第二章 基本微粒群算法

2.1 引言

自然界中各种生物体均具有一定的群体行为，而人工生命的主要研究领域之一就是探索自然界生物的群体行为从而在计算机上构建其群体模型。通常，群体行为可以由几条简单的规则进行建模，如鱼群、鸟群等。虽然每一个个体具有非常简单的行为规则，但群体的行为却非常复杂。Reynolds 将这种类型的个体称为 boid，并使用计算机图形动画对复杂的群体行为进行仿真。他在仿真中采用了下列三条简单规则：

（1）飞离最近的个体，以避免碰撞

（2）飞向目标

（3）飞向群体的中心

群体内每一个体的行为可采用上述规则进行描述，这是微粒群算法的基本概念之一。

Boyd 和 Richerson 在研究人类的决策过程时，提出了个体学习和文化传递的概念。根据他们的研究结果，人们在决策过程中使用两类重要的信息。一是自身的经验，二是其他人的经验。也就是说，人们根据自身的经验和他人的经验进行自己的决策。这是微粒群算法的另一基本概念。

微粒群算法最早是在 1995 年由美国社会心理学家 James Kennedy 和电气工程师 Russell Eberhart 共同提出的，其基本思想是受他们早期对许多鸟类的群体行为进行建模与仿真研究结果的启发。而他们的模型及仿真算法主要利用了生物学家 Frank Heppner 的模型。

Frank Heppner 的鸟类模型在反映群体行为方面与其它类模型有许多相同之处，所不同之处在于：鸟类被吸引飞向栖息地。在仿真中，一开始每一只鸟均无特定目标进行飞行，直到有一只鸟飞到栖息地，当设置期望栖息比期望留在鸟群中具有较大的适应值时，每一只鸟都将离开群体而飞向栖息地，随后就自然地形成了鸟群，。

由于鸟类使用简单的规则确定自己的飞行方向与飞行速度（实质上，每一只鸟都试图停在鸟群中而又不相互碰撞），当一只鸟飞离鸟群而飞向栖息地时，将导致它周围的其它鸟也飞向栖息地。这些鸟一旦发现栖息地，将降落在此，驱使更多的鸟落在栖息地，直到整个鸟群都落在栖息地。

由于 James Kennedy 和 Russell Eberhart 所具有的专业背景，就能很容易理解他们为什么会对 Hepper 的鸟类模型感兴趣。鸟类寻找栖息地与对一个特定问题寻找解很类似，已经找到栖息地的鸟引导它周围的鸟飞向栖息地的方式，增加了整个鸟群都找到栖息地的可能性，也符合信念的社会认知观点。

Eberhart 和 Kennedy 对 Heppner 的模型进行了修正，以使微粒能够飞向解空间并在最好解处降落。其关键在于如何保证微粒降落在最好解处而不降落在其它解处，这就是信念的社会性及智能性所在。

信念具有社会性的实质在于个体向它周围的成功者学习。个体与周围的其它同类比较，并模仿其优秀者的行为。将这种思想用算法实现将导致一种新的最优化算法。

要解决上述问题，关键在于在探索（寻找一个好解）和开发（利用一个好解）之间寻找一个好的平衡。太小的探索导致算法收敛于早期所遇到的好解处，而太小的开发会使算法不收敛。

另一方面，需要在个性与社会性之间寻求平衡，也就是说，既希望个体具有个性化，像鸟类模型中的鸟不互相碰撞，又希望其知道其它个体已经找到的好解并向它们学习，即社会性。

Eberhart 和 Kennedy 较好地解决了上述问题，提出了微粒群优化算法(Particle Swarm Optimization, PSO)[Kennedy 1995]。

2.2 基本微粒群算法

Kennedy 和 Eberhart 在 1995 年的 IEEE 国际神经网络学术会议上正式发表了题为“Particle

Swarm Optimization ”的文章，标志着微粒群算法的诞生。

2.2.1 算法原理

微粒群算法与其它进化类算法相类似，也采用“群体”与“进化”的概念，同样也是依据个体（微粒）的适应值大小进行操作。所不同的是，微粒群算法不像其它进化算法那样对于个体使用进化算子，而是将每个个体看作是在 n 维搜索空间中的一个没有重量和体积的微粒，并在搜索空间中以一定的速度飞行。该飞行速度由个体的飞行经验和群体的飞行经验进行动态调整。

