一种求解符号回归问题的粒子群优化算法

马 炫 [1, 2] 李 星 [1] 唐荣俊 [1] 刘 庆 [1, 2]

摘 要 符号回归以构建一个能拟合给定数据集的函数模型为目的, 是对基本函数、运算符、变量等进行组合优化的过程. 本文提出了一种求解符号回归问题的粒子群优化算法. 算法以语法树对函数模型进行表达, 采用基因表达式将语法树编码为一个粒子, 设计了粒子的飞行方法及 r-邻域环状拓扑的粒子学习关系. 为使粒子具有跳出局部极值的能力和减轻粒子快速趋同对全局寻优造成的不利影响, 分别设计了突变算子和散开算子. 此外, 为了得到比较简洁的函数模型, 在粒子的评价函数中以罚函数的方式对编码的有效长度进行控制. 仿真实验表明, 提出的算法可以获得拟合精度更高、简洁性更好的函数模型.

关键词 基因表达式编程, 粒子群优化算法, 符号回归, 演化建模

引用格式 马炫, 李星, 唐荣俊, 刘庆. 一种求解符号回归问题的粒子群优化算法. 自动化学报, 2020, **46**(8): 1714−1726

DOI 10.16383/j.aas.c180035

A Particle Swarm Optimization Approach for Symbolic Regression

MA Xuan[1, 2] LI Xing[1] TANG Rong-Jun[1] LIU Qing[1, 2]

Abstract Symbolic regression is to construct a function model that fits a given dataset. It is the process of optimally combining various basic functions, operators, and variables. This paper proposes a particle swarm optimization-based algorithm for symbolic regression. In the proposed algorithm, the functional model to be established is represented as a syntax tree, which is encoded as a particle through gene-expression. A specific implementation of particles flying and the r-neighborhood learning mechanism of particle swarm were designed. To make particles be capable of jumping out local extremum and to mitigate the negative influence on global optimization resulted from the fast convergence of the particle swarm, mutation and scatter are respectively introduced into the proposed algorithm as operators. Besides, in order to obtain concise functional model, the valid length of the gene-expression-based coding scheme is controlled in manner of introducing a penalty term to the particle evaluation function. Exhaustive simulation experiments are carried out and the results show that, the proposed algorithm can obtain the functional model with higher fitting precision and better conciseness.

Key words Gene-expression programming, particle swarm optimization, symbolic regression, evolutionary modeling

Citation Ma Xuan, Li Xing, Tang Rong-Jun, Liu Qing. A particle swarm optimization approach for symbolic regression. *Acta Automatica Sinica*, 2020, **46**(8): 1714−1726

在科学研究和工业生产中, 有许多复杂的系统或非线性现象, 为了明确系统因果关系或系统内部变量之间的相互关系, 需要对系统建立模型. 建模方法的研究一直是学术界和工程界共同关注的焦点[1−2]. 纯机理建模能够反映系统对象的本质特性, 具有可靠性高、解释性强的特点, 但是这种方法的建模过程耗时、繁琐, 而且建立复杂系统模型时一般也是经过简化而得到的模型. 数据驱动建模的方法可以在机理尚不清楚的情况下利用输入、输出信息建立数学模型. 回归分析和人工神经网络都属于数据驱动建模, 但一般的回归分析法, 需要假设一个待定参数的模型结构, 对参数进行优化, 这种方法只能优化模型的参数而不能优化模型的结构. 人工神经网络法通过输入、输出数据的训练来调整网络的连接权值, 从而构建虚拟的模型, 但是人工神经网络法属于黑箱建模法, 不能给出数学表达式, 这使得对模型的理解和解释变得困难[3]. 1992 年美国学者 Koza 提出的遗传程序设计 (Genetic programming, GP)[4] 为符号回归方法研究做出了开拓性贡献, 符号回归既能优化结构和参数, 又能给出明确的数学表达形式. 2001 年, Ferreira 在 GP 的基础上提出了基因表达式编程 (Gene expression programming, GEP)[5], 进一步推动了符号回归方法的研究, 目前 GEP 仍然受到人们的关注并得到广

收稿日期 2018-01-16 录用日期 2018-10-06
Manuscript received January 16, 2018; accepted October 6, 2018
国家自然科学基金 (61502385) 资助
Supported by National Natural Science Foundation of China (61502385)
本文责任编委 伍洲
Recommended by Associate Editor WU Zhou
1. 西安理工大学自动化与信息工程学院 西安 710048 2. 陕西省复杂系统控制与智能信息处理重点实验室 西安 710048
1. School of Automation and Information Engineering, Xi′an University of Technology, Xi′an 710048 2. Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing, Xi′an 710048

泛应用[6−7]. GP 和 GEP 为符号回归提供了解决问题的方法基础, 但它们都是基于优胜劣汰的竞争机制. 竞争机制在一定程度上会使种群多样性减小, 影响符号回归算法的性能. 本文根据群体智能算法的学习机制, 在粒子群优化算法基本原理的基础上, 针对符号回归的特点, 提出了一种基于粒子群算法的符号回归方法.

1 相关工作

符号回归在科学研究和解决实际工程问题等领域有着重要作用[8−10]. GP 采用了一种称为语法树 (Syntax tree) 的动态树形结构表示群体中的每一个个体, 这种树状结构, 随着遗传操作的迭代, 个体语法树很容易出现代码膨胀问题[11]. 针对这个问题, 文献 [12] 在多项式形式的函数建模问题中提出了通过构建模块保护和扩展机制的解决方案. 文献 [13] 提出了基于线性遗传规划 (Linear genetic programming, LGP) 的方法, 采用线性结构来表示个体, 并利用一定的代码膨胀控制技术增加种群中编码长度较小的个体的数量, 通过实验验证了算法的有效性. GEP[5] 采用固定长度的线性字符串表示种群中的个体, 有效解决了代码膨胀问题. 文献 [14] 提出了应用堆栈技术进行解码的 GEP (Stack-based method, S_GEP), 提高了解码效率. 文献 [15] 提出了一种自学习型的 GEP (Self-learning gene expression programming, SL-GEP), 设计了嵌有子功能的染色体, 子功能可以自学习和自演化, 也可以成为另一个子功能的输入参数, 并针对染色体的进化提出了基于差分进化的搜索机制.

上述研究的寻优机制都是以优胜劣汰的竞争为主的, 竞争机制容易使种群的多样性减小, 而且搜索性能也容易受到遗传操作技术的影响. 为了提高搜索到全局最优解的可能性, 种群规模一般取的比较大, 这也相应增加了算法的计算量和计算时间.

目前, 群智能算法在解决符号回归问题中表现出了令人振奋的能力, 并逐渐引起重视[16]. 文献 [17] 提出了一种用于符号回归的人工蜂群规划 (Artificial bee colony programming, ABCP) 算法, 由于沿用了 GP 中的语法树, 代码膨胀的问题仍然存在. 文献 [18] 则将人工鱼群算法 (Artificial fish school algorithm, AFSA) 的优化框架和 GEP 中的基因表达式编码进行了巧妙结合, 较好地解决了符号回归问题, 但 AFSA 较多的待调参数降低了该算法在更多问题上的实用性. 文献 [19] 提出了一种 GP 与粒子群算法 (Particle swarm optimization, PSO) 的混合算法 (Hybrid genetic programming with particle swarm optimization, HGPPSO), 因其仍由 GP 发展而来, 也存在代码膨胀问题. 文献 [20] 提出了一种通用的用于优化树型结构的 PSO (PSO-based tree discovering algorithm, TSO), 该算法在以语法树为优化对象时具有一定的求解符号回归问题的能力. 但是, 由于其并非针对符号回归而提出, 该算法不能满足符号回归问题的一些特殊要求, 如: 模型的化简.

本文根据粒子群优化算法基本原理提出了一种求解符号回归问题的优化算法, 采用 GEP 中的基因表达式对粒子进行编码, 针对粒子编码特征设计了相应的粒子飞行方法. 为了提高粒子的寻优性能, 一是采用 r-邻域环状粒子群拓扑结构, 增强粒子的相互协作性, 提高粒子的搜索效率; 二是通过设计突变算子和散开算子, 使粒子具有跳出局部极值的能力并减轻粒子快速趋同对全局寻优造成的不利影响. 而且, 还对编码的有效长度进行约束以得到简洁的函数模型结构. 本文在第 2 节简要介绍了粒子群优化算法基本原理, 然后给出了详细的符号回归粒子群优化算法. 在第 3 节的数值实验中通过与 GP、GEP、ABCP、AFSA 和 TSO 等符号回归算法进行比较, 验证并分析了提出算法的有效性. 最后是对本文的总结.

