

模拟退火算法在求解组合优化问题中的应用研究

肖思和¹, 鲁红英¹, 范安东¹, 宋弘²

(1. 成都理工大学网络教育学院, 成都 610059; 2 四川理工学院自动化与电子信息学院, 四川 自贡 643000)

摘要: 文章给出了模拟退火算法求解组合优化问题的实现步骤, 并提出了一种相邻状态的产生函数和迭代方案, 通过实例计算并分析了模拟退火过程的起始温度、终止温度以及降温速度等参数对优化计算的影响。结果表明, 模拟退火算法中各个参数值的选择和设置对运行的结果和效果有较大影响。

关键词: 模拟退火; Metropolis 准则; 组合优化

中图分类号: TP301.6

文献标识码: A

非数值算法是基础学科、工程技术和科学管理等领域中常见的一类计算方法, 如许多解组合优化问题的算法就是典型的非数值算法。由于这些问题本身所固有的计算复杂性, 求其精确解的计算量往往随问题规模呈指数型增长, 以致使用任何高速计算机都需要耗费大量的时间, 甚至根本无法实现。因此, 研究非数值计算的近似算法及其并行实现的途径具有十分重要的意义。在管理科学、计算机科学、大规模集成电路设计及电子工程等领域中, 存在着大量的组合优化问题, 大量的实验^[1-8]结果表明, 模拟退火算法是一种适合解大规模组合优化问题的有效方法, 典型的应用有货郎担问题、调度问题、0-1 背包问题、图着色问题、划分问题和布局问题等^[9]。

本文给出了模拟退火算法求解组合优化问题的实现步骤, 以及该算法在求解过程中需要解决相邻状态的接受准则、起始温度、终止温度、降温速度、某一温度下的迭代方案和相邻状态的产生方法等关键问题, 通过实例计算并分析了模拟退火过程的起始温度、终止温度以及降温速度等参数对优化计算的影响。

1 模拟退火算法求解组合优化问题的步骤

设有如下的组合优化问题^[9]:

$$C(x^*) = \min_{x \in X} C(x)$$

其中 $X = \{x_1, x_2, \dots, x_n\}$ 为参数空间所有可能的解组合 (或状态) 所构成的集合, $C: X \rightarrow R$ 的一个映射, 即目

标函数 (或能量), X^* 是 X 中使 C 最小的最优解。模拟退火法在求解上述问题的具体步骤可描述如下:

Step1: 任选一初始状态初状态 x_0 为当前初始解 $X(0) = x_0$, 并设初始温度为 T_0 , 令 $i = 0$;

Step2 令 $T = T_p$, 以 T 和 x_i 调用如下 Metropolis 抽样算法, 然后返回当前解 $x_i = x_0$

(1) 按某一规定方式根据当前解 x_i 所处的状态, 产生一个近邻子集 $N(x_i) \subset X$, $N(x_i) \neq \emptyset$, 由 $N(x_i)$ 随机产生一个新的状态 x' , 作为当前解的候选解, 且计算 $\Delta C = C(x') - C(x_i)$ 。

(2) 若 $\Delta C < 0$ 则接受 x' , 令 $x_i = x'$, 若 $\Delta C > 0$ 则接受以概率 $\exp(-\Delta C/T)$ 接受 x' 为下一个当前解。若被接受, 同样令 $x_i = x'$, 否则 x' 不变。

