: ZENG Wen-Hua, E-mail: whzeng64@163.com, http://www.xmu.edu.cn

Abstract: In addition to the need for satisfying several objectives, many real-world problems are also dynamic and require the optimization algorithm to continuously track the time-varying Pareto optimal set over time. This paper proposes a memory enhanced dynamic multi-objective evolutionary algorithm based on decomposition (denoted by dMOEAD-M). Specifically, the dMOEAD-M decomposes a dynamic multi-objective optimization problem into a number of dynamic scalar optimization subproblems and optimizes them simultaneously. An improved environment detection operator is presented. Also, a subproblem-based bunchy memory scheme, which allows evolutionary algorithm to store good solutions from old environments and reuse them as necessary, is designed to respond to the environment change. Simulation results on eight benchmark problems show that the proposed dMOEAD-M not only runs at a faster speed, more memory capabilities, and a better robustness, but is also able to find a much better spread of solutions and converge better near the

* 基金项目: 国家自然科学基金(60975076); 福建省教育厅科技项目(JA12221)
收稿时间: 2011-12-27; 定稿时间: 2012-07-26

© 中国科学院软件研究所 http://www.jos.org.cn

changing Pareto optimal front, compared with three other memory enhanced dynamic evolutionary multi-objective optimization algorithms.

Key words: evolutionary computation; multi-objective optimization; dynamic environment; memory scheme; decomposition

现实世界存在大量多目标优化问题(multi-objective optimization problems,简称 MOPs).MOPs 中,多个目标经常彼此冲突,需要在优化过程中对各个目标进行折中,最后得出一组折中解——Pareto 最优解集(Pareto optimal set,简称 POS)供决策者进行决策[1].进化多目标优化(evolutionary multi-objective optimization,简称 EMO)主要研究如何利用进化算法求解 MOPs,已成为进化计算领域的研究热点[2].MOPs 进一步可分为静态 MOPs 与动态 MOPs(dynamic MOPs,简称 DMOPs).受动态环境影响,DMOPs 的目标函数、约束函数和相关参数都可能随时间不断变化[3].因此,求解 DMOPs 的动态 EMO(dynamic EMO,简称 DEMO)算法必须能够自动检测与响应新变化,以快速的收敛速度及时跟踪时变的 POS,这给 EMO 研究带来新的挑战[4].近 5 年来,一些研究人员对 DEMO 产生了浓厚兴趣,他们分别根据遗传算法、人工免疫算法、粒子群优化算法、协同进化算法以及膜计算等自然计算方法设计了相应的 DEMO 算法[4−12].DEMO 在智能机器人导航、工业设计、管理工程、控制及优化调度等领域也有了初步应用[7,8,12−15].因此,对 DEMO 进行深入研究具有重要的理论意义和实际应用价值[16].

如何利用过去搜索到的最优解对新的环境变化做出快速响应,是设计 DEMO 算法的一大难点.由于记忆方法能够通过记忆复用以前搜索的最优解来帮助算法对新变化做出较好的响应,因此,已有一些动态单目标进化算法采用记忆方法有效地增强了算法的动态跟踪性能[17−19].但是至今为止,记忆方法在 DEMO 中却鲜有深入研究.2007 年,尚荣华等人提出了动态多目标免疫克隆优化算法(immune clonal algorithm for DMOPs,简称 ICADMO)[5].同年,Deb 等人在著名的 NSGA-II(non-dominated sorting genetic algorithm II)算法[20]基础上提出了动态 NSGA-II(dynamic NSGA-II,简称 DNSGA-II)算法[7].2009 年,Goh 等人提出了动态竞争-合作的协同多目标进化算法(dynamic competition-cooperation coevolutionary algorithm,简称 dCOEA)[4].2010 年,Koo 和 Goh 等人提出了动态多目标进化梯度搜索(dynamic multi-objective evolutionary gradient search,简称 dMO-EGS)算法[11].这些 DEMO 算法都曾试图采用记忆方法来快速响应新的变化,但是它们的实际记忆效果并不理想.

设计 DEMO 算法的另一大难点是,既要求算法具有快速的收敛速度,又要求算法在每次环境变化过程中所获得的解集都具备良好的收敛性与分布性.目前的 DEMO 算法绝大多数沿袭了静态 EMO 算法的传统思路——采用基于 Pareto 支配的适应值赋值方法与精英保留策略来保证算法的收敛性,采用多样性保持算子来保证算法获得解集的分布性[1].2007 年,张青富等人为静态 EMO 提出一种新思路,利用数学规划中的多目标分解方法将 MOPs 分解为若干个单目标优化子问题,并同时优化这些子问题.他们提出的基于分解的多目标进化算法(multi-objective evolutionary algorithm based on decomposition,简称 MOEA/D)[21]荣获 2009 年 IEEE 进化计算大会多目标优化国际竞赛第一名[22],比多数传统的静态 EMO 算法具有更快的运行速度、更好的解集收敛性与分布性[21,22].但据我们所知,这种新思路至今尚未被引入到 DEMO 领域.

针对以上两大难点,本文将多目标分解思想与记忆方法有机地结合起来,提出一种记忆增强的动态多目标分解进化算法(memory enhanced dynamic multi-objective evolutionary algorithm based on decomposition,简称 dMOEAD-M).其主要学术贡献表现在:

1) 将动态多目标优化问题分解为若干个动态单目标优化子问题,并同时优化这些子问题;
2) 提出了一个改进的环境变化检测算子,以更好地检测环境变化;
3) 设计了一种基于子问题的串式记忆方法,能够利用过去的最优解对新的环境变化做出有效响应.

算法的理论分析与实验结果说明,dMOEAD-M 比其他 3 种记忆增强的 DEMO 算法[5,7,11]具有更好的动态跟踪性能.

© 中国科学院软件研究所 http://www.jos.org.cn

1 问题描述

不失一般性,一个具有 n 个决策变量,m 个目标函数的 MOPs 可描述为[1]

$$\begin{cases}\min\limits_{x\in\Omega} \boldsymbol{y}=\boldsymbol{F}(\boldsymbol{x})=(f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),...,f_m(\boldsymbol{x}))^T\\ \text{s.t. } g_i(\boldsymbol{x})\leqslant 0,i=1,2,...,p;h_j(\boldsymbol{x})=0,j=1,2,...,q\end{cases} \tag{1}$$

其中,

- $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)^T\in\boldsymbol{\Omega}\subset\boldsymbol{R}^n$ 为 n 维决策(变量)向量,$\boldsymbol{\Omega}$为 n 维决策空间;
- $\boldsymbol{y}=(f_1,f_2,\ldots,f_m)^T\in\boldsymbol{\Lambda}\subset\boldsymbol{R}^m$ 为 m 维目标函数向量,$\boldsymbol{\Lambda}$为 m 维目标空间;
- 评价函数 $\boldsymbol{F}(\boldsymbol{x}):\boldsymbol{\Omega}\to\boldsymbol{\Lambda}$定义了 m 个由决策空间到目标空间的映射;
- $g_i(\boldsymbol{x})\leqslant 0,i=1,2,\ldots,p$ 定义了 p 个不等式约束;
- $h_j(\boldsymbol{x})\leqslant 0,j=1,2,\ldots,q$ 定义了 q 个等式约束.

定义 1(Pareto 支配(dominance)关系)[1]. 向量 $\boldsymbol{u}=(u_1,u_2,...,u_m)^T$Pareto 支配向量 $\boldsymbol{v}=(v_1,v_2,...,v_m)^T$,记为 $\boldsymbol{u}\prec\boldsymbol{v}$,当且仅当:

1) $\forall k\in\{1,2,\ldots,m\}$满足 $u_k\leqslant v_k$;
2) $\exists l\in\{1,2,\ldots,m\}$满足 $u_l<v_l$.

定义 2(Pareto 最优解). 决策向量 $\boldsymbol{x}\in\boldsymbol{\Omega}$被称为$\boldsymbol{\Omega}$上的 Pareto 最优解,当且仅当$\neg\exists\boldsymbol{x}'\in\boldsymbol{\Omega}$,使得 $\boldsymbol{F}(\boldsymbol{x}')\prec\boldsymbol{F}(\boldsymbol{x})$.

定义 3(Pareto 最优解集(POS)). 对于一个给定的 MOPs,其 **POS** 定义为[1]

$$\boldsymbol{POS}:=\{\boldsymbol{x}\in\boldsymbol{\Omega}\mid\neg\exists\boldsymbol{x}'\in\boldsymbol{\Omega},\boldsymbol{F}(\boldsymbol{x}')\prec\boldsymbol{F}(\boldsymbol{x})\}.$$

定义 4(Pareto 最优前沿(Pareto optimal front,简称 POF)). 对于一个给定的 MOPs,其 **POF** 定义为[1]

$$\boldsymbol{POF}:=\{\boldsymbol{y}=\boldsymbol{F}(\boldsymbol{x})\mid\boldsymbol{x}\in\boldsymbol{POS}\}.$$

作为自然的扩展,一个具有 n 个决策变量、m 个目标函数的 DMOPs 可描述为[3]

$$\begin{cases}\min\limits_{x\in\Omega} \boldsymbol{y}=\boldsymbol{F}(\boldsymbol{x},t)=(f_1(\boldsymbol{x},t),f_2(\boldsymbol{x},t),...,f_m(\boldsymbol{x},t))^T\\ \text{s.t. } g_i(\boldsymbol{x},t)\leqslant 0,i=1,2,...,p;h_j(\boldsymbol{x},t)=0,j=1,2,...,q\end{cases} \tag{2}$$

其中,评价函数 $\boldsymbol{F}(\boldsymbol{x},t)$、不等式约束条件函数 $g_i(\boldsymbol{x},t)$以及等式约束条件函数 $h_j(\boldsymbol{x},t)$都可能随时间 t 发生变化.

相应地,

- DMOPs 的 $\boldsymbol{POS}(t)$可定义为 $\boldsymbol{POS}(t):=\{\boldsymbol{x}\in\boldsymbol{\Omega}\mid\neg\exists\boldsymbol{x}'\in\boldsymbol{\Omega},\boldsymbol{F}(\boldsymbol{x}',t)\prec\boldsymbol{F}(\boldsymbol{x},t)\}$;
- DMOPs 的 $\boldsymbol{POF}(t)$可定义为 $\boldsymbol{POF}(t):=\{\boldsymbol{y}=\boldsymbol{F}(\boldsymbol{x},t)\mid\boldsymbol{x}\in\boldsymbol{POS}(t)\}$.

2 相关工作

2.1 多目标优化的分解方法

数学规划领域常采用标量化(scalarization)方法求解 MOPs.标量化是指 MOPs 可以被分解转换为多个标量(单目标)优化子问题.典型的标量化方法包括权重和法、柴贝彻夫法(Tchebycheff approach)、基于罚值的边界交集法(penalty-based boundary intersection,简称 PBI)等[21].下面首先以柴贝彻夫法为例来解释多目标优化的分解原理,然后再介绍基于分解的多目标进化算法(MOEA/D)的基本思想.

