

基于交叉和变异的多目标粒子群算法

刘衍民^{1,2}, 牛奔³, 赵庆祯²

(1. 遵义师范学院 数学系, 贵州 遵义 563002; 2. 山东师范大学 管理与经济学院, 济南 250014;

3. 深圳大学 管理学院, 广东 深圳 518060)

(yanmin7813@163.com)

摘要:为了保证粒子群算法求得的非劣解尽可能接近真实的 Pareto 前沿并保持多样性分布,提出一种基于交叉和变异的多目标粒子群算法(CMMOPSO)。在 CMMOPSO 中,首先识别 Pareto 前沿的稀疏部分包含的粒子,并对这些粒子进行交叉操作以增加多样性分布;接着对远离 Pareto 前沿的粒子进行变异操作,以提升粒子向真实的 Pareto 前沿飞行的概率。在基准函数的测试中,结果显示 CMMOPSO 比其他算法有更好的运行效果。

关键词:多目标优化;粒子群算法;交叉;变异;外部存档

中图分类号: TP301.6 **文献标志码:** A

Multi-objective particle swarm optimization based on crossover and mutation

LIU Yan-min^{1,2}, NIU Ben³, ZHAO Qing-zhen²

(1. Department of Mathematics, Zunyi Normal College, Zunyi Guizhou 563002, China;

2. School of Management and Economics, Shandong Normal University, Jinan Shandong 250014, China;

3. College of Management, Shenzhen University, Shenzhen Guangdong 518060, China)

Abstract: In order to minimize the distance of the Pareto front produced by Particle Swarm Optimization (PSO) with respect to the global Pareto front and maximize the spread of solutions found by PSO, a multi-objective particle swarm optimization based on crossover and mutation (CMMOPSO). In the CMMOPSO, firstly, the number of particle in sparse part of Pareto front was defined and the crossover operator was employed to increase the diversity of the nondominated solutions; next, the mutation operation was used for the particles far away from Pareto front to improve the probability to fly to Pareto front. In benchmark functions, CMMOPSO achieves better solutions than other algorithms.

Key words: multi-objective optimization; Particle Swarm Optimization (PSO); crossover; mutation; external archive

0 引言

粒子群算法(Particle Swarm Optimization, PSO)^[1]是一类基于群体智能的随机优化算法。该算法操作简单,收敛速度快。鉴于此,许多学者提出将粒子群算法来处理多目标问题(Multi-objective PSO, MOPSO)。Coello 等人^[2]提出通过合并 Pareto 占优的概念,并采用外部存档控制器来存储和决定每一代中哪些粒子将会成为非劣解的成员,这些成员被用来引导其他粒子的飞行。在文献[3]中,作者提出了动态人口多种群 MOPSO,这种算法应用了自适应局部外部存档来提升每个种群的多样性,进而提升每个粒子收敛到真实 Pareto 前沿的能力。Yen 等人^[4]提出了一种类似于文献[3]的动态多种群 MOPSO 算法,通过分配一定数量的子群来保证算法在运行过程中收敛到真实的 Pareto 前沿。这些研究都基于下述两个目标:1)所得非劣解集尽可能接近真实的 Pareto 前沿;2)所得非劣解集要尽可能保持多样性分布。因此,为了更好地实现这两个目标,在文献[2]提出的多目标粒子群算法基础上,提出一种基于交叉和变异的多目标粒子群算法(Multi-Objective PSO based on Crossover and Mutation, CMMOPSO)来提升粒子群算法求解多目标问题的能力。

