第35卷 第13期 计 算 机 工 程 2009年7月
Vol.35 No.13 Computer Engineering July 2009
·人工智能及识别技术· 文章编号：1000—3428(2009)13—0179—04 文献标识码：A 中图分类号：TP301.6

基于单纯形算子的混合差分进化算法

刘 洁[1]，吴亮红[2]，刘建勋[3]
(1. 湖南工程学院设计艺术学院，湘潭 411104；2. 湖南科技大学信息与电气工程学院，湘潭 411201；
3. 湖南科技大学计算机科学与工程学院，湘潭 411201)

摘 要：针对DE/rand/1/bin方案收敛速度慢的缺点，提出一种将单纯形确定性算法和差分进化随机搜索算法相结合的混合优化算法。利用差分进化算法搜索范围广、全局搜索能力强和单纯形算法局部搜索能力强、收敛速度快的特性，较大地提高了差分进化算法的收敛速度和搜索精度。典型Benchmarks复杂函数优化实验表明，该算法优化效率高、优化性能好、对初值具有较强的鲁棒性，性能优于单一的优化方法。

关键词：复杂非线性函数；差分进化算法；单纯形法；混合优化算法

Hybrid Differential Evolution Algorithm Based on Simplex Operator

LIU Jie[1], WU Liang-hong[2], LIU Jian-xun[3]
(1. School of Design Art, Hunan Institute of Engineering, Xiangtan 411104;
2. School of Information and Electric Engineering, Hunan University of Science and Technology, Xiangtan 411201;
3. School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201)

【Abstract】Aiming at the slow convergence of DE/rand/1/bin strategy, this paper presents a hybrid optimization algorithm named SMDE incorporated Simplex Method(SM) into Differential Evolution(DE) algorithm. It takes use of good global searching ability of DE and good local searching ability and fast convergence of SM, so that the convergence speed and solution precision of DE are improved. Experimental results on several classical Benchmarks complex functions show that the hybrid optimization algorithm is effective, efficient and fairly robust to initial conditions, and its performances excels those single optimization methods.

【Key words】complex nonlinear function; Differential Evolution(DE) algorithm; Simplex Method(SM); hybrid optimization algorithm

1 概述

进化算法是模拟自然界生物群体进化过程的随机优化方法，具有不依赖于问题模型、寻优过程的自适应性、隐含的并行性及解决复杂非线性问题的鲁棒性等优点，已广泛应用于复杂优化问题和优化设计中。差分进化(Differential Evolution, DE)算法[1]是一种采用浮点矢量编码在连续空间中进行随机搜索的优化算法。DE原理简单，受控参数少，实施随机、并行、直接的全局搜索，易于理解和实现，是一种非常有效的启发式进化算法，在许多复杂优化问题中得到了应用[2-3]。然而，与其他进化算法(如遗传算法、粒子群优化算法)一样，差分进化算法也存在全局搜索能力与收敛速度之间的矛盾。通过改变控制参数可以提高算法的收敛速度，但将导致全局搜索能力下降，得不到满意的优化解。由于算法本身操作算子的原因，仅通过调节控制参数往往无法达到搜索和开发能力之间的平衡。为了解决这一问题，人们将全局优化算法和局部搜索策略进行集成，充分发挥各种算法的优势，构成混合优化算法。单纯形算法(Simplex Method, SM)是一种利用目标函数值的简单且具有良好局部搜索能力的优化算法，因此，很多学者将单纯形算法嵌入到进化算法中，构成混合优化算法。文献[4]提出了遗传单纯形混合算法，提高了遗传算法的收敛速度；文献[5]将模拟退火算法和单纯形算法相结合，提高算法的优化能力；文献[6]将单纯形法算子引入微粒群算法中，构成单纯形微粒群混合优化算法。

本文将单纯形算法与差分进化算法的DE/rand/1/bin方案相结合，提出了一种单纯形差分进化混合算法(SMDE)。该混合算法融合了DE/rand/1/bin良好的全局搜索能力和单纯形算法快速的局部搜索能力，在保证全局搜索能力的条件下，大大提高了算法的搜索效率。

