现代优化计算方法

（仅限现代优化方法课程学生使用）

邢文训，谢金星

清华大学数学科学系

第 1 章 概　论

本书讨论的现代优化计算方法，主要包括禁忌搜索(tabu search)算法、模拟退火(simulated annealing)算法、遗传算法(genetic algorithms)、蚁群优化算法(ant colony optimization algorithm)和人工神经网络 (artificial neural networks) 算法等。这些算法借助现代计算机作为工具，对复杂的组合最优化问题的求解具有普遍适用性，在 20 世纪 80 年代以来得到了快速发展和广泛应用，所以我们称它们为现代优化计算方法。这类算法中的每一个算法都以人类、生物的行为方式或物质的运动形态为背景，经过数学抽象建立算法模型，通过计算机的计算来求解组合最优化问题，因此这些算法也被称之为元启发式(meta-heuristics)算法。

现代优化算法的发展非常迅速，例如，在 1996 年，Osman 在本章参考文献[1]中分类罗列了 1400 篇相关文章。因此，这些算法同人工智能、计算机科学和运筹学相融合就不足为奇了。现代优化计算方法涉及人工智能、分子运动、遗传学、动物学、神经系统和统计力学等学科的概念和理论，以模型的抽象为其关键点，以数学为其理论基础。随着人们对客观世界认识的发展及计算机技术的提高，现代优化计算方法所涵盖的内容将不断扩大。

本章主要介绍组合最优化问题、计算复杂性和启发式算法的概念。

1.1 组合最优化问题

组合最优化(combinatorial optimization)是通过数学方法的研究去寻找离散事件的最优编排、分组、次序或筛选等，是运筹学(operations research)中的一个经典和重要的分支，所研究的问题涉及信息技术、经济管理、工业工程、交通运输、通信网络等诸多领域。

我们知道，最优化问题的数学模型的一般描述是：

$$\min_{x\in F} f(x)$$

其中，x为决策变量，$f(x)$为目标函数，F为可行域，F中的任何一个元素称为该问题的一个可行解，满足 $f(x^*)=\min\{f(x)|x\in F\}$ 的可行解 x^* 称为该问题的最优解，对应的目标函数值$f(x^*)$称为最优值。

组合最优化问题的特点在于，可行域F表示的是有限个点组成的集合。通常情况下，可行域F可以表示为$\{x|x\in D, g(x)\geq 0\}$，所以组合最优化问题的一般形式为

$$\begin{aligned}\min\ & f(x)\\ \text{s.t. } & g(x)\geq 0,\\ & x\in D,\end{aligned}$$

因此，一个组合最优化问题可用三参数(D,F,f)表示，其中 D 表示决策变量的定义域，$F=\{x|x\in D, g(x)\geq 0\}$表示可行解区域（可行域），$f$ 表示目标函数。组合最优化的特点是可行解的集合 F 为有限点集，决策变量的定义域 D 通常也是有限点集。由直观可知，只要将 D 中有限个点逐一判别是否满足 $g(x)$的约束和比较目标值的大小，该问题的最优解一定存在且可以得到（除非可行域为空集）。因为现实生活中的大量优化问题是从有限个状态中选取最好的，所以大量的实际优化问题是组合最优化问题。

例 1.1.1　0-1 背包问题(knapsack problem)

设有一个容积为 b 的背包，n 个尺寸分别为 $a_i\ (i=1,2,\ldots,n)$，价值分别为 $c_i\ (i=1,2,\ldots,n)$的物品，如何装包以使装入物品的总价值最大？这个问题称为0-1背包问题，可用数学模型表示为：

$$\max \sum_{i=1}^{n} c_i x_i \tag{1.1.1}$$

$$s.t. \ \sum_{i=1}^{n} a_i x_i \le b \tag{1.1.2}$$

$$x_i \in \{0,1\},\ i=1,...,n. \tag{1.1.3}$$

其中目标(1.1.1)欲使包内所装物品的价值最大，(1.1.2)为包的能力限制，(1.1.3)表示x_i为二进制变量，x_i=1表示装第i个物品，x_i=0表示不装。此时若采用三参数表示法，D=$\{0,1\}^n$，F为D中满足(1.1.2)的解集合（可行解域），f为目标函数(1.1.1)。□

例 1.1.2 旅行商问题(TSP, traveling salesman problem)

一个商人欲到n个城市推销商品，每两个城市i和j之间的距离为d_{ij}，如何选择一条道路使得商人每个城市走一遍后回到起点且所走路径最短？TSP还可以细分为对称和非对称距离两大类问题。当$d_{ij}=d_{ji}, \forall i,j$时，称为对称距离TSP，否则为非对称TSP。对一般的TSP，一种数学模型描述为：

$$\min \sum_{i\ne j} d_{ij} x_{ij} \tag{1.1.4}$$

$$s.t. \ \sum_{j=1}^{n} x_{ij} = 1,\ \ i=1,2,...,n, \tag{1.1.5}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ \ j=1,2,...,n, \tag{1.1.6}$$

$$\sum_{i,j\in s} x_{ij} \le |s|-1,\ \ 2\le|s|\le n-2,\ \ s\subset\{1,2,...,n\}, \tag{1.1.7}$$

$$x_{ij}\in\{0,1\},\ \ i,j=1,...,n,\ i\ne j. \tag{1.1.8}$$

以上是基于图论的数学模型，其中，(1.1.8)中的决策变量$x_{ij}=1$表示商人行走的路线包含从城市i到城市j的路径，$x_{ij}=0$表示商人没有选择走这条路。$i\ne j$的约束可以减少变量的个数，使得共有$n\times(n-1)$个决策变量。目标(1.1.4)要求距离之和最小。(1.1.5)要求商人从城市i恰好出来一次，(1.1.6)要求商人恰好走入城市j一次。(1.1.5)和(1.1.6)合起来表示每个城市恰好经过一次。仅有约束(1.1.5)和(1.1.6)无法避免子回路的产生（一条回路是由$k(1\le k\le n)$个城市和k条弧组成的一个环），因此，(1.1.7)约束旅行商在任何一个城市真子集中不形成回路，其中|s|表示集合s中元素个数。此时若采用三参数表示法，D=$\{0,1\}^{n\times(n-1)}$，F为D中满足(1.1.5)，(1.1.6)和(1.1.7)的解集合（可行域），f为目标函数(1.1.4)。□

上面两个问题都是组合最优化中的经典问题。它们的共性是：通过对实际问题的数学简化，读者容易理解这些问题；决策变量的定义域和可行解集合都是有限的，在问题的规模较小时，通过枚举很容易得到最优解。

例 1.1.3 整数线性规划(integer linear programming)

$$\text{(IP)} \qquad \begin{aligned} &\min c^T x \\ &s.t. \ Ax=b \\ &x\ge 0,\ \ x\in Z^n。 \end{aligned}$$

其中，c为n维列向量，A为$m\times n$矩阵，b为m维列向量，x为n维决策变量，Z^n表示n维整数向量的集合。模型中的系数A、b和c都是整数。□

例 1.1.1 和 1.1.2 的数学模型都具有以上(IP)的形式。整数线性规划同线性规划形式上非常相似，不同之处是决策变量部分或全部取整数。

例 1.1.4 装箱问题(bin packing)

设有n个一维尺寸不超过1的物品集合$\{a_1,a_2,\cdots,a_n\}$，如何以个数最少的一维尺寸为1的箱子装进这n个物品？该问题为（一维）装箱问题。□

例1.1.5 约束机器排序问题[2](capacitated machine scheduling)

n个加工量为$\{d_i|i=1,2,\cdots,n\}$的产品在一台机器上加工，机器在第t个时段的工作能力为c_t，求出所有产品得以加工的最少时段数。它的数学模型可以表示为

$$\min T \tag{1.1.9}$$

$$s.t.\sum_{t=1}^{T} x_{it}=1, i=1,2,\cdots,n, \tag{1.1.10}$$

$$\sum_{i=1}^{n} d_i x_{it} \le c_t, t=1,2,\cdots,T, \tag{1.1.11}$$

$$x_{it}\in\{0,1\}, i=1,2,\cdots,n; t=1,2,\cdots,T, \tag{1.1.12}$$

其中，x_{it},T 为决策变量，$x_{it}=1$表示第 t 时段加工产品 i。(1.1.9)要求加工所用的时段数最少，(1.1.10)表示产品 i 一定在某一个时段加工，(1.1.11)表示每个时段的加工量不超过能力的限制。□

整数规划模型的建立有助于对问题的理解和采用已有的如分枝定界、割平面等算法求最优解，也利于松弛变量的整数约束条件得到问题的下界或上界。组合优化问题通常可以用整数规划模型的形式表示（如例1.1.1和1.1.2等），但有些组合最优化问题用整数规划模型表示比较复杂和不易被理解，不如问题的直接叙述容易理解，如例1.1.2、1.1.4和1.1.5。同时，由于组合最优化问题的复杂性，很多快速的算法只给出问题的可行解。因此，根据对解的不同精度要求和分析的需要，有大量的组合最优化问题是通过文字语言叙述的，不一定强求给出整数规划模型。

1.2 计算复杂性的概念

计算复杂性的概念是为评估算法的计算耗用时间和解的偏离程度等指标而提出的。它以目前2进制计算机中的储存和计算为基础，以理论的形式系统描述。这一套理论产生于20世纪70年代，目前仍然是评估算法性能的基础。

首先，我们以简单的实例来理解算法的计算耗用时间。由组合最优化问题的定义，每一个组合最优化问题都可以通过枚举的方法求得最优解。枚举是以时间为代价的，有的枚举时间还可以接受，有的则不可能接受。对例1.1.2的非对称距离TSP问题，可用另一个方法来表示它的可行解：n个城市的一个排列表示商人按这个排列顺序推销并返回起点。若固定一个城市为起终点，则需要$(n-1)!$次枚举。以计算机1秒可以完成24个城市所有路径枚举（固定起点后，实际上是余下的23个城市的所有排列）为单位，则25个城市的计算时间为：以第一个城市为起点，第二个到达城市有可能是第二、第三、…、或第二十五个城市。决定前两个城市的顺序后，余下是23个城市的所有排列，枚举这23个城市的排列需要1秒，所以，25个城市的枚举需要24秒。类似地归纳，城市数同计算时间的关系如表1.2.1所示。

表1.2.1 枚举时城市数与计算时间的关系

城市数	24	25	26	27	28	29	30	31
计算时间	1秒	24秒	10分	4.3小时	约4.9天	约136.5天	10.8年	约325年

通过表1.2.1可以看出，随着城市数的增多，计算时间增加非常之快，当城市数增加到30时，计算时间约10.8年，已无法接受。

上面的分析同样带来疑问：计算时间会不会随计算机的计算速度不同而不同？回答显然是肯定的。因此需要理论上规范计算时间这个概念。针对一个如TSP的问题，人们可以给出以上的枚举算法，这个算法并没有限定城市的个数和城市间的距离。但对具体的计算机，算法是以计算机语言实现的，它只能对城市数和城市间距离确定的问题给出答案。于是，我们先从计算复杂性中的“问题”和“实例”等基本概念开始讨论。

问题(problem)是一个抽象的模型或概念，是需要回答的一般性提问，通常含有若干个满足一定条件的参数。问题通过下面的描述给定：（1）描述所有参数的特性，（2）描述答案满足的条件。

如TSP是一个问题，由例1.1.2的文字或模型(1.1.4)--(1.1.8)表示，该问题的参数是：城市数和城市间的距离；而答案（经过所有城市并回到出发城市的旅行路线）满足的条件是：该旅行路线在所有可能的旅行路线中是最短的。例1.1.1-例1.1.5都是问题。

一个算法常常是针对一个问题来设计的，如TSP的枚举算法可以求解任何一个TSP的最优解。而若用计算机求解，则必须对问题中的参数赋予具体值，如TSP中的城市数和城市间的距离，问题中的参数赋予了具体值的例子称为问题的实例(instance)。一个问题通过它的所有实例表现，实例是问题的表现形式，问题是实例的抽象（集合）。

例1.1.1的背包问题，当物品的个数、包的容积、每个物品的价值和尺寸给定具体数值后，决定它的一个实例。在给出具体的城市数和城市间的距离后，确定了例1.1.2TSP的一个实例。

为了在计算机上实现算法对实例的求解，首先必须将实例输入并存储在计算机中。具体地说，需要将实例中的参数的具体数值存储。这些数值必然占据计算机的存储空间。以二进制编码存储的计算机为例，一个正整数 $x \in [2^s, 2^{s+1})$ 的二进制展开为

$$x = a_s 2^s + a_{s-1} 2^{s-1} + \cdots + a_1 \times 2 + a_0 \ (a_s = 1, a_i \in \{0,1\}, i = 0,1,...,s-1),$$

与系数 $(a_s a_{s-1} \cdots a_1 a_0)$ 一一对应。该数可由 $(a_s a_{s-1} \cdots a_1 a_0)$ 表示。这个二进制数的位数是 s+1，也就在计算机中占据s+1位的空间。由

$$2^s \le 2^{\log_2 x} = x < 2^{\log_2 (x+1)} = x+1 \le 2^{s+1} \le 2^{1+\log_2 x},$$

则

$$\log_2 (x+1) \le s+1 \le 1+\log_2 x \text{。}$$

因为

$$1+\log_2 x - \log_2 (x+1) < 1, \forall x \ge 1,$$

所以任意正整数 x 占用位数为

$$l(x) = \lceil \log_2 (x+1) \rceil, \tag{1.2.1}$$

其中 $\lceil x \rceil$ 表示不小于 x 的最小整数，且

$$\log_2 x < \lceil \log_2 (x+1) \rceil \le 1+\log_2 x \text{。} \tag{1.2.2}$$

需要注意的是整数 0、1 的二进制位数都是 1，所以特别定义 $\lceil \log_2 1 \rceil = 1$。再考虑一个符号位和一个数据分隔位，上面的二进制表示方法和输入规模的结论(1.2.1)可以推广到任何整数 x，任何一个整数 x 的输入规模为(包含一个符号位和一个数据分隔位)：

$$l(x) = \lceil \log_2 (|x|+1) \rceil + 2 \text{。} \tag{1.2.3}$$

一个数在计算机中存储时占据的位数称为这个数的规模或编码长度 (size)，由(1.2.3)决定。用 $l(I)$ 表示实例 I 的规模。一个实例的规模定义为这个实例所有参数数值的规模之和（包括不同数据之间的特殊分隔符）。常规理解的计算机只能用有限位存储一个实数，因此，在计算复杂性概念的介绍中，我们限定只考虑有理数，或者也可以限定只考虑整数。

例1.2.1 TSP实例的规模

对例1.1.2的TSP问题，它的任何一个实例由城市数n和城市间的距离$D=\{d_{ij}\mid 1\le i,j\le n, i\ne j\}$确定。于是，TSP的任何一个实例$I$的规模（考虑符号和数据分隔位）为

$$l(I)=\lceil\log_2(n+1)\rceil+2+\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\ne i}}^{n}\left\{\lceil\log_2(|d_{ij}|+1)\rceil+2\right\},$$

其中，$\lceil\log_2(n+1)\rceil+2$为$n$的存储空间。根据(1.2.2)，可以得到

$$2n(n-1)+2+\log_2 P<l(I)\le 3n(n-1)+3+\log_2 P,\qquad(1.2.4)$$

其中，$P=n\prod\{|d_{ij}|:d_{ij}\ne 0,i\ne j\}$。□

例1.2.2 整数线性规划的规模

例1.1.3的整数线性规划问题的实例需要存储系数m、n、c、A和b，其中，c为n维列向量，A为$m\times n$矩阵，b为m维列向量。在所有系数为有理数的假设前提下，可以通过通分，使得A，b，c各分量为整数。由(1.2.3)，$c=(c_1,c_2,\ldots,c_n)^T$，$A=\left(a_{ij}\right)_{mn}$，$b=(b_1,b_2,\ldots,b_m)^T$的二进制规模（考虑符号、数据分隔位）分别是

$$L_1=2n+\sum_{i=1}^{n}\lceil\log_2(|c_i|+1)\rceil,$$

$$L_2=2mn+\sum_{i=1}^{m}\sum_{j=1}^{n}\lceil\log_2(|a_{ij}|+1)\rceil$$

和

$$L_3=2m+\sum_{i=1}^{m}\lceil\log_2(|b_i|+1)\rceil。$$

占用的总位数为：$l(I)=4+\lceil\log_2(m+1)\rceil+\lceil\log_2(n+1)\rceil+L_1+L_2+L_3$。再根据(1.2.2)可以得到

$$2(mn+m+n+2)+\log_2|P|<l(I)\le 3(mn+m+n+2)+\log_2|P|,\qquad(1.2.5)$$

其中，其中P为mn与A、b和c中所有非零数的乘积。□

继续研究表1.2.1的有关TSP的计算量。可以肯定，随着计算机技术的发展，计算机计算速度会大幅度提高，因此可以计算更大规模的TSP实例。但是作为衡量计算量的一个标准，我们希望给出的衡量计算量的方法不能就计算机的不同而不同，而只与实例本身有关。

一个算法解一个实例的计算量定义为：算法求解中的加、减、乘、除、比较、读和写磁盘等基本运算的总次数。从定义中可以看出，一旦实例和算法给定，这个量是不变的（假定这里不考虑带有非确定性的算法）。如表1.2.1中提及的枚举算法，当城市数为n且第一个城市为始终点时，计算一条路径$(1,i_2,\cdots,i_n)$长度的基本运算为两两城市间距离的n个求和。因为有$(n-1)!$条路径，枚举所有路径进行$(n-1)!$次比较，可以得到最优路径。这个枚举算法的基本计算(加法和比较)总次数为$\{(n-1)!\}n+(n-1)!=\{(n-1)!\}(n+1)$。

记问题的一个实例为I，实例规模为$l(I)$，算法A在求解I时的计算量（基本计算总次数）为$C_A(I)$。当存在一个函数$g(x)$和一个常数α，使得对于该问题任意的实例I均满足

$$C_A(I)\le\alpha g(l(I)),\qquad(1.2.6)$$

则用$C_A(I)=O(g(l(I)))$表示。它的含义是：算法解实例I的计算总次数$C_A(I)$是实例规模$l(I)$的一个函数，这个函数被另一个函数$g(l(I))$控制。函数$g(x)$的函数特性决定了算法的性能。

定义1.2.1 假设问题和解决该问题的一个算法已经给定，若存在多项式函数$g(x)$，使得(1.2.6)对给定问题的所有实例成立，我们称该算法为解决对应问题的多项式时间算法。

根据定义1.2.1，一个多项式时间算法考虑了所有的基本运算，因此，它的输出结果的规模一定也与实例输入规模呈多项式关系。

为了方便，当不存在多项式函数$g(x)$使得(1.2.6)成立时，称相应的算法是非多项式时间算法或指数时间算法。相比较而言，随着变量的增加，多项式函数增长的速度比指数函数增长的速度要慢的多。例如，随着n（大于2的整数）的增加，$n^k\,(k>0\text{为整数})$的增长速度远比$a^n\,(a>1\text{为实数})$增长的速度慢。若一个多项式时间算法的计算总次数为$C_1=2^{10}n^2$，而另一个非多项式时间算法的计算次数为$C_2=2^n$。当n比较小的时候，多项式时间算法的优越性并不明显。如当n=10时，$\dfrac{C_1}{C_2}=10^2$，多项式时间算法的计算次数是非多项式时间算法的10^2倍。但当$n=2^5$时，$\dfrac{C_1}{C_2}=2^{-12}$，非多项式时间算法的计算次数上升非常快，是多项式时间算法的计算次数的2^{12}倍。

从(1.2.6)和定义1.2.1中可以发现，我们以算法计算总次数的上界和实例的规模之间存在的函数关系$g(x)$来评价算法的优劣。由于计算量根据计算总次数的上界来估计，观察(1.2.4)和(1.2.5)的上下界估计，发现任何一个实例规模的上下界偏差不超过这个实例的规模，且上下界与实例规模在同一个数量级上，于是，实际进行算法复杂性分析时，可用实例规模的下界作为一个实例规模的估计，计算总次数的上界作为计算量的估计。同样从上面的例子看出，无论是否考虑数据的分隔位或符号，实例规模都在一个数量级上，因此，可以简单地用参数的个数和所有非零数值乘积的绝对值的对数来估计。

重述一个观点，我们构造算法的目的是能够解决问题的所有实例而不单单是问题的一个实例。根据这个观点，人们逐步形成了计算复杂性的概念，主要包括算法复杂性分析和问题复杂性分析。

算法复杂性分析的研究目的是对算法进行评价。对于解决一个问题的算法，如何评估这个算法的性能？算法复杂性分析是通过(1.2.6)在最坏实例的条件下，确定是否存在多项式函数$g(x)$。

由实例的输入规模和算法在求解实例时的计算量可以找到它们之间的关系。通过这个关系可以衡量算法的好坏。对例1.1.2的TSP问题，任何一个实例的规模由(1.2.4)确定，且实例规模的上下界在同一个数量级，取下界为$2(n-1)^2+2+\log_2 P$。而枚举算法的计算总次数为$\{(n-1)!\}(n+1)$。此时发现，TSP实例的二进制规模是n和P的多项式函数。枚举算法的计算量是n的阶乘函数，无法将计算量同实例的规模建立多项式关系。

下面再以线性规划中的单纯形算法为例进行说明。先估计线性规划任何一个实例的二进制编码规模。

例 1.2.3 线性规划问题(Linear Programming)

$$\text{(LP)}\qquad \begin{aligned} &\min c^T x\\ &s.t.\ \ Ax=b\\ &\quad x\geq 0,\ \ x\in R^n, \end{aligned}$$

其中，c为n维列向量，A为$m\times n$矩阵，b为m维列向量，A，b，c各分量为整数，x为n维决策变量。与例1.2.2相同的讨论，它的每个实例I的规模$l(I)$ 同样满足关系式（1.2.5）。

单纯形算法是解决线性规划问题的主要经典算法之一，对于大多数实例，它的计算效果非常好，这可从实际工程、管理人员的大量实际应用得以证实。也可以说，对线性规划问题的很多实例，单纯形算法的计算量是实例规模的多项式函数，但也存在极端的情形。Klee和Minty在文[3]中给出下面这样一个实例：

$$
\begin{aligned}
&\min -x_n \\
&s.t. 4x_1 - 4r_1 = 1 \\
&\quad x_1 + s_1 = 1 \\
&\quad 4x_j - x_{j-1} - 4r_j = 0,\ j = 2,3,\ldots,n \\
&\quad 4x_j + x_{j-1} + 4s_j = 4,\ j = 2,3,\ldots,n \\
&\quad x_j, r_j, s_j \geq 0,\ j = 1,2,\ldots,n.
\end{aligned}
\tag{1.2.7}
$$

[3]中证明单纯形算法求解线性规划(1.2.7)的迭代步数是2^n-1。由于实例(1.2.7)共有$2n$个约束和$3n$个变量，且系数的最大值为4，P是(1.2.7)中所有非零数的乘积，所以$\log_2 |P| \leq (6n^2+1)\log_2 4$。由（1.2.5）知，$l(I)=O(n^2)$。

由此可见，对于线性规划的单纯形算法，不存在多项式函数$g(x)$和常数α，使得对问题的任意一个实例I，都有(1.2.6)成立。否则的话，不妨假设该多项式函数$g(x)$为非负增函数，则

$$2^n - 1 \leq C_A(I) \leq \alpha g(l(I)) \leq \alpha g(O(n^2)) = O(g_1(n)),$$

其中$g_1(x)$也是一个多项式函数。显然，上面的式子在n充分大时是不可能成立的。

因此，单纯形算法不是求解线性规划问题的多项式时间算法。□

计算复杂性理论的另一个研究方向是问题的复杂性分析，即对问题按计算复杂性归类。可以这样定义多项式问题：

定义1.2.2 对于给定的一个优化问题，若存在一个求最优解的算法、一个多项式函数$g(x)$和常数α，使得(1.2.6)对该问题的所有实例成立，则称给定的优化问题是多项式时间可解问题，或简称多项式问题。所有多项式问题的集合记为**P**(Polynomial)。

以例1.2.3的线性规划问题为例，已经得到单纯形算法不是线性规划问题的多项式时间算法的结论，但这并不能说明线性规划问题不属于多项式问题。Khachian在文[4]中成功地构造了椭球算法并证明了其算法是线性规划问题的多项式时间算法。因此，线性规划问题是多项式问题。

并不是所有的组合最优化问题都找到了求最优解的多项式时间算法。也就是说，还不能肯定一些组合最优化问题是否属于**P**。经过几代数学家的努力，他们研究、整理了一类难以求解的组合最优化问题，迄今为止，这些问题还没有一个有能求最优解的多项式时间算法。这一类组合最优化问题归为所谓的NP完全（NP-complete）或NP难（NP-hard）问题。受人类认识能力的限制，目前人们只能假设这一类难的组合最优化问题不存在求最优解的多项式时间算法。本章的例1.1.1、例1.1.2、例1.1.3、例1.1.4和例1.1.5都属于这类难解问题。有关计算复杂性的更多概念将在本章的第五、六节进一步讨论。感兴趣的读者可以继续阅读这一部分或参考[5]。

正因为一些组合最优化问题还没有找到求最优解的多项式时间算法，而这些组合最优化问题又有非常强的实际应用背景，人们才不得不尝试用一些并不一定可以求得最优解的算法，通常称为启发式算法，来求解组合最优化问题。

1.3 邻域的概念

在距离空间中，通常的邻域定义是以一点为中心的一个球体。光滑函数极值的数值求解中，邻域是一个非常重要的概念，求解中通过寻求上升或下降的方向，以点到点的迭代，达到函数值的上升或下降。在组合最优化中，欧氏距离的概念通常不再适用，但是在一点附近搜索另一个下降的点是组合最优化数值求解的基本思想。因此，需要重新定义邻域的概念。

定义 1.3.1 对于组合最优化问题(*D*,*F*,*f*)，D 上的一点到 D 的子集的一个映射：

$$N:\ x \in D \to N(x) \in 2^D,$$

且 $x \in N(x)$，称为一个邻域映射，其中 2^D 表示D的所有子集组成的集合。$N(x)$ 称为*x*的邻域，$y \in N(x)$ 称为*x*的一个邻居。增加 $x \in N(x)$ 的限制是为了同距离空间的邻域定义更为相似。

例 1.3.1 例 1.1.2 已给出 TSP 的一种数学模型，由模型 $D = \{x | x \in \{0,1\}^{n\times(n-1)}\}$，可以定义它的一种邻域为：

$$N(x) = \left\{ y \left| |y - x| = \sum_{i,j} |y_{ij} - x_{ij}| \le k, y \in D \right. \right\}$$，*k* 为一个正整数。

这个邻域定义使得 *x* 最多有 *k* 个位置的值可以发生变化，*x* 的邻居有 $1 + C^1_{n(n-1)} + C^2_{n(n-1)} + ... + C^k_{n(n-1)}$ 个（注意他们不一定都是可行解）。□

例 1.3.2 TSP 问题解的另一种表示法为 $D = \{S = (i_1, i_2, ..., i_n) \mid i_1, i_2, ..., i_n$ 是 1,2,...,*n* 的排列$\}$。[6]中定义它的邻域映射为 2-opt，即 *S* 中的两个元素进行对换，*N*(*S*)中共包含 *S* 的 C_n^2 个邻居和 S 本身。如四个城市的 TSP 实例，当 *S*=(1,2,3,4)时, *N*(*S*)={(1,2,3,4), (2,1,3,4), (3,2,1,4), (4,2,3,1), (1,3,2,4), (1,4,3,2), (1,2,4,3)}。□

2-opt 的定义思想可以推广到 *k*-opt $(k \ge 2)$，即对 *S* 中的 *k* 个元素按一定的规则互换。

例 1.3.3 0-1 背包问题。该问题解的另一种表示法为 D=$\{(i_1, i_2, ..., i_n) | i_1, i_2, ..., i_n$ 是 1,2,...,*n* 的排列$\}$。$(i_1, i_2, ..., i_n)$ 表示装包的排列顺序。根据这个顺序，再以容量约束判别装进包的物品，得到背包问题实例的一个可行装包方案。最优解一定可以从这些排列中选出。这样定义的邻域同上例有相同的结构（注意不同的排列可能对应于同一个可行解）。□

邻域的构造依赖于决策变量的表示，邻域的结构在现代优化算法中起很重要的作用，这一点将在以后的各章节中看出。定义了邻域后，类似连续函数，可以定义局部最优、全局最优概念。

定义 1.3.2 若 $x^* \in F$ 满足

$$f(x^*) \le (\ge) f(x),\ \text{其中}\ x \in N(x^*) \cap F,$$

则称*x**为*f*在*F*上的局部(local)最小(最大)解（点）。若

$$f(x^*) \le (\ge) f(x),\ x \in F,$$

则称*x**为*f*在*F*上的全局(global)最小(最大)解（点）。

可见，全局最优解（点）也一定是局部最优解（点）。以一维变量*x*为例，假设可行域为区间［1，10］中的整数点，目标函数值如图1.3.1，如果采用如下邻域定义：

$$N(x) = \{ y \in Z_+^n \mid\ | y - x | \le 1 \},$$

则 *x*=9 为 *f* 的全局最优（最小）点，*x*=5 为 *f* 的局部最优（最小）点；*x*=6 为 *f* 的全局最优（最大）点，*x*=3 为 *f* 的局部最优（最大）点；而 *x*=4 点既不是 *f* 的局部最大点也不是局部最小点。

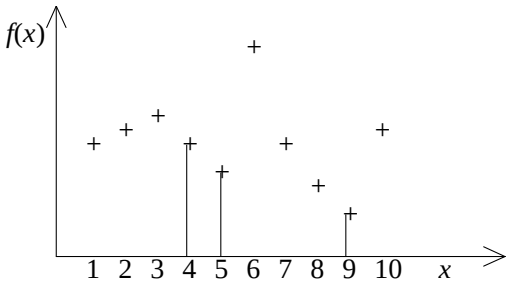


图1.3.1 局部最优示意图

在用数值方法求解使目标函数值最小的解时，传统的优化算法通常是从一个初始点出发，在邻域中寻找目标函数值更小的点，最后达到一个无法再下降的点。如图1.3.1，若以x=4为起点按传统的数值优化方法搜索最小值点，则搜索到x=5而停止，搜索到局部最优（最小）点。这种方法可能造成最终解的非全局最优性。现代优化算法所要解决的一个问题就是求解全局最优解。

1.4 启发式算法

一个问题的最优算法求得该问题每个实例的最优解。启发式算法(heuristic algorithm)是相对于最优算法提出的。启发式算法可以这样定义：

一个基于直观或经验构造的算法，在可接受的花费（指计算时间、占用空间等）下给出待解决组合最优化问题每一个实例的一个可行解，该可行解与最优解的偏离程度不一定事先可以预计。

另一种定义为：启发式算法是一种技术。这种技术使得在可接受的计算费用内去寻找最好的解，但不一定能保证所得解的可行性和最优性，甚至在多数情况下，无法阐述所得解同最优解的近似程度（参考[7]）。

在某些情况下，特别是实际问题中，最优算法的计算时间使人无法忍受或因问题的难度使其计算时间随实例规模的增加以指数速度增加，如表1.2.1列举的TSP枚举算法的情况，此时只能通过启发式算法求到实例的一个可行解。启发式方法是对启发式算法的综合应用。

上面启发式算法的两个定义中，一个共同的特点是不考虑算法所得解与最优解的偏离程度。如果采用最坏实例下的误差界限来评价启发式算法，则可以定义近似算法：

定义1.4.1 设A是一个问题。记问题A的任何一个实例I的最优解和启发式算法H得到的解的目标函数值分别为 $z_{OPT}(I)$ 和 $z_H(I)$。于是对于某个 $\varepsilon \geq 0$，称H是A的 ε 近似算法(ε-approximation algorithm)当且仅当

$$\left|z_H(I) - z_{OPT}(I)\right| \leq \varepsilon |z_{OPT}(I)|, \qquad \forall I \in A。 \tag{1.4.1}$$

由此可以发现启发式算法和近似算法两个概念的联系和区别。启发式算法定义的算法集合包含近似算法定义的算法集合。近似算法强调给出算法最坏情况的误差界限，而启发式算法不需考虑偏差程度。这种用最坏实例来评价算法的方法被称为最坏情形分析（worst case analysis）。由于很多组合最优化问题最坏情形误差界估计需要很强的数学基础和较强的技巧，甚至一些问题很难或无法给出最坏情形误差界，而实际问题又迫切需要求解的方法，因此，只能通过启发式算法解决问题。

早在20世纪40年代末期，由于实际问题的需要和科技发展的状况，人们已提出一些解决实际问题的快捷有效的启发式算法，有代表性的工作是1948年Polya的著作[8]。60-70年代，出于对最优算法的重视，这些启发式算法被称为“快速与丑陋（Quick and Dirty）”的。随着20世纪70年代计算复杂性理论的发展和完善，人们不再强调一定求到最优解，因此从20世纪80年代初开始，掀起了现代优化算法的研究热潮，现代优化算法得到了巨大的发展。

让我们通过下面的例子对启发式算法有一定的了解。

例 1.4.1 背包问题的贪婪算法(greedy algorithm)

构造下面的贪婪算法：

STEP1 对物品以 $\frac{c_i}{a_i}$ 从大到小排列，不妨把排列记成{1,2,...,*n*}, *k*:=1；

STEP2 若 $\sum_{i=1}^{k-1} a_i x_i + a_k \le b$，则 $x_k = 1$；否则，$x_k = 0$。*k*:=*k*+1；当*k*=*n*+1时，停止；否则，重复STEP2。

$(x_1, x_2, \ldots, x_n)$为贪婪算法的解。单位尺寸价值比越大越先装包是上面贪婪算法的原则。这样的算法非常直观，非常容易操作。$\frac{c_i}{a_i}(i = 1,2,\cdots,n)$ 的比值计算需要*n*次运算，$\frac{c_i}{a_i}(i = 1,2,\cdots,n)$ 从大到小排列需要 $n\log n$ 次运算。$\sum_{i=1}^{k-1} a_i x_i + a_k \le b(k = 1,2,\cdots,n)$ 对每一个*k*需要一个加和一个比较，共2*n*次运算。这个贪婪算法的计算量为 $O(n\log n)$，是一个多项式时间算法。□

例 1.4.2 简单的邻域搜索(local search)算法

给定组合最优化问题，假设其邻域结构已确定。设D为解集合（这里假设它就是可行域F），*f*为*D*上的费用函数，*N*为邻域结构。算法为：

STEP1 任选一个初始解 $s_0 \in$D；

STEP2 在N(s_0)中按一定规则选择一个s；若f(s)<f(s_0)，则 s_0:=s；否则，

N(s_0):=N(s_0)－{s}；若N(s_0)-{s_0}=ϕ，停止；否则，返回STEP2。

简单的邻域搜索从任何一点出发，达到一个局部最优值点。从算法中可以看出，算法停止时得到点的性质依赖算法初始解的选取、邻域的结构和邻域选点的规则。一个直观的看法是：只要选好初始点，就一定可以求到最优解。对NP难的组合最优化问题，确定这样的初始点是非常困难的，如非对称TSP问题，若固定一个城市为起终点，共有(*n*-1)!个解，也就是有(*n*-1)!个初始点。如何选初始点和如何跳出局部最优值点以达到全局最优点是本书后续章节的一个主要内容。□

启发式算法能够迅速发展是因为它有以下长处：

第， 数学模型本身是实际问题的简化，或多或少地忽略了一些因素；而且数据采集具有不精确性；参数估计具有不准确性；这些因素可能造成最优算法所得到的解比启发式算法所得到的解更差。

