基于决策树求解非线性规划问题的算法

杨昌霖 [1]
YANG Changlin

摘 要 针对群体智能优化算法具有随机性、盲目性、可编程性差的问题，提出了一种简单有效且不具有随机性的全局搜索算法，用于求解非线性规划问题。通过对每个决策变量的可行域离散化处理后得到的数据构建决策树，采用深度优先的规则对最优解进行搜索，搜索的同时用指数衰减函数调整搜索步长，从而逐步缩小搜索范围，直到结果收敛。算法不具有随机性，不需要编码、解码、交叉、变异等复杂操作，也不需要随机生成初始种群，可编程性强。对非线性规划的六个测试函数进行求解，并与文献中报道的结果对比，结果表明基于决策树的遍历搜索对解决非线性规划问题有效，对于多决策变量的复杂优化问题，采用分组搜索的策略既能保证求解精度，也能保证收敛速度。

关键词 非线性规划；决策树；搜索算法；最优化

doi：10.3969/j.issn.1672-9528.2024.03.023

0 引言

非线性规划问题（nonlinear programming problem，NLP）是运筹学的一个重要分支，这类问题需要求解 n 元非线性函数在一组等式或不等式约束条件下的极值问题，广泛存在于生产管理、质量控制、产品设计等领域，也是实现人工智能和最优化的基础 [1]。非线性规划问题的解决方法有两类：第一类是基于梯度信息进行搜索的算法，如中心法、梯度投影法、惩罚函数法、乘子法等，这类函数往往要求目标函数和约束函数必须是可微的，对于优化对象复杂的问题，这类方法只能求得局部最优解 [2]；第二类是全局搜索算法，包括遗传算法 [3]、粒子群算法 [4] 和各类演化算法 [5]。这类算法属于群体智能优化算法，无需目标函数连续可微的限制，具有并行性和全局搜索能力，但是这类算法容易出现早熟收敛，存在后期寻优能力差的缺陷。

为了解决第二类算法存在的诸多问题，研究者对遗传算法、粒子群算法和各种演化算法做了很多改进，包括遗传算法的进化策略、交叉率及变异率的调节、粒子群初始种群的演化、粒子群惯性权重的调整 [6-9]。从众多的研究成果中不难发现，改进后的全局搜索算法无论是解的精度，还是收敛速度，都得到了令人满意的结果。然而，在实际应用过程中仍然发现以下两方面的问题：一是计算结果具有随机性，对于决策变量超过 20 个的复杂优化问题，需要运行多次才能得到最优解，且每次得到的结果存在较大差异 [10-11]；二是算法中涉及的交叉、变异、种群进化等操作需要借助软件提供的库函数实现 [12]。由于软件自带的函数没有在文中予以说明，一些算法的准确性很难得到验证，仿真结果的重现性差。因此，提出一种简单有效、易于编程和验证的搜索算法，对解决非线性规划问题具有重要意义。

本文通过构建决策树描述非线性规划问题的可行解，基于深度优先的规则遍历搜索树，对最优解进行搜索，在搜索的同时对决策变量的搜索区间进行自适应调整，以提高收敛速度。算法无需编码、解码、交叉、变异等复杂操作，也不需要随机产生初始种群，结果不具有随机性，且编程容易实现。采用 Keane's Bump 问题对算法进行测试，得到了令人满意的结果，证明基于决策树的遍历搜索算法对求解非线性规划问题有效。

1 非线性规划问题

非线性规划问题的数学模型可描述为：

$$\begin{cases} \min f(x) \\ s.t.g_i(x) \le 0, i = 1,2,\cdots,m \\ \quad h_j(x) = 0, j = 1,2,\cdots,n \end{cases} \tag{1}$$

式中：$f(x)$ 为目标函数，$g_i(x)$ 和 $h_j(x)$ 分别是不等式和等式约束，且至少有一个是非线性函数。$\boldsymbol{x}=\{x_1,x_2,\cdots,x_n\} \in R_n$ 为决策变量，$F=\{x|g_i(x) \le 0,\ h_j(x)=0\}$ 为可行域，若存在 $x^* \in F$ 使得对任意 $x \in F$ 都有 $f(x^*) \le f(x)$，则 x* 是 $f(x)$ 在可行域 F 上的最优解，$f(x^*)$ 为最优值。