1 基本粒子群算法

Kennedy 等人^[1]在 1995 年首先提出粒子群算法,每个粒子根据自身的最好历史位置和全局最优的粒子位置共同调整飞行方向。通常来说,根据粒子邻居拓扑结构的不同,PSO 分为局部版本粒子群算法(Local Particle Swarm Optimization, LPSO)和全局版本粒子群算法(Global Particle Swarm Optimization, GPSO)。在 LPSO 算法中,每个粒子的邻居是由与它直接相连的那些粒子构成;而 GPSO 算法中,每个粒子的邻居由群体中除自己外的所有粒子构成。这样,GPSO 和 LPSO 算法的粒子速度和位置更新方程可分别描述为:

$$\begin{cases} \vec{v}_i(t+1) = \vec{v}_i(t) + \varphi_1 \cdot r_1 (\vec{p}_i(t) - \vec{x}_i(t)) + \\ \quad \varphi_2 \cdot r_2 (\vec{p}_g(t) - \vec{x}_i(t)) \end{cases} \quad (1)$$

$$\begin{cases} \vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \\ \vec{v}_i(t+1) = \vec{v}_i(t) + \varphi_1 \cdot r_1 (\vec{p}_i(t) - \vec{x}_i(t)) + \\ \quad \varphi_2 \cdot r_2 (\vec{p}_{neighbor_i}(t) - \vec{x}_i(t)) \end{cases} \quad (2)$$

其中: $\vec{p}_i(t)$ 表示在迭代时刻 t ,粒子 i 所经历的最好位置; $\vec{p}_g(t)$ 表示在迭代时刻 t ,群体中的粒子所经历的最好位置,称做粒子的“社会部分”; $\vec{p}_{neighbor_i}(t)$ 表示在迭代时刻 t ,粒子 i

收稿日期:2010-06-17;修回日期:2010-08-26。 基金项目:国家 863 计划项目(2008AA04A105);广东省自然科学基金资助项目(9451806001002294);贵州教育厅社科项目(0705204);遵义科技攻关项目([2008]21号)。

作者简介:刘衍民(1978-),男,黑龙江牡丹江人,讲师,博士研究生,主要研究方向:运筹学、进化计算;牛奔(1980-),男,安徽全椒人,博士,主要研究方向:进化计算;赵庆祯(1943-),男,山东利津人,教授,博士生导师,主要研究方向:运筹学、进化计算、物流工程。

的邻居中所有粒子经历的最好位置。这两种算法从本质上是相同的,只是在粒子速度更新时,学习样本的选择范围不同。

2 基于交叉和变异的多目标粒子群算法

2.1 外部存档

粒子群算法在每一次迭代都产生一组非劣解。因此,在算法运行中,采用外部存档来存储每一代产生的非劣解。随着迭代的进行,每一代所产生的非劣解用来更新外部存档;同时,为了保证产生更加接近真实 Pareto 前沿的非劣解,采用网格技术对外部存档中的非劣解进行筛选,并采用自适应网格策略控制外部存档规模。对于网格的确定方法,请参照文献[2]。

PSO 算法在求解单目标问题时,每次迭代产生一个全局最优粒子;而在求解多目标问题时,在每次迭代过程中,产生一组非劣解。在众多的非劣解中,无法确定哪一个非劣解是最优的。对于如何在非劣解中选取一个最优的,这里采用自适应网格和轮盘赌选择策略^[2]在外部存档中选择非劣解。

2.2 Pareto 前沿稀疏部分的确定

对于任何一种给定的多目标进化算法,使所得非劣解集尽可能保持多样性分布,即所得非劣解尽可能地均匀分布在 Pareto 前沿是研究目标之一。但是,由于粒子群算法是通过种群间相互学习来调整粒子的飞行方向,这样,在某种情况下会出现大量的粒子集聚一起,导致分布不均匀,出现 Pareto 前沿的稀疏部分,如图1所示。因此,识别 Pareto 前沿的稀疏部分,并在稀疏部分产生新的非劣解是提升非劣解分布均匀性的一种有效途径。式(3)、(4)给出了 Pareto 前沿的稀疏部分识别策略。