2 解符号回归问题的粒子群优化算法

2.1 标准粒子群优化算法

PSO 是 Kennedy 和 Eberhart 受鸟类集群规律性飞行的启发而提出的一种群智能算法[21]. 该算法模型以种群中个体之间的学习为核心, 在搜索空间每个粒子通过学习机制向最优解区域飞行. PSO 具有调节参数少且收敛速度快的优点, 在很多领域得到了应用, 如: 随机优化[22]、覆盖阵列生成[23]、资源分配[24]、机器人路径跟踪优化[25]、电池储能系统[26]等, 是受到普遍关注的群体智能计算方法.

在粒子群优化算法中, 群体中所有个体都被抽象为在 d 维空间中没有重量和体积的粒子, 并在搜索空间中以不同的速度飞行. 粒子在飞行时, 通过追踪两个 “极值” 来不断地更新自己的速度和位置, 一个是全局极值, 一个是个体极值, 其更新的速度和位置公式如下[21]:

$$\begin{aligned} v_i^d(t+1) &= \omega v_i^d(t) + c_1 r_1 (pbest_i^d(t) - x_i^d(t)) + \\ &\quad c_2 r_2 (gbest_i^d(t) - x_i^d(t)) \\ x_i^d(t+1) &= x_i^d(t) + v_i^d(t+1) \end{aligned} \tag{1}$$

其中, t 为群体当前迭代的代数, ω 为惯性权重, 起到对全局与局部搜索能力平衡的作用. c_1 和 c_2 为加速度常数, r_1 和 r_2 为 [0, 1] 之间均匀产生的随机数.

而 $v_i^d(t)$ 和 $x_i^d(t)$ 分别表示当前个体粒子 i 的速度和位置, $pbest_i^d(t)$ 则是粒子本身所经历的最好位置, 即个体极值; $gbest_i^d(t)$ 则是整个群体目前的最好位置, 即群体极值. 在式 (1) 中, $\omega v_i^d(t)$ 表示粒子根据自身当前部分速度进行惯性运动; $c_1r_1(pbest_i^d(t)-x_i^d(t))$ 表示当前粒子对自身最好位置的追踪, 体现了粒子的自我学习性; $c_2r_2(gbest_i^d(t)-x_i^d(t))$ 表示当前粒子对群体最好位置的追踪, 体现了粒子的社会学习性.

2.2 粒子编码

语法树是对函数模型的一种形式化表达, 具有直观、易解码的优点, 但也存在代码膨胀、操作不易实现等不足. Ferreira 进一步将语法树编码为一个由运算符 (如: +、−、×、÷)、函数 (如: sin、cos、ln、e^x)、和终止符 (如: 输入变量或常量) 构成的线性串, 称为 "基因表达式 (Gene expression)". 由于基因表达式结构简单, 易于实现, 语义表达灵活多变, 并且避免了代码膨胀等问题, 已被广泛采用[14, 18]. 因此, 本文亦采用基因表达式对粒子进行编码. 现以式 (2) 中的函数为例, 结合图 1 所示的语法树对基因表达式的编码规则予以说明. 将语法树上的节点以自上而下、从左到右的顺序进行罗列, 即得到基因表达式的编码区 (Coding region), 如表 1 中节点序号 0～4 对应的符号; 为了确保操作基因表达式不会产生无效语法树, Ferreira 为基因表达式设计了冗余机制, 即在编码区后追加一段非编码区 (Non-coding region). 非编码区是基因表达式的重要组成部分, 其作用是使得遗传操作后的编码仍具有合法性, 如表 1 中节点序号 5～10 对应的符号. 另一方面, 基因表达式又被分为首段和尾段, 其中, 首段 Head 全部由运算符、函数和终止符等构成, 而尾段 Tail 仅包含终止符号. 显然, 基因表达式的长度 Length 为首段 Len (Head) 与尾段长度 Len (Tail) 之和. 为了保证语法树叶节点上只出现终止符, 基因表达式的 Len (Tail) 为 Len (Head)×$(n-1)$, 其中 n 为基因表达式编码中运算符及函数所支配操作数的最大个数. 给定 Len (Head), 则 Len (Tail) 可确定, 进而得到基因表达式编码的总长度. 在初始化粒子时, 粒子编码首段 Head 和尾段 Tail 遵循不同的规则, 即: 首段各符号位从运算符、函数、及终止符中以等概率随机选取; 尾段各符号位则仅允许从终止符中进行选择.

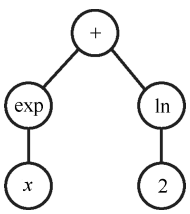


图 1 语法树

Fig. 1 Syntax tree

$$f(x)=\mathrm{e}^x+\ln 2 \tag{2}$$

表 1 粒子编码

Table 1 Particle coding

Node code	0	1	2	3	4	5	6	7	8	9	10
X	+	exp	ln	x	2	1	2	x	1	x	x

2.3 粒子评价

对粒子的评价主要考虑两个方面, 一个是粒子建模的拟合精度, 另一个是粒子编码的简洁性. 对于前者, 本文采用相对适应度函数对粒子个体进行评价, 评价函数如式 (3) 所示.

$$f=\frac{1}{n}\sum\nolimits_{i=1}^{n}\frac{|f(x_i)-y_i|}{|y_i|} \tag{3}$$

其中, f 表示粒子的适应度值, n 为观测数据 (x_i, y_i) 的个数; x_i, y_i 分别为模型输入, 输出的观测数据; $f(x_i)$ 为实际得到的模型输出数据. 符号回归实际上是一个误差驱动的过程, 求解拟合误差最小的函数模型, 即: 适应度值越小, 粒子越优. 而对于后者, 文献 [27] 将表达式的描述长度 (编码的有效长度) 作为表达式简洁性评价标准, 当编码的有效长度较大时, 会使组合优化的符号较多, 得到的数学表达式精度可能更高, 但会更复杂; 反之, 会使组合优化的符号较少, 得到的数学表达式更加简洁, 但精度可能较低. 为了使模型在到达一定精度的情况下具有较好的简洁性, 本文对编码的有效长度进行约束. 当编码的实际有效长度超过编码的约束长度时, 对适应度函数增加一个惩罚项, 即: 罚因子与超过部分的乘积, 为了使编码的有效长度向设定值 "L" 靠近, 本文采用外点罚函数的方式对不满足约束的粒子进行评价, 如式 (4) 所示.

$$f'=\begin{cases}\frac{1}{n}\sum\limits_{i=1}^{n}\frac{|f(x_i)-y_i|}{|y_i|}+\delta(l-L), & l>L\\ \frac{1}{n}\sum\limits_{i=1}^{n}\frac{|f(x_i)-y_i|}{|y_i|}, & l\le L\end{cases} \tag{4}$$

其中, f' 为被惩罚后的适应度值, δ 为罚因子, l 为编码的实际长度且最小长度为 1, L 为编码的约束长度. 当编码的实际有效长度超过编码的约束长度时, 适应度值会随着惩罚项的增加而增大, 使得编码长度超过约束长度的粒子的适应度变差, 并在迭代过

程中被淘汰, 实现将编码的实际有效长度控制在一定的范围之内, 进而得到比较简洁的模型结构.

2.4　粒子飞行

粒子群优化算法中粒子向最优个体学习就是粒子向学习目标飞行. 在符号回归问题中, 如何实现树型结构粒子的飞行是一个需要解决的关键问题. 由于粒子被表示成一个语法树, 根据基本粒子群优化算法原理, 如果目标树中的部分节点可以被一个树继承, 那么这个树就具有了目标树的部分信息, 意味着树型粒子的飞行学习. 由于语法树和粒子编码是一一对应的关系, 因此对编码进行操作可以实现粒子的飞行学习. 本文的粒子飞行实现方法如下: 假定当前粒子为 X, 全局极值粒子为 $gbest$, 个体极值粒子为 $pbest$. 当 X 向 $gbest$ 学习时, 在编码区中随机选择 $gbest$ 的一个或多个节点, 然后用选中的节点代替 X 中对应的节点. X 向个体极值 $pbest$ 的学习方式与其向 $gbest$ 的学习方式相同, 只是学习的目标不同. 下面以选择学习目标的一个节点为例说明粒子的飞行方法.