2.1.1 柴贝彻夫多目标分解方法

一个柴贝彻夫标量优化问题可描述为[21]

$$\begin{cases}\min\limits_{x\in\Omega} u^{te}(\boldsymbol{x}\mid\boldsymbol{\lambda},\boldsymbol{z}^*)=\max\limits_{1\leqslant i\leqslant m}\{\lambda_i\mid f_i(\boldsymbol{x})-z_i^*\mid\}\\ \text{s.t. } g_i(\boldsymbol{x})\leqslant 0,i=1,2,...,p;h_j(\boldsymbol{x})=0,j=1,2,...,q\end{cases} \tag{3}$$

其中,

- $\boldsymbol{\lambda}=(\lambda_1,...,\lambda_m)^T$ 是权重向量,对于所有 $i=1,...,m$,满足$\lambda_i\geqslant 0$ 且 $\sum_{i=1}^m\lambda_i=1$;

© 中国科学院软件研究所 http://www.jos.org.cn

- $\boldsymbol{z}^* = (z_1^*,...,z_m^*)^T$ 是参考点, $z_i^* = \min\{f_i(\boldsymbol{x}) \mid \boldsymbol{g}(\boldsymbol{x}) \leqslant \boldsymbol{0}; \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}; \boldsymbol{x} \in \boldsymbol{\Omega}\}$,i=1,…,m;
- $\boldsymbol{g}(\boldsymbol{x})$与 $\boldsymbol{h}(\boldsymbol{x})$分别为不等式与等式约束函数向量.

对于公式(1)中 MOPs 的每一个 Pareto 最优解 $\boldsymbol{x}^*$,都存在一个相应的权重向量$\boldsymbol{\lambda}$,使得 $\boldsymbol{x}^*$是公式(3)的最优解.公式(3)的每一个最优解也是公式(1)中 MOPs 的 Pareto 最优解[21].因此,通过变更权重向量$\boldsymbol{\lambda}$即可获得不同的 Pareto 最优解.这样,MOPs 就可以分解转换为多个带不同权重向量的柴贝彻夫标量优化子问题.

2.1.2 基于分解的多目标进化算法(MOEA/D)

MOEA/D[21]选择 N(N 为进化算法的种群规模)个均匀分布的不同权重向量($\boldsymbol{\lambda}^1,...,\boldsymbol{\lambda}^N$)作为输入参数,通过多目标优化分解方法将 MOPs 分解为 N 个标量优化子问题,并同时优化这些子问题(由此获得的最优解集可近似为 MOPs 的 **POS**[21]).因为采用了标量化目标函数快速、简易地计算个体的适应值,并通过子问题间的均匀分布自然地保持了种群的多样性与解集的分布性,所以 MOEA/D 能够轻易地解决非分解的 EMO 算法中的适应值赋值和多样性保持等难题[21].MOEA/D 的种群由 N 个标量优化子问题的当前最优解组成.MOEA/D 中任一权重向量$\boldsymbol{\lambda}^i$的邻域定义为与之距离最近的 T 个权重向量 $\{\boldsymbol{\lambda}^{i_1},...,\boldsymbol{\lambda}^{i_T}\}$,T 为邻域大小.因而第 i 个子问题的邻域则定义为 $B(i)=\{i_1,\ldots,i_T\}$.MOEA/D 通过在各子问题邻域内进行进化搜索来同时优化这些标量优化子问题.每个子问题的进化优化过程包括如下 3 个操作:

(1) 遗传操作;

(2) 参考点更新操作;

(3) 子问题的邻域更新操作.

MOEA/D 的算法时间复杂度为 $O(mNT)$,m 为目标数.因为一般有 T<N,所以 MOEA/D 具有较低的时间复杂度.

2.2 记忆增强的DEMO算法回顾

许多实际动态优化问题呈现出近似的周期性变化,比如城市交通变化[19],此类问题在新环境下的最优解可能又重新回到以前搜索过的位置.如果采用记忆方法帮助进化算法复用以前搜索的最有解,那么记忆增强后的动态进化算法将具有更好的动态跟踪性能[17,19].根据记忆信息量的多少,本文建议将记忆方法大致分为短期记忆和中期记忆两类(鉴于动态环境下计算资源的有限性,一般不宜采用长期记忆).短期记忆仅记忆前一次环境变化的最优解;相反地,中期记忆则记忆以前若干次环境变化的最优解.

2.2.1 短期记忆增强的 DEMO 算法

(1) 短期记忆增强的动态多目标免疫克隆优化算法(ICADMO)

ICADMO 算法[5]将前一个时刻的最终抗体群作为下一个时刻的初始种群,以保证算法较好的收敛速度.该方法保留(即记忆)了前一次环境变化的所有最优解,具有短期记忆效果.当抗体群规模为 N、非支配集大小为 N_n、克隆比例为 N_c、目标数为 m 时,该算法的时间复杂度**为 $O(mN^2+N_nN_c+mN_n\log N_n)$.

(2) 兼顾短期记忆与少量多样性引入的 DNSGA-II 算法

DNSGA-II 算法[7]通过一个短期记忆与多样性引入操作(本文将其命名为 short-term memory and diversity introduction,简称 SMDI)来响应新的环境变化***.SMDI 首先将环境变化前一代种群中的大多数个体保留(记忆)下来,因此具有短期记忆效果.然后对当前种群中其他小部分个体进行随机初始化,给种群引入少量多样性,以便响应新变化.当种群规模为 N、目标数为 m 时,DNSGA-II 算法的时间复杂度为 $O(mN^2)$,与 NSGA-II 算法复杂度[20]相同.

以上两种 DEMO 算法中的记忆方法虽然简单、易行,但是它们仅能记忆前一次环境变化的最优解,记忆能

** 文献[5]中 ICADMO 的算法时间复杂度原为 $O(N+N_nN_c+mN_n\log N_n)$,在我们与文献[5]作者联系之后,确认了 ICADMO 实际的算法时间复杂度应为 $O(mN^2+N_nN_c+mN_n\log N_n)$.

*** 文献[7]实际给出了随机初始化和超变异两种多样性引入方法,从而得到 DNSGA-II-A 和 DNSGA-II-B 两个版本的 DNSGA-II.为了简洁起见,本文的 DNSGA-II 仅指 DNSGA-II-A.

© 中国科学院软件研究所 http://www.jos.org.cn

力有限.特别是当前后两次环境变化差别较大时,则基本上失去了记忆效果.

2.2.2 中期记忆增强的 DEMO 算法

如果给进化算法增加这样一种中期记忆方法:记忆池(memory pool)+存入(store)过程+检索(retrieve)过程,其中,记忆池保存过去的最优解,存入过程负责将过去最优解存入记忆池,检索过程负责从记忆池检索出最优解并将其插入到新环境下的种群中,那么进化算法可以通过记忆复用以前若干次环境变化的最优解,对新变化做出更好的响应[17].1999 年,Branke 就如何设计中期记忆方法提出了以下 3 个重要问题[17]:

Q1. 应该在什么时候将种群中的哪些个体存入到记忆池中?

Q2. 记忆池中应该保存多少个体,哪些个体应被替换(replace)以便记忆池腾出空间来接纳其他新个体?

Q3. 应从记忆池中检索哪些个体并将它们重新插入到种群中?

下面首先围绕这 3 个问题简要介绍中期记忆增强的 DEMO 算法;然后在第 3 节提出的 dMOEAD-M 算法中再作深入探讨.

(1) 临时记忆增强的动态竞争-合作的协同多目标进化算法(dCOEA)

dCOEA 算法[4]中使用了一种临时记忆(temporal memory,简称 TM)方法来处理档案集(一种用于存放非支配解的外部种群)里的过期解.

- 关于 Q1,TM 在每次环境变化时从档案集中选出 R_{size} 个非支配解.如果 R_{size} 小于目标数目(m),则随机地选择 R_{size} 个极端非支配解保存至记忆池;否则,先将 m 个极端非支配解保存至记忆池,然后从档案集中随机选出 $R_{size}-m$ 个个体存入记忆池;
- 关于 Q2,TM 每次用新存入的 R_{size} 个个体来替换记忆池中最老的 R_{size} 个个体;
- 关于 Q3,TM 方法中没有设计相应的检索过程.因此,TM 是一种不完整的中期记忆方法.

(2) 基于中心点-方差的记忆方法增强的动态多目标进化梯度搜索算法(dMO-EGS)

dMO-EGS[11]算法中设计了一种基于中心点-方差的记忆方法(本文将其命名为 centroid-variance based memory,简称 CVM),以增强算法的动态跟踪性能.

- 关于 Q1,CVM 的存入过程大致为:首先计算档案集的中心点向量($\boldsymbol{C}_\tau$)和方差向量($\tilde{\boldsymbol{C}}_\tau$),再将 $\boldsymbol{C}_\tau$和 $\tilde{\boldsymbol{C}}_\tau$ 合并为一个记忆项目存入记忆池;
- 关于 Q2,如果记忆池存满,则 CVM 将存入时间最久的记忆项目移出记忆池,以便接纳新的记忆项目;
- 关于 Q3,CVM 的检索过程大致为:首先,备份记忆池;其次,截断备份的记忆池,使其减小为预定大小;然后,对备份记忆池中的每一个记忆项目,以 $\boldsymbol{C}_\tau$为均值,$\tilde{\boldsymbol{C}}_\tau$ 为方差,根据正态分布产生新解 $\boldsymbol{x}$,并将 $\boldsymbol{x}$ 插入到档案集中.

值得注意的是,CVM 的存入过程仅保存中心点向量和方差向量而不记忆档案集中的最优解,检索过程仅根据中心点向量和方差向量进行正态分布抽样.这两个过程之间容易产生失真,其记忆的准确性难以得到保障.

3 记忆增强的动态多目标分解进化算法(dMOEAD-M)

3.1 dMOEAD-M算法的整体设计流程

dMOEAD-M 主要包括以下 5 个步骤:

- 步骤 1 为初始化过程,负责完成种群、参考点及记忆池的初始化;
- 步骤 2 为动态多目标分解,负责将 DMOPs 分解为 N(N 为算法的种群规模)个动态单目标(标量)优化子问题,并计算子问题的邻域;
- 步骤 3 为检测并响应环境变化,本文为此设计了一种改进的环境变化检测算子.此外,结合已分解的子问题,还提出了一种基于子问题的串式记忆(subproblem-based bunchy memory,简称 SBM)方法来对检测到的新变化进行响应;
- 步骤 4 为进化优化 N 个动态标量优化子问题;

© 中国科学院软件研究所 http://www.jos.org.cn

- 步骤 5 为终止判断.

dMOEAD-M 的整体设计流程如图 1 所示,下面将重点对步骤 2~步骤 4 进行描述.

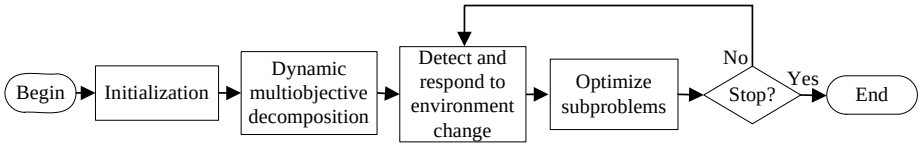


Fig.1 Flowchart of dMOEAD-M

图 1 dMOEAD-M 的整体设计流程

3.2 动态多目标分解

利用 N 个均匀分布的不同权重向量$(\boldsymbol{\lambda}^1,...,\boldsymbol{\lambda}^N)$将公式(2)中的 DMOPs 分解为 N 个动态标量优化子问题.下面分别以动态的柴贝彻夫法和动态的基于罚值的边界交集法(PBI)为例来阐明动态多目标分解的原理.