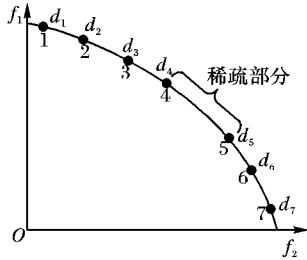


图1 Pareto 前沿稀疏部分

$$S = \{a \mid a = \text{find}(d_i > \lambda \cdot \bar{d}), i \in \text{rep}\} \quad (3)$$

$$P = \{x \mid x = \text{unique}(\bar{S}, S)\} \quad (4)$$

其中: S 表示 Pareto 前沿稀疏部分; rep 表示外部存档; d_i 表示任意相邻两点间的欧氏距离; λ 控制稀疏部分个数的参数,本文提出的算法中 $\lambda = 2.8$; \bar{d} 表示所有 d_i 的平均值; $\text{find}(\cdot)$ 函数表示寻找元素对应的位置; P 表示稀疏点集合; \bar{S}, S 表示 S 元素的端点; $\text{unique}(\cdot)$ 表示寻找所有 \bar{S}, S 端点中的不同元素。

通过对稀疏点进行扰动产生用于交叉操作的粒子,这样可以保证 Pareto 前沿的稀疏部分有更多的粒子产生,进而提升多样性分布。

$$y_{ij}^1 = x_{ij} + \frac{\bar{d}}{q} \cdot r - \frac{\bar{d}}{2q} \quad (5)$$

$$y_{ij}^2 = x_{ij} - \left(\frac{\bar{d}}{q} \cdot r - \frac{\bar{d}}{2q} \right) \quad (6)$$

其中: y_{ij}^1 和 y_{ij}^2 分别表示由稀疏点 x_{ij} 通过扰动因子 q 产生的两个粒子; i 表示粒子; j 表示维数; r 表示均值为0、方差为1的高

斯白噪声。注意如果稀疏点个数为 N ,经过式(5)、(6)扰动后,稀疏点个数变为 $2N$ 。

2.3 模拟二进制交叉

在文献[5]中,作者证实了模拟二进制交叉在帮助种群跳出局部最优解(类似于多目标问题中的伪 Pareto 前沿)起到了积极的作用。因此,对稀疏点粒子进行模拟二进制交叉以产生更多的粒子来提升多样性分布。该过程如下:

$$\begin{cases} x_{1,k} = \frac{1}{2}[(1 - \beta_k) \cdot x_{r,1,k} + (1 + \beta_k)x_{r,2,k}] \\ x_{2,k} = \frac{1}{2}[(1 + \beta_k) \cdot x_{r,1,k} + (1 - \beta_k)x_{r,2,k}] \end{cases} \quad (7)$$

其中: $x_{r,i,k}$ 是随机选择的粒子, $x_{i,k}(i = 1, 2)$ 表示在 SBC 后新产生的粒子的第 k 维, $\beta_k(\geq 0)$ 通过式(8)产生。

$$\beta_k = \begin{cases} (2u)^{1/(\eta_c+1)}, & u < 0.5 \\ [2(1 - u)]^{1/(\eta_c+1)}, & u \geq 0.5 \end{cases} \quad (8)$$

其中: u 是 $(0, 1)$ 内的随机数; η_c 定义新产生的粒子的分布指数,它等于粒子规模。

2.4 多项式变异

当一个种群停止进化时(即当种群中所有粒子的速度几乎等于零时),这将导致整个种群陷入局部最优解,而在多目标情况下,整个种群将陷入伪 Pareto 前沿。文献[6]提出通过变异来产生新的全局最优粒子使得当前种群跳出局部最优状态。因此,为了使所得非劣解集尽可能接近真实的 Pareto 前沿,对远离 Pareto 前沿的非劣解进行多项式变异,在 CMMOPSO 中,远离 Pareto 前沿的定义如下:

$$p = \{m \mid m = \text{find}(l_i > \bar{l}), i \in \text{rep}\} \quad (9)$$

其中: p 表示远离 Pareto 前沿的所有非劣解集合; rep 表示外部存档; l_i 表示每个非劣解和最接近的 Pareto 解间的欧氏距离; \bar{l} 表示所有 l_i 的平均值; $\text{find}(\cdot)$ 函数表示寻找元素对应的位置。图2给出了远离 Pareto 前沿的非劣解与最接近 Pareto 解间的示例图。