如表 2 所示, 假设在 $gbest$ 编码区随机选择了一个编号为 1 的节点, 就用 $gbest$ 中编号为 1 的节点替换粒子 X 的 1 号节点. 假设在 $pbest$ 编码区随机选择了一个编号为 3 的节点, 就用 $pbest$ 中编号为 3 的节点替换粒子 X 的 3 号节点, 这样可得到新的粒子 X' 的编码, X' 的语法树如图 2 所示.

表 2　粒子 X、$gbest$、$pbest$ 以及 X' 的编码

Table 2　The code of particle X, $gbest$, $pbest$ and X'

Node code	0	1	2	3	4	5	6	7	8	9	10
X	+	x	sin	−	ln	x	a	x	x	a	a
$pbest$	+	x	cos	/	−	x	a	x	a	x	a
$gbest$	exp	+	a	−	ln	a	a	x	a	x	a
X'	+	+	sin	/	ln	x	a	x	x	a	a

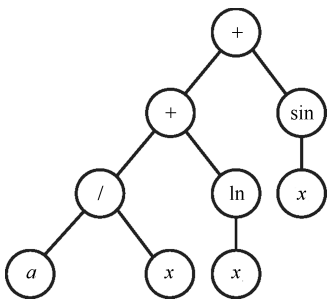
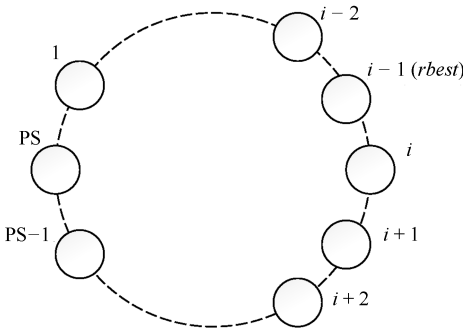


图 2　X' 的语法树

Fig. 2　The syntax tree of X'

2.5　r-邻域环状粒子群拓扑

GBest 模型的 PSO 收敛快速但易早熟, LBest 模型的 PSO 不易早熟但收敛缓慢[28]. 结合二者的优点, 本文采用 r-邻域环状粒子群拓扑, 如图 3 所示. 将所有粒子按照初始化顺序依次排列, 并将首尾粒子连接, 构成一个环形的种群结构. 为环状种群拓扑中的每个粒子设置一个感知范围 r, 即对于环上的任意一个粒子 i, 其 r-邻域为自身和与其相邻的 $2r$ 个粒子, 例如, 当 $r = 1$ 时, 粒子 i 和与它左右相邻的粒子构成具有 3 个粒子的邻域 $(i-1、i、i+1)$. 粒子 i 在飞行中以其 r-邻域内的最佳粒子 $(rbest)$ 为社会学习对象进行优化信息的传递. 当一个粒子要与邻域之外的粒子实现信息交流时, 至少需要经过一个粒子作为媒介, 这样, 媒介粒子就会把自身的部分结构信息融入语法树的优化过程, 从而增强了粒子的局部学习能力.



PS: 种群规模　i: 粒子标记从 1 到 PS

图 3　r-邻域环状粒子群拓扑

Fig. 3　The ring topology of r-neighborhood particles

2.6　突变算子与散开算子

协同搜索的学习机制使粒子群算法得以快速收敛, 不过也带来了两个缺陷: 一是过早出现的较优解使种群陷入局部区域而无法跳出, 二是粒子种群的快速趋同降低了粒子的搜索效率.

为了避免算法陷入局部解的现象, 有研究者将遗传算法的变异机制引入粒子群优化算法[29], 使粒子以一定概率进行变异从而形成算法跳出局部解的能力. 本文也采用变异的思想, 根据符号回归问题粒子编码的特点设计了突变算子, 突变方法如下: 首先产生一个随机整数 (不大于编码有效长度的正整数), 作为突变节点的个数, 然后在编码区内随机确定要突变的节点位置; 最后在突变位置上对该节点进行突变. 当突变位置为编码首段区域时, 随机替换为函数、运算符或终止符; 当位于编码尾段区域, 随机替换为终止符. 假设突变节点个数为 3, 如表 3 所示, 随机选择的节点编号分别为 1、3、5. 假设首段长度这一参数设置为 5, 则第 2 个节点 (1 号节点) 和第

4 个节点 (3 号节点) 位于编码首段区域, 而第 6 个节点 (5 号节点) 位于编码尾段区域, 其中 1 号位节点操作符 “+” 随机替换为 “∗”; 3 号位节点操作符 “/” 随机替换为 “cos”; 5 号位节点输入变量 “x” 随机替换为常量 “a”. 三个节点突变完成后, 粒子 X 被突变为 X'.

表 3 粒子 X 和 X' 的编码

Table 3 The code of particle X and X'

Node code	0	1	2	3	4	5	6	7	8	9	10
X	+	+	sin	/	ln	x	a	x	x	a	a
X'	+	×	sin	cos	ln	a	a	x	x	a	a

为了减轻粒子种群快速趋同带来的不利影响, 本文设计了散开算子. 对搜索空间中处于某一位置上的多个相同的粒子仅保留一个, 而对其余的粒子进行突变使其离开原位置, 图 4 给出了粒子散开算子的示意图. 假设搜索空间中 d 点同时聚集了 A、B、C、D、E、F、G 等 7 个粒子. 使粒子 A 保持不变, 其余 6 个粒子被移至其他的位置 (图 4 中虚线箭头所指). 在粒子散开算子中需要先检验种群中其余粒子是否与当前粒子相同, 即检验粒子的编码结构是否与当前粒子相同. 检验方法为: 首先检测粒子适应度值是否相等, 若不相等, 其编码结构必然不同; 当粒子适应度值相等时, 再比较编码结构. 在比较过程中, 只要出现节点不同, 就判定粒子编码结构不同. 对于相同的粒子, 使其突变而改变位置, 突变方法如前所述.

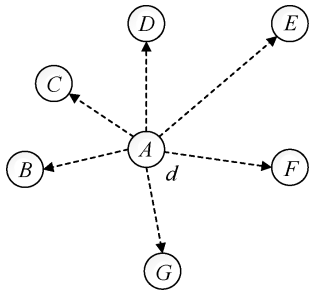


图 4 散开算子

Fig. 4 Scattering operator

2.7 算法流程

算法开始时按照设置的参数初始化种群粒子个体, 然后执行散开算子, 改变与当前粒子编码结构相同的粒子的位置, 并更新变化后粒子对应的个体极值. 接着产生 0∼1 之间的随机数, 并判断产生的随机数 rand 是否大于突变概率 PM, 若随机数 rand 大于突变概率 PM 则执行粒子飞行算子, 即: 粒子先向具有 r-邻域极值的粒子 ($rbest$) 学习, 再向具有个体极值的粒子学习; 若随机数 rand 小于突变概率 PM 则当前粒子执行突变算子. 当每个粒子都更新了个体极值后, 粒子在当前代的行为结束. 当粒子达到设定的迭代次数 MI, 算法结束, 否则算法继续执行下一代的行为. 算法执行流程如图 5 所示.