第二， 有些难的组合最优化问题可能还没有找到最优算法，即使存在，由算法复杂性理论，它们的计算时间是无法接受或不实际的。

第三， 一些启发式算法可以用在最优算法中。如在分枝定界算法中，可以用启发式算法估计下（上）界。

第四， 简单易行；比较直观；易被使用者接受。

第五， 速度快，这在实时管理中非常重要。

第六， 多数情况下，程序简单，因此易于在计算机上实现和修改。

虽说有诸多好处，但启发式算法也有其短处和不足，这些不足往往成为争论的焦点。

第一， 不能保证求得最优解（有时甚至不能保证求到可行解）。

第二， 表现不稳定。启发式算法在同一问题的不同实例计算中会有不同的效果，有些很好，而有些则很差。在实际应用中，这种不稳定性造成计算结果不可信，可能造成管理的困难。

第三， 算法的好坏依赖于实际问题、算法设计者的经验和技术，这一点很难总结规律，同时使不同算法之间难以比较。

1.4.1 启发式算法的分类

本节对启发式算法进行简单的分类。

1.4.1.1 简单直观的算法

一步算法

该算法的特点是：不在两个可行解之间比较，在未终止的迭代过程中，得到的中间解有可能不是一个可行解。

一步算法的一个典型例子是例1.4.1背包问题的贪婪算法。每一步迭代选一物品装包直到无法再装。该算法没有在两个可行解之间比较选择，算法结束时得到一解。

再以TSP中的路径搜索算法来解释上面的归类。

例 1.4.3 TSP 的路线搜索算法为：

STEP1 P={1}; N={1,2,...,n};i=1;

STEP2 从$Q=\{d_{ij}\mid i$可达j，但$j\notin P\}$选最小的值d_{ii_0}，对应城市为i_0,置P=P∪{i_0}，

若P={1,2,...,n}或$Q=\varnothing$，停止，否则$i=i_0$，重复STEP2。

该算法的特征是每一步选择一个城市进入集合P，算法没结束前没有得到可行解。算法结束时，若P={1,2,...,n}，则得到可行解。否则没有给出可行解。这个算法同样没有在两个可行解之间选择，也可能输出的不是可行解。□

改进算法

改进算法的迭代过程是从一个可行解到另一个可行解变换，通过两个解的比较而选择好的解，进而作为新的起点进行新的迭代，直到满足一定的要求为止。因此，也可以称之为迭代算法。例1.4.2的局部（邻域）搜索算法，再如TSP中简单的2-opt方法（参考[6]），是改进算法的一种。

例 1.4.4 四个城市{1,2,3,4}的 TSP 问题，两个城市 i 和 j 之间的距离 d_{ij} 构成的距离矩阵为

$$(d_{ij})=\begin{bmatrix}0 & 5 & 10 & 15\\ 5 & 0 & 7 & 8\\ 10 & 7 & 0 & 6\\ 15 & 8 & 6 & 0\end{bmatrix}$$

设初始点为：s_0=(1,2,3,4)，则目标函数值为

$$f(s_0)=d_{12}+d_{23}+d_{34}+d_{41}=5+7+6+15=33。$$

s_0的2-opt邻域是对s_0的任两个元素对换，如果固定第一个城市1，即$N(s_0)$={(1,2,3,4), (1,3,2,4), (1,4,3,2), (1,2,4,3)}，简单的2-opt方法是比较邻域中的所有点，选出最好解。比较可

知$N(s_0)$中最好的解为s_1=(1,2,4,3)，目标值为$f(s_1)$=29。下一次迭代是以s_1为起点，重复以上的计算过程，直到目标值无法改进为止。□

改进算法保证解的可行性，以局部最优作为算法的结束。

上面二个算法的共性是算法的构造非常直观，完全类似于实际的运作规则。这样的算法易于构造，也容易被使用者接受。但根据不同的问题、不同的设计者而千变万化。

1.4.1.2 数学规划算法

数学规划算法主要指用连续优化（如线性规划）的方法求解组合最优化问题（如整数线性规划模型），其中包括一些启发式规则。这一类方法中，比较典型的例子是线性规划及其对偶理论在网络流中的应用，如标号算法等一系列最优算法，可参考[9,10]。其次是基于整数规划分枝定界的启发式算法，只搜索一些特殊分支，或是基于整数规划的割平面法，产生考虑部分割平面的算法。

用连续优化的方法求解整数规划一般称为松弛（relaxation）。例如，线性规划松弛的主要步骤是先将整数线性规划问题 IP 松弛为线性规划 LP：

$$\text{(IP)}\quad \begin{aligned} z &= \min c^T x \\ s.t. Ax &\geq b \\ x \geq 0, x &\in Z^n \end{aligned} \qquad \text{(LP)}\quad \begin{aligned} z_L &= \min c^T x \\ s.t. Ax &\geq b \\ x &\geq 0 \end{aligned}$$

通过求解LP可以得到IP的一个下界$z_L \leq z$。LP是多项式可解问题（参考[4]），但LP的解不一定是IP的可行解。于是，解空间松弛算法的第二步是如何将LP的解转化为IP的可行解，如四舍五入法，取上整数、下整数等。

另一类常用的松弛方法是拉格朗日(Lagrange)松弛法，它主要用于求解下面这样的组合最优化问题（IP）：

$$z = \min c^T x \tag{1.4.2}$$
$$s.t. A_1 x \geq b_1 \tag{1.4.3}$$
$$A_2 x \geq b_2 \tag{1.4.4}$$
$$x \geq 0, x \in Z^n 。 \tag{1.4.5}$$

假设问题具有如下特性：如果没有(1.4.4)这些约束，IP问题就变成是一个多项式时间问题，而有了(1.4.4)这些约束，IP为NP难问题。此时通常称(1.4.4)为难约束。拉格朗日松弛法的主要步骤通常分为三个阶段。第一阶段是先将整数规划问题IP松弛为拉格朗日松弛问题（LR）：

$$z_{LR}(\lambda) = \min c^T x + \lambda^{\mathrm{T}}(b_2 - A_2 x) \tag{1.4.6}$$
$$s.t. A_1 x \geq b_1 \tag{1.4.7}$$
$$x \geq 0,\ \lambda \geq 0, x \in Z^n . \tag{1.4.8}$$

松弛约束(1.4.4)后，IP成为一个易解的问题。(1.4.4)松弛后，IP的解空间扩大，于是将(1.4.4)以罚函数的形式在LR的目标(1.4.6)体现，使得得到的解不至于使(1.4.4)的不可行性过大。在这一阶段中，提供了IP的一个下界$z_{LR}(\lambda)$。

第二阶段是求IP的拉格朗日对偶问题（LD）的解，即

$$z_{LD} = \max_{\lambda \geq 0} z_{LR}(\lambda) = \max_{\lambda \geq 0} \min_{\{x | A_1 x \geq b,\, x \geq 0,\, x \in Z^n\}} c^T x + \lambda^{\mathrm{T}}(b_2 - A_2 x) 。 \tag{1.4.9}$$

通过第二阶段的求解，得到(1.4.9)的解x和λ，且自然有$z \geq z_{LD} \geq z_{LR}(\lambda)$。从理论上可以讨论$z$同$z_{LD}$、$z$同$z_{LR}(\lambda)$、$z_{LR}(\lambda)$同$z_{LD}$等的差距，这些内容将在第七章——拉格朗日松弛算法中讨论。

第三个阶段将第二阶段所得解可行化，使之成为IP的可行解。可行化的方法往往依赖于问题本身，这些内容也将在第七章中讨论。

这些算法基于数学规划的理论，可以借鉴数学规划已有的理论和方法进行分析。

1.4.1.3 现代优化算法

现代优化算法是20世纪80年代初以来得到深入研究和广泛应用的启发式算法。这些算法主要包括禁忌搜索算法，模拟退火算法，遗传算法，蚁群优化算法，人工神经网络算法等。它们的共性是基于客观世界中的一些自然现象，通过与组合最优化求解进行类比，找出它们的一些共性，建立相应的算法。这些算法的目标是希望能够求解NP难问题的全局最优解，有一定的普适性，可用于解决大量的实际应用问题。

目前，这些算法在理论和实际应用都得到了较大的发展。虽说有些算法可以从理论上证明能够收敛到NP难组合最优化问题的全局最优解，但NP难的理论限制它们只能以启发式算法的方式去求解实际问题。本书后续章节将以这几类算法为重点，从基本理论、应用技术和应用实例等方面讲解。

1.4.1.4 其他方法

启发式方法包含的类型很多，有些方法根据实际问题而产生。如解空间分解，解空间的限制等。另一类算法是集成（混合）算法，这些算法是诸多启发式算法的合成。

1.4.2 启发式算法的性能分析

虽说启发式算法有诸多优点，但它的一个缺陷是无法保证得到最优解。于是，对算法的评价显得尤为重要。评价一个算法的优劣常用三个主要指标：算法的复杂性（计算效率）、解的偏离程度（计算效果），和算法的稳健性，即同一算法对不同实例、在不同时间、不同起点的计算效果的差异大小。

评价算法的优劣可以采用三种不同的手段。第一，最坏情形分析，即通过最坏实例来评价算法的复杂性和解的偏离程度，这是一个纯理论的方法。第二，概率分析的方法，即假设实例的数据服从不同的概率分布，研究算法的效率和效果，这也是一种理论方法。上面两种手段的优点是理论分析的可靠性，精确性，但回避不了的问题是需要非常强的数学基础，而且还有很多问题无法或很难进行这样的分析。第三，计算模拟分析，即通过大量的数据、实例进行计算测试和模拟，分析算法的平均复杂性和解的偏离程度，也可以对解的稳定性进行分析。计算模拟的优点是显而易见的：若要对算法进行分析，可以选择有代表性的数据实例进行计算，将计算结果用统计分析方法进行比较和处理。但计算模拟分析也会引起质疑，质疑的主要内容有：所选择的实例是否有代表性？是否全面？分析的结果是否适用问题的所有实例？等等。

无论如何，一个好的启发式算法可以使其解同最优解尽可能地接近，同时保证有较好的稳健性。下面仅简单介绍几种常用的方法。

1.4.2.1 最坏情形分析(worst case analysis)

最坏情形分析可以考虑算法的计算复杂性和计算得到的解的目标值与最优解目标值的偏离程度两个方面。最坏情形下的计算复杂性分析关注算法基本运算总次数同实例计算机二进制编码规模之间的关系，从最坏实例的角度来研究算法的计算时间复杂性。这一部分内容已在第二节讨论过，这里不再重复讨论。

用最坏情形分析来评价一个算法的计算效果时，其评价的指标是计算得到的目标值同最优目标值的差距。这个差距越小说明算法越好。这样的评价只对启发式算法有效，因为可求到最优解的最优算法没有这样的偏差。

记一个实例I的最优目标值为$z_{OPT}(I)$，启发式算法H所得的目标值为$z_H(I)$。若优化问题的目标是极大化目标函数，对优化问题的任意实例I，评价如下关系式是否成立：

$$d + \alpha z_{OPT}(I) \le z_H(I) \le z_{OPT}(I), \tag{1.4.10}$$

其中，$|d|<z_{OPT}(I),\alpha\le 1$，$\alpha$ 和d是与实例I无关的常数。对给定的问题和启发式算法H，我们对问题中所有实例I求满足(1.4.10)式的 α 。α 称为最坏性能比，越接近1说明构造的启发式算法越好。d 则是一个与实例I无关的偏差值。在常规的研究中，要给出 α 的值并设法证明它是否是紧的比率（所谓紧的比率是指存在一个实例使(1.4.10)左边的等号成立）。

当优化问题的目标是极小化目标函数时，对所有实例I，评价关系式为

$$\alpha z_{OPT}(I)+d\ge z_H(I)\ge z_{OPT}(I) \tag{1.4.11}$$

其中，$|d|<z_{OPT}(I)$，d 是一个与 I 无关的偏差值，$\alpha\ge 1$。同(1.4.10)类似，α 越接近 1 说明构造的启发式算法越好。

有时不易确定和验证 α 是紧的，与其相近的方法是确定渐近最坏比率，它的定义为

$$AR(H)=\lim_{k\to\infty}\sup_I\{\frac{z_H(I)}{z_{OPT}(I)}|z_{OPT}(I)=k\}。 \tag{1.4.12}$$

迄今为止，我们已经介绍了关于算法计算结果偏离程度的评价方法(1.4.1)，(1.4.10)，(1.4.11)和(1.4.12)。仔细比较这几个公式，可以发现它们之间的差别。(1.4.1)关心的是算法对所有实例计算结果的偏离程度，而为了估计得更为精确，(1.4.10)和(1.4.11)将常数项也考虑在内。(1.4.12)则考虑最优目标值逐渐增大时，计算的目标值同最优目标值的偏离程度。

通过下面的简单例子说明最坏情形分析方法。

例 1.4.5 例 1.4.1 已给出 0-1 背包问题的贪婪算法，记为 G 算法。用最坏情形分析该算法，可以构造这样一个例子 I：有两个物品，c_1=1+ε，$a_1=1$，$c_2=K$，$a_2=K$，$b=K$，ε为充分小的正数。用例 1.4.1 的算法得到$x_1=1$，$x_2=0$，则$z_G(I)$=1+ε，而$z_{OPT}(I)=K$。于是，随着容积 K 的增加，有

$$\frac{z_G(I)}{z_{OPT}(I)}=\frac{1+\varepsilon}{K}\to 0,K\to+\infty。 \tag{1.4.13}$$

这一渐近结果的物理意义是包中几乎没有装任何物品（价值渐近为0）。由于对任何一个启发式算法，最坏的情形莫过于一个物品也不装，所以(1.4.13)说明0-1背包问题的贪婪算法在最坏情况分析下非常不好。□

从上面的例子可以看出，要说明一个算法会达到最坏的比率，相对要容易一些，只需给出一个实例即可（当然，构造一个这样的实例有时也是非常需要技巧的）。但是，如果要估计一个准确的比率，需要做的工作可能要多得多。同样针对以上的贪婪算法，考虑将它应用到一般的背包问题，它的效果则不一样。

例 1.4.6 一般背包问题

$$z_{OPT}(I)=\max\sum_{j=1}^{n}c_jx_j$$

$$s.t.\sum_{j=1}^{n}a_jx_j\le b$$

$$x_j\ge 0\text{整数},j=1,2,...,n.$$

一般背包问题不同于0-1背包问题之处在于一种物品有无限多个，而且同一物品可以在包中装多个。用例1.4.1的贪婪算法，不妨设 $\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \cdots \geq \frac{c_n}{a_n}$，且 $a_j, c_j > 0$。明显的结论是 $x_1 = \left\lfloor \frac{b}{a_1} \right\rfloor$，$x_2 = \ldots = x_n = 0$ 是一个可行解，其中 $\lfloor x \rfloor$ 表示不超过x的最大整数。所以，

$$z_G(I) \geq c_1 \left\lfloor \frac{b}{a_1} \right\rfloor 。 \tag{1.4.14}$$

简单的推导可得

$$\sum_{j=1}^{n} c_j x_j = \sum_{j=1}^{n} \frac{c_j}{a_j} a_j x_j \leq \frac{c_1}{a_1} \sum_{j=1}^{n} a_j x_j \leq \frac{c_1}{a_1} b , \tag{1.4.15}$$

也可以对模型的决策变量线性松弛，将一般背包问题松弛成为线性规划问题。因为线性规划的最优解一定能在某个极点达到，而这里的模型仅有容量约束，所以由线性规划理论，也可以得到(1.4.15)。于是

$$z_{OPT}(I) \leq \max\{c_j \frac{b}{a_j} | j = 1,2,\ldots,n\} \leq c_1 \frac{b}{a_1} 。 \tag{1.4.16}$$

比较(1.4.14)和(1.4.16)，

$$\frac{z_G(I)}{z_{OPT}(I)} \geq \frac{\left\lfloor \frac{b}{a_1} \right\rfloor}{\frac{b}{a_1}} \geq \frac{\left\lfloor \frac{b}{a_1} \right\rfloor}{1 + \left\lfloor \frac{b}{a_1} \right\rfloor} \geq \frac{1}{2},$$

所以，$2z_G(I) \geq z_{OPT}(I)$，即最坏性能比 $\alpha = \frac{1}{2}$。当

$$c_1 = K + 3\varepsilon,\ c_2 = K,\ a_1 = \frac{K}{2} + \varepsilon,\ a_2 = \frac{K}{2},\ \varepsilon > 0,\ b = K > 2$$

时，贪婪算法*G*得到的解是：只装入一个第一类物品，对应的 $z_G(I) = K + 3\varepsilon$。然而，ε为充分小的正数，所以最优解是装两个第二类物品，$z_{OPT}(I) = 2K$。所以，比率

$$\alpha = \lim_{z_{OPT}(I) \to \infty} \frac{z_H(I)}{z_{OPT}(I)} = \lim_{K \to \infty} \frac{K + 3\varepsilon}{2K} = \frac{1}{2},$$

为渐近紧。□

1.2 节给出的实例（见(1.2.7)）说明线性规划的单纯形算法的计算复杂性是指数型的。这是有关算法计算复杂性最坏情形分析的一个很好的实例。

1.4.2.2 概率分析

最坏情形分析是以最坏的实例来评价算法，这样免不了只因一个（坏）实例而影响对一个算法的总体评价。从另一个方面讲，应该从总体来评价算法，概率方法就是从理论上这样考虑的。对一个算法同样可以从算法的计算时间复杂性和计算解的效果两个方面进行概率分析。无论进行那一个方面的分析，都需要假设实例的数据服从一定的概率分布。在这个概率分布的假设下，研究其算法或解的平均效果。

概率方法在评价算法方面的一个成功应用是对线性规划单纯形法的评价。Klee和Minty在[3]中证明了线性规划的单纯形法不是多项式时间的，其构造的实例为第二节中的(1.2.7)。Borgwardt[11]用概率分析方法研究了单纯形算法的平均计算量。他研究的线性规划问题为

$$\max c^T x \quad s.t.\ Ax \le e$$

其中，$A \in R^{m\times n}$，$e=1 \in R^m$，$m \ge n$，

其中，[11]中假设规划的输入数据$c^T, A_1, A_2, \ldots, A_m$是$R^n$中具有球面对称测度，得到如下一个结论：当输入数据在单位球面服从均匀分布，且 n 充分大时，其迭代次数的渐近上下界分别为$n^{1.5}m^{1/n-1}$和$n^2 m^{1/n-1}$。于是当 n>10，m<10^6时，$m^{1/n-1}$<4。当 n>100，m<10^6时，$m^{1/n-1}$<1.15。此时，单纯形算法的平均迭代次数为 $O(n^{1.5}\sim n^2)$。这说明单纯形算法的平均计算效果是较好的。

概率分析方法是一种理论分析方法，它需要对问题本身有较深入的理解，并且掌握概率模型的建立和概率理论。最坏情形分析和概率分析方法的共同特点是：用理论方法分析算法同最优解之间的误差界，要求有较强的数学基础。

1.4.2.3 大规模计算分析

前面两种分析方法都是理论分析方法，要求有很强的数学基础和推演能力。正因为如此，目前研究者只对一些特殊和简单的问题采用上面的理论分析方法。实际中大量的组合最优化问题是采用大规模计算分析方法。简单地说，大规模计算分析就是通过大量的实例计算（数值实验），评价算法的计算效果。算法的计算分析也分成两个方面：一方面是算法的计算复杂性，即通过计算耗费的计算机中央处理器（CPU）的计算时间表现，评价的是算法的计算效率；另一个方面是计算得到的解的性能，它通过计算停止时的输出结果的好坏表现，评价的是算法的计算效果。

大规模计算分析是对算法进行评价的方法之一，往往通过目前我们的计算机设备能否承受、用户能否接受现有的计算时间来衡量算法的计算复杂性。对它的计算输出结果进行评价时，一个简单的要求是用户是否满意；更进一步的分析需要知道问题的最优解或问题的一个下界（假设目标为最小），这时可以通过比较算法得到的解与最优解或下界的偏离程度，评价一个算法的计算效果。

采用大规模计算分析方法对多个算法进行评价时，可以比较分析不同算法的计算效率和效果。根据各个算法针对大量测试实例的的计算结果，可以采用简单的或统计的方法比较不同算法的性能。这种比较不一定需要已知问题的最优解或上下界。

在采用大规模数据计算分析方法时，需要注意两个基本问题：测试实例数据的产生方式和计算结果的分析方法。下面分别对此进行简要介绍。

首先，需要注意测试实例数据的产生方式。对实际应用问题，本身存在大量的实际数据，可以用这些数据作为测试对象。需要注意的是数据的偏颇性，因为实际数据往往在某一个限定范围内变化，一旦实际问题中的数据出现大的变化，有些算法的性能可能不具有稳健性（Robustness）。因此，用实际数据评价算法时，可以选出适合这类数据的最好算法。若实际问题数据有较大的变化时，应该注意算法的适应性。为了弥补这方面的不足，还应该用数值实验的方法产生一些有代表性的数据。

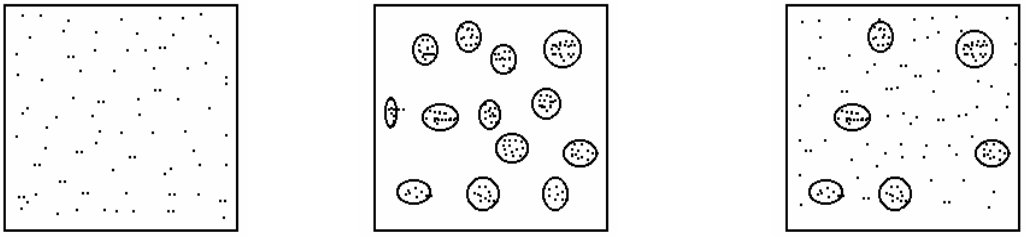
研究组合最优化问题时，一种常用的数值实验方法是产生一些具有代表性的随机数据，通过算法对这些数据的计算结果来评价算法的计算效率和效果。这些数据的产生应充分考虑代表性，只有这样对算法进行的评价才具有可信性。Soloman[12]在研究具有时间窗口的车辆线路问题时，给出了一个56个实例的产生方法，通过这个例子可以了解随机产生数据的一种考虑方法。

例 1.4.7 经典的具有时间窗口的车辆线路问题可以简单地描述为：m 个车辆服务于 n 个客户，同 n 个城市的 TSP 相同，客户与客户间有路径费用；同时，每个客户有特殊的服务

时间称为时间窗口，只有在服务时间内的服务才是有效的；如何安排 m 个车辆使得满足服务时间的要求且行驶的费用最小。随机数据产生的规则如下：

客户的位置分布

按均匀分布(uniform)、堆分布(cluster)和半堆分布(semi-cluster)三类情况。均匀分布表示客户位置在一个区域内服从均匀分布。堆分布表示以一些中心点密集分布。半堆分布则介于均匀分布和堆分布之间。参见图1.4.1表示的客户分布示意图。



均匀分布　　堆分布　　半堆分布

图1.4.1 客户位置分布形式

客户位置按上面三种分布产生坐标点。考虑这样三个典型分布的主要背景为：均匀分布认为客户在一个城市里呈现均匀的分布；堆分布表示一些客户集中在某些居住小区，小区以外基本上没有服务对象。半堆分布则介于前两者之间。

时间窗口

时间窗口是指各客户要求服务的开始和完成时间。[12]中分别按75%和100%的客户有时间窗口限制两种要求，时间跨度又分为大和小两种情况，由均匀分布产生时间窗口数据。

文[12]共产生56个数据实例。这些实例现已成为研究有时间窗口车辆路线问题的典型数据。□

其次，需要注意计算结果的分析方法。针对一个算法进行评价和针对多个算法进行比较，计算结果的分析方法略有不同。针对一个算法计算结果的分析，常常需要知道测试实例I的最优解 $z_{OPT}(I)$ 或下界 $z_{LB}(I)$（目标是极小）。记算法H的目标值为 $z_H(I)$，常用于评价解的性能指标有绝对差

$$z_H(I)-z_{OPT}(I) \text{ 或 } z_H(I)-z_{LB}(I)\,, \tag{1.4.17}$$

相对差

$$\frac{z_H(I)-z_{OPT}(I)}{z_{OPT}(I)} \text{ 或 } \frac{z_H(I)-z_{LB}(I)}{z_{LB}(I)}\,, \tag{1.4.18}$$

或

$$\frac{z_H(I)-z_{OPT}(I)}{z_H(I)} \text{ 或 } \frac{z_H(I)-z_{LB}(I)}{z_H(I)}\,。 \tag{1.4.19}$$

需要注意的是，在上述评价公式中，应尽量利用已有的信息，以避免过多的计算。如在(1.4.18)和(1.4.19)的评价公式中，每个公式只用到两个值进行计算，而没有同时采用三个值进行计算。

得到上述的评价指标后，就可以采用典型数据或统计处理的方法，评价一个算法的性能。例如，可以采用上述指标的平均值来评价算法的平均性能；可以采用上述指标的最大值来评价算法的最坏性能；等等。特别地，对于算法的稳健性分析，可采用统计学中的方差

$$D=\sum_{I\in S}(z_H(I)-\bar{z})^2, \tag{1.4.20}$$

其中，

$$\bar{z}=\frac{1}{|S|}\sum_{I\in S}z_H(I), \tag{1.4.21}$$

S表示所有数据实例的集合，$|S|$表示S中包含的实例数。

算法所耗用的计算时间也是一个重要的指标，可采用与上面类似的统计分析方法，如可以用类似于(1.4.20)的方差公式考虑耗用时间指标下算法的稳健性。

对多个算法进行分析比较时（不妨假设优化问题的目标是极小化目标函数），对一个实例I，若两种算法H_1和H_2满足，

$$z_{H_1}(I)<(\leq)z_{H_2}(I), \tag{1.4.22}$$

则称H_1关于实例I优于（dominate，不次于）H_2。

多个算法的分析比较同样可以考虑算法得到的解的性能、耗用时间、算法的稳健性等指标。 (1.4.22)的关系式是对每一个实例进行比较，同样，可以用算法对一组数据计算结果的平均值比较不同算法的平均计算效果。类似(1.4.22)，可以比较计算时间、稳健性等指标下不同算法的优劣关系。

最后，我们用图1.4.2表示不同的启发式算法得到的解之间的关系（不妨假设问题要求最小化目标函数）。启发式算法的一个主要目标是寻找比较好的可行解，于是前面提到的简单直观算法、基于数学规划的启发式算法和我们后续各章将要介绍的现代启发式算法都是提供可行解的算法。虽然我们很难对这些算法的好坏简单地进行理论比较，但无论如何，它们都有一个共同特点：算法得到的解的目标值不会好于最优值。

图中的也包括了一些提供下界的算法，如线性规划松弛，拉格朗日松弛等。这些算法的一个主要作用是提供问题的一个下界，以便通过比较启发式算法得到的目标值同下界的差，了解启发式算法的好与坏。

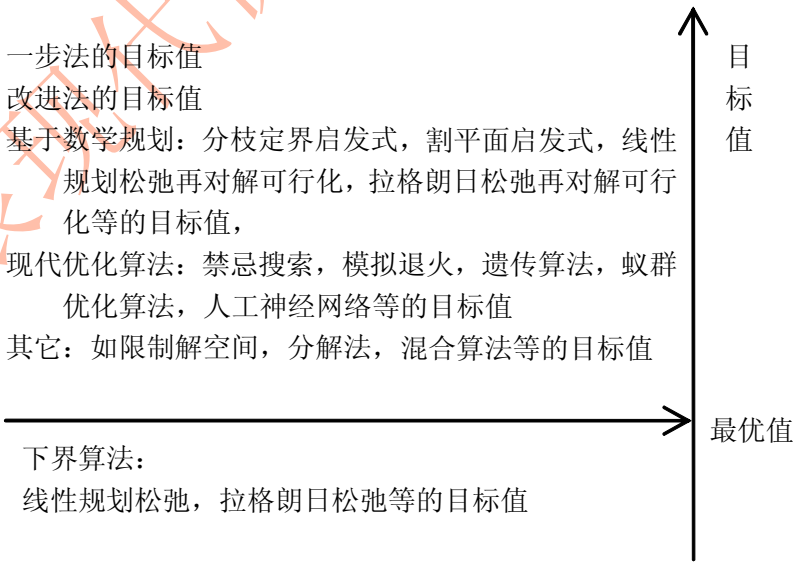


图 1.4.2 启发式算法目标值的关系

对极小化目标函数的优化问题，理论上可以保证并且很容易证明：任何一个可行解的目标值都是最优值的上界。在图1.4.2中，简单直观算法、分支定界启发式、割平面启发式、线性规划松弛的解可行化算法、拉格朗日松弛的解可行化、禁忌搜索、模拟退火、遗传算法、蚁

群优化算法、人工神经网络、限制解空间、分解法和混合算法等的目标值都是最优值的上界。理论上难以区分这些算法的目标值之间的顺序关系，要了解这些算法对一个问题的适用性，通常只有通过大规模数值计算分析比较。

1.5 NP,NP 完全和 NP 难

第1.2节初步介绍了计算复杂性的一些概念。我们知道了多项式问题，同时也知道例1.1.1-1.1.5的组合最优化问题还没有找到多项式时间的最优算法。本节将以组合最优化问题为研究对象，介绍计算复杂性的NP(nondeterministic polynomial)、NP完全(NP-Complete)和NP难(NP-hard)等概念。通过用几个典型组合最优化问题一步一步地展开、深入，逐渐理解有关NP、NP完全和NP难的概念。同时，以这些问题的一步步深入，介绍如何研究一个组合最优化问题的复杂性。

问题、实例和规模

在第二节，已经介绍了问题、实例和输入规模等概念。为了加深对这些概念的理解，用表1.5.1和表1.5.2的形式来说明它们之间的关系。

表1.5.1 问题和实例

问题	实例
TSP	如例1.1.2问题中的各参数为：100个城市，城市间距离d_{ij}已知。
背包问题	如例1.1.1问题中的各个参数为：4个物品，大小分别为4,3,2,2。价值分别为8,7,5,7。包的大小为6。
整数线性规划	如例1.1.3问题中的n,A,b,c已知。

实例的规模同计算机中的存储编码方式有关，如采用二进制编码，则通过表1.5.2了解规模的概念（不包括符号位和分隔位）。

表1.5.2 编码与规模

数据	二进制	规模
1	1	1
2	10	2
10	1010	4
64	1000000	7
k≥2		$\lceil \log_2(k+1) \rceil$

一般以正整数β$(\beta>1)$进制记录一个正整数$n\in[\beta^k,\beta^{k+1})$的展开形式为

$$n=a_k\beta^k+a_{k-1}\beta^{k-1}+\cdots+a_0\beta^0, \tag{1.5.1}$$

其中，$0\le a_i<\beta(0\le i\le k)$且$a_k\neq 0$，它的β进制数据为$(a_k a_{k-1}\cdots a_0)$。与1.2节类似的讨论可知，正整数$n\in[\beta^k,\beta^{k+1})$的规模为

$$l(n)=k+1=\left\lceil \log_\beta(n+1)\right\rceil 。 \tag{1.5.2}$$

这个数在$\log_\beta n$与$\log_\beta n+1$之间。对多个数据的实例，采用所有数据规模累计的方法记录

实例的规模。

此处都是对非负整数考虑规模，即没有考虑实际数据中的正负符号和数据分隔位。如果考虑符号和数据分隔位，只需在原有的规模上增加二位。

判定问题

迄今为止，许多的组合最优化问题都没有找到求最优解的多项式时间算法，比多项式问题更广泛的一个问题类是非确定多项式 (nondeterministic polynomial，简记NP)问题。NP的概念是由判定问题引入的。

定义1.5.1 如果一个问题的每一个实例只有“是”或“否”两种答案，则称这个问题为判定问题。称有肯定答案“是”的实例为“是”实例，称答案为“否”的实例为“否”实例或非“是”实例。

例 1.5.1 Hamilton 回路问题

给定 n 个节点及一些节点间的连线构成的图，是否存在一个连接所有节点的闭路(即圈)，使得每个节点在闭路（圈）上只出现一次？□

这是一个判定问题。对每一个实例，已经给定的是节点的个数和节点间的连线。“是”或“否”实例完全取决于连线。如：

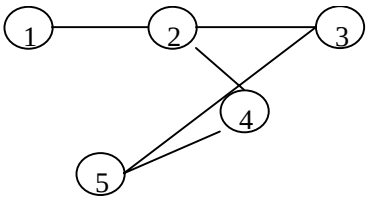


图 1.5.1(a) “否“实例

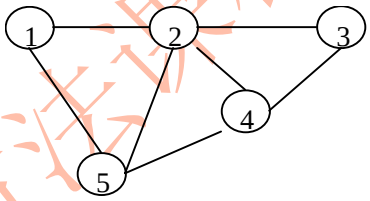


图 1.5.1(b) “是“实例

图 1.5.1(b)的一个闭路为 1-2-3-4-5-1，所以这个实例是 Hamilton 回路问题的一个“是”实例，而图 1.5.1(a)则无法找到这样的闭路，是 Hamilton 问题的一个“否”实例。这两个实例的解可以用 5 个节点的一个排列来表示，这个排列可以在计算机表示成一个字符串。回答一个实例为“是”或“否”可以通过对所有的解进行“验证”，如果存在一个解满足问题的提问要求（即可行解），这个实例就是“是”实例，否则为“否”实例。

例 1.5.2 三精确覆盖

已知集合$S=\{u_1,u_2,...,u_{3m}\}$的n个子集构成的子集族$F=\{S_1,S_2,...,S_n\}$，其中每个子集恰好包含S中三个元素，F中是否存在m个子集$S_{i_1},S_{i_2},\cdots,S_{i_m}$，使得$\bigcup_{j=1}^{m}S_{i_j}\supseteq S$？若存在$m$个子集$S_{i_1},S_{i_2},\cdots,S_{i_m}$满足$\bigcup_{j=1}^{m}S_{i_j}\supseteq S$，称$m$个子集$S_{i_1},S_{i_2},\cdots,S_{i_m}$覆盖$S$。本例中每个子集恰好包含$S$中三个元素，所以$m$个子集$S_{i_1},S_{i_2},\cdots,S_{i_m}$覆盖$S$时一定有$\bigcup_{j=1}^{m}S_{i_j}=S$，因此称为三精确覆盖。□

这是一个判定问题。对给定的一个实例，要验证给定的m个子集$S_{i_1},S_{i_2},\cdots,S_{i_m}$是否为S的一个覆盖。它的解是什么样的形式？S集合中共有$3m$个元素，按下标排列这些元素，此时，子集族F中的每一个元素$S_i(i=1,2,\cdots,n)$对应一个$3m$维向量。向量形式为：向量的$3m$个分量分别对应$3m$个元素，包含在对应子集中的元素对应的分量为1，余下的为0。例如，六个元素的集合S=$\{u_1,u_2,...,u_6\}$，F中的一个元素（即S的一个子集）$S_1=\{u_1,u_2,u_6\}$，对应向量为α_1=(1,1,0,0,0,1)，可以表示为一个字符串(110001)。F中m个元素$\{S_{i_1},S_{i_2},\cdots,S_{i_m}\}$是F的一个精确三覆盖的充分必要条件为字符串向量问题满足：相应m个字符串对应的向量$\alpha_{i_1},\alpha_{i_2},\cdots,\alpha_{i_m}$满足$\sum_{j=1}^{m}\alpha_{i_j}=(1,1,\cdots,1)$。

