关联变量分组的分解多目标进化算法研究

邱飞岳[1,2] 胡 烜[1] 王丽萍[3]

(浙江工业大学信息工程学院 杭州 310023)[1] (浙江工业大学现代教育技术研究所 杭州 310023)[2]

(浙江工业大学信息智能与决策优化研究所 杭州 310023)[3]

摘 要 含有大规模决策变量的优化问题是当前多目标进化算法领域中的研究热点和难点之一。在解决大规模变量问题时,目前的进化算法并没有寻找决策变量之间的关联信息,而都只是将所有变量视为一个整体来进行优化。但随着优化问题中决策变量的增多,"变量维度"成为瓶颈,从而影响算法的性能。针对上述问题,提出关联变量分组策略,通过识别决策变量间内在的关联信息把关联变量分配到同组中,将复杂高维变量的优化问题分解为简单低维的子问题来求解。该策略通过增加关联变量分配到同组中的概率来使算法尽可能地保留变量之间的关联性,减少分组后子问题间的依赖性,从而提高子问题最优解的质量并最终获得最佳的Pareto最优解集。将该算法在标准测试函数上进行变量扩展后再进行仿真对比实验,采用性能指标对算法的收敛性和多样性进行对比分析。实验结果表明,该算法在解决大规模变量的多目标优化问题中,随着决策变量维度的增加,比经典的多目标进化算法NSGA-II、MOEA/D以及RVEA具有更佳的收敛和更好的分布性能,所求得的Pareto解集质量更高。

关键词 大规模优化,关联变量,变量识别,分组分解

中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.12.037

Research on Multi-objective Evolutionary Algorithm Based on Decomposition Using Interacting Variables Grouping

QIU Fei-yue[1,2] HU Xuan[1] WANG Li-ping[3]

(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China)[1]

(Institute of Modern Education Technology, Zhejiang University of Technology, Hangzhou 310023, China)[2]

(Institute of Information Intelligence and Decision Optimization, Zhejiang University of Technology, Hangzhou 310023, China)[3]

Abstract Optimization problem with large-scale decision variable is one of the hot and difficult points in the multi-objective evolutionary algorithm research field. When solving the problem of large-scale variable, the current evolutionary algorithm does not find the related information between decision variables and treats all the decision variables as a whole to optimize. But the variable dimensionality will become the bottleneck as the decision variables in the optimization problem increase, which will affect the performance of the algorithm. To settle these problems, this paper proposed a interacting variable grouping strategy to identify the internal relation among the decision variables and allocate the inte-racting variables to the same group. Thus, it can decompose a difficult high-dimensional problem into a set of simpler and low-dimensional subproblems that are easier to solve. In order to make the algorithm as far as possible to retain the relationship between variables and keep the interdependencies among different subproblems minimal, this strategy increases the probability of assigning the interacting variables to the same group so as to improve the quality of the optimal solution of the subproblems and ultimately gets the best Pareto optimal solution set. Comparative simulation experiment was conducted after the variable extension on standard test function. The convergence and diversity of the algorithm were compared and analyzed using a variety of performance indicators. Experiment results show that this algorithm can produce higher quality Pareto optimal solution set and is of better convergence and distribution than the classical multi-objective evolutionary algorithms like NSGA-II, MOEA/D and RVEA as the dimension of decision variables increases in multi-objective optimization problem with large-scale variable.

Keywords Large-scale optimization, Interacting variables, Variable identification, Grouping decomposition

到稿日期:2016-12-25 返修日期:2017-03-27 本文受国家自然科学基金项目(61472366,61379077),浙江省自然科学基金项目(LY13F030010,LY17F020022)资助。

邱飞岳(1965—),男,博士,教授,博士生导师,主要研究方向为智能控制、优化理论与方法、学习科学与媒体技术;**胡 烜**(1992—),男,硕士生,主要研究方向为多目标进化算法;**王丽萍**(1964—),女,博士,教授,博士生导师,主要研究方向为信息处理、决策优化、商务智能。

1 引言

科学和工程中的优化问题往往非常复杂，不能通过直接的方式来求解最优解。由于多目标优化问题(Multi-objective Optimization Problems，MOPs)具有较大的复杂和困难性，因此将复杂问题简单化的研究势在必行，而决策变量的数目大小就极大地影响着优化问题的复杂性[1]。许多实际问题中的多目标优化问题往往含有成百上千个变量，一般的优化算法已经难以满足实际应用问题的需要。例如，在物流网络设计中为使供应链的物流成本和库存成本最小化，制造商需要将许多回收点和处理点设置在一定的区域范围内，由此将会产生相当数量的决策变量。而当前多目标进化算法(Multi-objective Evolutionary Algorithms，MOEAs)主要是将所有的决策变量当作一个整体来进行优化，随着所求问题变量维度的增加，算法的性能将会急剧下降[2]，这就是人们所说的“维度灾难”。

为解决该问题，现有的学者受到协同进化(Cooperative Coevolution)[3]和联动学习方法(Linkage Learning Methods)[4]的启发，提出决策变量分解的策略。该策略在解决大规模变量问题时通过将一个带有高维决策变量的 MOPs 分解为几个简单低维并且容易解决的子问题，并通过各自独立优化每个子问题来求解。该方法已经被证实可以有效提高进化算法在求解大规模变量优化问题中的效率，因此被广大学者所关注。

但是上述这种“分而治之”的策略存在的主要困难在于如何选择一种较好的分解方法来使不同的子问题间的关联性最小并且算法的优化性能对于所选分解策略具有一定的潜在敏感性。对于大规模变量问题，如果一个算法可以检测函数的结构特征并据此进行变量分解，将大大降低问题的困难程度。Potter[5]于 1994 年提出变量固定分组的思想，其主要内容是在种群进化之前，根据既定方式将变量固定地分解为若干个组，然后将每组变量通过合作协同优化来求解全局最优解，但是在处理含有关联变量的问题时，该算法的性能会显著下降。Yang[6]于 2008 年提出 MLCC 算法并将其应用于大规模连续型全局函数的优化中，该算法使用随机分组策略，在事先未给定任何先验信息的条件下，通过随机的方式来对决策变量进行分组，从而增加将关联变量放到同组中的概率并降低人为因素对变量间关联信息的破坏；同时引入自适应机制来利用种群大小弱化关联变量对分组的影响。Omidvar[7]于 2010 年提出的高频率随机分组的策略以及 Li[8]于 2012 年提出结合动态改变分组大小的随机分组策略均在一定程度上更好地保证了决策变量分组的随机性，并有效地降低了各分组间的依赖程度。但是，现有的随机分组策略一般都采用一次性分组，即决策变量随机分组后，在优化过程中每组的变量将不再改变。研究表明这种方式在随机分组的初始阶段虽能较好地缓解关联变量对算法性能的影响，但是随着关联变量的增加，使用随机分组将关联变量分到同组中的可能性将大大降低，这就要求对决策变量进行更为频繁的随机分组，从而增加算法的复杂度与计算代价，并导致算法的稳定性下降。为此，Omidvar[9]又提出 delta 分组策略，该策略通过优化每一维变量的改变值来确定其分组方式。当优化问题中含有关联变量时，该分组策略能在一定程度上缓解关联变量对解集的影响。最近，Li[10]提出基于差分分组的变量交互式学习方法，该方法可以使算法自动判断决策变量间的关联性，能较好地挖掘变量间的关联信息并据此进行分组。虽然差分分组较好地解决了关联变量的分组问题，但从差分分组的机制可以看出，该策略要求在运行优化算法前必须对决策变量逐个进行筛选，找出关联变量并对其进行分组，这无疑会致使算法的复杂度随变量个数的增加而急剧上升。以上算法大多用来解决大规模变量的单目标优化问题，且并没有应用到 MOPs 中，而 Coello[11]首次将随机分组策略应用于大变量的多目标问题，但是由于缺乏考虑关联变量的存在因素，导致很难使关联变量分配到同组中，因此该算法无法深入挖掘变量间存在的信息，从而使其性能受到影响。