1. 中国航空工业集团公司西安航空计算技术研究所
陕西西安 710064

2 算法实现

2.1 可行域离散化处理

根据式（1），任意一个决策变量的约束条件可以表示为：

$$l_i \le x_i \le u_i \tag{2}$$

假设 $[l_i, u_i]$ 区间内待搜索点的数量为 s_i，则决策变量 x_i 的搜索步长为：

$$d_i = \frac{u_i - l_i}{s_i - 1} \tag{3}$$

将 x_i 所对应的可行域 F 离散为 s_i 个数据组成的集合，其中任意一个点可以表示为：

$$p_{i,j} = l_i + d_i \cdot j,\ j = 0,1,2,\cdots,s_i - 1 \tag{4}$$

2.2 构建决策树

将每一个决策变量对应的可行域中的所有点用决策树的结构来描述。假设两个决策变量中给定其中一个变量 x_1 的值 p_1，另一个决策变量 x_2 可行域所对应的离散点构成的集合表示为 $\{p_{2,1}, p_{2,2}, \cdots, p_{2,s2}\}$，则根节点为 p_1，叶节点为 x_2 对应的点集，构建决策树为图 1。

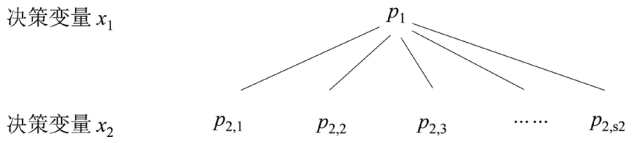

图 1 两个决策变量构建的决策树

图 1 为决策树的基本单元，显然，对于图 1 所示的情况，通过遍历 x_2 可行域中的所有点即可找到最优解。搜索顺序为：$(p_1, p_{2,1})$，$(p_1, p_{2,2})$，$(p_1, p_{2,3})$，…，$(p_1, p_{2,s2})$ 将两个决策变量的情况推广至 n 个决策变量的情况，可构建决策树如图 2 所示。

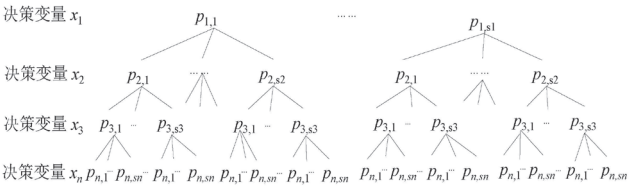

图 2 n 个决策变量构成的决策树

按照深度优先的遍历规则，即从最顶层的节点向下逐步检索，假如该节点存在叶节点，则优先遍历叶节点，假如该节点所属的叶节点全部遍历完成，则遍历该节点的兄弟节点。

2.3 构建决策树

搜索过程从最顶层至最底层逐步展开，决策树中的每一层对应一个决策变量，每个决策变量 x_i 的可行域由 s_i 个数据构成，搜索过程中记录下每一个决策变量已完成的搜索点的个数 t_i，通过判断搜索点的个数是否达到 s_i 保证搜索过程持续进行，具体过程如下。

（1）将每个决策变量所对应的已完成的搜索点的数量 t_i 的初始值设为 0，当前层设置为第一层。

（2）提取当前层对应的决策变量的 t_i 值，判断 t_i 是否等于 s_i，如果 $t_i<s_i$，则表明可行域中存在待搜索的数据点，如果 $t_i = s_i$，转步骤（4）。

（3）将可行域中第 t_i 个数据点提取到 sequence 数组中，如果当前当前层为最后一层，转步骤（6），如果当前层不是最后一层，则层序号加 1，当前层设置为第 i+1 层，开始提取 i+1 层的待搜索数据，回到步骤（2）。

（4）如果 $t_i = s_i$，则表明可行域中的所有数据点均已搜索完成。

（5）将该层对应的 t_i 设为 0，第 i-1 层已完成搜索的点的数量 t_{i-1} 加 1，当前层设置为 i-1 层，转步骤（7）。

（6）对于步骤（2）中 $t_i<s_i$ 的情况，如果当前层等于最后一层，则表明一组可行解已经产生，且数据已经存放在 sequence 中，此时需要计算目标函数的值并与历史数据进行对比，如果为历史最优值，则将最优值保存到 fmin 中。判断完成后，当前层为最后一层，t_i 值加 1，回到步骤（2）。