例 1.5.3 适定性问题(SAT)

在逻辑运算中，布尔变量 x 的取值只有两个："真"和"假"，逻辑运算有"或（+）"，"与（•）"和"非（¯）"。如果"真"用"1"表示，"假"用"0"表示，布尔运算遵循表1.5.3(a)(b)(c)。

表1.5.3(a) 运算"+"

布尔变量值		运算结果
0	0	0
0	1	1
1	0	1
1	1	1

表1.5.3(b) 运算"•"

布尔变量值		运算结果
0	0	0
0	1	0
1	0	0
1	1	1

表1.5.3(c) 运算"¯"

布尔变量值	运算结果
0	1
1	0

设 $\{x_1, x_2, \cdots, x_n\}$ 为 n 个布尔变量，布尔变量及其"非"组成的集合 $\{x_1, x_2, \cdots, x_n; \overline{x}_1, \overline{x}_2, \cdots, \overline{x}_n\}$ 称为文字集。文字集的任意一个子集中各元素的"或"运算组成一个句子。m个句子的"与"运算组成一个表达式。

如表达式

$$(x_1 + \overline{x_2}) \bullet x_2$$

的两个句子为 $x_1 + \overline{x_2}$ 和 x_2。当每一个逻辑变量取一定值时，采用逻辑运算关系，可以计算布尔表达式的值。在上式中，当x_1=1，x_2=1时，表达式为"真"。一组使表达式为"真"的变量取值称为真值分配。存在真值分配的表达式称为适定的。容易证明，表达式

$$(x_1 + \overline{x_2}) \bullet (\overline{x_1} + x_2) \bullet (x_1 + x_2) \bullet (\overline{x_1} + \overline{x_2})$$

不是适定的。

适定性问题定义为：

给定包含n个布尔变量的m个句子$c_1, c_2, \ldots, c_m$，布尔表达式$c_1 \bullet c_2 \bullet \ldots \bullet c_m$是适定的吗？□

这也是一个判定问题。它的任何一个实例的解可以用一个n维的0-1向量表示。0表示布尔变量取"假"，1表示取"真"。

总结上面3个判定问题会发现，它们每个实例的解在计算机中可以表示成占用一定空间的字符串。确定一个实例为"是"或"否"需要逐个验证解是否为"是"的回答。一旦存在一个解为"是"的解答，则说明这个实例为"是"实例。否则，要验证所有的解都为"否"的答案后才能回答这是一个"否"实例。

在处理组合最优化问题时，也分成两个阶段处理。首先，在给定某个目标值后，讨论优化问题对应的判定问题，然后再通过这个目标值的变化确定原问题的最优值。下面两个例子给出组合最优化问题对应的判定问题。

例 1.5.4 例 1.1.6 的线性规划问题(LP)对应的判定问题——LP 判定问题：

先给出一个目标值 z，将极小化目标函数的问题转化为：是否有可行解使目标函数值不超过这个 z？这个判定问题的数学表达式为：

$$\text{给定z，是否有 } \{x | c^T x \le z,\ Ax = b,\ x \ge 0\} \neq \varnothing ?$$

$x \in \{x | c^T x \le z,\ Ax = b,\ x \ge 0\}$ 称为对应线性规划判定问题的可行解。同样，整数线性规划（例1.1.3）也可以类似地写出对应的判定问题。□

例1.5.5 例1.1.2的TSP对应的判定问题。

用n个城市的一个排列$(i_1,i_2,\cdots,i_n)$表示商人从城市i_1出发依次通过$i_2,\cdots,i_n$，最后返回i_1这样一个路径。判定问题为：给定z，是否存在n个城市的一个排列W=$(i_1,i_2,\cdots,i_n)$，使得

$$f(W)=\sum_{j=1}^{n} d_{i_j i_{j+1}} \le z\ ?$$

其中$i_{n+1}=i_1$。满足$f(W)=\sum_{j=1}^{n} d_{i_j i_{j+1}} \le z$的一个排列$W$称为对应判定问题的一个可行解。□

除非特别声明，本节余下部分提到的优化问题都是指对应的判定问题。

对于判定问题，我们可以将求解该问题的算法分为两个阶段。第一阶段称为猜测阶段，它的主要工作是求解或猜测该问题的一个解。对于“是”实例，我们的目标是能够找到回答“是”实例的解。第二个阶段称为检查或验证阶段。一旦解选定，将猜测的解作为输入，验证此解是否为该实例“是”的回答。“是”的解也称为实例的可行解。确定一个实例是否存在可行解的难度是由问题本身所决定的。

在计算机计算的过程中，问题的每一个实例在计算机输入时都以字符串占据一定的空间，如例1.1.6的线性规划问题可以用一个长度为$O(mn+m+n+\log_2|P|)$的0-1字符串表示。实例的每一个解也同样可以用一个字符串表示，见例1.5.1、例1.5.2和例1.5.3的讨论。

NP问题

定义1.5.2 对一个判定问题，若存在一个多项式函数$g(x)$和一个验证算法A，满足：

I为判定问题“是”实例的充分必要条件：存在一个字符串S表示I的可行解，其规模$l(S)=O(g(l(I)))$，其中$l(I)$为I的规模，且算法A验证S为实例I的“是”答案的计算时间为$O(g(l(I)))$。

则称这个判定问题是非确定多项式的，简记为NP。在不混淆的情况下，用**NP**表示所有非确定多项式问题的集合。

由上面的定义可知，判定问题是否属于NP的关键是对“是”的判定实例，“存在”一个可行解字符串和验证算法，满足以上条件。我们不须回答“存在”的算法和可行解字符串是怎样得到的。定义1.5.2中两处用到多项式的限制，一处是“是”答案S的规模不超过$O(g(l(I)))$，另一处是算法A验证S为实例I的“是”答案的计算时间不超过$O(g(l(I)))$。可以将上面定义中这两处的多项式限制改变为：存在两个多项式函数$g_1(x)$和$g_2(x)$，使得“是”答案S的规模不超过$g_1(l(I))$，算法A验证S为实例I的“是”答案的计算时间不超过$g_2(l(I))$。因为只要选高阶的多项式作为定义1.5.2的$g(x)$就可以了，所以这样定义与原定义1.5.2等价。特别需要指出的是，对非“是”实例，我们不讨论它的相关判定的结果。

由**P**的定义1.2.2，**P**中包括优化问题和判定问题。我们同样可以讨论**P**中每个问题对应的判定问题，于是我们自然要问：**P**中的对应判定问题是否为非确定多项式问题？若在定义1.5.2中增加：存在多项式时间算法找到规模为$l(S)=O(g(l(I)))$的可行解字符串S，则称这个NP问题为P问题。由定义1.2.2，它的解的规模一定满足定义1.5.2。其次，需要考虑验证算法，这个算法就是原问题的算法，将得到的最优值同目前判定问题给定的目标值进行比较，就可以回答判定问题。这个验证算法也是一个多项式时间的算法，也满足定义1.5.2。由此可知**P**中的对应判定问题满足定义1.5.2。通常不严格区分一个多项式问题和它对应的判定问题，所以有**P**⊆**NP**。

下面用例1.5.6-1.5.8理解如何用定义确定一个判定问题是否属于**NP**。

例 1.5.6 有理系数的线性规划（LP）判定问题属于 **NP**。

例1.5.4已给出LP判定问题，它的一个解可由n个变量的取值决定，而且其中任何一个基本可行解的各分量满足下面的性质。下面用到线性规划的基本结论，主要包括：极点的概念、基本可行解的性质等，相关的内容可参考线性规划方面的论著(如[13])。

性质1.5.1 LP任何一个基本可行解的各分量有界，且其二进制字符串表达式的规模不超过 $2(2+mn+m+n+\log_2|P|)$，即基本可行解的规模可以被实例规模的多项式函数所控制，其中P为LP判定问题中所有非零参数的乘积。

证明：按假设LP的系数为有理数和 $m \le n$，因此只需通分就等价为LP的所有系数为整数。对LP的任何一个基本可行解，当分量 $x_j = 0$ 时，性质的结论显然成立。当基变量 $x_j \ne 0$ 时，由线性规划的结论，$x_j = B^{-1}b$，其中B为基矩阵。再由代数的基本结论 $B^{-1} = B^*/\det(B)$，其中 B^* 为B的伴随矩阵。在没有简约的情况下，知基变量x_j的分母为det(B)，分子为 $\sum_{i=1}^{m} b_i A_{ij}$，其中 A_{ij} 为B第i行j列的代数余子式。于是分子和分母分别满足

$$1 \le |\det(B)| \le m!|P|,$$

$$\left|\sum_{i=1}^{m} b_i A_{ij}\right| \le m!|P|.$$

由于 $m \le n$，$m < m+1 \le 2^m \le 2^n$ 和 $n < 2^n$，得到基本可行解的每一个分量值分子分母分别不超过

$$m!|P| \le 2^{nm}|P| = 2^{mn}2^{\log_2|P|},$$

所以，基本可行解的任何一个分量的规模（分别记录分子分母，一个除运算，符号位和隔离符）不超过 $2(mn+n+m+2+\log_2|P|)$。所以，基本可行解的规模不超过 $2n(mn+m+n+2+\log_2|P|)$。性质成立。□

对LP判定问题的一个“是”实例，由线性规划的结论，至少存在一个基本可行解。当给定LP的一个基本可行解x，只要验证是否满足 $\{cx \le z,\ Ax = b,\ x \ge 0\}$？验证时最多需 L_1=(m+1)n个乘，L_2=(m+1)(n-1)个加或减，L_3=n+m+1个比较。当 $m,n \ge 1$ 时，如此设计的验证算法计算时间不超过 $L_1 + L_2 + L_3 = 2mn + 3n$。

由例1.2.3的讨论，实例的规模$l(I)$至少是 $2(mn+m+n+2)+\log_2|P|$。再由上面的讨论，基本可行解的规模不超过 $n(2+mn+m+n+\log_2|P|)$，验证的计算量为 $2mn+3n$，于是，选 $g(x)=x^2$ 就可以满足定义1.5.2的要求。反之，由线性规划理论，任何一个基本可行解自然也是可行解。所以LP∈**NP**。□

例 1.5.7 TSP 问题属于 **NP**。

例 1.5.5 给出了 TSP 判定问题。由例 1.2.2 知，实例的规模 $l(I)$ 至少是 $2n(n-1)+2+\log_2|P|$，其中，$P = n\prod\{d_{ij} \mid d_{ij} \ne 0, i \ne j\}$。对任何一个$n$城市的TSP实例，它的解是(1,2,...,$n$)的一个排列W= $(i_1, i_2, \cdots, i_n)$，显然该字符串规模不超过 $\sum_{i=1}^{n}(\lceil \log_2(i+1) \rceil) + 2n < 3n + n\log_2 n$。上面的讨论表明，TSP任何一个解的规模不超过 $n(3+\log_2 n)$。

若给定TSP判定实例的一个可行解，只需验证是否满足

$$f(W)=\sum_{j=1}^{n} d_{i_j i_{j+1}} \le z\ ?$$

验证算法有n个加法和一个比较，算法的计算时间为n+1。于是，定义1.5.2中的多项式函数可选$g(x)=x$，验证算法的计算时间和判定问题实例可行解的规模都满足定义1.5.2的要求，故TSP∈NP。□

例 1.5.8 三精确覆盖属于 **NP**。

三精确覆盖问题任何一个实例的规模至少为n，不妨假定$n>m$。例1.5.2的讨论得到，三精确覆盖实例的一个解转化为一个字符串向量。精确三覆盖任何一个“是”实例的解可用规模为 $3m^2$ 个字符表示。验证 $\sum_{j=1}^{m}\alpha_{i_j}=(1,1,\cdots,1)$ 是否成立的算法需 $3m^2$ 个加法和$3m$个比较，算法的计算时间为 $3m^2+3m$。多项式函数可选$g(x)=x^2$，则定义1.5.2的条件满足，所以三精确覆盖属于**NP**。□

通过上面例子的讨论，发现**NP**问题类比多项式问题类**P**更广泛。**P**只是**NP**中的一部分，而且**NP**中还包含那些很难的问题，如TSP、三精确覆盖等问题。它们之间是什么样的关系？怎样找出不同问题之间的难度关系？

多项式时间转换

研究中常常采用这样一种思路，我们已经对一个问题有足够的了解，当遇到新的问题时，总要比较新问题同老问题之间的联系，如果能够将新问题化成老问题，就可以轻而易举地用解决老问题的方法解决新问题。我们称这种方法为归类或归约。处理判定问题也有类似的方法。

定义 1.5.3 给定问题 A1 和 A2，若存在一个多项式函数 $g(x)$和一个算法 T 满足：

（i）T 将 A1 问题的一个实例 I_1，计算得到 A2 问题的一个实例 I_2，且 I_2 为 A2 问题的一个“是”实例的充分必要条件是 I_1 为 A1 问题的一个“是”实例。（ii）T 是一个多项式算法，即从实例 I_1 求解得到 I_2 的规模为 $O(g(l(I_1)))$，其中 $l(I_1)$ 表示实例 I_1 的规模。

则称 A1 问题多项式转换(transformation)为 A2 问题。

定义 1 .5.3 中的算法 T 实际上是一种映射关系。将 I_1 在计算机存储的字符串映射到 I_2 的计算机存储字符串。算法的具体实现表现为：将 I_1 中的参数按一个算法规则运算成 I_2 的参数。在后续的例子中将加深对这个概念的理解。

一个直观的结论为：若 A1 问题多项式转换为 A2 问题且 A2∈**P**，则 A1∈**P**。求解 A1 的多项式时间算法为：多项式时间转换算法 T 加上 A2 的多项式时间算法。由 A2 的实例得到的“是”或“否”可以回答 A1 实例的“是”与“否”。

类似地，若 A1 问题多项式转换为 A2 问题且 A2∈**NP**，则 A1∈**NP**。

用下面例 1.5.9 和例 1.5.10 理解定义 1.5.3。

例 1.5.9 适定性问题多项式转换为整数线性规划问题。

将适定问题逻辑变量“真”对应“1”，“假”对应“0”，于是逻辑变量x对应整数变量，$\bar{x}$ 对应1-x。一个n个变量的句子 C_j (j=1,2,...,m)要求是“真”，对应下列不等式是否有可行解？

$$\sum_{x_i\in C_j} x_i+\sum_{x_i\in C_j}(1-x_i)\ge 1,\ \ j=1,2,\ldots,m,, \tag{1.5.3}$$

$$x_i\in\{0,1\},\ j=1,2,\ldots,n\ 。 \tag{1.5.4}$$

对于适定问题的任何一个实例，它是由n个逻辑变量和m个句子组成，按实例的定义，m个句子中包含那些文字是已知的数据，故适定问题实例的规模至少是max(m,n)。由(1.5.3)和(1.5.4)的映射（可以看成算法的运算），对应m组(1.5.3)和(1.5.4)形式的不等式是整数线性规划问题，对应整数线性规划实例的规模按例1.2.2的估计为$O(mn+m+n+m\log_2 n)$。因此，定义1.5.3中的多项式函数可选$g(x)=x^2$，就有定义1.5.3的(ii)成立。易证适定问题到对应(1.5.3)和(1.5.4)约束的整数规划问题满足(i)，说明适定问题多项式转换为整数线性规划问题。□

研究例1.1.1的0-1背包问题的一种特殊情形$c_j=a_j, j=1,2,\ldots,n$，我们可以给出一个判定问题：0-1背包判定问题

对给定的整数$c_j, j=1,2,\ldots,n$和b，是否存在{1,2,...,n}的子集B，使得$\sum_{j\in B}c_j=b$？

例 1.5.10 三精确覆盖问题多项式转换为 0-1 背包判定问题

由例1.5.2的S_j与二进制向量对应关系，一个二进制向量又可一一与一个整数对应：

$$S_j \longrightarrow c_j=\sum_{u_i\in S_j}(n+1)^{i-1}。\tag{1.5.5}$$

相当于使用了简单的乘法和加法运算，将一个向量一一对应于一个整数。选b为

$$(11\ldots1)\rightarrow b=\sum_{j=1}^{3m}(n+1)^{j-1}。\tag{1.5.6}$$

容易验证上面的对应关系满足定义1.5.3(ii)。由上面的构造，三精确覆盖实例的一个解$\{S_{i_1},S_{i_2},\cdots,S_{i_m}\}\in F$一一对应于背包问题实例的一个解$B=\{i_1,i_2,\cdots,i_m\}$。此时，$S$存在一个三精确覆盖$\{S_{j_1},S_{j_2},\ldots,S_{j_m}\}$的充要条件为$\sum_{i=1}^{m}c_{j_i}=b$。

充分性：假设存在$T\subseteq\{1,2,\ldots,n\}$，使得$\sum_{j\in T}c_j=b$。由(1.5.5)和(1.5.6)，$b$的每个指数项系数为1。又因为$T$的总项数不超过$n$，其同次幂项的系数和不超过$n$，因而，$n$+1的每个次幂项系数恰好为1。由(1.5.5)的反变换，得S_j，$j\in T$是S的一个三精确覆盖。

必要性：由(1.5.5)和(1.5.6)的定义，可得结论。

由上面的讨论，映射关系(1.5.5)和(1.5.6)满足定义 1.5.3(i)，所以，三精确覆盖多项式时间转换 0-1 背包判定问题。□

例 1.5.11 适定问题多项式转换为三精确覆盖问题

例1.5.3定义的适定性问题为：$x_1,x_2,\ldots,x_n$为逻辑变量，$C_1,C_2,\ldots,C_m$为句子。现在构造一个三精确覆盖问题，共分三个部分构造三精确覆盖的元素集合S。

第一部分对所有的逻辑变量及其“非”运算进行复制，对应每一句子复制一次，共2mn项，记成

$$U=\{x_i^j,\bar{x}_i^j\,|\,i=1,2,\ldots,n;\ j=1,2,\ldots,m\}。$$

第二部分包含三种类型的元素共2mn项。

$$V=\{a_i^j\,|\,i=1,2,\ldots,n;\ j=1,2,\ldots,m\}\cup\{v_j\,|\,j=1,2,\ldots,m\}$$
$$\cup\{c_i^j\,|\,i=1,2,\ldots,n-1;\ j=1,2,\ldots,m\}.$$

第三部分同第二部分的构造基本相同，

$$W=\{b_i^j|i=1,2,\ldots,n;\ j=1,2,\ldots,m\}\cup\{w_j|j=1,2,\ldots,m\}$$
$$\cup\{d_i^j|i=1,2,\ldots,n-1;\ j=1,2,\ldots,m\}.$$

于是，S中共包含6mn个元素。三元素子集族F仅由下列三部分子集族组成：

第一部分：由U，V，W中各选一个元素，组成$(x_i^j,a_i^j,b_i^j),i=1,2,\ldots,n;\ j=1,2,\ldots,m$或$(\bar{x}_i^j,a_i^{j+1},b_i^j),i=1,2,\ldots,n;\ j=1,2,\ldots,m$，其中，定义$a_i^{m+1}=a_i^1$。F中包含$a$和$b$类型的元素仅有上面这些组合形式，则对任意$i$，任何覆盖$a_i^j,b_i^j,j=1,2,...m$的所有三元素子集族中必是同由$a$和$b$类型的元素与$x_i^j,j=1,2,\ldots,m$组合，即$(x_i^j,a_i^j,b_i^j),j=1,2,\ldots,m$，或是同与$\bar{x}_i^j,j=1,2,\ldots,m$组合，即$(\bar{x}_i^j,a_i^{j+1},b_i^j),j=1,2,\ldots,m$。这一部分包含2$mn$个元素。

第二部分：由$\{(x_i^j,v_j,w_j)|x_i^j\in C_j\}\cup\{(\bar{x}_i^j,v_j,w_j)|\bar{x}_i^j\in C_j\},j=1,2,...,m$组成，其中，$x_i^j,\bar{x}_i^j$遍历句子$C_j$中的所有逻辑变量。因此，第二部分同第一部分仅在这一位置可能出现相同元素。这一部分最多包含2mn个元素。

由第一部分和第二部分的三元素集族的构造可以发现，第二部分以(v_j,w_j)为主构成，若其他三元子集合中不包含这些元素，一个S的三精确覆正好从$\{(x_i^j,v_j,w_j)|x_i^j\in C_j\}\cup\{(\bar{x}_i^j,v_j,w_j)|\bar{x}_i^j\in C_j\},j=1,2,...,m$中选出$m$个元素，记成$(y_j,v_j,w_j),j=1,2,\cdots,m$。根据第一部分子集族的构造，可以发现$\{y_j|j=1,2,\cdots,m\}$逻辑变量不互相矛盾，即若$y_p=x_i^p\in C_p,1\le p\le m$，则不存在$q$使得$y_q=\bar{x}_i^q\in C_q,1\le q\le m$。同理，若$y_p=\bar{x}_i^p\in C_p,1\le p\le m$，则不存在$q$使得$y_p=x_i^p\in C_p,1\le p\le m$。

由此可以发现，只要$\{y_j|j=1,2,\cdots,m\}$取“真”值，就是适定问题实例的真值答案。于是，任何一个S的三精确覆盖必须在第一部分集族中选mn个元素，在第二部分中选m个元素，余下m(n-1)个元素需要在下面第三部分中选取。

第三部分：$\{(y,c_i^j,d_i^j)|i=1,2,...,n-1,\ j=1,2,...,m,y\in U\}$。这一部分包含2$mn$($n$-1)$m$个元素。为了弥补前两部分在三精确覆盖选取中可能没有覆盖U集合中的一些元素，增加这一部分子集族。

三元子集族F中共有不超过$2m^2n^2-2m^2n+4mn$个子集元素。

假设适定性问题的一个实例由n个逻辑变量的m个句子组成，实例的规模由m个句子中的逻辑变量值确定，因此实例的规模为mn。上面已构造三精确覆盖问题一个特殊情况，这是一种映射，简单估计三精确覆盖问题实例的规模不超过$|S||F|$，其中$|S|$和$|F|$分别表示集合S和F中元素的个数，即不超过$6mn(2m^2n^2-2m^2n+4mn)$，定义1.5.3中的多项式函数可选为$g(x)=x^3$。

当三元素族F中存在S的一个三精确覆盖实例时，必包含第二部分的m组，取三精确覆盖中第二部分y_j的逻辑变量为“真”值，即，当$y_j=x\in C_j$时，x=1；$y_j=\bar{x}\in C_j$时，x=0。则对应适定性问题的一个“是”实例。反之，对适定问题的任何一个“是”实例，按上面的构造，对应三精确覆盖一个“是”实例，其三精确覆盖是：在F集合中的第二部分$(y_j,v_j,w_j),j=1,2,\ldots,m$中，取$y_j$为$C_j$中取值为“真”的逻辑变量；在$C_j$中逻辑变量$y_j$可能有两种形式$y_j=x_i$或$y_j=\bar{x}_i$，以$y_j=x_i$=1的情况来讨论（同法讨论$y_j=\bar{x}_i$=1的情况）：在F的第一部分中，取$\bar{x}_i^j(j=1,2,\ldots,m)$对应的所有三元组，至此，共选取($n$+1)$m$个三元组；最

后，在F的第三部分中，选取覆盖S中U，V，W的余下部分的$(n-1)m$个三元组。至此，适定问题可多项式转换为三精确覆盖问题。□

为了加深对多项式转换的理解，我们用直观的语言进一步对其描述。多项式转换是将问题 A1 的所有实例通过多项式时间的计算对应到 A2 问题中的一部分实例，并不一定是 A2 中的所有实例。然后，通过 A2 的算法对这些来自 A1 对应的实例求解。如果求解 A2 映射后的一个实例为“是”，那么，A1 中对应的实例为“是”，否则，A1 中对应的实例为“否”。

A2 的算法功能可能很强，适用范围可能很广，能够求解 A2 中的所有实例。求解 A1 对应过来的这一部分也许只是“小试牛刀”。正因为这样，若 A2 为多项式可解，那么，A1 也就多项式可解。A1 的求解算法是先进行多项式转换，然后调用一次 A2 的算法即可。

同样的思想，可以构造求解 A1 的算法如下：求解 A1 实例的过程中，将 A2 的算法看成 A1 的一个子程序多次调用。这样的算法在计算 A1 的一个实例的过程中，每调用一次 A2 的算法，需要先将 A1 这个实例计算成为一个 A2 的实例，然后才可以调用 A2 的算法。如果将调用 A2 算法的每一次计算看成一个单位，当存在一个多项式，使得这样的算法求解 A1 的任何一个实例的计算量可由这个多项式关于 A1 实例规模的函数值控制时，称问题 A1 多项式归约(reduction)为 A2 问题。

定义 1.5.4 当满足下列条件时，称判定问题 A1 多项式时间归约为 A2：

存在一个 A1 的算法和多项式函数 $g(x)$，算法求解 A1 任何实例 I 的过程中，将 A2 的算法作为子程序多次调用。如果将一次调用 A2 算法看成一个单位（1 次基本计算量），则这个 A1 算法的计算量为 $O(g(l(I)))$，其中 $l(I)$ 为实例 I 的规模。

从定义 1.5.4 看出，若 A2 存在多项式时间算法，设 A2 多项式算法对应的多项式控制函数为 $g_2(x)$，则存在 A1 的一个算法，每一次在调用 A2 算法之前，需要将 A1 的实例计算得到 A2 实例，这个 A2 实例的规模为 $O(g(l(I)))$，再调用 A2 算法的计算时间为 $O(g_2(g(l(I))))$。最多调用 A2 算法的次数及其它计算量为 $O(g(l(I)))$。因此，存在 A1 问题的算法，其计算时间 $O(g(l(I))g_2(g(l(I))))$，是 A1 的一个多项式时间算法。

因为多项式转换只一次调用 A2 的算法，所以是一种特殊的多项式归约。因此，多项式归约比多项式转换更广泛。

NP 完全和 NP 难

Cook[14]在 1971 年给出并证明了有一类问题具有下述性质：(1)这类问题中任何一个问题至今未找到多项式时间算法；(2)如果这类问题中的一个问题存在有多项式时间算法，那么这类问题都有多项式时间的算法，这类问题中的每一个问题称为NP完全（NP-Complete），这个问题的集合简记**NPC**。下面给出定义。

定义1.5.5 如果判定问题A满足：A∈**NP**且**NP**中的任何一个问题都可在多项式时间内归约为A，则称A为NP完全。若**NP**中的任何一个问题都可在多项式时间归约为判定问题A，则称A为**NP**难（NP-hard）。简记NP难问题的集合为**NPH**，于是显然有**NPC**⊆**NPH**。

NP完全和NP难问题的区别是NP难问题无须判断A是否属于**NP**。验证一个问题A是否为NPC的关键有两点，一是NP中的任何一个问题是否可在多项式时间内归约为A，其次，是否存在一个字符串，其规模为实例规模的多项式函数，以及是否存在一个多项式时间的验证算法。

由多项式转换和归约，NP完全，NP难等的定义，当我们已知一个问题为NP完全或NP难时，再遇到一个新问题时，只要已知问题可以多项式归约为新问题，则知新问题是NP难。当可以验证新问题是属于NP时，则该问题属于NPC。由此可知，研究一个新问题的复杂性，往往借助于已知问题的复杂性和多项式归约。通过下面例子可以了解这一过程。

例 1.5.12 适定性问题是 NP 完全。

证明可见[14]。□

例1.5.13 Hamilton回路是NP完全。

证明可见[10]。□

我们认为上面两个问题的NP完全性结论是已知的，如果可能的话，将这两个问题归约到所研究的问题，所研究的问题的复杂性也就由此而得到，这是一种比较典型的研究方法。

例 1.5.14 整数线性规划的判定问题是 NP 完全。

按例1.5.4的说明，线性整数规划的判定问题类似LP的判定问题。例1.5.6证明LP的判定问题属于NP，整数线性规划的判定问题属于NP的证明与LP判定问题属于NP的证明类似，比较复杂，可以参考[10](p.320，Theorem13.4)。例1.5.9已证明适定问题多项式转换为整数线性规划的判定问题，所以，整数线性规划的判定问题是NP完全。□

例 1.5.15 三精确覆盖是 NP 完全。

由例1.5.8和例1.5.11得结论。□

例 1.5.16 背包判定问题是 NP 完全。

背包问题的解对应一个n维0-1分量的向量，于是，很易验证属于NP。例1.5.10已证明三精确覆盖多项式转换为背包判定问题，所以，背包判定问题属于NPC。□

背包判定问题的一个特殊情况，包的容积$K=\frac{1}{2}\sum_{j=1}^{n}c_j$，此时，对应一类新的问题——集合划分(partition)问题：

给定整数$c_1,c_2,\ldots,c_n$，是否存在{1,2,...,n}的一个子集S，使得$\sum_{j\in S}c_j=\sum_{j\notin S}c_j$？

例 1.5.17 集合划分问题是 NP 完全。

因背包问题属于NP，所以，集合划分问题属于NP。例1.5.16已证明背包判定问题属于NPC，只须证明，背包问题可以多项式转换为集合划分问题。

对给定的0-1背包判定问题的任何一个实例，$c_1,c_2,\ldots,c_n,b$，构造集合分划问题的实例$c_1,c_2,\ldots,c_n,c_{n+1}=2M,c_{n+2}=3M-2b$，其中，$M=\sum_{j=1}^{n}c_j>b$。显然，这个映射满足定义1.5.3的(ii)。

存在(1,2,...,n)的子集S，使得$\sum_{j\in S}c_j=b$的充分必要条件是存在{1,2,...,n+2}的一个子集S'使得$\sum_{j\in S'}c_j=\sum_{j\notin S'}c_j$。下面证明定义1.5.3的(i)满足。

充分性：由于5M-2b>$M=\sum_{j=1}^{n}c_j$，因此存在{1,2,...,n}的一个子集S，使得

$$\sum_{j\in S}c_j+c_{n+2}=\sum_{j\notin S,\,j\neq n+1,n+2}c_j+c_{n+1}。\tag{1.5.7}$$

所以，计算上式可得$\sum_{j\in S}c_j=b$。

必要性：由(1.5.7)，令S'=S∪{n+2}，得结论。□

例1.1.4装箱问题的判定问题可以叙述为：给定K及n个物品，尺寸为$0<b_1,b_2,\ldots,b_n\le 1$，是否能在$K$个尺寸为1的箱子中装入这$n$个物品？

例 1.5.18 装箱判定问题是 NP 完全.

装箱判定问题属于NP易证。我们通过集合划分问题多项式转换为装箱判定问题而证明结论。由例1.5.17的证明，当集合划分问题的所有整数为正时，问题还是NP完全。对集合划分问题的任一实例：n个正整数$c_1,c_2,\ldots,c_n$，映射装箱问题的一个判定实例，计算

$$b_i=\frac{2c_i}{\sum_{j=1}^{n}c_j},i=1,2,\ldots,n\text{。}\tag{1.5.8}$$

当存在$b_i>1$时，有$c_i>\frac{\sum_{j=1}^{n}c_j}{2}$，说明是集合划分的一个“否” 实例。这一类的实例可以在多项式时间内解决。因此，去掉这些实例不影响集合划分问题的复杂性。因此，不妨设$b_i\le 1$为第 i 个物品的尺寸，箱子尺寸 1，是否能在两个箱子（即 K=2）内装完所有物品？显然(1.5.8)的映射满足定义 1.5.3 的(ii)。现验证定义 1.5.3 的(i)满足。需要证明下列推断：存在{1,2,...,n}的一个子集 S 使$\sum_{j\in S}c_j=\sum_{j\notin S}c_j$ 的充分必要条件是可在两个尺寸为 1 的箱子里装入所有的 n 个物品。

充分性：因为$\sum_{j=1}^{n}b_j=2$，即每个箱子都无空隙。记第一个箱子中的物品集为S，有$\sum_{j\in S}c_j=\sum_{j\notin S}c_j=1$。

必要性：明显。□

例1.1.5的约束单机排序问题对应的判定问题是：给定$d_i,i=1,2,\ldots,n$，c_t, t=1,2,...和T，在模型的约束条件下，能否在T个工作日内完成n个产品的加工？

例 1.5.19 约束单机排序的判定问题是 NP 完全.

约束单机排序的判定问题属于NP是很易验证的。当c_t(t=1,2,...)全部相同时，便成为装箱问题的一种特殊情况，因此，装箱问题就多项式转换为约束单机排序问题的这一特殊情况。所以，结论成立。□

例 1.5.20 TSP 判定问题是 NP 完全。

例 1.5.7 已经证明 TSP 的判定问题属于 NP。已知例 1.5.13 的 Hamilton 回路问题为 NP 完全。做下面的映射：对 Hamilton 回路问题的任何一个实例，它的节点对应 TSP 的城市，TSP 城市间的距离定义为：

$$d_{ij}=\begin{cases}1, & \text{若Hamilton节点}(i,j)\text{之间有边相连},\\ 2, & \text{其他}。\end{cases}$$

TSP 的一个判定实例为：是否存在一个 TSP 回路，使得它的距离不超过 n?