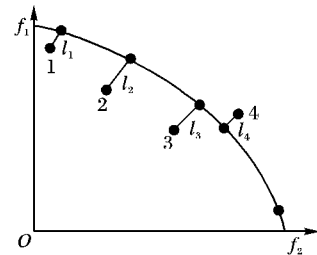


图2 非劣解与最接近 Pareto 解间距离

确定远离 Pareto 前沿的所有非劣解后,要对这些非劣解进行变异,这里采用文献[7]提出的多项式变异,变异过程如下:

$$x_i(t+1) = x_i(t) + (x_i^U(t) - x_i^L(t)) \cdot \delta_i \quad (10)$$

其中: $x_i(t)$ 表示在迭代时刻 t ,当前粒子 i 的位置; $x_i^U(t)$ 和 $x_i^L(t)$ 分别表示 $x_i(t)$ 的上界和下界; δ_i 是粒子 i 的扰动项,它的生成见式(11)。

$$\delta_k = \begin{cases} (2r_k)^{\frac{1}{\eta_m+1}} - 1, & r_k < 0.5 \\ 1 - [2 \cdot (1 - r_k)]^{\frac{1}{\eta_m+1}}, & r_k \geq 0.5 \end{cases} \quad (11)$$

其中: r_k 表示 $(0, 1)$ 内均匀分布的随机数; η_m 变异分布指数,它等于人口规模。

2.5 CMMOPSO

综合上述策略,采用全局版本粒子群算法。这样,粒子的学习样本包括两部分:向自身的最优位置学习和向全局最

优的粒子位置学习。这样粒子的进化方程由式(12)、(13)来确定:

$$\vec{v}_i(t+1) = \vec{v}_i(t) + \varphi_1 \cdot r_1(\vec{p}_i(t) - \vec{x}_i(t)) + \varphi_2 \cdot r_2(\vec{rep}_h(t) - \vec{x}_i(t)) \quad (12)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (13)$$

其中: $\vec{rep}_h(t)$ 表示在迭代时刻 t 选取的全局最优粒子, $\varphi_1 = \varphi_2 = 2$ 。

算法 1 给出了 CMMOPSO 流程。

算法 1

Initialize positions and associated velocities of all particles.

$V_{\max} = 0.25(X_{\max} - X_{\min})$.

Set the current particle position as pbest.

While (fitcount < Max_FES) && (k < iteration)

Find sparse part in Pareto front in terms of Eq. (3) (4).

Construct adaptive grid.

Produce the particles used for crossover from sparse part.

Produce the particles used for mutation.

For each particle (i = 1: ps)

Select an exemplar from external archive

Employ crossover for sparse part of Pareto front.

Employ mutation for the particles far away from Pareto front using Eq. (10).

Update velocity and position using Eq. (12) (13).

Evaluate the fitness values of the current particle i.

Update the pbest of each particle.

End for

Update external archive by Pareto dominance.

