第3章

生物个体行为模式与自适应优化方法

引　　言

本章在对生物个体自适应行为建模与仿真研究的基础上，设计两种新型生物启发计算模式：细菌自适应觅食优化和植物根系自适应生长优化。通过在标准测试函数上的仿真分析，细菌自适应觅食优化和植物根系自适应生长优化具有良好的优化精度和收敛速度，为求解实际工程应用中的连续优化和动态优化问题提供了新的思路。

3.1　自然进化中的个体行为模式

3.1.1　生物个体的觅食行为分类

觅食行为生态学是动物生态学研究的主要内容之一。在生物界，食物是生命物质和能量的来源，是生物赖以生存的基本条件，也是其生存、繁衍和进化的基础，任何种间及种群内部的进化关系都与食物有着密不可分的联系[1]。因此，动物的觅食行为机制、觅食活动时间、分配策略等对动物生存繁衍都具有重要的意义。

观察生物个体觅食行为特征，了解觅食行为对觅食效率的影响，有助于人们探索更为高效的搜索策略，发展出新的群体智能算法和实现群机器人的协调控制等，为复杂系统的优化提供新的理论指导与工程实践应用。本书根据觅食行为特点的不同，对生物个体进行分类。

1. 哺乳动物

食物的营养成分及其含量存在种间、种内个体间，时间及空间的异质性，哺乳动物为提高其本身的适合度，必须对食物质量，以及觅食地点、觅食时间、觅食方式作出行为反应与权衡，以便选择营养丰富的食物项目，并找到食物资源丰富的取食区域。每个哺乳动物个体不仅要与同伴分担探究环境的风险，也要共享食物资源信息，从而提高觅食效率，降低中毒与被捕食的风险。

下面以松鼠为例说明哺乳动物的觅食过程。松鼠每次在地面捡到食物后总是要回到树上去吃，就这样一次次往返于树上和地面之间。松鼠的上述行为模式是在食物和安全之间求得一种平衡。在就地吃食（牺牲了安全）和待在树上（忍受饥饿）之间权衡。理论上讲，当

往返移动对能量摄取率的影响比较小时，松鼠应把食物带回树上去吃，以便求得绝对安全。在人为投食实验中，如果食物很大而且离树又近，松鼠就会把食物带到树上去吃；对小食物则常常就地吃掉。

从哺乳动物个体对食物搜索的角度来看，这里以广泛被生物界所采用的局限区域搜索(Area-Restricted Search，ARS)策略为例进行说明(如图 3-1 所示)[2-4]，哺乳动物在捕食的过程中，会首先对某一区域以较快的移动速率、较长的移动步伐进行大范围的不精确探索；当这些捕食者定位食物源在某一较小的固定区域后，它们会对该区域以较慢的移动速率、较小的移动步伐进行精确的开发；一旦对该区域的食物源搜索失败或已将该区域食物源耗尽，捕食者会重新以较快的移动速率、较长的移动步伐离开该区域，并开始对新的区域进行下一阶段的探索和开发。可见，ARS 是哺乳动物对不断变化的捕食环境的一种自适应觅食策略，通过动态的平衡捕食过程中的探索和开发行为，哺乳动物找到食物源的概率大大提高，捕食所消耗掉的时间却被有效降低。

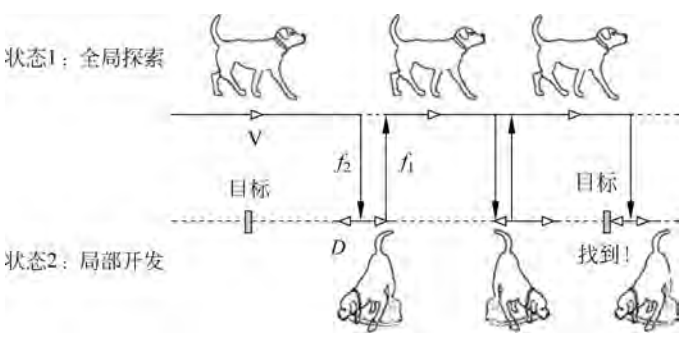


图 3-1　哺乳动物捕食过程

2. 禽类

在禽类觅食策略中，食物源(食物地点)的选择至关重要。禽类在取食前必须首先决定到哪儿去取食，取食什么类型的食物，什么时候转移取食地点，如图 3-2 所示。

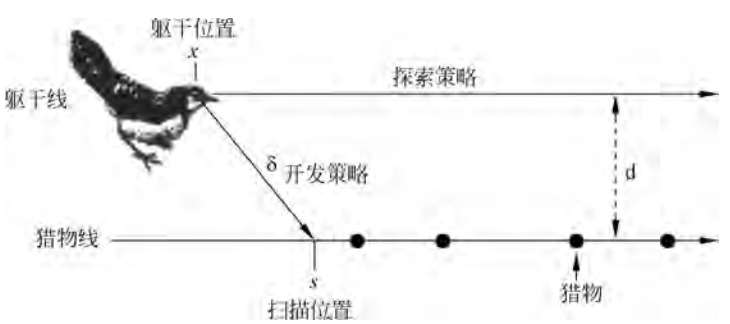


图 3-2　禽类觅食策略

例如，在一片空旷的场地中分布着成堆的稻谷，一群鸟正在这片场地中进行觅食。对某一只后来的鸟，它的觅食过程涉及两个选择：第一，是否进入当前的食物堆；第二，是否和自己的同伴竞争这堆食物还是自己去寻找其他的稻谷堆。它最终的觅食策略选择由最优觅食理论决定，即捕食者总是倾向于选择有利性更大的食物。特别地，在鸟的捕食食谱中，不仅应该包括有利性大的食物，也应该在一定程度上包含有利性小的食物，这中间存在着一种

最适权衡。

一种叫作椋鸟的禽类主要靠捕食大蚊幼虫和其他无脊椎动物来喂养雏鸟。在生殖季节最繁忙时，每只亲鸟每天要在鸟巢和取食地之间往返飞行达 400 多次，一次次地把食物带回巢中喂给雏鸟。亲鸟每次多带幼虫回巢就会减少往返次数，但椋鸟是用喙以特定方式搜寻食物的。起初，它的搜寻效率很高，但当它的口里已含有幼虫时，搜寻效率就会明显下降。因此，每次收集的猎物太多，其总效益不一定最好。禽类生物学者还研究了绒斑啄木鸟的觅食行为。在每根树干上都有 24 个洞，啄木鸟喜欢吃的种子就藏在洞内。啄木鸟事先并不知道哪些树干上的洞是空的，所以它们在开始觅食时就必须收集每一根树干的信息，并利用这些信息决定是否应该在一根树干上继续搜寻下去。

3. 鱼类

不同种类的鱼类对食物的选择受先天遗传基因制约，后天基本不变，但生存条件发生变化后，它们的食性也会发生变化。鱼类在长期觅食的过程中受条件反射的影响，逐渐形成觅食习惯，包括每天觅食的时间次数、路线、当水温、水流、水深、风向和水的溶氧量发生变化时，鱼的觅食规律也随之改变。这就是环境对鱼类搜索捕食策略的影响，如图 3-3 所示。

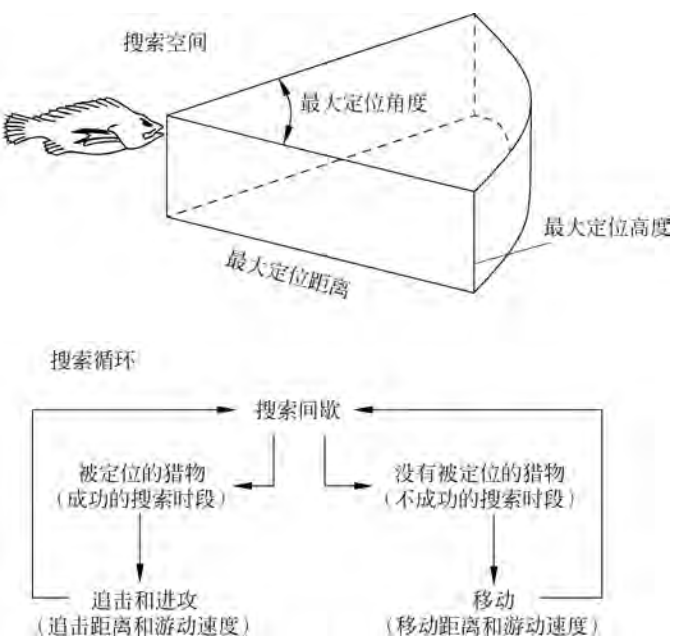


图 3-3　鱼类搜索捕食策略

不同大小的海刺鱼分别选择不同大小的猎物（新糠虾）作为最喜食的食物[5]。海滨蟹在选择贻贝时，贻贝太大，破碎贝壳会花费很长时间，因此单位处理时间的能量收入下降；贻贝太小，虽然破碎贝壳所花的时间较短，但小贻贝的含肉量太少，经济上不合算。海滨蟹对贻贝大小的选择往往是落在一定范围之内（以 3cm 大小的贻贝为中心）。如果海滨蟹需要花很长时间才能找到一个最合适大小的食物，那么吃一些更大或更小的食物反而能提高它的能量总摄取率。

4. 昆虫

蚂蚁种类繁多，是生活在地球上数量最多的社会性昆虫。蚂蚁觅食是多个个体参与组

织有序的集体过程。蚂蚁个体的行为模式相对简单，行为规则容易建立。因此，蚂蚁觅食行为模型不仅是昆虫学家、生态学家的研究对象，也是人工智能专家的研究课题。

蚂蚁一般采用个体和群体相结合的觅食方式。研究发现大多数蚂蚁种类都是少量个体分散寻觅食物，当发现超过自身所能搬运的、较为丰富的食物时，会立刻召唤同伴共同搬运。如我们在马路边常可以看到一群蚂蚁捕食大型昆虫的现象。这种方式不仅使蚂蚁能够共同搬运较大的食物，而且使蚂蚁总是趋向食物最为丰富的区域[6-7]。

蜜蜂具有非常奇妙和最有诱惑力的觅食行为，对其觅食行为的研究可以追溯到达尔文本人，他曾对这种社会性昆虫进行过大量的研究。蜜蜂在蜂巢外进行觅食时必须具有飞行定向的能力，关键是能从环境中获取必要的信息并加以利用。蜜蜂为了觅食往往要飞行几千米的距离，通常年幼的工蜂会留在巢内工作（称内勤蜂），主要责任是喂养照料幼虫和清洁蜂巢等，待发育到一定年龄后才会离开蜂巢进行觅食或寻找新蜜源地（称外勤蜂），此时它们的活动靠的是视觉和嗅觉。当它们第一次离开蜂巢时并不是马上就开始觅食活动，而是多次返回蜂巢并在蜂巢上方上下飞翔几分钟，这种行为称为“定向飞行”，它显然可以使觅食者辩明并记住蜂巢在环境中的相对位置。当蜜蜂发育到大约 1 周龄时开始进行这种“定向飞行”，而直到发育到 3 周龄时才会开始真正的觅食活动[8]。

从目前文献来看，蚂蚁和蜜蜂这类昆虫的觅食行为适合开发基于生物种群行为的计算模式和优化方法，但是其个体的觅食行为依然具有其他生物行为无法比拟的特色，可以对其进行继续探索和研究，以设计出更加新颖、效果更好的优化算法[9-14]。

5. 海星

海星在生物学上并不属于鱼类，看上去不像是动物，而且从其外观和缓慢的动作来看，很难想象，海星竟是一种贪婪的食肉动物。它对海洋生态系统和生物进化还起着非同凡响的重要作用，这也就是它为何在世界上广泛分布的原因。

海星主要捕食对象是一些行动较迟缓的海洋动物，如贝类、海胆、螃蟹和海葵等。它捕食时常采取缓慢迂回的策略，慢慢接近猎物，用腕上的管足捉住猎物，并用整个身体包住它，将胃袋从口中吐出、利用消化酶让猎获物在其体外溶解并被其吸收，如图 3-4 所示。

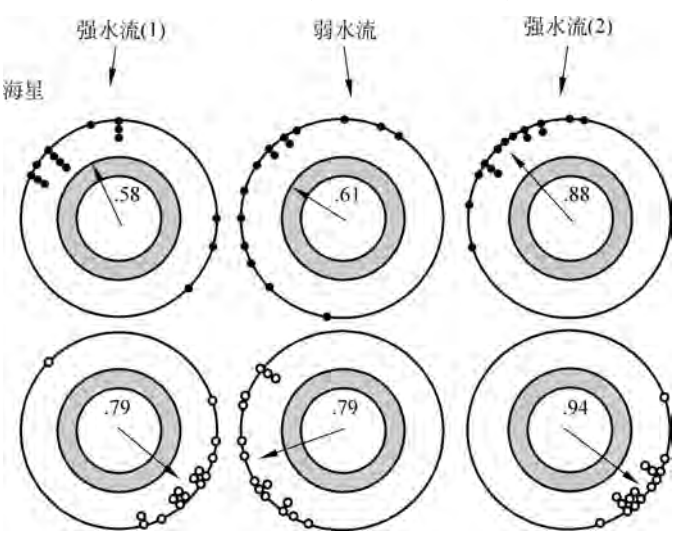


图 3-4　海星捕食策略

在自然界的食物链中，捕食者与被捕食者之间常常展开生与死的较量。为了逃脱海星的捕食，被捕食动物几乎都能作出逃避反应。有一种大海参，每当海星触碰到它时，它便会猛烈地在水中翻滚，趁还未被海星牢牢抓住便逃之夭夭。扇贝躲避海星的技巧也较独特，当海星靠近它时扇贝便会一张一合地迅速游走。有一种小海葵，每当海星接近它时，它便从攀附的礁石上脱离，随波逐流，漂流到安全之地。这些动物的逃避能力是从长期进化中产生的，这使它们避免了被大自然淘汰的命运。

3.1.2 适应性主体

当我们观察显微镜下的世界时，会发现一些由许多个体组成的群落，其每个部分都非常复杂。人体免疫系统就是这样的一个群落，它由大量快速活动着的被称为抗体的单位组成，这些抗体不断地抵抗或摧毁不断变化的被称作抗原的入侵者。这些入侵者基本上是各种不同类型的生物化学物质，如细菌和病毒。它们形态各异，像雪花那样变化多端。由于这种多样性，以及新的入侵者总是不断出现，免疫系统不可能简单地列举出所有可能的入侵者。随着新的入侵者出现，免疫系统必须使抗体改变自身或者去适应新的入侵者，而从来不保持于某种固定的构型。尽管有着变化多端的本性，免疫系统仍然保持着很强的协调性。

由各种生物个体组成的生态系统与免疫系统一样，也有着很多类似的特性。它们呈现出同样惊人的多样性。我们还没有能够分析出 $1m^3$ 的温带土壤中有机体的所有种类，更不用说热带森林的物种分类了。生态系统不断地变化着，呈现出绚丽多姿的相互作用及其种种后果，如共生、寄生、生物学“军备竞赛”和拟态等。在这个复杂的生物圈里，物质、能量和信息等结合在一起循环往复，各种生物个体，如哺乳动物、禽类、鱼类、昆虫和海星等，交融混杂。事实再次应验了这一点，整体大于部分之和。即使我们对绝大多数物种的活动进行了分类，也远远不能认识生态系统中种种变化的效应。例如，热带森林的巨大财富与其土壤的贫瘠程度形成了鲜明的对比。只有经过一系列复杂的相互作用，通过整个系统使稀缺的养分反复循环，热带森林才得以维持它的多样性。

虽然这些由生物个体组成的复杂系统在细节上有所不同，但是，在发展变化中的协调性问题，对这些系统而言是一个主要的共同点，以至于它们被冠以一个共同的名称——复杂适应系统(Complex Adaptive System，CAS)。

CAS 无例外地皆由大量具有主动性的元素(active element)组成，这一点我们从例子中已经看到。这些元素无论在形式上还是在性能上都各不相同(见图 3-5)。为了说明具有主动性的元素，同时不求助于专门的内容，借用了经济学中的主体(agent)一词。这个世界的状态，是由其中决策能力的个体间的相互作用来自然描述的，这一点和游戏很类似。当给这样的系统建模时，这个个体就称为主体，这个模型就称为基于主体的模型。这个术语是描述性的，应当避免先入之见。如果要弄清楚大量主体的相互作用，就必须首先能够描述单个主体的性能。将主体的行为看成是由一组规则决定的，这一点很有用。刺激-反应规则(stimulus response rules)非常典型而且通俗易懂。IF(若)刺激 s 发生，THEN(则)做出反应 r。IF 生物个体饥饿，THEN 有觅食行为。IF 松鼠确保安全，THEN 待在树上(忍耐饥饿)。IF 椋鸟喂养雏鸟，THEN 往返多次飞行取食。IF 大海参被海星触碰到，THEN 猛烈地翻滚逃生。这样的例子还可以举出很多很多。为了对一个给定的主体定义其刺激-反应规则，我们必须首先描述主体能够收到的刺激和它能够做出的反应(见图 3-6)。

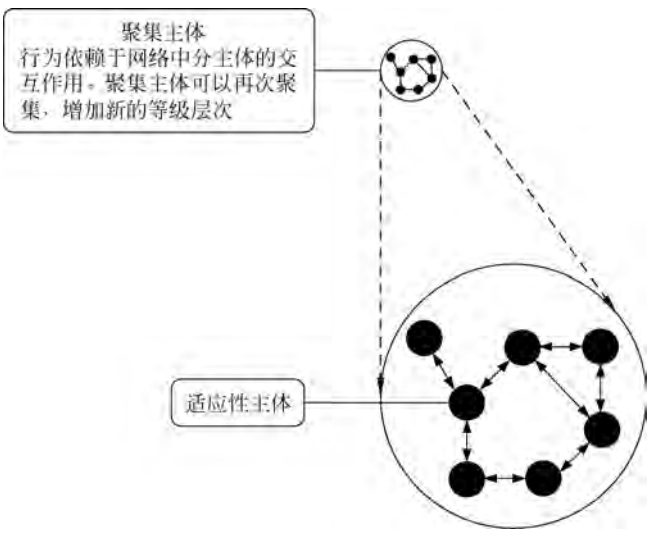


图 3-5　复杂自适应系统

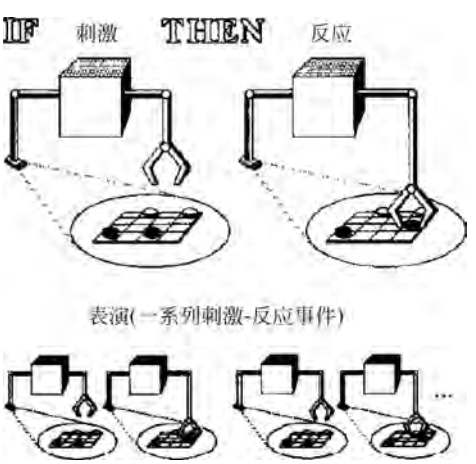


图 3-6　刺激-反应规则

虽然刺激-反应规则的应用范围是有限的，但是可以通过一些简单的方式拓展这个范围。事实上，通过很少的一些变化，应用范围就可以有效地扩大，使得使用一组规则，就能够生成可用计算方法描述的任何行为。在定义这些规则时，并不是说能够在真实主体中明确地找出其规则。规则只不过是用来描述主体策略的一种方便途径，可以把它看成是一个描述工具。

任何 CAS 的建模工作，主要都归结为选择和描述有关的刺激和反应，因为各个分主体(component agent)的行为和策略都由此而确定。对中枢神经系统中的主体(神经元)而言，刺激可以是到达每个神经元表面的脉冲，反应则是发出的脉冲。对免疫系统中的主体(抗体)来说，刺激可以是入侵抗原表面的分子构型，反应则是对抗原表面的不同的黏合(adhesion)。对于哺乳动物的主体(个体本身)而言，刺激可以是猎物、食物、天敌的出现，反应则是在食物和安全之间求得的一种平衡。对于蚂蚁这个有高度组织系统中的主体(每只蚂蚁)而言，刺激可以是发现较为丰富的食物源，反应则是召唤同伴共同搬运。对其他 CAS 都可以进行类似的选择。注意，上述每种情况使用“可以”一词是恰当的，因为也可以选择其他的事件和因素。不同的选择强调 CAS 不同的侧面，也就会得出不同的模型。这里并不存在孰对孰错的问题(虽然模型有时可能会构造得很蹩脚)，因为这要看当前究竟要解决什么问题。

对一个给定主体，一旦指定了可能发生的刺激的范围，以及估计到可能做出的反应集合，就已经确定了主体可以具有的规则种类。然后，按行为的顺序考查这些规则，就可以得到主体行为的描述。正是在这一点上，学习或适应的概念开始引入。在安排基本元素表的

时候，要把“适应”放在首位，因为适应是 CAS 必不可少的条件。但是，适应是一个非常广泛的话题，它涉及许多方面。

从生物学角度说，适应是生物体调整自己以适合环境的过程。粗略地说，生物体结构的变化是经验引导的结果。因此，随着时间的推移，生物体将会更好地利用环境达到自己的目的，如图 3-7 所示，生物个体的觅食行为就是最好的例证。在这里，我们把这一术语范围扩大，把学习与相关过程也包括进来。尽管不同的 CAS 过程具有不同的时间尺度，但适应的概念可以应用于所有的 CAS 主体。并且，事实上，时间尺度确实因情况而异。在神经系统中，个体神经元发生的适应性变化的间隔从数秒到数小时；免疫系统的适应性变化则需要数小时到数天；生态系统以及生物个体的适应性变化，可能需要数年到数千年，甚至更长时间。