可以简单地验证，上面的映射满足定义 1.5.3，所以 Hamilton 问题多项式转换为 TSP 问题，也就得到 TSP 为 NP 完全。□

由于 NPC 里包含很多著名的组合最优化问题，经过几代数学家的努力，迄今没有找到多项式时间算法，人们猜想：NPC 中的任何一个问题没有多项式时间算法，即$P\bigcap NPC=\varnothing$。

线性规划问题与其判定问题的关系

在上面的复杂性讨论中，特别是定义1.5.3中，我们只关注判定问题。一个求解判定问题的算法只需回答求解实例为“是”或“否”，并不要求解出“是”实例的可行解。这与我们通常对优化的理解是有一定差距的。

复杂性分析中所用到的针对判定问题的算法是否对求优化问题最优解本身有帮助？对于优化问题，可以考虑将求解过程分解成两个阶段：第一阶段是给定目标值的上界（假设是极小化问题）后求解判定问题，第二阶段是通过目标值的上界变化确定最优值。

例如，对于线性规划问题，可以将优化问题的算法同判定问题复杂性分析的算法成功地结合在一起。下面不妨考虑线性规划问题的标准型：

线性规划问题（LP）：

$$\begin{aligned} &\min c^T x \\ &s.t.\ Ax = b \\ &\quad x \geq 0 \end{aligned}$$

判定问题（DP）：

给定整数z，是否有

$$\{x \mid c^T x \leq z, Ax \geq b, Ax \leq b, x \geq 0\} \neq \phi ?$$

原问题LP同判定问题DP的联系是z：如果原问题可以求解，则显而易见地回答了判定问题。反过来如何？在书[10]中，通过对线性规划基础可行解每一分量、目标值的估计，得到结论：LP存在多项式时间最优算法的充分必要条件是DP为多项式时间问题。在存在最优解时，用二分法给出目标值的上界，并通过二分法的逼近求出最优值。二分法迭代的次数是实例规模的多项式函数。下面给出严格的证明。

性质1.5.2 假设x^1和x^2是LP的两个基本可行解，并且对某一个整数K满足

$$K2^{-2L} < c^T x^1, c^T x^2 \leq (K+1)2^{-2L},$$

其中，$L = mn + m + n + 2 + \log_2 |P|$，$P$为$A,b,c$中非零分量的乘积，则有$c^T x^1 = c^T x^2$。

证明(反证)：由性质 1.5.1 的证明，每一个基本可行解分母的绝对值不超过2^L。若$c^T x^1 \neq c^T x^2$，有$|c^T x^1 - c^T x^2| \geq 2^{-2L}$，此时同性质 1.5.2 的条件矛盾。□

由性质 1.5.1，所有基本可行解的各个分量的分子不超过2^{L_1}，其中$L_1 = mn + m + n + 2 + \log_2 |P_1|$，$P_1$为 A 和 b 中非零数的乘积。对应基本可行解的目标值为

$$\left| \sum_{j=1}^{m} c_j x_j \right| \leq \sum_{j=1}^{m} |c_j x_j| \leq \sum_{j=1}^{m} 2^{\log_2(|c_j|+1)+L_1} \leq 2^{\log_2 m + \log_2 |P_2| + 1 + L_1} \leq 2^{2L}, \qquad (1.5.9)$$

其中，P_2表示 c 中的非零数乘积。由基本可行解性质，不为零的分量最多有 m 个，因此(1.5.9)的求和项数为 m。

由(1.5.9)，任何一个基本可行解的目标值落入区间$(-2^{2L}, 2^{2L})$内，考虑区间$[-2^{4L}, 2^{4L}]$内的整数点 K，分下面三种情况讨论。

情况 1：当$K = 2^{4L}$，取 $z = K2^{-2L} = 2^{2L}$时，若 DP 是一个“否”实例，则 LP 无可行解。

情况 2：当$K = -2^{4L}$，取$z = K2^{-2L} = -2^{2L}$时，若 DP 还是一个“是”实例，则 LP 无下界。

情况 3：以上两种情况都不满足时，此时可以用下面的二分算法：

二分算法

STEP1 $u=-2^{4L}, v=2^{4L}$； $K=\dfrac{u+v}{2}=0$；

STEP2 当$v-u\le 1$时停止，记录此时的（u,v,K）为（u^*,v^*,K^*）；否则取$z=K2^{-2L}$，若DP为“是”实例，则令$u:=u, v:=K$；若DP为“否”实例，则令$u:=K, v:=v$；$K=\dfrac{b+a}{2}$，重复STEP2。

第一和第二种情况说明在第三种情况时一定有有界的可行解。初始$u=-2^{4L}, v=2^{4L}$的选择保证二分算法每步的K一定为整数。二分算法结束时一定有$v^*-u^*\le 1$，当$z=v^*2^{-2L}$时，判定问题DP为“是”实例，而当$z=u^*2^{-2L}$时，判定问题DP为“否”实例，性质1.5.2告知当$z=v^*2^{-2L}$且判定问题DP为“是”实例时，最优目标值唯一，在区间$(u^*2^{-2L}, v^*2^{-2L}]$内。二分算法的迭代次数为$\log_2 2^{4L+1}=4L+1$。

若判定问题有多项式时间算法，对情况 3，求解到最后区间$(u^*2^{-2L}, v^*2^{-2L}]$的计算量不超过 4L+1 倍的求解对应DP实例的计算时间。因此，这是一个多项式时间算法。此时，只得到最优目标值z^*在区间$(u^*2^{-2L}, v^*2^{-2L}]$内，仍然无法确定最优解。确定最优解的方法是继续求解判定问题

$$D_j=\{x\mid c^Tx\le v^*2^{-2L}, Ax\ge b, Ax\le b, x\ge 0; x_j\le 0, x_i\le 0, i\in S(j)\}\ne\Phi?$$

其中，$S(1)=\Phi$，$S(j)=\{i\mid i<j, D_i\ne\Phi\}$。由此，迭代 n 次后，得到$S(n+1)$。由线性规划的理论，$S(n+1)$ 以外的变量为基变量(至多为 m 个)，所以可以得到基矩阵（如果这样得到的基变量个数少于 m 个，可很容易地将这些基变量在 A 中对应的列向量进行必要的扩充）。再用线性方程组的理论可以求出唯一解。上面这些计算也是实例规模的多项式时间的。所以，只要 LP 的判定问题有多项式时间算法，则 LP 优化问题也有多项式时间算法。总结为：

定理 1.5.1 LP 有多项式时间最优算法的充分必要条件是 DP 有多项式时间算法。

对于组合最优化问题和其判定问题：

组合最优化问题（OP）：

$$\min f(x)$$
$$s.t.\ g(x)\ge 0$$
$$x\in D$$

判定问题（RP）：

给定整数L，是否有

$$\{x\mid f(x)\le L, g(x)\ge 0, x\in D\}\ne\phi?$$

若判定问题是NP完全或NP难，则知优化问题的难度不低于判定问题，此时，称组合最优化问题为NP难。

注：NP完全问题还可以进一步细分为成两类。一类NP完全问题虽然没有找到多项式时间算法，但存在一个算法，它的复杂度关于实例规模和实例的所有参数中绝对值最大数是成多项式关系的，这样的算法称为问题的一个伪多项式时间算法 (pseudo-polynomial time algorithm)。这类问题在NPC中属于比较简单的。除去这些简单的以外，还有一类更难，称为强NPC(strongly NP-complete)。感兴趣的读者可以参考1.6节的进一步讨论。

目前在学术界，大家公认的假设是：

假设 $P\ne NP$。

这个假设到目前为止，还无法从理论上证明。虽说这个假设是组合最优化研究的有待研究的一个热点，人们一般都接受这个假设！

算法复杂性的四类问题关系可用示意图1.5.2表示。

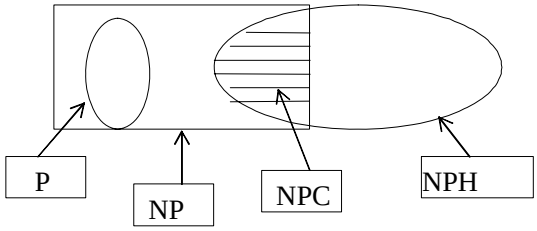


图1.5.2 算法复杂性示意图

示意图1.5.2表示P NP，NPC NPH，NP与NPH的公共部分为NPC。在NP中，除P和NPC，还有一部分问题的复杂性是未知的。按NPC的定义容易证明：若 $P \bigcap NPC \neq \varnothing$，则NPC=P。有关算法及复杂性的理论的更详细讨论，可以参考[5]。

1.6 多项式时间迫近格式

1.4 节和 1.5 节介绍了启发式算法和算法复杂性分析的一些基本概念。本节将结合这两个概念，介绍多项式时间迫近（或近似）格式（Polynomial-time approximation scheme），简记为 PTAS。

首先，研究例 1.5.10 的 0-1 背包问题：

对给定的整数 $c_j, j=1,2,\ldots,n$ 和 b，是否存在{1,2,...,n}的子集B，使得 $\sum_{j\in B} c_j = b$？

由复杂性理论的假设，所有系数为整数。不妨设 $c_j, j=1,2,\ldots,n$ 和 b 都是正整数且 $b \geq c_j, j=1,2,\ldots,n$，于是可以用图的形式表示 0-1 背包问题：图中由 b+1 个节点 $V=\{0,1,2,\cdots,b\}$，和满足下列条件的有向弧集 $A=A_1 \cup A_2 \cup \cdots \cup A_n$ 组成，其中：$A_j = \{(u,v) \mid u,v \in V, v-u=c_j\}$。

例1.6.1 0-1背包问题实例 $c_1=3, c_2=2, c_3=1, c_4=4, b=7$ 的图形式（图1.6.1）。

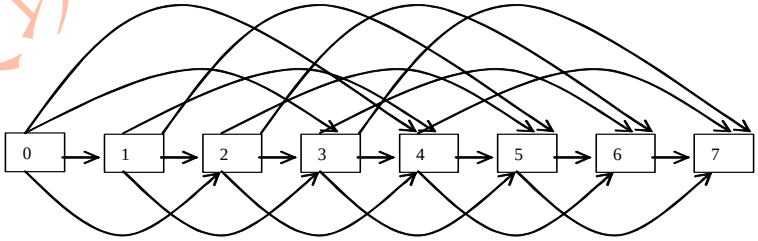


图 1.6.1 0-1 背包实例的图形式

图表达为 G=(V,A)，其中 $V=\{0,1,2,\cdots,7\}$，A={(0, 3)，(1, 4)，(2, 5)，(3, 6)，(4, 7)，(0, 2)，(1, 3)，(2, 4)，(3, 5)，(4, 6)，(5, 7)，(0, 1)，(1, 2)，(2, 3)，(3, 4)，(4, 5)，(5, 6)，(6, 7)，(0, 4)，(1, 5)，(2, 6)，(3, 7)}。□

从图中可以看出，0-1 背包问题的实例为“是”的充分必要条件为存在一条从节点 0 到达节点 b 且每一 $A_j(j=1,2,\cdots,b)$ 中最多选取一条弧的有向路径，如图 1.6.1 中的 0→3→7。继而，求解 0-1 背包问题等价为求解对应图的从 0 到 b 的一条满足一定性质的有向路径问题。

如果赋予图中每一条有向弧$\langle i, j\rangle$一个路长$d_{ij}=j-i$，令$d_0^0=d_1^0=\cdots=d_b^0=0$，从$j$=1 到 n，对每一个 j 分别从 v=1 到 b 计算

$$d_v^j=\begin{cases}\max\{d_v^{j-1}, d_{v-c_j}^{j-1}+c_j\}, v\geq c_j\\ d_v^{j-1}, v<c_j.\end{cases}\tag{1.6.1}$$

则得到下面的结论。

定理 1.6.1 0-1 背包问题的实例为“是”的充分必要条件为$d_b^n=b$。

证明：可以归纳证明$d_v^j\leq v(\forall j,v)$，且$d_v^j(\forall v)$关于j是非减的。

对 0-1 背包问题的一个“是”实例，存在$S=\{j_1,j_2,\cdots,j_k\}$使$\sum_{j\in S}c_j=b$，不妨设$1\leq j_1\leq j_2\leq\cdots\leq j_k\leq n$。此时必有：

$$d_{c_{j1}}^{j_1}=\max\{d_{c_{j1}}^{j_1-1}, d_0^{j_1-1}+c_{j_1}\}\geq c_{j_1},$$

$$d_{c_{j_1}+c_{j_2}}^{j_2}=\max\{d_{c_{j_1}+c_{j_2}}^{j_2-1}, d_{c_{j_1}}^{j_2-1}+c_{j_2}\}\geq d_{c_{j_1}}^{j_2-1}+c_{j_2}\geq d_{c_{j_1}}^{j_1}+c_{j_2}\geq c_{j_1}+c_{j_2},$$

$$\cdots,$$

$$d_b^{j_k}=d_{c_{j_1}+c_{j_2}+\cdots+c_{j_k}}^{j_k}\geq c_{j_1}+c_{j_2}+\cdots+c_{j_k}=\sum_{j\in S}c_j=b.$$

因此，$d_b^n=b$，必要性成立。

反过来，若$d_b^n=b$，根据已有的 nb 个数$\{d_v^j\mid j=1,2,\cdots,n, v=1,2,\cdots,b\}$，设$j_k$为第一个满足$d_b^{j_k}=b$的数，则必有$d_{b-c_{j_k}}^{j_k-1}=b-c_{j_k}$。否则，$d_{b-c_{j_k}}^{j_k-1}<b-c_{j_k}$，由$d_b^{j_k-1}<b$和(1.6.1)推出矛盾。同理，可得到$j_{k-1}>j_{k-2}>\cdots>j_1$,，满足

$$d_{b-c_{jk}-\cdots-c_{j_{i+1}}}^{j_i}=b-c_{j_k}-\cdots-c_{j_{i+1}}, d_{b-c_{jk}-\cdots-c_{j_{i+1}}}^{j_i-1}<b-c_{j_k}-\cdots-c_{j_{i+1}}, i=k-1,k-2,\cdots,1,$$

于是有

$$d_{b-c_{jk}-\cdots-c_{j_i}}^{j_i-1}=b-c_{j_k}-\cdots-c_{j_i}, i=k-1,k-2,\cdots,1,$$

且

$$d_{b-c_{jk}-\cdots-c_{j_1}}^{j_1-1}=b-c_{j_k}-\cdots-c_{j_1}=0,$$

则背包中装入$\{j_1,j_2,\cdots,j_k\}$为问题“是”实例的一个解。充分性得证。□

将上面算法应用例 1.6.1，由下列计算结果：

j=0：$d_0=d_1=\cdots=d_7=0$

j=1：$d_1=d_2=0, d_3=d_4=\cdots=d_7=3$,

j=2：$d_1=0, d_2=2, d_3=d_4=3, d_5=d_6=d_7=5$,

j=3：$d_1=1, d_2=2, d_3=3, d_4=4, d_5=5, d_6=d_7=6$,

j=4：$d_1=1, d_2=2, d_3=3, d_4=4, d_5=5, d_6=6, d_7=7$。

由此可以看出，这是一个“是”实例。

对应每一个固定的 j，v 从 1 变化到 b，(1.6.1)的计算最多需要 $3b$ 次比较或加法运算，因此算法的计算复杂性为 $O(nb)$，总结为下面的定理。

定理 1.6.2 参数为 $c_j, j=1,2,\ldots,n$ 和 b 的 0-1 背包问题可在 $O(nb)$ 时间内求解。

这样的结果引出一个有趣的现象。在 1.5 节中，例 1.5.10 和例 1.5.16 已经证明 0-1 背包问题是 NP 完全，现在又得到一个计算复杂性为 $O(nb)$ 的算法。研究现有的结果发现，这个算法与 0-1 背包问题实例中的最大数 b 有关，b 的规模为

$$L=\lceil \log_2(b+1)\rceil+2>\log_2 b+2\ 。$$

多项式时间算法要求计算复杂性是 n 和 L 的多项式函数，但这个算法的计算复杂性为 $O(n2^L)$，因此不是多项式算法。但这个算法也不错，只要控制 b 不太大，就在可以忍受的时间内得到答案。

定义 1.6.1 I 为一个问题的实例，I 中出现的最大数记为#(I)。

0-1 背包判定问题任何一个实例的参数为 $\{c_j, j=1,2,\ldots,n$ 和 $b\}$，所以，$\#(I)=\max\{c_j, j=1,2,\cdots,n;\ b;\ n\}$。对例 1.5.5 的 TSP 判定问题，它的任何一个实例有 $\#(I)=\max\{d_{ij}, i,j=1,2,\cdots,n;\ \ z;\ \ n\}$。例 1.5.4 的 LP 判定问题 $\#(I)=\max\{a_{ij}, i=1,2,\cdots,m, j=1,2,\cdots,n;\ b_i, i=1,2,\cdots,m;\ c_j, j=1,2,\cdots n;\ z;\ \ m;\ n\}$。

定义 1.6.2 对于问题的一个算法 A，如果存在 2 元多项式函数 $g(x,y)$，使得此算法求解任何实例 I 的计算复杂性 $C_A(I)=O(g(l(I),\#(I)))$，则称这个算法是伪多项式的(pseudo-polynomial)，其中 $l(I)$ 为实例 I 的规模.

0-1 背包判定问题存在伪多项式算法。

定义 1.6.3 若存在一个多项式 $g(x)$，当一个问题的所有满足条件 $\#(I)=O(g(l(I)))$ 的实例组成的子问题是 NP 完全时，则称这个问题是强 NP 完全(strongly NP-complete)，其中 $l(I)$ 为实例 I 的规模。

从例 1.5.20 的证明中得到，限定 $z=n$，城市间的距离为 1 或 2 的 TSP 仍然为 NP 完全，所以 TSP 是强 NP 完全。1.5 节中的整数规划判定问题，约束机器排序对应判定问题，3 精确覆盖，Hamilton 回路等为强 NP 完全。装箱判定问题，集合划分问题不是强 NP 完全。

例 1.5.10 将 3 精确覆盖多项式转换到 0-1 背包判定问题，3 精确覆盖是强 NP 完全，而 0-1 背包判定问题则不是，难道多项式转换后的问题不比转换前的问题难？这是为什么？从强 NP 完全的观点来看，多项式转换是否不再能说明转换后的问题难度不低于转换前问题？实际上，这取决于多项式转换算法的具体构造。在例 1.5.10 的转换中，用到

$$S_j \longrightarrow c_j=\sum_{u_i\in S_j}(n+1)^{i-1},$$

即将一个 3m维的 0-1 向量对应于一个指数形式表达的整数。可以发现，其中的一些整数（如 c_n和b）不再能被m,n的多项式函数所控制。这正是关键所在。因此，在问题的强NP完全的证明中，如果采用多项式转换，要特别避免这种情况出现。

定理 1.6.3 除非 P=NP，强 NP 完全问题不存在伪多项式时间算法。

证明（反证法）：假设一个强 NP 完全问题存在一个伪多项式时间算法，记算法计算量为 $O(g(l(I),\#(I)))$，其中 $l(I)$ 为实例的规模。对任意一个多项式 $f(x)$，当限定问题为 $\#(I)=O(f(l(I)))$ 的子问题时，该算法以一个多项式时间 $O(g(l(I),f(l(I)))$ 解决了子问题。除非 P=NP，这是不可能的。□

继续求解 0-1 背包优化问题：

$$\begin{aligned} z^* &= \max \sum_{i=1}^{n} c_i x_i \\ s.t.\ &\sum_{i=1}^{n} a_i x_i \le b \\ & x_i \in \{0,1\},\ j = 1,\ldots,n. \end{aligned} \tag{1.6.2}$$

同样假设该问题所有参数为正整数，采用如下算法 A：

Step 0：设 $S_0 = \{(\varnothing, 0)\}$，

Step 1：从 $j = 1,2,\cdots,n$ 分别计算

Step1.1 $S_j = \{(\varnothing, 0)\}$；

Step1.2 $S_j = S_{j-1} \cup \left\{ (S \cup \{j\}, z + c_j) \mid (S,z) \in S_{j-1} \text{且} \sum_{i \in S} a_i + a_j \le b \right\}$；

Step1.3 在 S_j 中按目标值 z 检查：若有 $(S,z),(T,z) \in S_j$，且 $\sum_{i \in S} a_i \le \sum_{i \in T} a_i$，则从 S_j 中删除 (T,z)。

Step 2： 找出 S_n 中具有最大目标值 z^* 的 (S, z^*)，S 和 z^* 分别为(1.6.2)的一个最优解和最优目标值（证明见习题 14）。

首先，理解算法 A 的主要计算过程。只考虑前 j 个物品装包时的所有可行解，用集合 S_j 表示这些可行解。从 Step1.2 可以看出 S_{j-1} 自然是 S_j 的一部分，再考虑 S_{j-1} 中的元素（物品）同物品 j 结合后是否为可行解？如果仅有 Step1.2 而没有 Step1.3 的计算，则 S_j 中元素的个数可能为 $O(2^j)$，则计算出 S_n 的复杂性为 $O(2^n)$。这是一个指数算法。

考虑 Step1.3 后，每次计算可将 z 从小到大排列，顺序检查，删除满足条件的一些可行解。于是 S_j 中的元素个数不超过 S_j 中目标值的最大数，也就不超过 z^*。

综合考虑 Step1.2 和 Step1.3，对每一个固定的 j，Step1.2 对每一个 z 的计算或生成、记忆 S_j 需要 $O(n)$ 的计算量。同时，由 Step1.3，每一个 z 只对应 S_j 中一个元素。当 Step1.2 计算得到 S_j 中所有元素 (S,z) 后，比较 z 可能对应的其他 (S',z)，按 Step1.3 的比较去掉一个元素即可。因此，Step1.2 和 Step1.3 对一个 j 的计算量为 $O(nz^*)$。A 算法的总计算复杂性为 $O(n^2 z^*)$。

算法 A 为(1.6.2)的一个伪多项式时间算法。用动态规划的方法，可以在 $O(nb)$ 的时间内求(1.6.2)的最优解（见习题 15）。算法 A 的计算时间显然不比动态规划方法好，那么，算法 A 的优点是什么？观察下面的例子。

例 1.6.2 用算法 A 计算如下 0-1 背包问题实例：

$a_1 = 1, a_2 = 1, a_3 = 4, a_4 = 3, c_1 = 20, c_2 = 10, c_3 = 31, c_4 = 20, b = 5$。

解：$S_0 = \{(\varnothing, 0)\}$。

$j = 1, S_1 = \{(\varnothing,0),(1,20)\}$，

$j=2, S_2=\{(\varnothing,0),(1,20),(2,10),(\{1,2\},30)\}$,

$j=3, S_3=\{(\varnothing,0),(1,20),(2,10),(\{1,2\},30),(3,31),(\{1,3\},51),(\{2,3\},41)\}$。

这一步中，{1, 2, 3}的组合因超出包容量被删除。

$$j=4, S_4=\left\{\begin{array}{l}(\varnothing,0),(1,20),(2,10),(\{1,2\},30),(3,31),\\(\{1,3\},51),(\{2,3\},41),(\{1,4\},40),(\{1,2,4\},50)\end{array}\right\}。$$

这一步中，{3, 4}，{1, 3, 4}，{2, 3, 4}的组合因超出包容量被删除，(4, 20)和(2, 4)因同目标值相同而占用包体积过大被删除。最终得到最优解为 S={1, 3}，最优值为 51。

简单修改上面的实例为：

$a_1=1, a_2=1, a_3=4, a_4=3, c_1'=20, c_2'=10, c_3'=30, c_4'=20, b=5$。

计算过程为：

$S_0'=\{(\varnothing,0)\}$。

$j=1, S_1'=\{(\varnothing,0),(1,20)\}$,

$j=2, S_2'=\{(\varnothing,0),(1,20),(2,10),(\{1,2\},30)\}$,

$j=3, S_3'=\{(\varnothing,0),(1,20),(2,10),(\{1,2\},30),(\{1,3\},50),(\{2,3\},40)\}$。

这一步中，{1, 2, 3}的组合因超出包容量被删除。(3, 30) 因同目标值相同而占用包体积过大被删除。

$j=4, S_4'=\{(\varnothing,0),(1,20),(2,10),(\{1,2\},30),(\{1,3\},50),(\{2,3\},40)\}$。

最优目标值为 50。□

观察 S_4 和 S_4'，发现存储量已经有变化，后一种情况存储量较前一种情况少，但两个问题得到的目标值的相对偏差约为百分之二。这样的结果是否有共性？例 1.6.2 中，将已有的每一个物品价值的个位数用 0 替代，得到修改了物品价值的背包实例，从计算的过程中发现，算法 A 求解修改物品价值后的背包实例等价于求解物品价值为 $\bar{c}_1=2, \bar{c}_2=1, \bar{c}_3=3, \bar{c}_4=2$，这样，Step1. 3 中 S_j 的存储量减少。

对(1. 6. 2)给定的一个实例，如果将每一 j 物品的价值 c_j 用一个不超过它且后 t 位为 0 的最大整数 c_j' 替代，则得到另一个 0-1 背包问题的实例：

$$\begin{aligned} z_1^* &= \max \sum_{i=1}^{n} c_i' x_i \\ s.t.\ &\sum_{i=1}^{n} a_i x_i \le b \\ & x_i \in \{0,1\},\ j=1,\ldots,n. \end{aligned} \tag{1.6.3}$$

假设 S 和 S' 分别为(1.6.2)和(1.6.3)的最优解对应的物品集合，则有

$$z^*=\sum_{j\in S} c_j \ge \sum_{j\in S'} c_j \ge \sum_{j\in S'} c_j' = z_1^* \ge \sum_{j\in S} c_j' \ge \sum_{j\in S} (c_j - 10^t) \ge \sum_{j\in S} c_j - 10^t n。$$

最优值的差满足：$z^*-z_1^* \le 10^t n$。相对差满足：

$$\frac{z^* - z_1^*}{z^*} \le \frac{10^t n}{\max\{c_j \mid j = 1,2,\cdots,n\}}。 \tag{1.6.4}$$

对(1.6.2)的 0-1 背包问题，若给定任意小的一个相对差 $\varepsilon > 0$，是否存在一个算法，使得该算法计算得到的目标值 z_2^* 满足：$\frac{z^* - z_2^*}{z^*} \le \varepsilon$，并且将 ε 看成常数时，算法的复杂性为原实例规模的多项式关系？

01KS-PTAS 算法：对给定的 $\varepsilon > 0$，

（1）对(1.6.2)满足 $\frac{n}{\max\{c_j \mid j = 1,2,\cdots,n\}} \le \varepsilon$ 的实例，则取

$$t = \left\lfloor \log_{10} \frac{\max\{c_j \mid j = 1,2,\cdots,n\}\varepsilon}{n} \right\rfloor, \tag{1.6.5}$$

其中 $\lfloor x \rfloor$ 表示不超过 x 的最大整数。将(1.6.2)每一 j 物品的价值 c_j 用一个不超过它且后 t 位为 0 的最大整数 c_j' 替代，得到(1.6.3)。用算法 A 计算(1.6.3)得到的最优解视为(1.6.2)的最终解。

（2）对(1.6.2)满足 $\frac{n}{\max\{c_j \mid j = 1,2,\cdots,n\}} > \varepsilon$ 的实例，用算法 A 计算(1.6.2)得到最优解。

定理 1.6.4 01KS-PTAS 算法的最终解目标值 z_2^* 满足：$\frac{z^* - z_2^*}{z^*} \le \varepsilon$ 且算法的计算复杂性为 $O(n^4 \frac{1}{\varepsilon})$。

证明：对 01KS-PTAS 算法的第（1）种情况，由(1.6.5)关于 t 的取法和(1.6.4)的误差估计，得到 $\frac{z^* - z_2^*}{z^*} \le \varepsilon$。

因为 A 算法的计算复杂性为 $O(n^2 z^*)$，等价为 $O(n^3 \max\{c_j \mid j = 1,2,\cdots,n\})$。算法 A 计算(1.6.3)的算法复杂性为 $O(n^3 \max\{c_j \mid j = 1,2,\cdots,n\} * 10^{-t})$。由

$$\log_{10} \frac{\max\{c_j \mid j = 1,2,\cdots,n\}\varepsilon}{10n} = \log_{10} \frac{\max\{c_j \mid j = 1,2,\cdots,n\}\varepsilon}{n} - 1$$
$$\le t \le \log_{10} \frac{\max\{c_j \mid j = 1,2,\cdots,n\}\varepsilon}{n},$$

得到 $O(n^3 \max\{c_j \mid j = 1,2,\cdots,n\} * 10^{-t}) = O(n^4 \frac{1}{\varepsilon})$。

对 01KS-PTAS 算法的第（2）种情况，应用算法 A 求解(1.6.2)，最优目标值同最终解目标值的偏差为 0。A 的计算复杂性为 $O(n^2 z^*) = O(n^3 \max\{c_j \mid j = 1,2,\cdots,n\})$。由于 $\frac{n}{\max\{c_j \mid j = 1,2,\cdots,n\}} > \varepsilon$，则 $\frac{1}{\varepsilon} > \frac{\max\{c_j \mid j = 1,2,\cdots,n\}}{n}$，所以，算法复杂性为

$O(n^4\frac{1}{\varepsilon})$。□

定义 1.6.4 对给定的$\varepsilon>0$，若一个优化问题存在一个算法满足：它按定义 1.4.1 是ε近似算法；且将ε看成常数时，算法复杂性对任何一个实例，是实例输入规模的多项式函数。则称这个算法为该问题的多项式时间迫近格式，简称 PTAS；如果是实例输入规模和$\frac{1}{\varepsilon}$的多项式函数，则称这个算法为完全多项式时间迫近格式(fully Polynomial-time approximation scheme)，简记为 FPTAS。

01KS-PTAS 算法为背包问题的一个 PTAS。

1.7 小结

计算复杂性理论是组合最优化的基础。只有了解所研究问题的复杂性才可能有针对性地设计算法，才能提高工作效率，起到事半功倍的作用。一些实际应用人员往往忽略这一基础工作，这不仅仅是因为数学的难度，更重要的是一种模糊观念在起主导作用。这种观念是：无论解的效果如何，能找一个算法或比较已有的各种方法得到解决方案就可以了。由这一章的讨论可以了解到，这对 NP 完全或 NP 难问题是不得已的办法。应该注意的是有些问题本来是多项式时间可解，而此时再采用启发式方法势必造成解的效果较差。对多项式问题，如果采用本书后续章节的现代优化计算方法，也可能造成计算时间的浪费，所以，一个多项式时间可解的问题一般不必用那些求解 NP 完全或 NP 难问题的启发式算法。

对于启发式算法的评价，最坏情况分析适用于比较简单的组合最优化问题的启发式算法，它需要较深厚的数学基础。另一种是概率分析的方法，同样因为具有较深的数学理论基础和较高的研究难度，使得这一方法难以推广使用。通过大规模数值计算进行评价的方法简单、易行、易于理解和有较好的说服力，因此得到研究者和实际应用者的广泛使用。

本书的后续章节将介绍目前新兴的全局优化启发式算法：禁忌搜索算法、模拟退火算法、遗传算法、蚁群优化算法、人工神经网络算法，最后一章将介绍用于估计下界的拉格朗日松弛算法。它们主要用于求解 NP 完全或 NP 难的组合最优化问题。这些全局优化启发式算法有一个共同的特点是：在一些限定条件下，理论上可以收敛到全局最优，但付出的计算时间可能无法接受；在限定计算时间后，算法成为启发式算法，但无法保证收敛到全局最优；随着计算时间的增加，算法的解变得越来越好。这些算法在一对矛盾的策略中寻求平衡：一个策略是集中(intensification)，它的主要目标是集中搜索，求得局部最优解；另一个策略是扩散(diversification)，它的主要目标是扩大搜索区域，以达到全局最优。

练习题

1、举例说明什么是组合最优化问题？并以此说明问题与实例的区别。

2、设计求解下列每一个问题的算法，并给出计算复杂性估计。

a) 给定平面上 n（>2）条直线$\{a_i x+b_i y=c_i | i=1,2,\cdots,n\}$，其中$\{a_i,b_i,c_i | i=1,2,\cdots,n\}$为整数。这些线有公共点吗？

b) 给定一个整数 p，p 是素数吗？

c) 给定 n 个整数，选出最大数。

d) 0-1 背包问题:

$$\max \sum_{i=1}^{n} c_i x_i$$

$$s.t. \sum_{i=1}^{n} a_i x_i \le b$$

$$x_i \in \{0,1\},\ j = 1,...,n.$$

3、给出集合 $P = \{x \in R_+^n | \sum_{j=1}^{n} a_j x_j \le b\}$ 中所有系数的规模及任何一个极点的表示形式的规模同系数规模的关系估计，其中 $a_j (j = 1,2,\cdots,n), b$ 为正整数。

4、证明：

对一切给定的整数 k 和 $\varepsilon > 0$，有 $(\log n)^k = O(n^\varepsilon),\ n^k = O((\log n)^{\log n})$。

5、f 和 g 的定义如下

$$f(n) = \begin{cases} n^2, \text{若} n \text{为素数}, \\ n^3, \text{其余}, \end{cases} \qquad g(n) = \begin{cases} n^2, \text{若} n \text{为奇数}, \\ n^3, \text{其余}. \end{cases}$$

下述哪个结论对？

$f(n) = O(g(n))$,

$g(n) = O(f(n))$,

$f(n), g(n) = O(n^4)$,

$f(n), g(n) = O(n^2)$.