Output results

End while

3 CMMOPSO 性能测试与分析

3.1 检测函数及算法的参数设置

与本文提出的 CMMOPSO 进行比较的算法为: NSGAII^[8] 算法和 MOPSO^[2] 算法。所选取检测函数为: ZDT1、ZDT2 和 ZDT3, 它们的表达式收集在文献[3]中, 这三个检测函数分别用来测试算法解决 Pareto 最优前沿是凸、非凸和不连续的优化问题的能力。这里, 每个检测函数含有两个目标函数和 30 维变量。所有算法的迭代次数为 200, 存档规模为 100, 粒子规模为 100, 其他参数设置与各种算法提出时所用参数设置一致。

3.2 实验结果及其分析

仿真实验在型号为 ThinkPad-SL400 电脑上应用 Matlab 7.7.0(R2008b) 软件完成。实验中, 每个函数独立运行 20 次。图 3 给出了各种算法所获得 Pareto 前沿图, 其中 f_1 和 f_2 表示目标函数值。可以看出 CMMOPSO 得到的 Pareto 前沿比 MOPSO 和 NSGAII 算法更要接近真实的 Pareto 前沿, 表明 CMMOPSO 处理这类多目标问题要优于其他几种算法。

同时, 为了证明算法的有效性, 应用分布指标 (Δ)^[8] 和收敛性指标 (γ)^[2] 来评价不同算法的运行, 其中 γ 越小表示算法的收敛性越好; Δ 越小意味着所得非劣解有较好的分布。表 1 给出了不同算法在 20 次独立运行中的结果, 其中, B 表示运行最好的结果, W 表示运行最差的结果, M 表示平均值。从表 1 可以看出 CMMOPSO 在分布指标和收敛性指标方面要优于其他算法, MOPSO 和 NSGAII 几乎有相同的运行特征。

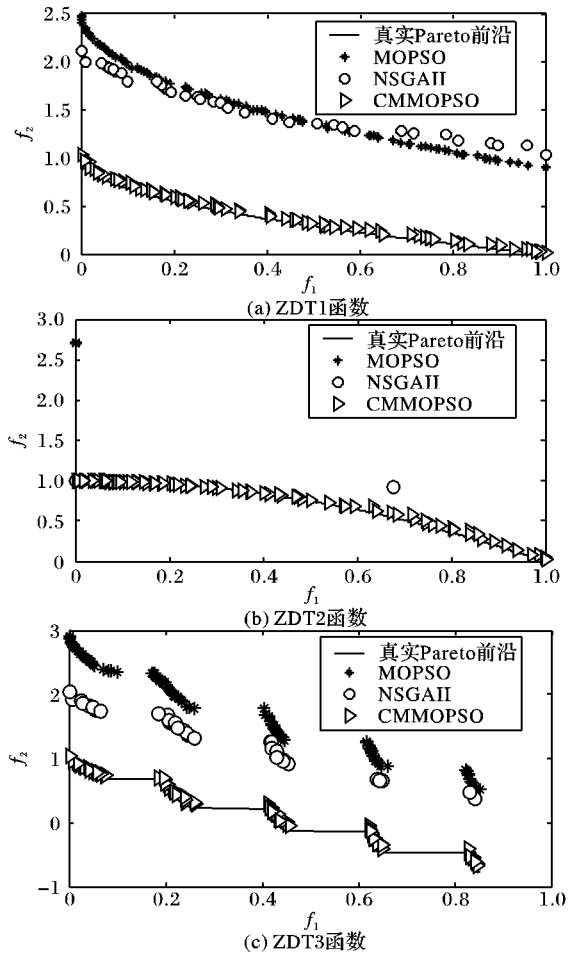


图 3 各种算法所得 Pareto 前沿

表 1 不同算法评价指标

算法		CMMOPSO		MOPSO		NSGAII	
		γ	Δ	γ	Δ	γ	Δ
ZDT1	B	0.0104	0.32	0.069	0.56	0.071	0.59
	W	0.0486	0.53	0.216	0.89	0.234	0.87
	M	0.0325	0.41	0.089	0.72	0.086	0.66
ZDT2	B	0.0128	0.36	0.126	0.63	0.198	0.63
	W	0.0463	0.69	0.603	0.83	0.619	0.81
	M	0.0293	0.43	0.381	0.74	0.352	0.69
ZDT3	B	0.0127	0.31	0.156	0.49	0.114	0.43
	W	0.0235	0.46	0.317	0.81	0.286	0.74
	M	0.0179	0.38	0.215	0.63	0.185	0.56