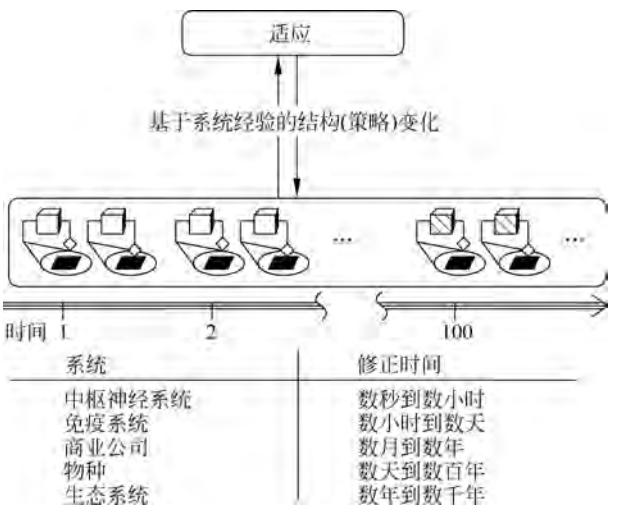


图 3-7　适应和学习

总之，我们将 CAS 看成是由用规则描述的、相互作用的主体组成的系统。这些主体随着经验的积累，靠不断变换其规则来适应。在 CAS 中，任何特定的适应性主体所处环境的主要部分，都由其他适应性主体组成，所以，任何主体在适应性上所做的努力就是要去适应别的适应性主体。这个特征是 CAS 生成的复杂动态模式的主要根源。

3.1.3　效率与最优觅食理论

所谓最优觅食理论，是指动物为获得最大的觅食效率所采取的各种方法和措施。如选择最有利的食物，或最优食谱，或最有利的生态小区等。自然法则倾向于保留觅食能力强的生物，并淘汰觅食能力弱的生物，这里所指的觅食能力包括确定食物位置的能力，捕获食物的能力和消化食物的能力。最优觅食理论基于下面的这个假设：生物在觅食的过程中，都在尽力寻找一种方式使其在较短的时间内获得较多的食物，这便是生物的最优觅食理论[15-16]。用数学公式表述如下：

$$\max\left(\frac{E}{T}\right)$$

其中，E 代表通过捕食食物获得的能量；T 代表觅食行为耗费的时间。从上式可以看出，在

自然进化的过程中，觅食能力较强的生物会在一定时间内获得足够多的食物供它们生存并繁殖。所以这些生物生存下来的可能性就会变大，它们优秀的觅食基因进而被保留下来，经过数代的进化之后，觅食能力较弱的生物或者被淘汰，或者经过进化拥有了较强的觅食能力。

大多数生物的捕食过程可以分成以下几部分。首先，捕食者搜寻和确定猎物的位置；其次，他们接近并袭击猎物；最后，将猎物捕获并消化掉。在大多数情况下，猎物要比捕食者小，所以捕食者一定要频繁并大量地进行捕食搜索活动，以满足自身对食物供给的需要。在这种情况下捕食所消耗掉的时间极大影响了捕食者的捕食行为。于是，搜索行为就成了觅食过程起决定作用的部分。

在进化过程中，自然界中的各种生物开发出了很多能在最短时间内确定食物源位置的自适应搜索策略。这里的“自适应”是指捕食者在搜索过程中，会根据觅食环境的变化来改变其移动的速率、步伐的长度和等待的时间来最大化成功捕食的概率。

以广泛被生物界所采用的局限区域搜索（Area-Restricted Search，ARS）策略为例进行说明，生物在捕食的过程中，会首先对某一区域以较快的移动速率、较长的移动步伐进行大范围的不精确探索；当这些捕食者定位食物源在某一较小的固定区域后，它们会对该区域以较慢的移动速率、较小的移动步伐进行精确的开发；一旦对该区域的食物源搜索失败或已将该区域食物源耗尽，捕食者会重新以较快的移动速率、较长的移动步伐离开该区域，并开始对新的区域进行下一阶段的探索和开发。可见，ARS 是生物对不断变化的捕食环境的一种自适应觅食策略，通过动态的平衡捕食过程中的探索和开发行为，生物找到食物源的概率大大提高，捕食所消耗掉的时间却被有效降低了。

3.2 基于生物个体行为的计算模式设计

3.2.1 基于生物个体行为的统一优化框架

基于生物个体行为的计算模式主要是指：在优化过程中，一个或多个个体没有信息的共享和交互，个体只能从环境中获得信息，向着最优解方向移动。每个生物个体可用如下二元组表示：

$$\mathrm{IBCM}=(\mathrm{Individual},\mathrm{Environment})$$

式中，Individual 为个体集合，且

$$\mathrm{Individual}=\{\mathrm{Individual}_1,\mathrm{Individual}_2,\cdots,\mathrm{Individual}_i,\cdots,\mathrm{Individual}_{\mathrm{INUM}}\}$$

式中，$\mathrm{Individual}_i=<\mathrm{IID},\mathrm{INOBJ},\mathrm{STATUS},\mathrm{BHV},\mathrm{MESS},\mathrm{ACT},\mathrm{TANS}>$，具体含义详见 3.3.1 节。

Environment 为环境，即优化的目标函数，且

$$\mathrm{Environment}=\{\mathrm{Range},\mathrm{Food}\}$$

具体含义详见 3.4.3 节。

基于此种模式的优化方法主要依赖个体的行为方式，个体自身的探索和开发能力决定了优化效果。另外，生物个体的行为多种多样，包括觅食、分根、移动、搬运、变异、生长、发光、释放激素、趋化和筑巢等，如图 3-8 所示。从计算模式角度来看，这些行为主要分为两类：开发和探索。

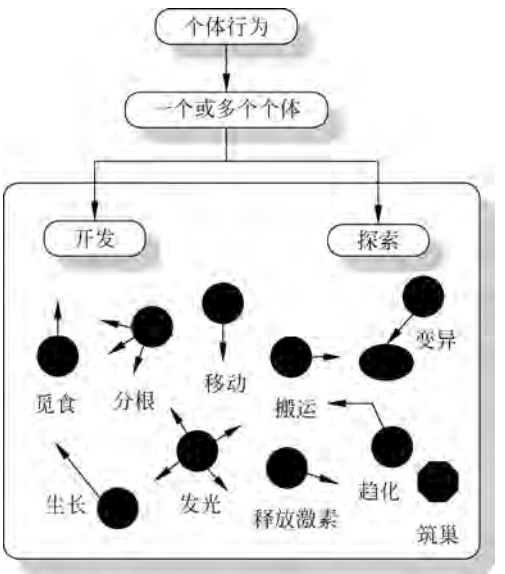


图 3-8　生物个体行为的统一框架

3.2.2　基于生物个体行为的基本操作

生物个体行为的基本操作将随着个体习性的不同而有差异，本书考虑几种典型的生物个体搜索机制，即区域集中搜索、跳跃搜索、分根操作和自适应生长操作建立优化模型。自然界大部分的生物搜索策略可以由其中一种或多种搜索策略融合产生。

1. 区域集中搜索

由上文可知，区域集中策略 ARS 是一种典型的基于生物个体行为搜索策略。在这种策略中，捕食者在一个能遇见单个猎物个体的特殊区域里集中搜索，尝试着寻找猎物的自然趋向。这种搜索方式通常是以带有大步长和小转角特点的广泛搜索模式开始，然后根据特有的线索进行精细化的搜索。这个策略在行为生物学领域被称为“慢赢-速错”机制。当成功地发现猎物，捕食者就会缓慢地进行密集搜索，但是如果捕食失败，就会快速进行大范围搜索[19]。幼年的比目鱼是一个典型的例子[20]，虽然目前还不知道它的感知器官，但是它就是利用局部的线索从大范围搜索到集中搜索。这项转换要缩短 15%的间隔时间，增加 14%的移动时间，减速 22%，增加平均绝对转角 33%。这种改变的作用就是捕食者能够快速、有效地搜索一小片区域，因为它知道一旦发现了猎物，就很有可能在附近发现其他的猎物。

Krakauer 和 Rodrl'guez-Girone's 两位学者在 1995 年提出了一个区域集中搜索模型用来检验捕食规则[19]。

(1) 跟踪误差。当捕食者广泛地搜索资源好的区域或者集中地搜索资源差的区域时会出现跟踪误差。在区域集中搜索模型中，利用斑片状分布减少跟踪误差，仅仅在遇到猎物时初始化密集搜索。误差的数量和类型取决于资源斑片的分布情况。

(2) 学习。捕食者为了进行最大限度的开发需要了解环境和资源的分布。这项行为必须在开发所用的时间和从开发中获得的价值之间达到平衡。

(3) 信息的评估。捕食者必须确定它自己收集的信息是否有价值或者是否可以使用其

他的策略代替。随着斑块增加，先前假设是错误的可能性、评估猎物分布和选择恰当的开发策略所花费的时间都会增加。

(4) 机能反映。猎物的捕获量对应着区域中的食物源密度。资源约束策略的选择能够改变从在猎物密集区域限制因素是处理猎物所花费的时间到捕食者选择花费更多的时间搜索的响应曲线。

但是，对于上述规则，Krakauer 和 Rodrl'guez-Girone's 也对标准慢赢-速错的区域集中搜索策略是否一直最好做了评估。他们证明，在资源分布在好坏之间规则交替的区域，这种方法并不是很有效，因为切换到密集搜索模式只对连续遇见机会较高的区域是有价值的。

2. 跳跃搜索

动物并不是一直需要在局部范围内进行密集的猎物搜索。当目标分布稀疏很难被发现时，生物们经常使用间断地寻找和迁徙捕食的方法，也就是跳跃搜索策略[21]。这种策略包括两个不同的阶段：一个是搜索阶段，搜索者停顿下来开发他的周边；另一个是移动阶段，搜索者快速地移动，正常来讲是不可能侦测到隐藏的猎物的。

在跳跃搜索的搜索阶段，捕食者为了找到一个隐藏很好的目标可能不得不多次重复搜索一个小的区域。Benichou 等在 2005 年利用扩散过程(随机游动)方法对动物的搜索阶段建模，扩散过程的一个显著的地方是，它是感知搜索而不是动物的物理运动[21]。建模的结果再加上一些动物物种的实例表明，这种扩散搜索对于搜索阶段是一个很好的选择。

虽然 Benichou 等在模型中采用了从自然界中观察到的有效的重新定位机制：直线弹道式的运动到下一个搜索区域，但是在区域集中搜索变化到广泛搜索，跳跃搜索策略的移动阶段和幼年比目鱼使用的偏向策略是相似的。模型的结果表明搜索阶段和移动阶段的平均持续时间的最佳关系应该是双峰分布的(移动阶段是搜索阶段的 3/5 或 2/3 次幂)。

综上所述，区域集中搜索策略适合于捕食者敏锐、猎物可见的情况，这样猎物能在广域搜索中被发现；跳跃搜索适用于感知敏度受限、猎物可见性较低、在广域搜索不容易找到猎物的情况。

3. 分根操作

分根操作是生物个体行为计算模式的一种典型操作，也是 3.4.3 节介绍的植物自适应生长算法的一个重要操作。植物生长从一颗种子开始，每一次生长过程(每一次循环)，选出优秀个体作为本代的生长点，选择条件主要根据个体适应度，但是选择的生长点不能距离太近。根据这个原则选出每一代的生长点进行生长，选择过程如图 3-9 所示。生长点选好后，这些生长点进行分根操作，也就是说，只有这些个体才有资格产生新的个体(生长点)，分根操作的执行伪代码如表 3-1 所示。

表 3-1　分根操作伪代码

```
FOR each point pg_i
  FOR each new point X
  k = rand();
    X_S = pg_ik + (2 × rand() − 1);
    S = S + 1;
  END
END
```

4. 自适应生长操作

植物自适应生长操作主要是指在每一次计算中，植物根系中的每条根须生长长度的自适应变化。在整个优化过程中，每条根须的生长长度在每一代都有可能不同，或者每次适应度计算后变化，或者每隔一定循环次数进行变化。这里提出两种自适应生长方式：

生长方式 1：设定根须生长长度的初始值，每隔一定循环次数，根须生长长度按照一定比例下降，当每个生长点在循环一定次数后，适应度没有变化，根须生长长度按照一定比例增长，直到达到初始值为止。伪代码如表 3-2 所示。利用 Rastrigrin 函数作为土壤环境，仿真出这种自适应根须生长方式，如图 3-10 所示，图中每个不同颜色的点代表植物根系中不同的生长点，在一定循环次数后，生长点呈两极分化，一部分生长点朝着最优点方向继续生长，另一部分生长在土壤中大范围探索最优点。

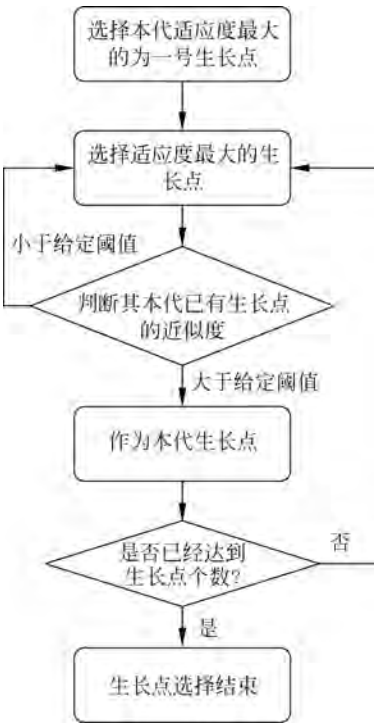


图 3-9　生长点选择流程

生长方式 2：每次适应度计算后，根须的生长长度根据适应度变化，执行公式和伪代码如表 3-3 所示。同样，利用 Rastrigrin 函数作为土壤环境，仿真出这种自适应根须生长方式，如图 3-11 所示，所有生长点走向一致，都会朝着最优点方向生长，并在最优附近来回移动。这种自适应生长方式有利于生长点更加靠近最优点，但也容易陷入局部最优值。

表 3-2　根须自适应生长方式 1

```
FOR each iteration
  FOR each point X_i
    IF no change count > default count
      δ_i = δ_i × 1.2;
      IF δ_i > Initial value
        δ_i = Initial value
      END
    END
  END
  IF 200 | iteration
    δ_i = δ_i × 0.8
  END
END
```

表 3-3　根须自适应生长方式 2

```
FOR each iteration
FOR each point X_i
  δ_i = |E_i| / |E_i + τ|
END
END
```

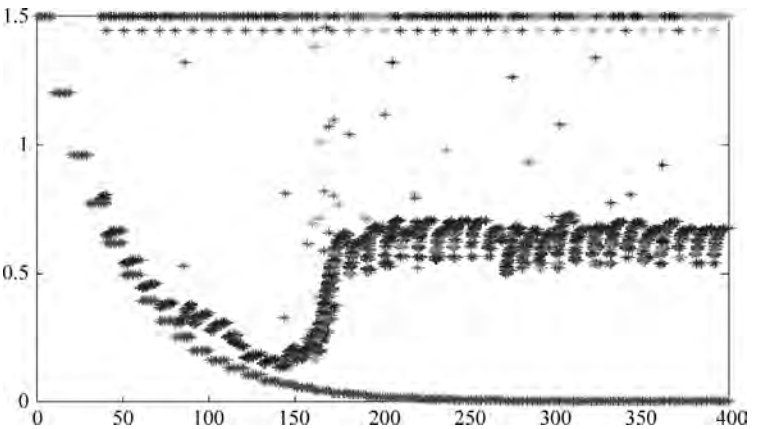


图 3-10　Rastrigrin 函数仿真自适应根须生长方式 1

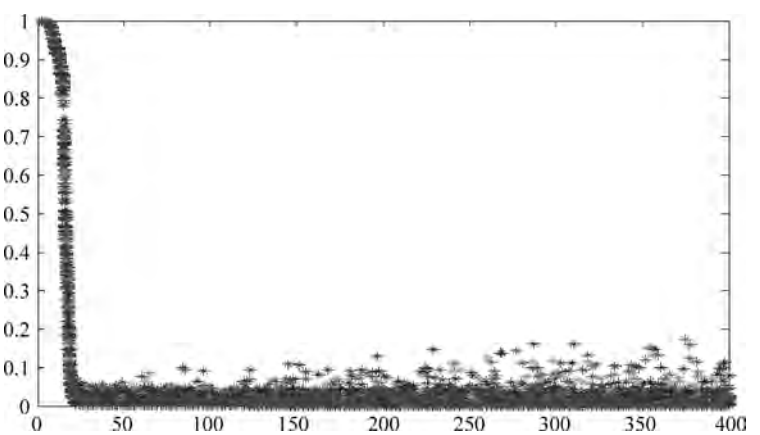


图 3-11　Rastrigrin 函数仿真自适应根须生长方式 2

3.3　生物个体建模与仿真分析

3.3.1　生物系统个体的形式化定义

生物系统中的个体是一个自治的实体，在群体智能优化理论与模型中是基本的功能模块。群智能系统中的大部分个体在结构和功能上是同构的，它们遵循简单的规则进行交互和协作，每个个体只能得到局部信息，仅具有局部规划和推理能力。在进行问题求解时，个体的目标通常与整体的目标并不相同，甚至个体并没有明确的行为目标，而是通过个体的活动，使求解问题的知识在群体层面涌现出来。要实现个体的功能特性，需要解决的问题包括：个体应该由什么样的模块组成；个体的规则如何设定；个体目标如何确定；个体之间如何交互信息；个体感知到的外部环境信息如何影响它的行动和内部状态；等等。

个体的描述分为自然属性和社会属性两个部分。自然属性包括个体的位置、大小、能量、速度、目标、行为集合等特性。社会属性是指个体能与其他个体或环境发生交互，交互的结果会对改变个体的自然属性。社会属性实现方式采用认知结构和反应结构相结合的方

式，如图 3-12 所示。即个体接收外界的信息后，会对其进行简单的分析，然后根据自己的当前状态，通过个体目标和行为规则的判断，来决定自己的下一步行动。

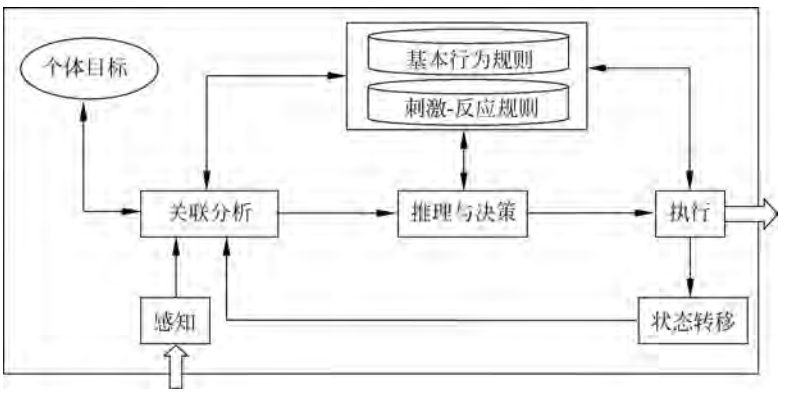


图 3-12　个体社会属性

生物系统中个体的形式化描述如下：

$$\text{Individual}_i = <\text{IID},\text{INOBJ},\text{STATUS},\text{BHV},\text{MESS},\text{ACT},\text{TANS}>$$

式中，(1) IID 为个体标识符，用来识别该个体的身份。

$$\text{IID} = (\{\text{II},(j,x)\} \mid 1 \leqslant i \leqslant \text{INUM},\quad 1 \leqslant j \leqslant \text{PNUM},\quad 1 \leqslant x \leqslant \text{SwarmSize}_j)$$

其中 II 表示当前个体的绝对身份，即代表整个系统中的标识；(j,x)描述的是当前个体是第 j 个种群的第 x 个个体。此集合表示当前个体的相对身份，即代表个体在其群体中的标识。SwarmSize_j 表示第 j 个群体的个体总数。

(2) INOBJ 为个体生存目标，如觅食目标、执行任务的目标等。

(3) STATUS 为个体自然状态集合，主要涉及个体的位置、速度和能量，除此以外还可以包含个体年龄、性别等基本信息。

$$\text{STATUS} = (P,V,E)$$

其中，$P=(p_1,p_2,\cdots,p_d,\cdots,p_D)$表示个体空间位置，$p_d \in [L_d,U_d]$，$L_d$ 和 U_d 分别为第 d 维搜索空间的下限和上限。

$V=(v_1,v_2,\cdots,v_d,\cdots,v_D)$表示个体速度。其中 $v_d \in [V_{\min,d},V_{\max,d}]$，$V_{\min,d}$ 和 $V_{\max,d}$ 分别为第 d 维搜索空间的最小和最大速度。

$E=F(\text{Individual}_i)$表示个体能量。

(4) BHV 为个体行为集合，涉及个体基本行为规则，如觅食策略、繁殖方式、迁徙等策略。

(5) RULE 为个体行为规则的集合。

(6) MESS 为个体接收和发出信息的集合，$\text{MESS} = <\text{MESS}_{\text{IN}},\text{MESS}_{\text{OUT}}>$。其中 MESS_{IN} 表示从外部接收到消息的集合，MESS_{OUT} 表示个体对外发出消息的集合。

(7) ACT 为个体活动函数，$\text{ACT}: \text{STATUS} \times \text{MESS}_{\text{IN}} \xrightarrow{\text{RULE}} \text{BHV}$。表示个体在当前状态下接受了特定的信息后，按照相应的刺激-反应规则采取行动。

(8) TANS 为状态转换函数，$\text{TANS}: \text{STATUS}_i \times \text{BHV} \xrightarrow{\text{RULE}} \text{STATUS}_{i+1}$。表示个体在执行一定行为后，将按照一定的规则转换到下一个状态。