一个动态柴贝彻夫标量优化问题可描述为

$$\begin{cases}\min\limits_{x\in\Omega} u^{te}(\boldsymbol{x},t\,|\,\boldsymbol{\lambda},\boldsymbol{z}^*)=\max\limits_{1\leqslant i\leqslant m}\{\lambda_i\,|\,f_i(\boldsymbol{x},t)-z_i^*|\}\\ \text{s.t. } g_i(\boldsymbol{x},t)\leqslant 0, i=1,2,...,p; h_j(\boldsymbol{x},t)=0, j=1,2,...,q\end{cases} \tag{4}$$

其中,

- $u^{te}(\boldsymbol{x},t|\boldsymbol{\lambda},\boldsymbol{z}^*)$为动态柴贝彻夫标量目标函数,$t$ 是时间变量, $\boldsymbol{z}^*=(z_1^*,...,z_m^*)^T$ 是参考点;
- $z_i^*=\min\{f_i(\boldsymbol{x},t)\,|\,\boldsymbol{g}(\boldsymbol{x},t)\leqslant\boldsymbol{0};\boldsymbol{h}(\boldsymbol{x},t)=\boldsymbol{0};\boldsymbol{x}\in\boldsymbol{\Omega}\}$,$i$=1,…,$m$;
- $\boldsymbol{g}(\boldsymbol{x},t)$与 $\boldsymbol{h}(\boldsymbol{x},t)$分别为不等式与等式约束函数向量.

因为公式(4)的最优解 $\boldsymbol{x}^*$是公式(2)中 DMOPs 的一个 Pareto 最优解,所以由 N 个不同动态柴贝彻夫标量优化子问题所获得的最优解集可近似为 DMOPs 的 Pareto 最优解集 $\boldsymbol{POS}(t)$[21].

与文献[21]类似,一个动态 PBI 标量优化问题可描述为

$$\begin{cases}\min\limits_{x\in\Omega} u^{pbi}(\boldsymbol{x},t\,|\,\boldsymbol{\lambda},\boldsymbol{z}^*)=d_1+\theta d_2\\ d_1=\dfrac{\|(\boldsymbol{z}^*-F(\boldsymbol{x},t))^T\boldsymbol{\lambda}\|}{\|\boldsymbol{\lambda}\|},\ d_2=\|F(\boldsymbol{x},t)-(\boldsymbol{z}^*-d_1\boldsymbol{\lambda})\|\\ \text{s.t. } g_i(\boldsymbol{x},t)\leqslant 0, i=1,2,...,p; h_j(\boldsymbol{x},t)=0, j=1,2,...,q\end{cases} \tag{5}$$

其中,$u^{pbi}(\boldsymbol{x},t|\boldsymbol{\lambda},\boldsymbol{z}^*)$为动态 PBI 标量目标函数,$\theta$是预设的罚参数,$\boldsymbol{F}(\boldsymbol{x},t)$是公式(2)中的评价函数,$d_1$ 和 d_2 是两个距离值[21],其余符号定义与公式(4)相同.类似地,由 N 个动态 PBI 标量优化子问题所获得的最优解集可近似为 DMOPs 的 $\boldsymbol{POS}(t)$.与动态柴贝彻夫法相比,在求解两个以上目标的 DMOPs 时,动态 PBI 分解法所获得的最优解集分布更均匀[21],但是动态 PBI 分解法需要多设置一个罚参数θ.

动态多目标分解方法具有静态多目标分解方法相似的优点,可以利用动态标量化目标函数快速且简易地计算个体的适应值,并利用子问题间的均匀分布自然地保持了种群的多样性与解集的分布性,从而轻易解决了非分解的 DEMO 算法中的适应值赋值和多样性保持等难题.

此外,对于分解后的任意一个动态标量优化子问题 i(i=1,…,N),本文采用 MOEA/D 中的邻域方法[21]为其定义一个邻域 $B(i)$,用以进化优化该子问题.

3.3 环境变化检测算子

文献[3,6,12]给出了基于目标函数变化的检测算子,但它们缺少对约束函数变化的检测.虽然文献[7]建议同时检测目标函数与约束函数变化(即每进化一代时都随机选择少量个体进行重新评价,如果任意一个个体目标函数或约束函数数值有变化,则认为环境发生变化),但是这种方法在现实应用中容易将目标或约束函数评价过程中的细小噪音[18]所引起的变化也误认为环境变化.为此,本文设计了一种改进的环境变化检测算子,同时检测

© 中国科学院软件研究所 http://www.jos.org.cn

目标函数与约束函数的变化,并通过显示平均法[18]与设置阈值来尽可能地消除噪音的不利影响.新算子定义如下:

$$\delta(t)=\frac{\sum_{i}^{K}\left(\frac{\|\boldsymbol{F}(\boldsymbol{x}^i,t)-\boldsymbol{F}(\boldsymbol{x}^i,t-1)\|}{\|\boldsymbol{F}(\boldsymbol{x}^i,t-1)\|+\varepsilon}+\frac{\|\boldsymbol{g}(\boldsymbol{x}^i,t)-\boldsymbol{g}(\boldsymbol{x}^i,t-1)\|}{\|\boldsymbol{g}(\boldsymbol{x}^i,t-1)\|+\varepsilon}+\frac{\|\boldsymbol{h}(\boldsymbol{x}^i,t)-\boldsymbol{h}(\boldsymbol{x}^i,t-1)\|}{\|\boldsymbol{h}(\boldsymbol{x}^i,t-1)\|+\varepsilon}\right)}{K}>\tilde{\delta} \tag{6}$$

其中,ε为一个任意小的正数,$\|\cdot\|$为欧氏距离,K 表示从种群中随机选出的个体数(一般可取 $1\leqslant K\leqslant 5$), $\tilde{\delta}$ 为预先设定的阈值(一般可取 $10^{-3}<\tilde{\delta}<10^{-2}$),其他符号定义参见公式(2).当 $\delta(t)>\tilde{\delta}$ 时,则表示环境发生了动态变化.

3.4 响应环境变化的SBM方法

3.4.1 SBM 方法的设计原理

SBM 的设计原理可简述为:每当检测到环境变化后,先从若干个代表性子问题上抽取出一串代表解保存至记忆池;然后通过检索记忆信息,复用以前环境变化中的最优解来对新变化做出响应.

为此,下面围绕第 2.2.2 节中的 Q1~Q3,在 SBM 方法中设计了抽取(sample)、存入和检索 3 个过程.此外,还采用一个与种群规模大小相等的记忆池(memory pool,简称 ***M***).本文将要保存的每串个体视为一个有机整体,将 ***M*** 设计成串式队列的形式(令串为队列的基本元素).下面利用图 2 来详细阐明 SBM 方法.

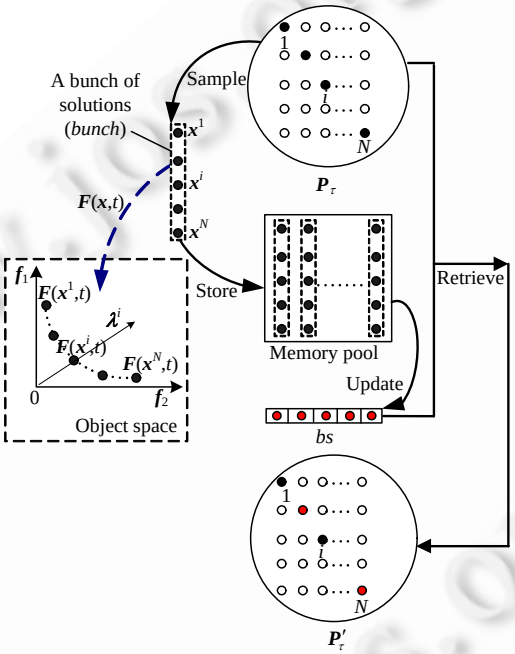


Fig.2 Principle of SBM scheme

图 2 SBM 方法的设计原理

- 关于 Q1,抽取过程先从 N 个子问题中均匀地选出 *Bsize* 个代表性子问题$\{i_1,\ldots,i_{Bsize}\}$,然后选择这些子问题的当前最优解所对应的个体作为要抽取的一串个体;
- 关于 Q2,存入过程先将抽取出的一串个体插入到 ***M*** 的队尾,如果 ***M*** 存满,则按先进先出策略进行替换;存入完毕后,针对新环境重新对 ***M*** 中的个体进行评价(evaluate),即计算个体的目标函数值、约束函数值和标量目标函数值(scalar objective function,简称 SOF);最后,更新一个大小为 *Bsize* 的最优个体数组 *bs*,让元素 $bs[j](j=1,\ldots,Bsize)$保存在 $\boldsymbol{\lambda}^{ij}$ 权重向量方向上具有最优标量目标函数值的个体;
- 关于Q3,检索过程首先针对新环境重新对当前种群 $\boldsymbol{P}_\tau$中的个体进行评价;然后,令 $\boldsymbol{P}'_\tau=\boldsymbol{P}_\tau$;最后,让最优数组 *bs* 与 $\boldsymbol{P}_\tau$进行竞争,即对于 *Bsize* 个子问题$\{i_1,\ldots,i_{Bsize}\}$中任意一个子问题 i_j,如果个体 $bs[j]$的标量目标

© 中国科学院软件研究所 http://www.jos.org.cn

函数值小于个体 $\boldsymbol{P}_\tau[i_j]$的标量目标函数值,则令 $\boldsymbol{P}'_\tau[i_j]=bs[j]$,进行种群更新,得到记忆后的种群 $\boldsymbol{P}'_\tau$.

算法 1. 基于子问题的串式记忆方法(SBM).

输入:种群 $\boldsymbol{P}_\tau$;记忆池 $\boldsymbol{M}$;

输出:种群 $\boldsymbol{P}'_\tau$.

第 1 步. 抽取过程:

1) 均匀选择 *Bsize* 个代表性子问题$\{i_1,\ldots,i_{Bsize}\}$;
2) 用数组 *bunch* 保存这些子问题的当前最优解.

第 2 步. 存入过程:

1) 将 *bunch* 存入到 $\boldsymbol{M}$ 的队尾;如果 $\boldsymbol{M}$ 存满,则将 $\boldsymbol{M}$ 队首的串删除;
2) 重新评价 $\boldsymbol{M}$ 中的个体;
3) 初始化最优个体数组 *bs*:令 *bs* 等于 $\boldsymbol{M}$ 的队首串;
4) 更新 *bs*:for(*bunch*=$\boldsymbol{M}$ 中的第 2 串,...,最后串){
   for (*j*=1,...,*Bsize*),若 *bunch*[*j*].sof<*bs*[*j*].sof,则 *bs*[*j*]=*bunch*[*j*]}.

第 3 步. 检索过程:

1) 重新评价 $\boldsymbol{P}_\tau$中的个体;
2) 令 $\boldsymbol{P}'_\tau=\boldsymbol{P}_\tau$;
3) 让 *bs* 与 $\boldsymbol{P}_\tau$竞争:for (*j*=1,...,*Bsize*){
   if (*bs*[*j*].sof<$\boldsymbol{P}_\tau[i_j]$.sof),令 $\boldsymbol{P}'_\tau[i_j]=bs[j]$;}
4) 输出 $\boldsymbol{P}'_\tau$.