（7）步骤（5）的执行意味着第 i 层至最后一层中的任意一层的 t_i 值均为 0，此时需要更新根节点开启下一轮搜索，如果 i<0，表明根节点已无法更新，所有决策变量可行域中的点均已参与了搜索过程，程序结束，如果 i≥0，用新的根节点开启下一轮搜索，回到步骤（2）。

按照上述规则可以保证每一轮搜索得到的解不会重复，程序最终输出的 fmin 即是搜索得到的最优值。

2.4 搜索步长的动态调整

上述算法是一种近似于枚举法的算法，因此是一种全局搜索算法，用式（4）确定的 p_{ij} 建立可行域，则优化结果与理论最优解的误差完全取决于搜索步长 d_i 的大小，显然，d_i 越小，优化结果越准确，但是这样会大大增加计算量。为了同时保证算法的效率与可靠性，本文采用动态调整步长的方法实现在有限个 s_i 中搜索到最优解。搜索步长与迭代次数的关系可表示为：

$$d_{i,q} = d_0 \exp(r \cdot q) \tag{5}$$

式中：d_0 是搜索步长的初始值，q 是迭代次数，r 是搜索步长的衰减系数，值小于 0，影响搜索的收敛速度。

2.5 多决策变量的搜索策略

当决策变量较多时，即便采取动态调整搜索步长的方法也很难提高搜索效率，此时需要采取分组搜索的办法提高收敛速度。将决策变量分成 w 个组，每个组包含的决策变量的数目为 g_k，则决策变量总数 n 等于所有组的决策变量之和：

$$n = \sum_{k=1}^{k=w} g_k \tag{6}$$

对于任意一组决策变量，用 i_m 表示第 k 组决策变量的起始序号，i_n 表示第 k 组决策变量的末尾序号，则：

$$i_m = \sum_{k=1}^{k-1} g_k + 1 \tag{7}$$

$$i_n = \sum_{k=1}^{k} g_k \tag{8}$$

采用分组的方式组织搜索时，每一次迭代只针对一个组中的决策变量进行搜索，其他组的决策变量的值为固定值。每一次迭代所针对的变量组不同，即待搜索变量组 k 的取值与迭代次数 q 有关：

$$k = q\%w + 1 \tag{9}$$

“%”表示取余数。当针对第 k 组决策变量进行搜索时，s_i 的取值为给定值，其余组决策变量的 s_i 等于 1：

$$s_i = \begin{cases} s_i & i \in [i_m, i_n] \\ 1 & i \notin [i_m, i_n] \end{cases} \tag{10}$$

分组搜索的计算过程如下。

（1）令迭代次数 q 为 0，设定初始值，包括组的个数 w，每个组的决策变量的数量 g_k，每个决策变量对应可行域中的点的个数 s_i，每个决策变量的初始上限 l_i，每个决策变量的初始下限 u_i 以及控制搜索步长衰减速度的系数 r。

（2）根据式（3）计算搜索步长的初始值 d_0。

（3）根据式（9）计算待搜索的组的序号 k。

（4）根据式（7）和式（8）计算第 k 组决策变量的初始序号和末尾序号。

（5）根据式（10）重新计算每个决策变量对应可行域中点的数量 s_i。

（6）根据式（4）得到决策变量可行域中离散点的值。

（7）用给出的深度优先遍历算法计算最优值。

（8）判断迭代是否收敛，如果收敛程序结束，如果不收敛，迭代次数加 1。

（9） 根据式（5）计算第 q 次迭代的搜索步长 $d_{i,q}$，返回步骤（3）。

3 计算实例

采用文献 [3] 中提供的五个测试函数以及 Keane's Bump 函数对算法的可靠性进行验证。其中，$G_1(x)$ 是区域约束优化问题，$G_2(x)$ 是不等型约束优化问题，$G_3(x)$ 是多峰问题，$G_4(x)$ 是可行空间非凸非线性问题，$G_5(x)$ 是惩罚方法难问题，$G_6(x)$ 是 Keane's Bump 问题，具有三超特性（超非线性、超多峰、超高维），已成为国际上通用的衡量优化算法的测试函数。G_1(x) ～ $G_5(x)$ 取最小值，$G_6(x)$ 取最大值。本文以 n=20 的 Bump 函数作为测试函数，函数表达式为：

$$G_6(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{11}$$

受限于：

$$\begin{cases} 0 \le x_i \le 1, i = 1,2,\cdots n \\ \prod_{i=1}^{n} x_i \ge 0.75 \\ \sum_{i=1}^{n} x_i \le 7.5n \end{cases} \tag{12}$$