3.3.2 典型生物个体行为的建模与仿真分析

1. 局域限制搜索模型与仿真

大肠杆菌 E. Coli 是细菌的一种，是单细胞生物体，它自身有一个觅食行为控制系统，保证它向着食物源的方向前进并及时地避开有毒的物质。例如，它会避开碱性和酸性的环境向着中性的环境移动。通过对每一次状态的改变进行效果评价，控制系统进而为下一次状态的改变（例如，前进的方向和前进步长的大小）提供信息。本书对大肠杆菌的这种自适应特性进行建模与仿真，用来验证生物个体行为中的局域限制搜索策略。本书设计的自适应规则是将算法执行过程（即细菌种群觅食过程）分为探索阶段和开发阶段。在算法运行初期的探索阶段，整个细菌种群都以较大的趋化步长在解空间里进行大范围的全局探索，当解的精度达到一定的阈值，即菌群定位全局最优解于一定的较小区域，整个种群进入下一开发阶段；此时每个细菌个体在其上一阶段发现的最好位置重新初始化，整个种群以与目标解精度相应的较小的趋化步长在该区域进行小范围开发，同样，当解的精度达到该阶段要求的阈值时，算法进入下一开发阶段；如此循环，直到解的精度达到最终要求。自适应规则的伪代码如下：

```
1:  IF (t mod n = 0)then
2:    IF (f_best < ε(t))then
3:      C(t + 1) = C(t − n)/α;
4:      ε(t + 1) = ε(t)/β;
5:    ELSE
6:      C(t + 1) = C(t − n);
7:      ε(t + 1) = ε(t − n);
8:    END IF
9:  ELSE
10:  C(t + 1) = C(t);
11:  ε(t + 1) = ε(t);
12: END IF
```

其中，t 表示算法当前的迭代次数；f_{best} 表示当前种群找到的最优解；$\varepsilon(t)$ 表示当前要达到的求解精度；n、α 和 β 是 3 个自定义的常数。

本书选择 Sphere、Rosenbrock、Rastrigrin 和 Griewank 这 4 个函数作为细菌的觅食环境，仿真的参数设置种群规模为 $S=1$，表 3-4 列出了其他参数的设置。对于每一个觅食环境，E. Coli 个体都进化 1000 代。从图 3-13 的仿真结果中可以观察到，在单峰觅食环境中，E. Coli 个体首先在探索状态定位全局最优解所在区域，然后进入开发状态对该区域进行高精度搜索；在多峰觅食环境中，E. Coli 个体不断地在探索状态和开发状态之间转换，即个体在发现某一区域的局部最优之后，就会逃离该区域，对下一个区域进行探索和开发，直到找到全局最优解为止。

表 3-4　自适应规则的参数设置

函　　数	S	C_{initial}	$\varepsilon_{\text{initial}}$	K_u	α	β
Sphere	100	0.1	100	20	10	10
Rosenbrock	100	0.1	100	20	10	10
Rastrigrin	100	0.1	100	20	10	10
Griewank	100	10	100	20	10	10

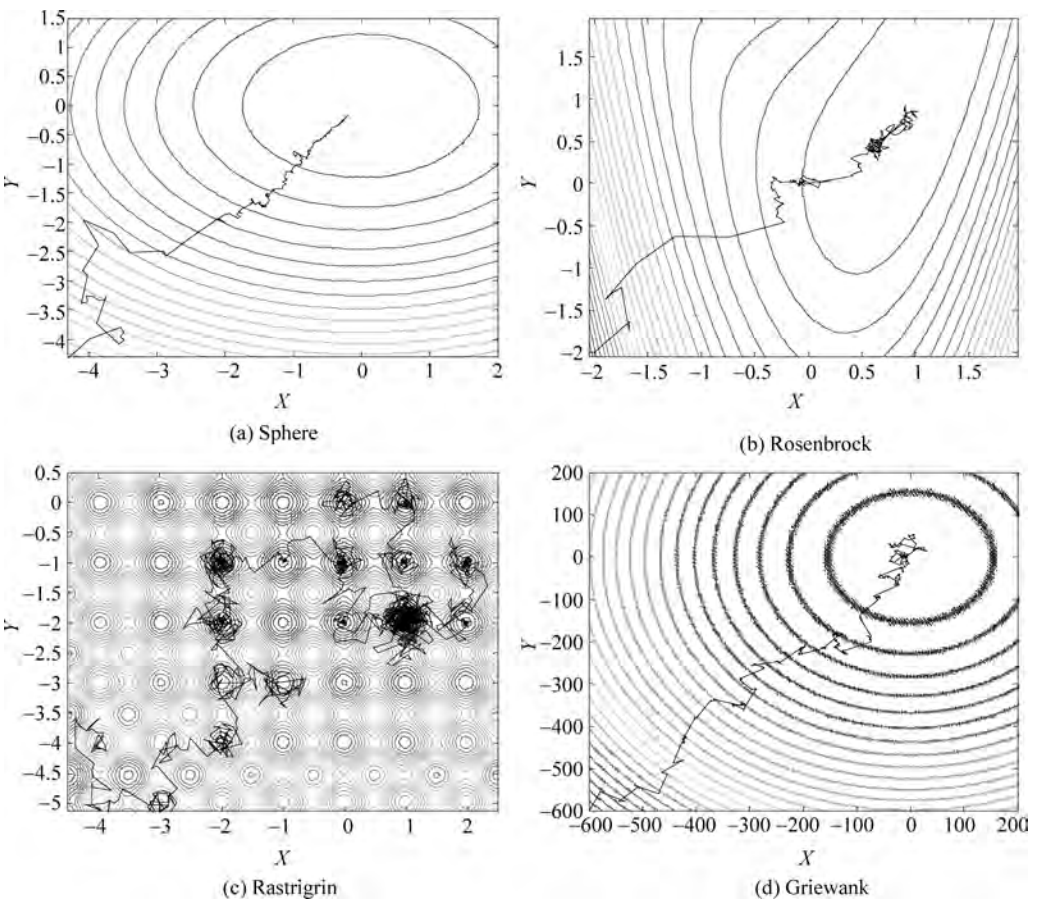


图 3-13　细菌个体自适应行为仿真

2. 繁殖与死亡模型与仿真

E. Coli 通过细胞壁直接从环境中获得营养。当吸入足够的营养后，E. Coli 将其转换成能量，这些能量可以保证 E. Coli 复制 DNA 的消耗。E. Coli 的繁殖遵照简单的细胞分裂规律，大肠杆菌随着自身的生长不断变长，然后在其中部一分为二，变为两个相同的子代。如果给予合适的条件，自然界的大肠杆菌将每隔 30min 进行一次繁殖行为。

为了描述这一过程，首先进行如下假设：

（1）细菌生存的时间越长，其繁殖后代机会越多。

（2）细菌获得更多食物后，其生活周期将延长。

（3）当细菌找不到食物时，其生活周期将缩短。

在仿真中，首先设定能量初始化时的上下限 α 和 β，即所有的细菌个体能量限定在$[\alpha,\beta]$，然后在此范围内随机给定每个细菌一起始能量，细菌通过不断与环境交互获得能量或消耗能量。进行繁殖操作后，父代的能量和子代的能量均分。消亡操作时，将消亡的细菌直接从环境拿走。繁殖与消亡过程的模型见图 3-14，具体实现伪代码如表 3-5 所示。

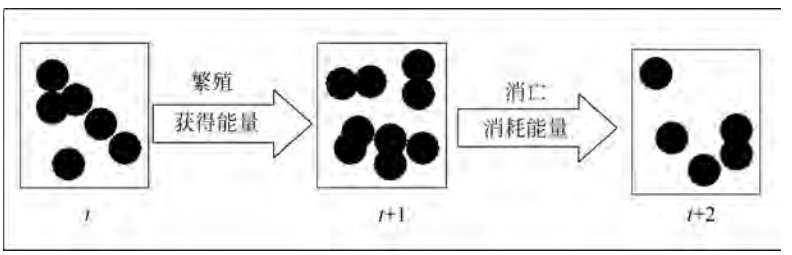


图 3-14　细菌的繁殖与消亡模型

表 3-5　复制与消亡操作伪代码

```
FOR each alive bacteria i
IF F(B_i(t+1))< F(B_i(t))                    //感知环境
E(i) = E(i) + Benifits_from_action(i)
IF P_r > rand() B_i(t+1) = B_i(t) + R_i(t) END
IF E(i)≥β
        Division;     // E.Coli 分裂成两个相同的子代，一个位于原来父代的位置，另一个随机
                      //放置在环境领域中
        E_{P+1}(t+1) = E_i(t)/2;        //P 为当前群体规模
        E_i(t+1) = E_i(t)/2;
END
ELSE
E(i) = E(i) - Costs_from_action(i)
IF P_r > rand() B_id(t+1) = B_id(t) + O_i(t) END
IF E(i)≤α
Die;                                         //从人工环境中删除
 E(i) = 0
END
END
END
```

3. 植物根系在土壤中生长行为的仿真

20 世纪 60 年代中期，为定量化研究植物的生长规律，研究人员开始了植物生长的模拟研究。所建立的模型通过对植物生理生态过程的模拟能够预测不同环境条件下生长的植物的某些综合指标，并在植物形态结构和环境因素的时空变异对植物生长的影响等方面也进行相应地处理[17,18]。这类模型与专家系统结合对农业生产等领域具有重要的指导意义。

到目前为止，虚拟植物研究的主要工作集中在地上部分，根系是其薄弱环节。植物主要从土壤获取水分、养分，而其获取能力一方面依赖于根系的长度和表面积，另一方面是根系的空间分布特征，它决定了根系获取水、肥的空间范围和与其他植物根系的资源竞争能力。植株个体的生命活动是以整体进行的，植物在生长过程中总是不断调节根冠关系以适应变化的环境，抵御逆境。由于根系是植物与环境进行物质与能量交换的重要界面，要增强虚拟模型模拟水、

肥等环境因子对植物生长影响的机制，也必须加强对根系的研究。根系势必成为虚拟植物未来的研究重点。由于植物根系隐藏于地下，因此很难对其进行直接的观察，这样通过智能算法根据其生长方式进行仿真就显得很有意义。根系的生长是由一粒种子开始，生长则可以看作是互不交叉、独立生长的多条根须向着营养丰富区域的延伸，是典型基于个体的行为方式。

根据植物根系的生长特点，设计出植物根系自适应生长算法，既可以用于植物根系生长行为的仿真，也可以用于生物启发优化算法设计。根据植物根系生长自适应算法对植物根系在土壤中的仿真可以将测试函数当作植物生长的土壤环境，将函数最优解当作土壤中最肥沃的那一个点，模拟真实植物根系的生长状况，建立节点和根须在不同肥沃程度土壤环境下的生长演绎方式。本节建立了植物生长行为仿真的伪代码，如表 3-6 所示。其中的具体参数设置请参考 3.5 节。

表 3-6　生长行为仿真的伪代码

```
Set t: = 0;
INITIALIZE.
WHILE (the termination conditions are not met)
FOR (each root tip x)
Select S root tips pg
END FOR
FOR (each root tip selected pg)
Produce n new growing points
END FOR
FOR (each root tip x)
Produce new root tips
END FOR
Tune the parameter δ
Set t: = t + 1;
END WHILE
```

本节采用 Ackley 函数作为土壤环境，函数如下：

$$f_7(x)=20+\mathrm{e}-20\mathrm{e}^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)}-\mathrm{e}^{\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)}$$

仿真出植物根系自适应生长算法的寻优过程图，如图 3-15 和图 3-16 所示。

3.3.3　个体环境间作用关系描述与规则模型

个体环境间作用关系可以看作主体与外界刺激之间的关系，这样就要得到主体的一般描述，第一步就要回到有关适应性主体的描述。用规则作为描述的工具，需要更加严格地把规则作为定义主体的一种正规手段。由于规则应该是成功的、统一的描述工具，无论对于具有什么外部形式的主体都能适用，因此它必须满足三个标准：

(1) 规则必须使用单一的语法去描述所有的 CAS 主体。

(2) 规则的语法必须规定主体间的所有相互作用。

(3) 必须有一个可接受的程序以适应性地调整规则。

与前面一样，先来看看最简单的一类规则：IF(一些条件为真)THEN(执行一些动作)。IF/THEN 规则用在很多不同的领域：在心理学规则上，它们被称为刺激-反应规则(见

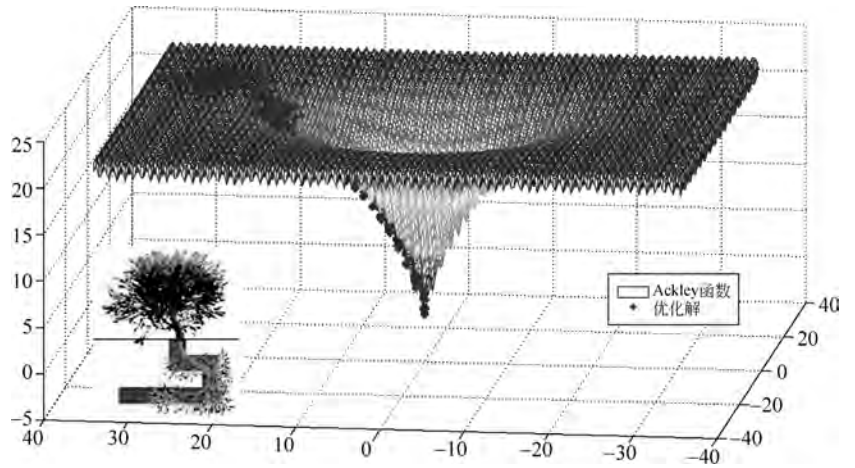


图 3-15　植物根系生长仿真图

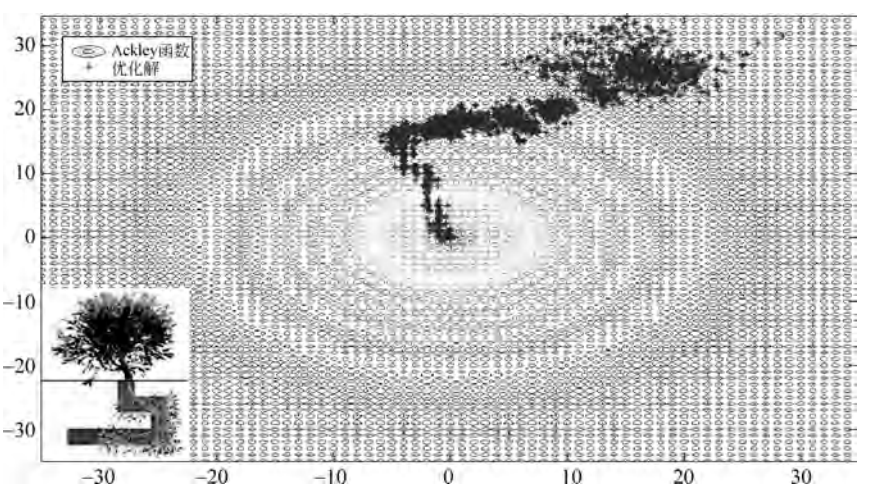


图 3-16　植物根系生长等高线图

图 3-17)；在人工智能上，它们被称为条件-行动规则；在逻辑学上，它们被称为产生规则。我们现在的目的是要为 IF/THEN 规则找到一个简单的语法，一种适用于各种主体的语法。以后，我们将加入一些简单的修改，使 IF/THEN 规则足够强大，能够为任何主体建立可以在计算机上模拟的模型。

我们使用的 IF/THEN 规则的语法，严格依赖于主体与其环境的相互作用。我们从输入端开始。一般来说，主体通过对于刺激的分类来感知环境。如果主体是抗体，则刺激就是抗原表面分子的构型，即标识；如果主体是哺乳动物，则刺激就来自它的各种感官；如果主体是禽类，则刺激就是食物、危险、同类或雏鸟；等等。通常情况下，主体被刺激所包围，它所收到的信息比能够使用的要多得多。

图 3-17　刺激-反应(IF 小飞行物离开 THEN 生物个体头向左转 15°)

那么，主体的第一个任务就是过滤其周边环境产生的、大量涌入的信息。为了描述这个过滤作用，此处采用通常的观点，即环境通过一组探测器(detector)将信息传递给主体。

最简单的一种探测器，就是用来感觉环境的特殊性质，当特性出现时转向“开”，否则转向“关”（见图 3-18）。也就是说，探测器是一个二进制装置，它传递环境的一比特的信息。这种探测器用来感知环境似乎很有限，但任意大量的信息却可以通过足够多的一组探测器来传递。

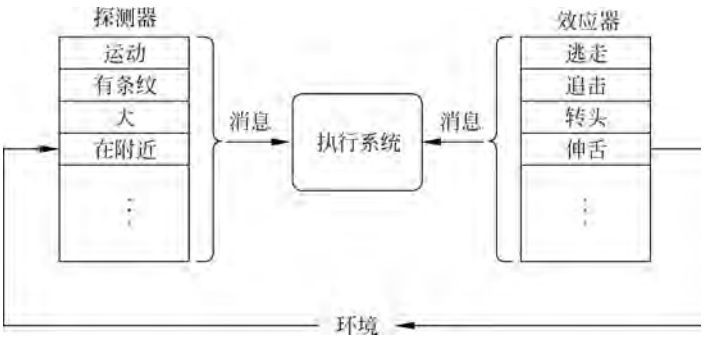


图 3-18　执行系统的探测器和效应器

用这种方式描述与主体有关消息的输入和输出，似乎能以相同的方式很方便地处理主体规则的相互作用。制定规则的相互作用是关键的一步，能够给予简单的 IF/THEN 规则以一种程序设计语言的全部能力。因为一个 IF/THEN 规则与另一个相互作用，必然会是一个规则的 IF/THEN 部分导致对由另一个规则 THEN 部分所指定的动作十分敏感。如果我们把每个规则想象成为某种微主体（micro-agent），就可以把消息的输入输出作用扩展到相互作用上。设想一下，每个规则都有自己的探测器和效应器，或者再进一步，每个规则就是一个消息处理工具。那么，规则就有下面的形式：

IF（有合适的消息）**THEN**（发出指定的消息）

也就是说，主体现在已经被描述为一组消息处理规则。有些规则作用于探测器产生的消息，处理环境信息，有些规则作用于其他规则发出的消息。有些规则通过主体的效应器，发出作用于环境的消息，而有些规则发出激活其他规则的消息（见图 3-19）。

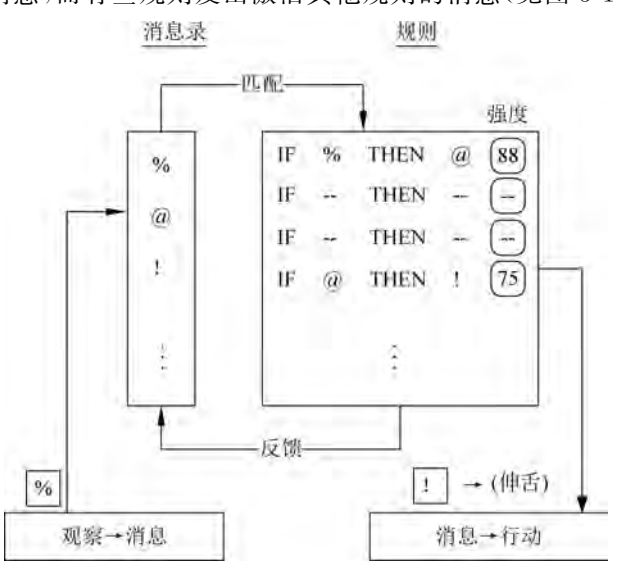


图 3-19　消息传递执行系统

3.4 细菌自适应觅食优化算法

3.4.1 算法的基本思想与流程

1. 基本思想

细菌是一种单细胞生物体，是构成地球上各种高级生命体的最简单的形体，存在于地球上至少具有 35 亿年的历史，见证并参与了漫长的生物进化过程，其行为模式和演变过程充满了复杂性、动态性、混沌性等诸多特征。

大肠杆菌(Escherichia Coli，简称 E. Coli)是目前为止研究较为透彻的微生物之一[22-25]。大肠杆菌外形为杆状，直径约 1μm，长约 2μm，质量为 1kg(10^{-12}g)。它身体的 70%由水组成，包括细胞膜、细胞壁、细胞质和细胞核。细菌的表面遍布着纤毛和鞭毛，纤毛是直径约 0.2μm，长为数微米至数十微米的能运动的突起状细胞器，纤毛的作用是细菌之间某种基因的传递。而鞭毛是具有细长而弯曲的丝状物，它的长度常超过菌体若干倍，鞭毛自细胞膜长出，游离于细胞外，是用来帮助细胞移动的[26]。大肠杆菌的外形和结构如图 3-20 所示[27]。

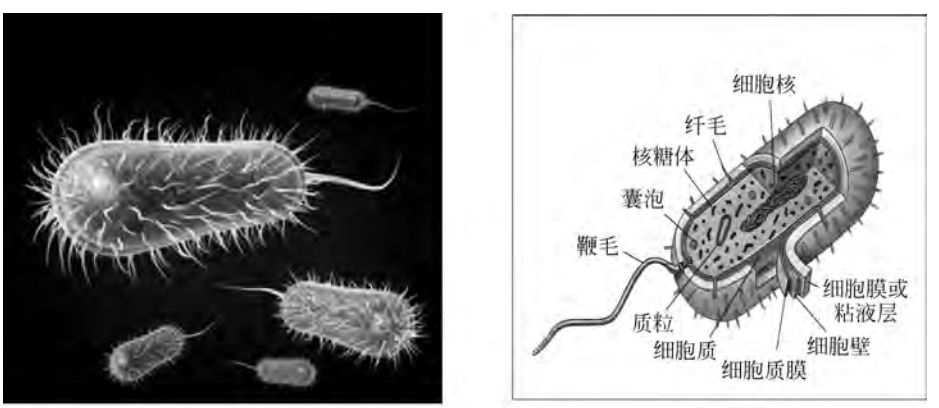


图 3-20　大肠杆菌(E. Coli)