2 单纯形差分进化混合算法

2.1 差分进化算法

DE算法是由 NP 个 n 维参数矢量 $\boldsymbol{X}_i^t=(x_{i1}^t,x_{i2}^t,\cdots,x_{in}^t)$，$i=1,2,\cdots,NP$ 构成一代种群在搜索空间进行寻优，其中，NP 为种群规模；n 为问题的决策变量个数；t 表示第 t 代。初始种群在参数空间中随机产生，并且应覆盖整个参数空间。在进化的每一代，对每个目标个体进行变异和交叉操作以产生试验个体，然后目标个体与试验个体进行选择操作，选择适应值更优的个体进入下一代。差分进化算法试验个体的产生有多种方案，本文选择有良好全局搜索能力且应用广泛的DE/rand/1/bin方案。

根据DE/rand/1/bin方案，对于每一目标个体 $\boldsymbol{X}_i^t$，对应

基金项目：国家自然科学基金资助项目(60673119)；国家“863”计划基金资助项目(2006AA04Z152)；湖南省科技计划基金资助项目(2006GK3071)；南京大学计算机软件新技术国家重点实验室开放基金资助项目

作者简介：刘 洁(1982－)，女，硕士，主研方向：智能优化算法；吴亮红，博士研究生；刘建勋，教授、博士

收稿日期：2008-07-22 **E-mail：**liujie227@gmail.com

的变异矢量 $\boldsymbol{V}_i^{t+1}=\{v_1^{t+1},v_2^{t+1},\cdots,v_n^{t+1}\}$ 由下式确定:

$$\boldsymbol{V}_i^{t+1}=\boldsymbol{X}_{r1}^t+F\times(\boldsymbol{X}_{r2}^t-\boldsymbol{X}_{r3}^t) \tag{1}$$

其中，F 为大于 0 的实数，称为变异常数，控制差分矢量 $(\boldsymbol{X}_{r2}^t-\boldsymbol{X}_{r3}^t)$ 的缩放以避免搜索的停滞。根据文献[1]，F 的取值范围为[0, 2]，在实际使用过程中，F 的取值一般为[0, 1]。$r1$, $r2$, $r3$ 为互不相同的个体索引，随机取自于种群集 $\{1,2,\cdots,NP\}$。同时这些索引与当前目标个体索引 i 不相同，所以，种群规模不能少于 4。

变异之后进行交叉操作，以产生试验个体。对于每一个变异矢量 $\boldsymbol{V}_i^{t+1}$，根据式(2)生成一个试验个体 $U_i^{t+1}=\{u_1^{t+1},u_2^{t+1},\cdots,u_n^{t+1}\}$。

$$u_{ij}^{t+1}=\begin{cases}v_{ij}^{t+1} & rand(j)\leqslant CR \ \text{ or } \ j=randn(i)\\ x_{ij}^t & rand(j)>CR \ \text{ and } \ j\neq randn(i)\end{cases} \tag{2}$$

其中，j=1, 2, …, n；$rand(j)$ 为[0, 1]之间均匀分布的随机数；CR 为交叉概率常数，其取值范围为[0, 1]。$randn(i)\in(1,2,\cdots,n)$ 是一随机选择的变量索引，保证 $\boldsymbol{U}_i^{t+1}$ 至少有一维变量由 $\boldsymbol{V}_i^{t+1}$ 贡献，否则可能没有新的个体产生而使整个种群不发生任何变化，导致搜索陷入停顿。

DE 采用“贪婪”的搜索策略，经过变异与交叉操作后生成的试验个体 $\boldsymbol{U}_i^{t+1}$ 与 $\boldsymbol{X}_i^t$ 进行竞争，只有当 $\boldsymbol{X}_T^i$ 的适应度较 $\boldsymbol{X}_i^t$ 更优时才被选作子代，否则，直接将 $\boldsymbol{X}_i^t$ 作为子代。选择操作的方程为

$$\boldsymbol{X}_i^{t+1}=\begin{cases}\boldsymbol{U}_i^{t+1} & f(\boldsymbol{U}_i^{t+1})<f(\boldsymbol{X}_i^t)\\ \boldsymbol{X}_i^t & \text{otherwise}\end{cases} \tag{3}$$