(3) 重复上述 (1) 和 (2), 直到给定的收敛准则满足为止 (即达到热平衡)。

Step3 按一定方式将 T 降温, 即 $T = T_{i+1}$, $T_{i+1} < T_p$, $i = i + 1$

Step4 若满足算法终止判断 (即退火过程结束), 是, 以当前解 x_i 为最优解输出, 否, 则转 Step2。模拟退火法的算法如下图 1 所示。

2 模拟退火算法求解组合优化问题的几个关键问题

(1) 相邻状态的接受准则。当 x 从某一状态 x_0 变为新的相邻状态 x' 时, 该新状态是否被接受为新的迭

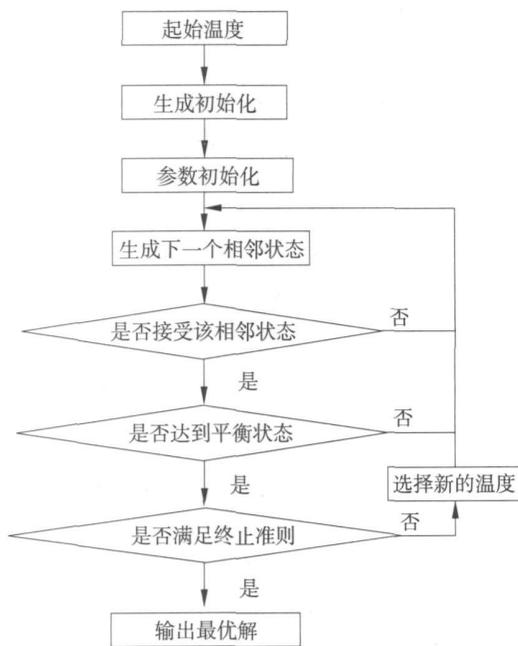


图 1 SA法算法框图

代点, 需要一个判断准则, 这个准则就是前面提到的 Metropolis 准则。我们可以求出 x_0 和 x' 的目标函数 $C(x_0)$ 和 $C(x')$, 如果 $C(x') \leq C(x_0)$, 则目标函数下降, 新状态一定可以接受, 如果目标函数上升, 求 $\rho = \exp(\frac{-cc}{T})$, 这里 $cc = c(x') - c(x_0)$, T 为当前退火温度。产生一个随机数 $\xi \in [0, 1)$, 若 $\rho \geq \xi$ 则接受新状态, 否则, 仍以 x_0 继续迭代。

(2) 起始温度 T_0 。模拟退火的起始温度 T_0 要取得足够大, 使 $cc = c(x') - c(x_0) \rightarrow 1$, 从而能够跳出局部陷阱, 但为了减少计算数量, T_0 也不能取得太大。

(3) 终止温度 T_e 。终止温度 T_e 应该是 $T \rightarrow 0$ 时判断终止。但是由于使 $T = 0$ 需要迭代无穷多次。因此, T_e 为一较小数使 $\exp(\frac{cc}{T_e}) \rightarrow 0$ 收敛到最优解。

(4) 温度的降温速度。 T_k 到 T_{k+1} 的降温速度采用 $T_{k+1} = aT_k$ 的方法。其中 a 为 $0-1$ 间的实数, a 的取值决定了降温的速度, 当 a 较大时, 降温较慢, 反之较快。该值对优化结果有较大的影响。

(5) 某一温度下的迭代方案。从理论上讲, 某一温度下的迭代方案应为无穷多次, 但实际问题中, 在每个温度下考虑迭代若干轮以接近平衡, 这为迭代长度 Length, 每次迭代考虑对一个决策变量 $X(x_1, x_2, \dots, x_n)$ 进行, 则每一轮迭代需要 n 次。

(6) 相邻状态的产生方法。本文通过随机产生方法产生新的相邻状态。假设某一状态下的决策变量 X 的取值为 (x_1, x_2, \dots, x_n) , 它的相邻状态是对某一个 x_i

在一定邻域内进行随机变化产生的值, 产生函数为 $x'_i = x_i \pm \gamma * scale$ 。其中 γ 为 $0-1$ 间的随机数, $scale$ 为自适应邻域因子, 它与温度下降类似: $(scale)_{k+1} = b * (scale)_k$ 。它在迭代开始时应足够大, 而在结束时应较小。 b 为自适应系数, 为 $0-1$ 间的实数。该值对优化计算有较大的影响。

3 实例分析

通过实例验证模拟退火算法的有效性, 对计算结果来分析各参数对优化结果的影响。

例: $\min f(x) = 120x_1 + x_2$

$$s.t. \begin{cases} 4 - x_2 \leq 0 \\ 6.429 - x_1 x_2 \leq 0 \\ 6.429 - x_1^3 x_2 \leq 0 \\ 321.4 - x_1 x_2^2 \leq 0 \end{cases}$$

应用模拟退火法求解该问题。

(1) 当参数分别取 $a = 0.95$, $b = 0.95$, Length = 20, $T_0 = 200.0$, $T_e = 0.0001$, $scale$ 初值为 1000 时, 不论从任何初值点开始均可得到全局最优解, 误差很小 (表 1)。

表 1 SA法求得的计算结果

初值	目标函数	相对误差
(0.7 40)	101.3357	0.021%
(2.0 40)	101.3139	0
(0.5 60)	101.3357	0.021%
(2.5 60)	101.3139	0
(6.0 100)	101.3139	0

(2) 对降温速度与优化结果进行分析。从同一初值 (2.0 40) 开始, 取 $T_0 = 200.0$, $T_e = 0.0001$, $scale = 1000.0$, $b = 0.95$ 对 a 和 Length 的不同值进行了计算。结果见表 2。可以看出, 每一温度下迭代长度较小时, 温度下降速度要求较慢, 而迭代长度大时, 温度下降更快, 说明此两值对优化结果有重要的影响。