4 结语

考虑到所有的多目标进化算法都要满足所得非劣解集尽可能接近真实的 Pareto 前沿并且所得非劣解集要尽可能保持多样性分布, 提出一种基于交叉和变异的多目标粒子群算法 (CMMOPSO)。该算法通过对 Pareto 前沿的稀疏部分进行交叉操作以提升多样性分布, 并用过多项式变异操作增加粒子向真实的 Pareto 前沿飞行的概率。通过典型测试函数对 CMMOPSO 在处理 Pareto 前沿为凸、凹、不连续的目标空间中解分布不均匀问题进行了测试, 结果表明 CMMOPSO 是一种处理多目标问题的有效算法。

参考文献:

- [1] KENNEDY J, EBERHART R C. Particle swarm optimization [C]// Proceedings of IEEE International Conference on Neural Networks. Washington, DC: IEEE, 1995: 1942 - 1948.

$[0,1] \wedge R \in [0,0] \wedge A \in [0,0]$ 的所有请求,在删除 R_i 前这些请求的结论为 Deny,删除 R_i 后属于 $S \in [0,0] \wedge R \in [0,0] \wedge A \in [0,0]$ 的请求结果仍然为 Deny,但属于 $S \in [1,1] \wedge R \in [0,0] \wedge A \in [0,0]$ 的请求结果变为 Permit。输出删除规则 R_i 变更分析结果如下:

$S \in [0,0] \wedge R \in [0,0] \wedge A \in [0,0] \rightarrow$

Deny VS Deny

$S \in [1,1] \wedge R \in [0,0] \wedge A \in [0,0] \rightarrow$

Deny VS Permit

3.2 规则插入

根据定理2,计算规则插入的影响类似于规则删除。若策略 P 包含 n 条规则 $\langle R_1, R_2, \dots, R_n \rangle$, 假设插入规则 R 得到策略 $P' \langle R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n \rangle$, 为计算插入的影响,首先计算规则 R 在 P' 中的有限集合。然后构造 $\langle R_{i+1}, R_{i+2}, \dots, R_n \rangle$ 类似如图1的树形结构($\langle R_{i+1}, R_{i+2}, \dots, R_n \rangle$ 也可以构成一个独立策略,因为最后一条规则匹配所有前面规则未曾匹配的请求)。最后遍历树形结构检查哪一个路径修改前后的结论不同然后得知变更分析结果。

3.3 规则修改

根据定理3,计算策略中规则 R_i 修改成 R_i' 后的变更影响必需先计算 $R(R_i, P) \cap R(R_i', P')$ 和 $R(R_i, P) - R(R_i', P')$ 。

接下来讨论怎么计算它们。给定两个匹配集合 R_a 和 R_b 分别可表示成两个有效集合 $\{e_1, e_2, \dots, e_m\}$ 和 $\{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n\}$ 。可得:

$$R_a \cap R_b = \bigcup_{i=1}^m \bigcup_{j=1}^n (M(e_i) \cap M(\varepsilon_j))$$

其中 $M(e_i) \cap M(\varepsilon_j)$ 可由 $(Se_i \cap S\varepsilon_j) \wedge (Re_i \cap R\varepsilon_j) \wedge (Ae_i \cap A\varepsilon_j)$ 求得,可以看做是规则 e_i 和 ε_j 在主体、资源和动作分别的交集。

给定两个匹配集合 R_a 和 R_b 分别可表示成两个有效集合 $\{e_1, e_2, \dots, e_m\}$ 和 $\{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n\}$, R_∞ 为完整性说明规则。 $R_a - R_b$ 为策略 $\{e_1, \dots, e_m, \varepsilon_1, \dots, \varepsilon_n, R_\infty\}$ 中每一个 e_i 的有效集合的并集。