2.2 单纯形算法

直接用于最优化计算的单纯形算法有 2 种形式：(1)有约束的单纯形法，即线性规划中使用的单纯形法；(2)无约束的单纯形法。本文使用的是后一种单纯形法——Nelder-Mead 单纯形法[7]。单纯形法是一种直接寻优方法，不需要求导数，局部搜索能力强。该方法首先在 n 维空间 R^n 中构造一个具有 n+1 个顶点的多面体，求出各顶点的适应值，并确定其中的最优点、次差点和最差点，然后通过反射、扩张、收缩或压缩等策略找出一个较好点，取代最差点，从而构成新的多面体，这样重复迭代可以找到或逼近一个最优点。单纯形法的步骤描述如下：

Step1 初始化。给定初始单纯形，其顶点 $x_i\in R^n$，i=1, 2,…, n，反射系数 $\delta>0$，扩展系数 $\gamma>1$，压缩系数 $\beta\in(0,1)$，收缩系数 $\lambda\in(0,1)$，允许误差 $\varepsilon>0$，计算函数值 $f(x_i)$, i = 1, 2,…, n，置 k = 1。

Step2 确定最优点 x_b、次差点 x_g、最差点 x_w、其中，$b,g,w\in\{1,2,\cdots,n\}$，计算除 x_b 外的 n-1 个点的形心 $\bar{x}$，即 $\bar{x}=\frac{1}{n-1}(\sum_{i=1}^n x_i-x_w)$，并计算 $f(\bar{x})$。

Step3 进行反射，令 $x_{n+1}=\bar{x}+\alpha(\bar{x}-x_w)$，计算 $f(x_{n+1})$。

Step4 若 $f(x_{n+1})<f(x_b)$，进行扩展：$x_{n+2}=\bar{x}+\gamma(x_{n+1}-\bar{x})$，计算 $f(x_{n+2})$，转 Step5；若 $f(x_b)\leqslant f(x_{n+1})\leqslant f(x_g)$，置 $x_w=x_{n+1}$，转 Step7；若 $f(x_{n+1})>f(x_g)$，进行压缩，令 $f(x_p)=\min\{f(x_w),f(x_{n+1})\}$，$p\in\{w,n+1\}$；令 $x_{n+3}=\bar{x}+\beta(x_p-\bar{x})$，计算 $f(x_{n+3})$，转 Step6。

Step5 若 $f(x_{n+2})<f(x_{n+1})$，置 $x_w=x_{n+2}$、$f(x_w)=f(x_{n+2})$，转 Step7；否则，置 $x_w=x_{n+1}$、$f(x_w)=f(x_{n+1})$，转 Step7。

Step6 若 $f(x_{n+3})\leqslant f(x_p)$，则 $x_w=x_{n+3}$、$f(x_w)=f(x_{n+3})$，进行 Step7；否则进行收缩，令 $x_i=x_i+\lambda(x_b-x_i)$，进行 Step7。

Step7 若 $\sqrt{\frac{1}{n}\sum_{i=1}^n(f(x_i)-f(\bar{x}))^2}\leqslant\varepsilon$，则停止计算，当前最好点可作为极小点的近似；否则，置 k=k+1，返回 Step2。

每一次单纯形操作将改良最差解 x_w，使得 x_w 朝目标函数值更优的方向改变，更新的 n 个顶点将作为下一次迭代的初始单纯形。图 1 描述了在二维空间中以 x_1, x_2, x_3 为出发点的单纯形基本操作。