求解过程中 s_i、d_0 和 r 的取值见表 1。

表 1 求解过程中用到的参数

目标函数	参数		
	s_i	d_0	r
$G_1(x)$	6	2	-0.006 734
$G_2(x)$	8	2	-0.010 101
$G_3(x)$	20	0.005	-0.006 734
$G_4(x)$	10	1	-0.010 101
$G_5(x)$	8	2	-0.006 734
$G_6(x)$	9	2	-0.020 203

对于 $G_6(x)$，采用分组搜索的方法，每组对应的决策变量的数量见表 2。对于 $G_1(x)$ ～ $G_5(x)$，本文算法与其他算法所得到的目标函数的最优值见表 3。$G_1(x)$ ～ $G_5(x)$ 的最优解见表 4。对于 $G_6(x)$，本文算法与其它算法所得到的目标函数的最优值见表 5，本文算法得到的最优解见表 6。

表 2 $G_6(x)$ 中 20 个决策变量的分组情况

k	1	2	3
g_k	7	7	6

表 3 $G_1(x)$ ～ $G_5(x)$ 目标函数最优解计算结果对比

目标函数	算法名称					
	解类型	遍历搜索	GA	IFGA	PSO	Pareto 演化
$G_1(x)$	最优值	30 665.539	-30 665.5	-30 665.539	-30 664.7	—
	最差值	—	-30 236.6	-30 665.4432	-30 656.1	—
	平均值	—	-30 533.8	-30 665.486 08	-30 662.8	—
$G_2(x)$	最优值	24.313 23	24.856	24.306 209 02	24.808 18	24.306 209 1
	最差值	—	26.11	24.306 209 1	26.182 73	24.362 999 9
	平均值	—	25.237	24.306 209 09	25.408 52	24.325 487 7
$G_3(x)$	最优值	0.095 825	0.095 825	—	0.095 825	—
	最差值	—	0.095 825	—	0.095 825	—
	平均值	—	0.095 825	—	0.095 825	—
$G_4(x)$	最优值	680.630 060	680.67	680.630 063	680.675	680.630 1
	最差值	—	683.74	—	680.904 7	—
	平均值	—	681.62	—	680.786 7	—
$G_5(x)$	最优值	7 049.857 6	7115	—	7 188.84	7 049.248 021
	最差值	—	7 384.29	—	7 225.98	7 058.235 358
	平均值	—	7 795.93	—	7 222.39	7 051.287 429

表 4 本文算法得到的 $G_1(x)$ ～ $G_5(x)$ 决策变量最优解计算结果

目标函数	x_1	x_2	x_3	x_4	x_5
G_1	78.000 000	33.000 000	29.995 256	45.000 000	36.775 813
G_2	2.161 275	2.390 506	8.779 185	5.101 734 48	0.995 316 56
G_3	1.227 973	4.245 08	—		—
G_4	2.330 756	1.951 53	-0.477 199	4.365 240	-0.624 449
G_5	549.923	1 361.300 43	5 138.633 437	179.511 532	294.454 663
目标函数	x_6	x_7	x_8	x_9	x_{10}
G_1	—	—	—	—	—
G_2	1.429 575 00	1.300 701 32	9.811 608 22	8.258 968 19	8.392 825 36
G_3	—	—	—	—	—
G_4	1.037 988	1.594 356	—	—	—
G_5	220.488 468	285.056 869	394.454 663	—	—

表 5 $G_6(x)$ 目标函数最优解计算结果对比

最优解类型	算法名称			
	本文算法	HPSO 混合粒子群	FPDC 遗传算法	PSODE 粒子群算法
最优值	0.803 619 1	0.803 502 8	0.802 747	0.803 618 6
最差值	—	0.800	—	0.803 604

表 6 本文算法得到的 $G_6(x)$ 中 20 个决策变量的最优解

x_1	x_2	x_3	x_4	x_5
3.162 338 06	3.128 245 60	3.094 736 81	3.061 425 22	3.027 928 24
x_6	x_7	x_8	x_9	x_{10}
2.993 840 05	2.958 710 65	2.921 895 19	0.494 973 57	0.488 511 802
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
0.482 474 411	0.476 802 353	0.471 403 474	0.466 279 336	0.461 409 682
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}
0.456 767 563	0.452 345 596	0.448 120 5	0.444 076 158	0.440 200 266