6、无向图中的圈是从一个起始节点出发经过其他若干节点以后再回到这个起点的路（路上的节点不重复出现。证明判定问题“一个无向图中是否有圈？”属于 P，给出一个多项式时间的算法并估计算法的复杂性。

7、证明：表达式

$$(x_1 + \overline{x_2}) \bullet (\overline{x_1} + x_2) \bullet (x_1 + x_2) \bullet (\overline{x_1} + \overline{x_2})$$

不适定。

8、给出一个由 0-1 背包问题到一般整数背包问题:

$$\max \sum_{i=1}^{n} c_i x_i$$

$$s.t. \sum_{i=1}^{n} a_i x_i \le b$$

$$x_i \ge 0\text{的整数},\ j = 1,...,n.$$

的多项式转换。

9、证明集合覆盖问题属于 NPC。集合覆盖问题为：给定一个 $m \times n$ 的 0-1 矩阵 A、整系数 $(c_1, c_2, \cdots, c_n)$ 和一个整数 K，是否存在取值为 0-1 的变量 $x = (x_1, x_2, \cdots, x_n)^T$ 使得 $Ax \ge 1$（这里 1 表示分量全为 1 的向量）且 $\sum_{j=1}^{n} c_j x_j \le K$？

10、哈密顿(Hamilton)问题为：给定一个无向图 G=(N,E)，其中 N={1,2,⋯,*n*}为所有的节点组成的集合，E⊆{(*i*,*j*)|*i*,*j*∈N}为边集合，是否存在一个圈通过所有节点正好一次？假设哈密顿问题是 NP 完全，证明：TSP 属于 NPH。

11、证明：

$$\begin{aligned}&\max \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\\ &s.t.\ \sum_{i=1}^{n} x_{ij}=1,\forall j\\ &\sum_{j=1}^{n} x_{ij}=1,\forall i\\ &\sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}x_{ij}\le T\\ &x_{ij}\in\{0,1\}\forall i,j\end{aligned}$$

是 NP 难问题。

12、我们一再强调后续章节介绍的禁忌搜索、模拟退火、遗传算法、蚁群优化算法和人工神经网络是全局优化算法。全局最优的含义是什么？计算复杂性怎样？求解组合最优化问题的最优解所付出的代价是什么？

13、是否 NP 难或 NP 完全问题的每一个实例都不存在多项式时间的最优算法？

14、证明：算法 A 结束时得到的 $z^{*}=\max\{z\mid(S,z)\in S_n\}$ 为(1.6.2)的最优目标值。

15、对 0-1 背包问题：

$$\begin{aligned}&\max \sum_{i=1}^{n} c_i x_i\\ &s.t.\ \sum_{i=1}^{n} a_i x_i\le b\\ &x_i\in\{0,1\},\ j=1,...,n.\end{aligned}$$

假设所有参数 $a_j,c_j(j=1,2,\cdots,n)$ 和 b 为整数，设计动态规划算法为：

$$z_0(d)=0,d=0,1,\cdots,b,\quad z_k(d)=-\sum_{j=1}^{n} c_j-1,k=1,2,\cdots,n;d<0,$$

从 $k=1,2,\cdots,n$ 分别就 $d=0,1,\cdots,b$ 计算

$$z_k(d)=\max\{z_{k-1}(d),c_k+z_{k-1}(d-a_k)\}。$$

证明：$z_n(b)$ 为该问题的最优值，且算法的复杂性为：$O(nb)$。

参考文献

1. Osman H I. Metaheuristics: a bibliography. Annals of Operations Research, 1996, 63:513~623

2. Xing W, K Lam. Capacitated single machine scheduling and its on-line heuristics. IIE Transactions, 34, 991-998, 2002

3. Klee V, Minty G J. How good is the simplex algorithm? In: Shisha O ed. Inequalities III. New York: Academic Press Inc, 1972:150~175

4. Khachian L G. A polynomial algorithm for linear programming. Doklady Akad. Nauk USSR, 1979, 244(5):1093~1096

5. 加里 M R，约翰逊 D S(张立昂等译). 计算机和难解性：NP-C性理论导论. 科学出版社，1987

6. Lin S, Kernighan B W. An effective heuristic algorithm for the traveling salesman problem. Operations Research, 1973, 21:498~516

7. Reeves C R(Ed). Modern heuristic techniques for combinatorial problems. Oxford: Blackwell Scientific Publications, 1993

8. Polya G. How to Solve It? Princeton: Princeton University Press,1948

9. Lawler E L. Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, 1976

10. Papadimitriou C H, Steiglitz K. Combinatorial Optimization: Algorithms and Complexity. New Jersey: Prentice-Hall INC, 1982

11. Borgwardt K H. Some distribution-independent results about the asymptotic order of the average number of pivot steps of the simplex method. Math Oper Res, 1982, 7:441~462

12. Solomon M M. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res, 1987, 35:254~265

13.索尔·加斯（王建华等译）. 线性规划：方法及应用. 高等教育出版社，1990

14. Cook S A. The complexity of theorem proving procedures. In: Proc 3rd ACM Symp on the Theory of Computing ACM. 1971:151~158

第二章 禁忌搜索

禁忌搜索(tabu search)是局部邻域搜索算法的推广，是人工智能在组合最优化算法中的一个成功应用。Glover[1][2]在 1986 年提出这个概念，进而形成一套完整算法，参考[2,3,4]。禁忌搜索算法的特点是采用了禁忌技术。所谓禁忌就是禁止重复前面的工作。为了回避局部邻域搜索陷入局部最优的主要不足，禁忌搜索算法用一个禁忌表记录下已经到达过的局部最优点或达到局部最优的一些过程，在下一次搜索中，利用禁忌表中的信息不再或有选择地搜索这些点或过程，以此来跳出局部最优点。因此，有很多技术的细节问题有待下面讨论。

禁忌搜索算法中充分体现了集中和扩散两个策略。它的集中策略体现在局部搜索，即从一点出发，在这点的邻域内寻求更好的解，以达到局部最优解而结束。为了跳出局部最优解，扩散策略通过禁忌表的功能来实现。禁忌表中记录下已经到达点的某些信息，算法通过对禁忌表中点的禁忌，而达到一些没有搜索的点，从而实现更大区域的搜索。

本章 2.1 节介绍局部搜索算法，2.1 节介绍禁忌搜索算法的思想和模型，然后在 2.3 节讨论算法实现的技术问题，最后用示例理解它的应用。

2.1 局部搜索

在这一章中，除特别强调外，我们都假设算法解决如下组合最优化问题：

$$\min f(x)$$
$$s.t.\ g(x) \geq 0,$$
$$x \in D.$$

其中 $f(x)$为目标函数，$g(x)$为约束方程，D 为定义域，是一个离散的点集合。

因为禁忌搜索算法中用到局部搜索算法，我们首先介绍局部搜索算法。该算法可以简单的表示为：

局部搜索算法：

STEP1 选定一个初始可行解：x^0；记录当前最优解：$x^{best} := x^0$，T= N(x^{best});

STEP2 当T\{x^{best}}=∅时，或满足其他停止运算准则时，输出计算结果，停止运算；否则，从 T 中选一集合 S，得到 S 中的最好解 x^{now}；若 $f(x^{now}) < f(x^{best})$，则 $x^{best} := x^{now}$，T= N(x^{best})；否则，T:=T-S；重复 STEP2。

在局部搜索算法中，STEP1 的初始可行解可用随机的方法选择，也可用一些经验的方法或是其他算法计算得到。STEP2 中的集合 S 选取可以大到是 N(x^{best})本身，也可以小到只有一个元素，如用随机的方法在 N(x^{best})中选一点。从直观可以看出，S 选取的小使得每一步的计算量减少，但可比较的范围也小；S 选取大时每一步计算时间增加，比较的范围自然增加。这两种情况的应用效果依赖于实际问题。在 STEP2 中，其他停止准则是除 STEP2 中 T\{x^{best}}=∅以外准则。这些准则的给出往往取决于人们对算法的计算时间、计算结果的要求。通过下面的例子来理解局部搜索算法。

例 2.1.1 五个城市的对称 TSP 数据如图 2.1.1

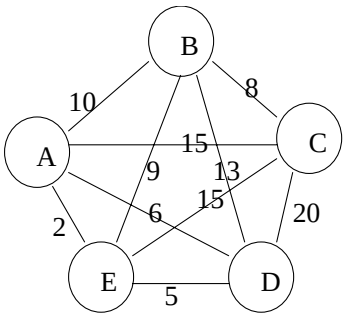


图2.1.1 五城市TSP

对应的距离矩阵为

$$D=(d_{ij})=\begin{bmatrix}0&10&15&6&2\\10&0&8&13&9\\15&8&0&20&15\\6&13&20&0&5\\2&9&15&5&0\end{bmatrix}$$

初始解为x^{best}={ABCDE}，$f(x^{best})$=45。本例中，定义邻域映射为对换两个城市位置的2-opt。选定A城市为起点，我们用两种情况解释局部搜索算法。

情况1：全邻域搜索，即S:=$N(x^{best})$。

第一循环：$N(x^{best})$={(ABCDE), (ACBDE), (ADCBE), (AECDB), (ABDCE), (ABEDC), (ABCED)}，对应目标函数值为：$f(x)$={45, 43, 45, 60, 60, 59, 44}。

x^{best}：=x^{now}=(ACBDE)。

第二循环：$N(x^{best})$={(ACBDE), (ABCDE), (ADBCE), (AEBDC), (ACDBE), (ACEDB), (ACBED)}，对应目标函数值为：$f(x)$={43, 45, 44, 59, 59, 58, 43}

x^{best}：=x^{now}=(ACBDE)。

此时，$N(x^{best})$－S为空集，于是所得解为(ADCBE)，目标值为43。

情况2：一步随机搜索。

x^{best}=（ABCDE), $f(x^{best})$=45

第一循环：由于采用$N(x^{best})$中的一步随机搜索，可以不再计算N(x^{best})中每一点的值。若从中随机选一点，如x^{now}=(ACBDE)。因$f(x^{now})$=43<45，所以x^{best}：=(ACBDE)。

第二循环：若从$N(x^{best})$中又随机选一点x^{now}=(ADBCE)，$f(x^{now})$=44>43。$N(x^{best})$=$N(x^{best})$-{x^{now}}。最后得到的解为(ACBDE)。□

局部搜索算法的优点是简单易行，容易理解，但其缺点是无法保证全局最优性。

例2.1.2 四城市非对称TSP如图2.1.2

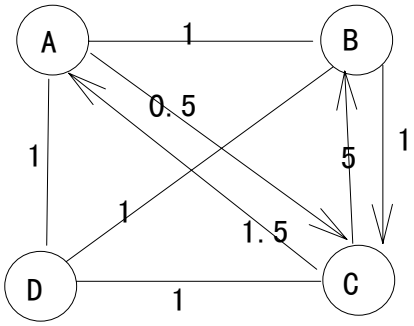


图2.1.2 四城市TSP

距离矩阵为

$$D=(d_{ij})=\begin{bmatrix}0 & 1 & 0.5 & 1\\ 1 & 0 & 1 & 1\\ 1.5 & 5 & 0 & 1\\ 1 & 1 & 1 & 0\end{bmatrix}。$$

若初始解为x^{best}=(ABCD)，并且假设城市A为起始点，$f(x^{best})$=4。

邻域$N(x^{best})$={(ABCD),(ACBD),(ADCB),(ABDC)}中，局部最优解是(ABCD)。读者可以按例2.1.1局部搜索讨论的两种情况进行验证，该算法终止时的解是局部最优解(ABCD)。而全局最优解是x^{best}=(ACDB)，$f(x^{best})$=3.5。□

局部搜索算法的计算结果主要依赖起点的选取和邻域的结构。同一个起点，不同的邻域结构会得到不同的计算结果。同样，同一个邻域结构，不同的初始点会得到不同的计算结果。因此，在使用局部搜索算法时，为了得到好的解，可以比较不同的邻域结构和不同的初始点。一个非常直观的结论是：如果初始点的选择足够多，总可以计算出全局最优解。

2.2 禁忌搜索

禁忌搜索是一种人工智能的算法，是局部搜索算法的扩展。它的一个重要思想是标记已得到的局部最优解或求解的过程，并在进一步的迭代中避开这些局部最优解或过程。如何避开和记忆这些点是本章主要讨论的问题。首先，用一个示例来理解禁忌搜索算法。

例 2.2.1(例 2.1.2 续) 假设：初始解$x^0=(ABCD)$，邻域映射为两个城市顺序对换的 2-opt，始终点都为 A 城市。目标值为$f(x^0)=4$。城市间的距离为：

$$D=(d_{ij})=\begin{bmatrix}0 & 1 & 0.5 & 1\\ 1 & 0 & 1 & 1\\ 1.5 & 5 & 0 & 1\\ 1 & 1 & 1 & 0\end{bmatrix}。$$

依据局部搜索的思想，在给定一个解后，下一步从其邻域中获得一个新的解。解的变化是由城市顺序的调换造成的，因此，关注这个变化过程。

第一步：

解的形式

A	B	C	D

$f(x^0)=4$

禁忌对象及长度

	B	C	D
A			
B			
C			

候选集

对换	评价值
C,D	4.5★
B,C	7.5
B,D	8

此处评价值为目标值，即从 A 出发并返回 A 的一个 TSP 回路的长度。由于假设了 A 城市为起终点，故候选集中有两两城市对换对 3 个。分别对换城市顺序并按目标值由小到大排列，三个评价值都劣于原值 4。在原有的局部搜索算法中，此时已达到局部最优解而停止。但现在，我们允许从候选集中选一个最好的对换——CD 城市的位置交换，用★标记入选的对换。此时，解从(ABCD)变化为(ABDC)。虽说目标值上升，但此法可能跳出局部最优。

第二步：

解的形式　　禁忌对象及长度　　候选集

A	B	D	C

$f(x^1)=4.5$

	B	C	D
A			
B			
C			3

对换	评价值
B,C	3.5★
B,D	4.5
C,D	4.5T

由于第一步中选择了 CD 交换，于是，我们希望这样的交换在下面的若干次迭代中不再出现，以避免计算中的循环，CD 成为禁忌对象并限定在以下 3 次迭代计算中不允许 CD 或 DC 对换。在对应位置记录 3。在 $N(x^1)$ 中又出现被禁忌的 CD 对换，故用 T 标记不选此交换。在选择最佳的候选对换后，到第三步。

第三步：

解的形式　　禁忌对象及长度　　候选集

A	C	D	B

$f(x^2)=3.5$

	B	C	D
A			
B		3	
C			2

对换	评价值
B,C	4.5T
B,D	7.5★
C,D	8T

新选的 BC 对换被禁后，CD 在被禁一次后还有二次禁忌。虽说候选集中的评价值都变坏，但为达到全局最优，还是从中选取。由于 BC 和 CD 对换被禁，只有 BD 对换入选。

第四步：

解的形式　　禁忌对象及长度　　候选集

A	C	B	D

$f(x^3)=7.5$

	B	C	D
A			
B		2	3
C			1

对换	评价值
B,D	3.5T
B,C	4.5T
C,D	4.5T

此时，所有候选对换被禁，怎么办？□

通过这一个示例，我们会产生如下问题：

第一， 选择什么为禁忌的对象？

例 2.2.1 禁忌的是城市顺序对换，是否会造成求全局最优解的困难？如 BC 对换造成 ABCD 到 ACBD 的变化，于是，我们希望不再考虑 B-C 和 C-B 的顺序对换，其原因之一是刚进行了 B-C 顺序对换，再对换可能不会有更好的解；其二，C-B 的顺序对换可能又回到了原有的解。禁忌 BC 双向顺序对换包括如下城市间顺序的对换： ABCD 到 ACBD、ACBD 到 ABCD、ADCB 到 ADBC、ABDC 到 ACDB、ADBC 到 ADCB、ACDB 到 ABDC 等的变化，也就是有潜在的 6 个解被禁忌到达。若 ACBD 是刚才由 ABCD 变化而来的，结合解的变化，禁忌 ACBD 到 ABCD 和 ABCD 到 ACDB 的变化是可以接受的。但对其他变化的禁忌是否会影响求解的效率？如 ADBC 到 ADCB 是否允许？

更直观一些，选择局部搜索算法的解作为禁忌对象又怎样实现？这些问题说明禁忌对象是禁忌算法中一个基本的因素。

第二，禁忌的长度如何选取？

例 2.2.2

若例 2.2.1 中禁忌长度从 3 更改为 2，在前三步与例 2.2.1 相同的情况下，有下列情形：

第四步：

解的形式

A	C	B	D

$f(x^3)=7.5$

禁忌对象及长度

	B	C	D
A			
B	1		2
C		0	

候选集

对换	评价值
B,D	3.5T
B,C	4.5T
C,D	4.5★

第五步：

解的形式

A	D	B	C

$f(x^4)=4.5$

禁忌对象及长度

	B	C	D
A			
B	0		1
C		2	

候选集

对换	评价值
B,D	4.5T
C,D	7.5T
B,C	8★

再迭代一步，又回到状态（ABCD），此时出现循环。□

禁忌长度短会造成循环，也就可能在一个局部最优解附近循环。禁忌长度长会造成算法的记忆存储量增加，使得算法计算时间增加，同时可能造成算法无法继续计算下去。因此，必须权衡这对矛盾，确定禁忌长度。

被禁的顺序对换能否再一次解禁？在例 2.2.2 的第四步中，候选集中的交换都被禁忌，若此时停止，得到的解不是一个局部最优解。候选集中 BD 顺序对换的评价值最小，是否不考虑对 BD 的禁忌而选择这个对换？有这样的直观现象，当搜索到一个局部最优解后，它邻域中的其他状态都被禁，我们是否解禁一些状态以便跳出局部最优？解禁的功能就是为了获得更大的搜索范围，以免陷入局部最优。

第三，候选集合如何选取？上面的例子是将城市所有的可对换城市对作为候选集，再从候选集中没有被禁的对换对中选最佳。对 n 个城市的 TSP，这样构造候选集使得每个集中有 C_{n-1}^2 个交换对。为了节省每一步的计算时间，有可能只在邻域中随机选一些对换，而不一定是比较邻域中的所有对换。

第四，是否有评价值的其他替代形式？上面例中用目标值作为评价值。有时计算目标值的工作量较大，或计算时间不允许那样长，于是需要其他的方法。

第五，如何利用更多的信息？禁忌搜索算法是局部搜索算法的变形。加入禁忌后的一个主要目的是得到全局最优解。根据例 2.2.1 和例 2.2.2 的计算可以看出，有时可能会产生循环，因此，在禁忌搜索算法中，通过记录其他一些信息，如当前最好解，一个被禁对象（交换）被禁的次数，评价值的大小等，来提高算法的效率。例 2.2.2 中记忆同一个对换出现的此数，假如我们得到如下的一个有关禁忌对象的信息，

	A	B	C	D
A	X			
B		X	2	3
C		5	X	1
D		2	3	X

其中，矩阵的左下角部分出现的数据表示对换被选为最佳的次数。B 与 C 对应 5 表示 BC 交换 5 次成为最佳候选。这些数字提供了状态出现的频率。 B-C 或 C-B 出现的高频率反映出这对顺序交换对目标值影响较大，在现有的禁忌条件下对 BC 的禁忌长度应该增加。

最后，终止原则怎样给出？禁忌搜索是一个启发是算法，在可接受的计算时间内，应

该使所求解尽量接近最优解。

综合上面的讨论，禁忌搜索算法的特征由禁忌对象和长度、候选集和评价函数、停止规则和一些计算信息组成。禁忌表特别指禁忌对象及其被禁的长度。禁忌对象多选择造成解变化的状态，如上面例子中的两个城市的对换。候选集中的元素依评价函数而确定，根据评价函数的优劣选择一个可能替代被禁对象的元素，是否替代取决于禁忌的规则和其他一些特殊规则，在本章后续部分将介绍一个特殊规则——特赦原则。计算中的一些信息，如被禁对象对应的评价值、被禁的频率等，将对禁忌的长度和停止规则提供帮助。

禁忌搜索算法

STEP 1 给以禁忌表(tabu list) $H=\varnothing$ 并选定一个初始解 x^{now};

STEP 2 满足停止规则时，停止计算，输出结果；否则，在 x^{now} 的邻域 $N(x^{now})$ 中选出满足不受禁忌的候选集 $Can\_N(x^{now})$；在 $Can\_N(x^{now})$ 中选一个评价值最佳的解 x^{next}， $x^{now}:=x^{next}$;更新历史记录 H，重复 Step 2。

禁忌搜索算法的 STEP2 中， x^{now} 的邻域 $N(x^{now})$ 中不受禁忌的元素包含两类：一类是那些没有被禁忌的元素，另一类是可以被解除禁忌的元素。详细的技术问题将在第 2.3 节讨论。

禁忌搜索算法是人工智能同局部搜索算法的结合，沿袭了局部搜索的的邻域构造，增加了一个禁忌表。于是，邻域连通条件为禁忌搜索可以达到全局最优解的一个必要条件。

定义 2.1 集合 C 称为相对邻域映射 N： $x\in C\to 2^C$ 是连通的，若对 C 中的任意两点 x,y，存在互异的 $x=x_1,x_2,\cdots,x_l=y$，使得 $N(x_i)\cap\{x_{i+1}\}\neq\varnothing, i=1,2,\cdots,l-1$。

定理 2.1（最优性必要条件） 在禁忌搜索算法中，若解区域相对邻域映射 N 是连通的，则可以构造禁忌算法，使得算法求得全局最优解。

证明：算法的构造如下：从任何初始解 x 开始，禁忌表中记录所有一步（除 x 外）可达的所有点，即邻域中的所有点，再以这些点为起点，一步步地记录下去。禁忌的对象是不许出现循环。很明显，这样的算法一定可以到达任何一点 y。□

这实际上就是遍历所有的状态，这不是我们希望的。实际使用时，我们期望禁忌表不能过大，因此，实用的禁忌搜索算法是一类启发式的算法。

作为禁忌搜索算法设计的第一步，邻域的构造非常重要，一旦邻域映射无法保证连通性，那么，无论禁忌表构造的多么精巧，都可能出现从某些点开始计算无法达到全局最优解。

2.3 技术问题

禁忌搜索算法是一种人工智能算法，它的一个显著特点是计算过程中用于记忆的一张禁忌表。因此，实现的技术问题是算法的关键。本节按禁忌对象、候选集合的构成、评价函数的构造、特赦规则、记忆频率信息和终止规则等分别给予介绍和讨论。

2.3.1 变化因素

禁忌表中的两个主要指标是禁忌对象和禁忌长度。顾名思义，禁忌对象指的是禁忌表中被禁的那些变化元素。因此首先需要了解状态是怎样变化的。我们将状态变化分为解的简单变化、解向量分量的变化和目标值变化三种情况。在这三种变化的基础上，讨论禁忌对象。本小节同时介绍禁忌长度和候选集确定的经验方法。

解的简单变化

这种变化最为简单。假设 $x, y \in D$，邻域映射为 N，其中 D 为优化问题的定义域，则简单解变化为

$$x \to y \in N(x)$$

是从一个解变化到另一个解。这种变化在局部搜索算法中经常采用，如例 2.1.1 第一循环中从(ABCDE)变化到(ACBDE)。这种变化将问题的解看成变化最基本因素。

向量分量的变化

这种变化考虑的更为精细，以解向量的每一个分量变化为最基本因素。设原有的解向量为 $(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n)$，用数学表达式来描述向量分量的最基本变化为：

$$(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n) \to (x_1, \cdots, x_{i-1}, y_i, x_{i+1}, \cdots, x_n),$$

即只有第 i 个分量发生变化。第 i 个分量的变化可能造成最多为 2^{n-1} 个向量的变化。向量的分量变化包含多个分量发生变化的情形。

部分优化问题的解可以用一个向量形式 $x = (x_1, x_2, \cdots, x_n)^T \in \{0,1\}^n$ 表示。解的变化可以表示某些分量的变化，如用分量从 x_j=0 变化为 x_j=1 或从 x_k=1 变化为 x_k=0，或是两者的结合。可以通过下面两个例子理解。

例 2.3.1 例 1.1.1 的 0-1 背包问题的状态变化是向量分量变化形式。如果每次只允许一个变量变化，即 $N: x \in D \to N(x) = \{y | \sum_{i=1}^{n} |y_i - x_i| \le 1\} \in 2^D$，则可能由 x_j=0 变化为 y_j=1 或由 x_j=1 变化为 y_j=0。此时，状态变化的情况是一个变量 x_j=0 变化为 x_j=1 或一个变量 x_k=1 变化为 x_k=0。□

例 2.3.2 TSP 问题采用例 1.3.2 的 2-opt 位置交换规则。当四城市 TSP 一个解为(ABCD)时，两个城市 BD 的交换可以理解为：一个 $\{0,1\}^{n\times(n-1)}$ 的向量表示的解，其中 $(x_{AB}=1, x_{BC}=1, x_{CD}=1, x_{DA}=1)$，其他分量为零；城市 BD 的交换为 $(x_{AD}=1, x_{DC}=1, x_{CB}=1, x_{BA}=1)$，其他分量为零；分量的变化是：由 x_{AB}=1 变化为 x_{AB}=0，x_{BC}=1 变化为 x_{BC}=0，x_{CD}=1 变化为 x_{CD}=0，x_{DA}=1 变化为 x_{DA}=0，x_{AD}=0 变化为 x_{AD}=1，x_{DC}=0 变化为 x_{DC}=1，x_{CB}=0 变化为 x_{CB}=1，x_{BA}=0 变化为 x_{BA}=1。于是，一个 2-opt 交换共需 8 个分量发生变化。这种情况归类于多个分量变化的结合。□

目标值变化

在优化问题的求解过程中，我们非常关心目标值是否发生变化，是否接近最优目标值。这就产生一种观察状态变化的方式：观察目标值或评价值的变化。就犹如等位线的道理一样，把处在同一等位线的解视为相同。这种变化是考察

$$H(a) = \{x \in D | f(x) = a\},$$

其中，$f(x)$ 为目标函数。它表面是两个目标值的变化，即从 $a \to b$，但隐含着两个解集合的各种变化 $\forall x \in H(a) \to \forall y \in H(b)$ 的可能。

例 2.3.3 考虑目标函数 $f(x) = x^2$，$N(x) = \{y \mid f(y) - f(x) = 3\}$。当 x=1，目标值从 1 变化到 4 隐含着二个解的变化可能：$1 \to -2$，$1 \to 2$。□

以上三种状态变化的情形，第一种的变化比较单一，而第二和第三种变化则隐含着多个解的变化可能。因此在选择禁忌对象时，可以根据实际问题采用适当的变化组合。

2.3.2 禁忌表

禁忌表是禁忌算法的一个重要特征，它主要由禁忌对象和禁忌长度组成。禁忌的对象

是那些引起解变化的因素，禁忌长度为禁忌对象的受禁时间，在算法中以迭代长度表示。

禁忌对象的选取

根据上面状态变化三种形式的讨论，禁忌的对象可以是上面的任何一种。现用示例来分别理解。

第一种情况考虑解为简单变化。当解从 $x \to y$ 时，y 可能是局部最优解，为了避开局部最优解，算法应尽可能跳出局部最优解 y。算法的规则是：当 y 的邻域中有比它更优的点时，则选择更优的解；当 y 为 $N(y)$ 的局部最优时，不再选 y，而选择比 y 较差的解，但禁忌已经出现的点。

例 2.3.4 (例 2.1.1 续) 五城市 TSP 的距离矩阵为

$$D=(d_{ij})=\begin{bmatrix} 0 & 10 & 15 & 6 & 2 \\ 10 & 0 & 8 & 13 & 9 \\ 15 & 8 & 0 & 20 & 15 \\ 6 & 13 & 20 & 0 & 5 \\ 2 & 9 & 15 & 5 & 0 \end{bmatrix},$$

禁忌对象为简单的解变化。禁忌表最多只记忆四个被禁的解，即禁忌长度为 4。从 2-opt 邻域 $N(x^{now})$ 中选出最佳的五个解组成候选集 $Can\_N(x^{now})$；初始解 $x^{now}=x^0=(ABCDE)$，$f(x^0)=45$, $H=\{(ABCDE)\}$

第一步

$x^{now}=(ABCDE)$，$f(x^{now})=45$，$H=\{(ABCDE;45)\}$，

$Can\_N(x^{now})$ = {(ACBDE;43), (ABCDE;45), (ADCBE;45), (ABEDC;59), (ABCED;44),}，

x^{next}=(ACBDE).

从候选集中选最好的一个。

第二步

$x^{now}=(ACBDE)$，$f(x^{now})=43$，$H=\{(ABCDE;45),(ACBDE;43)\}$，

$Can\_N(x^{now})$={(ACBDE;43), (ACBED;43), (ADBCE;44), (ABCDE;45), (ACEDB,58)}，

从候选集中选最好的一个。由于(ACBDE)受禁，所以选 x^{next}=(ACBED)。

第三步

$x^{now}=(ACBED)$，$f(x^{now})=43$，$H=\{(ABCDE;45),(ACBDE;43),(ACBED;43)\}$，

$Can\_N(x^{now})$={(ACBED;43), (ACBDE;43), (ABCED;44), (AEBCD;45), (ADBEC;58)}。

由于 $(ACBDE;43),(ACBED;43)$ 受禁，所以选 x^{next}=(ABCED)。

第四步

$x^{now}=(ABCED)$，$f(x^{now})=44$，

$H=\{(ABCDE;45),(ACBDE;43),(ACBED;43),(ABCED;44)\}$，

$Can\_N(x^{now})$={(ACBED; 43), (AECBD;44), (ABCDE;45),(ABCED;44),(ABDEC;58)}.

x^{next}=(AECBD)。此时 H 已达 4 个解，新选入的解替代最早被禁的解。

第五步

x^{now}=(AECBD)，$f(x^{now})=44$，

$H=\{(ACBDE;43),(ACBED;43),(ABCED;44),(AECBD;44)\}$，

$Can\_N(x^{now})$={(AEDBC; 43), (ABCED;44), (AECBD,44), (AECDB;44),(AEBCD;45)}.

x^{next}=(AEDBC)。□

第二种情况考虑禁忌 TSP 城市间顺序的对换变化，观察下例。

例 2.3.5(例 2.3.4 续) H 只记忆三对对换时。若(X,Y)在禁忌表中，表示禁忌 X 与 Y 或 Y 与 X 的交换。从 2-opt 邻域 $N(x^{now})-\{x^{now}\}$ 中选出最佳的五个状态对应的交换对组成候选集 $Can\_N(x^{now})$，因为受禁的是交换对，因此不考虑没有变化的 x^{now}。初始解 $x^{now}=x^0=(ABCDE)$，$f(x^0)=45$,

第一步

$x^{now}=(ABCDE)$，$f(x^{now})=45$，$H=\varnothing$，$Can\_N(x^{now})$ ={(ACBDE;43), (ADCBE;45), (AECDB;60), (ABEDC;59), (ABCED;44)}，x^{next} =(ACBDE).

由于 H 为空集，从候选集中选最好的一个，它是 B 与 C 的对换构成。

第二步

$x^{now}=(ACBDE)$，$f(x^{now})=43$，$H=\{(B,C)\}$，

$Can\_N(x^{now})$ ={(ACBED;43), (ADBCE;44), (ABCDE;45), (ACEDB,58), (AEBDC;59)}，

选 x^{next} =(ACBED)。它是 D 和 E 对换。

第三步

$x^{now}=(ACBED)$，$f(x^{now})=43$，$H=\{(B,C),(D,E)\}$，

$Can\_N(x^{now})$ ={(ACBDE;43), (ABCED;44), (AEBCD,45), (ADBEC;58), (ACEBD;58)}。

由于受禁，所以选 x^{next} =(AEBCD)。

前两步与例 2.3.4 的前两步计算相同，但第三步与例 2.3.4 的第三步不同。因为根据禁忌表中的条件，禁忌选取由(ACBED)经 BC 对换后的解(ABCED)。由此看出禁忌对换的范围要大于例 2.3.4 只对简单解禁忌的范围。□

经过例 2.3.4 和 2.3.5 的简单计算，可以发现禁忌还应该考虑方向的变化。变化被禁忌时，可以考虑 x 与 y 交换和 y 与 x 交换都被禁忌。如禁忌 B 和 C 对换时，可以考虑禁忌 C 与 B 对换和 B 与 C 对换，这是双向禁忌。反方向禁忌避免上一步已经变化的两个元素再变化回去。如例 2.3.5，若某一步得到解为(ABCDE)，迭代一步得到 $x^{now}=(ACBDE)$ 时，此时，应该考虑禁忌 C 与 B 的对换，否则，可能回到(ABCDE)。正向禁忌的出发点是已经选择正向的变化，为了考虑更大范围的变化，不应该马上再考虑这种变化。如例 2.3.5 进行第四步迭代，发现 B 与 C 对换得到(AECBD)，目标值为 44，如果不禁这个交换方向，则这个交换又被选入。为了搜索有较大的遍历性，我们不再考虑 B 与 C 的对换，即两个方向的对换都禁忌。

在有些情况下，可以考虑一个方向对换的禁忌。无法从理论上说明这两种选择哪个好，只能通过模拟计算比较两种选择的好坏。

第三种情况为目标值的变化。继续观察例 2.3.5。

例 2.3.6 (例 2.3.5 续) H 只记忆三组元素（解及其目标值），从 2-opt 邻域 $N(x^{now})$ 中选出最佳的五个元素为候选集 $Can\_N(x^{now})$；在 $Can\_N(x^{now})$ 中选一个目标值最佳的解 x^{next}。初始解 $x^{now}=x^0=(ABCDE)$，$f(x^0)=45$,

第一步

$x^{now}=(ABCDE)$，$f(x^{now})=45$，$H=\{45\}$，

$Can\_N(x^{now})$ ={(ABCDE;45),(ACBDE;43), (ADCBE;45),(ABEDC;59),(ABCED;44)}，

x^{next} =(ACBDE).

从候选集中选最好的一个。

第二步

$x^{now}=(ACBDE)$，$f(x^{now})=43$，$H=\{45,43\}$，

$Can\_N(x^{now})$ ={(ACBDE;43), (ACBED;43), (ADBCE;44), (ABCDE;45), (ACEDB,58)}，

x^{next}=(ADBCE)。

由于函数值 43 受禁，选候选集中不受禁的最佳函数值 44 的状态。

采用禁忌目标值的方法所禁忌的范围比例 2.3.4 和例 2.3.5 更大，在这一例中明显地体现。□

状态变化的主要因素归结为三种形式：简单的解的变化，解向量的分量变化和目标值的变化。针对三种不同的状态变化方式，例 2.3.4、例 2.3.5 和例 2.3.6 体现了禁忌搜索算法在计算中的不同。从上面示例的计算中也可以看出，解的简单变化比解的分量变化和目标值变化的受禁忌范围要小，这可能造成计算时间的增加，但它也给予了较大的搜索范围。解分量的变化和目标值变化的禁忌范围要大，这减少了计算的时间，可能引发的问题是禁忌的范围太大以至陷在局部最优点。

禁忌搜索算法中的技术很强，在算法的设计过程中，一方面要求尽量少的占用机器内存，这就要求禁忌长度、候选集合尽量小。正好相反，禁忌长度过短造成搜索的循环，候选集合过小造成过早地陷入局部最优。

禁忌长度的确定

禁忌长度是被禁对象不允许选取的迭代次数。一般是给被禁对象 x 一个数（禁忌长度）t，要求对象 x 在 t 步迭代内被禁，在禁忌表中采用 tabu(x)=t 记忆，每迭代一步，该项指标做运算 tabu(x)=t-1，直到 tabu(x)=0 时解禁。于是，我们可将所有元素分成两类，被禁元素和自由元素。有关禁忌长度 t 的选取，可以归纳为下面几种情况：

第一，t 为常数，如 $t=10$, $t=\sqrt{n}$, 其中 n 为邻居的个数。这种规则容易在算法中实现。

第二，$t\in[t_{\min},t_{\max}]$。此时 t 是可以变化的数，$t_{\min},t_{\max}$ 是确定的。确定 $t_{\min},t_{\max}$ 的常用方法是根据问题的规模 T，限定变化区间 $[\alpha\sqrt{T},\beta\sqrt{T}](0<\alpha<\beta)$。也可以用邻域中邻居的个数 n 确定变化区间 $[\alpha\sqrt{n},\beta\sqrt{n}](0<\alpha<\beta)$。当给定了变化区间，确定 t 的大小主要依据实际问题、实验和设计者的经验。如从直观可见，当函数值下降较大时，可能谷较深，欲跳出局部最优，希望被禁的长度较大。也可以利用禁忌对象出现的频率信息，当频率越高，其禁忌的长度越长。

第三，$t_{\min},t_{\max}$ 的动态选取。有的情况下，用 $t_{\min},t_{\max}$ 的变化能达到更好的解。它的基本思想同第二种情况类似。

禁忌长度的选取同实际问题、实验和设计者的经验有紧密的联系，同时它决定了计算的复杂性。过短会造成循环的出现。如 $f(1)=1$，$f(2)=3.5$，$f(3)=2.5$，$f(4)=2$，$f(5)=3$，$f(6)=6$，极端情况禁忌长度是 1，邻域为相距不超过 1 的整数点，一旦陷入局部最优点 $x=4$，则出现循环而无法跳出局部最优。过长又造成计算时间较大。上面第一和第二情形中给出的公式或数值都是一些经验的估计。

特赦规则

在禁忌搜索算法的迭代过程中，会出现候选集中的全部对象都被禁忌，或一对象被禁但若解禁后其当前最优值将下降的情况。在这样的情况下，为了达到全局的最优，我们会让一些禁忌对象重新可选。这种方法称为特赦，相应的规则称为特赦规则(aspiration criteria)。先用下面的例子理解特赦规则的应用。

例 2.3.7 五城市的 TSP，其城市间的距离为：

$$D=(d_{ij})=\begin{bmatrix}0 & 2 & 10 & 10 & 1\\ 1 & 0 & 2 & 10 & 10\\ 10 & 1 & 0 & 2 & 10\\ 10 & 10 & 1 & 0 & 2\\ 2 & 10 & 10 & 1 & 0\end{bmatrix},$$

假设禁忌表中受禁的对象是 2-opt 的城市位置顺序对换，已经过若干步计算，得到目前的一个解为(AEDBC)，f(AEDBC)=24，且此时被禁的对换包括 BC。若交换 BC，目标值 5。这个目标值好于前面的任何一个最佳候选解，有理由解禁 BC 对换。这就是一种特赦规则。□

以下罗列三种常用的特赦规则，在下面的讨论中，评价值越小越好。

第一，基于评价值的规则。例 2.3.7 就是这样的规则。在整个计算过程中，记忆已出现的最好解 x^{best}。当候选集中出现一个解 x^{now}，其评价值（可能是目标值）满足 $c(x^{best})>c(x^{now})$ 时，虽说从 x^{best} 达到 x^{now} 的变化是被禁忌的，此时，解禁限制到达 x^{now} 的变化使其自由。直观理解，我们得到一个更好的解。

第二，基于最小错误的规则。当候选集中所有的对象都被禁忌时，而第一规则又无法使程序继续下去。为了得到更好的解，从候选集的所有元素中选一个评价值最小的状态解禁。

第三，基于影响力的规则。有些对象的变化对目标值的影响很大，而有的变化对目标值的变化较小。我们应该关注影响大的变化。从这个角度理解，如果一个影响大的变化成为被禁对象，我们应该使其自由，这样才能得到问题一个更好的解。需要注意的是，我们不能理解为：禁忌的对象对目标影响大就一定使得目标（或是评价值）变小，它只是一个影响力指标。下面用 0-1 背包问题的一禁忌搜索算法理解影响力指标。