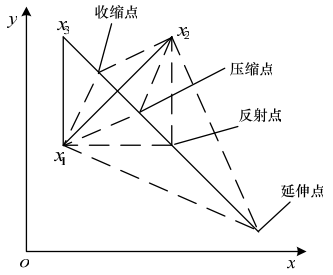

图 1　单纯形算法基本操作

2.3 单纯形差分进化算法

对于 DE/rand/1/bin 变异方案，由式(1)可知，变异个体 $\boldsymbol{V}_i^t$ 由 3 个互不相同的随机个体组成，无需任何适应度值信息，有利于保持种群的多样性，因此，全局搜索能力强，但收敛速度很慢。同时，DE 算法的每一代种群都做随机搜索，搜索时极有可能在最优解附近由于搜索不当而错过最优解。此时若在最优解附近有几步精度较高的局部搜索，则找到最优解的概率会大大增加。因此，当整个种群中最优个体连续 l 代(如 5 代)没有改进时，以当前种群最优个体为中心，按式(4)随机生成一初始单纯形进行局部搜索，并以搜索的结果替代当前种群中最差个体，从而加速进化过程。

$$x_{gbest,d}^t=x_{gbest,d}^t\times(1+0.5\eta) \tag{4}$$

其中，$x_{gbest,d}^t$ 为 x_{gbest}^t 的第 d 维取值；η 是服从 Gauss(0,1)正态分布的随机变量。

若在连续 l 代内当前种群最优个体所有改进，说明差分进化算法正在引导种群向全局最优解逼近，不必加入局部搜索策略而破坏其进化进程。这就是单纯形差分进化混合算法的基本原理，算法实现步骤如下。

Step1 初始化种群规模 NP、差分变异矢量收缩因子 F、交叉概率 CR，在参数空间随机初始化每一个个体，设置最大迭代次数 T，令 t=1。初始化单纯形的反射系数 $\delta>0$，扩展系数 $\gamma>1$、压缩系数 $\beta\in(0,1)$、收缩系数 $\lambda\in(0,1)$、允许误差 $\varepsilon>0$。

Step2 计算每个个体的适应度 $fitness$，求出当前种群中最优适应度 $bestfitness$ 及最优个体 x_{gbest}^t。

Step3 判断最优适应度 $bestfitness$ 是否达到精度要求或迭代次数是否达到最大，若是，则退出，否则，执行下一步。

Step4 若当前最优适应度 $bestfitness$ 连续 l 代没有改进，执行 Step5，否则，执行 Setp7。

Step5 以当前最优个体为中心，按高斯分布随机生成 n+1 个顶点的初始单纯形。

Step6 启动单纯形算法，并将搜索结果替代当前种群中最差个体。

Step7 对 $\boldsymbol{X}_i^t$(i=1,2,…,NP)执行 Setp8~Setp10，生成第 t+

1 代种群。

Step8 在种群中随机选择 4 个不同的个体，按式(1)进行变异操作，生成变异个体 $\boldsymbol{V}_i^{t+1}$ 。

Step9 按式(2)进行交叉操作，生成试验个体 $\boldsymbol{U}_i^{t+1}$ 。

Step10 按式(3)进行选择操作，生成 t+1 代个体 $\boldsymbol{X}_i^{t+1}$ 。

Step11 t=t+1，返回 Step2。

3 实验结果

为验证本文算法的有效性，下面通过 5 个典型的 Benchmarks 函数进行测试，同时与基本差分进化算法和单纯形算法进行比较。f_1 为 Ackley 函数，由于带有指数项，存在大量局部最优解。f_2 为 Schaffer 函数，在离全局极大点大约 3.14 的范围内存在无限多的局部极大点，函数强烈震荡的性态使其很难全局最优化。f_3 为 Rosenbrock 函数，是一个非凸、病态函数。f_4 为 Rastrigin 函数，多峰，在 S={$x_i \in$ (-5.12, +5.12), i=1,2,⋯,n}范围内大约存在 10n 个局部极小点。f_5 为 Griewank 函数，多峰，存在大量局部极小点。

$$f_1 = 20 + \mathrm{e} - 20\exp(-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}) - \exp(\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)),$$
$$-32.768 \leqslant x_i \leqslant 32.768 \tag{5}$$

$$f_2 = 0.5 - \frac{(\sin^2\sqrt{x_1^2 + x_2^2} - 0.5)}{(1 + 0.001(x_1^2 + x_2^2))^2},\ -100 \leqslant x_1, x_2 \leqslant 100 \tag{6}$$

$$f_3 = \sum_{i=1}^{30}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2],\ -30 \leqslant x_i \leqslant 30 \tag{7}$$

$$f_4 = \sum_{i=1}^{30}(x_i^2 - 10\cos(2\pi x_i) + 10),\ -5.12 \leqslant x_i \leqslant 5.12 \tag{8}$$

$$f_5 = \frac{1}{4\ 000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1,\ -50 \leqslant x_i \leqslant 50 \tag{9}$$

