**模拟退火算法**

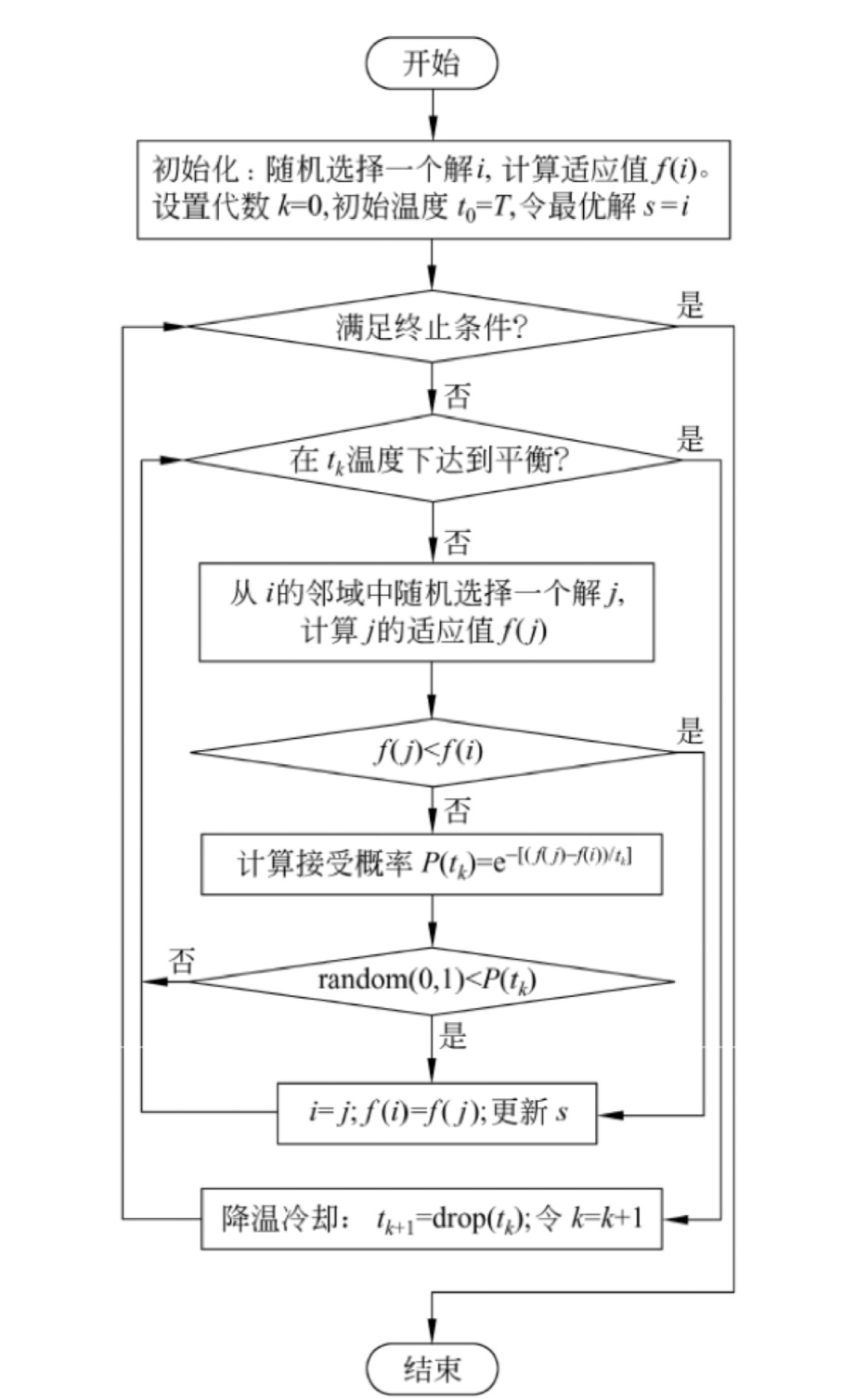
1. 算法思想

通过模仿高温固体退火原理将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温度升高变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。

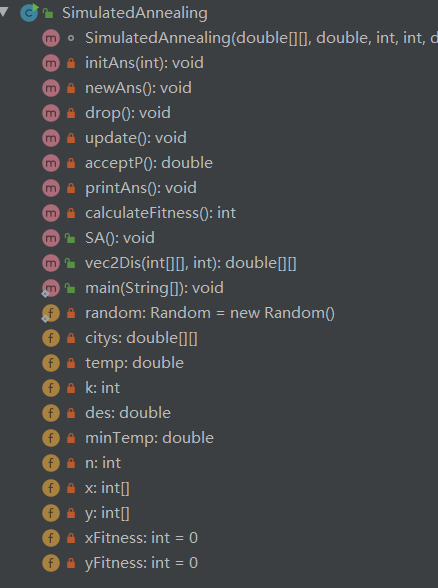
该算法在系统向着能量减小的趋势变化过程中，偶尔允许系统跳到能量较高的状态，以避开局部最小，最终稳定在全局最小。

模拟统计物理学中物体加热后冷却这一退火过程而建立的随机优化算法，意图是避免陷入局部极小解，早期用于组合优化，后来发展成一种通用的优化算法。

1. 算法设计



1. 代码设计



public void SA() {

initAns(n);

int times = 0;

while (temp > minTemp) {

for (int i = 0; i < k; i++) {

newAns();

yFitness = calculateFitness();

if (yFitness <= xFitness)

update();

else {

if (random.nextDouble() <= acceptP())

update();

}

}

drop();

times++;

}

printAns();

System.out.println("路径长度" + yFitness);

System.out.println("进行" + times + "降温");

}

1. 源代码

见附件

1. 实验测试（包括测试数据、测试结果、结果分析等）

城市临接矩阵 {{-1, 9, 19, 13, 15, 98, 56, 12, 54, 65},

{21, -1, 7, 14, 8, 54, 5, 34, 6, 7},

{1, 40, -1, 17, 25, 5, 79, 54, 2, 35},

{41, 80, 10, -1, 54, 23, 98, 54, 21, 4},

{25, 42, 12, 7, -1, 23, 91, 54, 24, 88},

{7, 21, 65, 78, 45, -1, 98, 6, 78, 12},

{45, 78, 57, 12, 52, 13, -1, 54, 1, 64},

{51, 85, 35, 11, 57, 83, 91, -1, 21, 44},

{47, 30, 33, 15, 54, 27, 18, 51, -1, 5},

{3, 21, 14, 35, 4, 35, 8, 53, 21, -1},};

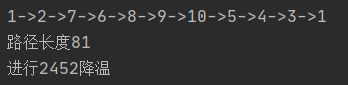
初始温度50000.0

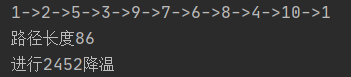
城市个数 10

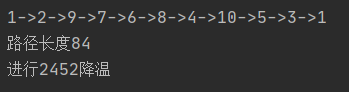
内部平衡循环代数 50

降温系数0.98

最小温度1E-8







经过多次重复进行之后仍可发现比先前好的解，说明计算得到的不是全局最优解，模拟退火只能说是可以在一定程度上跳出局部最优，但是仍不能避免局部最优的情况。

1. 结论

在计算前期还会依概率接受交叉的解，但当迭代次数逐渐增加之后，概率就逐渐的逼向于0，导致除了比当前解好的解概不接受，这也是他为什么仍可能存在局部最优的情况。