例 2.3.8 0-1 背包问题的禁忌搜索算法

解的形式为：$x\in\{0.1\}^n$，邻域定义为：

$$N:x\to N(x)=\{y\mid\sum_{i=1}^{n}|y_i-x_i|\le 1\}。$$

这样的邻域定义要求每次最多只能有一个变量变化，从 0 变为 1 或是从 1 变为 0，直观理解是装一个物品进去或是取一个物品出来。评价值定义为目标值，此时目标值越大越好。在计算的每一个迭代过程中，需要验证包容量的可行性，当所装物品尺寸大于包的容积时为不可行解，其目标值定义为－K（K 是一个充分大的整数，表示得到的解为不可行解）。除了目标值以外，一个有影响力的指标是物品尺寸的大小。取出一个尺寸大的物品可以装进多个小的物品，装进一个大的物品很快使包的容积变小。假设刚装进一个大的物品，此时包正好装满，马上取出是应该禁忌的。如五个物品的 0-1 背包问题，它们的尺寸和价值 $\{a_i,c_i\}(i=1,2,3,4,5)$ 分别为 $\{4,4\},\{2,4),\{1,1.5\},\{3,4\},\{1.5,4\}$，包的容积为 6。假设在计算的一步有 $x_1=x_5=1$ 和 $x_2=x_3=x_4=0$，$x_1:1\to 0$ 和 $x_5:1\to 0$ 受禁。由于包中不可能再装入任何物品，为了不使计算停止，只能采取特赦的方法。因为尺寸是能力约束的最重要的指标，同时对目标函数最具有影响力，因此选尺寸大的特赦即 $x_1:1\to 0$ 解禁。这样其他物品就有被装包的可能性。□

当然，这一规则应结合禁忌长度和评价函数使用。如在候选集中目标值都不及当前的最好解，而一个禁忌对象的影响指标很高且很快将被解禁时，我们可以通过解禁这个状态以期望得到更好的解。

2.3.3 其他

候选集合的确定

候选集合由邻域中的邻居组成。常规的方法是从邻域中选择若干个目标值或评价值最佳的邻居入选。如在 TSP 中采用 2-opt 邻域定义，一个状态的邻域中共有 C_n^2 个邻居，计算目标值后，例 2.3.4、例 2.3.5 和例 2.3.6 选择五个目标值最佳的邻居入选。

有时认为上面的计算量还是太大，则不在邻域的所有邻居中选择，而是在邻域中的一部分邻居中选择若干个目标值或评价值最佳的状态入选。也可以用随机选取的方法实现部分邻居的选取。

评价函数

评价函数赋予候选集合中每元素一个实数值，通过这些评价函数值来选取下一步计算的替代点。以目标函数作为评价函数是比较容易理解的。目标值是一个非常直观的指标，但有时为了方便计算或易于理解，会用一些更简单、直观的计算值来代替目标值。我们将评价函数分为直接和间接目标函数两类。

第一，直接评价函数，主要包含以目标函数的运算而得到评价值的方法。记评价函数为 $p(x)$，目标函数为 $f(x)$，则评价函数可以采用目标函数

$$p(x)=f(x);$$

目标函数值与 x^{now} 目标值的差值

$$p(x)=f(x)-f(x^{now}),$$

其中 x^{now} 是上一次迭代计算的解；目标函数值与当前最优解 x^{best} 目标值的差值

$$p(x)=f(x)-f(x^{best}),$$

其中 x^{best} 是目前计算中的最好解。

它们主要通过对目标函数进行简单的运算，变形很多。

第二，间接评价函数。构造其他评价函数的基本原则是尽量反映目标函数的特性，因此这一类评价函数实质上还是基于目标函数，。

有时目标值的计算比较复杂或耗时较多，解决这一问题的方法之一是采用替代的评价函数。替代的评价函数还应该反映原目标函数的一些特性，如原目标函数对应的最优点还应该是替代函数的最优点。构造替代函数的目标是减少计算的复杂性。它的一个例子是在生产计划中的约束批量计划与调度(Capacitated Lotsize Planning and Scheduling)模型中的应用。

例 2.3.9 生产计划批量问题。一个工厂安排 n 个产品在 T 个计划时段里加工，其各个产品的需求量、加工费用和加工能力等之间的关系用下面数学模型(CLPS)描述，

$$\min\sum_{i=1}^{n}\sum_{t=1}^{T}(p_i x_{it}+s_i y_{it}+h_i I_{it}) \tag{2.3.1}$$

$$s.t \quad I_{it-1}+x_{it}-I_{it}=d_{it},\ i=1,2,\cdots,n;t=1,2,\cdots,T, \tag{2.3.2}$$

$$.\sum_{i=1}^{n}a_i x_{it}\le c_t,t=1,2,\cdots,T, \tag{2.3.3}$$

$$y_{it}=\begin{cases}1, & \text{若 } x_{it}>0,\\ 0, & \text{其它.}\end{cases} \tag{2.3.4}$$

$$x_{it},I_{it}\ge 0,i=1,2,\cdots,n;t=1,2,\cdots,T.$$

其中， s_i 表示生产 i 产品的生产准备费用；

h_i 表示单位 i 产品的库存费用；
p_i 表示生产单位 i 产品的费用；
x_{it} 表示第 t 时段，i 产品的生产批量；
I_{it} 表示第 t 时段，i 产品的库存量；
d_{it} 表示第 t 时段，i 产品的外部需求量；
a_i 表示生产单位 i 产品占用的资源量；
c_t 表示第 t 时段能力的提供量。

上面模型中，x_{it}, I_{it} 为决策变量，所有参数为非负值。目标(2.3.1)要求生产、生产准备和库存三项费用和最小，(2.3.2)为外部需求、生产和库存之间的平衡关系，(2.3.3)为资源约束，要求每个时段生产占用的能力不超过可提供的能力。CLPS 模型是一个混合整数规划问题。因为 y_{it} 取 0,1 两个值，当 $Y^0=(y_{it})$ 的值选定后，CLPS 模型为一个线性规划问题，对应的最优目标值为 $Z(Y^0)$。这样，禁忌搜索算法可以只将 (y_{it}) 看成决策变量进行计算求解。如果采用基于目标函数的评价函数方法，对应每一个 $Y^0=(y_{it})$ 要解一个线性规划问题。虽说线性规划问题有多项式时间的最优算法，但对每一个给定的 Y^0 求解一次 $Z(Y^0)$ 还是很费时的。一个简单的替代方法是对给定的 Y^0，用启发式的算法求解 CLPS。基本思想是根据 Y^0、d_{it} 和 c_t 安排各时段的生产（可参见[4]），最后得到一个启发式算法的目标值 $Z^H(Y^0)$，以此为评价函数值。

求解的基本思想是：当没有资源约束(2.3.3)时，CLPS 模型存在一个最优解满足

$$I_{it-1}x_{it}=0,$$

即上一个时段有库存时，本时段不生产；反之本时段要生产则上一个时段必然库存为零；由此对应唯一一个解 $Y\in\{0,1\}^{n\times T}$ 满足

$$x_{i\tau_1}=\begin{cases}\sum_{t=\tau_1}^{\tau_2}d_{it}, y_{i\tau_1}=y_{i\tau_2+1}=1, y_{it}=0, \tau_1<t\le\tau_2,\\ 0, \qquad \text{其它.}\end{cases} \tag{2.3.5}$$

增加(2.3.3)资源约束后，从最后一个时段 T 到第一个时段反向按能力约束逐个时段验证解(2.3.5)是否满足(2.3.3)。当不满足时，将超出资源约束的部分前移一个时段加工。如此修正，若第一个时段资源不足时，则认为无可行解，此时目标值记为充分大的一个数。若可行，则按修正后的解 $x_{it}, I_{it}, Y_{it}\ (i=1,2,\cdots,n; t=1,2,\cdots,T)$，用(2.3.1)计算目标函数值。这个目标值作为评价值。□

例 2.3.9 用一个启发式算法解的目标值替代最优值。取代的主要原因是认为线性规划算法的计算时间还是太长。如果计算目标值比较复杂或所花费的时间较长，可以用替代的方法。应该注意的是替代函数应尽可能地反映原目标函数的特性。

记忆频率信息

在计算的过程中，记忆一些信息对解决问题有利。如一个最好的目标值出现的频率很高，这使我们有理由推测：现有参数的算法可能无法再得到更好的解。根据解决问题的需要，我们可以记忆解集合、被禁有序对象组、目标值集合等的出现频率。频率信息有助于进一步加强禁忌搜索的效率。我们可以根据频率信息动态控制禁忌的长度。当一个元素或一个序列重复出现，我们可以增加禁忌长度以避开循环。当一个序列的目标（评价）值变化较小时，有必要增加该序列每一个对象的禁忌长度，反之，减少每一个对象的禁忌长度。一个最佳的目标值出现的频率很高，有理由终止计算而将这一值认定为是最优值。一般可以根据状态的变化将频率信息分为两类：静态和动态。

静态的频率信息主要是某些变化，诸如解、对换或目标值在计算中出现的频率。求解

它们的频率相对比较简单。如可以记录它们在计算中出现的次数，出现的次数与总的迭代数的比率，从一个状态出发再回到该状态的迭代次数等。这些信息有助于我们了解解、对换或目标值的重要性，是否出现循环和循环的次数。在禁忌搜索中，为了更充分的利用信息，一定要记忆目前最优解。

动态的频率信息主要是从一个解、对换或目标值到另一个解、对换或目标值的变化趋势，如记忆一个解序列的变化，或记一个解序列变化的若干个点等。由于记录比较复杂，因此，它提供的信息量也较大。在计算动态频率时，通常需要记录的信息有：一个序列的长度，即序列中元素个数；第二，从序列中的一个元素出发，再回到该序列该元素的迭代次数；第三，一个序列的平均目标（评价）值，从序列中一个元素到另一个元素目标（评价）值的变化情况；第四，该序列出现的频率。

终止规则

禁忌搜索算法是一个启发式算法。我们不可能让禁忌长度充分大，只希望在可接受的时间里给出一个满意的解。于是很多直观、易于操作的原则包含在终止规则中。通常使用的终止规则有以下几种。

第一，确定步数终止。给定一个充分大的数 K，总的迭代次数不超过 K 步。既使算法中包含其他的终止原则，算法的总迭代次数有保证。这种原则的优点是易于操作和可控计算时间，但无法保证解的效果。在采用这个规则时，应记录当前最优解。

第二，频率控制原则。当某一个解、目标值或元素序列的频率超过一个给定的标准时，如果算法不做改进，只会造成频率的增加，此时的继续计算无法不会得到更好的解，因此，终止计算。

第三，目标控制原则。在禁忌搜索算法中，提倡记忆当前最优解。如果在一个给定的步数内，当前最优值没变，同第二规则相同的观点，停止运算。

第四，目标值偏离程度原则。对一些问题可以简单地计算出它们的下界（目标为极小），记一个问题的下界为 Z_{LB}，目标值为 $f(x)$。对给定的误差ε，当 $f(x)-Z_{LB}\le\varepsilon$ 时，终止计算。这表示目前计算得到的解与以满足误差要求。

2.4 应用实例——图节点着色和车间作业排序

本节介绍两个应用实例，一个是比较简单的图节点着色(node coloring)问题，另一个是车间作业调度，也称为车间作业排序问题。从两个实例中可以了解禁忌搜索算法应用的一些理论和技术问题。

2.4.1 图节点着色问题

给定一个无向图 $G=(V, E)$，其中 V 是节点集 $V=\{1,2,...,n\}$，E 是边集 $E=\{(i,j)|i,j\in V\}$，(i,j)表示有连接 i,j 的一个边。若 $V_i\subset V$、$V=\bigcup_{i=1}^{k}V_i$ 且 V_i 内部的任何两个节点没有 E 中的边直接相连，则称 $(V_1,V_2,\cdots,V_k)$ 为 V 的一个划分。图的节点着色问题可以描述为：求一个最小的 k，使得 $(V_1,V_2,\cdots,V_k)$ 为 V 的一个划分。

例 2.4.1 如图 2.4.1 所示，5 个节点的无向图图 $G=(V,E)$为：

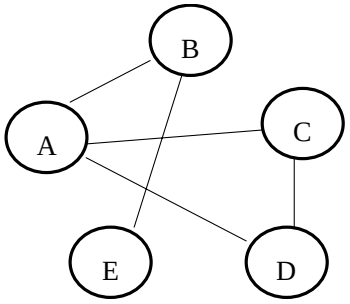


图 2.4.1 五节点图

它的一个划分是：$V_1=\{A,E\}$，$V_2=\{B,D\}$，$V_3=\{C\}$。当然 $V_1=\{A\}$，$V_2=\{B\}$，$V_3=\{C\}$，$V_4=\{D,E\}$ 也是一个划分。□

对给定 n 个节点的无向图，禁忌搜索算法求解图节点覆盖问题可以分为两步。第一步是给定一个常数 k，考虑目标函数

$$f(V_1,V_2,\cdots,V_k)=|E(V_1)|+|E(V_2)|+\cdots+|E(V_k)|,$$

其中，$|E(V_i)|$ 为 V_i 中直接连接两个节点的边数。$(V_1,V_2,\cdots,V_k)$ 为 V 的一个划分的充分必要条件是 $f(V_1,V_2,\cdots,V_k)=0$。

第二步的主要工作是：对不满足 $f(V_1,V_2,\cdots,V_k)=0$ 的 $(V_1,V_2,\cdots,V_k)$，增加子集的个数 k；从满足 $f(V_1,V_2,\cdots,V_k)=0$ 的划分中选择最小的划分数 k。

将解的形式用下面形式表示：

$$S=\begin{pmatrix}1 & 2 & \cdots & n\\ i_1 & i_2 & \cdots & i_n\end{pmatrix},\ 1\le i_j\le k,\ 1\le j\le n,$$

其中 i_j 表示第 j 个节点在第 i_j 集合中。

解的第 x 个分量变化为：

$$\begin{pmatrix}x\\ i_x\end{pmatrix}\to\begin{pmatrix}x\\ i'_x\end{pmatrix},$$

从原在第 i_x 集合中改变到第 i'_x 集合中。

每一个解 S 的邻域由那些满足上面的变化且只有一个分量变化的解组成，共有 nk 个邻居（包含自身），每一个节点（分量）可以选择 k 个划分集中的任何一个。

禁忌：

$$\begin{pmatrix}x\\ i_x\end{pmatrix}\leftarrow\begin{pmatrix}x\\ i'_x\end{pmatrix},$$

即还原到原有状态。这种禁忌考虑了变化的方向。

特赦规则：若 x^* 是当前最优解，当一个受禁的邻居 x 满足 $f(x)\le f(x^*)-1$ 时，则受禁的变化特赦。

因为目标值为非负整数，条件 $f(x)\le f(x^*)-1$ 表示 x 的目标值小于当前最优解 x^* 的目标值。本例采用了基于评价值的第一特赦规则。

Hertz 和 Werra[5]在集合的划分数给定的情况下，给出的程序的框架如下

已知：
划分集合数 k，解的形式 S，目标函数 f，解的变化，邻域映射 N(S)，禁忌长度，目标值下界 f*，两个目标值没有改进的最大允许迭代次数 nbmax。
开始
——任选一个初始解；

——nbiter:=0;（*当前的迭代步*）
——bestiter:=0;（*当前最优解所在的迭代步*）
——bestsol:=S;（*当前的最优解*）
——令 z=f(S), A(z):=z-1;（特赦值）
——初始化禁忌表 H;
当（f(S)>f*）且（nbiter-bestiter<nbmax）时，计算
——nbiter:=nbiter+1;
——产生 N(S)的一个候选集 V*，要求候选元素 x 是非禁忌的或是特赦的 f(x)≤A(f(S));
—在 V*中选目标值最优的解 S*;
—若 f(S*) ≤A(f(S))，则 A(f(S*)):=f(S*)-1；否则若 f(S)≤A(f(S*))，则 A(f(S*)):=f(S)-1;
—更新禁忌表;
—若 f(S*)<f(bestsol)，则 bestsol:=S*; bestiter:=nbiter;
—S:=S*
继续;
结束。
输出：计算中的最优解。

在[5]中，随机产生实例的参数为：无向图共有 1000 个节点，边集的密度是 50%（大约是 $C_{1000}^2/2$ 个边），禁忌表长度|H|=7，候选解个数$|V^*|=|V|/2$。最终对上面规模实例的计算结果是平均可用 87 种颜色划分。概率分析的理论结果是 85 中颜色。

2.4.2 车间作业调度问题

问题的描述

车间作业调度(job shop scheduling)问题可以简单地描述为 n 个工件(job)（这是一个统称），$J_1, J_2, \cdots, J_n$ 在 m 台机器(machine)（也是统称）$M_1, M_2, \cdots, M_m$ 上加工。每一个工件 J_i 有 n_i 个工序(operation) $O_{i1}, O_{i2}, \cdots, O_{in_i}$，第 O_{ij} 工序的加工时间为 p_{ij}。必须按工序顺序进行加工且每一工序必须一次加工完成（无抢占，no preemption)。一台机器在任何时刻最多只能加工一个产品，一个工件不能同时在两台机器上加工。如何在上面的条件下使最后一个完工的工件加工时间最短？

车间作业问题可以用析取图(disjunctive graph)表示：

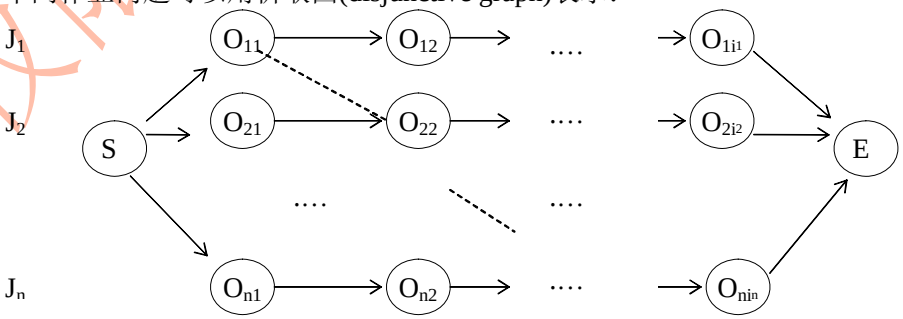


图 2.4.2 车间作业排序的析取图

在图 2.4.2 中，S 和 E 是虚拟的起终点，实线弧表示两个工序的前后关系，这是不允许改变的。每一行表示一个工件的所有工序。虚线边暂时是没有方向的，一条虚线段的两个

端点（工序）表示在同一台机器上加工，如 O_{11},O_{22} 的连接线表示它俩在同一台机器上加工。车间作业的一个调度是给予所有虚线边一个方向，成为一个虚线弧。一个虚线弧所指方向表示加工顺序。寻找车间作业调度问题的一个可行解就是在图 2.4.2 中给虚线赋方向，使其成为一个有向且无圈的图。

车间作业调度问题在两个工件时是存在多项式时间的最优算法，参考[6]，但当工件数超过 2 时，问题的复杂性是 NP 难，证明见[7]。非常直观，问题的一个解对应 m 台机器上的一个加工排序。

在应用禁忌搜索之前，由于车间作业的邻域构造有较强的代表性，我们首先考虑问题的邻域构造。第一种方法是选一台机器上的两个工序交换加工顺序，如一台机器上原有的加工序为(a,b,c,d,e)，交换 ac 序后，新的加工序为(c,b,a,d,e)。这样的邻域映射使得每个解有 $\sum_{i=1}^{n} C_{n_i}^2$ 个邻居。进一步推广上面的方法，可以研究若干个位置交换的情形。一种特殊的情况是将一个工序移到另一个位置加工。如在一台机器上原有的加工序是(abcdef)，现将 f 移到第一个位置加工，加工序则为(fabcde)。这些是我们常见的方法。

第二种方法是关键路(critical path)法。这种方法的一个基本思想是：在第一种方法中，一些交换对目标值没有影响，于是在计算中，这些交换浪费计算时间，应抓住最长的、加工中没有时间空闲的一条路（即关键路），交换同在这条路上且同在一台机器上加工的两个加工工件的位置。

关键路的理论

例 2.4.1 三个工件在两台机器上加工的车间作业问题，其加工如图 2.4.3

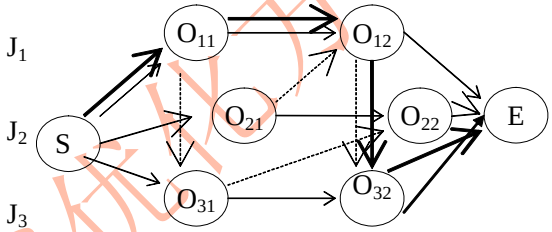


图 2.4.3 三工件车间作业图

从图中可以看到，O_{11},O_{22},O_{31} 在一台机器上加工，O_{12},O_{21},O_{32} 在另一台机器上加工。假设各工序的加工时间为：$p_{11}=5,p_{22}=2,p_{31}=4$，$p_{12}=7,p_{21}=1,p_{32}=2$。两台机器的加工序分别为：

M_1：$O_{11}\to O_{31}\to O_{22}$

M_2：$O_{21}\to O_{12}\to O_{32}$

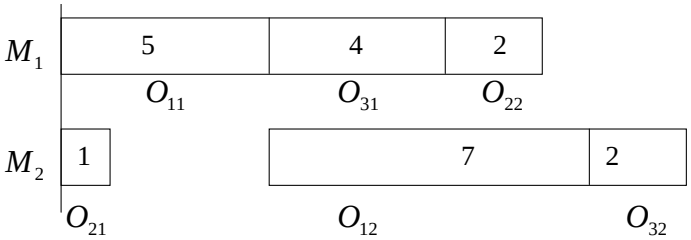


图 2.4.4 关键路：$O_{11}\to O_{12}\to O_{32}$

关键路是：$O_{11}\to O_{12}\to O_{32}$，长度为 14。以较粗的线标记在图 2.4.3 上。从图 2.4.4 可以看出，交换 $O_{31}O_{22}$ 的加工位置对最长完工时间没有影响，但交换关键路上的一个工件，如 $O_{11}O_{31}$ 交换，则如图 2.4.5

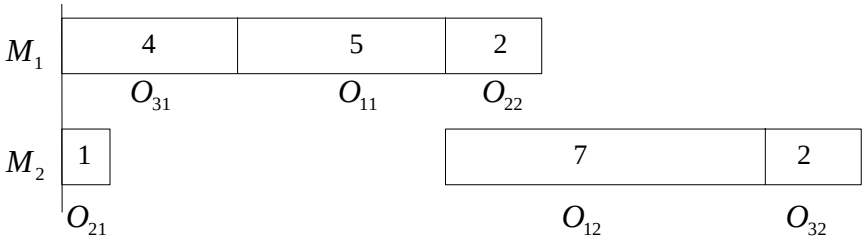


图 2.4.5 关键路： $O_{31} \to O_{11} \to O_{12} \to O_{32}$

关键路是： $O_{31} \to O_{11} \to O_{12} \to O_{32}$，长度为 18。由此可以看到，关键路上的加工工件的位置改变对目标值有直接影响。

定义 2.4.1 在关键路上，满足下列条件的相邻节点集称为块(block)：

(1) 由关键路上的相邻节点组成，至少包含两个工序；

(2) 集合中的所有工序在一台机器上加工；

(3) 增加一个工序后，不满足(1)或(2)。

定理 2.4.1 若一解的关键路不包含块，则一定是最优解。

证明：由定理的条件，关键路上不包含块。由图 2.4.3，同一台机器的相邻工序由虚线相连，由此可知，任意相邻的两个工序由一条实线连接。这条关键路长度是某一个工件的所有工序加工时间和，是车间作业完工时间的一个下界。因此定理结论成立。□

若 y 是车间作业的一个可行解，则对应 y，图 2.4.3 为一个有向无圈图。若 y 和 y’是作业调度的两个可行解，且 y 和 y’对应的有向图 S 和 S’。当图 S’的关键路 P’的长度小于图 S 的关键路 P 的长度时，称可行解 y’改进了 y。

定理 2.4.2[8] 若 y 和 y’是车间作业调度的两个可行解，且 y 和 y’对应的有向图为 S 和 S’。若 y’改进 y，则一定满足下面两条件之一：

(1) 至少有一个工序，它在 y 的一个块 B 中且不是 B 块中的第一个工序，但在 y’中它在 B 的其他工序之前加工；

(2) 至少有一个工序，它在 y 的一个块 B 中且不是 B 块中的最后一个工序，但在 y’中它在 B 的其他工序之后加工。

证明：y 对应的一条关键路为 P：

$$P: S, u_1^1, \cdots, u_{b_1}^1, u_1^2, \cdots u_{b_2}^2, u_1^k \cdots, u_{b_k}^k, E,$$

其中，$u_1^j, \cdots, u_{b_j}^j$ 表示一个块，即在同一台机器上加工。现用反证法证明结论。假设 y’和 y 的块且所在的机器相同，且 y’属于 P 的块中的任何一个工序不在块的其他工序之前或后加工，则有：

情形 1：在 y’中，$u_{b_j}^j$ 在 u_1^{j+1} 之前加工，且它们是同一个工件的不同工序；

情形 2：若 $b_j \geq 2$，无论是在 y 还是在 y’中，$u_1^j, \cdots, u_{b_j}^j$ 在同一台机器上加工；

情形 3：若 $b_j \geq 2$，无论是在 y 还是在 y’中，u_1^j 总是在 $u_2^j, \cdots, u_{b_j}^j$ 之前加工，$u_{b_j}^j$ 总是在 $u_1^j, \cdots, u_{b_j-1}^j$ 之后加工；

在 y’中，$u_1^j, \cdots, u_{b_j}^j$ 的加工序为 $v_1^j, \cdots, v_{b_j}^j$，但由上面的讨论，$u_1^j = v_1^j$ 和 $u_{b_j}^j = v_{b_j}^j$。于是 y’对应的一条路（不一定为关键路）为 P’：

$$P': S, v_1^1, \cdots, v_{b_1}^1, v_1^2, \cdots v_{b_2}^2, v_1^k \cdots, v_{b_k}^k, E,$$

y’的最大完工时间 L(y’)为：

$$L(y') \geq \sum_{j=1}^{k}\sum_{l=1}^{b_j} p_{v_l^j} = \sum_{j=1}^{k}\sum_{l=1}^{b_j} p_{u_l^j} = L(y),$$

y'不是 y 的一个改进，矛盾。故定理结论成立。□

邻域结构

通过定理 2.4.2，可以给出基于关键路的邻域构造方法。

邻域 N1：设 y'为一个可行解，若 y'将 y 的关键路中一个块中的一个作业前移到最前或后移到最后位置加工，则称 y'为 y 的一个邻居。所有这样的移动组成的集合为 y 的邻域。

邻域 N2：设 y'为一个可行解，若 y'是 y 的关键路中一个块中的作业的前移或后移的所有可能位置加工，则称 y'为 y 的一个邻居。所有这样的移动组成的集合为 y 的邻域。

注：在上面的邻域定义中，我们强调 y'为一个可行解，这是因为在车间作业调度中，交换两个工序的加工位置可能出现死锁(deadlock)现象，如图 2.4.6，原机器的加工序为：

M_1: $O_{11} \to O_{22}$,

M_2: $O_{12} \to O_{21}$,

改变第一台机器的加工序为 M_1: $O_{22} \to O_{11}$，如图 2.4.6，此时任何一个工件无法开工，因此称为死锁现象。

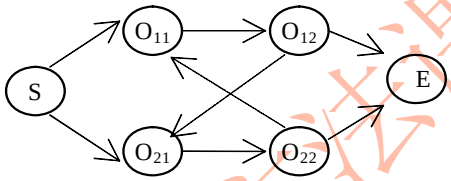


图 2.4.6 死锁现象

定理 2.4.3 车间作业调度的可行解集合相对 N2 是连通的，即以任何一个可行解为起点，可以通过 N2 达到一个全局最优解。

证明参见[9]。

计算结果

文[8]中研究了车间作业的一种更广泛的情形，多功能机器车间作业排序问题。在他们的模型中，工序 O_{ij} 可由一个机器集合 $G_{ij} \subset \{M_1, M_2, \cdots, M_m\}$ 中的任何一个机器加工。极端的情况 $|G_{ij}|=1$ 为常规的车间作业排序问题。他们产生 53 组随机数据进行数值计算。数据生成的原则是：以[7]和[10]文中的典型数据为基础，以表 2.1 的多功能系数分组随机产生数据文件，所产生的数据分别用 edata、rdata 和 vdata 标记。文[7]和[10]中典型数据的主要参数是：机器数分别为 5、6、10 和 15，工件数分别为 6、10、15、20 和 30，且每一个工件的工序数都相同，满足 $n_i = m, i = 1,2,\cdots,n$。

表 2.4.1 多功能参数

	$\|G_{ij}\|$ave	$\|G_{ij}\|$max
Edata	1.15	$2\,(m \leq 6)$ $3\,(m \geq 10)$
Rdata	2	3
Vdata	$\frac{1}{2}m$	$\frac{4}{3}m$

其中，$|G_{ij}|$ave 表示所有$|G_{ij}|$的平均数，$|G_{ij}|$max 表示所有$|G_{ij}|$的最大数。

用上面的 N1 和 N2 邻域结构，禁忌对象选择为：禁忌上一步位置变化的复原，如一个

块为(abcde)，c 原在的位置是 3，经过位置移动后，禁忌回到 3。禁忌表的长度为 30，总的迭代长度分为 1000 和 5000 次两种。停止规则为：在迭代总长达到之前，若邻域中的所有可行解都被禁忌、目前解的目标值等于下界或出现循环三种情况之一出现则停止计算。

功能类型	相对误差	N1-1000	N2-1000	N1-5000	N2-5000
edata	平均	5.2	5.3	4.8	4.5
	最大	24.0	22.8	23.4	19.8
rdata	平均	2.8	2.9	2.3	2.3
	最大	13.4	14.5	12.0	10.7
vdata	平均	0.5	0.5	0.4	0.4
	最大	3.2	3.1	1.9	2.1

其中，表中所有数据为禁忌搜索算法所得解的目标值同已知最佳下界的相对误差；N1-1000 表示采用 N1 邻域和最大迭代次数为 1000；N2-5000 表示采用 N2 邻域和最大迭代次数为 5000。

计算结果显示，邻域 N1 和 N2 对计算结果的影响不明显，此处 N1 是否具有定理 2.4 的连通性还没有得到证明；总体的平均效果，禁忌搜索算法相当好，所得目标值同已知最好下界的相对差的平均效果在 0.4%到 5.2%之间；随着机器功能的增加，相对误差逐渐变小；迭代次数 1000 和 5000 的差别不是很大，也就是在迭代总次数限制在 1000 次时，计算效果同迭代 5000 次基本相同。因此可以采用 1000 次的迭代而节省计算时间。

练习题

1. 对例 2.1.1 的五城市 TSP，初始解为(ACBDE)，城市交换对被禁，候选集合采用 2-opt 邻域，禁忌长度为 3，问：算法能否达到最优解(AEDCB)？在什么样的条件下可以达到最优解？
2. 就禁忌状态变化和目标值分别讨论例 2.1.2 的收敛情况。
3. 证明：图的节点着色问题是 NP 难。
4. 在车间作业排序问题中，对 N1 和 N2 邻域，在什么条件下，一个块的两个工序位置交换保持可行解。
5. 在问题 4 的程序实现中，如何判别交换后的解是不可行解？
6. 实现禁忌搜索算法求解图的节点着色问题。
7. 实现禁忌搜索算法求解车间作业调度问题。
8. 实现禁忌搜索算法求解 0—1 背包问题。

参考文献

1. Glover F. Future paths for integer programming and links to artificial intelligence. Computers and Operations Reseach, 1986, 13:533~549
2. Glover F. Tabu search: part I. ORSA Journal on Computing, 1989,1:190~206
3. Glover F. Tabu search: part II. ORSA Journal on Computing,1990, 2:4~32
4. Maes J, McClain J O, Van Wassenhove L N. Multilevel capacitated lotsizing complexity and LP-based heuristics. Euro J of Opnl Res, 1991, 53:131~148
5. Hertz A, de Werra D. The tabu search metaheuristic: how we use it. Annals of Mathematics and Artificial Intelligence, 1990, 1:111~121
6. Brucker P, Schilie R. Job-shop scheduling with multi-purpose machines. Computing, 1990,

45:369~375
7. Adams J, Balas E, Zawack D. The shifting bottleneck procedure for job-shop scheduling. Management Sciences, 1988, 34:391~401
8. Hurink J, Jurisch B, Thole M. Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spektrum, 1994, 15:205~215
9. 邢文训. Jobshop 排序问题的模拟退火算法. 见：中国运筹学会第二届全国排序学术会议论文集（武汉），1993. 5~9
10. Fisher H, Thompson G L. Probabilistic learning combinations of local job-shop scheduling Rules. In: Muth J F, Thompson G L, ed. Industrial Schedulinh, Englewood Cliffs: Prentice Hall,1963. 225~251

第三章 模拟退火算法

模拟退火(simulated annealing)算法是局部搜索算法的扩展，它不同于局部搜索之处是以一定的概率选择邻域中费用值大的状态。从理论上来说，它是一个全局最优算法。模拟退火算法最早的思想由 Metropolis[1]在 1953 年提出，Kirkpatrick[2]在 1983 年成功地应用在组合最优化问题。本章 3.1 节介绍模拟退火算法的思想和模型，3.2 节给出马尔可夫链的基本理论，3.3 和 3.4 节分别讨论时齐和非时齐算法的收敛性，3.5 节讨论算法实现的技术问题，最后用示例理解它的应用。

3.1 模拟退火算法及模型

退火是一种物理过程，金属物体在加热至一定的温度后，它的所有分子在状态空间 D 中自由运动。随着温度的下降，这些分子逐渐停留在不同的状态。在温度最低时，分子重新以一定的结构排列。由统计力学的研究表明，在温度 T，分子停留在状态 r 满足波兹曼(Boltzmann)概率分布

$$\Pr\{\bar{E} = E(r)\} = \frac{1}{Z(T)}\exp(-\frac{E(r)}{k_B T}), \qquad (3.1.1)$$

其中，$E(r)$表示状态 r 的能量，k_B>0 为波兹曼常数，$\bar{E}$ 表示分子能量的一个随机变量。Z(T)为概率分布的标准化因子

$$Z(T) = \sum_{s\in D}\exp\left\{-\frac{E(s)}{k_B T}\right\}。$$

先研究由(3.1.1)随 T 变化的趋势。选定两个能量 $E_1 < E_2$，在同一个温度 T，有

$$\Pr(\bar{E} = E_1) - \Pr(\bar{E} = E_2) = \frac{1}{Z(T)}\exp(-\frac{E_1}{k_B T})[1-\exp(-\frac{E_2 - E_1}{k_B T})]。$$

因为

$$\exp(-\frac{E_2 - E_1}{k_B T}) < 1, \quad \forall T > 0,$$

所以

$$\Pr(\bar{E} = E_1) - \Pr(\bar{E} = E_2) > 0, \quad \forall T > 0。 \qquad (3.1.2)$$

在同一个温度，(3.1.2)表示分子停留在能量小的状态的概率比停留在能量大的状态的概率要大。当温度相当高时，(3.1.1)的概率分布使得每个状态的概率基本相同，接近平均值 $1/|D|$，|D|为状态空间 D 中状态的个数。结合(3.1.2)，当状态空间存在至少两个不同能量时，具有最低能量状态的波兹曼概率超出平均值 $1/|D|$。由

$$\frac{\partial \Pr\{\bar{E} = E(r)\}}{\partial T} = \frac{\exp\left\{-\frac{E(r)}{k_B T}\right\}}{Z(T)k_B T^2}\left(E(r) - \frac{\sum_{s\in D}E(s)\exp\left\{-\frac{E(s)}{k_B T}\right\}}{Z(T)}\right), \qquad (3.1.3)$$