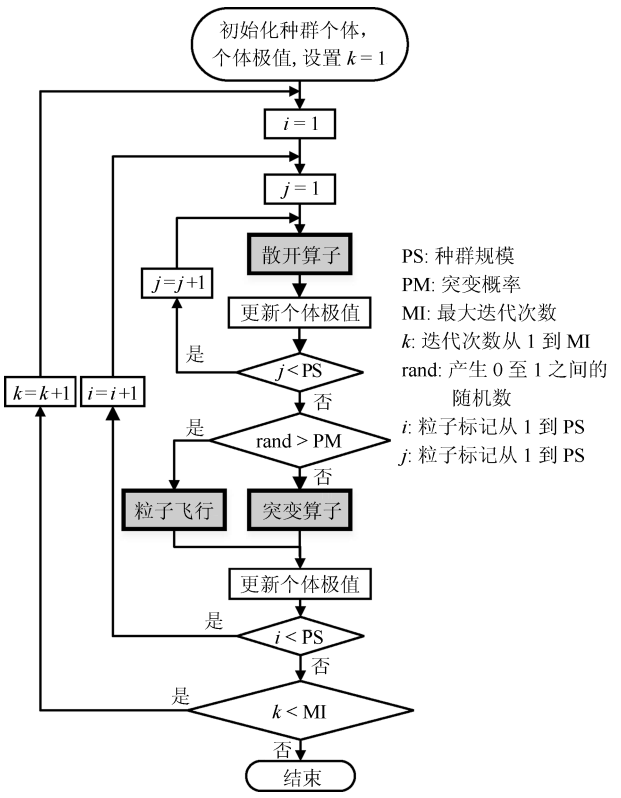


图 5 算法流程图

Fig. 5 Algorithm flowchart

3 数值实验

为验证本文算法的性能, 通过以下 4 个实验项目进行考察. 算法用 C++ 编程, 在 Windows7 系统 (64 位) 上运行, CPU 为 Intel(R) Core (TM) i5-2450M (2.5 GHz), 内存 8 GB.

3.1 Benchmark 函数测试和同类算法比较

采用文献 [17] 所使用的 10 个具有不同形态的函数对本文算法性能进行测试, 并与其他算法的结果进行比较. 实验中使用相同的算法终止条件、符号类型、适应度测试函数和性能评价指标等以保证实验公平性. 比较的算法为文献 [30] 中的 SC (Standard crossover)、NSM (No same mate)、SAC (Semantics aware crossover)、CAC (Context aware crossover)、SBS (Soft brood selection) 和 SSC (Semantic similarity-based crossover), 以及文献 [17] 的 ABCP 和文献 [18] 的 AFSA. 本文算法参数见表 4. 所有算法均采用绝对误差之和作为评价函数如式 (5) 所示.

表 4 算法参数

Table 4 Algorithm parameters

参数	参数取值
节点评价个数	15 000 000
种群数量	50
首段长度	10
约束长度	18
罚因子	0.0001
环状结构邻域大小	$2r+1, r=2$
终止符	x, 1 (单变量); x, y (双变量)
函数及运算符	$+, -, \times, /$, sin, cos, e^x, ln
测试次数	100

$$f=\sum_{i=1}^{n}|f(x_i)-y_i| \tag{5}$$

其中, n 为观测数据 (x_i, y_i) 的个数; x_i, y_i 分别为模型输入, 输出的观测数据; $f(x_i)$ 为实际得到的模型输出数据. 测试用的 10 个 Benchmark 函数如表 5 所示. 每个函数后面列出了变量取值区间和样本个数. 为了比较算法性能, 本文使用了文献 [30] 中的经典性能指标 (成功运行的百分比). 算法分别对这 10 个函数独立运行 100 次, 当被评价的节点总个数到达 15 000 000 时算法终止. 每次运行时只要存在适应度函数值小于 0.01 的个体就认为该次运行成功. 表 6 给出了各算法成功运行的百分比, 表 7 给出了各算法 100 次独立运行所得最佳个体的平均适应度. 被测试算法在各函数上的最佳表现以标粗字体显示. 在本文使用的实验平台上, 提出的算法对各测试函数的平均收敛时间分别达到 6.32 s, 6.63 s, 6.71 s, 6.52 s, 6.46 s, 6.69 s, 6.54 s, 6.39 s, 32.07 s, 33.15 s.

表 5 测试函数

Table 5 Benchmark functions

Functions	Fitcases
$F1=x^3+x^2+x$	20 random points $\subseteq[-1,1]$
$F2=x^4+x^3+x^2+x$	20 random points $\subseteq[-1,1]$
$F3=x^5+x^4+x^3+x^2+x$	20 random points $\subseteq[-1,1]$
$F4=x^6+x^5+x^4+x^3+x^2+x$	20 random points $\subseteq[-1,1]$
$F5=\sin(x^2)\cos(x)-1$	20 random points $\subseteq[-1,1]$
$F6=\sin(x)+\sin(x+x^2)$	20 random points $\subseteq[-1,1]$
$F7=\ln(x+1)+\ln(x^2+1)$	20 random points $\subseteq[0,2]$
$F8=\sqrt{x}$	20 random points $\subseteq[0,4]$
$F9=\sin(x)+\sin(y^2)$	100 random points $\subseteq[-1,1]$
$F10=2\sin(x)\cos(y)$	100 random points $\subseteq[-1,1]$

表 6 成功运行百分比 (%)

Table 6 Percentage of successful runs (%)

Techniques	Functions									
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
本文算法	**100**	**82**	**22**	3	**60**	84	11	79	**94**	72
AFSA	97	80	9	2	47	73	26	**98**	78	**80**
ABCP	89	50	**22**	**12**	57	**87**	58	37	33	21
SC	48	22	7	4	20	35	35	16	7	18
NSM	48	16	4	4	19	36	40	28	4	17
SAC2	53	25	7	4	17	32	25	13	4	4
SAC3	56	19	6	2	21	23	25	12	3	8
SAC4	53	17	11	1	20	23	29	14	3	8
SAC5	53	17	11	1	19	27	30	12	3	8
CAC1	34	19	7	7	12	22	25	9	1	15
CAC2	34	20	7	7	13	23	25	9	2	16
CAC4	35	22	7	8	12	22	26	10	3	16
SBS31	43	15	9	6	31	28	31	17	13	33
SBS32	42	26	7	8	36	27	44	30	17	27
SBS34	51	21	10	9	34	33	46	25	26	33
SBS41	41	22	9	5	31	34	38	25	19	33
SBS42	50	22	17	10	41	32	51	24	24	33
SBS44	40	25	16	9	35	43	42	28	33	34
SSC8	66	28	**22**	10	48	56	59	21	25	47
SSC12	67	33	14	**12**	47	47	66	38	37	51
SSC16	55	39	20	11	46	44	**67**	29	30	59
SSC20	58	27	10	9	52	48	63	26	39	51

表 7 平均最佳适应值 (%)

Table 7 Mean best fitness values (%)

Techniques	Functions									
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
本文算法	**0.00**	**0.03**	0.30	0.35	**0.02**	**0.01**	0.14	0.05	**0.01**	0.82
AFSA	**0.00**	0.06	0.35	0.33	**0.02**	0.05	0.13	**0.01**	0.23	**0.69**
ABCP	0.01	0.05	**0.07**	**0.10**	0.05	0.02	**0.06**	0.10	0.47	1.06
SC	0.18	0.26	0.39	0.41	0.21	0.22	0.13	0.26	5.54	2.26
NSM	0.16	0.29	0.34	0.40	0.19	0.17	0.11	0.19	5.44	2.16
SAC2	0.16	0.27	0.42	0.50	0.22	0.23	0.15	0.27	5.99	3.19
SAC3	0.13	0.27	0.41	0.48	0.18	0.23	0.15	0.27	5.77	3.13
SAC4	0.15	0.29	0.40	0.46	0.17	0.22	0.15	0.26	5.77	3.03
SAC5	0.15	0.29	0.51	0.46	0.17	0.21	0.15	0.26	5.77	2.98
CAC1	0.33	0.41	0.52	0.53	0.31	0.42	0.17	0.35	7.83	4.40
CAC2	0.32	0.41	0.53	0.53	0.31	0.42	0.17	0.35	7.38	4.30
CAC4	0.33	0.41	0.33	0.53	0.30	0.42	0.17	0.19	7.80	4.32
SBS31	0.18	0.29	0.30	0.36	0.17	0.30	0.15	0.18	4.78	2.75
SBS32	0.18	0.23	0.29	0.36	0.13	0.28	0.10	0.18	4.47	2.77
SBS34	0.16	0.23	0.31	0.33	0.13	0.21	0.11	0.19	4.17	2.90
SBS41	0.18	0.26	0.27	0.38	0.12	0.20	0.13	0.20	4.40	2.75
SBS42	0.12	0.24	0.29	0.30	0.12	0.18	0.10	0.16	3.95	2.76
SBS44	0.18	0.24	0.33	0.35	0.15	0.16	0.11	0.19	2.85	1.75
SSC8	0.09	0.15	0.19	0.29	0.10	0.09	0.07	0.15	3.91	1.53
SSC12	0.17	0.17	0.18	0.28	0.10	0.12	0.07	0.13	3.54	1.45
SSC16	0.10	0.15	0.23	0.26	0.10	0.10	**0.06**	0.14	3.11	1.22
SSC20	0.10	0.18	0.23	0.30	0.09	0.10	**0.06**	0.14	2.64	1.23