为了提高细菌觅食算法的性能，使其能够胜任求解实际的工程应用问题，在以往研究工作的基础上，将细菌的趋化行为和自适应搜索策略相结合，提出了一种新型群体智能优化方法——细菌自适应搜索算法(Adaptive Bacterial Foraging Optimization，ABFO)[28-30]。

2. ABFO 的群体感应机制

E. Coli 的运动是依靠其表面上的鞭毛大约 100～200rad/s 的转动实现。当鞭毛逆时针摆动时，会给 E. Coli 一个推动力，从而使其按照原来的运动方向向前直行[swim，如图 3-21(a)所示]；与之相反，当鞭毛顺时针摆动时，将会向细菌施加一个拉动的力量，使细胞产生翻转[tumble，如图 3-21(b)所示]，从而改变其运动方向。生物学研究表明，E. Coli 的觅食(即趋化)行为如图 3-22 所示，其主要包括以下步骤：

(1) 在环境中随机游动，寻找可能存在食物源的区域。

(2) 发现食物源后，朝着有丰富食物的方向开始前进。

(3) 根据其在觅食搜索过程中得到的经验，适当改变其运动方向。

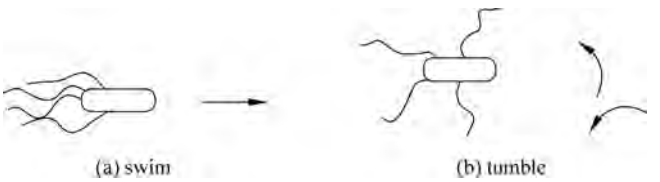


图 3-21　细菌的趋化行为

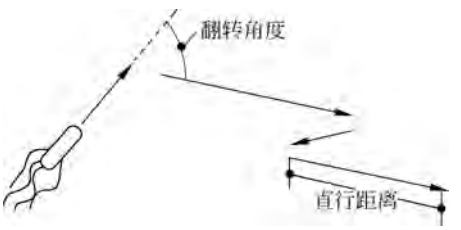


图 3-22　E. Coli 个体的觅食过程

(4) 在该区域消耗掉一定量的食物后,迁移到一个更理想的区域。

3. ABFO 的自适应搜索策略

在 ABFO 中需要自适应搜索策略来动态地控制 ABFO 模型中细菌的趋化步长,从而在算法运行时能动态地平衡细菌的探索和开发能力。

自适应搜索策略的对象是单个细菌个体控制规则,使每个个体根据其所处的环境状态动态地在两种觅食状态(即探索状态和开发状态)之间转换。每个细菌会首先处于探索状态,即个体对解空间以较大的趋化步长进行大范围不精确搜索;当个体定位到解的精度达到一定阈值时,个体会进入开发状态,即对该位置邻近区域以较小的趋化步长进行精确开发;一旦对该区域搜索失败或已发现该区域内的局部最优解,个体会重新以较大的趋化步长迅速离开该区域,并开始对新的区域进行下一步探索和开发;如此循环,直到整个种群的最优解精度达到最终要求。自适应规则的伪代码如下:

```
1:  FOR (each bacterium i) IN PARALLEL
2:    IF (Criterion - 1) then
3:        C^i(t + 1) = C^i(t)/α;      //exploit
4:        ε^i(t + 1) = ε^i/β;
5:    ELSE IF (Criterion - 2) then
6:        C^i(t + 1) = C_initial;      //explore
7:        ε^i(t + 1) = ε_initial;
8:    ELSE
9:        C^i(t + 1) = C^i(t);
10:       ε^i(t + 1) = ε^i(t);
11:   END IF
12: END FOR IN PARALLEL
```

其中,t 表示算法当前的迭代次数;$C^i(t)$表示个体 i 当前的趋化步长;$\varepsilon^i(t)$表示个体 i 当前的求解精度;C_{initial} 和 $\varepsilon_{\text{initial}}$ 分别表示每个个体在重新进入探索状态时的初始趋化步长和初始求解精度;α 和 β 是自定义的常数。

3.4.2 算法的形式化描述

在自适应细菌算法中，细菌个体的属性和在计算模式中的行为方式是核心，细菌个体通过与环境以及其他个体进行交流，在这种交流的过程中“学习”或“积累”经验，并且根据学到的经验改变自身的行为状态与行为方式。细菌个体被定义为主动的动态实体。用集合表示如下：

$$\mathrm{Bacs}=\{\mathrm{Bacs}_i^t \mid i=1,2;t=1,2,\cdots,T\}$$

其中，

$$\mathrm{Bacs}_i^t=\langle \partial_i^t,\varepsilon_i^t,\phi_i^t,\lambda_i^t,\tau_i^t,\delta_i^t,\varphi_i^t\rangle$$

这代表一个或几个觅食的细菌，i 是细菌编号，这里 i 允许的最大数值为 2；T 代表整个觅食寻优过程的时间；Bacs_i^t 的具体描述见表 3-7。

表 3-7 细菌个体的属性

∂_i^t	细菌个体 i 的位置	ε_i^t	细菌个体 i 的能量
ϕ_i^t	细菌个体 i 状态成熟、消亡等	τ_i^t	细菌个体 i 行为模式直行或翻转
δ_i^t	细菌个体 i 处于直行时的步长	φ_i^t	细菌个体 i 处于翻转时的角度
λ_i^t	决定细菌个体 i 行为模式的参数		

当 E. Coli 个体翻转时，根据下式进行位置更新：

$$\varphi_i^{t+1}=\mathrm{rand}() \tag{3-1}$$

$$\partial_i^{t+1}=\partial_i^t+\delta_i^t N(\varphi_i^{t+1}) \tag{3-2}$$

当 E. Coli 个体直行时，根据下式进行位置更新：

$$\partial_i^{t+1}=\partial_i^{t+1}+\delta_i^t N(\varphi_i^{t+1}) \tag{3-3}$$

其中，$N(*)$是角度的标准化函数。细菌个体的这些属性状态是随着菌群与环境之间交互不断变化的。

3.4.3 ABFO 算法实现步骤

ABFO 算法的基本实现步骤如下：

(1) 初始化。包括细菌种群规模 S、自定义的常数 n、α 和 β、初始趋化步长 C_{initial} 和收敛精度 $\varepsilon_{\mathrm{initial}}$，并随机给定所有细菌个体的初始位置。

(2) 群体感应。细菌群体执行群体感应操作时，每个 E. Coli 个体计算其他个体释放的信号分子浓度，然后更新下一步要翻转的角度。

(3) 趋化操作。根据群体感应步骤(2)所确定的前进方向，对每一个细菌执行一个趋化步长的游动操作，计算个体的适应度值，如果好于该个体上一代的适应度值，个体将在该方向继续直行。

(4) 自适应进化。根据自适应规则对菌群所有个体的趋化步长进行进化操作。

(5) 复制与消亡。为了加快收敛速度，将细菌群体根据适应度值排序，对适应度值高的 $S/2$ 个体进行复制，对适应度值低的 $S/2$ 个体进行灭绝操作。

(6) 迁徙操作。对每一个个体以一定的概率随机在搜索空间初始化，以增加种群多样性。

(7) 检索是否满足条件，若不满足则返回步骤(2)，否则输出结果，算法终止。

ABFO 算法的流程图由图 3-23 给出，其中 S 表示菌群规模、t 表示当前迭代次数、X^i 表示个体 i 在解空间的位置、N_s 表示每次迭代个体在营养梯度上游动的最大次数、flag^i 表示个体适应度连续没有提高的代数。

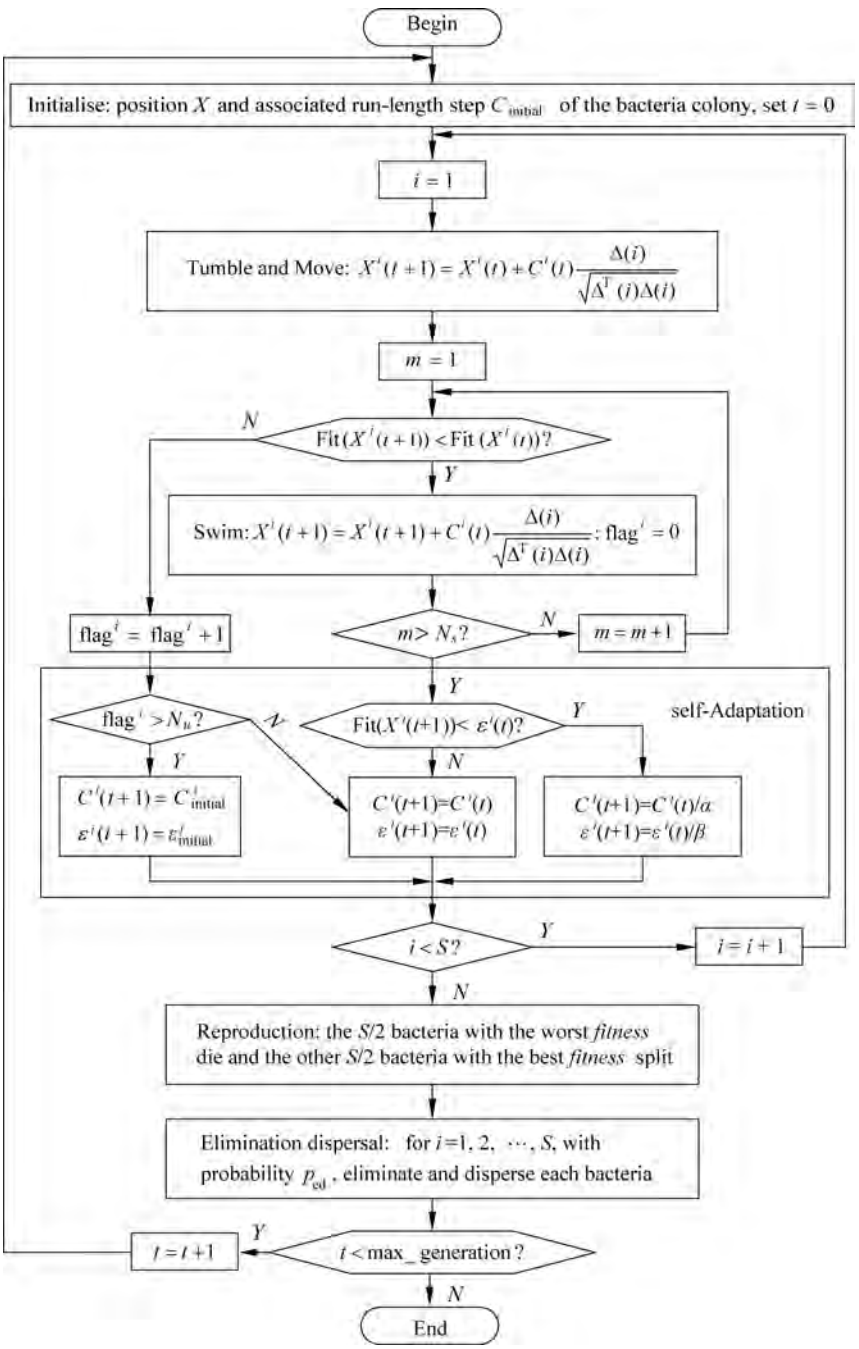


图 3-23　ABFO 算法的流程图

3.4.4 算法效能分析

1. 函数优化

为了测试菌群自适应觅食算法的优化性能，选择了常用的一组标准测试函数[31,32]，它们分别是 Sphere、Rosenbrock、Rastrigrin 和 Griewank 函数。这 4 个函数的求解针对的都是最小化问题。

实验中将两种新提出的算法 ABFO 与基本粒子群算法(Particle Swarm Optimization，PSO)、细菌觅食算法(Bacterial Foraging Optimization，BFO)及实数编码的遗传算法(Genetic Algorithm，GA)进行了比较。

ABFO 算法的参数设置见表 3-8，文献[33]的 BFO 算法参数设置见表 3-9。

表 3-8 ABFO 算法的参数设置

函　数	ABFO					
	S	C_{initial}	$\varepsilon_{\text{initial}}$	K_u	α	β
Sphere	100	0.1	100	20	10	10
Rosenbrock	100	0.1	100	20	10	10
Rastrigrin	100	0.1	100	20	10	10
Griewank	100	10	100	20	10	10

表 3-9 BFO 算法的参数设置

函　数	BFO				
	S	N_c	N_s	N_{re}	N_{ed}
Sphere	100	100	4	5	2
Rosenbrock	100	100	4	5	2
Rastrigrin	100	100	4	5	2
Griewank	100	100	4	5	2

PSO 的参数设置参考文献[34,35]：最大速度 $V_{\max}$、最小速度 $V_{\min}$ 为自变量边界范围的一半，学习因子 c_1 和 c_1 设为 2.0。惯性权重对 PSO 算法的收敛行为至关重要，合适的惯性权重有助于保持算法的开发与探索能力从而获得优秀解。在试验中 PSO 惯性权重设置为随着迭代次数的增加从 0.9 线性减少到 0.4。

实数编码 GA 算法的参数设置与文献[36]相同，使用中间交叉和高斯变异，选取交叉概率 $P_c=0.9$，变异概率 $P_m=0.01$。

4 种算法对二维问题的求解结果见表 3-10，所有算法针对每个二维问题的优化过程收敛曲线见图 3-24～图 3-27。

对于较为简单的函数 Sphere，所有的算法都能迅速收敛到全局最优解。由于这个函数是单峰函数，因此只有一个全局最优解，迅速收敛到全局最优解并不是太困难。然而比较而言，ABFO 方法具有最快的收敛速度。

函数 Rosenbrock 是一个典型的优化问题，也称为“香蕉问题”。它的全局最优解隐藏于一条狭长扁平的通道中。发现这条通道不是很困难，但是发现通道中的最优点却非常困难。相对 BFO 与 GA，PSO、ABFO 有着较快的收敛速度，获得的平均最优解也要好于 BFO 与 GA。

表 3-10　所有函数二维问题的测试结果

2D		BFO	ABFO	PSO	GA
f_1	最优值	5.6423e－007	0	5.0576e－124	2.8638e－048
	最差值	3.3404e－005	0	2.4892e－114	1.0587e－045
	平均值	1.4291e－005	0	8.5226e－116	2.3105e－046
	标准差	9.6035e－006	0	4.5408e－115	2.7237e－046
	排序	5	1	2	4
f_2	最优值	1.5294e－006	7.8886e－031	0	6.8564e－010
	最差值	9.2935e－004	2.1905e－014	5.3075e－025	3.8578e－005
	平均值	1.7353e－004	7.3528e－016	3.9893e－026	1.9021e－006
	标准差	2.1344e－004	3.9983e－015	1.1947e－025	7.0743e－006
	排序	5	3	2	4
f_3	最优值	9.0825e－004	4.6509e－011	0	0
	最差值	0.1256	2.7349e－009	0	0
	平均值	0.0259	1.2655e－009	0	0
	标准差	0.0306	7.4950e－010	0	0
	排序	5	4	1	1
f_4	最优值	0.0296	0	0.0311	0
	最差值	2.9974	7.7411e－005	3.8791	0.0271
	平均值	0.9975	5.1542e－006	1.1178	0.0063
	标准差	0.8142	1.5892e－005	1.0521	0.0061
	排序	4	2	5	3
平均排序		4.75	2.5	2.5	3
最终排序		5	2	2	3

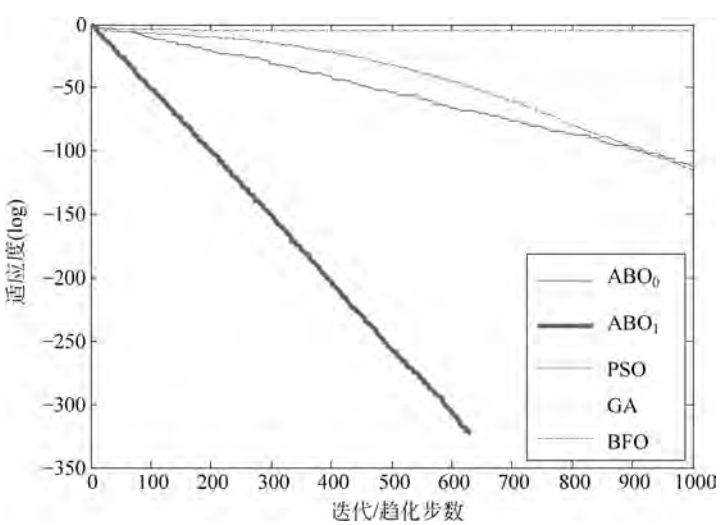


图 3-24　2D Sphere 函数的收敛曲线比较图

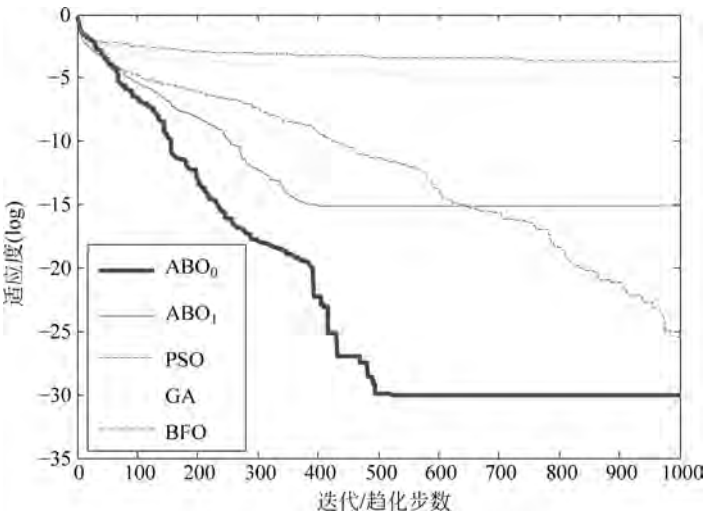


图 3-25　2D Rosenbrock 函数的收敛曲线比较图

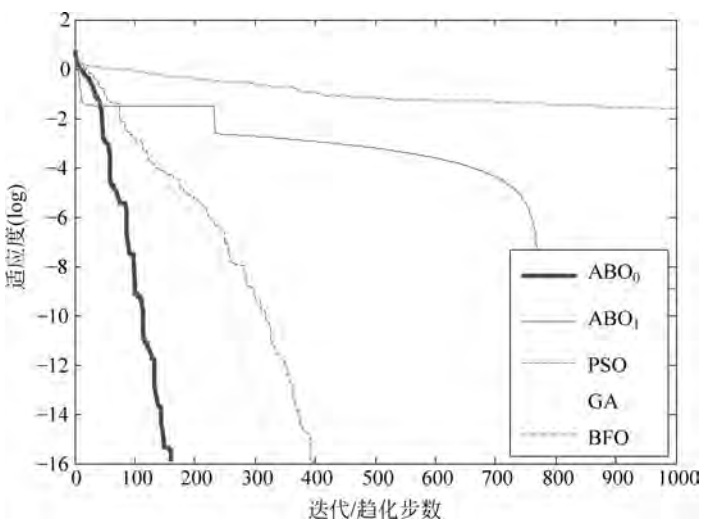


图 3-26　2D Rastrigrin 函数的收敛曲线比较图

对于复杂的多模态函数 Rastrigrin。除了 BFO，其他 3 种算法也都具有较好的性能。ABFO 方法始终获得了较好的求解性能，但是在收敛速度和最后的求解精度上并不如 GA 和 PSO。

函数 Griewank 体现了与 Rastrigrin 较为类似的结果。ABFO 以非常快的速度向近优解收敛，在 1000 代以后仍然没有停滞陷入局部最优。GA 表现出相对较好的寻优性能，但在 400 代以后停滞陷入局部最优。PSO 和 BFO 对 Griewank 函数的求解是失败的，它们的收敛曲线几乎表现为一条直线。

4 种算法在 10 维问题上的求解结果与二维问题相近，见表 3-11 和图 3-28～图 3-31。

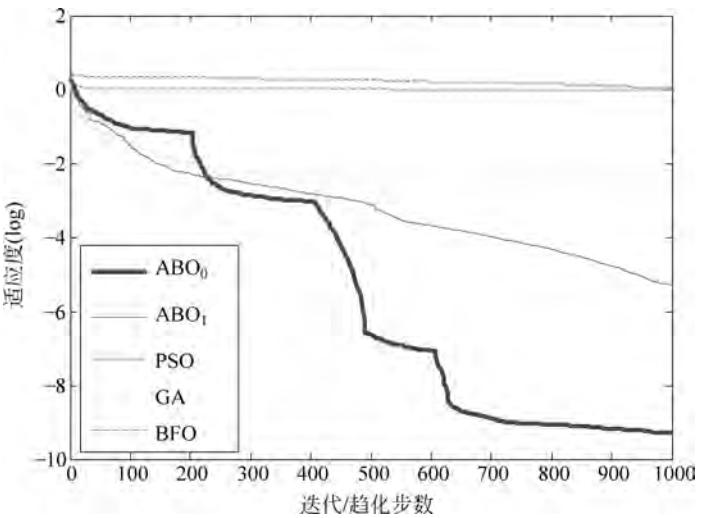


图 3-27　2D Griewank 函数的收敛曲线比较图

表 3-11　所有算法 10 维问题的测试结果

10D		BFO	ABFO	PSO	GA
f_1	最优值	0.0131	5.0821e−071	9.7085e−052	9.2242e−004
	最差值	0.0456	1.3038e−066	1.1714e−045	0.0654
	平均值	0.0325	9.3583e−068	7.6021e−047	0.0133
	标准差	0.0086	2.5360e−067	2.2319e−046	0.0149
	排序	5	1	3	4
f_2	最优值	7.8436	0.3620	0.4578	6.6031
	最差值	11.3192	4.3232	5.2075	10.1496
	平均值	9.8819	1.8660	0.8852	8.5506
	标准差	1.0079	1.8064	1.0667	0.7340
	排序	5	3	2	4
f_3	最优值	17.2125	5.9785	2.9849	1.0413
	最差值	29.0473	32.0221	17.9092	12.3464
	平均值	22.6397	15.5429	10.8119	6.0909
	标准差	3.1330	8.4543	3.8555	2.9628
	排序	5	4	3	2
f_4	最优值	39.0662	0.0910	52.3209	0.1405
	最差值	133.2687	0.7516	133.6037	0.9791
	平均值	88.7932	0.3551	100.0376	0.4061
	标准差	25.0455	0.1605	17.2171	0.1875
	排序	4	2	5	3
平均排序		4.75	2.5	3.25	3.25
最终排序		5	2	3	3