实验中单纯形参数设置如下：反射系数 $\delta = 1.0$ ，扩展系数 $\gamma = 2.0$ ，压缩系数 $\beta = 0.5$ ，收缩系数 $\lambda = 0.5$ ，允许误差 $\varepsilon = 0.001$ 。差分进化算法的相关参数如表 1 所示。

表 1 差分进化算法实验参数

函数	种群规模	变异常数 F	交叉概率常数 CR	最大迭代次数
f_1	20	0.5	0.1	50
f_2	20	0.5	0.1	200
f_3	50	0.5	0.1	600
f_4	50	0.5	0.1	600
f_5	50	0.5	0.1	500

为避免随机性的影响，每种算法对每个函数均独立运行 50 次，统计结果如表 2~表 4 所示。

表 2 SMDE 运行 50 次的函数优化结果

函数	最优值	平均值	均方差
f_1	2.097 1e-6	2.997 7e-5	2.525 8e-5
f_2	1	0.998 7	0.003 4
f_3	2.594 0e-4	0.002 9	0.001 9
f_4	6.951 4e-7	2.051 2e-5	2.586 2e-5
f_5	2.012 2e-9	1.748 1e-7	2.449 3e-7

表 3 DE 运行 50 次的函数优化结果

函数	最优值	平均值	均方差
f_1	1.337 0e-6	7.613 2e-5	9.634 4e-5
f_2	0.999 8	0.996 3	0.003 5
f_3	61.474 0	125.825 0	32.759 2
f_4	27.201 0	33.555 9	3.459 7
f_5	2.337 2e-5	1.577 3e-4	1.355 3e-4

表 4 SM 运行 50 次的函数优化结果

函数	最优值	平均值	均方差
f_1	4.959 6e-4	7.969 4	8.496 3
f_2	0.921 7	0.570 0	0.117 7
f_3	20 005	2.653 9e+6	5.078 2e+6
f_4	195.68	270.370 5	34.655 5
f_5	0.251 5	1.036 0	0.383 4

图 2~图 6 为 50 次实验平均最优适应度进化曲线，为便于比较，除图 3 之外，其他图中纵坐标都采用适应度的对数值。同时，由于 SM 是一种确定性算法，而且与初始点密切相关，不便于用图形表示，因此只给出 DE 和 SMDE 的进化曲线。