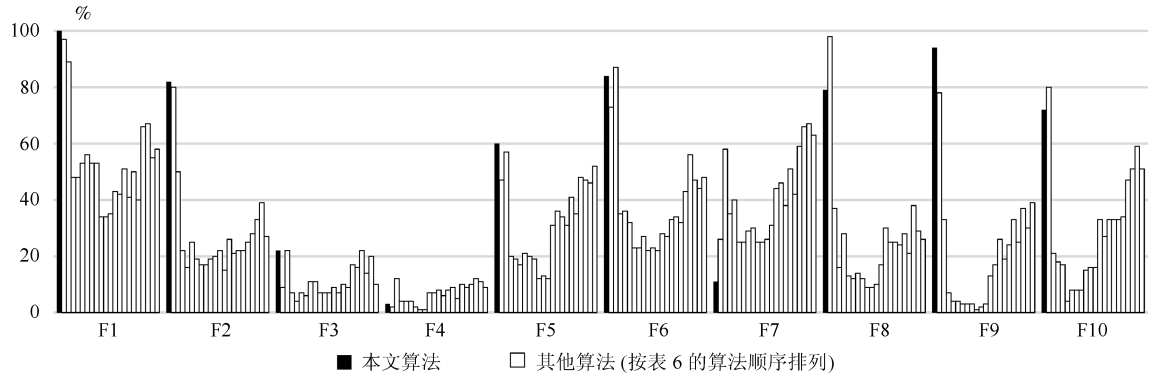


图 6 成功运行百分比柱状图

Fig. 6 Histogram of successful running percentage

表 6 中百分比数值越高, 表明算法在对应的测试函数上表现越好. 为了便于观察, 将表 6 中的数据用图 6 表示. 可以看出, 本文算法在 F1、F2、F3、F5、F9 等 5 个测试函数上表现最佳; 尽管在 F6 上略弱于 ABCP、在 F8 和 F10 上也不如 AFSA, 但其依然大幅优于绝大多数比较算法; 而仅在 F4 和 F7 上表现不佳. 表 7 中的数值越小, 表明算法在对应的测试函数上表现越好. 同样, 表 7 中的数据见图 7 所示. 可以看出, 本文算法在 F1、F2、F5、F6、F9 等 5 个测试函数上表现最佳; 尽管在 F8 和 F10 上不及 AFSA, 其表现依然大幅优于绝大多数比较算法; 仅在 F3、F4 和 F7 上表现不佳. 上述比较结果表明了本文算法的有效性与可靠性总体上优于其他算法.

3.2 “V” 型函数建模

“V” 型函数模型如式 (6) 所示, 该函数已经被许多研究者用于验证符号回归的建模能力[5, 18]. 由于文献 [18] 的 AFSA 通过 “V” 型函数建模验证了比 GEP 具有更高的性能, 本文也采用该函数并通过

与文献 [5] 中的 GEP 和文献 [18] 中的 AFSA 进行比较, 以验证提出的本文算法的有效性. AFSA 和 GEP 得到的最好的数学模型分别如式 (7) 和式 (8) 所示, 本文算法得到的如式 (9) 所示, 算法参数见表 8. 图 8 给出了三种函数模型的比较曲线, 在图 8 (a) 中分别表示了函数曲线与观测数据的拟合情况, 其

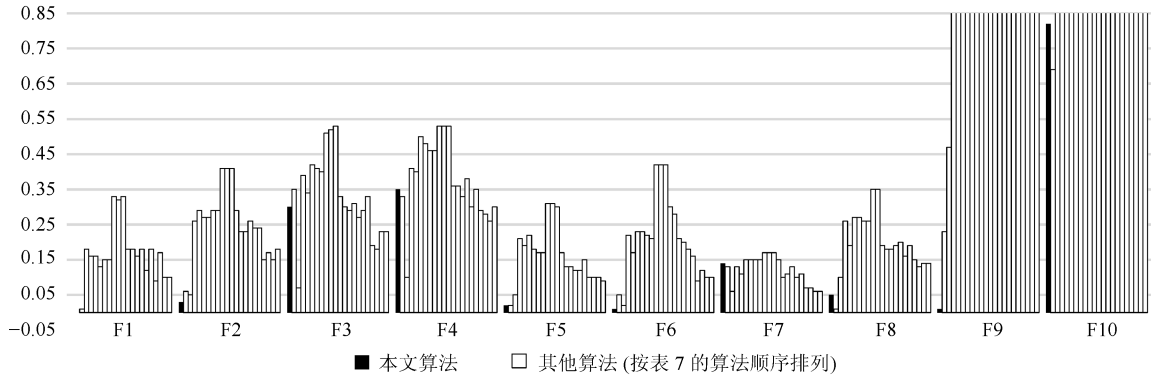
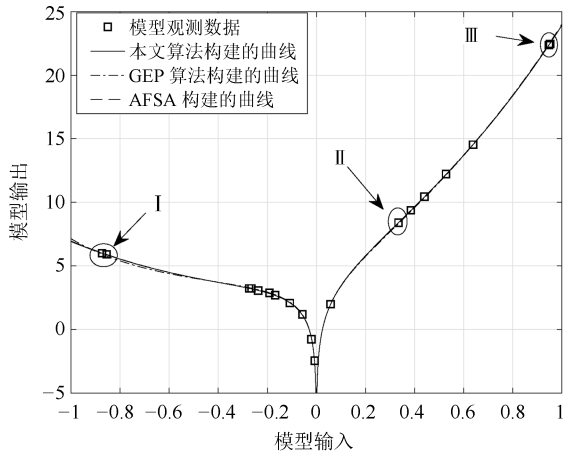


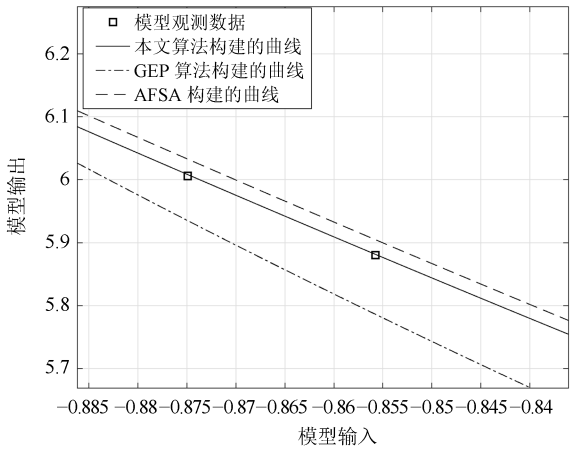
图 7　平均最佳适应值柱状图

Fig. 7　Histogram of average best fitness value



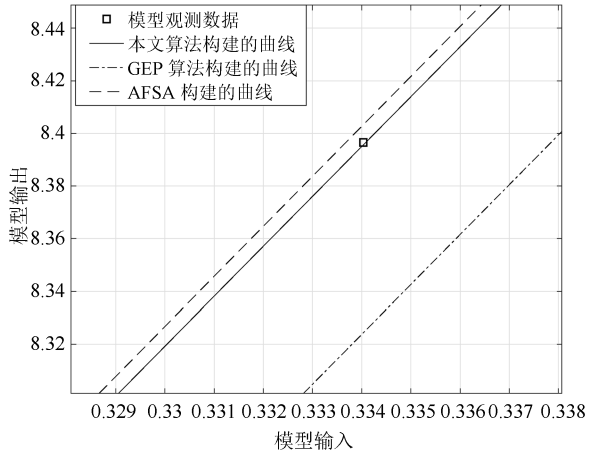
(a) 构建模型曲线的整体比较

(a) The overall comparison of the model curve



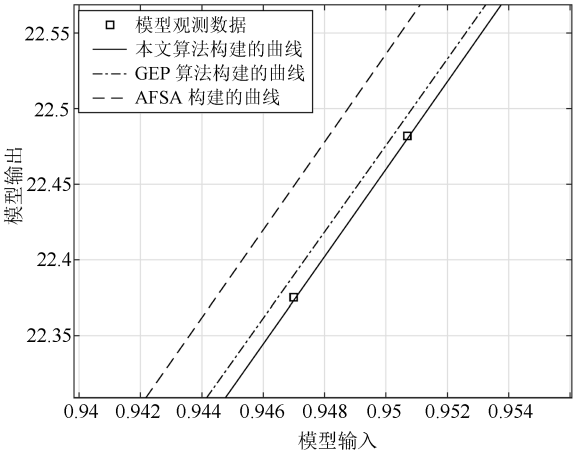
(b) 局部 Ⅰ 的比较

(b) Comparison of local Ⅰ



(c) 局部 Ⅱ 的比较

(c) Comparison of local Ⅱ



(d) 局部 Ⅲ 的比较

(d) Comparison of local Ⅲ

图 8　函数模型曲线

Fig. 8　Function model curve

输入、输出观测数据来自文献 [5].