设

$X_i = (x_{i1}, x_{i2}, ..., x_{in})$ 为微粒 i 的当前位置；

$V_i = (v_{i1}, v_{i2}, ..., v_{in})$ 为微粒 i 的当前飞行速度；

$P_i = (p_{i1}, p_{i2}, ..., p_{in})$ 为微粒 i 所经历的最好位置，也就是微粒 *i* 所经历过的具有最好适应值的位置，称为个体最好位置。对于最小化问题，目标函数值越小，对应的适应值越好。

为了讨论方便，设 *f(X)*为最小化的目标函数，则微粒 *i* 的当前最好位置由下式确定：

$$P_i(t+1) = \begin{cases} P_i(t) \text{若 } f(x_i(t+1)) \geq f(P_i(t)) \\ X_i(t+1) \text{若 } f(X_i(t+1)) < f(P_i(t)) \end{cases} \tag{2.1}$$

设群体中的微粒数为 *s*，群体中所有微粒所经历过的最好位置为 $P_g(t)$，称为全局最好位置。则

$$P_g(t) \in \{P_0(t), P_1(t), ..., P_s(t)\} \mid f(P_g(t)) = \min\{f(P_0(t)), f(P_1(t)), ..., f(P_s(t))\} \tag{2.2}$$

有了以上定义，基本微粒群算法的进化方程可描述为：

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(p_{ij}(t) - x_{ij}(t)) + c_2 r_{2j}(t)(p_{gj}(t) - x_{ij}(t)) \tag{2.3}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{2.4}$$

其中：下标“*j*”表示微粒的第*j*维，“*i*”表示微粒*i*，*t*表示第*t*代，c_1、c_2为加速常数，通常在 0~2 间取值，r_1~U(0,1)，r_2~U(0,1)为两个相互独立的随机函数。

从上述微粒进化方程可以看出，c_1调节微粒飞向自身最好位置方向的步长，c_2调节微粒向全局最好位置飞行的步长。为了减少在进化过程中，微粒离开搜索空间的可能性，v_{ij}通常限定于一定范围内，即 $v_{ij} \in [-v_{\max}, v_{\max}]$。如果问题的搜索空间限定在[$-x_{max}$，$x_{max}$]内，则可设定$v_{max}=k \cdot x_{max}$，$0.1 \leqslant k \leqslant 1.0$。

基本微粒群算法的初始化过程为：

1） 设定群体规模 *N*

2） 对任意*i*，*j*，在[$-x_{max}$，x_{max}]内服从均匀分布产生x_{ij}；

3） 对任意*i*，*j*，在[$-v_{max}$，v_{max}]内服从均匀分布产生v_{ij}；

4） 对任意*i*，设$y_i = x_i$。

2.2.2 算法流程

基本微粒群算法的流程如下：

step1，依照初始化过程，对微粒群的随机位置和速度进行初始设定；

step2，计算每个微粒的适应值；

step3，对于每个微粒，将其适应值与所经历过的最好位置P_i的适应值进行比较，若较好，则将其作为当前的最好位置；

step4，对每个微粒，将其适应值与全局所经历的最好位置 P_g 的适应值进行比较，若较好，则将其作为当前的全局最好位置；

step5，根据方程（2-3）(2-4)对微粒的速度和位置进行进化；

step6，如未达到结束条件通常为足够好的适应值或达到一个预设最大代数（Gmax），则返回step2。

基本微粒群算法的源程序具体见附录 1。

2.3 基本微粒群算法的社会行为分析

在(2.3)式所描述的速度进化方程中，其第一部分为微粒先前的速度；其第二部分为“认知”部分，因为它仅考虑了微粒自身的经验，表示微粒本身的思考。如果基本微粒群算法的速度进化方程仅包含认知部分，即：

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(p_{ij}(t) - x_{ij}(t)) \tag{2.5}$$

则其性能变差。主要原因是不同的微粒间缺乏信息交流，即没有社会信息共享，微粒间没有交互，使得一个规模为 N 的群体等价于运行了 N 个单个微粒，因而得到最优解的概率非常小。