现有的决策变量分解方法大多集中在单目标优化问题上，且缺乏发现变量间相关性的有效办法。而解决大规模变量问题的关键在于寻找决策变量之间的关联性。Salomon[12]研究了不同变量之间的关联性可以显著影响算法在连续域上的优化性能。尽管变量分解在求解大规模变量问题时对算法性能有着关键作用，但是由于缺少给定问题结构的先验信息，不能设计一个合适的分解策略。因此，挖掘决策变量之间的关联性从而将决策变量进行分解就成为了解决大规模变量问题的核心。

在大规模变量优化问题上，可以通过将高维的决策变量分解为低维且相关的变量组，并对各个变量组分别进行优化来求解最优解。研究表明大规模变量问题中决策变量之间存在一定的相关性。当变量规模增大时，变量间的关联性也随之加强。根据决策变量之间的关联性对变量进行分解，使关联变量尽可能地分到同一组中，将高维问题转换为低维的子问题来进行求解，从而可以尽可能地保证所求解集的质量。

鉴于此，提出一种关联变量分组的分解多目标算法(MOEAD/IVG)。MOEAD/IVG 将一个复杂高维决策变量的 MOPs 分解为一组简单低维变量的子问题，基于变量两两之间的关联性分析，将决策变量分解为相互之间关联性最小的低维变量组，并各自独立优化每个变量组所对应的子问题，从而获得最佳的全局最优解。与其他将所有决策变量视为一个整体来进行优化的 MOEAs 相比，MOEAD/IVG 具有一定的优势，能较好地搜索决策变量间的关联性，能有效地利用计算资源来提高求解的效率，并大大增加了算法的稳定性。

2 背景

2.1 多目标优化的概念

一个连续 MOPs 的数学表达式可以表示如下[13]：

$$\begin{cases}\min \boldsymbol{F}(\boldsymbol{x})=(f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_m(\boldsymbol{x}))\\ \text{subject to:}\boldsymbol{x}\in\Omega\end{cases}\tag{1}$$

其中，$\boldsymbol{F}(\boldsymbol{x}):\Omega\rightarrow\boldsymbol{R}^m$ 由 m 个需同时优化的实值连续函数组成，Ω 为决策空间中的可行域，$\boldsymbol{x}$ 为在可行域 Ω 中的决策向量。

若 $\forall i\in\{1,\cdots,m\}$，$f_i(\boldsymbol{x}_u)\leqslant f_i(\boldsymbol{x}_v)$，$\boldsymbol{F}(\boldsymbol{x}_u)\neq\boldsymbol{F}(\boldsymbol{x}_v)$，则说

明决策向量 $\boldsymbol{x}_u$ 支配 $\boldsymbol{x}_v$（表示为 $\boldsymbol{x}_u \prec \boldsymbol{x}_v$）。此外，当 $\forall i \in \{1,\cdots,m\}$，$f_i(\boldsymbol{x}_u) < f_i(\boldsymbol{x}_v)$ 时，表明 $\boldsymbol{x}_u$ 强支配 $\boldsymbol{x}_v$。当不存在一个矢量 $\boldsymbol{x} \in \Omega$ 支配 $\boldsymbol{x}^* \in \Omega$ 时，称 $\boldsymbol{x}^*$ 为 Pareto 最优解。称所有 Pareto 最优解的集合为 Pareto 最优解集（PS），即满足 $PS=\{\boldsymbol{x}^* \mid \neg \exists \boldsymbol{x} \in \Omega, \boldsymbol{x} \prec \boldsymbol{x}^*\}$。PS 在目标空间中的映射被定义为 Pareto 最优前沿（PF），即满足 $PF=\{\boldsymbol{F}(\boldsymbol{x}) \mid \boldsymbol{x} \in PS\}$。

2.2 分解策略

近几年，Pareto 排序法在多目标进化算法中应用得较普遍，将 Pareto 前沿分解为标量子问题来求解的策略也得到了广泛应用。该分解策略使用一组被视为搜索方向的权重矢量来定义标量函数，通过协同求解每个子问题的最优值来获得整个 Pareto 解集[14]。在这些分解策略中，最常见的有加权和法（Weighted Sum Approach，WS）和切比雪夫法（Tchebycheff Approach，TCH）。本文中大规模变量分组后对子问题优化时所利用的 MOEA/D 算法[15]采用切比雪夫法，即通过设置一组均匀分布的权重矢量得到分布性能较好的 Pareto 前沿，该方法的描述如下：

$$\begin{aligned} &\text{minimize } g^{tch}(\boldsymbol{x} \mid \boldsymbol{w}, \boldsymbol{z}^*) = \max_{1 \le i \le m} \{ |f_i(\boldsymbol{x}) - z_i^*| / w_i \} \\ &\text{subject to } \boldsymbol{x} \in \Omega \end{aligned} \tag{2}$$

其中，$\boldsymbol{w}=(w_1,\cdots,w_m)^{\mathrm{T}}$ 为分解后子问题中的权重矢量。$\boldsymbol{z}^*$ 为参考点，即 $\boldsymbol{z}^*=(z_1^*,\cdots,z_m^*)^{\mathrm{T}}$，且 $z_i^*=\min\{f_i(\boldsymbol{x}) \mid \boldsymbol{x} \in P\}$，$i \in \{1,\cdots,m\}$。

2.3 关联变量定义

在遗传学中，若两个基因共同影响着生物的特性，则称这两个基因是相互关联的。现有研究表明，在求解大规模变量问题时，不同变量之间也存在着相类似的关联性。如果其中一个变量发生变化将导致另一个变量也发生相应的变化，将这类变量称为不可分离变量，即关联变量。反之，如果变量之间是相互独立的，即某个变量值的改变不影响其他变量，则称其为可分离变量[10]。

定义 1 当且仅当每个决策变量 $x_i (i=1,\cdots,n)$ 能被独立优化时，称函数 $f(x)$ 为可分离函数[1]，即满足：

$$\begin{aligned} \arg\min f(x_1,\cdots,x_n) = [&\arg\min_{x_1} f(x_1,\cdots,x_i,\cdots,x_n),\cdots, \\ &\arg\min_{x_i} f(x_1,\cdots,x_i,\cdots,x_n),\cdots, \\ &\arg\min_{x_n} f(x_1,\cdots,x_i,\cdots,x_n)] \end{aligned} \tag{3}$$

否则，称 $f(\boldsymbol{x})$ 为不可分离函数。

定义 1 表明一个可分离函数可以通过优化每个决策变量来求解。变量可分离意味着每个决策变量与其他变量之间是相互独立的。可分离函数表明优化问题中的决策变量可以被独立优化，全局最优解可以通过对不同子问题进行优化而得到，不同子问题间的局部最优解不会相互影响；而不可分离函数则表明至少有两个决策变量之间存在着关联性。