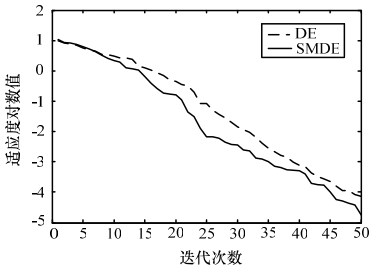

图 2 $f_1(x)$ 50 次平均最优适应度进化曲线

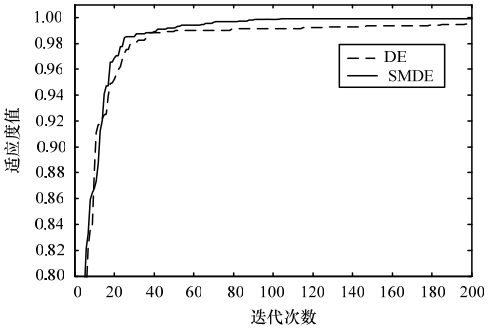

图 3 $f_2(x)$ 50 次平均最优适应度进化曲线

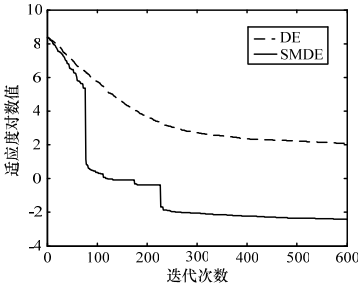

图 4 $f_3(x)$ 50 次平均最优适应度进化曲线

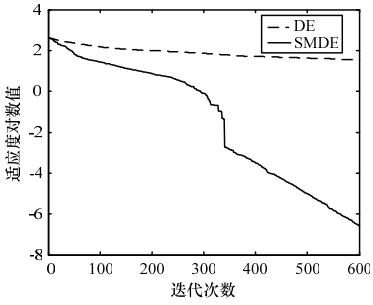

图 5 $f_4(x)$ 50 次平均最优适应度进化曲线

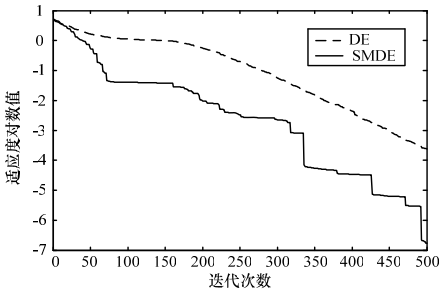

图 6 $f_5(x)$ 50 次平均最优适应度进化曲线

从表 2 可以看出，对于函数 f_1，由于函数维数和复杂度均不高，SMDE 和 DE 只需 50 次迭代就能收敛到全局最优解，解的精度也很高，但 SMDE 的收敛速度明显比 DE 快很多；SM 受初始解的影响很大，实验中其初始解是在参数空间中随机生成的，当生成的初始解落在全局最优解附近时，SM 能以较高的精度收敛到全局最优解，但绝大多数情况下都不能收敛到全局最优解。f_2 虽然维数不高，但无穷多的局部极大点将全局最优点包围，SM 无法得到最优解；DE 大多数情况下能收敛到全局最优解，但 50 次实验中有 11 次没有收敛到全局最优点；SMDE 在绝大多数情况下能收敛到全局最优解，而且解的精度比 DE 高得多，收敛速度也比 DE 快，但 50 次实验中有 3 次收敛到局部最优解。对于 f_3 和 f_4 这种高维复杂函数，SM 无法收敛到全局最优解；在给定的迭代次数下，对于 f_4，DE 无法收敛到全局最优解，但增加迭代次数至 1 500 可以收敛到全局最优解，而对于 f_3，增加迭代次数也不能收敛到全局最优解，必须调整控制参数；SMDE 对于 f_3 和 f_4 均能以很高的精度和速度收敛到全局最优解，从图中可以清楚地看到 SM 对 DE 的加速作用。对于 f_5，DE 和 SMDE 均能收敛到全局最优点，但从图 6 中可以看出，SMDE 的收敛速度和精度明显高于 DE；SM 对于高维复杂函数 f_5 不能收敛到全局最优解。另外，在进化的初始阶段，由于 DE 向全局最优解逼近较快，没有出现停滞的现象，SMDE 中 SM 没有发挥作用，因此 DE 和 SMDE 在初始阶段有相似的进化曲线。但随着 DE 连续多代(实验中设置为 5 代)没能对当前种群最优解改进，SM 对当前最优解进行局部搜索并将搜索结果替代当前种群中最差解，因此，SMDE 的收敛速度明显加快。同时，SM 的局部搜索使得全局最优解的精度更高。

4 结束语

DE/rand/1/bin 方案具有很强的全局搜索能力，实际中应用得较多，但由于具有随机性，因此收敛速度很慢，在全局最优点附近的精细搜索效果也不好。为加快 DE 算法的收敛速率，提高其精度，本文结合 SM 确定性搜索和 DE 算法全局随机性搜索的优点，提出一种高效易实现的混合优化算法 SMDE。对高维多模态复杂函数的优化表明，SMDE 的搜索精度和收敛速度明显优于 DE，具有很强的初值鲁棒性和很高的优化效率，可广泛应用于各种实际工程优化问题中。

参考文献

[1] Storn R, Price K. Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces[Z]. International Computer Science Institute, 1995.