（2.3）式的第三部分为“社会”部分，表示微粒间的社会信息共享。若速度进化方程中仅包含社会部分，即：

$$v_{ij}(t+1) = v_{ij}(t) + c_2 r_{2j}(t)(p_{gj}(t) - x_{ij}(t)) \tag{2.6}$$

则微粒没有认识能力，也就是“只有社会(social-only)”的模型。这样，微粒在相互作用下，有能力到达新的搜索空间，虽然它的收敛速度比基本微粒群算法更快，但对于复杂问题，则容易陷入局部最优点。

Kennedy 以 XOR 问题的神经网络训练为例进行了仿真实验，证明了上述结论。

总之，基本微粒群算法的速度进化方程由认识和社会两部分组成。虽然目前的一些研究表明，对一些问题，模型的社会部分显得比认识部分更重要，但两部分的相对重要性还没有从理论上给出结论。

2.3.1 与其它进化算法的比较

很显然，微粒群算法与其它进化算法有许多共同之处。首先，微粒群算法和其它所有进化类算法相同，均使用“群体”概念，用于表示一组解空间中的个体集合。如果将微粒所经历的最好位置y_i看作是群体的组成部分，则微粒群的每一步进化呈现出弱化形式的“选择”机制。在（μ + λ）进化策略算法中，子代与父代竞争，若子代具有更好的适应值，则用来替换父代，而PSO算法的进化方程（2.1）式具有与此相类似的机制，其唯一的差别在于，只有当微粒的当前位置与所经历的最好位置y_i相比具有更好的适应值时，其微粒所经历的最好位置（父代）才会唯一地被该微粒的当前位置（子代）所替换。总之，PSO算法隐喻着一定形式的“选择”机制。

其次，（2.3）式所描述的速度进化方程与实数编码遗传算法的算术交叉算子相类似，通常，算术交叉算子由两个父代个体的线性组合产生两个子代个体，而在PSO 算法的速度进化方程中，假如先不考虑$v_{ij}(t)$项，就可以将该方程理解为由两个父代个体产生一个子代个体的算术交叉运算。从另一个角度，在不考虑$v_{ij}(t)$项的情况下，速度进化方程也可以看作是一个变异算子，其变异的强度（大小）取决于两个父代微粒间的距离，即代表个体最好位置和全局最好位置的两个微粒的距离。至于$v_{ij}(t)$项，也可以理解为一种变异的形式，其变异的大小与微粒在前代进化中的位

置相关。

通常在进化类算法的分析中，人们习惯于将每一步进化迭代理解为用新个体（即子代）代替旧个体（即父代）的过程。这里，如果我们将PSO算法的进化迭代理解为一个自适应过程，则微粒的位置x_i就不是被新的微粒所代替，而是根据速度向量v_i进行自适应变化。这样，PSO算法与其它进化类算法的最大不同点在于：PSO算法在进化过程中同时保留和利用位置与速度（即位置的变化程度）信息，而其它进化类算法仅保留和利用位置的信息。

另外，如果将（2.4）看作一个变异算子，则 PSO 算法与进化规划很相似。所不同之处在于，在每一代，PSO 算法中的每个微粒只朝一些根据群体的经验认为是好的方向飞行，而在进化规划中可通过一个随机函数变异到任何方向。也就是说，PSO 算法执行一种有“意识（Conscious）”的变异。从理论上讲，进化规划具有更多的机会在优化点附近开发，而 PSO 则有更多的机会更快地飞到更好解的区域，如果“意识”能提供有用的信息。

从以上分析可以看出，基本 PSO 算法也呈现出一些其它进化类算法所不具有的特性，特别是，PSO 算法同时将微粒的位置与速度模型化，给出一组显式的进化方程，是其不同于其它进化类算法的最显著之处，也是该算法所呈现出许多优良特性的关键点。

2.3.2 两种基本进化模型

在基本 PSO 算法中,根据直接相互作用的微粒群定义可构造 PSO 算法的两种不同版本,也就是说，可以通过定义全局最好微粒（位置）或局部最好微粒（位置）构造具有不同社会行为的 PSO 算法。