在求解大规模变量问题时，决策变量间的关联性尤为重要。如果变量间的关联信息能提早获取，就可以根据变量间的关联程度将相互之间具有一定关联性的决策变量分为一组，将各自独立且不具有关联性的变量分为另一组。从而将高维变量问题分解为简单低维变量问题来求解，通过分别优化变量分组后的各个子问题来求解目标函数的最优解。

大规模变量问题中的决策变量之间往往具有一定的关联性。当 MOPs 中的变量增多时，关联变量也会随之增加。如果忽视决策变量间的这种内在联系，随意对关联变量进行分解，将导致分组得到的子问题之间不再相互独立且存在依赖性，从而降低了子问题最优解的质量，最终将影响全局最优解。因此，为了有效地对大规模变量进行分解，需要将关联变量从全局变量中区分出来，并使关联变量尽可能地被分到同一组中，以降低子问题间的依赖性。关联变量分组是一种有效的将高维变量分解为低维子问题的方式，通过“分而治之”策略来同时优化一组关联变量而不是优化所有变量，从而有效地求解全局最优解。下面给出了关联变量的数学定义。

定义 2 两个决策变量 x_i 和 x_j 是相互关联的，当且仅当存在 $\boldsymbol{x}, a_1, a_2, b_1$ 和 b_2 使下式成立[16]。

$$\begin{aligned} &f(\boldsymbol{x})|_{x_i=a_2,x_j=b_1} < f(\boldsymbol{x})|_{x_i=a_1,x_j=b_1} \wedge \\ &f(\boldsymbol{x})|_{x_i=a_2,x_j=b_2} > f(\boldsymbol{x})|_{x_i=a_1,x_j=b_2} \end{aligned} \tag{4}$$

其中，$f(\boldsymbol{x})|_{x_i=a_2,x_j=b_1} \triangleq f(x_1,\cdots,x_{i-1},a_2,\cdots,x_{j-1},b_1,\cdots,x_n)$。

若一个不可分离函数 $f(\boldsymbol{x})$ 中任意两个不同的决策变量 x_i 和 x_j 相互关联，则称函数 $f(\boldsymbol{x})$ 为完全不可分离函数。在可分离和完全不可分离函数中，也存在着许多部分可分离的函数。一般而言，不可分离的程度越大，函数越难求解最优解。

2.4 变量关联性分析

存在如下两个判断连续可微函数决策变量关联性的必要条件[10]。

定理 1 假设 $f(\boldsymbol{x})$ 为连续可微函数，若两个决策变量 x_i 和 x_j 相互关联，则 $\partial f(\boldsymbol{x})/\partial x_i$ 依赖于 x_j。

证明：假设 $\partial f(\boldsymbol{x})/\partial x_i$ 不依赖于 x_j，则 $\forall \boldsymbol{x}=(x_1,\cdots,x_j,\cdots,x_n), b_1, b_2$，满足 $(\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_1} = (\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_2}$。

对于 $\forall \boldsymbol{x}=(x_1,\cdots,x_j,\cdots,x_n), a_1, a_2, b_1, b_2$，满足：

$$\int_{a_1}^{a_2} (\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_1} \mathrm{d}x_i = \int_{a_1}^{a_2} (\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_2} \mathrm{d}x_i$$

$$f(\boldsymbol{x})|_{x_i=a_2,x_j=b_1} - f(\boldsymbol{x})|_{x_i=a_1,x_j=b_1} = f(\boldsymbol{x})|_{x_i=a_2,x_j=b_2} - f(\boldsymbol{x})|_{x_i=a_1,x_j=b_2}$$

根据定义 2，x_i 和 x_j 不相关，这与 x_i 和 x_j 相互关联相矛盾。因此，定理 1 得证。证毕。

定理 2 $f(\boldsymbol{x})$ 为连续可微函数，若两个决策变量 x_i 和 x_j 相互关联，则 $\exists \boldsymbol{x}, a_1, a_2, b_1$ 和 b_2 满足下式：

$$f(\boldsymbol{x})|_{x_i=a_2,x_j=b_1} - f(\boldsymbol{x})|_{x_i=a_1,x_j=b_1} \neq f(\boldsymbol{x})|_{x_i=a_2,x_j=b_2} - f(\boldsymbol{x})|_{x_i=a_1,x_j=b_2}$$

证明：根据定理 1，若两个决策变量 x_i 和 x_j 相互关联，则 $\partial f(\boldsymbol{x})/\partial x_i$ 依赖于 x_j。因此 $\exists \boldsymbol{x}, a_1, a_2, b_1, b_2$ 满足：

$$(\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_1} \neq (\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_2}$$

$$\int_{a_1}^{a_2} (\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_1} \mathrm{d}x_i \neq \int_{a_1}^{a_2} (\partial f(\boldsymbol{x})/\partial x_i)|_{x_j=b_2} \mathrm{d}x_i$$

即 $\exists \boldsymbol{x}=(x_1,\cdots,x_j,\cdots,x_n), a_1, a_2, b_1, b_2$ 满足：

$$f(\boldsymbol{x})|_{x_i=a_2,x_j=b_1} - f(\boldsymbol{x})|_{x_i=a_1,x_j=b_1} \neq f(\boldsymbol{x})|_{x_i=a_2,x_j=b_2} - f(\boldsymbol{x})|_{x_i=a_1,x_j=b_2}$$

证毕。

3　关联变量分组的分解多目标进化算法（MOEAD/IVG）

3.1　关联变量分组算法的框架

本文使用定义 2 来识别和对决策变量进行分组优化。算法 1 给出了两个决策变量之间关联性识别和分组的详细步骤。

关联变量分组的算法步骤如下：

步骤 1　在种群中随机选出一个个体进行关联变量识别和分组，并用定理 2 对其中第 i 个变量和第 j 个变量进行关联变量的识别，同时通过改变 a_1，a_2，b_1，b_2 的大小来获得不同目标维度上 $f_k(\boldsymbol{x}^l)$ 的值。

步骤 2　通过计算 $\Delta=|\Delta_{f_{k1}}-\Delta_{f_{k2}}|$ 求出不同目标函数下 Δ 的最大值，并进行如下判断：若 Δ 大于充分小的数 ε，则可以表明第 i 个变量和第 j 个变量之间存在一定的关联性，从而可以将它们分到同一组中；否则，说明两者之间不存在关联性。

步骤 3　重复上述过程来识别其他所有的决策变量是否与第 i 个变量存在关联性，若存在，则将它们分到同一组中。然后对与第 $i+1$ 个变量间存在关联性的决策变量进行检测和分组，直到所有的决策变量都被检测完为止。算法 1 的伪代码如下所示。

算法 1　*allgroups*←*grouping*(*pop*,*obj*,*m*,*n*)

Input: number of objective functions m, number of variables n, current evolutionary population pop ←$\{\mathbf{x}^1,\cdots,\mathbf{x}^N\}$, and their objective values obj←$\{\mathbf{F}^1,\cdots,\mathbf{F}^N\}$