当 $r_{\min}$ 是 D 中具有最低能量的状态时，得

$$\frac{\partial \Pr\{\overline{E} = E(r_{\min})\}}{\partial T} < 0,$$

所以，$\Pr\{\overline{E} = E(r_{\min})\}$ 关于温度 T 是单调下降的。又有

$$\Pr\{\overline{E} = E(r_{\min})\} = \frac{1}{Z(T)}\exp(-\frac{E(r_{\min})}{k_B T}) = \frac{1}{|D_0|+R},$$

其中，D_0 是具有最低能量的状态集合，

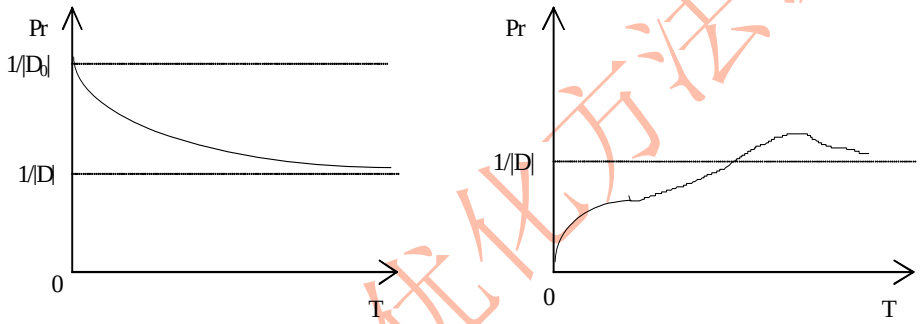
$$R = \sum_{s\in D: E(s)>E(r_{\min})} \exp(-\frac{E(s)-E(r_{\min})}{k_B T}) \to 0,\ T \to 0。 \qquad (3.1.4)$$

因此得到，当 T 趋向于 0 时，

$$\Pr\{\overline{E} = E(r_{\min})\} \to \frac{1}{|D_0|},\ T \to 0。$$

当温度趋向 0 时，(3.1.1)决定的概率渐近 $1/|D_0|$。由此可以得到，在温度趋向 0 时，分子停留在最低能量状态的概率趋向 1。综合上面的讨论，分子在能量最低状态的概率变化趋势由图 3.1.1(a)表示。

对于非能量最小的状态，由(3.1.2)和分子在能量最小状态的概率是单调减的事实，在温度较高时，分子在这些状态的概率在 $1/|D|$ 附近，依赖于状态的不同可能超过 $1/|D|$；由(3.1.3)和(3.1.4)可知存在一个温度 t，使(3.1.1)决定的概率在(0,t)是单调升的（留作练习）；再由(3.1.4)可知，当温度趋于 0 时，(3.1.1)定义的概率趋于 0。概率变化曲线见图 3.1.1(b)。



(a) 在能量最低状态　　(b) 在非能量最低状态

图 3.1.1 波兹曼函数曲线

从上面的讨论得到，温度越低｛$T \to 0$｝，能量越低的状态的概率值越高。在极限状况，只有能量最低的点概率不为零。可以从下面的例子了解(3.1.1)的变化规律。

例 3.1.1 简化概率分布(3.1.1)为

$$p(x) = \frac{1}{q(t)}\exp(-\frac{x}{t}),$$

其中，$q(t)$ 为标准化因子。设共有四个能量点 x=1,2,3,4，在此 $q(t) = \sum_{x=1}^{4}\exp(-\frac{x}{t})$。观察 t=20,5,0.5 三个温度点概率的变化。

从表 3.1.1 可以看到，当温度较高时(t=20)，四点的概率分布相差比较小，也可以看成概率是均匀分布，但能量最低状态 x=1 的概率 0.269 超出平均值 0.25。这相当于分子的随机游动。在温度下降后(t=5)，状态 x=4 的发生概率变得比较小了，也就是说，它的活跃程度下降。在 t=0.5 时，x=1 的概率达 0.865，而其他三个状态的概率都很小，合起来为 0.135。在这个

表中也可以看出，非能量最低状态 x=2 的概率在三个温度点(0.5,5,20)有一个上升和下降的过程，在 t=20 和 5 时的概率都超过平均概率 0.25。□

表 3.1.1 t=20,5,0.5 三个温度点概率分布

	x=1	x=2	x=3	x=4
t=20	0.269	0.256	0.243	0.232
t=5	0.329	0.269	0.221	0.181
t=0.5	0.865	0.117	0.016	0.002

通过这个示例，我们可以将组合优化问题同金属物体退火进行类比：

组合优化问题	金属物体
解	状态
最优解	能量最低的状态
费用函数	能量

由以上的类比及(3.1.1)，组合优化的最优解可以类比为退火过程中能量的最低状态，也就是温度达到最低点时，(3.1.1)概率分布中具有最大概率的状态。于是组合优化问题 $z=\min\{f(x)\mid g(x)\geq 0, x\in D\}$ 的求解过程类比为退火过程，其中 D 是有限离散定义域。

在这一章中，除特别强调外，我们都假设算法用以解决如下组合最优化问题：

$$\min f(x)$$
$$s.t.\ g(x)\geq 0,$$
$$x\in D.$$

其中 $f(x)$为目标函数，$g(x)$为约束方程，D 为定义域。

在不混淆的条件下，为了符号的书写简单，状态 x_i 简单地用 i 表示，简单的模拟退火算法为：

模拟退火算法：

STEP1 任选一个初始解 i_0；$i:=i_0$；k:=0；t_0:=tmax（初始温度）；

STEP2 若在该温度达到内循环停止条件，则到 STEP3；否则，从邻域 N(i)中随机选一 j，计算 $\Delta f_{ij}=f(j)-f(i)$；若 $\Delta f_{ij}\leq 0$，则 $i:=j$，否则若 $\exp(-\Delta f_{ij}/t_k)$>random(0,1)时，则 $i:=j$；重复 STEP2；

STEP3 $t_{k+1}:=d(t_k)$；k:=k+1；若满足停止条件，终止计算；否则，回到 STEP2。

在上述的模拟退火算法中，包含一个内循环和一个外循环。内循环是 STEP2，它表示在同一个温度 t_k 时，在一些状态随机搜索。外循环主要包括 STEP3 的温度下降变化 $t_{k+1}:=d(t_k)$，迭代步数的增加 k:=k+1 和停止条件。这些内容将在 3.3 节中讨论。模拟退火算法的直观理解是：在一个给定的温度，搜索从一个状态随机地变化到另一个状态。每一个状态到达的次数服从一个概率分布。当温度很低时，由(3.1.4)的讨论，以概率 1 停留在最优解。这些理论将在 3.3 节介绍。

模拟退火算法的每一次迭代都体现集中和扩散两个策略的平衡。对遇到的下一个迭代解，如果这个解更好，则采用集中策略，选择这个解为新解。不然的话，采用扩散策略，以某一概率选择这个解为新解。

模拟退火算法的数学模型可以描述为：在给定邻域结构后，模拟退火过程是从一个状态到另一个状态不断地随机游动。我们可以用马尔可夫(Markov)链描述这一过程。对给定的温度 t，两个状态的转移概率(transition probability)定义为：

$$p_{ij}(t)=\begin{cases} G_{ij}(t)A_{ij}(t), & \forall j\neq i,\\ 1-\sum_{l=1,l\neq i}^{|D|}G_{il}(t)A_{il}(t), & j=i, \end{cases} \tag{3.1.5}$$

其中，|D|表示状态集合（解集合）中状态的个数。$G_{ij}(t)$称为从 i 到 j 的产生概率(generation probability)。$G_{ij}(t)$表示在状态 i 时，j 状态被选取的概率。比较容易理解的是 j 为 i 的邻居，如果在邻域中等概率选取，则 j 被选中的概率为

$$G_{ij}(t)=\begin{cases} 1/|N(i)|, & j\in N(i),\\ 0, & j\notin N(i). \end{cases} \tag{3.1.6}$$

$A_{ij}(t)$称为接受概率(acceptance probability)。$A_{ij}(t)$表示产生状态 j 后，接受 j 的概率，如在模拟退火算法中接受的概率是

$$A_{ij}(t)=\begin{cases} 1, & f(i)\geq f(j),\\ \exp(-\Delta f_{ij}/t), & f(i)<f(j). \end{cases} \tag{3.1.7}$$

$G(t)=(G_{ij}(t))_{|D|\times|D|}$，$A(t)=(A_{ij}(t))_{|D|\times|D|}$和$P(t)=(p_{ij}(t))_{|D|\times|D|}$分别称为产生矩阵，接受矩阵和一步转移概率矩阵。

(3.1.5)，(3.1.6)，(3.1.7)为模拟退火算法的主要模型。模拟退火算法主要可以分为两类。第一类为时齐算法，基本计算框架为：在(3.1.5)中对每一个固定的 t，计算对应的马尔可夫链，直至达到一个稳定状态，然后再让温度下降。第二类为非时齐算法，基本计算框架为：由一个马尔可夫链组成，要求在两个相邻的转移中，温度 t 是下降的。

无论从实际和直观，描述模拟退火过程的马尔可夫链应满足下列条件：第一，可达性。无论起点如何，任何一个状态都可以到达。这样使我们有得到最优解的可能，否则，从理论上无法达到最优解的算法是无法采用的。第二，渐近不依赖起点。由于起点的选择有非常大的随机性，我们的目的是达到全局最优，因此，应渐近的不依赖起点。第三，分布稳定性。包含两个内容：其一是当温度不变时，其马尔可夫链的极限分布存在；其二是当温度渐近 0 时，其马尔可夫链也有极限分布。第四，收敛到最优解。当温度渐近 0 时，最优状态的极限分布和为 1。

3.2 马尔可夫链

设$\{X(k)\}_{k=0,1,2,\cdots}$为随机变量序列，$X(k)=i$称在时刻 k 处于状态 i，$i\in D$。D 称为状态空间。当 D 中的状态数有限时，称为有限状态空间。若对任意的正整数 n,

$$\begin{aligned} &P\{X(n)=j\mid X(1)=i_1,X(2)=i_2,\cdots,X(n-2)=i_{n-2},X(n-1)=i\}\\ &=P(X(n)=j\mid X(n-1)=i), \end{aligned} \tag{3.2.1}$$

$i_1,i_2,\cdots,i_{n-2},i,j\in D$。

则称$\{X(k)\}_{k=0,1,2,\cdots}$为马尔可夫链，简称马氏链。马氏链具有记忆遗忘特性，它只记忆前一时刻的状态。(3.2.1)可以简单地记成

$$p_{ij}(n-1)=P(X(n)=j\mid X(n-1)=i),$$

称为一步转移概率。当状态空间有限时，称为有限马氏链，一步转移概率矩阵记为

$$P(n-1)=\left(p_{ij}(n-1)\right)_{|D|\times|D|}。$$

当

$$p_{ij}(n-1)=p(n),\ \forall n\ , \tag{3.2.2}$$

成立时，称马氏链为时齐(homogeneous)的。用

$$p_{ij}^{(n)} = P(X(n) = j \mid X(0) = i), \tag{3.2.3}$$

表示 n 步转移概率。

例 3.2.1 当温度 t 给定时，(3.1.5)，(3.1.6)，(3.1.7)确定一个时齐马氏链。

由(3.1.5)，(3.1.6)，(3.1.7)可以看出，一步转移概率只同状态 i 转移到 j 有关，同第几次迭代无关。因此，马氏链是时齐的。正是这个原因，将这一类算法取名为时齐算法。□

在讨论算法的收敛性之前，先介绍概率论中的常用概念。

若存在 n，使得 $p_{ij}^{(n)} > 0$，则称状态 i 可达状态 j，记成 $i \to j$。若状态 i 和状态 j 满足 $i \to j$ 且 $j \to i$，则称状态 i 和状态相通，记成 $i \leftrightarrow j$。可达关系具有半序性(semi-order)。

定理 3.2.1 若 $i \to j$，$j \to k$，则 $i \to k$。

证明：由 $i \to j$，则存在 n，使得 $p_{ij}^{(n)} > 0$。由 $j \to k$，则存在 m，使得 $p_{jk}^{(m)} > 0$。于是，

$$\begin{aligned} p_{ik}^{(m+n)} &= P(X(n+m) = k \mid X(0) = i) \\ &= \sum_{l=0}^{\infty} P(X(m+n) = k, X(n) = l \mid X(0) = i) \\ &= \sum_{l=0}^{\infty} P(X(m+n) = k \mid X(n) = l, X(0) = i) P(X(n) = l \mid X(0) = i) \\ &= \sum_{l=0}^{\infty} P(X(m) = k \mid X(0) = l) P(X(n) = l \mid X(0) = i) \\ &\geq p_{ij}^{(n)} p_{jk}^{(m)} > 0, \end{aligned}$$

所以，定理成立。□

定义从 i 到达 j 首达时刻的随机变量为

$$T_{ij} = \min\{n \mid X(0) = i, X(n) = j, n \geq 1\}, \tag{3.2.4}$$

其概率定义为

$$\begin{aligned} f_{ij}^{(n)} &= P(T_{ij} = n \mid X(0) = i) \\ &= P(X(n) = j, X(m) \neq j, m = 1,2,\cdots,n-1 \mid X(0) = i)。 \end{aligned} \tag{3.2.5}$$

迟早到达概率定义为

$$f_{ij} = \sum_{n=1}^{\infty} f_{ij}^{(n)}。 \tag{3.2.6}$$

定理 3.2.2 $f_{ij} > 0$ 的充分必要条件是 $i \to j$。

证明：“充分性”：由 $i \to j$，则存在 n，使得 $p_{ij}^{(n)} > 0$。又

$$\begin{aligned} p_{ij}^{(n)} &= P\{X(n) = j \mid X(0) = i\} = P\{T_{ij} \leq n, X(n) = j \mid X(0) = i\} \\ &= \sum_{l=1}^{n} P\{T_{ij} = l, X(n) = j \mid X(0) = i\} \\ &= \sum_{l=1}^{n} P\{T_{ij} = l \mid X(0) = i\} P\{X(n) = j \mid T_{ij} = l, X(0) = i\} \\ &= \sum_{l=1}^{n} P\{T_{ij} = l \mid X(0) = i\} P\{X(n) = j \mid X(l) = j, X(l-1) \neq j, \cdots, X(1) \neq j, X(0) = i\} \\ &= \sum_{l=1}^{n} P\{T_{ij} = l \mid X(0) = i\} P\{X(n) = j \mid X(l) = j\} \end{aligned}$$

$$= \sum_{l=1}^{n} f_{ij}^{(l)} p_{jj}^{(n-l)},$$

当 $p_{ij}^{(n)} > 0$ 时，至少有一个 l 使得 $f_{ij}^{(l)} > 0$，由此，得 $f_{ij} > 0$。

“必要性”若 $f_{ij} > 0$，则存在 n，使得 $f_{ij}^{(n)} > 0$，得到

$$p_{ij}^{(n)} \geq f_{ij}^{(n)} p_{jj}^{(0)} = f_{ij}^{(n)} > 0。\square$$

当 $f_{ii} = 1$ 时，状态 i 称为常返的，当 $f_{ii} < 1$ 时称为非常返的。

定理 3.2.3 状态 j 是常返的，则以概率 1，系统无穷次返回状态 j。状态 j 是非常返的，则以概率 1，系统只有有限次返回状态 j。

证明：记 $A_{ij}(m)$ 表示自状态 i 出发，系统通过 j 状态至少 m 次的概率。记 $Y_{ij}(m)$ 为从状态 i 出发通过 j 状态至少 m 次的事件。

删除的内容: 随机变量

$$\begin{aligned} A_{ij}(m+1) &= \sum_{l=1}^{\infty} P(T_{ij} = l, Y_{jj}(m) \mid X(0) = i) \\ &= \sum_{l=1}^{\infty} P(Y_{jj}(m) \mid T_{ij} = l) P(T_{ij} = l \mid X(0) = i) \\ &= \sum_{l=1}^{\infty} A_{jj}(m) f_{ij}^{(l)} \\ &= A_{jj}(m) f_{ij}, \end{aligned}$$

于是有，$A_{jj}(m) = A_{jj}(0)(f_{jj})^m = f_{jj}^m$，进而，$A_{ij}(m+1) = f_{ij}(f_{jj})^m$，所以，

$$A_{jj}(m+1) \to \begin{cases} 1, & f_{jj} = 1, \\ 0, & f_{jj} < 1, \end{cases} \quad (m \to \infty)。\square$$

常返中的一种特殊情况为正常返，定义

$$u_i = \sum_{n=1}^{\infty} n f_{ii}^{(n)}, \tag{3.2.7}$$

当 $u_i < \infty$ 时为正常返，当 $u_i = \infty$ 时为零常返。定理 3.2.3 表明常返是以概率 1 无穷次返回同一状态，(3.2.7)表明有些常返状态（正常返）的平均返回次数是有限的，而有些是无穷的。如果 $p_{ii}^{(n)}$ 除 $n=t,2t,3t,...$ 外均为 0，且 $t(t>1$ 正整数)是满足条件的最大整数，称状态 i 具有周期(periodic)t。不存在周期的状态称为非周期状态。非周期的正常返状态称为遍历状态。

例如，$p_{ii}^{(n)} \neq 0, n = 4k(k \geq 1), p_{ii}^{(n)} = 0, n \neq 4k(k \geq 1)$，则 t=2,4 都满足 $p_{ii}^{(n)}$ 除 $n=t,2t,3t,...$ 外均为 0，此时，状态 i 的周期为 4。

在模拟退火的理论中，经常用到的一个概念是不可约(irreducible)。不可约中用到的一个概念是闭集。一个集合 C 是闭集的定义为：$\forall i \in C, j \notin C$，有 $p_{ij} = 0$。这表示 C 是一个封闭的集合 $\forall i \in C, j \notin C$，$i$ 不可达 j，即 $p_{ij}^{(n)} = 0$，对任意 n 成立。除整个状态空间外，没有别的闭集的马氏链称为不可约的马氏链。我们不加证明地给出下面重要的结论定理 3.2.4 和定理 3.2.5。相应的结论和证明可参见随机过程的教科书。

定理 3.2.4 不可约、有限状态且时齐的马氏链是正常返的。

模拟退火中用到的一个重要分布是平稳分布(stationary distribution)。一个概率分布 $\{v_j \mid j = 1,2,\cdots\}$，即 $v_j \geq 0, j = 1,2,\cdots$，$\sum_{j=1}^{\infty} v_j = 1$，如果对马氏链的一步转移概率阵 $P = (p_{ij})$ 满足

$$v_j = \sum_{i=1}^{\infty} v_i p_{ij} \ , \tag{3.2.8}$$

则称 $\{v_j \mid j = 1,2,\cdots\}$ 为马氏链的平稳分布。

定理 3.2.5 非周期、不可约且时齐的马氏链是正常返的充分必要条件：存在唯一平稳分布 $\{v_j \mid j = 1,2,\cdots\}$，满足

$$v_j = \sum_{i=1}^{\infty} v_i p_{ij}^{(n)} = \sum_{i=1}^{\infty} v_i p_{ij} \ ,$$

$$v_j = \lim_{n\to\infty} p_{ij}^{(n)} = \frac{1}{u_j} > 0 \ ,$$

其中 u_j 由(3.2.7)定义给出。

由不可约和闭集的定义，可以得到马氏链是不可约的一个充分条件，用定理叙述如下：

定理 3.2.6 有限状态的时齐马氏链是不可约的一个充分条件为：任给两个状态 i 和 j，存在 n，使得 $p_{ij}^{(n)} > 0$。

证明：由条件，状态空间的任何一个真子集不形成闭集。反证法。若形成闭集，则存在 C 和 $j \notin C$，使得 $\forall i \in C, j \notin C$，$i$ 与 j 是不可达的，即 $p_{ij}^{(n)} = 0$，对任意 n 成立。这与条件矛盾。因此，整个状态空间为一个闭集，是不可约的。□

在模拟退火中，用到的一个重要结论是定理 3.2.5。定理 3.2.6 对有限状态的马氏链给出不可约的一个充分条件。定理 3.2.5 中用到的另一个概念是非周期。由下面的定理，给出判别非周期的一个充分条件。

定理 3.2.7 若 i 和 j 相通，则它们或同为非周期的或同为周期的。

证明：由 $i \leftrightarrow j$，存在 $n \geq 1$ 和 $l \geq 1$，使得 $p_{ij}^{(n)} = a > 0$ 和 $p_{ji}^{(l)} = b > 0$。类似定理 3.2.1 的证明，可得：

$$p_{ii}^{(l+m+n)} \geq p_{ij}^{(n)} p_{jj}^{(m)} p_{ji}^{(l)} = abp_{jj}^{(m)}, \tag{3.2.9}$$

$$p_{jj}^{(l+m+n)} \geq p_{ji}^{(l)} p_{ii}^{(m)} p_{ij}^{(n)} = abp_{ii}^{(m)} 。 \tag{3.2.10}$$

再由周期的定义，若 i 的周期为 t，当 m 是 t 的倍数时，有 $p_{ii}^{(m)} > 0$。再由(3.2.9)，得到

$$p_{ii}^{(l+n)} \geq p_{ij}^{(n)} p_{jj}^{(0)} p_{ji}^{(l)} = ab > 0,$$

则 $l+n$ 是 t 的倍数，且由(3.2.10)得 $p_{jj}^{(l+m+n)} > 0$。当 m 不是 t 的倍数时，由 $l+n$ 是 t 的倍数，得 $l+m+t$ 不是 t 的倍数，由(3.2.9)得 $p_{jj}^{(m)} = 0$。总结上面的讨论，$l+n$ 是 t 的倍数，m 不是 t 的倍数时得 $p_{jj}^{(m)} = 0$，故状态 j 是周期的，其周期不小于 t。同法，当 j 是周期的，可知 i 是周期的。于是，它们或同为周期的或同不是周期的。□

模拟退火算法分为时齐(homogeneous)和非时齐(inhomogeneous)算法。由马氏链性质的讨论，当下面性质得以证明后，才能说明模拟退火算法收敛全局最优解。

时齐算法应该具有下列性质。

第一，在每一个给定的温度 t，给出(3.1.5)的一步转移概率 $p_{ij}(t)$ 的一些限定条件，使得定理 3.2.5 成立。由此得到平稳分布概率

$$v_j(t) = \lim_{n\to\infty} p_{ij}{}^{(n)}(t) = \lim_{n\to\infty} P(X(n,t) = j \mid X(0,t) = i) = \frac{1}{u_j} > 0 \ ,$$

其中，$X(n,t)$表示在温度 t 时，马氏链第 n 步运动的随机变量。

第二，给出平稳分布应该满足的条件，使得：当温度渐近达到零度时，平稳分布的极限

存在，即要求 $\pi_j = \lim_{t\to 0} v_j(t)$。

第三，进一步要求平稳分布的极限具有全局最优性条件

$$\pi_j = \begin{cases} \dfrac{1}{|D_{OPT}|}, & j \in D_{OPT}, \\ 0, & j \in D \setminus D_{OPT}, \end{cases}$$

其中 D_{OPT} 为最优状态集合。

综合上面三条，得到全局性收敛的结论

$\lim_{t\to 0}\{\lim_{n\to\infty} P(X(n,t) \in D_{OPT})\} = 1$。

对非时齐算法，也需要下面的条件保证其全局收敛性。

第一，因为非时齐算法的每步迭代温度在变化，因此需要给出(3.1.5)一步转移概率 $p_{ij}(t)$ 的一些限定条件，即需要给出 $A(t_k)$ 和 $G(t_k)$ 的限定条件，使得非时齐马氏链能收敛到一个分布。

第二，温度渐近达到零度，即 $\lim_{k\to\infty} t_k = 0$。给出保证达到全局最优的温度变化关系。

第三，总体要求 $\lim_{k\to\infty} P\{X(k,t_k) \in D_{OPT}\} = 1$。

3.3 时齐算法的收敛性

由于组合最优化问题的状态空间是有限的，要达到定理 3.2.5 的条件，根据定理 3.2.4，只需讨论模拟退火的一步转移概率(3.1.5)对应的马氏链满足不可约性和非周期性。由定理 3.2.6，有限状态的时齐马氏链是不可约的一个充分条件为：任给两个状态 x_i 和 x_j，存在 n，使得 $p_{ij}^{(n)} > 0$。于是

定理 3.3.1 时齐算法的马氏链是不可约一个充分条件为：

$$\forall i, j, t, \text{ 有 } A_{ij}(t) > 0,$$

且存在 $q \ge 1, l_0, l_1, \cdots, l_q \in D, l_0 = i, l_q = j$，使得

$$G_{l_k l_{k+1}}(t) > 0, k = 0,1,\cdots,q-1。$$

证明：当 t 是给定时，由于(3.1.5)对应的是有限的时齐马氏链，又由

$$\begin{aligned} p_{ij}^{(q)} &\ge p_{l_0 l_1} p_{l_1 l_2} \cdots p_{l_{q-1} l_q} \\ &= A_{l_0 l_1}(t) G_{l_0 l_1}(t) A_{l_1 l_2}(t) G_{l_1 l_2}(t) \cdots A_{l_{q-1} l_q}(t) G_{l_{q-1} l_q}(t) \\ &> 0, \end{aligned}$$

所以，i 可达 j。同理可证 j 可达 i。于是再由定理 3.2.6，证明不可约结论成立。□

定理 3.3.2 对给定的 $t>0$，若存在 $i \ne j \in D$, 使得 $0 < A_{ij}(t) < 1, G_{ij}(t) > 0$，则不可约的马氏链是非周期的。

证明：由

$$\begin{aligned} p_{ii}(t) &= 1 - \sum_{l=1, l\ne i}^{|D|} A_{il}(t) G_{il}(t) \\ &= 1 - \sum_{l=1, l\ne i, j}^{|D|} A_{il}(t) G_{il}(t) - A_{ij}(t) G_{ij}(t) \end{aligned}$$

$$
\begin{aligned}
&> 1 - \sum_{l=1, l\neq i, j}^{|D|} G_{il}(t) - G_{ij}(t) \\
&= 1 - \sum_{l=1, l\neq i}^{|D|} G_{il}(t) \\
&= G_{ii}(t) \geq 0,
\end{aligned}
$$

此时， $p_{ii}(t) > 0$，于是，对任意状态 i 是非周期的。再由定理 3.2.7，知结论成立。□

定理 3.3.1 和定理 3.3.2 给出存在平稳分布的一个充分条件。进一步，我们将讨论(3.1.5)中 $A(t)$和 $G(t)$应该满足的条件，使得定理 3.2.5 的平稳分布可以解析地表达。

定理 3.3.3 若(3.1.5)中 $A(t)$和 $G(t)$满足：

(1) G(t)与 t 无关；

(2) $\forall i, j \in D$，都有 $G_{ij} = G_{ji}$ 且存在 $q \geq 1, l_0, l_1, \cdots, l_q \in D, l_0 = i, l_q = j$，使得

$$G_{l_k l_{k+1}}(t) > 0, k = 0,1,\cdots,q-1;$$

(3) $\forall i, j, k \in D, f(i) \leq f(j) \leq f(k)$，都有 $A_{ik}(t) = A_{ij}(t)A_{jk}(t)$；

(4) $\forall i, j \in D, f(i) \geq f(j)$，都有 $A_{ij}(t) = 1$；

(5) $\forall i, j \in D, t > 0, f(i) < f(j)$，都有 $0 < A_{ij}(t) < 1$；

则模拟退火的时齐算法对应的马氏链有平稳分布 $v = (v_1, v_2, \cdots, v_{|D|})$，满足

$$v_i(t) = \frac{A_{i_0 i}(t)}{\sum_{j\in D} A_{i_0 j}(t)}, \forall i \in D, \tag{3.3.1}$$

其中， $i_0 \in D_{OPT}$。

证明：由(4)和(5)得到 $\forall i, j, t$ 有 $A_{ij}(t) > 0$。再由(1)、(2)和定理 3.3.1 知时齐算法对应的马氏链是不可约的。当目标函数的所有值相同时，每一个解为最优解。否则，目标函数值不为一个值，选一个最优解和一个非最优解，由(2) 知定理 3.3.1 的条件成立和(5)知定理 3.3.2 的条件成立，推导出有限时齐的马氏链是非周期的。因此，由定理 3.2.5 知平稳分布存在且唯一。现只需验证(3.3.1)的概率分布满足平稳分布的定义。由

$$
\begin{aligned}
&\sum_j v_j(t) p_{ji}(t) \\
&= \sum_{j\neq i, f(j)\leq f(i)} \frac{1}{K} A_{i_0 j}(t) G_{ji} A_{ji}(t) + \sum_{j\neq i, f(j)> f(i)} \frac{1}{K} A_{i_0 j}(t) G_{ji} A_{ji}(t) + v_i(t) p_{ii}(t) \\
&= \sum_{j\neq i, f(j)\leq f(i)} \frac{1}{K} A_{i_0 i}(t) G_{ij} + \sum_{j\neq i, f(j)> f(i)} \frac{1}{K} A_{i_0 j}(t) G_{ij} + v_i(t) p_{ii}(t) \\
&= v_i(t) \sum_{j\neq i, f(j)\leq f(i)} G_{ij} + \sum_{j\neq i, f(j)> f(i)} v_j(t) G_{ij} + v_i(t) p_{ii}(t),
\end{aligned}
$$

其中， $K = \sum_{j\in D} A_{i_0 j}(t)$，

$$
\begin{aligned}
v_i(t) p_{ii}(t) &= v_i(t)\Big(1 - \sum_{j\neq i, f(j)\leq f(i)} A_{ii}(t) G_{ij} - \sum_{j\neq i, f(j)> f(i)} A_{ij}(t) G_{ij}\Big) \\
&= v_i(t) - v_i(t) \sum_{j\neq i, f(j)\leq f(i)} G_{ij} - \sum_{j\neq i, f(j)> f(i)} \frac{1}{K} A_{i_0 i}(t) A_{ij}(t) G_{ij}\Big) \\
&= v_i(t) - v_i(t) \sum_{j\neq i, f(j)\leq f(i)} G_{ij} - \sum_{j\neq i, f(j)> f(i)} v_i(t) G_{ij}\Big),
\end{aligned}
$$

推出

$$v_i(t)=\sum_j v_j(t)p_{ji}(t),$$

即满足定理 3.2.5 的平稳分布方程。由定理 3.2.5 的唯一性结论，知定理结论成立。□

很容易验证，(3.1.7)满足定理 3.3.3 的(3)(4)(5)。(2)要求任何两个状态或互为邻居或互不为邻居，当互为邻居时，它们的产生概率相同。任何两个状态或互为邻居或互不为邻居时，下面的产生概率：

$$G_{ij}(t)=\begin{cases}\dfrac{1}{L}, & \forall j\in N(i),\\ 0, & j\notin N(i),\end{cases},$$

满足(1)和(2)的 $G_{ij}=G_{ji}$，其中 L 为一正数。(2)的其他条件需要邻域是连通的。(3.1.6)形式对应的结论将在以后的定理中叙述。

定理 3.3.4 在满足定理 3.3.3 的条件下，若

$$\forall i,j\in D, f(i)<f(j)\Rightarrow \lim_{t\to 0}A_{ij}(t)=0,$$

则有

$$\lim_{t\to 0}v_i(t)=\begin{cases}\dfrac{1}{|D_{OPT}|}, & i\in D_{OPT},\\ 0, & i\in D\setminus D_{OPT}。\end{cases}$$

证明：由(3.3.1)

$$v_i(t)=\frac{A_{i_0i}(t)}{\sum_{j\in D}A_{i_0j}(t)},\forall i\in D,$$

因为 D 为有限集，故当 $i\notin D_{OPT}$ 时，有

$$f(i)<f(i_0),i_0\in D_{OPT},$$

于是，

$$A_{i_0i}(t)\to 0,t\to 0。\square$$

定理 3.3.3 和定理 3.3.4 给出时齐算法全局收敛应满足的充分条件，但这些条件的限制是比较强的，如(3.1.6)的发生概率就不一定满足定理 3.3.3 的(2)，因此，研究模拟退火算法全局收敛应满足的充分条件是理论研究的一个方向。

定理 3.3.5 若模拟退火时齐算法中的 $A_{ij}(t)$ 和 $G_{ij}(t)$ 满足定理 3.3.1 和定理 3.3.2 的条件，且存在 $\Phi(x,t)$ 满足以下条件：

(1) $\forall i\in D,t>0$，都有 $\Phi(f(i),t)>0$;

(2) $\forall i\in D$，都有 $\sum_{i=1,i\neq j}^{|D|}\Phi(f(i),t)G_{ij}(t)A_{ij}(t)=\Phi(f(j),t)\sum_{i=1,i\neq j}^{|D|}G_{ji}(t)A_{ji}(t)$;

(3) $\lim_{t\to 0}\Phi(x,t)=\begin{cases}0, & x>0,\\ 1, & x\le 0;\end{cases}$

(4) $\dfrac{\Phi(x_1,t)}{\Phi(x_2,t)}=\Phi(x_1-x_2,t)$;

(5) $\forall t>0$，都有 $\Phi(0,t)=1$;

则马氏链的平稳分布为

$$v_i(t)=\frac{\Phi(f(i),t)}{\sum_{j\in D}\Phi(f(j),t)},\forall i\in D, \tag{3.3.2}$$

且

$$\lim_{t\to 0} v_i(t)=\begin{cases}\dfrac{1}{|D_{OPT}|}, & i\in D_{OPT},\\ 0, & i\notin D_{OPT}。\end{cases} \tag{3.3.3}$$

证明：定理 3.3.1 和定理 3.3.2 已保证平稳分布存在且唯一。下面仅证明(3.3.2)为平稳分布。由(1)知(3.3.2)定义的值为正数。又由(2)可知下列等式成立。

$$\begin{aligned}\sum_j v_j(t)p_{ji}(t)&=\sum_{j\neq i}\frac{\Phi(f(j),t)}{K}G_{ji}(t)A_{ji}(t)+v_i(t)p_{ii}(t)\\ &=\frac{\Phi(f(i),t)}{K}\sum_{j\neq i}G_{ij}(t)A_{ij}(t)+v_i(t)p_{ii}(t)\\ &=v_i(t)[\sum_{j\neq i}p_{ij}(t)+v_i(t)p_{ii}(t)]\\ &=v_i(t),\end{aligned}$$

其中，$K=\sum_{j\in D}\Phi(f(j),t)$。由定理 3.2.5 平稳分布和唯一性知(3.3.2)定义的值为平稳分布。

再由定理中的(3)(4)(5)知

$$\lim_{t\to 0} v_i(t)=\lim_{t\to 0}\frac{\Phi(f(i),t)}{\sum_j\Phi(f(j),t)}=\lim_{t\to 0}\frac{\dfrac{\Phi(f(i),t)}{\Phi(f_{OPT},t)}}{\dfrac{\sum_j\Phi(f(j),t)}{\Phi(f_{OPT},t)}}$$

$$=\lim_{t\to 0}\frac{\Phi(f(i)-f_{OPT},t)}{\sum_j\Phi(f(j)-f_{OPT},t)}=\begin{cases}\dfrac{1}{|D_{OPT}|}, & i\in D_{OPT},\\ 0, & i\notin D_{OPT},\end{cases}$$

其中，f_{OPT} 表示最优目标值。(3.3.3)得证。□

现在讨论(3.1.6)在一些特定的条件下，平稳分布的存在性和表达式形式。

定理 3.3.6 若 $A_{ij}(t),G_{ij}(t)$ 满足定理 3.3.3 中除(2)以外的条件，且定理 3.3.3 条件(2)改为：任何两个状态 i 和 j 或相互为邻居或不为，且 $G_{ij}(t)$ 满足

$$\forall i\in D, G_{ij}=\begin{cases}\dfrac{1}{|N(i)|}, & j\in N(i),\\ 0, & j\notin N(i),\end{cases}$$