$$y = 4.251x^2 + \ln(x^2) + 7.243\mathrm{e}^x \tag{6}$$

$$y = \ln(10^{x+0.613} \times \frac{x^2}{0.203}) + \mathrm{e}^{x^2+0.811} + 10^{\sin(x)} + 10^{\sin(\sin(x))} \tag{7}$$

$$y = \ln(x^2) - (\mathrm{e}^{\frac{0.618}{\ln(0.618)^3}})^2 \times (\sin(\frac{1.112}{\frac{\pi}{\mathrm{e}^x}}) - \frac{x^2}{10^{0.618-0.811-0.811}}) - \mathrm{e}^{2+x} \tag{8}$$

$$y = \left(\sqrt{\ln(\frac{0.7}{0.1^2} + \cos(0.7x^2(x+\sin(x))))}\right)^2 + \mathrm{e}^{\mathrm{e}^{2\times(0.2+0.1415)}+x} + \ln(x^2) \tag{9}$$

表 8 “V” 型函数建模的算法参数

Table 8 Algorithm parameters for “V” function

参数	参数取值
迭代次数	50 000
种群数量	300
首段长度	25
约束长度	35
罚因子	0.05
环状结构邻域大小	$2r+1, r=2$
终止符	$x, 1, 0.2, 2, 4, 0.7, 0.1, 0.1415, 0.01, 3$
函数及运算符	$+, -, \times, /, \sqrt{}, \sin, \cos, \mathrm{e}^x, \ln, x^2$
测试次数	100

从图 8 (a) 可以看出三种算法得到的函数曲线都与观测数据点基本吻合. 为了能够明显看出三者的区别, 对图 8 (a) 的局部区域 I、II、III 等分别进行放大, 如图 8 (b)、8 (c)、8 (d) 所示. 从放大图可以看出, 本文算法构建的函数模型曲线在数据点的拟合精度高于 GEP 和 AFSA 所得函数的拟合精度. 对拟合点进行统计, 在全部 20 个观测数据点中, 本文算法比 GEP 拟合精度更高的点有 20 个, 比 AFSA 拟合精度更高的点有 12 个. 这表明本文算法可以比 GEP 和 AFSA 得到拟合精度更高的函数模型.

上述算法实验中的罚因子和种群规模大小的设置是经验值, 对于比较复杂的函数, 而且要求简洁性比较高时, 如对 “V” 型函数建模, 罚因子和种群规模应设置的大一些, 以利于种群发现更多更好的个体, 以及获得更加简洁的函数模型.

下面, 对算法的收敛性进行考察. 将本文算法和 AFSA 各自独立运行 30 次, 并对所得函数模型的相对误差进行统计分析. 采用盒图表示的形式表征数据的离散程度, 图 9 为两种算法的相对误差的盒图, 本文算法与 AFSA 中的适应度值均依据式 (3) 计算.

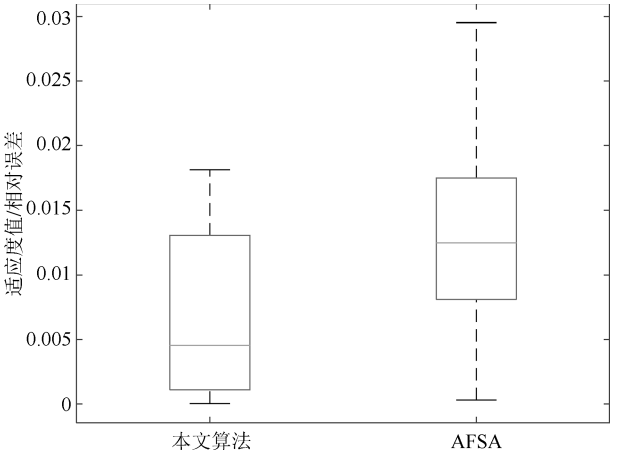


图 9 相对误差分布比较

Fig. 9 Comparison of the relative error distribution

通过数据分布下界、下四分位数、中位数、上四分位数、上界等 5 个指标描述; 盒框的上、下边即上、下四分位数, 盒框内的横线即中位数; 数据群体上、下四分位数外的部分构成了盒框外的上、下两条延伸线; 明显偏离数据群体的 “离群点” 则以加号 “+” 表示. 从图中可以看出, 本文算法的结果具有更小的相对误差中位数、更小的相对误差上四分位数及上界. 此外, 本文算法所得结果的上下界之差明显小于 AFSA 所得结果, 表明本文算法具有比 AFSA 更好的收敛性.

3.3 算子性能

为了验证本文算法中散开算子和突变算子的有效性, 分别将有无这些算子的算法运行结果进行比较. 采用的函数模型如式 (10)[18] 所示, 算法的相关测试参数为: 迭代次数 5 000, 种群数量 100, 首段长度 22, 约束长度 30, 其余与表 8 所列参数相同. 输入观测数据是从区间 [1, 10] 均匀选择的 10 个数据, 输出观测数据根据式 (10) 分别代入输入观测数据得到.

$$y = 3(x+1)^3 + 2(x+1)^2 + x + 1 \tag{10}$$

各算法分别独立运行 50 次, 种群最优粒子的适应度平均值的变化如图 10 所示. 根据粒子评价函数, 适应度值越小粒子越优. 由于突变算子使算法具有跳出局部极值的可能性, 散开算子能减轻粒子快速趋同对全局寻优造成的不利影响, 本文算法能达到较好的性能. 从图 10 中可以看出, 当从算法中删除相关算子时, 算法性能出现了不同程度下降, 其中, 同时删除突变算子和散开算子的算法其性能大幅退化; 只删除散开算子的算法其性能亦出现了明

显的下降; 当只删除突变算子时, 算法性能有小幅弱化. 这些现象表明了散开算子、突变算子在算法中的重要作用.

下面通过比较有无散开算子这两种情况下的算法耗时来考察粒子散开算子的时间开销, 测试函数模型仍然采用式 (10). 两种算法分别独立运行 50 次, 以 5 000 次迭代作为终止条件. 有粒子散开算子的算法平均值和方差分别为 1.7379 s 和 0.0132, 无粒子散开算子的算法平均值和方差分别为 1.4362 s 和 0.0319, 前者的平均值比后者仅增加 0.3017 s, 显然, 时间开销上的增加很小, 但换取了求解精度上的较大提高. 此外, 有粒子散开算子算法的方差小于无粒子散开算子算法所得结果, 可以表明有粒子散开算子的算法具有更小的分散性.