1. Gbest 模型（全局最好模型）

Gbest 模型以牺牲算法的鲁棒性为代价提高算法的收敛速度，基本 PSO 算法就是该模型的具体实现。在该模型中，整个算法以该微粒为吸引子，将所有微粒拉向它，使所有微粒将最终收敛于该位置。这样，如果在进化过程中，该最好解得不到有效地更新，则微粒群将出现类似于遗传算法中的早熟现象。为了讨论方便，将 PSO 的进化方程重新表述如下：

$$P_g(t) \in \{P_0(t), P_1(t), ..., P_s(t)\} \mid f(P_g(t)) = \min\{f(P_0(t)), f(P_1(t)), ..., f(P_s(t))\} \tag{2.7}$$

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)[p_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[p_{gj}(t) - x_{ij}(t)] \tag{2.8}$$

其中，$P_g(t)$称为全局最好位置，对应于称之为全局最好微粒所处的位置。

2.Lbest 模型（局部最好模型）

为了防止 Gbest 模型可能出现的早熟现象，Lbest 模型采用多吸引子代替 Gbest 模型中的单一吸引子。首先将整个微粒群分解为若干个子群，在每一子群中保留其局部最好微粒$P_i(t)$，称之为局部最好位置或邻域最好位置。假设第 i 个子群处于长度为 l 的领域内，则 Lbest 模型的进化方程可描述为：

$$N_i = \{P_{i-l}(t), P_{i-l+1}(t), ..., P_{i-1}(t), P_i(t), P_{i+1}(t), ..., P_{i+l-1}(t), P_{i+l}(t)\} \tag{2.9}$$

$$P_i(t+1) \in \{N_i \mid f(P_i(t+1)) = \min f(a)\}, \forall a \in N_i \tag{2.10}$$

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)[p_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[P_{gj}(t) - x_{ij}(t)] \tag{2.11}$$

注意：子群N_i中的微粒与其在搜索空间域内所处的位置无关，仅依赖微粒的索引数或者微粒的编码。这样一方面避免了微粒间的聚类分析，节省了计算时间，另一方面能够加快更好解信息在整个群体间的扩散。

很容易看出，Gbest 模型实际上是 Lbest 模型在 $l=s$ 时的特殊情况。实验证明：$l=1$ 时的 Lbest 模型，其收敛速度低于 Gbest 模型，而 $1<l<s$ 时，但收敛速度仍低于 Gbest 模型，但收敛的全局最优性明显改善。

2.4 带惯性权重的 PSO 算法

为了改善基本 PSO 算法的收敛性能，Y．Shi 与 R．C．Eberhart 在 1998 年的 IEEE 国际进化计算学术会议上发表了题为“A Modified Particle Swarm Optimizer”的论文，首次在速度进化方程中引入惯性权重，即：

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[p_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[p_{gj}(t) - x_{ij}(t)] \tag{2.12}$$

式中 w 称为惯性权重，因此，基本 PSO 算法是惯性权重 $w=1$ 的特殊情况。惯性权重 w 使微粒保持运动惯性，使其有扩展搜索空间的趋势，有能力探索新的区域。

引入惯性权重w可清除基本PSO算法对v_{max}的需要，因为w本身具有维护全局和局部搜索能力的平衡的作用。这样，当v_{max}增加时，可通过减少w来达到平衡搜索。而w的减少可使得所需的迭代次数变小。从这个意义上看，可以将$v_{max,d}$固定为每维变量的变化范围，只对w进行调节。

对全局搜索，通常的好方法是在前期有较高的探索能力以得到合适的种子，而在后期有较高的开发能力以加快收敛速度。为此，可将 w 设定为随着进化而线性减少，例如由 0.9 到 0.4 等等。Y．Shi 和 R．C．Eberhart 的仿真实验结果也表明 w 线性减少取得了较好的实验结果。有关这一方面的详细讨论将在第六章进行。

目前，有关 PSO 算法的研究大多以带惯性权重的 PSO 算法为基础进行扩展和修正。为此，在大多文献中将带惯性权重的 PSO 算法称之为 PSO 算法的标准版本，或简称标准 PSO 算法；而将基本 PSO 算法称为 PSO 的初始版本。