SBM 的 sample 过程容易抽出一串分布比较均匀的个体,在目标空间上保持了较好的多样性.例如,当 *N*=100 且 *Bsize*=5 时,如图 2 所示,sample 过程先从 100 个子问题中选出 5 个子问题$\{i|i=1,25,50,75,100\}$;然后,在这些子问题上从种群 $\boldsymbol{P}_\tau$中抽出 5 个解$\{\boldsymbol{x}^1,\boldsymbol{x}^{25},\ldots,\boldsymbol{x}^{100}\}$,这 5 个解在目标空间上的映射为$\{\boldsymbol{F}(\boldsymbol{x}^1,t),\boldsymbol{F}(\boldsymbol{x}^{25},t),\ldots,\boldsymbol{F}(\boldsymbol{x}^{100},t)\}$.由于 *Bsize* 个代表性子问题间分布均匀,因此目标空间上的这 5 个映射点也分布比较均匀.

3.4.2 SBM 方法的时间复杂度分析

设种群 $\boldsymbol{P}_\tau$和记忆池 $\boldsymbol{M}$ 的规模都为 *N*,令 *Bsize*<0.1×*N*.从 *N* 个子问题中选出 *Bsize* 个子问题的时间复杂度为 *O*(*N*),从 *Bsize* 个子问题上选出 *Bsize* 个当前最优解的时间复杂度为 *O*(*Bsize*),故抽取过程的时间复杂度为 *O*(*N*);存入一串个体与删除一串个体的时间复杂度同为 *O*(*Bsize*),更新 *bs* 数组的时间复杂度为 *O*(*N*),故存入过程的时间复杂度为 *O*(*N*);检索过程的时间复杂度为 *O*(*Bsize*).总之,SBM 的时间复杂度为 *O*(*N*),具有较低的复杂度.

3.5 子问题的进化优化

与 MOEA/D 的方法相似,通过一个循环同时优化 *N* 个子问题.每个子问题的进化优化包括如下 3 个操作:

(1) 遗传操作:从子问题的邻域内任选两个个体,通过模拟的二进制交叉和多项式变异[20]产生一个新个体;
(2) 参考点更新操作:比较新个体的目标函数向量与参考点向量,一旦子目标出现更小的目标函数值,则修改参考点向量的相应分量;
(3) 邻域更新操作:对于子问题邻域内的任意个体 $\boldsymbol{x}^j$,如果新个体在$\boldsymbol{\lambda}^j$权重向量方向上的动态标量目标函数值比 $\boldsymbol{x}^j$ 的动态标量目标函数值小,则更新个体 $\boldsymbol{x}^j$.

3.6 dMOEAD-M算法详细描述及其时间复杂度分析

算法 2. 记忆增强的动态多目标分解进化算法(dMOEAD-M).

输入:DMOPs;终止条件(算法的总进化代数 *G*);*N* 个均匀分布的不同权重向量$(\boldsymbol{\lambda}^1,\ldots,\boldsymbol{\lambda}^N)$;每个权重向量邻域内的权重向量个数(*T*);环境变化频率(τ_T);环境变化剧烈程度(n_T);

输出:种群 $\boldsymbol{P}_\tau$.

© 中国科学院软件研究所 http://www.jos.org.cn

第 1 步. 初始化:

1) 初始化种群:令迭代次数计数器τ=0;$\boldsymbol{P}_\tau$={$\boldsymbol{x}^1$,...,$\boldsymbol{x}^N$};$\boldsymbol{y}^i$=$\boldsymbol{F}$($\boldsymbol{x}^i$,t),i=1,…,N,$\boldsymbol{F}$($\boldsymbol{x}$,t)为评价函数;
2) 初始化参考点 $\boldsymbol{z}$=(z_1,…,z_m):for (j=1,...,m),令 z_j=$\min_{1\leqslant i\leqslant N} f_j(\boldsymbol{x}^i,t)$;
3) 初始化记忆池 $\boldsymbol{M}$:令 $\boldsymbol{M}$=∅.

第 2 步. 动态多目标分解:

1) 根据 N 个均匀分布的($\boldsymbol{\lambda}^1$,...,$\boldsymbol{\lambda}^N$),利用动态柴贝彻夫分解法(或动态 PBI 分解法),将 DMOPs 分解为 N 个动态标量优化子问题;
2) 计算每个子问题的邻域[21]:计算任意两个权重向量间的欧氏距离;for (i=1,...,N){求出与权重向量$\boldsymbol{\lambda}^i$最近的 T 个权重向量 $\boldsymbol{\lambda}^{i_1}$,...,$\boldsymbol{\lambda}^{i_T}$, 令第 i 个子问题的邻域 B(i)={i_1,…,i_T}}.

第 3 步. 检测并响应环境变化:

利用公式(6)定义的环境变化检测算子检测环境变化;
if (环境发生了变化){输出中间结果:输出环境变化前的种群 $\boldsymbol{P}_\tau$;
响应变化:调用算法 1,记忆复用以往环境下的最优解,获得记忆后的种群 $\boldsymbol{P}'_\tau$;令 $\boldsymbol{P}_\tau$= $\boldsymbol{P}'_\tau$;
}.

第 4 步. 子问题的进化优化:

for (i=1,...,N){

1) 遗传操作:从邻域 B(i)内任选两个编号 k 和 l,对 $\boldsymbol{x}^k$ 和 $\boldsymbol{x}^l$ 进行遗传操作产生 $\boldsymbol{x}'$;
2) 更新参考点 $\boldsymbol{z}$:for (j=1,...,m){if ($f_j(\boldsymbol{x}',t)<z_j$),令 z_j=$f_j(\boldsymbol{x}',t)$};
3) 邻域更新:∀j∈B(i),if ($\boldsymbol{x}'$的动态标量目标函数值≤$\boldsymbol{x}^j$的动态标量目标函数值){令 $\boldsymbol{x}^j$=$\boldsymbol{x}'$,$\boldsymbol{y}^j$=$\boldsymbol{F}$($\boldsymbol{x}'$,t)}

}.

第 5 步. 终止判断:

如果不满足终止条件(例如τ<G),则令τ=τ+1,并跳转至第 3 步;否则,输出 $\boldsymbol{P}_\tau$并停止.

下面考虑 dMOEAD-M 算法运行一代的时间复杂度.dMOEAD-M 的时间开销主要集中在检测并响应环境变化以及子问题的进化优化这两个步骤:

(1) 检测与响应步骤:检测操作的时间复杂度为 O(K),K 为公式(6)中用于检测的个体数;响应环境变化(即调用 SBM)的时间复杂度为 O(N).因为 K<N,所以该步骤时间复杂度为 O(N);
(2) 子问题的进化优化步骤:因为涉及 N 个子问题,而每个子问题的邻域大小为 T,所以该步骤的时间复杂度为 O(mNT),m 为目标个数.

综合步骤(1)、步骤(2),dMOEAD-M 算法的时间复杂度为 O(mNT),与 MOEA/D 的时间复杂度[21]相同.因为一般有 T<N,所以 dMOEAD-M 算法时间复杂度优于 DNSGA-II 的时间复杂度[7]——O(mN^2),而且还优于 ICADMO 的时间复杂度[5]——O(mN^2+N_nN_c+mN_nlogN_n).

4 仿真实验

首先,参照文献[5,7,20,21]和公开的 NSGA-II 源代码[23]、MOEA/D 源代码[22],我们用 C++语言编程实现了 DNSGA-II,ICADMO 和本文提出的 dMOEAD-M 算法.鉴于文献[11]中的 dMO-EGS 算法是一种非主流的 EMO 算法,且文献[11]对 dMO-EGS 的某些细节没有详细描述,我们考虑将 dMO-EGS 中的 CVM 记忆方法集成到更广为人知的 DNSGA-II 算法中,即用 CVM 替代 DNSGA-II 中的 SMDI,从而得到 DNSGA-II-CVM 算法以进行实验比较.此外,因为 dCOEA 使用的临时记忆是一种不完整的中期记忆方法,所以不对其进行实验比较;然后,分别用 DNSGA-II,DNSGA-II-CVM,ICADMO 和 dMOEAD-M 这 4 种记忆增强的 DEMO 算法求解 DMOPs 标准测试问题;最后对实验结果进行比较分析.

4.1 测试问题

采用 FDA 系列[3]与 dMOP 系列问题[4]作为测试问题.文献[7]修改了 FDA2 问题(修改版本记作 FDA2mod).

© 中国科学院软件研究所 http://www.jos.org.cn

因为 FDA2mod 无周期性变化,本文对 FDA2mod 进行改进,给出一个新的具有周期性变化的 FDA2(记作 FDA2new).此外,FDA3 是文献[24]中提出的改进版本(记作 FDA3mod).dMOP 系列问题是对 FDA 系列问题的扩展[4].以上测试问题的定义参见表 1.其中,当环境变化剧烈程度(n_T)值较小时,环境变化剧烈程度较大,每个周期内不同的 ***POF***(t)个数较少;n_T 值较大时,环境变化剧烈程度较小,每个周期内不同的 ***POF***(t)个数较多.