1. dims←{1,2,⋯,n}
2. seps←{}
3. allgroups←{}// contains a set of all identified groups
4. FOR i∈ dims do
5. group← {i}
6. FOR j∈ dims∧i≠j do
7. Randomly select an individual $\mathbf{x}^l$ from population $\{\mathbf{x}^1,\cdots,\mathbf{x}^N\}$ and uniformly random sampling a_2, b_2 in their feasible domain, where $a_1=x_j^l$, $b_1=x_j^l$. Evaluate $\mathbf{F}(\mathbf{x}^l)$, where $\mathbf{F}(\mathbf{x}^l)|_{x_i=a_2,x_j=b_2}=\mathbf{F}(x_1^l,\cdots,x_{i-1}^l,a_2,x_{i+1}^l,\cdots,x_{j-1}^l,b_2,x_{j+1}^l,\cdots,x_n^l)$
8. FOR k=1 to m
9. $\Delta_{fk1}|_{x_i=b_1}\leftarrow f_k(\mathbf{x}^l)|_{x_i=a_2,x_j=b_1}-f_k(\mathbf{x}^l)|_{x_i=a_1,x_j=b_1}$
10. $\Delta_{fk2}|_{x_j=b_2}\leftarrow f_k(\mathbf{x}^l)|_{x_i=a_2,x_j=b_2}-f_k(\mathbf{x}^l)|_{x_i=a_1,x_j=b_2}$
11. Where $f_k(\mathbf{x}^l)|_{x_i=a_2,x_j=b_2}=f_k(x_1^l,\cdots,x_{i-1}^l,a_2,x_{i+1}^l,\cdots,x_{j-1}^l,b_2,x_{j+1}^l,\cdots,x_n^l)$
12. $\Delta=|\Delta_{fk1}-\Delta_{fk2}|$ // detect interaction between x_i and x_j for $f_k(\mathbf{x})$
13. END FOR
14. Calculate the maximum value of Δ
15. IF Δ>ε THEN
16. group←group∪j
17. END IF
18. END FOR
19. dims←dims−group
20. IF length(group) =1 then
21. seps←seps∪group
22. ELSE
23. allgroups←allgroups∪{group}
24. END IF
25. END FOR
26. allgroups←allgroups∪{seps}

算法 1 开始时先检测第一个决策变量和所有其他决策变量之间的关联性。若算法检测到第一个变量与其他某个变量存在关联性，则将该变量从决策变量集中剔除并将它放在同一组中；若不存在关联性则不进行任何操作，持续该过程直到完成第一个变量与其他所有变量之间的关联性检测，并形成第一个子集。若检测完成之后仍未发现关联变量存在，则将该变量视为一个可分离变量。对剩下的决策变量重复该检测过程直至对所有变量的关联性都检测完为止。

值得注意的是，算法 1 中 ε 的选择会在一定程度上影响算法对决策变量之间关联性识别的敏感度。为了检测 ε 对关联变量识别的敏感程度，在测试函数 UF1[17] 上分别设置 $\varepsilon=0.0001,0.0005,0.001$ 和 0.01 进行仿真实验。图 1 给出了 ε 大小的选择在测试函数 UF1 上的效果。由图 1 可知，ε 值越小，算法对决策变量间关联性的敏感度越高，对关联变量的识别能力越强，所得到的变量分组越精确，因此在下文实验中，设置 ε 的大小为 0.0001。

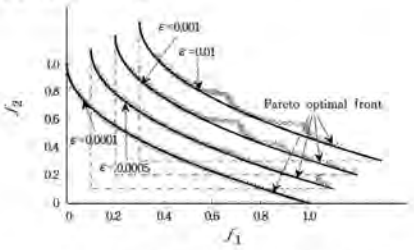


图 1　ε 对变量关联性识别敏感程度的效果演示

3.2　MOEAD/IVG 算法框架

本节将介绍关联变量分组融合到分解的多目标进化算法（MOEA/D）[15] 中形成的 MOEAD/IVG 算法的主要框架。

算法 2 详细介绍了将关联变量分组并结合 MOEA/D 的算法步骤。算法 2 主要有以下两个步骤：决策变量分组过程和分组优化阶段。在变量分组阶段中，通过识别决策变量中潜在的关联性将关联变量分到同一组中，由此形成的各个子集中分别包含与各自相关的决策变量，在一定程度上尽可能地降低了变量之间的依赖性，提高了算法的性能。在算法的分组优化过程中，通过 MOEA/D 并尽可能地利用分组后所形成的信息来分别优化分组阶段中形成的子问题。

MOEAD/IVG 算法的步骤如下：

步骤 1　设置参数大小并随机产生初始种群。

步骤 2　对初始种群进行交叉和变异操作产生子种群，并记录函数评价次数 FE。

步骤 3　根据算法 1 对进化产生的种群进行关联变量识别和分组操作。

步骤 4　利用 MOEA/D 关联变量分组产生的子问题分别优化。

步骤 5　判断是否达到函数评价的最大值，若是，则根据优化分组后子问题的最优解来得到全局最优解并输出，否则转步骤 3。

在 MOEAD/IVG 中，将关联变量分组和基于聚合函数分解策略进行协同合作来解决 MOPs。MOEAD/IVG 首先通过关联变量分组策略将大规模变量的复杂优化问题分解为一

系列简单低维变量的子问题。决策变量间关联性判断的次数越多，两个变量间关联性的判断越准确，变量分组的精度越高。然后通过 MOEA/D 来逐一协同优化每一个分组得到的子问题。算法 2 的整体流程的伪代码如下所示。

算法 2 MOEAD/IVG

Require: number of objective functions m, number of variables n, the maximal number of functions evaluations FE_{max} and the maximum number of tries required to judge the interaction between two variables N

Ensure: the optimized population pop and their objective values obj

1. Initialize the population pop←$\{\mathbf{x}^1,\cdots,\mathbf{x}^N\}$ and set FE←0
2. Evaluate these solutions obj ←$\mathbf{F}$(pop) and set FE=FE+N
3. WHILE FE< FE_{max}
4. IF time to try < N
5. Subcomponents←grouping(pop,obj,m,n) //use Algorithm 1 to put the interacting variables into the same subcomponent
6. best←min(obj)
7. FOR j=1 to size(Subcomponents)
8. indexes←Subcomponents[j]
9. subpop←pop[:,indexes]
10. subpop←SubcomponentOptimizer(subpop,best)
11. pop[:,indexes]←subpop
12. best ←min($\mathbf{F}$(pop))
13. END FOR
14. FE=FE+N
15. ELSE
16. Use MOEA/D to evolve [pop,obj]
17. FE=FE+N
18. END IF
19. END WHILE

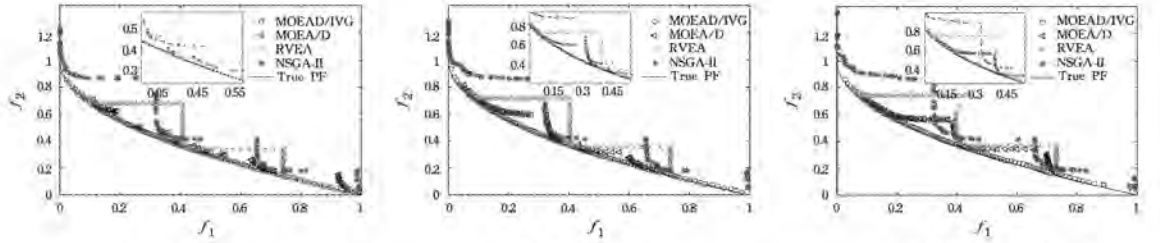
4 实验结果

为了评价本文所提算法的性能，将该算法与目前较先进的其他进化算法(包括 NSGA-II[18]，MOEA/D[15]以及最新提出的 RVEA[19])进行实验对比。本文选取的测试函数为二维测试函数 UF1，UF2，UF3 和 UF4[17]，4 个测试函数中的决策变量是相关的，且能较全面地包含各种类型的测试问题。