[2] Vesterstrom J, Thomsen R. A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems[C]//Proceedings of CEC'04. Oregon, Portland: IEEE Press, 2004.

[3] 吴亮红, 王耀南, 周少武, 等. 双群体伪并行差分进化算法研究及应用[J]. 控制理论与应用, 2007, 24(3): 453-458.

[4] 何大阔, 李延强, 王福利. 基于单纯形算子的混合遗传算法[J]. 信息与控制, 2001, 30(3): 276-279.

[5] 曹志国, 汪 勇. 基于模拟退火-单纯形法的目标函数的优化[J]. 华中科技大学学报, 2005, 33(6): 67-69.

[6] 陈国初, 俞金寿. 单纯形微粒群优化算法及应用[J]. 系统仿真学报, 2006, 18(4): 862-866.

[7] Nelder J A, Mead R. A Simplex Method for Function Minimization[J]. Computer Journal, 1965, 7(4): 308-313.

编辑 张 帆

（上接第 178 页）

码成 39 维的特征矢量，包括 13 维 MFCC 系数及其一阶、二阶差分，同时训练了不同高斯分量数的 GMM 模型。

实验通过分别计算基线系统、SIMD 改造后的系统在识别过程中所用到的似然率计算时间来评测 SIMD 算法的性能。所有实验结果都是在一台 1 GB 内存、2.8 GHz 主频的 Intel Pentium 4 微机上测得的。其系统平台为 Fedora Core 4，并安装了 Intel C++ Compiler 9.1.047 编译器。实验中测试了高斯分量数从 8~64 的 GMM 模型。表 1 给出了实验结果，图 1 中为不同高斯分量数的情况下 SIMD 算法的加速比。

表 1 似然率计算算法时间对比

高斯分量个数	算法耗时/s	
	SIMD	Baseline
8	190.08	150.33
16	310.77	394.45
32	583.02	745.56
64	1 105.64	1 417.87

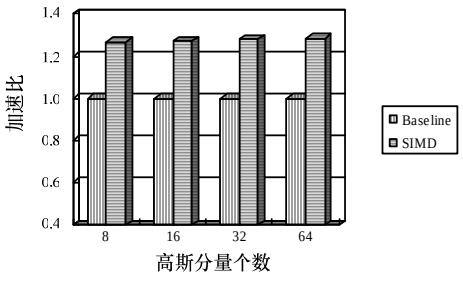

图 1 不同高斯分量的加速比

从图 1 可以看出，SIMD 算法可以提高速度 27%左右，从而节省识别过程所需要的时间开销，提高系统的实时性能。

6 结束语

本文在分析基于 HMM 的连续语音识别系统中似然率计算的基础上，提出一种基于 SSE 的似然率快速计算算法，该算法与其他似然率快速算法[5]不同，它不对似然率计算的数学模型进行修改或简化，而是利用现代微机体系结构中的硬件级并行处理方式来实现似然率计算的并行性，因而是在不影响系统的识别率的前提下加快似然率计算。实验结果表明本算法可以提高似然率计算速度，从而提高系统的实时性能。本算法在高性能微处理机日益普及的背景下具有较大的实际应用价值。

参考文献

[1] Gales M J F. State-based Gaussian Selection in Large Vocabulary Continuous Speech Recognition Using HMM's[J]. IEEE Trans. on Speech and Audio Processing, 1999, 7(2): 152-161.

[2] Kanthac S. Using SIMD Instructions for Fast Likelihood Calculation in LVCSR[C]//Proc. of ICASSP'00. Istanbul, Turkey: [s. n.], 2000.

[3] Lee Y. Mobile CPU-based Optimization of Fast Likelihood Computation for Continuous Speech Recognition[C]//Proc. of ICASSP'07. Honolulu, USA: [s. n.], 2007.

[4] 王炳锡, 屈 丹, 彭 煊. 实用语音识别基础[M]. 北京: 国防工业出版社, 2005.

[5] Pellom B L. Fast Likelihood Computation Techniques in Nearest-Neighbor-based Search for Continuous Speech Recognition[J]. IEEE Signal Processing Letters, 2001, 8(8): 221-224.

编辑 陈 文