Table 1 Test problems for dynamic multi-objective optimization

表 1 动态多目标优化的测试问题

Problems	Objective functions	Variable bounds	n
FDA1	$\begin{cases} f_1(\boldsymbol{x}_{\mathrm{I}})=x_1,\ f_2(\boldsymbol{x})=g\cdot h \\ g(\boldsymbol{x}_{\mathrm{II}})=1+\sum\limits_{x_i\in x_{\mathrm{II}}}(x_i-G(t))^2, h(f_1,g)=1-\sqrt{f_1/g} \\ G(t)=\sin(0.5\pi t), t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{I}}=(x_1)\in[0,1]$ $\boldsymbol{x}_{\mathrm{II}}=(x_2,\ldots,x_n)\in[-1,1]$	20
FDA2 (new)	$\begin{cases} f_1(\boldsymbol{x}_{\mathrm{I}})=x_1,\ f_2(\boldsymbol{x})=g\cdot h \\ g(\boldsymbol{x}_{\mathrm{II}})=1+\sum\limits_{x_i\in x_{\mathrm{II}}}x_i^2, h(\boldsymbol{x}_{\mathrm{III}},f_1,g)=1-(f_1/g)^{2\left(H(t)+\sum\limits_{x_i\in x_{\mathrm{III}}}(x_i-H(t)/4)^2\right)} \\ H(t)=2\sin(0.5\pi(t-1)), t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{I}}=(x_1)\in[0,1]$ $\boldsymbol{x}_{\mathrm{II}}=(x_2,\ldots,x_6)\in[-1,1]$ $\boldsymbol{x}_{\mathrm{III}}=(x_7,\ldots,x_n)\in[-1,1]$	20
FDA3 (mod)	$\begin{cases} f_1(\boldsymbol{x}_{\mathrm{I}})=x_1^{F(t)},\ f_2(\boldsymbol{x})=g\cdot h \\ g(\boldsymbol{x}_{\mathrm{II}})=1+G(t)+\sum\limits_{x_i\in x_{\mathrm{II}}}(x_i-G(t))^2, h(f_1,g)=1-\sqrt{f_1/g} \\ G(t)=\lvert\sin(0.5\pi t)\rvert, F(t)=10^{2\sin(0.5\pi t)}, t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{I}}=(x_1)\in[0,1]$ $\boldsymbol{x}_{\mathrm{II}}=(x_2,\ldots,x_n)\in[-1,1]$	30
FDA4	$\begin{cases} f_1(\boldsymbol{x})=(1+g(\boldsymbol{x}_{\mathrm{II}}))\prod\limits_{i=1}^{m-1}\cos(0.5\pi x_i) \\ f_k(\boldsymbol{x})=(1+g(\boldsymbol{x}_{\mathrm{II}}))\left(\prod\limits_{i=1}^{m-k}\cos(0.5\pi x_i)\right)\sin(0.5\pi x_{m-k+1}), k=2:m-1 \\ f_m(\boldsymbol{x})=(1+g(\boldsymbol{x}_{\mathrm{II}}))\sin(0.5\pi x_1) \\ g(\boldsymbol{x}_{\mathrm{II}})=\sum\limits_{x_i\in x_{\mathrm{II}}}(x_i-G(t))^2, G(t)=\lvert\sin(0.5\pi t)\rvert, t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{II}}=(x_m,\ldots,x_n)$ $x_i\in[0,1], i=1:n$	12
FDA5	$\begin{cases} f_1(\boldsymbol{x})=(1+g(\boldsymbol{x}_{\mathrm{II}}))\prod\limits_{i=1}^{m-1}\cos(0.5\pi y_i) \\ f_k(\boldsymbol{x})=(1+g(\boldsymbol{x}_{\mathrm{II}}))\left(\prod\limits_{i=1}^{m-k}\cos(0.5\pi y_i)\right)\sin(0.5\pi y_{m-k+1}), k=2:m-1 \\ f_m(\boldsymbol{x})=(1+g(\boldsymbol{x}_{\mathrm{II}}))\sin(0.5\pi y_1) \\ g(\boldsymbol{x}_{\mathrm{II}})=G(t)+\sum\limits_{x_i\in x_{\mathrm{II}}}(x_i-G(t))^2, G(t)=\lvert\sin(0.5\pi t)\rvert \\ y_i=x_i^{F(t)}, \text{for } i=1,\ldots,(m-1), F(t)=1+100\sin^4(0.5\pi t) \\ t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{II}}=(x_m,\ldots,x_n)$ $x_i\in[0,1], i=1:n$	12
dMOP1	$\begin{cases} f_1(\boldsymbol{x}_{\mathrm{I}})=x_1,\ f_2(\boldsymbol{x})=g\cdot h \\ g(\boldsymbol{x}_{\mathrm{II}})=1+9\cdot\sum\limits_{x_i\in x_{\mathrm{II}}}x_i^2, h(f_1,g)=1-(f_1/g)^{H(t)} \\ H(t)=0.75\cdot\sin(0.5\pi t)+1.25, t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{I}}=(x_1)\in[0,1]$ $\boldsymbol{x}_{\mathrm{II}}=(x_2,\ldots,x_n)\in[0,1]$	10
dMOP2	$\begin{cases} f_1(\boldsymbol{x}_{\mathrm{I}})=x_1,\ f_2(\boldsymbol{x})=g\cdot h \\ g(\boldsymbol{x}_{\mathrm{II}})=1+\sum\limits_{x_i\in x_{\mathrm{II}}}(x_i-G(t))^2, h(f_1,g)=1-(f_1/g)^{H(t)} \\ G(t)=\sin(0.5\pi t), H(t)=0.75\cdot\sin(0.5\pi t)+1.25, t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{I}}=(x_1)\in[0,1]$ $\boldsymbol{x}_{\mathrm{II}}=(x_2,\ldots,x_n)\in[-1,1]$	10
dMOP3	$\begin{cases} f_1(x_r)=\lvert x_r\rvert, f_2(\boldsymbol{x}\setminus x_r)=g\cdot h \\ g(\boldsymbol{x}\setminus x_r)=1+\sum_{i=1}^{x\setminus x_r}(x_i-G(t))^2, h(f_1,g)=1-\sqrt{f_1/g} \\ G(t)=\sin(0.5\pi t), r=\bigcup(1,2,\ldots,n), t=\lfloor \tau/\tau_T \rfloor/n_T \end{cases}$	$\boldsymbol{x}_{\mathrm{I}}=(x_1)\in[0,1]$ $\boldsymbol{x}_{\mathrm{II}}=(x_2,\ldots,x_n)\in[-1,1]$	20

© 中国科学院软件研究所 http://www.jos.org.cn

4.2 性能评价指标

采用动态逆向世代距离 $rGD(t)$和超容量比率 $HVR(t)$作为 DEMO 算法性能评价指标[7,9,21],定义如下:

$$rGD(t)=\frac{\sum_{i=1}^{|\boldsymbol{POF}^*(t)|}d_i}{|\boldsymbol{POF}^*(t)|},d_i=\min_{k=1}^{|\boldsymbol{Q}(t)|}\sqrt{\sum_{j=1}^{m}(f_j^{*(i)}-f_j^{(k)})^2} \tag{7}$$

$$HVR(t)=\frac{HV(\boldsymbol{Q}(t))}{HV(\boldsymbol{POF}^*(t))},HV=volume(\bigcup_{i=1}^{|\boldsymbol{Q}|}v_i) \tag{8}$$

其中,$\boldsymbol{POF}^*(t)$为 t 时刻抽样的 Pareto 最优前沿,$\boldsymbol{Q}(t)$为 t 时刻算法所得解集,HV 表示解集 $\boldsymbol{Q}$ 的超容量, $f_j^{*(i)}$ 为 $\boldsymbol{POF}^*(t)$中第 i 个抽样点的第 j 个目标函数值.$rGD(t)$和 $HVR(t)$都可用来评价所得解集的近似收敛程度与分布性,$rGD(t)$值越小(或 $HVR(t)$值越接近 1),表示算法获得解集的收敛性和分布性越好.

4.3 实验参数设置

4 种记忆增强的 DEMO 算法均采用实数制染色体编码.DNSGA-II,DNSGA-II-CVM 与 dMOEAD-M 采用模拟二进制交叉和多项式变异,交叉分布式指数(η_c)及变异分布式指数(η_m)分别设定为 10 和 20;ICADMO 采用整体克隆(克隆比例为 5),非一致变异[5];DNSGA-II 的多样式引入比例(ζ)设定为 0.2;DNSGA-II-CVM 和 dMOEAD-M 的记忆池大小设定为 100;dMOEAD-M 的邻域内权重向量的个数(T)设定为 20,采用动态柴贝彻夫法分解双目标 DMOPs($Bsize$ 设定为 5),采用动态 PBI 方法分解三个目标 DMOPs(设定 $Bsize$ 为 15,罚参数θ为 5).4 种算法的公共参数设置见表 2,其中,n 为决策变量维数.为研究非确定环境下各种动态变化的影响,根据不同 DMOPs 给出不同(τ_T,n_T)组合,对每个问题的各种组合都做了 30 次独立实验.每次实验,各种算法统一跟踪 100 次环境变化,每种算法每次运行的总进化代数(G)=环境变化次数×环境变化频率.

Table 2 Common parameter setting for experiment

表 2 实验中公共的参数设置

Parameter	Value
Population size (N)	Two objectives: N=100; Three objectives: N=300
Crossover probability	0.9
Mutation probability	1/n
Frequency of change (τ_T)	5, 10, 15, 20, 25, 35
Severity of change (n_T)	5, 10

4.4 实验结果与分析

4.4.1 性能评价指标的统计结果对比

表 3 是 4 种算法求解不同 DMOPs 测试问题时所获得解集的性能评价指标的统计结果,该表显示:

(1) 在所有的测试问题与不同(τ_T,n_T)组合上,dMOEAD-M 获得解集的 $rGD(t)$与 $HVR(t)$值都具有最好的均值,且大多数情况下具有最好的方差.这表明,dMOEAD-M 获得解集的收敛性与分布性明显好于其他 3 种算法;

(2) 对于每个测试问题,当(τ_T,n_T)组合中 τ_T 值不变而 n_T 值减小时(即环境变化剧烈程度增大时),dMOEAD-M 的均值变化不大,但其他 3 种算法的均值一般都较大程度地变差,这说明 dMOEAD-M 对不同环境变化剧烈程度都具有较好的适应能力,其鲁棒性更好.究其原因,这主要得益于 dMOEAD-M 所采用的 SBM 记忆方法能够较准确地记忆过去若干次环境变化的最优解,可对不同环境变化做出更好的响应.

Table 3 Mean and variance of the performance metric of solution sets founded by four algorithms

表 3 4 种算法获得解集的性能评价指标的均值与方差

Problems	(τ_T,n_T)	Statistic	$rGD(t)$				$HVR(t)$			
			DNSGA-II	DNSGA-II-CVM	ICA DMO	dMOEAD-M	DNSGA-II	DNSGA-II-CVM	ICA DMO	dMOEAD-M
FDA1	(10,10)	Mean	3.85E−01	6.24E−01	2.98E−01	**5.08E−02**	7.12E−01	6.16E−01	7.68E−01	**9.55E−01**
		Variance	8.37E−02	3.44E−01	4.03E−02	**3.01E−03**	4.49E−02	1.04E−01	2.16E−02	**2.22E−03**
	(25,10)	Mean	3.65E−02	3.52E−02	5.33E−02	**1.27E−02**	9.72E−01	9.74E−01	9.54E−01	**9.90E−01**
		Variance	6.37E−04	5.80E−04	1.31E−03	**2.80E−05**	4.03E−04	3.50E−04	9.92E−04	**5.07E−05**
	(25,5)	Mean	1.63E−01	1.23E−01	1.50E−01	**9.00E−03**	8.76E−01	9.06E−01	8.76E−01	**9.94E−01**
		Variance	1.81E−02	1.02E−02	7.74E−03	**3.03E−05**	4.98E−03	5.94E−03	4.98E−03	**4.99E−05**

© 中国科学院软件研究所 http://www.jos.org.cn

Table 3 Mean and variance of the performance metric of solution sets founded by four algorithms (Continued)

表 3 4 种算法获得解集的性能评价指标的均值与方差(续)