在仿真实验中，所有测试函数的种群大小都保持相同，均为 100。将测试函数 UF1，UF2，UF3 和 UF4 中的决策变量个数分别扩展到 100，200 和 300 进行实验。当函数评价次数达到最大值 3000000 时终止算法。每个算法均独立运行 30 次。

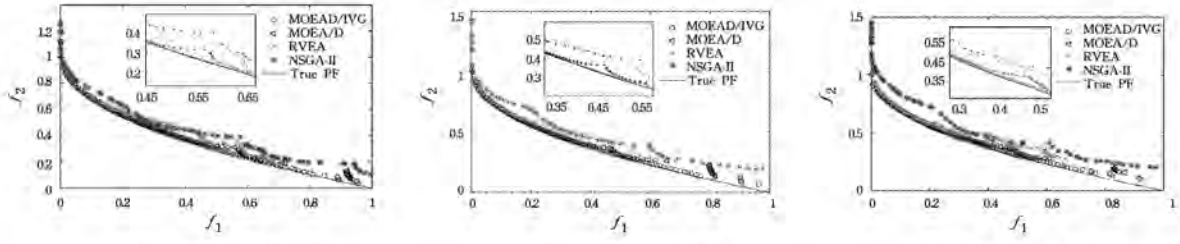
4.1 Pareto 前沿对比分析

仿真实验得到的 UF1，UF2，UF3 以及 UF4 测试函数 Pareto 前沿的对比图分别如图 2—图 5 所示。具体地，图 2 给出了 UF1 测试函数在决策变量分别为 100，200 和 300 的情况下由 NSGA-II，MOEA/D，RVEA 以及 MOEAD/IVG 得到的 Pareto 前沿对比图。同理，不同决策变量下各算法在 UF2，UF3 和 UF4 测试函数上 Pareto 前沿的对比图如图 3—图 5 所示。



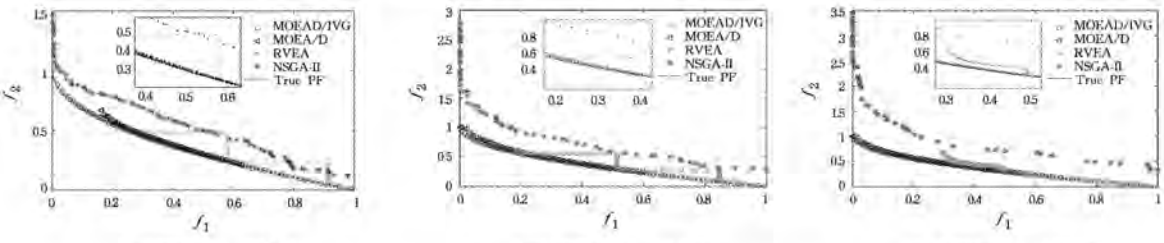
(a) UF1(100 variables) (b) UF1(200 variables) (c) UF1(300 variables)

图 2 不同决策变量下 UF1 测试函数的 Pareto 前沿对比图



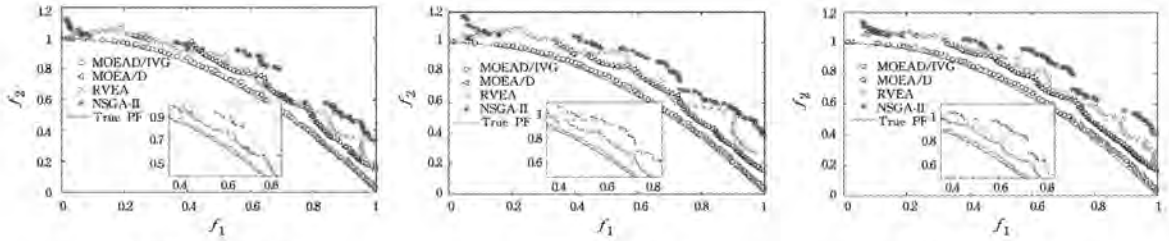
(a) UF2(100 variables) (b) UF2(200 variables) (c) UF2(300 variables)

图 3 不同决策变量下 UF2 测试函数的 Pareto 前沿对比图



(a) UF3(100 variables) (b) UF3(200 variables) (c) UF3(300 variables)

图 4 不同决策变量下 UF3 测试函数的 Pareto 前沿对比图



(a)UF4(100 variables)　(b)UF4(200 variables)　(c)UF4(300 variables)

图 5　不同决策变量下 UF4 测试函数的 Pareto 前沿对比图

从上述 Pareto 前沿对比图中可以看出，通过 MOEAD/IVG 得到的 Pareto 解集在决策变量个数为 100，200 以及 300 时，均能较好地收敛到真实的 Pareto 前沿上。在 4 种算法的比较中，MOEAD/IVG 得到的解集质量最优，其次为 MOEA/D，而 RVEA 和 NSGA-II 得到的解集质量最差。MOEA/D 在求解 UF2 和 UF3 测试函数时能够较好地收敛到真实的 Pareto 前沿上，但在求解 UF1 和 UF4 测试函数时，所得到的解集质量较差，未能收敛于 Pareto 前沿。但无论是 MOEAD/IVG 还是 MOEA/D，求得的解集质量都要好于 RVEA 和 NSGA-II。从仿真图中可以发现，当决策变量个数增加时，由 MOEA/D，RVEA 以及 NSGA-II 得到的前沿明显恶化，而由 MOEAD/IVG 得到的 Pareto 前沿无论在收敛性还是多样性上都能保持较好的性能，其所求得的解集质量都要明显优于其他 3 个算法。

4.2　算法性能分析

为评价各个算法的性能差异，本文采用世代距离(Generation Distance，GD)[30]、反向世代距离(Inverted Ge-neration Distance，IGD)[17]这两个性能指标来评价不同算法之间的性能差异。其中，GD 指标主要用来评价算法逼近 Pareto 前沿的性能，某算法所求得的 GD 指标值越小，该算法对 Pareto 前沿的逼近性能越好。IGD 指标则可用于测试算法所得 Pareto 解集的整体质量，某算法的 IGD 指标越小，该算法对 Pareto 前沿具有越好的收敛性，所求得的近似解对真实的 Pareto 前沿具有更高的拟合度。

仿真实验得到的结果如表 1、表 2 所列。表中统计了使用各个算法求解每个测试函数的性能时的均值和标准差。每个测试函数中最佳算法所对应的性能指标均用黑体标出。

在表 1 的 GD 指标对比中，MOEAD/IVG 得到的 Pareto 前沿具有最好的性能，其次为 MOEA/D，而 RVEA 和 NSGA-II 的性能最差。在决策变量为 100，200 和 300 的情况下，MOEA/D 在 UF1 测试函数上得到的 GD 指标值分别是 MOEAD/IVG 上 GD 指标值的 20.8，12.2 和 4.6 倍。同时在决策变量为 100，200 和 300 时，MOEA/D 在 UF2 测试函数上得到的 GD 指标值分别为 MOEAD/IVG 上 GD 指标值的 2.6，2.0 和 1.9 倍。对于 UF3 测试函数，在决策变量为 100，200 和 300 时，由 MOEA/D 求得的 GD 指标值分别为由 MOEAD/IVG 求得的 GD 指标值的 6.4，4.9 和 3.3 倍。在 UF4 测试函数上，当决策变量为 100，200 和 300 时，MOEA/D 得到的 GD 指标值分别是 MOEAD/IVG 所求值的 5.3，4.6 和 4.6 倍。这表明 MOEAD/IVG 在 UF1－UF4 测试函数上求得的解集质量比 MOEA/D 有较大的提高，同时说明在求解大规模变量问题时，在 MOEA/D 中引入关联变量识别和分组策略是成功的，它可以显著提高大变量优化问题下 Pareto 解集的质量，从而在一定程度上增加解集的收敛性和多样性，使求得的解尽可能地收敛到真实的 Pareto 前沿上。