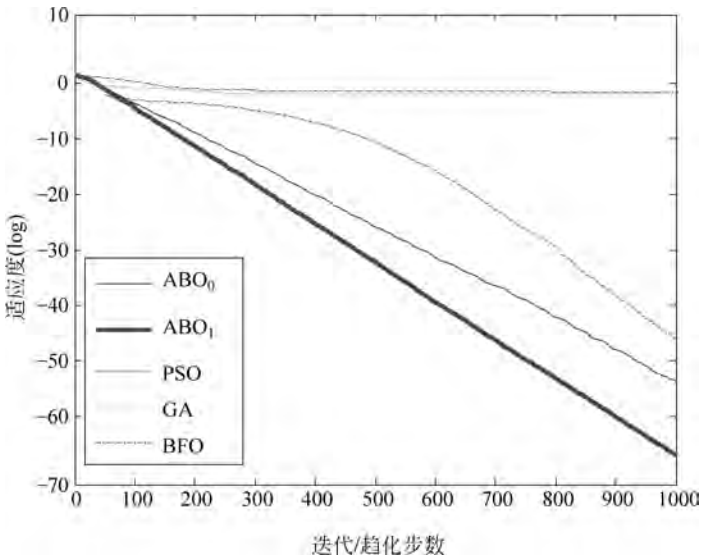


图 3-28　10D Sphere 函数的收敛曲线比较图

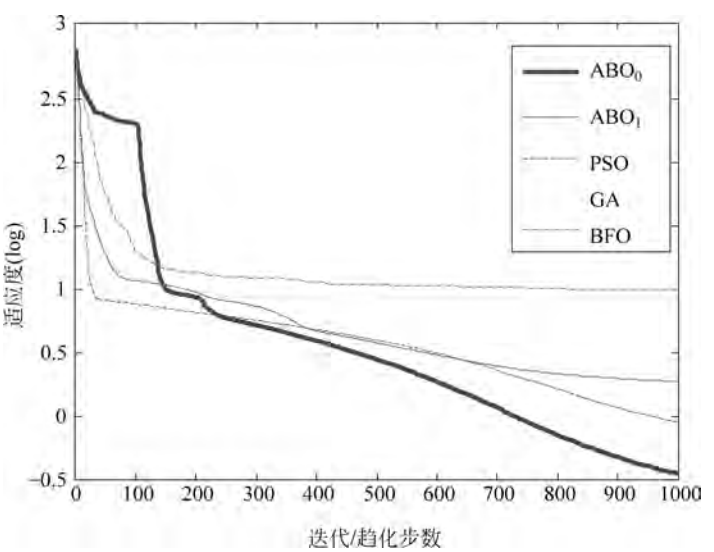


图 3-29　10D Rosenbrock 函数的收敛曲线比较图

2. 自适应动态性

为了进一步对 ABFO 算法的动态特性进行研究，本实验对 ABFO 模型中 E. Coli 在觅食过程中的基本行为模式进行了仿真观察。

对侧重个体多样性的 ABFO 模型在觅食过程中的个体行为为模式进行了仿真，本书选择 Sphere、Rosenbrock、Rastrigrin 和 Griewank 函数作为细菌的觅食环境，E. Coli 个体自适应觅食过程的趋化步长动态特性如图 3-32。仿真的参数设置除了种群规模为 $S=1$ 外，其他参数如表 3-8。对于每一个觅食环境，E. Coli 个体都进化 1000 代。从仿真结果中可以观

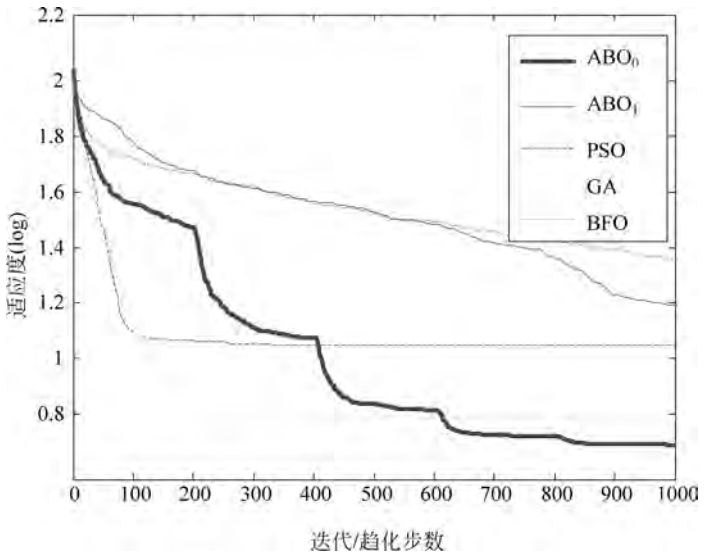


图 3-30　10D Rastrigrin 函数的收敛曲线比较图

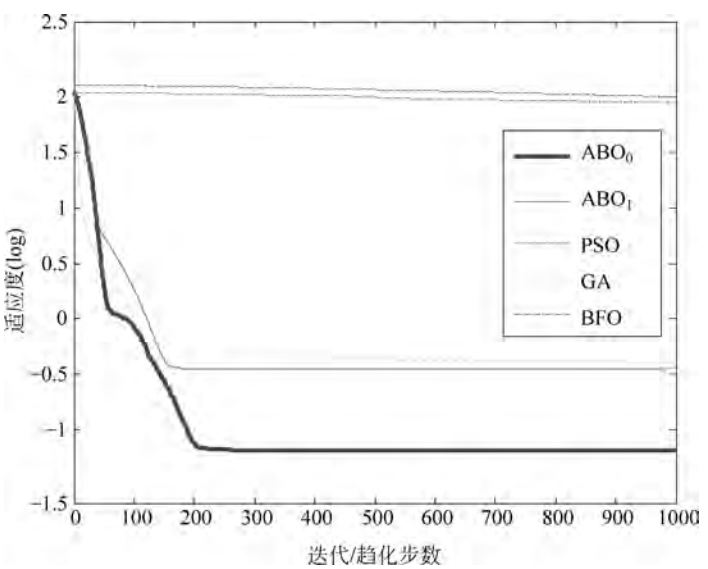


图 3-31　10D Griewank 函数的收敛曲线比较图

察到：在单峰觅食环境中，E. Coli 个体首先在探索状态定位全局最优解所在区域，然后进入开发状态对该区域进行高精度搜索；在多峰觅食环境中，E. Coli 个体不断地在探索状态和开发状态之间转换，即个体在发现某一区域的局部最优之后，就会逃离该区域，对下一个区域进行探索和开发，直到找到全局最优解为止。

从上述仿真实验可以看出，ABFO 模型的动态特性与自然界的自适应搜索规则相符合。

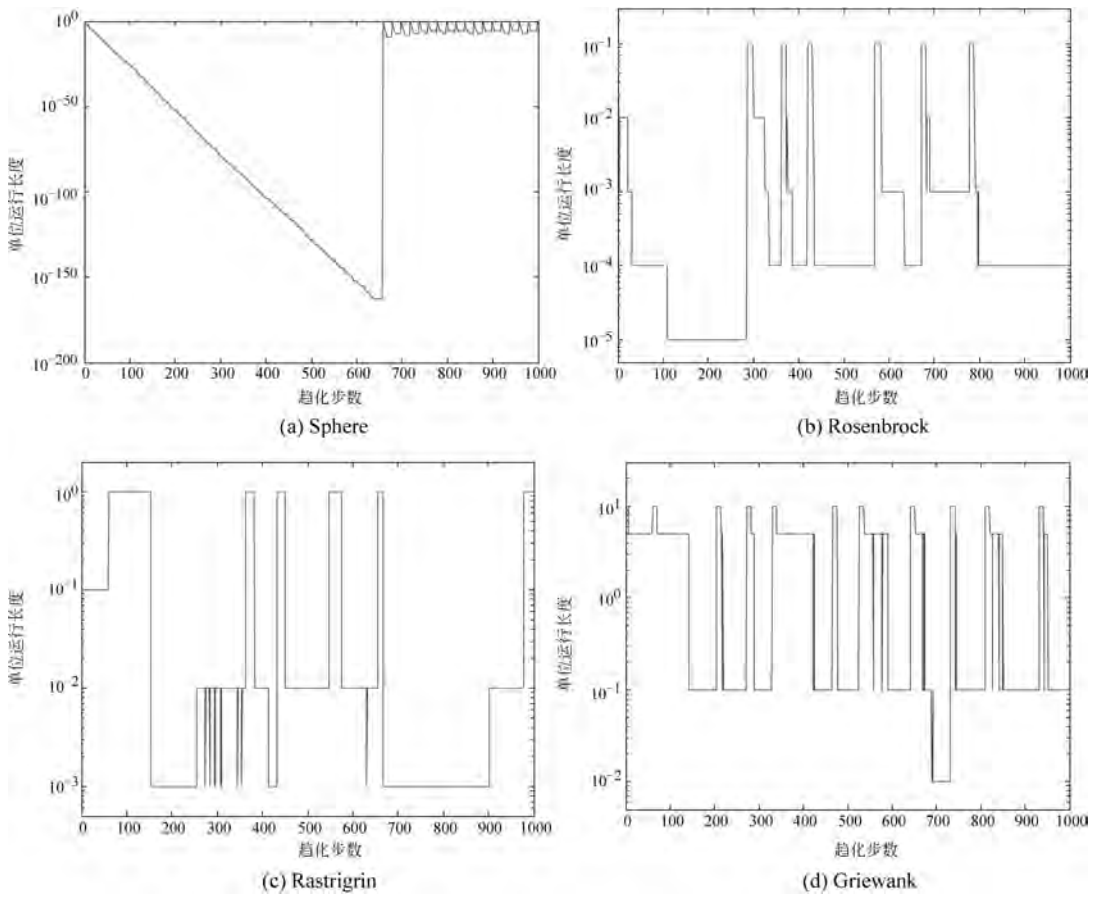


图 3-32　E. Coli 个体趋化步长的动态特性

3.5　植物根系自适应生长优化算法

3.5.1　算法的基本思想

植物的生长是由一颗种子开始，而根系的生长是植物生长必不可少的一部分。植物的整个根系可看作由大量根须、节组成的系统。20 世纪 60 年代末，荷兰数学家 Lindenmayer A. 把乔姆斯基的生成转换语法引入生物学，以简单的重写规则和分枝规则为基础，建立了关于植物的描述、分析和发育模拟的形式语法，称为 L-系统。该方法成功地对植物生长作了形式化的描述。L-系统的核心思想可以概括为以下几点：

（1）一粒种子在土壤里发芽，一部分破土而出成为茎，另一部分深埋地下，向下发展成为根须，在一些叫作节（本书也称为生长点）的部位长出新根须。

（2）大多数新根须上又长出更新的根须，这种分根行为反复进行。

（3）不同的根彼此有相似性，整个植物的根系有自相似结构，每个植物根系最终由大量相似的根须和节组成[37]。

L-系统构造的人工植物结构解决了模拟植物生长的演绎问题，但是其关键问题还没有解决，即在众多生长点中，每一次到底确定其中哪一个生长点进行新的生长，如何保证植物

根系在土壤肥沃程度不同的影响下向最优解方向生长，其关键核心问题是植物根系生长策略在优化算法中的实现问题。

植物根系生长涉及生物学理论中的形态发生模型。该模型是一个著名生物学实例，它利用复杂动力系统的理论为生物生长建模。此种模式的形成可以理解为一个关于植物根系生长的复杂过程，是由其中一个细胞发生分化，产生出新的明确定义的空间结构。植物根茎（根茎生长，从顶部开始，即种子）每生长出来一个根须，都包含着未分化的细胞。一个细胞被看作是一个流体袋，其中有均匀的化学组分，其中的一种化学组分是生长激素，叫作形态素。这种形态素的浓度名是此模型的观察参量，随着参量在 0 和 1 之间变动，模型的状态空间是一条线段。这种形态素的浓度决定细胞的生长函数是否开始起作用，即细胞分裂，根须开始出现。在细胞分裂的某一时刻，根须将在对于 3 或 4 个均与分布的对称旋转方向出现，进而生长（植物自适应算法中将旋转方向随机生成）[38]。

关于植物的生长过程（图 3-33），生物学上已有以下结论：

（1）当植物根系有一个以上的节时，具体哪个节能生长出新的根须取决于其形态素浓度，形态素浓度大的节获得长出新根须的机会较浓度小的节为大。

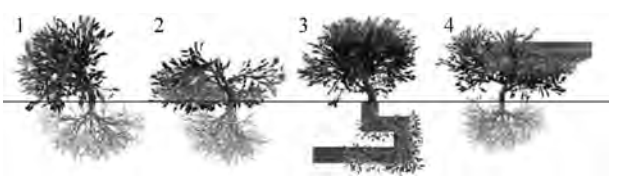


图 3-33　真实植物生长过程图

（2）决定根须生长的形态素浓度并非是预先一个个赋予细胞的，而是细胞系统从其环境中（土壤肥沃程度）接收到了它的位置信息，依据这种信息确定其形态素浓度；当后续新的根须产生后，形态素浓度将根据新系统所在环境的改变，重新在各生长点之间进行分配。

植物根系生长自适应算法（Root Growth Algorithm，RGA）[39]就是将优化问题的解空间当作植物生长土壤环境，将最优解当作土壤中最肥沃的那一个点，模拟真实植物根系的生长状况（形态素浓度理论），建立节点和根须在不同肥沃程度土壤环境下的快速生长方式（L-系统）。植物自适应算法的研究重点是建立以根节点生长规则为基础的植物系统演绎方式和以植物根系生长趋向模型，两者结合所形成的优化模式，就是实现人工植物在优化问题解空间中从初始状态到完整形式的终态（没有新的根须生长）的过程。寻优过程如图 3-34 所示。

图 3-34　寻优过程示意图

3.5.2　算法的形式化描述

植物根系自适应生长算法只有个体层，一粒种子发展成整个根系，各个生长点没有信息共享和交流，完全依赖由土壤环境影响的形态素浓度进行生长。

1. 个体层

植物自适应生长算法的个体层可表示为如下集合：

$$\text{Individual}=\{\text{Individual}_1,\text{Individual}_2,\cdots,\text{Individual}_m\}$$

第 i 个生长点表示如下：

$$\text{Individual}_i^t=\{\text{Position}_i^t,\text{Consistency}_i^t,\text{Enrichment}_i^t,\text{State}_i^t,\text{Direction}_i^t,\text{Length}_i^t\}$$

集合中的各元素分别表示生长点 i 在 t 时刻的位置、形态素浓度、土壤环境、生长状态、根须生长方向和根须生长长度。

2. 环境

植物自适应生长的生活环境可以具体定义为

$$\text{Env}\langle\sigma,\beta\rangle$$

其中，σ 代表环境中土壤中养分的分布；β 代表环境的边界特征。

在多细胞系统中，如果把任意一个细胞形态素浓度记为 $E_i(i=1,2,\cdots,n)$，则多细胞封闭系统形态素状态空间的浓度和是恒定的(设定为 1)。生物学实验已经证明，决定植物细胞分裂和根须生长的生长素信息(形态素浓度)并非是预先一个个赋予细胞的，而是细胞系统从其环境中(土壤肥沃程度)接收到了它的位置信息，依据这种信息，植物根系表现出明显的趋营养性特点。模拟这一过程，当有 n 个生长点 $S_i=(S_1,S_2,\cdots,S_n)$ 时，设每一个生长点的形态素浓度为 $E_i=(E_1,E_2,\cdots,E_n)$，当目标函数实现最小化时，计算各生长点形态素浓度值为

$$E_i=\frac{1/f(S_i)}{\sum_{i=1}^{n}1/f(S_i)} \tag{3-4}$$

其中以 $f(^*)$ 为目标函数值。式(3-4)中，各生长点形态素浓度是由各点的相对位置以及该位置的环境信息(目标函数值)所确定，这与真实植物细胞的形态素浓度生成机理相一致。因此，n 个生长点均对应 n 个形态素浓度值，每次产生新的根须，该浓度值都将发生变化。

每次生长点分出新的根须，根据下面的公式进行计算：

$$k=\text{rand}() \tag{3-5}$$

$$X_S=\text{pg}_{ik}+(2\times\text{rand}-1) \tag{3-6}$$

其中，pg_{ik} 代表需要分裂的生长点；k 代表一个随机维度。

每条根须的生长角度随机生产，根据下式进行计算：

$$(\phi_1,\phi_2,\cdots,\phi_n)=\text{rand}(n) \tag{3-7}$$

$$\varphi_i^{t+1}=\frac{(\phi_1,\phi_2,\cdots,\phi_n)}{\sqrt{\phi_1^2+\phi_2^2+\cdots+\phi_n^2}} \tag{3-8}$$

当每条根须在生长时，根据下式进行计算：

$$\partial_i^{t+1}=\partial_i^{t+1}+\delta_i^t\varphi_i^{t+1} \tag{3-9}$$

3.5.3 算法流程

植物根系自适应生长算法的过程如下：

Step 1，确定初始种子位置、初始整体根须生长长度、生长代数和最大生长点的个数。

Step 2，计算基点适应度值(形态素浓度)。

Step 3，计算每个生长点适应度值。

Step 4，选择适应度值高的生长点，然后在这些生长点进行分根，生成新的根须。

Step 5，计算每个生长点适应度值，每个根须按照一定角度（随机）生长一定长度。

Step 6，计算生长后的适应度值。如果适应度更好，则继续生长，生长达到一定次数停止生长；如果适应度没有改进或退步，则停止生长。

Step 7，根据该适应度的改进情况调整每条根须的生长长度。

Step 8，如果生长点个数达到最大生长点个数，部分适应度差的生长点不再允许新的根须生长。

Step 9，每隔一定代数调整整体根须的生长长度。

Step 10，最优生长点连续保持不变的最大持续迭代步数达到事先规定迭代步数，程序终止，否则返回 Step 3。

植物根系自适应生长算法的流程图如图 3-35 所示。

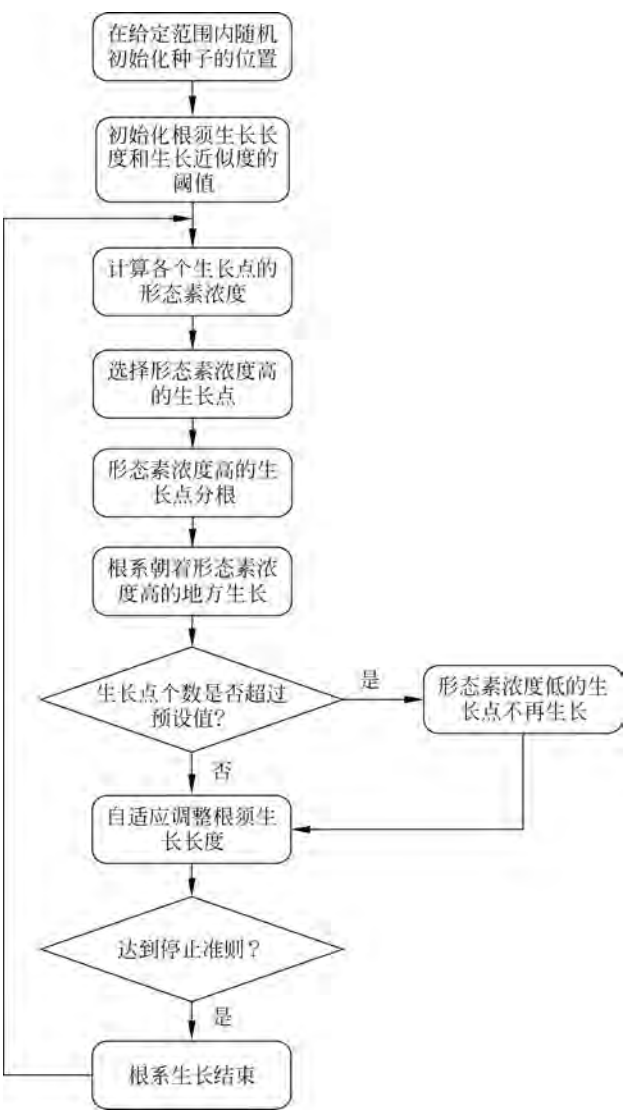


图 3-35　植物根系自适应生长算法的流程图

3.5.4 算法效能分析

1. 测试函数

为了验证算法 RGA 的有效性，本章采用 Sphere、SumSquares、Rosenbrock、Schwefel 2.22、Rastrigrin、Schwefel、Ackley 和 Griewank 这 8 个标准测试函数对其性能进行测试。这 8 个函数的求解针对的都是最小化问题。其中，前 4 个函数是单峰函数，后 4 个函数是多峰函数。这 8 个测试函数分别选自文献[40-41]。它们的优化求解问题都是 GA-难度大的问题，用它们来验证和比较算法的性能是比较合适的。