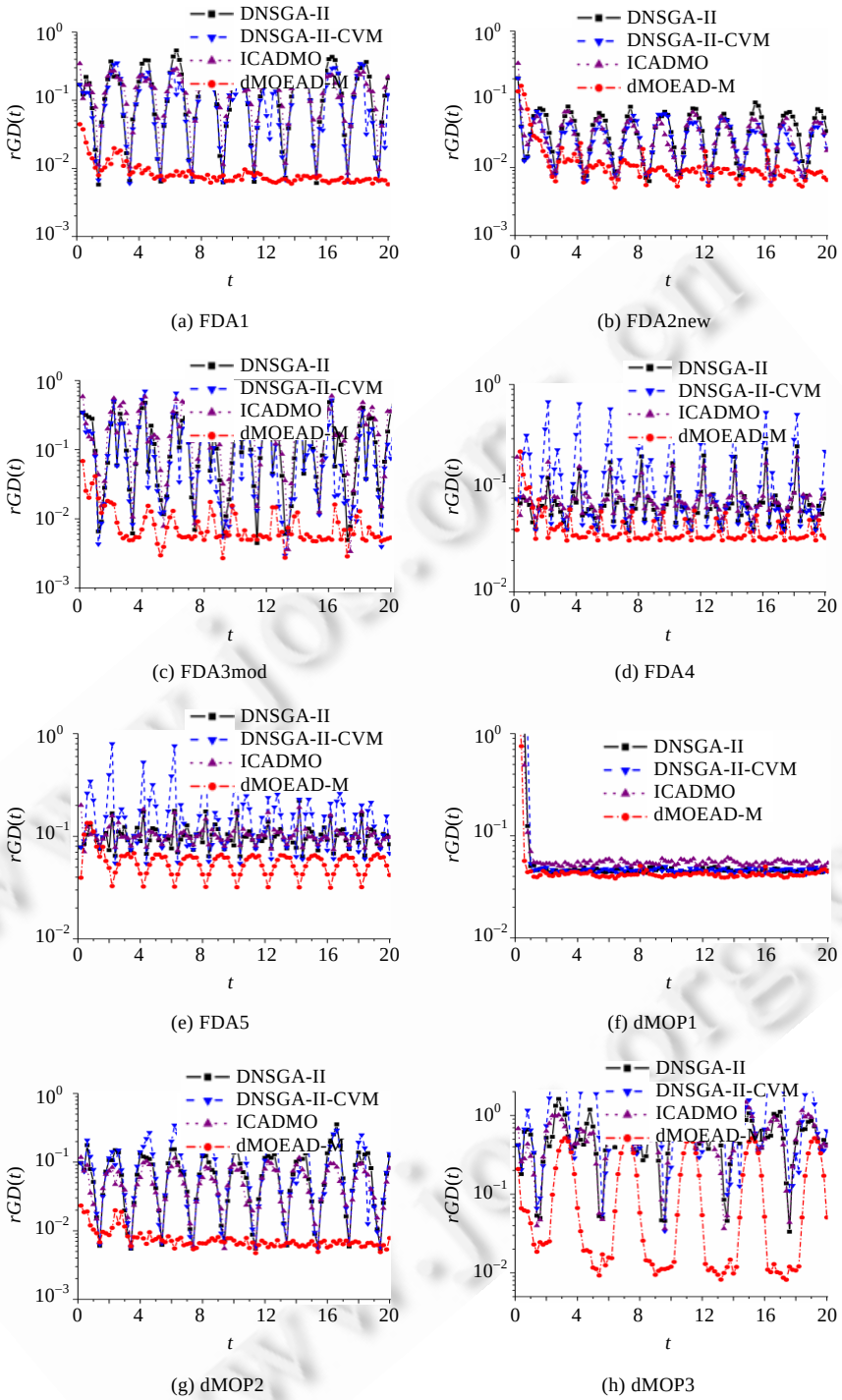
Problems	(τ_T,n_T)	Statistic	*rGD*(*t*)				*HVR*(*t*)			
			DNSGA-II	DNSGA-II-CVM	ICA DMO	dMOEAD-M	DNSGA-II	DNSGA-II-CVM	ICA DMO	dMOEAD-M
FDA2 (new)	(5,10)	Mean	9.64E−02	8.15E−02	7.43E−02	**4.04E−02**	9.58E−01	9.61E−01	9.63E−01	**9.73E−01**
		Variance	5.32E−03	5.44E−03	9.34E−03	**3.14E−03**	1.91E−03	1.91E−03	4.59E−03	**1.84E−03**
	(15,10)	Mean	1.51E−02	1.54E−02	1.88E−02	**1.30E−02**	**9.93E−01**	**9.93E−01**	9.91E−01	**9.93E−01**
		Variance	4.01E−04	4.01E−04	1.14E−03	**2.19E−04**	**1.11E−04**	**1.11E−04**	4.34E−04	1.30E−04
	(15,5)	Mean	3.80E−02	2.94E−02	3.22E−02	**1.47E−02**	9.83E−01	9.87E−01	9.85E−01	**9.92E−01**
		Variance	8.08E−04	5.77E−04	1.24E−03	**5.31E−04**	1.95E−04	**1.48E−04**	4.45E−04	2.51E−04
FDA3 (mod)	(25,10)	Mean	1.07E−01	8.63E−02	1.54E−01	**1.60E−02**	8.82E−01	8.99E−01	8.22E−01	**9.85E−01**
		Variance	1.43E−02	1.06E−02	2.26E−02	**3.44E−04**	6.61E−03	5.20E−03	1.54E−02	**2.64E−04**
	(35,10)	Mean	4.36E−02	4.59E−02	8.18E−02	**1.14E−02**	9.48E−01	9.45E−01	9.08E−01	**9.89E−01**
		Variance	2.06E−03	2.71E−03	9.03E−03	**1.30E−04**	1.13E−03	1.23E−03	7.64E−03	**1.45E−04**
	(35,5)	Mean	1.74E−01	1.36E−01	2.30E−01	**9.08E−03**	8.21E−01	8.64E−01	7.45E−01	**9.88E−01**
		Variance	3.06E−02	2.62E−02	3.38E−02	**7.26E−05**	1.73E−02	1.39E−02	1.93E−02	**1.79E−04**
FDA4	(20,10)	Mean	7.01E−02	7.58E−02	6.90E−02	**5.77E−02**	9.81E−01	9.79E−01	9.79E−01	**9.87E−01**
		Variance	**6.23E−04**	1.09E−03	6.88E−04	6.33E−04	**2.97E−05**	7.22E−05	1.16E−04	7.91E−05
	(25,10)	Mean	5.57E−02	5.77E−02	6.22E−02	**5.08E−02**	9.83E−01	9.83E−01	9.81E−01	**9.89E−01**
		Variance	**1.23E−04**	1.89E−04	1.82E−04	3.10E−04	**4.81E−06**	9.16E−06	3.49E−05	3.70E−05
	(25,5)	Mean	7.61E−02	1.49E−01	8.06E−02	**4.29E−02**	9.79E−01	9.48E−01	9.75E−01	**9.92E−01**
		Variance	1.87E−03	2.46E−02	1.01E−03	**5.97E−04**	1.62E−04	5.08E−03	1.69E−04	**8.08E−05**
FDA5	(20,10)	Mean	9.73E−02	1.10E−01	9.13E−02	**6.17E−02**	9.06E−01	8.96E−01	9.16E−01	**9.50E−01**
		Variance	3.16E−04	9.73E−04	4.86E−04	**1.39E−04**	1.76E−03	1.75E−03	1.09E−03	**9.34E−04**
	(25,10)	Mean	8.35E−02	8.65E−02	8.43E−02	**5.80E−02**	9.19E−01	9.16E−01	9.21E−01	**9.55E−01**
		Variance	**9.72E−05**	1.55E−04	1.10E−04	1.07E−04	1.32E−03	1.32E−03	1.08E−03	**7.38E−04**
	(25,5)	Mean	1.02E−01	1.69E−01	1.05E−01	**5.69E−02**	9.02E−01	8.40E−01	9.01E−01	**9.57E−01**
		Variance	8.43E−04	1.86E−02	6.09E−04	**3.05E−04**	2.72E−03	1.13E−02	1.64E−03	**1.14E−03**
dMOP1	(5,10)	Mean	1.30E−01	1.91E−01	1.49E−01	**2.01E−02**	9.44E−01	9.19E−01	9.56E−01	**9.85E−01**
		Variance	3.84E−01	5.83E−01	7.45E−01	**1.55E−02**	3.57E−02	5.21E−02	3.29E−02	**8.38E−03**
	(15,10)	Mean	3.45E−02	3.25E−02	3.64E−02	**6.94E−03**	9.79E−01	9.80E−01	9.82E−01	**9.95E−01**
		Variance	4.31E−02	4.17E−02	6.33E−02	**5.27E−04**	1.28E−02	1.25E−02	1.21E−02	**6.79E−04**
	(15,5)	Mean	3.09E−02	3.32E−02	3.40E−02	**7.14E−03**	9.89E−01	9.84E−01	9.80E−01	**9.95E−01**
		Variance	4.14E−02	4.20E−02	6.16E−02	**5.36E−04**	2.47E−03	1.27E−02	1.11E−02	**7.23E−04**
dMOP2	(5,10)	Mean	2.72E−01	6.49E−01	1.47E−01	**3.90E−02**	7.55E−01	5.25E−01	8.55E−01	**9.62E−01**
		Variance	3.60E−02	2.94E−01	1.41E−02	**7.13E−04**	2.71E−02	1.10E−01	1.21E−02	**6.64E−04**
	(15,10)	Mean	2.60E−02	2.50E−02	2.42E−02	**1.01E−02**	9.74E−01	9.76E−01	9.71E−01	**9.90E−01**
		Variance	2.52E−04	2.38E−04	2.37E−04	**1.32E−05**	3.10E−04	2.55E−04	3.92E−04	**2.99E−05**
	(15,5)	Mean	7.57E−02	9.76E−02	5.19E−02	**7.72E−03**	9.28E−01	9.09E−01	9.39E−01	**9.94E−01**
		Variance	3.60E−03	7.39E−03	9.65E−04	**1.01E−05**	3.39E−03	6.46E−03	1.38E−03	**2.18E−05**
dMOP3	(5,10)	Mean	1.18E+00	4.17E+00	1.23E+00	**2.70E−01**	3.48E−01	2.20E−01	3.23E−01	**7.74E−01**
		Variance	8.56E−01	1.50E+01	6.70E−01	**3.15E−02**	7.59E−02	9.04E−02	1.02E−01	**1.64E−02**
	(15,10)	Mean	2.79E−01	3.03E−01	2.40E−01	**1.25E−01**	7.90E−01	7.62E−01	8.19E−01	**9.06E−01**
		Variance	3.17E−02	3.10E−02	2.90E−02	**2.63E−02**	1.67E−02	1.67E−02	1.45E−02	**1.28E−02**
	(15,5)	Mean	6.13E−01	1.63E+00	5.73E−01	**1.41E−01**	5.58E−01	3.57E−01	5.83E−01	**8.98E−01**
		Variance	1.32E−01	2.27E+00	1.03E−01	**3.08E−02**	5.69E−02	1.27E−01	4.64E−02	**1.51E−02**

4.4.2 性能评价指标随时刻 *t* 变化的图形对比

下面从表 3 中每个测试问题的最后一种动态变化组合中各抽出一次代表性实验,画出 4 种算法求解不同 DMOPs 时所获得解集的 *rGD*(*t*)指标随时刻 *t* 变化的结果,如图 3 所示.图 3(a)是 4 种算法在 100 次环境变化下分别求解 FDA1 问题时所获得解集的 *rGD*(*t*)结果.其中,每 4 个时刻(即 20 次环境变化)为 1 个周期,共 5 个周期.在这 5 个周期内,dMOEAD-M 获得解集的 *rGD*(*t*)值绝大多数情况下都明显优于其他 3 种算法的结果,表明其具有更好的动态跟踪性能.此外,从第 2 周期开始,凭借 SBM 方法第 1 周期形成的良好记忆,dMOEAD-M 获得解集的 *rGD*(*t*)值与第 1 周期相比有了显著下降,表明其获得解集的收敛性和分布性得到了很大改善,即 SBM 方法有效地增强了 dMOEAD-M 算法的动态跟踪性能.相反地,其他 3 种算法后 4 个周期 *rGD*(*t*)的变化情况基本上与第 1 周期相似,*rGD*(*t*)值并未因记忆而改善.

© 中国科学院软件研究所 http://www.jos.org.cn

图 3(b)~图 3(h)中多数情形与图 3(a)类似.此外,*HVR*(*t*)的时变图与 *rGD*(*t*)有相似结果,这里不再赘述.