表 1　GD 指标的均值和标准差

测试函数		NSGA-II	MOEA/D	RVEA	MOEAD/IVG
UF1	100	0.012797	0.002802	0.010804	**0.000135**
		0.001007	0.000531	0.000310	**0.000009**
	200	0.014957	0.004721	0.011445	**0.000388**
		0.000931	0.001992	0.000538	**0.000101**
	300	0.015053	0.005399	0.013532	**0.001185**
		0.001430	0.002452	0.000542	**0.000135**
UF2	100	0.017216	0.002162	0.005531	**0.000836**
		0.002779	0.000809	0.001366	**0.000213**
	200	0.025135	0.002616	0.006194	**0.001329**
		0.002076	0.001527	0.001504	**0.000213**
	300	0.026011	0.003090	0.006678	**0.001598**
		0.003549	0.000553	0.001690	**0.000145**
UF3	100	0.030170	0.000991	0.010112	**0.000156**
		0.009801	0.000287	0.001153	**0.000084**
	200	0.105982	0.001099	0.011041	**0.000223**
		0.016826	0.000525	0.001375	**0.000057**
	300	0.141985	0.001157	0.013361	**0.000354**
		0.003443	0.000087	0.003403	**0.000028**
UF4	100	0.013254	0.006540	0.007768	**0.001231**
		0.004376	0.000327	0.002185	**0.000103**
	200	0.016493	0.006902	0.010512	**0.001512**
		0.006461	0.000736	0.003264	**0.000136**
	300	0.017276	0.007806	0.010608	**0.001690**
		0.008149	0.000251	0.004891	**0.000178**

由于各算法执行相同大小的函数评价次数不能充分说明其求解的效率，因此为了继续研究当决策变量个数增加时各算法在求解大变量问题时的效率，比较了各算法求得的解集在满足相同 GD 指标的条件下，其函数评价次数随决策变量的变化情况。在仿真中设定各算法求得解集的 GD 指标值小于或等于 0.02 或其达到最大函数评价次数 3000000 时结束算法。对于每个测试函数，各个算法分别独立运行 30 次求函数评价次数的均值。若算法的 30 次独立运行都能在 3000000 次函数评价中获得所要求的解集，则意味着该算法在有限的函数评价次数中都能百分之百地取得满足 GD 指标值小于或等于 0.02 的解集。而在 30 次独立运行过程中，算法不能在 3000000 次函数评价中每次都取得所要求的解集，则认为其不能在有限的函数评价中都取得满足上述要求的解集。仿真实验的结果如图 6 所示。

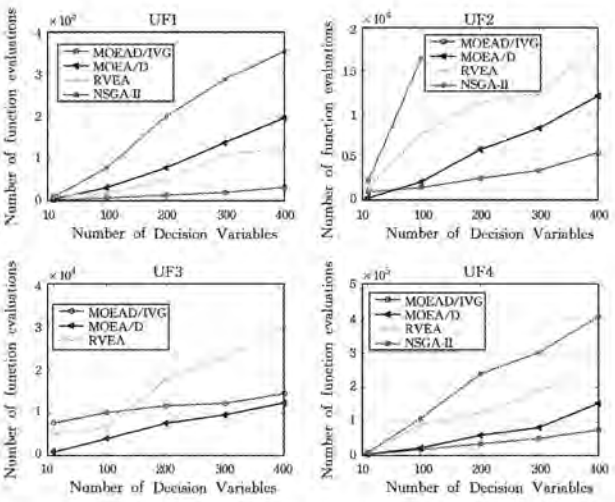


图 6 各算法函数的评价次数随决策变量的变化情况

从图 6 中可以看出在测试函数 UF1 中，当决策变量个数从 10 一直增加到 400 时，MOEAD/IVG 在相同决策变量的情况下求得所要求的解集需要的函数评价次数最少，RVEA 和 MOEA/D 次之。这表明 MOEAD/IVG 在决策变量个数从 10 增加到 400 的过程中均保持最好的收敛速度，RVEA 和 MOEA/D 次之，而 NSGA-II 最差。对于测试函数 UF2 而言，MOEAD/IVG 的收敛速度最快，MOEA/D 次之。NSGA-II 则只能在决策变量为 10 和 100 的情况下才能均取得满足 GD 指标值要求的解集，而 MOEAD/IVG 均能取得最好的收敛效果。在测试函数 UF3 中，当决策变量个数增加时，算法收敛性能优劣的排名依次为 MOEA/D、MOEAD/IVG 和 RVEA。其中，NSGA-II 不能在有限的函数评价次数中获得满足 GD 指标值要求的解集。同样在测试函数 UF4 中，可以看出 MOEAD/IVG 的收敛速度比其他 3 种算法都要快。

上述实验结果表明在大规模变量问题中，MOEAD/IVG 通过关联变量识别和分组策略可以有效地提高大变量优化问题求解的收敛速度，同时达到优化算法性能的目的。

在表 2 的 IGD 指标对比中，当决策变量分别为 100，200 和 300 时，在 UF1－UF4 测试函数中，由 MOEAD/IVG 所求得的解集的 IGD 指标值均最小，说明该算法所求解集对 Pareto 前沿具有较好的收敛性，所获解集的整体质量较高。而 MOEA/D 所求得的解集的 IGD 指标次之，RVEA 和 NSGA-II 得到的 IGD 指标最差。表 2 的对比结果表明，MOEAD/IVG 所获得的解集的 IGD 指标值均优于其他算法，说明该算法所求得的解集的整体质量最高，更逼近真实的 Pareto 前沿。同时还可以发现随着决策变量个数的增加，由 NSGA-II，MOEA/D 和 RVEA 所产生的解集的 IGD 指标值会显著增加，而 MOEAD/IVG 所求解集的 IGD 指标值仍保持较小值，且不会有明显的增加趋势。

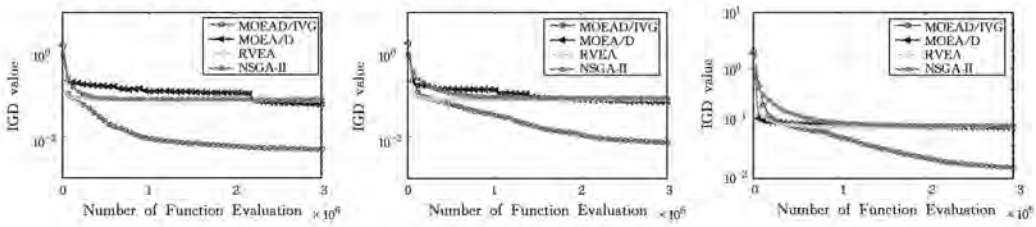
表 2 IGD 指标的均值和标准差

测试函数		NSGA-II	MOEA/D	RVEA	MOEAD/IVG
UF1	100	0.081324	0.063482	0.078255	**0.005108**
		0.000608	0.018744	0.003550	**0.001059**
	200	0.086891	0.071412	0.080089	**0.007491**
		0.001529	0.022133	0.006194	**0.001555**
	300	0.091493	0.083242	0.086047	**0.016001**
		0.001610	0.061688	0.006863	**0.002447**
UF2	100	0.064082	0.038858	0.063208	**0.010855**
		0.003071	0.011059	0.012432	**0.002864**
	200	0.086088	0.048754	0.069880	**0.014464**
		0.003298	0.027145	0.019434	**0.001636**
	300	0.100730	0.059046	0.088085	**0.019743**
		0.003914	0.018662	0.031231	**0.001687**
UF3	100	0.161416	0.104545	0.135424	**0.004322**
		0.020881	0.014093	0.006143	**0.000314**
	200	0.238027	0.127276	0.152974	**0.004633**
		0.009453	0.019323	0.006857	**0.000244**
	300	0.314078	0.135729	0.171405	**0.005436**
		0.011144	0.013152	0.007368	**0.000134**
UF4	100	0.104002	0.068432	0.067420	**0.013166**
		0.004132	0.011748	0.015982	**0.002378**
	200	0.140849	0.077674	0.088243	**0.016123**
		0.008714	0.028367	0.018735	**0.002962**
	300	0.145726	0.078755	0.094831	**0.017574**
		0.008579	0.013358	0.022943	**0.003421**