表 2 降温速度与迭代长度不同的计算结果

a	k	Length	目标函数
0.98	719	20	110.812
0.95	283	20	101.313
0.90	138	30	101.979
0.80	66	50	136.87
0.70	41	50	150.124
0.50	21	100	240.0

(3) 对于 T_0 和 T_e 的取值也进行了分析。表 3 中变量初值为 (2.0 40), $a = 0.95$, $b = 0.95$, Length = 20, $scale = 1000$ 变化 T_0 时, $T_e = 0.0001$ 变化 T_e 时, $T_0 = 200.0$, 结果表明 T_0 应取足够大, 太小, 就跳不出局部

极小值, T_c 的取值增大时精度下降。

表 3 起始温度和终止温度不同的计算结果

T_0	T_c	目标函数
200.0	0.0001	101.3139
2.0	0.0001	101.3887
0.2	0.0001	101.8339
200.0	0.001	101.338
200.0	0.01	101.4012
200.0	0.1	101.697

(4) 自适应邻域因子 $scale$ 初值及自适应系数 b 的取值对优化计算的影响见表 4。变量初值为 (2, 0, 40), $a = 0.95$, $Length = 20$, $T_0 = 200.0$, $T_c = 0.0001$ 。计算结果表明, 应使 $scale$ 初值容量大些。一般取 $b = a$, 效果较好。

表 4 自适应因子和自适应系数不同的计算结果

$scale$ 初值	b	$scale$ 终值	目标函数
1000.0	0.98	3.2883	101.7568
1000.0	0.95	0.00049	101.3139
1000.0	0.90	0.000485	136.40
100.0	0.98	0.3288	101.3266
100.0	0.95	0.00005	103.190
10000.0	0.95	0.00496	101.621
10000.0	0.92	0.000001	101.3317

4 结束语

通过例子可以看出, 模拟退火算法不需要有较好的近似值作为初值, 有效避免了常规算法对初值的敏感性, 其次算法具有较好的稳定性, 多次实验都能收敛到非常好的结果。模拟退火算法中各个参数值的选择和

设置对运行的结果和效果有较大影响, 对参数进行合理的优化, 才能得到最理想的结果。算法中关于初始温度、温度终止、降温速度、相邻状态的产生函数和迭代方案的确定也是求解组合优化问题重要的环节。

参考文献:

- [1] 段文英, 岳琪. 模拟退火求解组合优化问题的研究 [J]. 森林工程, 2004, 20(4): 26-27.
- [2] 刘怀春, 刘怀亮, 李秀焕, 等. 改进的混合遗传模拟退火算法及其在组合优化中的应用研究 [J]. 现代计算机, 2004, 1: 58-61.
- [3] 赵晶, 唐煊文, 朱动芝. 模拟退火算法的一种改进及其应用研究 [J]. 大连理工大学学报, 2006, 46(5): 775-780.
- [4] 赵天玉. 模拟退火算法及其在组合优化中的应用 [J]. 计算机与现代化, 1999(3): 17-21.
- [5] 顾锦注, 杨佰新. 电力系统机组组合优化的快速模拟退火算法 [J]. 中国电机工程学报, 1992, 12(6): 69-73.
- [6] 宫丽红, 刘则毅, 唐万生. 模拟退火算法在贷款组合优化决策中的应用 [J]. 吉林大学学报: 信息科学版, 2003, 21(2): 143-147.
- [7] 唐茂, 罗阳, 刘胜青. 作业车间调度问题中的协同优化算法 [J]. 中国测试技术, 2003, 29(1): 43-45.
- [8] 谢秉磊, 李良. 求解配送收集旅行商问题的模拟退火算法 [J]. 系统工程理论方法应用, 2002, 11(3): 240-243.
- [9] 康立山, 谢云, 尤夫勇, 等. 非数值并行算法—模拟退火算法 [M]. 北京: 科学出版社, 1994.

Applied Research of Simulated Annealing Algorithm on Solving Combinatorial Optimization Problems

XIAO Sihe¹, LU Hongying¹, FAN An-dong¹, SONG Hong²

(1. College of Network Education, Chengdu University of Technology, Chengdu 610059, China

2. School of Automation and Electronic Information Engineering, Sichuan University of Science & Engineering, Zigong 643000, China)

Abstract The simulated annealing algorithm is an effective method to solve combinatorial optimization problems, this paper brought about the realizing steps. A function of producing adjacent state is proposed, and effects of initial temperature, terminal temperature as well as annealing schedule to the optimal calculation are analyzed. The results showed that how to set the parameters has a greater influence on the results and effectiveness.

Key words simulated annealing algorithm; Metropolis; combinatorial optimization