(a) FDA1 (b) FDA2new

(c) FDA3mod (d) FDA4

(e) FDA5 (f) dMOP1

(g) dMOP2 (h) dMOP3

Fig.3 *rGD*(*t*) results obtained by four algorithms during 100 environment changes

图 3 100 次环境变化下 4 种算法的 *rGD*(*t*)结果

© 中国科学院软件研究所 http://www.jos.org.cn

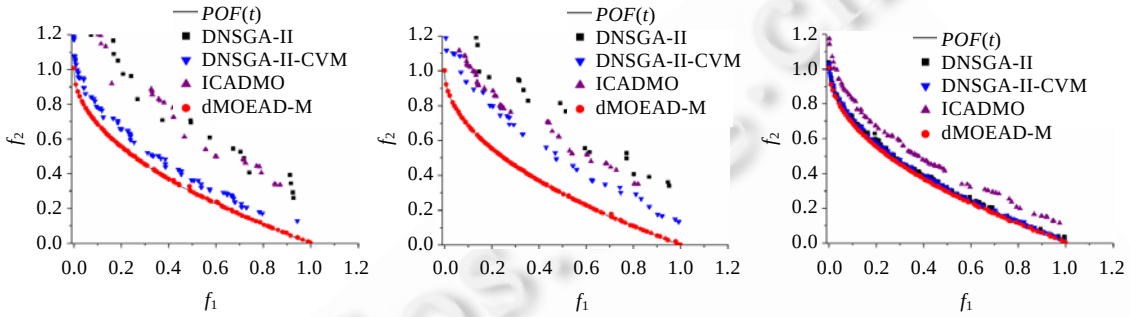
4.4.3 获得解集的分布图对比

下面仍以第 4.4.2 节的那组代表性实验为例,通过绘出 4 种算法在 3 个不同时刻求解不同 DMOPs 所获得的解集分布图,来直观地分析各种算法的性能.

(1) 双目标的 DMOPs

FDA1 的最优解的某些分量(x_{II})随 $G(t)$以正弦函数方式变化(参见表 1),故 FDA1 的 ***POS***(t)随时间发生变化.但是,任意时刻 t,其 ***POF***(t)均为 $f_2=1-\sqrt{f_1}$,不随时间发生变化.

图 4 为 4 种算法在第 4 周期内 3 个不同时刻求解 FDA1 时所获得的解集.这 3 个时刻所对应的环境变化幅度依次由大变小.dMOEAD-M 所获得的解集都最好地均匀收敛到 ***POF***(t).其他 3 种算法在图 4(a)与图 4(b)中的结果都明显偏离 ***POF***(t),且分布不均匀.



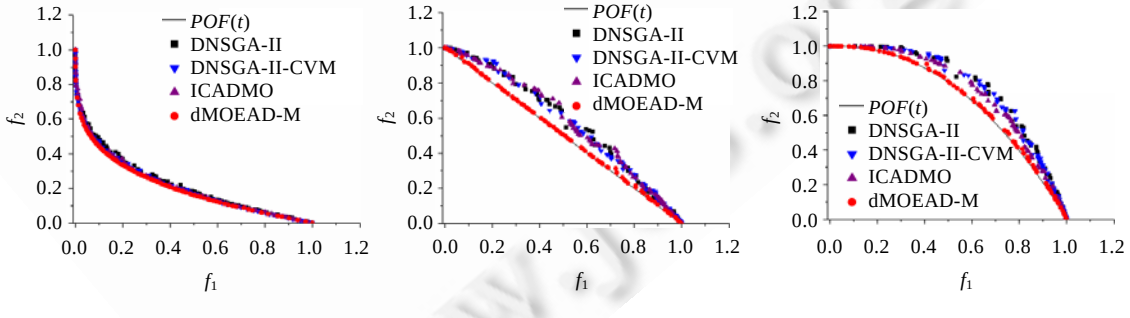
(a) t=12.2 (b) t=12.8 (c) t=13.2

Fig.4 Solution sets founded by four algorithms at three different time steps on FDA1

图 4 4 种算法求解 FDA1 过程中在 3 个不同时刻所获得的解集

FDA2new 的最优解的某些分量(x_{III})随 $H(t)/4$ 以正弦函数方式变化,故 FDA2new 的 ***POS***(t)随时间发生变化.FDA2new 的 ***POF***(t)为 $f_2=1-(f_1/g)^{2^{H(t)}}$,也随 $H(t)$以正弦函数方式发生变化.

图 5 为 4 种算法在第 4 周期内 3 个不同时刻求解 FDA2new 时所获得的解集.图 5(a)~图 5(c)所对应的环境变化幅度依次由小变大,且问题的 ***POF***(t)的形状发生凹凸变化.dMOEAD-M 在 3 个不同时刻下获得的解集都与相应的 ***POF***(t)最为接近,表明其收敛性与分布性好于其他 3 种算法.



(a) t=12.2 (b) t=13.2 (c) t=13.6

Fig.5 Solution sets founded by four algorithms at three different time steps on FDA2new

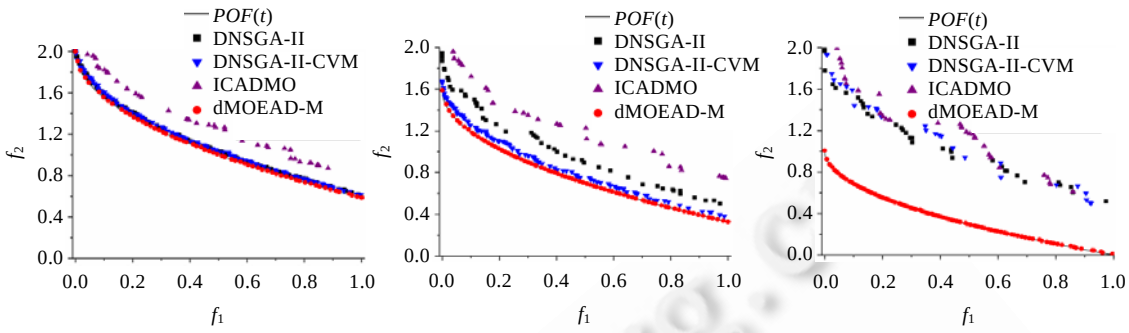
图 5 4 种算法求解 FDA2new 过程中在 3 个不同时刻所获得的解集

FDA3mod 的最优解的某些分量(x_{II})随 $G(t)$以正弦函数方式变化,故 FDA3mod 的 ***POS***(t)随时间发生变化.FDA3mod 的 ***POF***(t)为 $f_2=(1+G(t))\times(1-\sqrt{f_1/(1+G(t))})$,也随 $G(t)$而变化,并且,***POF***(t)上解的密度也随时间而

© 中国科学院软件研究所 http://www.jos.org.cn

发生变化.

图 6 为 4 种算法在第 3 周期内 3 个不同时刻求解 FDA3mod 时所获得的解集.图 6(a)~图 6(c)所对应的环境变化幅度依次由小变大,问题的 ***POF***(t)由上向下移动.3 个子图都表明,dMOEAD-M 的收敛性与分布性都是最好的,特别是在最后一个时刻,其结果远远优于其他 3 种算法.



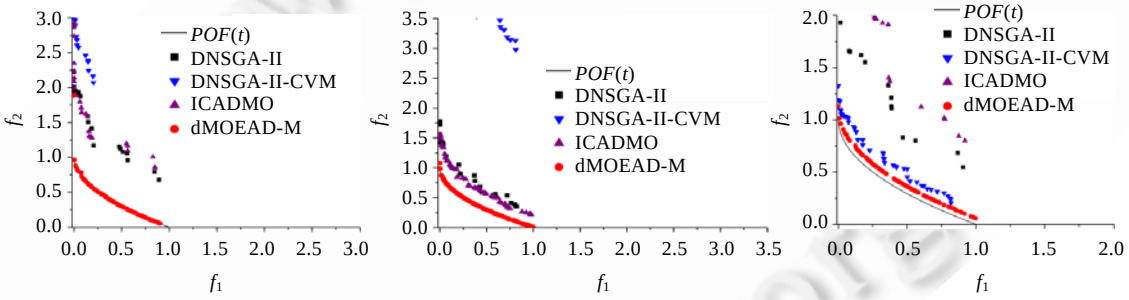
(a) t=9.2 (b) t=9.8 (c) t=10.2

Fig.6 Solution sets founded by four algorithms at three different time steps on FDA3mod

图 6 4 种算法求解 FDA3mod 过程中在 3 个不同时刻所获得的解集

下面在 dMOP 系列问题中,以 dMOP3 为例来比较 4 种算法的性能.dMOP3 是对 FDA1 的一种扩展,它在 FDA1 的基础上,允许控制最优解集分布的决策变量(x_r)发生随机变化,这对算法的跟踪性能具有更大的挑战[4].

图 7 为 4 种算法在 3 个不同时刻求解 dMOP3 时所获得的解集,dMOEAD-M 在这 3 个时刻同样具有最优的跟踪性能.



(a) t=12.2 (b) t=13.2 (c) t=14.4

Fig.7 Solution sets founded by four algorithms at three different time steps on dMOP3

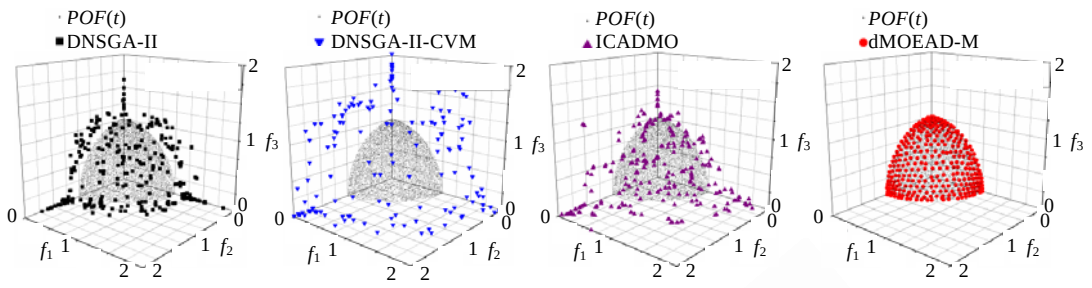
图 7 4 种算法求解 dMOP3 过程中在 3 个不同时刻所获得的解集

(2) 多目标的 DMOPs

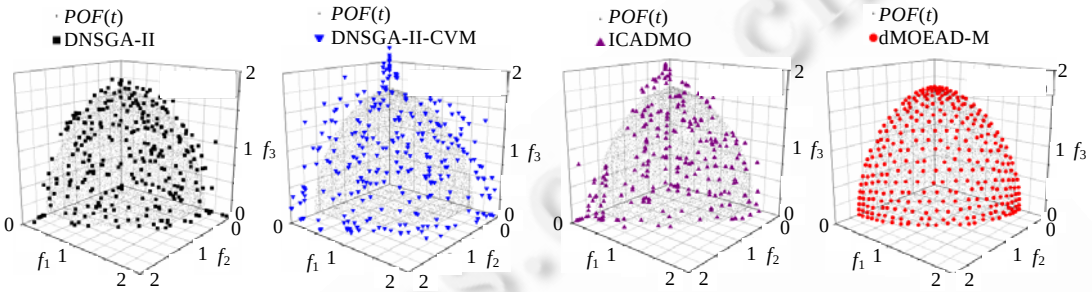
下面以 FDA5 为例来比较 4 种算法的性能.FDA5 的最优解的某些分量($\boldsymbol{x}_{\mathrm{II}}$)随 $G(t)$以正弦函数方式变化,故其 ***POS***(t)随时间发生变化.任意时刻 t,当目标数(m)等于 3 时,FDA5 的 ***POF***(t)为八分之一球面(图 8 中细小黑点标识的阴影区域),但该球面半径大小随时刻 t 在 1~2 之间不断变化.