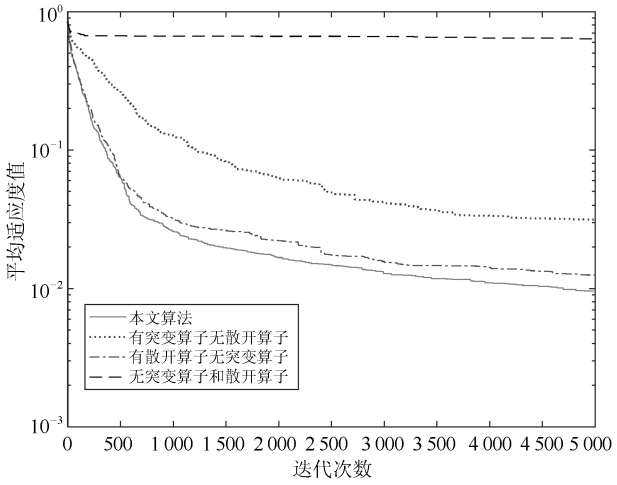


图 10　具有不同算子的算法迭代曲线比较

Fig. 10　Iterative curve comparison of algorithms with different operators

本文算法所具有的性能得益于 PSO 高效而简洁的内在寻优机理, 也得益于 r-邻域粒子种群拓扑、散开算子、突变算子等. 事实上, 还有其他研究者也在符号回归中使用了 PSO[19−20]. 由于文献 [19] 没有给出具体指标无法比较, 本文以文献 [20] 提出的算法 TSO 进行比较. TSO 的参数和测试算例也取自文献 [20]. 表 9 中的 F1、F2、F3 的函数形式见表 5, F11 为 $y(x) = x^5 - 2x^3 + x$, 其中 x 取 $[-1, 1]$ 上随机的 20 个点. 两种算法的适应度值仍按式 (5) 计算.

从表 9 可以看出, 在平均适应度值 (Average fitness), 标准差 (Standard deviation), 测试 100 次成功找到原表达式的百分比 (Perfect hits in %), 和找到原表达式所需要的平均评价次数 (Number of evaluations) 这四项指标中, 本文算法仅在 F1 的前三项指标上与 TSO 相同, 其余的均优于 TSO. 这表明了在 PSO 算法框架中引入的 r-邻域粒子种群拓扑、散开算子、突变算子等的有效性.

表 9　与文献 [20] 结果的比较

Table 9　Comparison with results in reference [20]

算法	评价指标	F1	F2	F3	F11
TSO	Average fitness	0	0.09	0.63	0.26
	Standard deviation	0	0.35	0.79	0.36
	Perfect hits in %	100	92	57	52
	Number of evaluations	1 220	4 660	7 000	9 020
本文算法	Average fitness	0	0	0.04	0.23
	Standard deviation	0	0	0.26	0.30
	Perfect hits in %	100	100	97	73
	Number of evaluations	692	1 589	6 645	8 350

3.4　函数模型的简洁性

为了得到简洁的函数模型, 文献 [27] 将编码的有效长度作为简洁性评价标准, 即编码的有效长度越短, 表明模型越简洁. 本实验考察不同约束长度对编码的有效长度进行约束的效果. 待建模型的输入、输出观测数据 (x_i, y_i) 如表 10 所示[18], 本文算法的相关测试参数将表 8 中的种群数量改为 100, 首段长度改为 20, 终止符集改为 x. 函数及运算符集改为 $+$, $-$, $\times$, $/$, $\sqrt{\ }$, sin, cos, ln, e^x, 其余参数相同. 不同约束长度演化实验所得数据如表 11 所示.

表 10　观测数据

Table 10　Observation data

x	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
y	0.250841, −0.373517, −1.957016, −2.699129, 1.619885 4.684452, 2.686799, −0.501882, −4.718648, −7.787834 0.056755, 9.363730, 6.761007, 0.824883, −5.821076 −12.512195, −5.349206, 12.234454, 12.324229, 3.656108

本文对编码的有效长度进行约束, 实验中使用的约束长度如表 11 的第 1 列所示, 演化得到的编码如表 11 的第 3 列所示, 该编码的有效长度如表 11 的第 2 列所示, 对应的模型表达式如表 11 的第 4 列所示. 所构建的 5 个模型的拟合误差均小于 0.04. 从表中可以看出, 当约束长度值在一定范围 (上边界为编码的总长度) 内变小时, 获得的编码的有效长度也相应地变短, 从而可获得更加简洁的函数模型.

3.5　算法应用

将本文的算法应用于对日本人口数据进行建模并预测其变化趋势, 数据来源为日本国立社会保障和人口问题研究所 2017 年报告[31]. 该报告给出了日本人口从 1960 年至 2015 年每年的人口数量统计

表 11 不同约束长度下的函数模型

Table 11 Function models with different constrained length

约束长度	有效长度	编码	模型表达式
25	24	$\times, \cos, \times, x, /, \cos, x, \sqrt{\ }, \cos, \exp, \cos, \sin, \sin, x, \sqrt{\ }, \cos,$ $/, -, \times, \sqrt{\ }, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x$	$\frac{x \cdot \cos x}{\sqrt{e^{\sin x}}} \cdot \cos(\cos(\cos(\sin(\sqrt{\cos(\frac{\sqrt{x}-x}{x^2})}))))$
22	21	$\times, \cos, /, x, x, \sqrt{\ }, \exp, +, \sin, \sin, \sin, x, \cos, /, \ln, \cos, \sqrt{\ },$ $x, \exp, \cos, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x$	$\frac{x \cdot \cos x}{\sqrt{\exp(\sin x + \sin(\sin(\cos(\frac{\ln \sqrt{e^{\cos x}}}{\cos x}))))}}$
20	19	$\times, /, x, \cos, \sqrt{\ }, x, \exp, +, \cos, \sin, \sin, x, \sqrt{\ }, \cos, /, x, +,$ $/x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x$	$\frac{x \cdot \cos x}{\sqrt{\exp(\sin x + \cos(\sin(\sqrt{\cos(\frac{x}{2x})})))}}$
18	18	$/, \cos, /, x, \sqrt{\ }, x, +, \exp, -, \sin, \exp, \ln, x, \sin, /, x, x,$ $x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x$	$\frac{x \cdot \cos x}{\sqrt{e^{\sin x} + e^{\sin x} - \ln(\frac{x}{x})}}$
15	13	$/, \cos, /, x, \sqrt{\ }, x, +, \exp, \exp, \sin, \sin, x, x, x, x, x, \times,$ $x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x, x$	$\frac{x \cdot \cos x}{\sqrt{e^{\sin x} + e^{\sin x}}}$

值共 56 个, 和对未来人口数量的预测. 其中 5 个预测值是通过对各个地区的生育率, 死亡率等多种详细数据, 用社会学的研究方法得到的. 我们将 56 个历史数据作为算法的训练样本, 得到的模型的曲线如图 11 所示, 本模型的 5 个预测值和文献 [31] 的预测值见表 12. 算法参数为, 将表 8 中的约束长度改为 40, 以 10 000 次迭代作为终止条件, 其余参数不变.

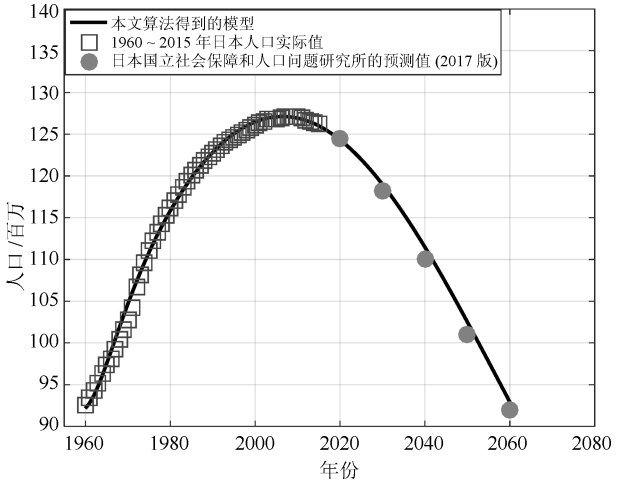


图 11 日本人口预测模型

Fig. 11 Model of Japanese population prediction

从图 11 和表 12 可以看出, 在 5 个预测值中, 对 2020 年的人口预测二者得到的值最接近, 其余的预测值尽管不一致的程度较大, 但预测的趋势基本一致. 这表明本文算法可以不需要有关社会学的先验知识而仅根据已知数据通过符号回归建立函数模型并做预测.

表 12 预测值 (百万)

Table 12 Predicted values (million)

年份	文献 [31]	本文算法	二者之差
2020	125.325	125.238	0.087
2030	119.125	119.902	0.777
2040	110.919	112.385	1.466
2050	101.923	103.425	1.502
2060	92.840	93.845	1.005

4 结束语

符号回归方法的研究在实际中具有重要的意义. GP 和 GEP 开创了符号回归的方法基础, 它们及后续的改进方法大多数是以优胜劣汰的竞争机制为核心的. 本文从群体智能的学习角度出发, 针对符号回归的特点, 根据粒子群优化算法的基本原理, 提出了一种基于粒子群优化算法的符号回归方法, 通过设计粒子的飞行学习机制, 提高了粒子在搜索空间的学习协作能力, 进而较大幅度地提高了符号回归的性能. 多项数值实验的结果表明, 本文提出的算法可以获得拟合精度更高、简洁性更好的函数模型, 具有比较广泛的应用价值.