从实验数据中可以发现，通过识别关联变量并进行恰当分组可以有效地解决大规模变量问题。实验对比结果表明 MOEAD/IVG 能够提高所求解集的收敛性能，并能够较好地维护解集的多样性，增强解集收敛到 Pareto 前沿的能力，提高解集的整体质量，从而大大提高算法的整体性能。

为了更加清楚地反映算法在进化过程中所得解集 IGD 指标的变化情况，仿真了各算法在 30 次独立运行的条件下，求解全局最优解的过程中 IGD 指标均值的变化情况，实验结果分别如图 7－图 10 所示。

图 7 为在 UF1 测试函数中，不同决策变量下各算法 IGD 指标均值随其迭代的变化情况。从图 7 中可以发现，MOEAD/IVG 与其他算法相比具有较快的收敛速度和较好的性能，且求得的 IGD 指标值最小，说明获得的解集质量最佳。同时，在该算法的进化求解过程中 IGD 指标值一直在减小，说明随着算法的不断迭代可以获得解集质量显著改善的 Pareto 前沿。

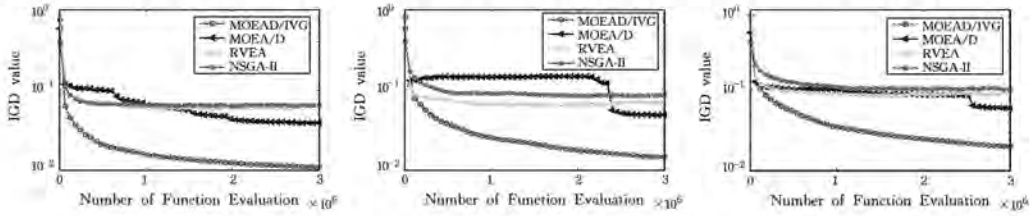


(a)UF1(100 variables) (b)UF1(200 variables) (c)UF1(300 variables)

图 7 在 UF1 测试函数下各算法 IGD 指标均值随进化过程的变化情况

在测试函数 UF2 中，从算法的收敛速度上看，MOEAD/IVG 的收敛速度最快。从所求解集的质量来看，同样是

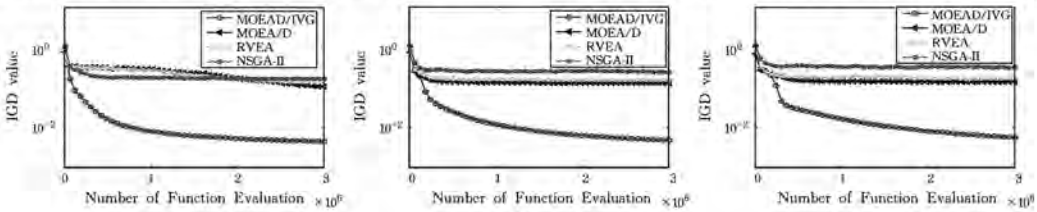
MOEAD/IVG 的性能最佳，MOEA/D 次之，RVEA 和 NSGA-II 最差。同时在各算法求解最优解的过程中，RVEA 和 NSGA-II 在进化到一定程度时，IGD 指标的减小趋势逐渐平缓，表明算法的收敛速度会逐渐变慢；而 MOEAD/IVG 的 IGD 指标值一直在下降，说明其在求解过程中随着算法的不断进化可以加快求得高质量解集的速度。



(a)UF2(100 variables)　(b)UF2(200 variables)　(c)UF2(300 variables)

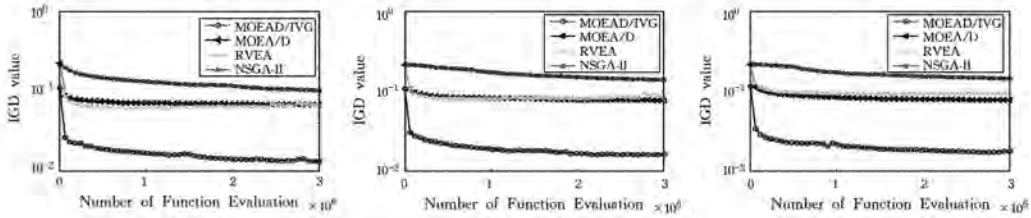
图 8　在 UF2 测试函数下各算法 IGD 指标均值随进化过程的变化情况

图 9 和图 10 给出了测试函数 UF3 和 UF4 中各算法的 IGD 指标均值在算法进化过程中的变化情况。显然从图 9、图 10 中同样可以发现，MOEAD/IVG 较其他算法在求解大规模变量问题时具有较快的收敛速度并可以获得较优质量的解集。综上，由 IGD 指标值的对比结果可以得出，NSGA-II，MOEA/D 以及 RVEA 在求解大规模变量的优化问题上所产生的解集质量要比 MOEAD/IVG 产生的解集差，表明 MOEAD/IVG 在解决大变量优化问题时具有明显的优势。



(a)UF3(100 variables)　(b)UF3(200 variables)　(c)UF3(300 variables)

图 9　在 UF3 测试函数下各算法 IGD 指标均值随进化过程的变化情况



(a)UF4(100 variables)　(b)UF4(200 variables)　(c)UF4(300 variables)

图 10　在 UF4 测试函数下各算法 IGD 指标均值随进化过程的变化情况

MOEAD/IVG 比其他算法具有更好的性能的原因在于 MOEAD/IVG 将关联变量从大规模变量中识别出来并进行分组优化，使得分组后子问题间的依赖性降到最低，从而可以逐个进行优化求解。相反，NSGA-II，RVEA 和 MOEA/D 在解决大变量问题时将所有的决策变量视为一个整体来进行优化，导致算法求解效率低下并最终影响解集质量。

为了比较算法的计算复杂度，本文统计了重复实现 30 次 4 种算法在变量分别为 100，200，300 时，UF1－UF4 测试函数运行至 3000000 次函数评价时所耗费时间的平均值（单位：分），实验结果如表 3 所列。从表 3 中可以看出，MOEAD/IVG 所花费的计算时间要小于 NSGA-II，MOEA/D 和 RVEA，而 NSGA-II 的运行时间则明显大于其他 3 种算法。同时随着优化问题中决策变量数目的增加，NSGA-II，MOEA/D 和 RVEA 所消耗的时间呈显著递增趋势，而 MOEAD/IVG 所耗时间则相对保持平稳，决策变量数目的增加对算法运行效率产生的影响相对较小。究其原因是 MOEAD/IVG 采用了关联变量分组策略，将高维变量的优化问题分解为低维变量的子问题，并利用各子问题所对应的后代解来更新自身的解，同时使用各子问题所对应的后代个体来进化当前的种群。由此，当决策变量数目增加时，经关联变量分组后，平均每组增加的变量数目相对较小，因此优化各子问题所消耗的时间也较小，从而使得 MOEAD/IVG 相比其他 3 种算法的计算复杂度明显降低。