图 8 为 4 种算法在第 6 周期内 3 个不同时刻求解 FDA5 时所获得的解集.这 3 个时刻所对应的环境变化幅度依次由大变小,并且问题的 ***POF***(t)的球面半径由 1 增大到 2.图 8(a)~图 8(c)显示,dMOEAD-M 能最好地跟踪这 3 个不断变化的 ***POF***(t),其获得解集的收敛性与分布性明显优于其他 3 种算法.类似地,dMOEAD-M 在求解 FDA4 时也具有最优结果,这里不再详述.

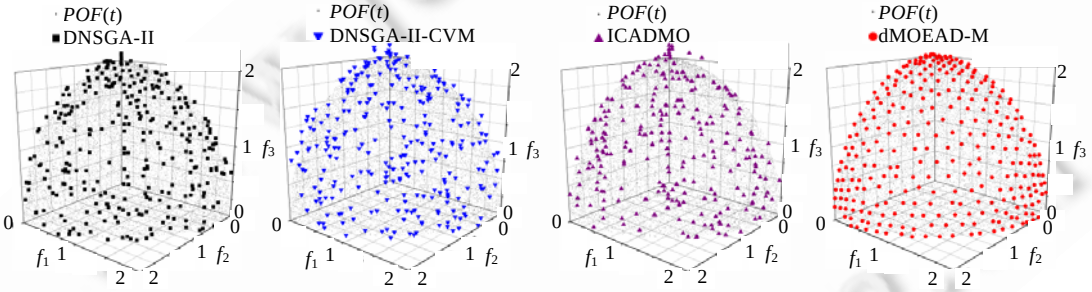
© 中国科学院软件研究所 http://www.jos.org.cn



(a) t=10.2



(b) t=10.6



(c) t=11.2

Fig.8 Solution sets founded by four algorithms at three different time steps on FDA5

图 8 4 种算法求解 FDA5 过程中在 3 个不同时刻所获得的解集

4.4.4 运行时间的对比

统一设定问题的动态变化组合为(25,5),且发生 100 次环境变化.实验硬件为 Intel Core i3 2.93GHz CPU, 4GB 内存的电脑,软件采用 Visual C++ 6.表 4 列出了 4 种算法求解不同 DMOPs 时所耗费的平均计算时间.

Table 4 Average CPU time used by four algorithms (s)

表 4 4 种算法求解不同 DMOPs 所耗费的平均计算时间 (秒)

Problems	DNSGA-II	DNSGA-II-CVM	ICADMO	dMOEAD-M
FDA1	4.50	5.02	5.28	**3.08**
FDA2new	4.91	5.52	8.16	**2.95**
FDA3mod	4.95	5.59	6.19	**3.39**
FDA4	17.32	19.88	29.71	**12.77**
FDA5	18.00	19.74	31.63	**12.91**
dMOP1	3.85	4.70	6.96	**2.58**
dMOP2	3.78	4.01	6.06	**2.49**
dMOP3	4.48	4.99	5.89	**3.24**

© 中国科学院软件研究所 http://www.jos.org.cn

表 4 中:dMOEAD-M 在所有问题上都具有最少的计算时间,反映了 dMOEAD-M 具有最快的运行速度;ICADMO 耗费的时间最多,这可能与克隆个体所增加的目标函数评价次数有关.此外,DNSGA-II-CVM 的时间比 DNSGA-II 稍多一点,这主要是因为 DNSGA-II-CVM 中的 CVM 比 DNSGA-II 中的 SMDI 更耗时而引起的.

5 结束语

本文将多目标分解与记忆方法有机地结合起来,提出了一种记忆增强的动态多目标分解进化算法——dMOEAD-M.为了降低计算复杂度并提高解集的收敛性与分布性,dMOEAD-M 将 DMOPs 分解为若干个动态单目标(标量)优化子问题.给出了一个改进的环境变化检测算子,以更好地检测环境变化.设计了一种基于子问题的串式记忆方法(SBM),能够利用过去的最优解对新的环境变化做出有效的响应,增强了算法的动态跟踪性能.算法分析说明,dMOEAD-M 具有较低的时间复杂度.仿真实验结果表明,在不同的动态环境变化参数组合以及不同的测试问题上,dMOEAD-M 比其他 3 种记忆增强的 DEMO 算法具有更强的记忆能力,能以最快的速度获得具有更好收敛性与分布性的解集.此外,dMOEAD-M 能够较好地适应各种不同剧烈程度的环境变化,其鲁棒性能更好.因为数学规划方法与进化算法有机结合往往可以取得更好的优化效果[21],所以,新的分解策略以及进化范例(例如差分进化)可以考虑集成到 dMOEAD-M 中.此外,设计其他方案的记忆方法也将是我们今后的研究工作.

致谢 在此,感谢 Deb 教授和张青富教授分别在各自的网页上开放了 NSGA-II 与 MOEA/D 算法的源代码.感谢西安电子科技大学的尚荣华副教授对我们邮件中的问题做出了认真的答复.

References:

[1] Coello Coello CA, Lamont GB, Van Veldhuizen DA. Evolutionary Algorithms for Solving Multi-objective Problems. 2nd ed., New York: Springer-Verlag, 2007.

[2] Gong MG, Jiao LC, Yang DD, Ma WP. Research on evolutionary multi-objective optimization algorithms. Ruan Jian Xue Bao/Journal of Software, 2009,20(2):271−289 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3483.htm [doi: 10.3724/SP.J.1001.2009.03483]

[3] Farina M, Deb K, Amato P. Dynamic multiobjective optimization problems: Test cases, approximations, and applications. IEEE Trans. on Evolutionary Computation, 2004,8(5):425−442. [doi: 10.1109/TEVC.2004.831456]

[4] Goh CK, Tan KC. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Trans. on Evolutionary Computation, 2009,13(1):103−127. [doi: 10.1109/TEVC.2008.920671]

[5] Shang RH, Jiao LC, Gong MG, Ma WP. An immune algorithm for dynamic multi-objective optimization. Ruan Jian Xue Bao/Journal of Software, 2007,18(11):2700−2711 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2700.htm [doi: 10.1360/jos182700]

[6] Liu CA, Wang YP. Dynamic multi-objective optimization evolutionary algorithm based on new model. Journal of Computer Research and Development, 2008,45(4):603−611 (in Chinese with English abstract).

[7] Deb K, Rao UBN, Karthik S. Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In: Obayashi S, *et al*., eds. Proc. of the 4th Int'l Conf. on Evolutionary Multi-Criterion Optimization (EMO 2007). Berlin, Heidelberg: Springer-Verlag, 2007. 803−817. [doi: 10.1007/978-3-540-70928-2_60]

[8] Zhang ZH. Multiobjective optimization immune algorithm in dynamic environments and its application to greenhouse control. Applied Soft Computing, 2008,8(2):959−971. [doi: 10.1016/j.asoc.2007.07.005]

[9] Li XD, Branke J, Kirley M. On performance metrics and particle swarm methods for dynamic multiobjective optimization problems. In: Proc. of the 2007 Congress on Evolutionary Computation (CEC 2007). Singapore: IEEE Press, 2007. 576−583. [doi: 10.1109/CEC.2007.4424522]

[10] Wang Y, Li B. Multi-Strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. Memetic Computing, 2010,2(1):3−24. [doi: 10.1007/s12293-009-0012-0]

© 中国科学院软件研究所 http://www.jos.org.cn

[11] Koo WT, Goh CK, Tan KC. A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment. Memetic Computing, 2010,2(2):87−110. [doi: 10.1007/s12293-009-0026-7]

[12] Huang L, Suh IH, Abraham A. Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants. Information Sciences, 2011,181(11):2370−2391. [doi: 10.1016/j.ins.2010.12.015]

[13] Vadakkepat P, Tan KC, Wang ML. Evolutionary artificial potential fields and their application in real time robot path planning. In: Proc. of the 2000 Congress on Evolutionary Computation (CEC 2000). La Jolla: IEEE Press, 2000. 256−263. [doi: 10.1109/CEC.2000.870304]

[14] Barba PD. Dynamic multiobjective optimization: A way to the shape design with transient magnetic fields. IEEE Trans. on Magnetics, 2008,44(6):962−965. [doi: 10.1109/TMAG.2007.916502]

[15] Hutzschenreuter AK, Bosman PAN, Poutre HL. Evolutionary multiobjective optimization for dynamic hospital resource management. In: Ehrgott M, *et al*., eds. Proc. of the 5th Int'l Conf. on Evolutionary Multi-criterion Optimization (EMO 2009). Berlin, Heidelberg: Springer-Verlag, 2009. 320−334. [doi: 10.1007/978-3-642-01020-0_27]

[16] Deb K. Single and multi-objective dynamic optimization: Two tales from an evolutionary perspective. Technical Report, No.2011004, Kanpur: Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology Kanpur, 2011.

[17] Branke J. Memory enhanced evolutionary algorithms for changing optimization problems. In: Proc. of the '99 Congress on Evolutionary Computation (CEC'99). Washington: IEEE Press, 1999. 1875−1882. [doi: 10.1109/CEC.1999.785502]

[18] Jin YC, Branke J. Evolutionary optimization in uncertain environments—A survey. IEEE Trans. on Evolutionary Computation, 2005,9(3):303−317. [doi: 10.1109/TEVC.2005.846356]

[19] Yang SX, Yao X. Population-Based incremental learning with associative memory for dynamic environments. IEEE Trans. on Evolutionary Computation, 2008,12(5):542−561. [doi: 10.1109/TEVC.2007.913070]

[20] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002,6(2):182−197. [doi: 10.1109/4235.996017]

[21] Zhang QF, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. on Evolutionary Computation, 2007,11(6):712−731. [doi: 10.1109/TEVC.2007.892759]

[22] Zhang ZF. MOEA/D homepage. University of Essex. 2011. http://cswww.essex.ac.uk/staff/zhang/webofmoead.htm

[23] Deb K. The source code of NSGA-II. Kanpur Genetic Algorithms Laboratory, 2005. http://www.iitk.ac.in/kangal/codes.shtml

[24] Camara M. Parallel processing for dynamic multi-objective optimization [Ph.D. Thesis]. Granada: University of Granada, 2010.

附中文参考文献:

[2] 公茂果,焦李成,杨咚咚,马文萍.进化多目标优化算法研究.软件学报,2009,20(2):271−289. http://www.jos.org.cn/1000-9825/3483.htm [doi: 10.3724/SP.J.1001.2009.03483]

[5] 尚荣华,焦李成,公茂果,马文萍.免疫克隆算法求解动态多目标优化问题.软件学报,2007,18(11):2700−2711. http://www.jos.org.cn/1000-9825/18/2700.htm [doi: 10.1360/jos182700]

[6] 刘淳安,王宇平.基于新模型的动态多目标优化进化算法.计算机研究与发展,2008,45(4):603−611.

刘敏(1974－),男,湖南湘潭人,博士,讲师,主要研究领域为进化计算,多目标优化.
E-mail: liumin_xt@163.com

曾文华(1964－),男,博士,教授,博士生导师,主要研究领域为软计算,网格计算,云计算.
E-mail: whzeng64@163.com

© 中国科学院软件研究所 http://www.jos.org.cn