2. 实验结果与分析

测试实验分别在标准测试函数的 2 维、15 维和 30 维上进行，并与 GA 算法、DE 算法和 PSO 算法进行性能比较。RGA 可以采用两种自适应策略：第一种自适应策略对应的 RGA 算法称作 RGA-α；第二种自适应策略对应 RGA 称作 RGA-τ。如不做特殊说明，RGA 指的就是 RGA-τ。

1）2 维函数优化算法效果

表 3-12 展示植物根系生长算法(RGA)优化 2 维函数得到的数值结果。可见，RGA 在优化 2 维函数上能取得一定的优化效果，但是整体上相对于其他算法(GA、DE 和 PSO)并没有优势，个别函数优化效果相对较差。

表 3-12 2 维函数优化比较

函数		GA	DE	PSO	RGA-τ
Sphere	平均值	9.254 75e−010	0	2.692 2e−011	3.3514e−006
	标准差	7.158 14e−011	0	3.576 39e−011	4.617 62e−006
	最小值	2.046 02e−011	0	1.939e−012	4.748 93e−007
	最大值	2.157 52e−010	0	7.976 44e−011	8.677 68e−006
SumSquares	平均值	1.013 19e−004	0	0	6.622 69e−007
	标准差	6.887 83e−005	0	0	6.550 02e−007
	最小值	2.745 55e−005	0	0	2.223 27e−007
	最大值	1.637 91e−005	0	0	1.415 03e−006
Rosenbrock	平均值	0.7648	0	0	0.694 768
	标准差	0.8652	0	0	1.061 22
	最小值	0.0968	0	0	0.008 752 21
	最大值	0.8465	0	0	1.917 11
Schwefel 2.22	平均值	0.009 035 9	0	0	0.148 505
	标准差	0.003 123 5	0	0	0.254 426
	最小值	0.005 678 56	0	0	0.001 549 82
	最大值	0.011 855 8	0	0	0.442 291
Rastrigrin	平均值	0.0019	0	0	5.043 19e−005
	标准差	0.0016	0	0	5.273 55e−005
	最小值	1.1255e−004	0	0	1.686 81e−005
	最大值	0.0031	0	0	0.000 111 215
Schwefel	平均值	−421.343	−837.966	−837.966	−429.738
	标准差	80.8736	0	0	69.1907
	最小值	−602.32	−837.966	−837.966	−501.886
	最大值	−209.365	−837.966	−837.966	−363.943

续表

函　数		GA	DE	PSO	RGA-τ
Ackley	平均值	0.1014	7.287 62e−016	8.881 78e−016	0.009 833 91
	标准差	0.0591	9.838 27e−015	3.384 76e−016	0.009 893 39
	最小值	0.0637	6.536 72e−016	9.233 45e−016	0.002 571 14
	最大值	0.1695	9.827 32e−016	7.832 46e−015	0.021 101 9
Griewank	平均值	0.0222	0.032 869 1	0	0.2019
	标准差	0.0135	0.048 638 6	0	0.2872
	最小值	0.0101	0	0	0.1134
	最大值	0.0367	0.0887426	0	0.3835

在表 3-12 中可以看到，DE 和 PSO 算法优化效果相当好，在所有函数上，RGA-τ 优化效果都无法与之相比。GA 优化效果相对较差，在 SumSquares、Rosenbrock、Rastrigrin 和 Ackley 四个测试函数上，RGA-τ 都明显好于 GA。SumSquares、Rosenbrock、Rastrigrin 和 Ackley 四个测试函数分别属于单峰变量分离函数、单峰变量非分离函数、多峰变量分离函数和多峰变量非分离函数，所以 RGA-τ 在 2 维函数的优化上能够取得一定的优化效果。

2）15 维函数优化效果

表 3-13 展示植物根系生长算法(RGA)优化 15 维函数得到的数值结果。可见，RGA 在 15 维函数上能取得不错的优化效果，在 Rastrigrin 函数上相对于其他算法(GA、DE 和 PSO)取得了更好的结果，在其他函数上也能取得不逊于其他算法的结果。

表 3-13　15 维函数优化比较

函　数		GA	DE	PSO	RGA-τ
Sphere	平均值	3.230 18e−008	3.0996e−008	4.260 22e−017	8.484 53e−016
	标准差	2.369 85e−008	4.751 43e−008	5.845 34e−017	1.248 57e−015
	最小值	1.256 82e−009	2.759 36e−009	5.828 44e−018	9.459 17e−024
	最大值	9.256 87e−008	8.585 29e−008	1.100 05e−016	2.282 14e−015
SumSquares	平均值	0.467 211	16.3759	4.059 35e−007	5.272 33e−006
	标准差	0.239 559	23.3584	6.930 71e−007	3.011 37e−006
	最小值	0.238 748	2.51918	4.301 54e−009	1.861 33e−008
	最大值	0.716 507	43.3444	1.206 22e−006	7.562 34e−006
Rosenbrock	平均值	442.9023	6.854 01	1.328 87	1.5895
	标准差	203.8842	5.022 71	2.301 68	0.664 571
	最小值	258.5536	1.286 61	0.318 131	0.877 842
	最大值	661.8853	11.045	3.986 62	2.193 95
Schwefel 2.22	平均值	0.285 536	1.580 62	3.051 71e−003	0.012 476 7
	标准差	0.077 656 7	0.879 049	1.607 88e−003	2.766 15e−003
	最小值	0.206 410	0.6959	1.209 64e−003	9.336 33e−003
	最大值	0.361 635	2.453 89	4.173 72e−003	0.014 552
Rastrigrin	平均值	10.8695	55.0541	7.959 67	4.190 82
	标准差	2.5754	19.632	2.632 41	6.990 52
	最小值	8.3097	41.7881	5.969 75	0.000 137 63
	最大值	13.4601	77.6065	10.9445	8.5722

续表

函　数		GA	DE	PSO	RGA-τ
Schwefel	平均值	−6521.37	−5333.33	−3388.11	−3757.9
	标准差	556.965	114.925	755.62	171.047
	最小值	−7034.70	−5465.09	−3955.41	−4230.33
	最大值	−5929.21	−5253.75	−2769.62	−3659.12
Ackley	平均值	2.6047	9.049 68	9.703 59e−010	0.049 895 7
	标准差	0.2706	0.638 332	6.777 66e−010	0.085 165 3
	最小值	2.4314	8.338 11	5.170 95e−010	9.794 14e−005
	最大值	2.9165	9.571 95	1.749 51e−009	0.148 233
Griewank	平均值	1.2788	8.551 58	0.048 296 9	1.636 82
	标准差	0.0942	5.614 15	0.030 698 7	0.839 965
	最小值	1.2052	2.613 55	0.027 037	0.879 819
	最大值	1.3849	13.773	0.083 491 5	2.540 44

从表 3-13 可以看出，GA 和 DE 算法优化的效果与 PSO 和 RGA-τ 算法相比较结果较差，只是在 Schwefel 函数上取得了较好的优化效果。RGA-τ 算法整体的优化效果仍然稍差于经典 PSO 算法。与 PSO 算法相比，RGA-τ 在 Sphere、SumSquares、Rosenbrock、Schwefel 2.22 和 Schwefel 函数上优化效果的差别在一个数量级以内，在 Ackley 和 Griewank 函数上优化效果的差别在一个数量级以上，而在 Rastrigrin 函数上优化效果要好于 PSO 算法。Ackley 和 Griewank 函数属于多峰变量非分离函数，在 15 维上，RGA-τ 在这类函数上的优化效果并不是很明显。

3）30 维函数优化效果

为了进一步证明 RGA 算法的有效性，对 8 个测试函数在 30 维上进行了更为详细的测试。表 3-14 给出了 8 个测试函数在 30 维上的参数信息，包括维度、初始范围和最小值。

表 3-14　RGA 优化 30 维测试函数的参数

函　数	维　度	初始范围	最　小　值
Sphere	30	[−100,100]D	0
SumSquares	30	[−10,10]D	0
Rosenbrock	30	[−30,30]D	0
Schwefel 2.22	30	[−10,10] D	0
Rastrigrin	30	[−10,10] D	0
Schwefel	30	[−500,500]D	−12 569.5
Ackley	30	[−32.768,32.768] D	0
Griewank	30	[−600,600]D	0

表 3-15 列出了 GA、DE、PSO、RGA-α 和 RGA-τ 的比较结果，记录了由这些算法计算所得的平均值、标准差、最小值和最大值。图 3-36～图 3-43 分别给出了这些算法经 30 次运行最优平均值的函数优化过程收敛曲线。

表 3-15　30 维函数优化比较

函　数		GA	DE	PSO	RGA-α	RGA-τ
Sphere	平均值	0.925 05	22.6724	2.254 09e−008	0.212 980	1.098 52e−008
	标准差	1.298 11	28.6836	1.760 46e−008	0.546 753	1.615 96e−008
	最小值	0.502 20	3.694 61	1.063 66e−008	6.235 95e−003	5.905 61e−024
	最大值	3.199 16	55.6673	4.276 32e−008	1.040 37	2.954 10e−008
SumSquares	平均值	0.042 340 8	366.243	1.159 88e−003	0.013 396 2	4.073 59e−003
	标准差	6.917 78e−003	144.776	6.236 41e−004	6.564 23e−003	5.531 86e−004
	最小值	0.037 295 5	249.874	9.679 14e−004	9.376 78e−003	3.497 10e−003
	最大值	0.050 226 6	528.369	3.289 71e−003	0.016 769 5	4.600 07e−003
Rosenbrock	平均值	4300.21	11 859.6	36.0147	1440.22	21.2087
	标准差	638.258	5221.57	37.1823	1837.75	0.808 404
	最小值	3662.62	7170.21	12.1717	150.406	20.3453
	最大值	4939.16	17 486.4	78.8579	3241.94	21.9481
Schwefel 2.22	平均值	0.194 223	21.3174	0.095 886 5	0.122 179	0.054 966 2
	标准差	0.032 346	2.488 56	0.070 950 8	0.020 989 2	1.998 53e−003
	最小值	0.162 737	19.4056	0.071 494 8	0.097 943 5	0.046 028 7
	最大值	0.227 365	24.1311	0.176 012	0.134 432	0.067 553 4
Rastrigrin	平均值	87.1626	156.074	28.5221	17.2605	4.285 74e−004
	标准差	12.4927	70.4707	9.028 02	6.672 54	9.690 91e−005
	最小值	79.1588	96.8458	21.8891	12.8178	3.358 07e−004
	最大值	101.558	234.011	38.8034	24.9334	5.291 53e−004
Schwefel	平均值	−8793.98	−8746.91	−5251.66	−7808.34	−8487.84
	标准差	587.525	1328.42	1176.508	1060.61	1041.19
	最小值	−9285.05	−10 175.6	−6469.01	−8558.30	−9666.01
	最大值	−8143.08	−7548.97	−4935.01	−7058.38	−7691.38
Ackley	平均值	4.903 92	12.8558	6.440 18e−007	3.034 18	3.276 35e−004
	标准差	0.428 56	1.061 69	4.684 84e−007	0.655 751	1.287 37e−005
	最小值	4.579 18	11.6323	2.428 85e−007	2.545 42	1.570 42e−004
	最大值	5.389 55	13.5347	1.1589e−006	3.779 40	4.345 91e−004
Griewank	平均值	1.814 73	1.388 65	0.033 825 2	0.659 995	3.700 74e−003
	标准差	0.206 79	0.290 33	0.048 303 8	0.445 041	5.226 26e−003
	最小值	1.583 55	1.053 81	4.983 59e−006	0.336 426	3.537 63e−007
	最大值	1.982 06	1.570 37	0.089 146 3	1.171 55	4.171 91e−003

从表 3-15 中的数值可以看出，RGA-α 和 RGA-τ 在所有 30 维函数上都能取得较好的优化效果。RGA-τ 在 Sphere、Rosenbrock、Schwefel 2.22、Rastrigrin 和 Griewank 等函数上能取得明显的优化效果；PSO 在 SumSquares 和 Ackley 两个函数上具有更好的性能；GA 和 DE 在函数 Schwefel 能得到较好的结果。可见，RGA 在高维函数优化上具有明显的优势。

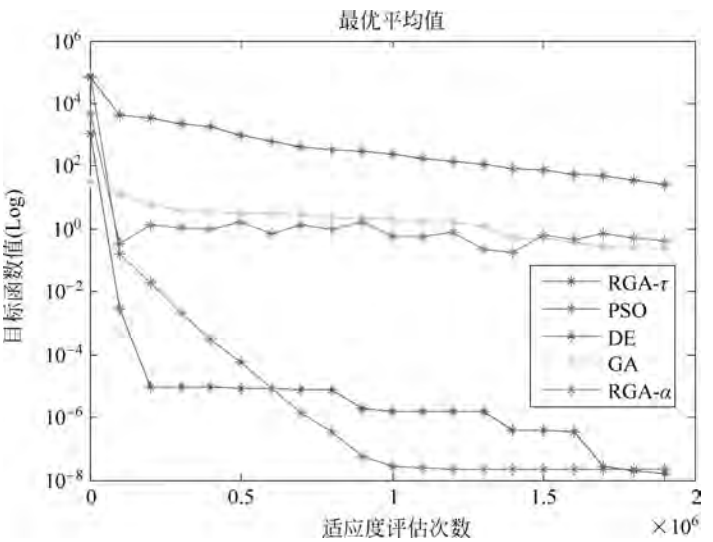


图 3-36　30 维的 Sphere 函数图像

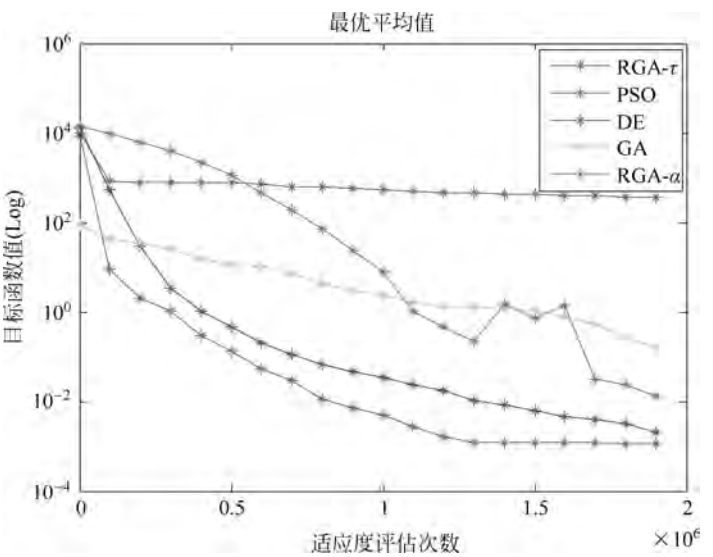


图 3-37　30 维的 SumSquares 函数图像

Sphere 和 SumSquares 是两个单峰变量分离函数(见图 3-36 和图 3-37),PSO 和 RGA-τ 优化这两个函数都能取得不错的效果。RGA-τ 在 Sphere 函数上取得了最好的优化结果,所有算法按其优化性能排序为 RGA-τ > PSO > RGA-α > GA > DE; PSO 在 SumSquares 函数上取得了最好的优化结果,所有算法按其优化性能排序为 PSO > RGA-τ > RGA-α > GA > DE。

Rosenbrock 和 Schwefel 2.22 是两个单峰非变量分离函数(见图 3-38 和图 3-39),在这两个函数上 RGA-τ 取得了最好的优化结果。在 Rosenbrock 函数上,PSO 比 RGA-τ 稍差,而 GA 和 DE 的优化效果较差; 在 Schwefel 2.22 函数上,PSO、RGA-α 和 GA 都能取得较好的优化效果。其统计结果表明,针对单峰非变量分离函数,所有算法按其优化性能排序为

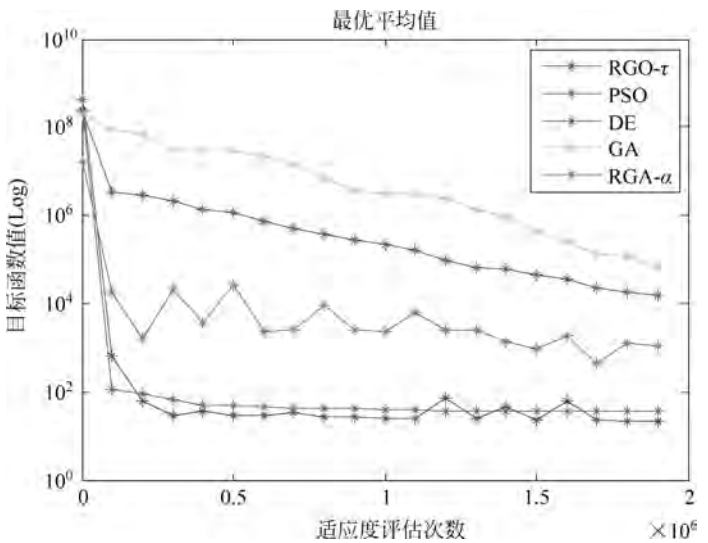


图 3-38　30 维的 Rosenbrock 函数图像

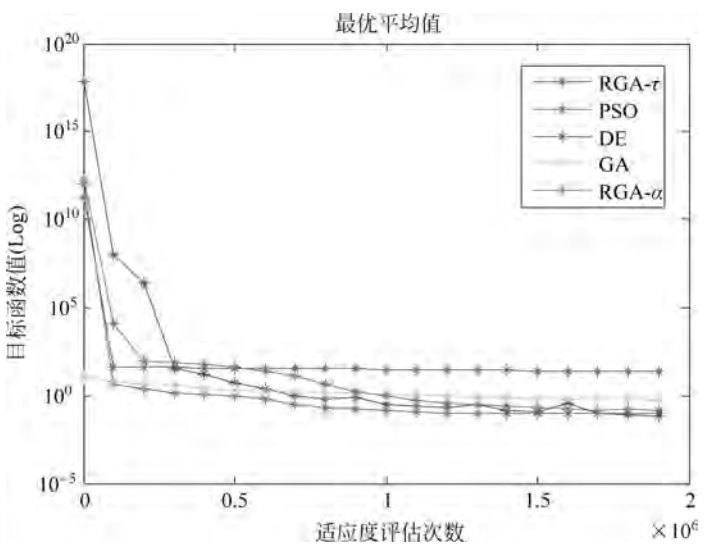


图 3-39　30 维的 Schwefel 2.22 函数图像

RGA-τ > PSO > RGA-α > GA > DE。

Rastrigrin 和 Schwefel 是两个多峰变量分离函数。从图 3-40 和图 3-41 可以看出，RGA-τ 的收敛性能明显好于其他 4 个算法，在 Rastrigrin 函数上，PSO、DE、GA 和 RGA-α 很快陷入了局部最优解，所有算法按其优化性能排序为 RGA-τ > RGA-α > PSO > GA > DE；在 Schwefel 函数上，GA 和 DE 表现出了比较好的性能，所有算法按其优化性能排序为 GA > DE > RGA-τ > RGA-α > PSO。

Ackley 和 Griewank 是两个多峰非变量分离函数(见图 3-42 和图 3-43)，在这两个函数上，RGA-τ 和 PSO 取得的优化效果明显优于 GA 和 DE。在 Ackley 函数上，PSO 比 RGA-τ 具有更好的收敛性能，所有算法按其优化性能排序为 PSO > RGA-τ > RGA-α > GA > DE；

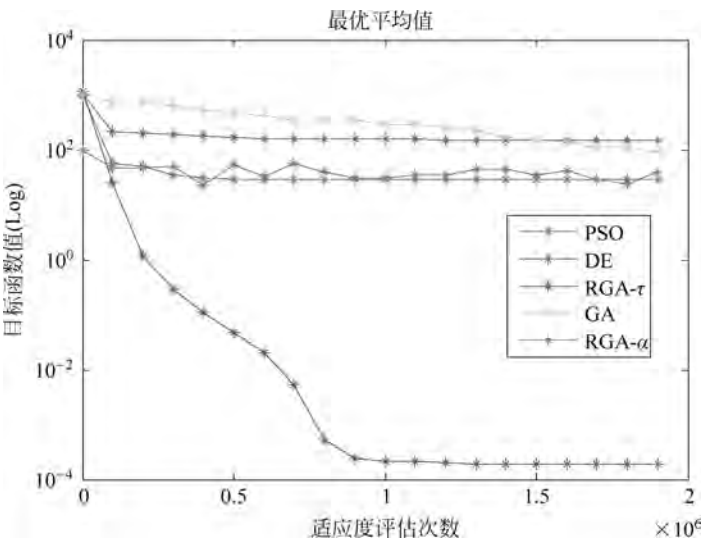


图 3-40　30 维的 Rastrigrin 函数图像

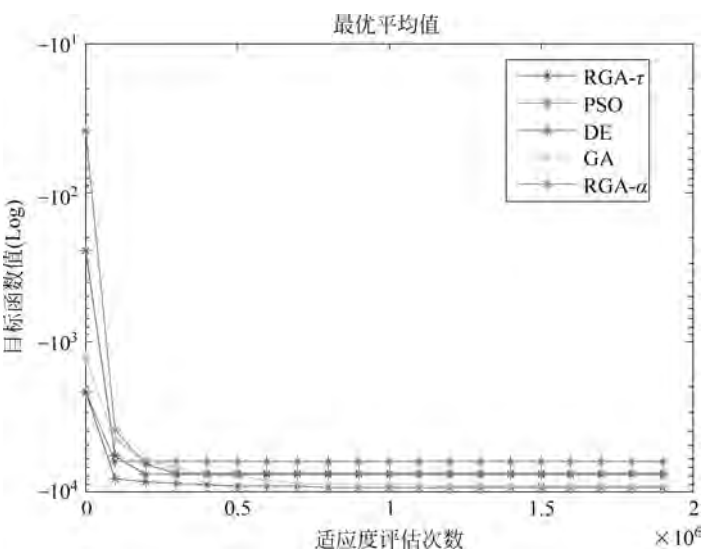


图 3-41　30 维的 Schwefel 函数图像

但是在 Griewank 函数上，RGA-τ 收敛得更好，而 PSO 开始快速收敛，致使其过快地陷入局部最优解，所有算法按其优化性能排序为 RGA-τ > PSO > RGA-α > DE > GA。