表 3　算法运行时间/min

算法运行时间		NSGA-II	MOEA/D	RVEA	MOEAD/IVG
UF1	100	36.0	26.5	18.2	**16.1**
	200	44.4	28.7	19.2	**16.7**
	300	61.0	32.5	20.6	**17.3**
UF2	100	40.9	27.8	18.4	**16.9**
	200	44.3	34.1	20.0	**17.7**
	300	56.4	36.2	21.8	**19.3**
UF3	100	37.7	32.5	18.6	**17.2**
	200	39.7	33.8	20.5	**17.4**
	300	57.2	36.8	22.4	**18.9**
UF4	100	36.3	28.2	18.5	**16.2**
	200	43.5	29.1	19.9	**19.5**
	300	44.4	32.0	21.3	**20.5**

从上述 GD，IGD 指标和算法运行时间的对比可以看出，MOEAD/IVG 在 UF1－UF4 测试函数中均有着较大的优势，

可以获得较优的性能指标和较好的均匀分布在 Pareto 前沿上的最优解。由于 Pareto 占优在解决大规模变量问题上的局限性，NSGA-II 在处理大变量问题时，取得的性能指标均最差，且获得的 Pareto 前沿出现了较明显的恶化现象。虽然 RVEA 在解决高维目标时具有较大的优势，但是它在处理大规模变量的优化问题时，存在解集收敛效果不佳的现象。为了能够较好地解决大变量问题，MOEAD/IVG 在 MOEA/D 的基础上引入关联变量识别和分组策略，通过识别决策变量中内在的关联性，将关联变量分到同一组，将高维变量问题分解为简单低维变量的子问题，并减少所形成子问题之间的依赖性，提高了优化子问题所获解集的质量并以此来获得最佳的全局最优解。从仿真实验的对比中发现，引入关联变量识别和分组的 MOEAD/IVG 在解决大规模变量问题时有着其他算法所不具有的优势和巨大潜力，该算法在求解大变量优化问题时所获解集的整体质量更高，比其他算法有着更好的收敛性和多样性。

结束语 本文提出的关联变量分组的分解多目标进化算法，通过分析大规模决策变量中的关联信息，并通过关联变量分组使得复杂多变量的优化问题分解为简单低维变量的子问题来进行优化求解。通过分析变量之间的关联性进行分组，可以尽可能地降低子问题间的依赖性，提高解集的性能。在算法的性能指标测试中可以发现，MOEAD/IVG 在求解大规模变量问题上相比其他算法拥有较大的优势，传统的进化算法在求解大变量优化问题时将所有决策变量视为一个整体来进行优化求解，并没有识别变量之间的关联信息，这在一定程度上会降低算法性能且不能获得最优的 Pareto 解集。MOEAD/IVG 则通过分析决策变量间的关联性并利用将关联变量分到同组中的策略来分组优化子问题，可以降低分解后子问题间的依赖性，加快算法的收敛性能并获得高质量的解集。通过对标准测试函数的变量个数进行扩展后再进行仿真实验来验证算法的有效性。实验结果表明，MOEAD/IVG 有效地弥补了 MOEA/D 在解决大规模变量方面的缺陷，且与 NSGA-II，MOEA/D 和 RVEA 相比，其在求解精度和收敛速度上都有显著的提升，但实验未将其进一步应用到实际优化问题中。因此，下一步将考虑把本文中的算法应用到更多实际问题中以全面评价其性能。

参 考 文 献

[1] WEISE T, CHIONG R, TANG K. Evolutionary optimization: Pitfalls and booby traps[J]. Journal of Computer Science and Technology, 2012, 27(5): 907-936.

[2] LIU Y, YAO X, ZHAO Q F, et al. Scaling up fast evolutionary programming with cooperative coevolution[C]// Proceedings of IEEE Congress on Evolutionary Computation. Seoul, 2001: 1101-1108.

[3] YANG Z Y, TANG K, YAO X. Large scale evolutionary optimization using cooperative coevolution [J]. Information Sciences, 2008, 178(15): 2985-2999.

[4] CHEN Y P, YU T L, SASTRY K, et al. A survey of linkage learning techniques in genetic and evolutionary algorithms[R]. Urbana IL: University of IUinois at Urbana-champaign, 2007.

[5] POTTER M A, DE JONG K A. A cooperative coevolutionary approach to function optimization[C]// Proceedings of International Conference on Parallel Problem Solving from Nature. Springer Berlin Heidelberg, 1994: 249-257.

[6] YANG Z Y, TANG K, YAO X. Multilevel cooperative coevolution for large scale optimization[C]// Proceedings of IEEE Congress on Evolutionary Computation. Hong Kong, 2008: 1663-1670.

[7] OMIDVAR M N, LI X D, YANG Z Y, et al. Cooperative co-evolution for large scale optimization through more frequent random grouping[C]// Proceedings of IEEE World Congress on Computation Intelligence. Barcelona, 2010: 1754-1761.

[8] LI X D, YAO X. Cooperatively coevolving particle swarms for large scale optimization[J]. IEEE Transactions on Evolutionary Computation, 2012, 16(2): 210-224.

[9] OMIDVAR M N, LI X D, YAO X. Cooperative co-evolution with delta grouping for large scale non-separable function optimization[C]// Proceedings of IEEE World Congress on Computation Intelligence. Barcelona, 2010: 1762-1769.

[10] OMIDVAR M N, LI X D, MEI Y, et al. Cooperative co-evolution with differential grouping for large scale optimization[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(3): 378-393.

[11] ANTONIO L M, COELLO C A C. Use of cooperative coevolution for solving large scale multiobjective optimization problems [C]// Proceedings of IEEE Congress on Evolutionary Computation. Cancun, 2013: 2758-2765.

[12] SALOMON R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms[J]. BioSystems, 1996, 39(3): 263-278.

[13] LI K, ZHANG Q F, KWONG S, et al. Stable matching-based selection in evolutionary multiobjective optimization [J]. IEEE Transactions on Evolutionary Computation, 2014, 18(6): 909-923.

[14] DEB K. Multi-objective optimization using evolutionary algorithms[M]. John Wiley & Sons, 2001.

[15] ZHANG Q F, LI H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.

[16] CHEN W X, WEISE T, YANG Z Y, et al. Large-scale global optimization using cooperative coevolution with variable interaction learning[C]// Proceedings of International Conference on Parallel Problem Solving from Nature. Springer Berlin Heidelberg, 2010: 300-309.

[17] ZHANG Q F, ZHOU A M, ZHAO S Z, et al. Multiobjective optimization test instances for the CEC 2009 special session and competition[R]. Colchester: University of Essex, 2008.

[18] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[19] CHENG R, JIN Y C, OLHOFER M, et al. A reference vector guided evolutionary algorithm for many-objective optimization [J]. IEEE Transactions on Evolutionary Computation, 2016, 20(5): 773-791.

[20] VAN VELDHUIZEN D A, LAMONT G B. On measuring multiobjective evolutionary algorithm performance [C]// Proceedings of IEEE Congress on Evolutionary Computation. California, 2000: 204-211.