所以规则修改的影响可以用以下步骤计算。

1) 计算 R_i 在 P, R_i' 在 P' 中的有效集合。

2) 假如 R_i 和 R_i' 有相同结论,直接跳到第3)步。若结论不同,计算 $R(R_i, P) \cap R(R_i', P')$, 如果 $R(R_i, P) \cap R(R_i', P') \neq \emptyset$, 输出变更分析结果。

3) 计算 $R(R_i, P) - R(R_i', P')$ 。

4) 由规则 $\langle R_{i+1}, \dots, R_n \rangle$ 构造树图。遍历树图,检查每个路径的结论是否合变更前冲突,若有则输出变更分析结果。

4 算法实现

算法采用 Java 语言实现了一个原型系统,策略评估引擎采用 SUN PDP,设置策略导入功能以及统一变更输入界面,针对规则删除、插入、修改的变更分析算法采用独立的功能模块,并把结果输出到统一界面。测试平台采用一台 2.56 GHz 的 AMD 双核 CPU, 8 GB 内存并运行 Windows Server 2003 操作系统的 PC 机。对一个有 1000 条规则的策略中一条规则的变更分析运行 20 次,平均耗时 100 ms。

5 结语

本文分析了 XACML 策略定制中变更影响的各种可能性,归纳出变更分析理论基础,并提出规则删除、规则插入和规则修改的变更分析算法,对可能的变更结果做出预测输出分析结果,为系统管理员提供直观的决策参考,降低了策略定制中出现安全纰漏的几率。相关方法结论还可以应用到其他安全隐私策略语言。实验证明本文算法在保证正确准确情况下耗时短,可应用于各种策略语言。

参考文献:

- [1] ANDERSON A. A comparison of two privacy policy languages: EPAL and XACML[C]// Proceedings of the 3rd ACM Workshop on Secure Web Services. New York: ACM, 2006: 53-60.
- [2] 王雅哲, 冯登国. 一种 XACML 规则冲突及冗余分析方法[J]. 计算机学报, 2009, 32(3): 516-530.
- [3] PIETRO M, BRUNO C, SWAMINATHAN S, et al. XACML policy integration algorithms[J]. ACM Transactions on Information and System Security, 2008, 11(1): 1-29.
- [4] RYDER B, TIP F. Change impact analysis for object-oriented programs[C]// Proceedings of the 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. New York: ACM, 2001: 46-53.
- [5] LIU A, CHEN F, HWANG J, et al. Xengine: A fast and scalable XACML policy evaluation engine[C]// Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. New York: ACM, 2008: 265-276.
- [6] LIU A, GOUDA M. Complete redundancy detection in firewalls[C]// Proceedings of the 19th Annual IFIP Conference on Data and Applications Security. Berlin: Springer-Verlag, 2005: 196-209.
- [7] COELLO C A G, PULIDO G T, LECHUGA M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 256-279.
- [8] LEONG W-F, YEN G G. PSO-based multiobjective optimization with dynamic population size and adaptive local archives[J]. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, 2008, 38(5): 1270-1293.
- [9] YEN G G, LEONG W-F. Dynamic multiple swarms in multiobjective particle swarm optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2009, 39(4): 890-911.
- [10] MARTINEZ S Z, COELLO C A G. Hybridizing an evolutionary algorithm with mathematical programming techniques for multi-objective optimization[C]// Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference. New York: ACM, 2008: 769-770.
- [11] STACEY A, JANCIC M, GRUNDY I. Particle swarm optimization with mutation[C]// CEC '03: Proceedings of IEEE Congress on Evolutionary Computation. Washington, DC: IEEE, 2003: 1425-1430.
- [12] DEB K, GOYAL M. A combined genetic adaptive search (GeneAS) for engineering design[J]. Computer Science and Informatics, 1996, 26(4): 30-45.
- [13] DEB K, PRATAP A, AAGRAWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

(上接第84页)