综上所述，采用两种策略的 RGA 算法在大多数高维度函数上都取得了相对较好的优化效果，特别是在单峰非变量分离函数和多峰非变量分离函数上。RGA-τ 展现出求解复杂优化问题的潜力，由此可见，第二种自适应策略与 RGA 算法结合得更好。

4）参数 τ 的调节

当使用智能算法优化一个实际问题时，算法关键参数的调节对算法的性能是至关重要的。对大多数算法来说，要获得理想的结果，对控制参数的适当调节是必要的。

RGA 算法使用了多个控制参数，这里只对关键参数 τ 对 RGA 性能的影响进行分析。

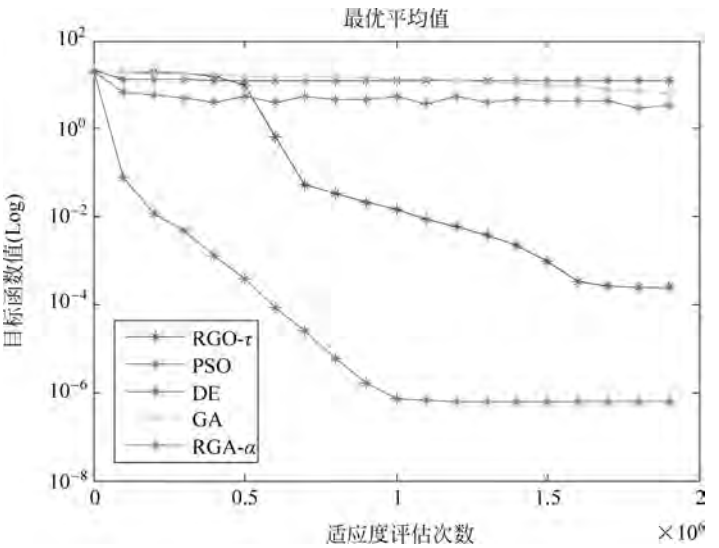


图 3-42　30 维的 Ackley 函数图像

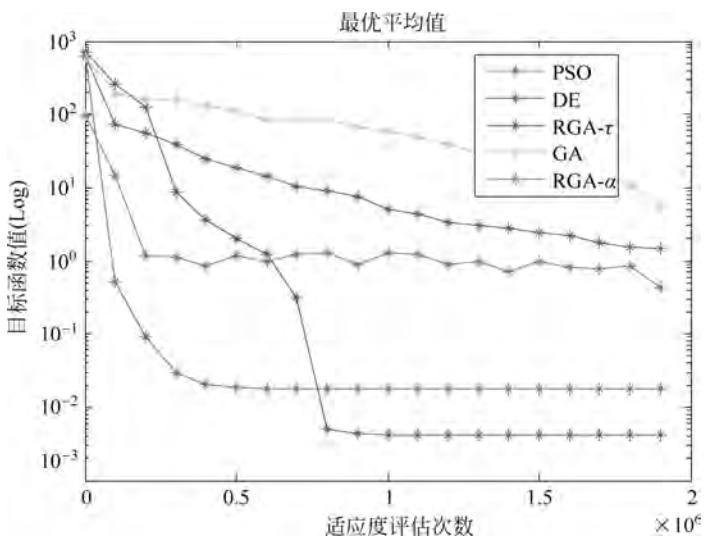


图 3-43　30 维的 Griewank 函数图像

因为参数 τ 是第二种自适应策略中一个非常有特色的参数，它是对算法结构至关重要的参数，在实验中可以发现该参数对算法结果影响较为明显。

在这次实验中，参数 τ 的值在 1、50、100、500 和 1000 这 5 个不同值中变化。实验结果在表 3-16 中列出。实验结果表明，当 RGA 算法的其他参数相同时，参数 τ 值的变化会使 RGA 在相同函数上的优化效果明显不同，主要原因在于参数 τ 决定了根须生长长度，而根须生长长度又是 RGA 在新区域探索好坏的重要因素。但是在 Schwefel 函数上，参数 τ 的变化对优化效果影响不大。图 3-44～图 3-51 是不同 τ 的值下，关于最优平均值的函数优化过程收敛曲线。这些图像也印证了表 3-16 的结果。可见，参数 τ 对 RGA 的优化有着重要影响。

表 3-16　不同 τ 值对 RGA-τ 取得的优化结果

函　数		τ				
		1	50	100	500	1000
Sphere	平均值	6.037 50	1.396 62e−005	0.010 950 5	1.592 17e−008	1.486 42e−005
	标准差	0.405 303	1.929 88e−007	0.015 483 0	7.791 35e−009	1.922 48e−006
SumSquares	平均值	150.623	0.040 513 1	9.845 51e−003	0.041 714 6	3.617 94e−003
	标准差	44.2301	5.940 27e−003	2.014 07e−003	0.058 128 3	4.581 69e−004
Rosenbrock	平均值	2504.52	1841.08	3117.08	60.098	18.6324
	标准差	2018.48	184.827	1758.65	8.362 71	0.970 460
Schwefel 2.22	平均值	14.9426	0.175 768	0.062 873 2	0.221 833	0.218 181
	标准差	2.134 43	0.034 957 9	1.874 90e−003	4.112 23e−3	0.025 114 9
Rastrigrin	平均值	361.247	354.175	355.371	0.018 598 2	5.053 23e−003
	标准差	42.1935	11.6724	6.095 98	0.439 82e−003	7.139 57e−003
Schwefel	平均值	−8372.39	−8678.21	−8145.58	−7977.11	−8037.08
	标准差	1267.7	712.181	2398.20	1498.3069	1872.12
Ackley	平均值	3.600 46	3.713 68e−003	9.393 40	3.687 86e−004	2.808 95e−003
	标准差	0.321 164	5.599 02e−004	12.7740	2.312 68e−006	3.6843e−004
Griewank	平均值	9.148 04e−003	3.700 74e−003	5.894 45e−003	3.631 94	24.9365
	标准差	1.915 72e−003	5.226 26e−003	8.296 61e−003	1.529 99	9.279 19

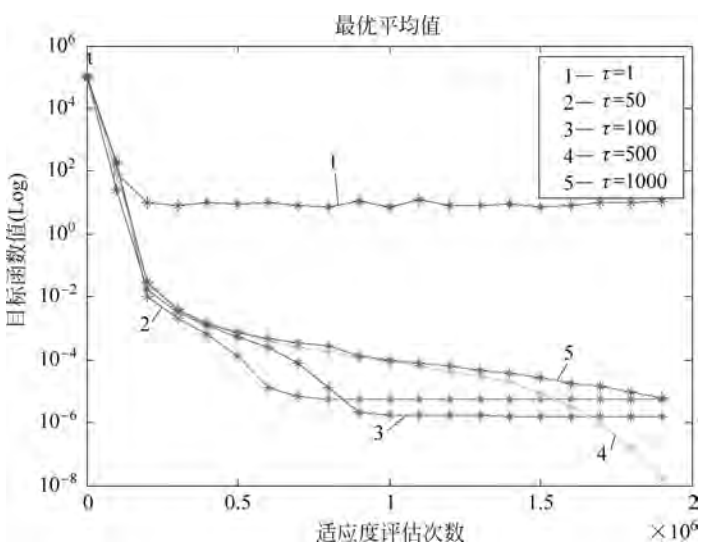


图 3-44　在 30 维的 Sphere 函数参数 τ 的影响

3. 动态优化测试问题及性能分析

现实世界很多复杂的优化问题都是动态的，例如在生产调度过程中，新的任务不断到达，需要立即加入到当前调度队列中；机器可能会发生故障或降低加工速度；原材料的质量会发生变化等。所谓动态优化问题(DOP)，是指适应度值函数、问题实例或者限制条件实时发生变化的优化问题[42]。

与静态优化不同，动态环境下的优化问题的最优解会随时间而变化，这就要求优化

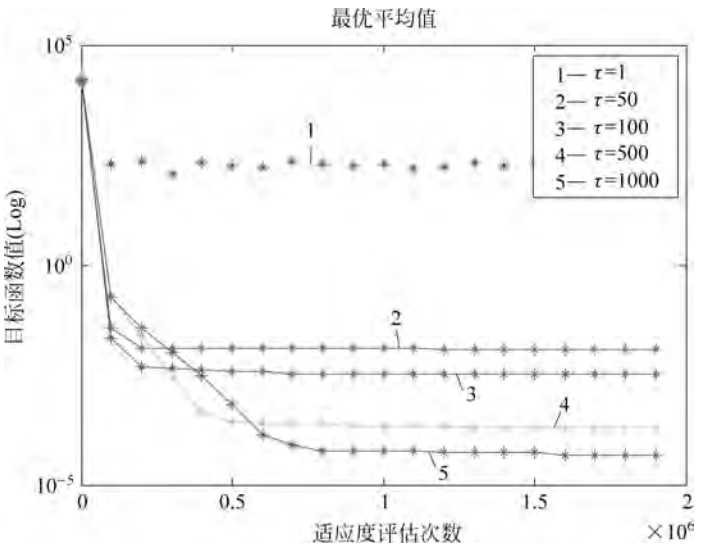


图 3-45　在 30 维的 SumSquares 函数参数 τ 的影响

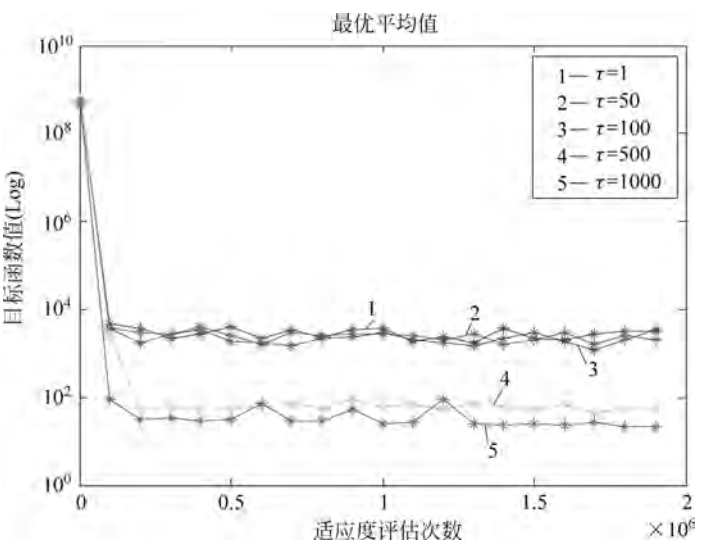


图 3-46　在 30 维的 Rosenbrock 函数参数 τ 的影响

算法必须同时具有良好的收敛性能和多样性保持能力，以便能够及时检测并快速追踪变化中的最优解。然而，收敛性与多样性的平衡一直以来都是基于生物行为的启发式优化算法的难点问题。因此，研究动态环境下的优化问题及其求解方法也具有很高的研究价值。

本节将通过实验方法来研究 RGA 算法在动态环境中的优化特性及算法参数对算法性能的影响，并与经典的细菌算法 BSFO(Bacterial Swarm Foraging For Optimization)进行比较。本章实验所用到的测试平台是 MPB(Moving Peaks Benchmarks)。

1) 动态环境设置

MPB 是由 Branke[44] 提出的一个动态优化测试平台，它已成为当今动态优化领域的一

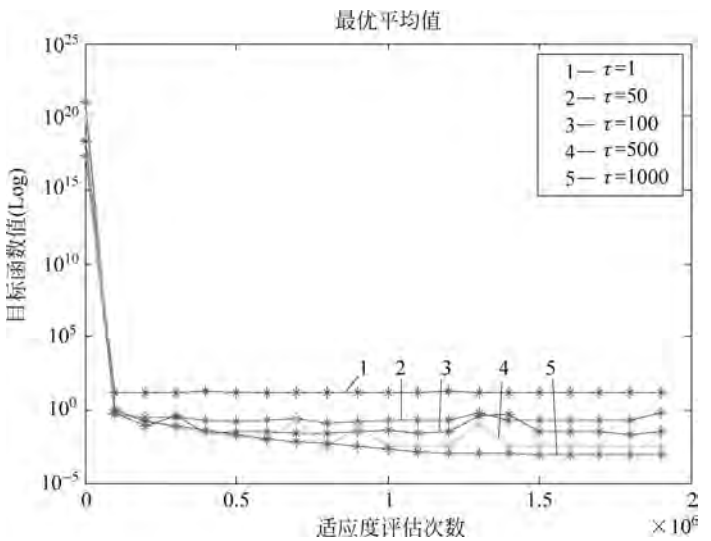


图 3-47　在 30 维的 Schwefel 2.22 函数参数 τ 的影响

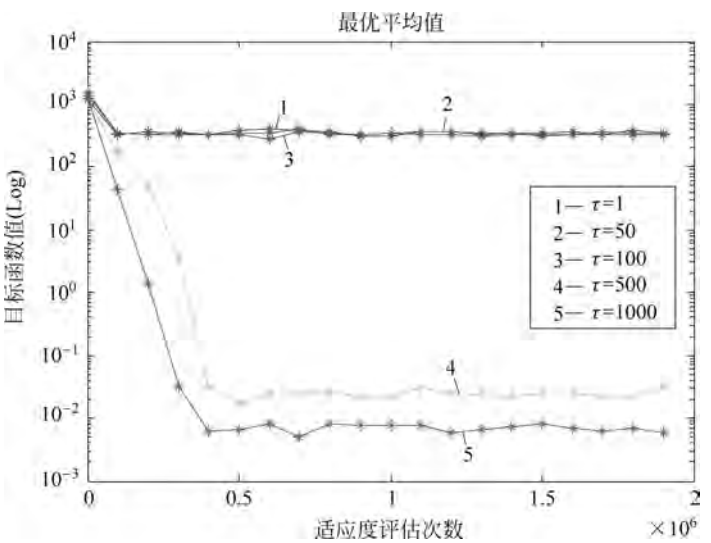


图 3-48　在 30 维的 Rastrigrin 函数参数 τ 的影响

个标准测试问题。MPB 定义了一个 n 维的连续动态函数空间，函数空间可以预先设定其峰的个数 N、每个峰的位置 X、高度 H 和坡度 R。当这些参数设置好后，软件就根据这些参数值在限定的区域内随机产生一系列的函数峰值。这些函数峰值可以在算法执行过程中(随时间变化)改变它们的位置、高度和宽度。MPB 函数空间的适应度用下式来评估：

$$y = -\max_{i=1,2,\cdots,N}\left\{H_i - R_i\sqrt{\sum_{j=1}^{D}(x_j - X_{ij})^2}\right\} \tag{3-10}$$

对于第 i 个峰来说，H_i 代表它的高度变量；R_i 代表它的坡度变量；X_{ij} 则代表该峰的坐标中心。参数的初始化值如表 3-17 所示。

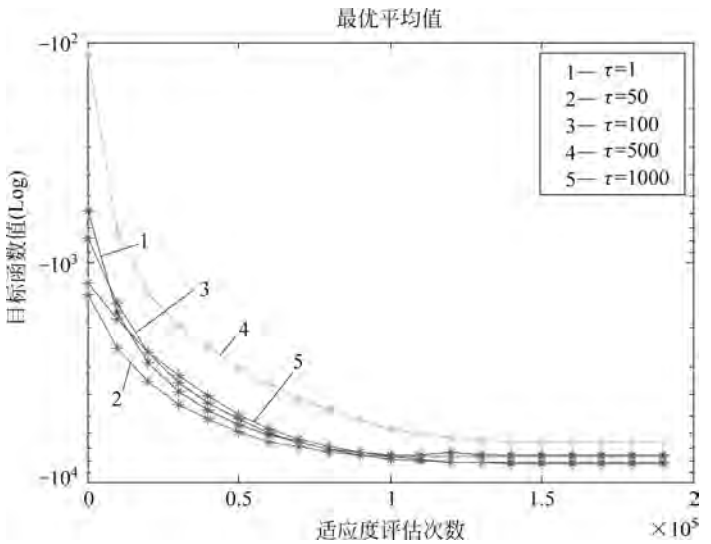


图 3-49　在 30 维的 Schwefel 函数参数 τ 的影响

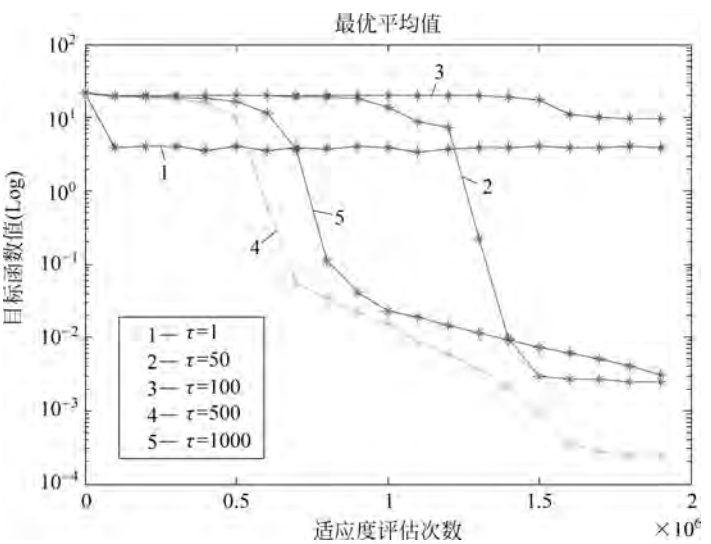


图 3-50　在 30 维的 Ackley 函数参数 τ 的影响

表 3-17　MPB 参数设置

参　数	设　置　值	参　数	设　置　值
N	15	H_i	[1,10]
H_1	0	R_i	[8,20]
R_1	0	X_{ij}	[−1,1]
X_{1j}	0.5	i	2,3,…,N

表 3-17 中的所有参数都可以调节用来动态改变环境，但是在本节的仿真过程中，设置每个峰的高度和坡度为恒定值，仅仅变化其坐标中心的位置以达到改变环境的目的。在这

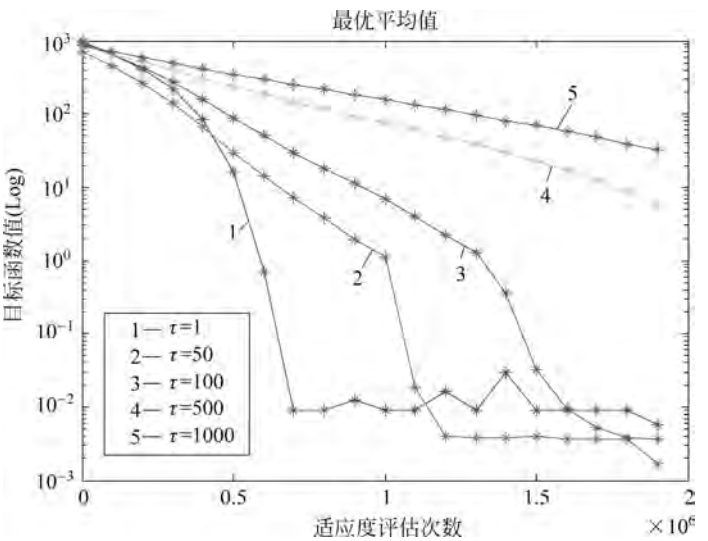


图 3-51　在 30 维的 Griewank 函数参数 τ 的影响

种情况下，用 $x_{\text{step},j}$ 代表了在 x_j 方向上移动的步长，对于每一阶段 t，第 i 个峰的 $X_{t+1,i}$ 可以用下式表示：

$$x_{\text{step},i}=\mu_i\times(1-x_{\text{step},i}) \tag{3-11}$$

$$X_{t+1,i}=X_{st,i}+x_{\text{step},j}\times\varphi \tag{3-12}$$

其中，μ_i 是常数；$\varphi=-1$ 或 $\varphi=1$，选择概率为 0.5。由上列各式产生的 4 个阶段的动态环境示例如图 3-52 所示。

2）RGA 在动态环境中的设置

这个实验设计的目的是为了估算 RGA 在动态环境中的优化性能，动态环境的变化可分为以下 3 种情况[45]：

情况 1，环境缓慢变化。

情况 2，环境适中变化。

情况 3，环境快速变化。

这里定义一个概率参数 μ 用来表征在每一代根系生长中环境改变的频率。对于情况一，$\mu\in[0,0.01]$；对于情况二，$\mu\in[0.05,0.2]$；对于情况三，$\mu\in[0.3,0.8]$。

3）评估标准

动态优化算法表现的评估标准可以有多种，例如离线表现(offline performance)、离线误差(offline error)、期间平均最好适应度(average best over a period)、算法的准确性(accuracy)和稳定性(stability)等。本实验所采用的评估标准是算法的准确性和稳定性。为了获得算法 RGA 在函数 MPB 上的准确性，首先计算每一阶段 t 的准确性：

$$\text{Accuracy}_{F,A}(t)=\frac{f_{F,A}(t)-V_{wF,A}(t)}{V_{oF,A}(t)-V_{wF,A}(t)} \tag{3-13}$$

然后，整体的准确性定义为

$$\text{Accuracy}_{F,A}=\frac{1}{N}\sum_{t=1}^{N}\text{Accuracy}_{F,A}(t) \tag{3-14}$$