References

1 Liu Qiang, Qin S. Joe. Perspectives on big data modeling of process industries. *Acta Automatica Sinica*, 2016, **42**(2): 161−171
(刘强, 秦泗钊. 过程工业大数据建模研究展望. 自动化学报, 2016, **42**(2): 161-171)

2 Si Xiao-Sheng, Hu Chang-Hua, Zhou Dong-Hua. Nonlinear degradation process modeling and remaining useful life estimation subject to measurement error. *Acta Automatica Sinica*, 2013, **39**(5): 530−541
(司小胜, 胡昌华, 周东华. 带测量误差的非线性退化过程建模与剩余寿命估计. 自动化学报, 2013, **39**(5): 530−541)

3 Yao X, Liu Y, Li J, He J. Current developments and future directions of bio-inspired computation and implications for ecoinformatics. *Ecological informatics*, 2006, **1**(1): 9−22

4 Koza J R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: MIT Press, 1992.

5 Ferreira C. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Computer Science*, 2001, **21**(2): 87−129

6 Yassin M A, Alazba A A, Mattar M A. A new predictive model for furrow irrigation infiltration using gene expression programming. *Computers and Electronics in Agriculture*, 2016, **122**: 168−175

7 Mohamed M. Mostafa, Ahmed A. El-Masry. Oil price forecasting using gene expression programming and artificial neural networks. *Economic Modelling*, 2016, **54**(1): 40−53

8 Weatheritt J, Sandberg R. A novel evolutionary algorithm applied to algebraic modifications of the RANS stress-strain relationship. *Journal of Computational Physics*, 2016, **325**: 22−37

9 Phukoetphim P, Shamseldin A Y, Adams K. Multimodel approach using neural networks and symbolic regression to combine the estimated discharges of rainfall-runoff models. *Journal of Hydrologic Engineering*, 2016, **21**(8): 04016022

10 Razaq S A, Shahid S, Ismail T, et al. Prediction of flow duration curve in ungauged catchments using genetic expression programming. *Procedia Engineering*, 2016, **154**: 1431−1438

11 Soule T. Code growth in genetic programming. In: Proceedings of the 1995 Conference Genetic Programming. Atlanta, GA, USA: ACM, 1995. 148−159

12 Ragalo A W, Pillay N. A building block conservation and extension mechanism for improved performance in polynomial symbolic regression tree-based genetic programming. In: Proceedings of the 2012 Nature and Biologically Inspired Computing. New York, USA: IEEE, 2012. 123−129

13 Sotto L F D P, de Melo V V. Investigation of linear genetic programming techniques for symbolic regression. In: Proceedings of the 2014 Brazilian Conference on Intelligent Systems. New York, USA: IEEE, 2014. 146−151

14 Peng Y Z, Yuan C A, Qin X, et al. An improved gene expression programming approach for symbolic regression problems. *Neurocomputing*, 2014, **137**(15): 293−301

15 Zhong J, Ong Y S, Cai W. Self-learning gene expression programming. *IEEE Transactions on Evolutionary Computation*, 2016, **20**(1): 65−80

16 Olmo J L, Romero J R, Ventura S. Swarm-based metaheuristics in automatic programming: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2014, **4**(6): 445−469

17 Karaboga D, Ozturk C, Karaboga N, et al. Artificial bee colony programming for symbolic regression. *Information Sciences*, 2012, **209**: 1−15

18 Liu Q, Odaka T, Kuroiwa J, et al. Application of an artificial fish swarm algorithm in symbolic regression. *IEICE Transactions on Information & Systems*, 2013, **E96. D**(4): 872−885

19 Qi F, Ma Y H, Liu X Y, Ji G Y. A hybrid genetic programming with particle swarm optimization. In: Proceedings of the 2013 International Conference in Swarm Intelligence. Berlin, Germany: Springer-Verlag, 2013. 11−18

20 Veenhuis C, Koppen M, Kruger J, Nickolay B. Tree swarm optimization: An approach to PSO-based tree discovery. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation. New York, USA: IEEE, 2005. 1238−1245

21 Kennedy J, Eberhart R C. Particle swam optimization. In: Proceedings of the 1995 Neural Networks. *Picataupy*, USA: IEEE, 1995. 1942−1948

22 Zhang S, Xu J, Lee L H, et al. Optimal computing budget allocation for particle swarm optimization in stochastic optimization. *IEEE Transactions on Evolutionary Computation*, 2017, **21**(2): 206−219

23 Wu H Y, Nie C H, Kuo F C, Leung H, Colbourn C J. A discrete particle swarm optimization for covering array generation. *IEEE Transactions on Evolutionary Computation*, 2015, **19**(4): 575−591

24 Gong Y J, Zhang J, Chung H S H, et al. An efficient resource allocation scheme using particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 2012, **16**(6): 801−816

25 Lee K B, Kim J H. Multiobjective particle swarm optimization with preference-based sort and its application to path following footstep optimization for humanoid robots. *IEEE Transactions on Evolutionary Computation*, 2013, **17**(6): 755−766

26 Thongchart Kerdphol, Kiyotaka Fuji, Yasunori Mitani, Yaser Qudaih. Optimization of a battery energy storage system using particle swarm optimization for stand-alone microgrid. *International Journal of Electrical Power & Energy Systems*, 2016, **81**: 32−39

27 Iba H, Garis H D, Sato T. Genetic programming using a minimum description length principle. *Advances in Genetic Programming*, **1994**: 265−284

28 Liu Q F, Wei W H, Yuan H Q, Li Y. Topology selection for particle swarm optimization. *Information Sciences*, 2016, **363**: 154−173

29 Zhou Xiao-Jun, Yang Chun-Hua, Gui Wei-Hua, Dong Tian-Xue. A particle swarm optimization algorithm with variable random functions and mutation. *Acta Automatica Sinica*, 2014, **40**(7): 1339−1347
(周晓君, 阳春华, 桂卫华, 董天雪. 带可变随机函数和变异算子的粒子群优化算法. 自动化学报, 2014, **40**(7): 1339−1347)

30 Uy N Q, Hoai N X, O′Neill M, et al. Semantically-based crossover in genetic programming: Application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 2011, **12**(2): 91−119

31 Population Statistics of Japan 2017 [Online], available: http:// www.ipss.go.jp/p-info/e/psj2017/PSJ2017.asp, October 9, 2018

马 炫 西安理工大学自动化与信息工程学院教授. 2002 年获得日本福井大学系统工程专业博士学位. 主要研究方向为进化计算, 机器学习, 数据驱动建模. 本文通信作者.
E-mail: maxuan@xaut.edu.cn
(**MA Xuan** Professor at the School of Automation and Information Engineering, Xi′an University of Technology. He received his Ph. D. degree in systems engineering from University of Fukui, Japan, in 2002. His research interest covers evolutionary computation, machine learning, and data-driven modeling. Corresponding author of this paper.)

李 星 西安理工大学自动化与信息工程学院硕士研究生. 主要研究方向为数据驱动建模, 机器学习.
E-mail: peagle4@126.com
(**LI Xing** Master student at the School of Automation and Information Engineering, Xi′an University of Technology. His research interest covers data-driven modeling, machine learning.)

唐荣俊 西安理工大学自动化与信息工程学院研究生. 主要研究方向为数据驱动建模. E-mail: tangrj_xian@126.com
(**TANG Rong-Jun** Master student at the School of Automation and Information Engineering, Xi′an University of Technology. His main research interest is data-driven modeling.)

刘 庆 博士, 西安理工大学自动化与信息工程学院讲师. 主要研究方向为数据驱动建模, 网络路由优化, 群智能算法.
E-mail: liuqing@xaut.edu.cn
(**LIU Qing** Ph. D., lecturer at the School of automation and Information Engineering, Xi′an University of technology. His research interest covers data-driven modeling, network routing optimization, and swarm intelligence algorithms.)