4 结果分析

从表 1 中可以看出，对于 $G_1(x)$ 和 $G_3(x)$，本文提出的算法与遗传算法和粒子群算法都能得到最优结果。对于 $G_2(x)$，遍历搜索算法得到的结果优于 GA 和 PSO 算法给出的最优解，与 Pareto 算法相比，遍历搜索算法得到的结果优于 Pareto 算法的平均值。对于 $G_4(x)$，本文算法得到了已知最优解 680.630 06，但是对于遗传算法、改进遗传算法和粒子群算法，得到的 $G_4(x)$ 的最优解均大于 680.630 06。需要说明的是，文献 [5] 和文献 [6] 中虽然明确指出得到了已知最优解，但实际上，将两篇文献中提供的 $G_4(x)$ 中 7 个决策变量的最优值代入目标函数中得到的结果均大于 680.630 06，表 1 中给出的 IFGA 和 Pareto 的结果是根据文献中给出的决策变量的最优解计算得到的结果。对于 $G_5(x)$，本文算法得到的结果同样优于遗传算法和粒子群算法，与 Pareto 算法相比，遍历搜索算法得到的结果优于 Pareto 算法的平均值。对于 $G_6(x)$，文献中报道的最优结果是 0.803 618 6，本文算法找到了新的最优解 0.803 619 1，对应的 20 个决策变量的值见表 6。

通过上述分析可以发现，本文提出的基于决策树的搜索算法与遗传算法和粒子群算法相比，求解能力更强，即便是与改进的遗传算法和演化算法相比，遍历搜索算法的结果也比这两种算法得到的平均值要好。更重要的是，基于决策树的搜索算法独立运行一次就能得到最优结果，其他几种算法具有随机性，需要独立运行多次才能找到最优结果，因此，本文提出的算法稳定性更好。

5 结论

6 个测试函数的求解结果表明，对于非线性规划问题，采用基于决策树的搜索算法可以得到比遗传算法和粒子群算法更好的结果，尤其是对于多决策变量的规划问题，该方法在求解精度上的优势更为明显。用指数衰减函数调整搜索步长可以有效避免早熟收敛。

参考文献：

[1] 朱会霞，王福林，张勇，等 . 改进遗传算法优化非线性规划问题 [J]. 数学的实践与认识，2013, 43(7): 117-125.

[2] 戴彧虹，刘新为 . 线性与非线性规划算法与理论 [J]. 运筹学学报 ,2014,18(1):69-92.

[3] 谢晓峰，张文俊，阮骏，等 . 针对带约束非线性规划问题的遗传算法 [J]. 计算机工程与应用 ,2002(21):64-66.

[4] 梅海涛，华继学，王毅 . 求解非线性规划问题的改进直觉模糊遗传算法 [J]. 计算机科学，2016, 43(9): 250-254.

[5] 董颖，唐加福，许宝栋，等 . 一种求解非线性规划问题的混合粒子群优化算法 [J]. 东北大学学报 (自然科学版), 2003,24(12):1141-1144.

[6] 周于人，李元香，王勇，等 .Pareto 强度值演化算法求解约束优化问题 [J]. 软件学报，2003, 14(7): 1243-1249.

[7] 朱会霞，李微微，李彤煜，等 . 区间自适应遗传算法优化无约束非线性规划问题 [J]. 数学的认识与实践，2019, 49(4): 110-116.

[8] 唐加福，汪定伟 . 一种求解非线性规划问题的改进遗传算法 [J]. 东北大学学报（自然科学版）,1997,18(5):490-493.

[9] 廖锋，高兴宝 . 求解非线性规划问题的混合粒子群算法 [J]. 计算机工程与应用 ,2008,44(11):43-46.

[10] 林丹，李敏强，寇纪淞 . 基于遗传算法求解约束优化问题的一种算法 [J]. 软件学报 ,2001,12(4):628-632.

[11] 李炳宇，萧蕴诗，吴启迪 . 一种基于粒子群算法求解带约束优化问题的混合算法 [J]. 控制与决策 ,2004,19(7):804-813.

[12] 唐冲 . 基于 Matlab 的非线性规划问题的求解 [J]. 计算机与数字工程，2013,41(7):1100-1103.

【作者简介】

杨昌霖（1992—），男，彝族，云南昆明人，硕士研究生，工程师，研究方向：智能制造。

（收稿日期：2023-12-28）