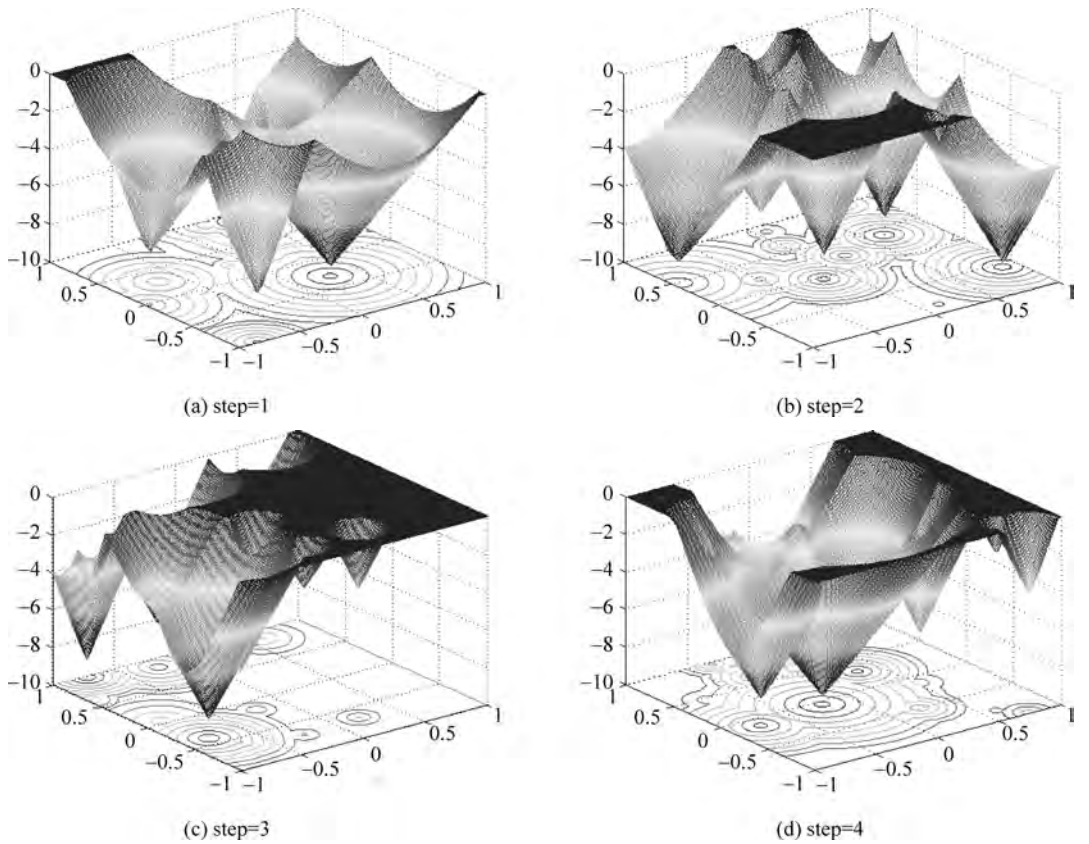


图 3-52　MPB 示例(T 表示时间)

与算法的准确性类似,算法 RGA 在函数 MPB 上的稳定性定义为

$$\text{Stability}_{F,A}(t)=\text{Accuracy}_{F,A}(t)-\text{Accuracy}_{F,A}(t-1) \tag{3-15}$$

$$\text{Stability}_{F,A}=\frac{1}{N}\sum_{t=1}^{N}\max\{0,\text{Stability}_{F,A}(t)\} \tag{3-16}$$

4) 仿真结果

对于算法 RGA 采用 3.2.2 节中的两种自适应策略进行仿真,分别为 RGA-α 和 RGA-τ,并与算法 BSFO 进行比较(见图 3-53)。为了证明算法在动态环境中的性能,环境改变的频率分别选定为 $\mu=0.001$、$\mu=0.005$、$\mu=0.01$、$\mu=0.05$ 和 $\mu=0.1$,它们分别属于动态环境的设置中的情况一和情况二。不同的 μ 值在 1000 次运行中分别动态地改变环境 2 次、7 次、12 次、46 次和 91 次。对比 RGA-α 和 BSFO,RGA-τ 能够更加稳定有效地跟踪时变的全局最优解。

表 3-18 展示了在 3 种动态环境下 BSFO、RGA-α 和 RGA-τ 准确性的比较结果。$\mu=[0.001,0.005,0.01]$代表动态环境变化的第一种情况;$\mu=[0.05,0.1,0.2]$代表动态环境变化的第二种情况;$\mu=[0.3,0.5,0.8]$代表动态环境变化的第三种情况。当 $\mu=0.005$ 时,RGA-τ 准确性最高,在其他情况下,无论 μ 增大或者减小,准确性都在降低。在第三种情况下,RGA 和 BSFO 都获得了较高的准确率。因为在这种情况下,环境改变的较频繁,多样性比起局部搜索更加重要。

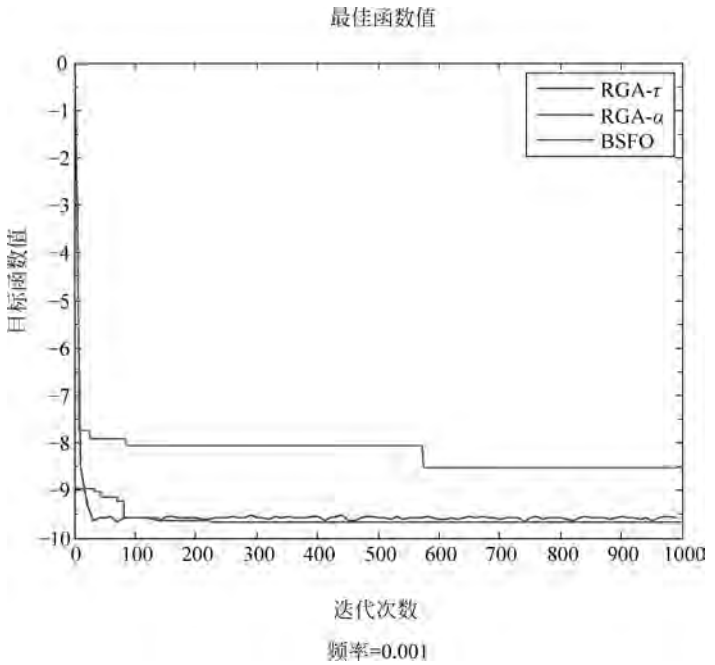
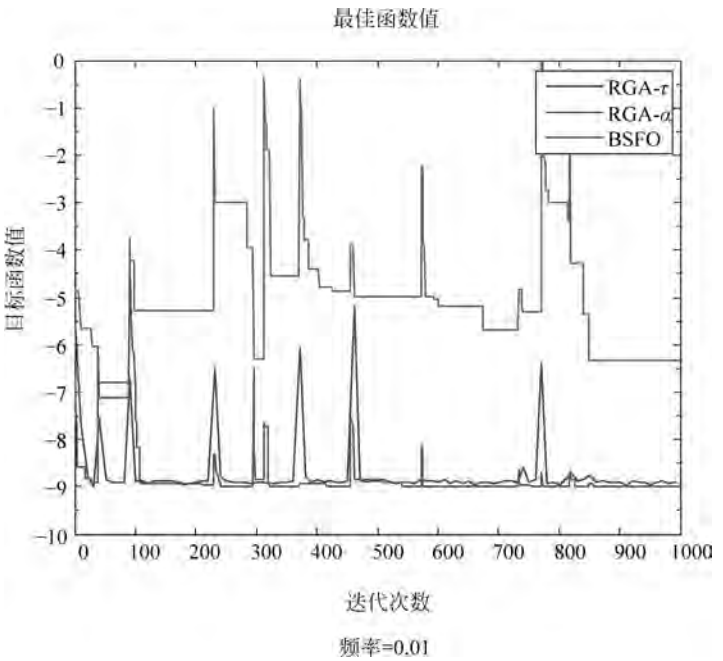
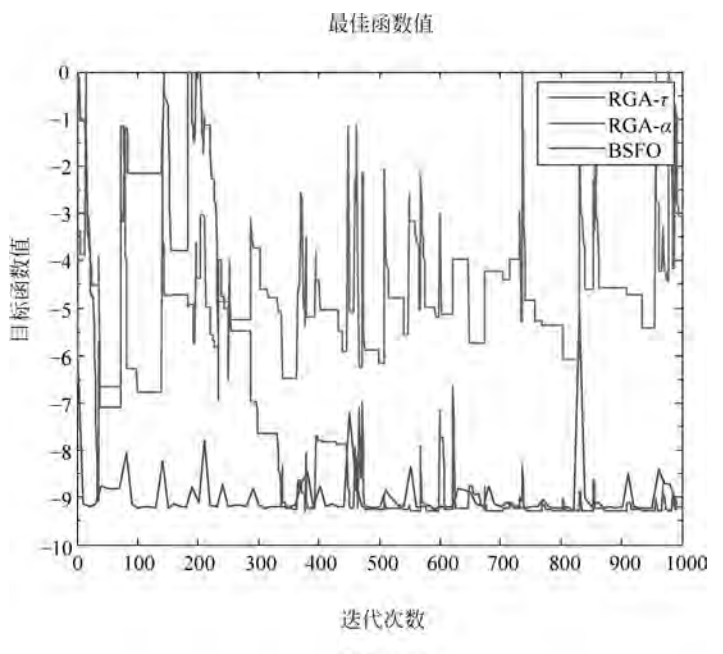


图 3-53　动态环境下的收敛比较图



频率=0.01



频率=0.05

图 3-53　（续）

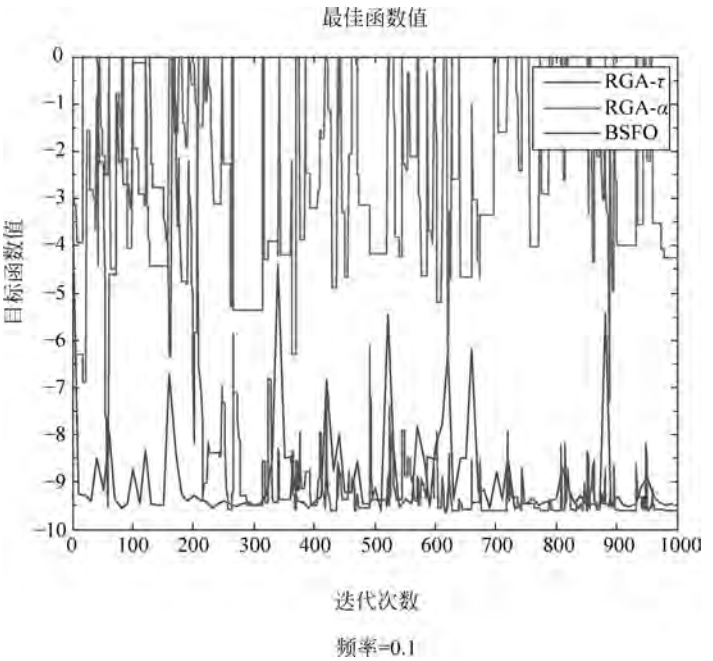


图 3-53 （续）

表 3-18 准确性比较

频　　率	BSFO	RGA-α	RGA-τ	对比(%)	
				BSFO 和 RGA-τ	RGA-α 和 RGA-τ
0.001	0.870 254	0.537 877	0.984 100	13.0819	82.9601
0.005	0.728 225	0.217 122	0.964 340	32.4233	3441.47
0.01	0.853 273	0.279 796	0.968 422	13.4950	246.118
0.05	0.694 721	0.385 752	0.943 577	35.8211	144.607
0.1	0.754 733	0.118 220	0.907 868	20.2899	667.946
0.2	0.839 162	0.233 340	0.914 781	9.011 32	292.037
0.3	0.866 910	0.348 427	0.909 543	4.917 89	161.043
0.5	0.870 479	0.471 658	0.844 206	−3.018 24	78.9868
0.8	0.627 434	0.190 54	0.742 113	18.2775	289.480

表 3-19 展示了在 3 种动态环境下 BSFO、RGA-α 和 RGA-τ 稳定性的比较结果。稳定性是与准确性相关的。稳定性的期望值是零，其越小越稳定。从表 3-19 可以看出，在 3 种情况下，RGA-τ 的稳定性明显要高于 RGA-α。在第 3 种情况下，也就是环境改变非常频繁的情况下，RGA-τ 的稳定性并不如 BSFO，可见在环境高频变化下，RGA-τ 仍需要更好的策略对其进行改进，以求达到更好的效果。

表 3-19 稳定性比较

频 率	BSFO	RGA-α	RGA-τ	对比(%)	
				BSFO 和 RGA-τ	RGA-α 和 RGA-τ
0.001	0.037 255 2	0.088 198	0.001 313 14	−96.4753	−98.5111
0.005	0.023 270 0	0.088 051 9	0.005 149 48	−77.8707	−94.1518
0.01	0.024 338 4	0.104 358	0.008 057 72	−66.8929	−92.2788
0.05	0.026 272 8	0.106 878	0.018 664 2	−28.9599	−82.5369
0.1	0.030 436 1	0.071 544 3	0.036 058 0	18.4735	−49.5995
0.2	0.027 976 8	0.098 881 1	0.040 549 6	44.9401	−58.9916
0.3	0.020 832 5	0.094 661 5	0.049 762 1	138.867	−47.4316
0.5	0.042 307 9	0.097 739 1	0.079 343 0	87.5372	−18.8217
0.8	0.040 856 0	0.101 382	0.088 785 2	117.312	−12.4254

参 考 文 献

[1] Li J N, Liu J K. Ecological implication and behavior mechanism of food selection of mammalian herbivores [J]. Chinese Journal of Applied Ecology, 2003, 14(3): 439-442.

[2] Gendron R P, Staddon J E R. Searching for Cryptic Prey: The Effect of Search Rate[J]. American Naturalist, 1983, 121(2): 172-186.

[3] Krakauer D C, Rodríguez-Gironés, M. A. Searching and Learning in a Random Environment [J]. Journal of Theoretical Biology, 1995, 177(4): 417-429.

[4] Smith J N M. The food searching behaviour of two European thrushes. II: the adaptiveness of the search patterns[J]. Behaviour, 1974, 49(1-2): 1-60.

[5] Kislalioglu M, Gibson R N. Prey 'handling time' and its importance in food selection by the 15-spined stickleback, Spinachia (L.)[J]. Journal of Experimental Marine Biology and Ecology, 1976, 25(2): 151-158.

[6] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]//Proceedings of the first European conference on artificial life. 1992, 142: 134-142.

[7] Dorigo M, Birattari M, Stutzle T. Ant colony optimization [J]. IEEE Computational Intelligence Magazine, 2007, 1(4): 28-39.

[8] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm[J]. Applied soft computing, 2008, 8(1): 687-697.

[9] Drigo M. The ant system: optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 1-13.

[10] Stiitzle T, Hoos H. The MAX-MIN ant system and local search for the traveling salesman problem [C]//Proceedings of IEEE international conference on evolutionary computation. 1997: 309-314.

[11] Bullnheimer B, Hartl R F, Strauss C. A new rank-based version of the Ant System. A computational study[J]. 1997, 7(1): 25-38.

[12] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm[J]. Applied soft computing, 2008, 8(1): 687-697.

[13] Karaboga D, Akay B. A survey: algorithms simulating bee swarm intelligence [J]. Artificial intelligence review, 2009, 31(1-4): 61-85.

[14] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization:

artificial bee colony (ABC) algorithm[J]. Journal of global optimization,2007,39(3): 459-471.
[15] Stephens D W,Krebs J R. Foraging theory[M]. Princeton: Princeton University Press,1986.
[16] Giraldeau L A,Caraco T. Social foraging theory[M]. Princeton: Princeton University Press,2000.
[17] De Reffye P H,Blaise F,Houllier F. Modelling plant growth and architecture: some recent advances and applications to agronomy and forestry[J]. Current science,1997,73(11): 984-992.
[18] 金明现,王天铎. 玉米根系生长及向水性的模拟[J]. 植物学报,1996(05): 384-390.
[19] Krakauer D C,Rodrìguez-Gironés M A. Searching and learning in a random environment[J]. Journal of Theoretical Biology,1995,177(4): 417-429.
[20] Hill S,Burrows M T,Hughes R N. Increased turning per unit distance as an area - restricted search mechanism in a pause - travel predator,juvenile plaice,foraging for buried bivalves[J]. Journal of Fish Biology,2000,56(6): 1497-1508.
[21] Bénichou O,Coppey M,Moreau M,et al. Optimal search strategies for hidden targets[J]. Physical review letters,2005,94(19): 198101.
[22] Tamplin M L. Inactivation of Escherichia coli O157: H7 in simulated human gastric fluid[J]. Appl. Environ. Microbiol.,2005,71(1): 320-325.
[23] Bollinger C J T,Bailey J E,Kallio P T. Novel Hemoglobins to Enhance Microaerobic Growth and Substrate Utilization in Escherichiacoli[J]. Biotechnology progress,2001,17(5): 798-808.
[24] Byappanahalli M,Fowler M,Shively D,et al. Ubiquity and persistence of Escherichia coli in a Midwestern coastal stream[J]. Appl. Environ. Microbiol.,2003,69(8): 4549-4555.
[25] Schierack P,Steinrück H,Kleta S,et al. Virulence factor gene profiles of Escherichia coli isolates from clinically healthy pigs[J]. Appl. Environ. Microbiol.,2006,72(10): 6680-6686.
[26] Marykwas D L,Berg H C. A mutational analysis of the interaction between FliG and FliM,two components of the flagellar motor of Escherichia coli[J]. Journal of bacteriology,1996,178(5): 1289-1294.
[27] http: //www. ericsecho. org/whatisec. htm
[28] Chen H,Zhu Y,Hu K. Adaptive Bacterial Foraging Optimization [C]//Abstract and Applied Analysis. Hindawi,2011,2011.
[29] Chen H,Zhu Y,Hu K,et al. Dynamic RFID network optimization using a self-adaptive bacterial foraging algorithm[J]. International Journal of Artificial Intelligence,2011,7(11): 219-231.
[30] Chen H,Zhu Y,Hu K. Self-adaptation in bacterial foraging optimization algorithm[C]//2008 3rd International Conference on Intelligent System and Knowledge Engineering. IEEE,2008,1: 1026-1031.
[31] Shi Y,Eberhart R,Empirical C. Study of Particle Swarm Optimization[C]//Proceeding of the World Multiconference on Systemics. Cybernetics and Informatics. Orlando: IEEE,2000,1945-1950.
[32] Angeline P J. Using selection to improve particle swarm optimization[C]//1998 IEEE International Conference on Evolutionary Computation Proceedings. Anchorage: IEEE,1998: 84-89.
[33] Passino K M. Biomimicry of bacterial foraging for distributed optimization and control[J]. IEEE control systems magazine,2002,22(3): 52-67.
[34] Hsiao Y T,Chuang C L,Chien C C. Ant colony optimization for best path planning[C]//IEEE International Symposium on Communications and Information Technology,2004. ISCIT 2004. IEEE,2004,1: 109-113.
[35] Shi Y,Eberhart R C. Parameter selection in particle swarm optimization [C]//International conference on evolutionary programming. Berlin: Springer,1998: 591-600.
[36] Sumathi S,Hamsapriya T,Surekha P. Evolutionary intelligence: an introduction to theory and applications with Matlab[M]. Berlin: Springer,2008.

[37] 王东生，曹磊. 混沌、分形及其应用[M]. 合肥：中国科学技术大学出版社，1995.

[38] Mainzer K. Thinking in complexity：The computational dynamics of matter，mind，and mankind[M]. Augsburg：Springer Science & Business Media，1997.

[39] Zhang H，Zhu Y，Chen H. Root growth model for simulation of plant root system and numerical function optimization[C]//International Conference on Intelligent Computing. Berlin：Springer，2012：641-648.

[40] KRINK T. Particle swarm optimisation with spatial particle extension[C]//Proceedings of the IEEE Congress on Evolutionary Computation. New York：IEEE，2002.

[41] Shi Y，Eberhart R. A modified particle swarm optimizer[C]//1998 IEEE international conference on evolutionary computation proceedings. Anchorage：IEEE，1998.

[42] Branke J. Evolutionary optimization in dynamic environments[M]. Karlsruhe：Springer Science & Business Media，2002.

[43] Cruz C，González J R，Pelta D A. Optimization in dynamic environments：a survey on problems，methods and measures[J]. Soft Computing，2011，15(7)：1427-1448.

[44] Branke J. The moving peaks benchmark website.[Online]. Available：http：//www. aifb. uni-karlsmhe. de/～jbr/movpeaks.

[45] Walker R. "Niche Selection" and the Evolution of Complex Behavior in a Changing Environment—A Simulation[J]. Artificial life，1999，5(3)：271-289.

[46] Chang C，Zhu Y，Guo H，et al. Study on electronic commerce recommendation system by CBR and AIS[C]//2008 7th World Congress on Intelligent Control and Automation. IEEE，2008：2110-2114.

[47] Niu B，Zhu Y L，Hu K Y，et al. A novel neural network training algorithm based on particle swarm optimizer and optimal foraging theory[J]. Dynamics of Continuous Discrete and Impulsive Systems，2007，14(3)，370-375.

[48] He X，Zhu Y，Hu K，et al. A Swarm-Based Learning Method Inspired by Social Insects[C]// International Conference on Intelligent Computing. Berlin：Springer，2007：525-533.

[49] Xu J W，Zhu Y L，Zhang F. An Operating Model for a Class of Manufacturing/Remanufacturing System[J]. Proceedings of The International Conference on Management of Technology，2007. 8，506-511.

[50] Niu B，Zhu Y，He X，et al. MCPSO：A multi-swarm cooperative particle swarm optimizer[J]. Applied Mathematics and Computation，2007，185(2)：1050-1062.