状态空间 D 对邻域连通。则平稳分布是

$$v_i(t)=\frac{|N(i)|A_{i_0i}(t)}{\sum_{j\in D}|N(j)|A_{i_0j}(t)},\forall i\in D。 \tag{3.3.4}$$

证明：由状态空间对邻域连通的条件、定理 3.3.3 的(4)和(5)，得到定理 3.3.1 的条件满足，因此，马氏链不可约。很容易验证定理 3.3.2 的条件满足，所以非周期性得证。由定理 3.2.5 得到分布存在且唯一。下面验证(3.3.4)是平稳分布。

$\sum_j v_j(t)p_{ji}(t)$

$$= \sum_{j\neq i:i\in N(j)} \frac{|N(j)|}{K} A_{i_0 j}(t) G_{ji} A_{ji}(t) + \frac{|N(i)|}{K} A_{i_0 i}(t)(1 - \sum_{l=1,l\neq i}^{|D|} G_{il}(t) A_{il}(t))$$

$$+ \sum_{j:i\notin N(j)} \frac{|N(j)|}{K} A_{i_0 j}(t) G_{ji} A_{ji}(t)$$

$$= \sum_{j\neq i:i\in N(j)} \frac{1}{K} A_{i_0 j}(t) A_{ji}(t) + \frac{|N(i)|}{K} A_{i_0 i}(t)(1 - \sum_{l\in N(i),l\neq i} G_{il}(t) A_{il}(t))$$

$$= \sum_{j\neq i:i\in N(j)} \frac{1}{K} A_{i_0 j}(t) A_{ji}(t) + \frac{|N(i)|}{K} A_{i_0 i}(t) - \sum_{l\in N(i),l\neq i} \frac{1}{K} A_{i_0 i} A_{il}(t)$$

其中，$K = \sum_{j\in D} |N(j)| A_{i_0 j}(t)$。由任何两个状态或相互为邻居或不为，考察

$$\sum_{j\neq i:i\in N(j)} \frac{1}{K} A_{i_0 j}(t) A_{ji}(t) - \sum_{l\in N(i),l\neq i} \frac{1}{K} A_{i_0 i} A_{il}(t)$$

$$= \sum_{j\neq i:i\in N(j),f(i)\le f(j)} \frac{1}{K} A_{i_0 j}(t) + \sum_{j\neq i:i\in N(j),f(i)>f(j)} \frac{1}{K} A_{i_0 i}(t) \quad ,$$

$$- \sum_{l:l\in N(i),l\neq i,f(l)<f(i)} \frac{1}{K} A_{i_0 i} - \sum_{l:l\in N(i),l\neq i,f(l)\ge f(i)} \frac{1}{K} A_{i_0 l}$$

$$= 0。$$

因此，(3.3.4)是平稳分布。□

例 3.3.1 四个状态 1，2，3，4 的邻居分别为

N(1)={1,2}, N(2)={1,2,3,4}, N(3)={2,3,4}, N(4)={2,3,4}。

很容易验证，它们满足或互为邻居或互不为邻居的假设。但需要注意的是(3.3.4)不满足 $G_{ij} = G_{ji}$，如 $G_{12} = \frac{1}{2}$ 和 $G_{21} = \frac{1}{4}$。这个例子说明定理 3.3.6 成立的条件比定理 3.3.3 成立的条件要广泛。□

满足 $A(t)$和 $G(t)$为非周期、不可约条件，还有其他形式的一步转移概率，如文[3]给出的发生概率为

$$\exists |D| \times |D| \text{矩阵 Q 满足}$$

$$\forall i, j \in D : Q_{ij} = Q_{ji};$$

$$\forall i \in D, j \in N(i) : G_{ij} = \frac{Q_{ij}}{\sum_{l\in D} Q_{il}},$$

它的平稳分布为：

$$v_i(t) = \frac{(\sum_l Q_{il}) A_{i_0 i}(t)}{\sum_{j\in D} (\sum_l Q_{jl}) A_{i_0 j}(t)}, \forall i \in D。 \qquad (3.3.5)$$

(3.3.5)的证明同定理 3.3.6 的证明类似，读者可以自己证明。

文[4]给出接受概率为

$$A_{ij}(t) = (1 + \exp(-\frac{f(j) - f(i)}{t}))^{-1}。 \qquad (3.3.6)$$

它不满足定理 3.3.3 的(2)(3)，但它的平稳分布是

$$v_i(t)=\frac{\exp(-\frac{f(i)-f_{OPT}}{t})}{\sum_{j\in D}\exp(-\frac{f(j)-f_{OPT}}{t})},\forall i\in D。\tag{3.3.7}$$

3.4 非时齐算法收敛性简介

非时齐算法一步转移概率的形式为：

$$\begin{aligned}p_{ij}(k-1,k)&=P(X(k)=j\mid X(k-1)=i)\\&=\begin{cases}G_{ij}(t_k)A_{ij}(t_k), & \forall j\neq i,\\ 1-\sum_{l=1,l\neq i}^{|D|}G_{il}(t_k)A_{il}(t_k), & j=i,\end{cases}\end{aligned}\tag{3.4.1}$$

其中，$t_{k-1}\geq t_k$，$\lim_{k\to\infty}t_k=0$。一步转移概率矩阵为：$P(k-1,k)=\left(p_{ij}(k-1,k)\right)_{|D|\times|D|}$。多步转移概率 $p_{ij}(m,k)$ 表示第 m 步在状态 i，第 k 步在状态 j 的转移概率。

出于第 3.2 节对非时齐算法的要求，我们需要了解收敛性的一些定义。

定义 3.4.1 若非时齐马氏链满足下列条件，则称为弱遍历(weakly ergodic)：

$$\forall i,j,l\in D,\ m\geq 1,$$

$$\lim_{k\to\infty}(p_{il}(m,k)-p_{jl}(m,k))=0。$$

这个定义说明，无论起点如何，当运动步数增加时，到达同一状态的概率渐近相同。这是一种失去记忆功能。

定义 3.4.2 若存在向量 $v=(v_1,v_2,\cdots,v_{|D|})$，满足

$$\sum_{i=1}^{|D|}v_i=1,\quad \forall i,v_i\geq 0,$$

且

$$\forall i,j\in D,\ m\geq 1,$$

$$\lim_{k\to\infty}p_{ij}(m,k)=v_j,$$

则非时齐马氏链称为强遍历(strongly ergodic)。

定义 3.4.2 为非时齐马氏链的渐近稳定性。同时齐算法相同的思路，需要研究(3.4.1)的一些限定条件，使得非时齐马氏链具有定义 3.4.2 的强遍历性质。同时，使得分布 $v=(v_1,v_2,\cdots,v_{|D|})$ 具有全局最优的性质。我们不加证明地给出下面的定理。

定理 3.4.1[5] 非时齐马氏链是弱遍历的充分必要条件是存在一个严格上升的正序列 $\{k_l\}_{l=0,1,\cdots}$ 使得

$$\sum_{l=0}^{\infty}(1-\tau_1(P(k_l,k_{l+1})))=\infty,\tag{3.4.2}$$

其中，$\tau_1(P)=1-\min_{i,j}\sum_{l=1}^{n}\min(p_{il},p_{jl})$，$P=(p_{ij})_{n\times n}$。

定理 3.4.2[6] 非时齐的马氏链是强遍历的充分必要条件是马氏链是弱遍历且存在 $P(k\text{-}1,k)$ 的特征值为 1 的特征向量 $V(k)=(v_1(k),v_2(k),\cdots,v_{|D|}(k))$ 使得 $\sum_{i=1}^{|D|}|v_i(k)|=1$，且

$$\sum_{k=0}^{\infty}\sum_{i=1}^{|D|}|v_i(k)-v_i(k+1)|<\infty,\tag{3.4.3}$$

另外，若$V=\lim_{k\to\infty}V(k)$，则 V 满足$\lim_{k\to\infty}p_{ij}(m,k)=v_j$。

分析定理 3.4.1 和定理 3.4.2，在给定温度t_k和一步转移概率满足一定的条件时，由 3.3 节关于时齐算法的讨论，存在平稳分布$V(t_k)=(v_1(t_k),v_2(t_k),\cdots,v_{|D|}(t_k))$。现在的主要内容是讨论如何选择$t_k$，使得：

（1）马氏链满足定理 3.4.1 的(3.4.2)，

（2）$V(t_k)=(v_1(t_k),v_2(t_k),\cdots,v_{|D|}(t_k))$满足(3.4.3)，

（3）极限分布 V 满足全局收敛$\lim_{k\to\infty}P\{X(k)\in D_{OPT}\}=1$。

定理 3.4.3 在温度 *t* 时，一步转移概率(3.1.4)中的 *A*(t)按(3.1.6)定义，即

$$A_{ij}(t)=\begin{cases}1, & f(i)\geq f(j),\\ \exp(-\dfrac{\Delta f_{ij}}{t}), & f(i)<f(j),\end{cases}$$

G(*t*)为

$$\forall i\in D, G_{ij}=\begin{cases}\dfrac{1}{|D|}, & j\in N(i),\\ 0, & j\notin N(i),\end{cases} \tag{3.4.4}$$

当

$$\text{存在 } k_0\geq 2\text{，使得 } \forall k\geq k_0, t_k\geq\frac{|D|\Delta f_{\max}}{\log k}\text{，} \tag{3.4.5}$$

其中，

$$\Delta f_{\max}=\max\{f(i)\mid i\in D\}-\min\{f(i)\mid i\in D\}, \tag{3.4.6}$$

则马氏链$\{X(t_k)\}_{k=1,2,\cdots}$为强遍历。

文[7]首先证明了满足定理 3.4.3 条件的马氏链是弱遍历。再由定理 3.3.3，(3.1.6)和(3.4.4)确定的马氏链有(3.3.7)的平稳分布，即

$$v_i(t_k)=\frac{\exp(-\dfrac{f(i)-f_{OPT}}{t_k})}{\sum_{j\in D}\exp(-\dfrac{f(j)-f_{OPT}}{t_k})},\forall i\in D. \tag{3.4.7}$$

可以验证(3.4.7)满足(3.4.3)（留作练习），于是，非时齐马氏链是强遍历。由定理 3.3.4 和 *A*(*t*)的性质，本定理对应的非时齐马氏链收敛到全局最优解，即

$$\lim_{k\to\infty}P(X(k)\in D_{OPT})=1\text{。}$$

另外，温度下降的最快速度为

$$t_k=\frac{|D|\Delta f_{\max}}{\log k}\text{。} \tag{3.4.8}$$

关于非时齐算法的下降速度还有一些其他精细估计。在定理 3.4.3 的条件下，文[8]中证明了：温度下降满足

$$\forall k\geq 2, t_k\geq\frac{n\Delta_0}{\log k} \tag{3.4.9}$$

其中，$\Delta_0=\max_{i,j}\{f(j)-f(i)\mid i\in D, j\in N(i)\}$，*n* 是从任何一个状态可达到最优解的转移步数的上限，明显不超过|D|，*k* 是 *n* 的倍数。

非时齐马氏链收敛的一个充分必要在[9]中给出，先给谷深的定义。

定义 3.4.3 称为一个状态 j 由 i 以高度 L 可达，若存在 $i=l_0,l_1,\cdots,l_p=j$ 使得

$$G_{l_k l_{k+1}}>0,k=0,1,\cdots,p-1,$$
$$f(l_k)\le L,k=0,1,\cdots,p,$$

其中，$G(t)$为(3.1.4)的产生概率。一个深度不超过 L 的谷定义为V_L，满足

$$V_L=\{j\in D\mid \forall i\in V_L,i \text{ 以高度 L 达 } j\}。$$

谷的实际深度可以由

$$\underline{V_L}=\min\{f(j)\mid j\in V_L\},$$
$$\overline{V_L}=\min\{f(j)\mid j\notin V_L\wedge\exists i\in V_L,G_{ij}>0\},$$

的差值决定，即谷深 H=$\overline{V_L}-\underline{V_L}$。谷中的一个局部最小点 i 可以看成状态 i 无法在高度 $f(i)$ 达到其他状态 j。

定理 3.4.4 若一步转移概率由(3.4.1)的形式给出，且 $A(t_k)$取(3.1.6)形式，$G(t_k)$满足

(1) 对任给 i 和 j，满足：存在 $i=l_0,l_1,\cdots,l_p=j$ 使得

$$G_{l_k l_{k+1}}>0,k=0,1,\cdots,p-1;$$

(2) 对一实数 L 和任意两个状态 i 和 j，i 以高度 L 可达 j 的充分必要条件是 j 以高度 L 可达 i;

(3)$t_k\ge t_{k+1},\lim_{k\to\infty}t_k=0$;若 H 是所有不包含最优解谷的最大谷深，则有

$$\lim_{k\to\infty}P(X(k)\in D_{OPT})=1$$

的充分必要条件是

$$\sum_{k=1}^{\infty}\exp(-\frac{H}{t_k})=\infty。 \tag{3.4.10}$$

从定理 3.4.3 和定理 3.4.4 可以看出，模拟退火非时齐算法要全局收敛的一个充分条件是温度下降具有形式

$$t_k=\frac{\Gamma}{\log k},k>1, \tag{3.4.11}$$

其中Γ是一个同问题参数相关。如定理 3.4.3 中，

$$\Gamma\ge|D|\Delta f_{\max},$$

(3.4.9)式中$\Gamma\ge n\Delta$，定理 3.4.4 中的Γ不小于所有不包含最优解谷的最大谷深。

3.5 实现的技术问题

前面各节介绍了模拟退火的算法结构、模型、时齐和非时齐算法的收敛理论。本节从应用的角度讨论算法实现中的一些技术问题。主要包括解的形式、邻域的构成、初始温度的选取、温度下降的规则、每一温度马氏链的迭代步长和停止规则等。非时齐算法的收敛理论表明，其下降的规则是有固定形式的，因此，本节所讨论的技术问题以时齐算法为主。

前几节的理论研究可以看出，根据理论要求达到平稳分布来实现模拟退火算法是不可能的，这是因为时齐的算法要在每一个温度迭代无穷步，以达到平稳分布，而非时齐要求温度下降的迭代步数是指数次的。从应用的角度来看，我们只需在可接受的计算时间里得到满意的解。因此本节介绍的技术问题无法保证模拟退火算法得到全局最优解。应用这些技术的模拟退火算法还是一种启发式算法。

由于本节是讨论应用的技术问题，因此，它同理论部分有一定的联系，同时可能相差较

远。部分技术问题虽说给出了一些理论结果，这些结果也只是指导性的。

3.5.1 解的形式和邻域结构

解的形式和邻域结构是一个老话题，在第二章中已讨论。本节再度讨论这个问题以引起读者的注意。解的表现形式直接决定于邻域的构造。

对求 $f(x)=x^2$ 在区域 $0\le x\le 100$ 最大值的实例，其中 x 为整数，采用 0-1 编码表示解，可以表示为

$$S=\begin{pmatrix}1 & 2 & \cdots & 7\\ * & * & \ldots & *\end{pmatrix},$$

其中，解 S 的上面一行表示 7 个位置，下面的*是 0-1 码，对应 x 的二进制码，解的一个邻域自然可以构造成

$$N(S)=\{S'\mid S'\text{是一个}0-1\text{码且}\mid S-S'\mid=\sum_{i=1}^{7}\mid s_i-s_i'\mid\le k,k\ge 1\text{整数}\}。$$

对 TSP 这一类问题，可采用 n 个城市的一个排序表示问题的一个解。很直观可通过城市间不同位置交换构造邻域。

上面两个示例中，两种邻域表示的共同特征是：第一，每一个邻域所含的状态个数相同，第一种情况每个解的邻域中都有 $C_7^k+C_7^{k-1}+\cdots+C_7^0$ 个邻居，TSP 中采用 2-opt 时每个邻域有 C_n^2+1 个邻居；第二，每个邻居都是可行解；第三，解空间中的任何两个状态可达。由时齐算法的理论，只要接受概率满足一定的条件，一定以概率 1 收敛到全局最优解。

同样解的表达形式和邻域结构，将第一种表示形式应用到背包问题时，第二种表示形式应用到车间作业问题(第二章 2.4.2 节)时，就会出现新的问题。由于背包问题有包容积约束和车间作业的死锁现象，无法保证每个邻域中邻居都是可行解。因此，理论上就无法保证收敛到全局最优解。为了满足收敛到全局最优解的条件，处理这类问题的常用方法有两种。第一种是罚值法。将不可行解视为可行解，目标值为一个充分大的数（罚值）。这使得问题转化为上面讨论的两个问题，原有的方法可以继续使用。罚值法处理不可行解时，一个主要缺陷是扩大了搜索区域，从而使计算时间增加。为了克服这个不足，第二种方法是研究解和邻域结构，从理论上保证模拟退火算法以概率 1 收敛到全局最优解。这一个方法要求对问题有相当深入的了解，因此，可以说其难度是比较大的。通过下面的例子来了解罚函数引起了搜索区域的扩大。

例 3.5.1 背包问题：三个物品，尺寸分别为{5,3,1}，包的容积为 4，则{ (0,0,0), (0,1,0), (0,0,1), (0,1,1)}是可行解集合。采用罚值法时，解{(1,0,0), (1,1,0), (1,0,1), (1,1,1)}也成了搜索的对象，于是搜索空间扩大了一倍。□

罚值法可以很简单地保证理论要求的不可约和非周期性。但如此计算一定会增加计算时间，有时增加时间之多是不可以接受的。如按例 1.1.2 的 TSP 的表示方法，若商人走(i,j)弧则 $x_{ij}=1$，否则 $x_{ij}=0$。这样每一个解的维数是 $n\times(n-1)$，每个分量取 0 或 1，一共有 $2^{n(n-1)}$ 个解，其中只有一部分是可行解。假设每两个节点有弧相连，固定一个城市为始终点，任意一个解用城市的排列表示，则可行解数为(n-1)!。若采用例 1.1.2 解表达形式和对不可行解采用罚值法，则将搜索 $2^{n(n-1)}-(n-1)!$ 个不可行解。搜索空间扩大了 $\dfrac{2^{n(n-1)}}{(n-1)!}$ 倍。当 n=5 时，扩大倍数 $\dfrac{2^{n(n-1)}}{(n-1)!}\approx 4.37\times 10^4$，当 n=10 时，扩大倍数 $\dfrac{2^{n(n-1)}}{(n-1)!}\approx 3.41\times 10^{21}$。

为了节省计算时间，可以考虑在可行解集合内构造邻域，如第二章车间作业问题的邻域构造，但需要类似 3.3 和 3.4 节那样从理论上证明对应的马氏链的全局最优性。

在此，邻域的构造同计算时间紧密联系在一起，也就产生了对实际问题的理论分析同实际应用的矛盾。从科学的角度，我们希望对问题进行较深入的理论研究，构造比较精巧的邻域结构，这样可以节省大量的计算时间。不得不承认，在大量的实际应用中，受各种客观因素的影响，人们对问题还无法深入地了解，且在有些情况下，人们关注的是应用的效果，而不去研究问题的结构和收敛性就直接将算法套用于实际问题。

3.5.2 温度参数的控制

温度参数是模拟退火时齐算法的关键参数之一。主要包括起始温度的选取、温度的下降方法和停止温度的确定等。

起始温度的选取

从理论上来说，起始温度t_0应保证平稳分布中每一状态的概率相等，也就是使

$$\exp(-\frac{\Delta f_{ij}}{t_0}) \approx 1,$$

其中，$\Delta f_{ij} = f(j) - f(i)$。很容易估计一个值为

$$t_0 = K\Delta_0,\ K\text{充分大的数}, \tag{3.5.1}$$

其中，

$$\Delta_0 = \max(f(i) \mid i \in D\} - \min\{f(i) \mid i \in D\}。$$

实际计算中，可以选 K=10,20,100…等试验值。

对一些问题，有时可以简单地估计Δ_0。如例 1.1.2 的 TSP，记对应的距离矩阵为$(d_{ij})_{n\times n}$，则

$$\max\{f(x) \mid x \in D\} \le \sum_{i=1}^{n} \max\{d_{ij} \mid j \ne i, j = 1,2,\cdots,n\},$$

$$\min\{f(x) \mid x \in D\} \ge \sum_{i=1}^{n} \min\{d_{ij} \mid j \ne i, j = 1,2,\cdots,n\},$$

$$\Delta_1 = \sum_{i=1}^{n} \max\{d_{ij} \mid j \ne i, j = 1,2,\cdots,n\} - \sum_{i=1}^{n} \min\{d_{ij} \mid j \ne i, j = 1,2,\cdots,n\}。$$

则可用Δ_1替代(3.5.1)中的Δ_0。

再如 2.4.2 的车间作业调度问题，记第$i(1 \le i \le n)$工件的第$j(1 \le j \le n_i)$工序的加工时间为p_{ij}。完成的最短时间是所有工件都从时刻零开始加工且每一台机器都没有闲置，此时

$$\min\{f(x) \mid x \in D\} \ge \max\left(\sum_{j=1}^{n_i} p_{ij} \middle| i = 1,2,\cdots,n\right)。$$

完成的最长时间是调度最坏情况

$$\max\{f(x) \mid x \in D\} \le \sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{ij}。$$

令

$$\Delta_2 = \sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{ij} - \max\left(\sum_{j=1}^{n_i} p_{ij} \middle| i = 1,2,\cdots,n\right)。$$

则可用Δ_2替代(3.5.1)中的Δ_0。

有时候，会出现Δ_0比较难估计或很难精确地知道最大值和最小值而使上面的估计太粗。

另外，K 过大造成计算时间的增加，K 过小则使算法过早陷入局部最优点。由此产生数值计算估计和统计推断估计方法。

数值计算估计方法的基本思想是给出一个值 χ_0（χ_0 接近 1，如 χ_0=0.9,0.8 等），对给定的初始温度 t_0，用以下的算法

初始温度数值计算算法

STEP1 给定一个常量 T；初始温度 t_0；χ_0；R_0=0；k:=1;

STEP2 在这个温度迭代 L 步（L 为一个给定的常数），分别记录模拟退火算法中接受和被拒绝的状态的个数，计算接受的状态数同迭代步数 L 的比率 R_k;

STEP3 当$|R_k-\chi_0|<\varepsilon$时，停止计算；否则，当 R_{k-1} 和 $R_k<\chi_0$ 时，则 k:=k+1，$t_0:=t_0$ +T，返回 STEP2；当 R_{k-1} 和 $R_k\geq\chi_0$ 时，则 k:=k+1，$t_0:=t_0$ -T，返回 STEP2；当 $R_{k-1}\geq\chi_0$ 且 $R_k\leq\chi_0$ 时，则 k:=k+1，$t_0:=t_0$ +T/2，T:=T/2，返回 STEP2；当 $R_{k-1}\leq\chi_0$ 且 $R_k\geq\chi_0$ 时，则 k:=k+1，$t_0:=t_0$ —T/2，返回 STEP2；

通过数值计算，可以估计出温度 t_0。

统计推断估计的方法是针对(3.5.1)中 $\Delta_0=\max(f(i)\,|\,i\in D\}-\min\{f(i)\,|\,i\in D\}$ 比较难得到，通过统计的方法估计费用函数的上下限。统计的方法总是在一定的概率分布下研究问题的均值、方差等。假设 $\{f(i)\,|\,i\in D\}$ 是一个大样本空间，且服从正态分布，即 $\{f(i)\,|\,i\in D\}$ 的密度函数为

$$g(x)=\exp(-\frac{(x-\mu)^2}{2\sigma^2})。$$

从状态空间 D 中随机选样本容量为 n 的样本 $\{X_l\,|\,l=1,2,\cdots,n\}$，样本均值统计量为

$$\overline{f(X,t)}=\frac{1}{n}\sum_{l=1}^{n}f(X_l),\qquad(3.5.2)$$

样本方差统计量为

$$f^2(X,t)=\frac{1}{n-1}\sum_{l=1}^{n}(f(X_l)-\overline{f(X,t)})^2。\qquad(3.5.3)$$

由概率论的分布特性，(3.5.2)，(3.5.3)分别是μ，σ^2 的统计量。用它们作为近似，以 3σ原则，目标值以 99.97%落入(μ-3σ, μ+3σ)(参考[10])，简单的区间近似为

$$(\overline{f(X,t)}-3\sqrt{f^2(X,t)},\ \overline{f(X,t)}+3\sqrt{f^2(X,t)})。$$

估计(3.5.1)的

$$\Delta_0=6\sqrt{f^2(X,t)}。$$

时齐算法的温度下降方法

由于非时齐算法温度下降具有固定的形式（见 3.4 节），因此，这里只讨论时齐算法的温度下降方法。从实际应用来看，非时齐算法的下降速度太慢，可以有变通的形式以利于节省计算时间，在此不讨论。

时齐算法的理论要求温度下降到零，整个系统以概率 1 收敛全局最优解。无论直观理解还是理论要求，温度总是下降的，因此，一个非常直观的下降方法是

(1) $t_{k+1}=\alpha t_k, k\geq 0$，其中 0<α<1。α越接近 1 温度下降的越慢。这种方法简单易行，很受应用人员的欢迎。它的每一步以相同的比率下降。

(2) $t_k=\frac{K-k}{K}t_0$，其中 t_0 为起始温度，K 为算法温度下降的总次数。这一下降的方法的优点是易于操作，而且可以简单地控制温度下降的总步数。它的每一步下降长度相等。

(3)以理论中的平稳分布$\{v_i(t), i \in D\}$为依据，通过推导而得到。为了方便以后的讨论，首先在温度t，定义

$$\langle f(t)\rangle = \sum_{i\in D} f(i)v_i(t), \tag{3.5.4}$$

$$\langle f^2(t)\rangle = \sum_{i\in D} f^2(i)v_i(t), \tag{3.5.5}$$

$$\overline{f(t)} = \frac{1}{n}\sum_{l=1}^{n} f(X_l), \tag{3.5.6}$$

$$\overline{f^2(t)} = \frac{1}{n}\sum_{l=1}^{n} f^2(X_l)。 \tag{3.5.7}$$

(3.5.4)和(3.5.5)分别是平稳分布时的期望值和二阶矩。(3.5.6)和(3.5.7)为温度 t 模拟退火 n 个样本的均值和二阶矩。为以后使用方便，先讨论(3.5.4)和(3.5.5)的一些基本性质。

定理 3.5.1 若 $A(t)$和 $G(t)$满足定理 3.3.6 的条件，且 $A(t)$和 $G(t)$分别满足(3.1.6)和(3.1.7)，则

(1) $\dfrac{d\langle f(t)\rangle}{dt} = \dfrac{\langle f^2(t)\rangle - \langle f(t)\rangle^2}{t^2}$,

(2) $\langle f(t)\rangle$为变量 t 的单调升函数，且$\langle f(t)\rangle \to f(i_{OPT}), t \to 0$。

(3)当$i \neq i_{OPT}$时，平稳分布$v_i(t)$在 t=0 的一个邻域内为 t 的单调升函数。当$i = i_{OPT}$时，平稳分布$v_i(t)$为 t 的单调减函数。

证明：(1) $\dfrac{d\langle f(t)\rangle}{dt} = \sum_{i\in D} f(i)\dfrac{dv_i(t)}{dt}$。

由

$$v_i(t) = \frac{|N(i)|\exp(-\frac{f(i)-f_{OPT}}{t})}{\sum_{j\in D}|N(i)|\exp(-\frac{f(j)-f_{OPT}}{t})} = \frac{|N(i)|\exp(-\frac{f(i)}{t})}{\sum_{j\in D}|N(i)|\exp(-\frac{f(j)}{t})} \quad \forall i \in D,$$

$$\frac{dv_i(t)}{dt} = \frac{d\left(\dfrac{|N(i)|\exp(-\frac{f(i)}{t})}{\sum_{j\in D}|N(j)|\exp(-\frac{f(j)}{t})}\right)}{dt}$$

$$= \frac{|N(i)|\exp(-\frac{f(i)}{t})f(i)}{t^2K} - \frac{|N(i)|\exp(-\frac{f(i)}{t})\sum_{j\in D}|N(j)|\frac{f(j)}{t^2}\exp(-\frac{f(j)}{t})}{K^2}$$

$$= \frac{v_i(t)f(i)}{t^2} - \frac{v_i(t)\langle f(t)\rangle}{t^2},$$

其中，$K=\sum_{j\in D}|N(j)|\exp(-\dfrac{f(j)}{t})$。于是，可得

$$\frac{d\langle f(t)\rangle}{dt} = \frac{\langle f^2(t)\rangle - \langle f(t)\rangle^2}{t^2}。$$

(2) 由

$$\frac{d\langle f(t)\rangle}{dt}=\frac{\langle f^2(t)\rangle-\langle f(t)\rangle^2}{t^2}=\frac{\sum_{i\in D}(f(i)-\langle f(i)\rangle)^2 v_i(t)}{t^2}\geq 0,$$

得$\langle f(t)\rangle$对 t 单调上升。再由定理 3.3.4 很易证$\langle f(t)\rangle \to f(i_{OPT}), t\to 0$。

(3) 由(1)的证明，

$$\frac{dv_i(t)}{dt}=\frac{v_i(t)f(i)}{t^2}-\frac{v_i(t)\langle f(t)\rangle}{t^2}。$$

当$i\neq i_{OPT}$时，若 $f(i)-f(i_{OPT})=\varepsilon>0$，由(2)的结论，存在$\delta>0$, 当$0\leq t<\delta$ 时，$\langle f(t)\rangle<f(i_{OPT})+\varepsilon$。于是，$\frac{dv_i(t)}{dt}>0$，故$v_i(t)$在 t=0 的一个邻域内为 t 的单调升函数。同法可证，当$i=i_{OPT}$时，平稳分布$v_i(t)$为 t 的单调减函数。□

假设在相邻温度，其平稳分布的变化相对稳定。由定理 3.5.1，平稳分布的变化相对稳定等价于用数学公式表达：给定一个较小的正数δ，相邻温度的平稳分布应满足

$$\forall i\in D,\ \frac{1}{1+\delta}<\frac{v_i(t_k)}{v_i(t_{k+1})}<1+\delta, k=0,1,\cdots。 \tag{3.5.8}$$

按定理 3.3.6 且 $A(t)$满足(3.1.7)，(3.5.8)近似为

$$\exp(-\frac{f(i)-f(i_{OPT})}{t_k})<(1+\delta)\exp(-\frac{f(i)-f(i_{OPT})}{t_{k+1}})。 \tag{3.5.9}$$

(3.5.9)变形为

$$\frac{f(i)-f(i_{OPT})}{t_{k+1}}<\log(1+\delta)+\frac{f(i)-f(i_{OPT})}{t_k}, \tag{3.5.10}$$

进一步可得

$$t_{k+1}(1+\frac{t_k\log(1+\delta)}{f(i)-f(i_{OPT})})>t_k。 \tag{3.5.11}$$

当模拟退火算法对应的马氏链时齐、非周期和不可约时，由定理 3.2.5，在温度t_k，$\{f(i)\,|\,i\in D\}$ 服从 $\{v_1(t_k),v_2(t_k),\cdots,v_{|D|}(t_k)\}$ 分布，其对应的均值和二阶矩分别为$<f(t_k)>$和$<f^2(t_k)>$，均值的统计量为

$$u(t_k)=\overline{f(t_k)},$$

方差的统计量为

$$\sigma^2(t_k)=\overline{f^2(t_k)}-(\overline{f(t_k)})^2。$$

用概率论的中心极限定理的思想，$\{f(i)\,|\,i\in D\}$ 的上下界可用下面区间近似

$$(-3\sigma(t_k)+\mu(t_k),\ \mu(t_k)+3\sigma(t_k))。$$

由上面的近似结果，$t_{k+1}(1+\frac{t_k\log(1+\delta)}{f(i)-f(i_{OPT})})$ 可以用 $t_{k+1}(1+\frac{t_k\log(1+\delta)}{6\sigma(t_k)})$ 估计。当

$$t_{k+1}(1+\frac{t_k\log(1+\delta)}{6\sigma(t_k)})>t_k$$

时，选择温度变化

$$t_{k+1}=t_k(1+\frac{t_k\log(1+\delta)}{6\sigma(t_k)})^{-1}。 \tag{3.5.12}$$

计算 $\mu(t_k)$，$\sigma(t_k)$时，要求对每一个温度t_k采用累计求和的方法，而不需要记录所有的状态。

Lundy 和 Mees[11]采用基本相同的想法，使(3.5.8)近似为

$$\forall i \in D : \exp(\frac{(f(i)-f(i_{OPT}))(t_k - t_{k+1})}{t_k t_{k+1}}) < (1+\delta),$$

当 U 是一个 $f(i)-f(i_{OPT})$ 的上界，r 是一个充分小的正数，保证上式成立的条件为：

$$\frac{t_k - t_{k+1}}{t_k t_{k+1}} = \frac{r}{U},$$

于是可以得降温规则

$$t_{k+1} = t_k (1+\frac{rt_k}{U})^{-1}。 \tag{3.5.13}$$

文[12]基于 $A(t)$是(3.1.7)的形式，得（练习题）

$$\frac{d}{d\log t}\langle f(t)\rangle = \frac{\langle f^2(t)\rangle - \langle f(t)\rangle^2}{t},$$

近似为

$$\frac{\overline{f(t_{k+1})} - \overline{f(t_k)}}{\log t_{k+1} - \log t_k} \approx \frac{\overline{f^2(t_k)} - \overline{f(t_k)}^2}{t_k},$$

由此推导

$$t_{k+1} = t_k \exp\left(\frac{t_k(\overline{f(t_{k+1})} - \overline{f(t_k)})}{\overline{f^2(t_k)} - \overline{f(t_k)}^2}\right)。$$

若要求 $\overline{f(t_{k+1})} - \overline{f(t_k)}$ 不超过它的标准差，即

$$\overline{f(t_{k+1})} - \overline{f(t_k)} = -\lambda\sqrt{\overline{f^2(t_{k+1})} - \overline{f(t_k)}^2}, 0 < \lambda < 1,$$

则有[12]中的结果

$$t_{k+1} = t_k \exp\left(-\frac{t_k \lambda}{\sqrt{\overline{f^2(t_k)} - \overline{f(t_k)}^2}}\right)。 \tag{3.5.14}$$

时齐算法每一温度的迭代长度规则

同样，实际计算中达到理论的平稳分布是不可能的，只能近似这一结果。下面就常用的方法给予讨论。

（1）固定长度。

这是一个简单的方法。在每一温度，迭代相同的步数。步数的选取同问题的大小有关。通常采用与邻域大小直接相关的规则。如 TSP 的两城市位置交换所定义的邻域，它的大小是 $\frac{n(n-1)}{2}$（其中 n 是城市数），在同一温度，它的迭代步长可取 $n, \frac{n}{2}, n^2, \frac{n^2}{2}, n^k (k \geq 3)$ 等。

（2）由接受和拒绝的比率控制迭代步数

当温度很高时，每一个状态被接受的频率基本相同，而且几乎所有的状态都被接受。此时，在同一温度应使迭代的步数尽量小。当温度渐渐变低时，越来越多的状态被拒绝。如果在此温度的迭代太少，则可能造成过早地陷入局部最优状态。比较直观和有效的方法是随着温度的下降，将同一温度的迭代步长增加。实现的一种方法是给定一个充分大的步长上限 U 和一个接受次数指标 R，当在给定温度接受次数等于 R 时，在这一温度不再迭代而使得温度下降，否则，一直迭代到上限步数。实现的第二种方法是给定一个接受比率指标 R、迭代步长上限 U 和下限 L，每一温度至少迭代 L 次且记录同一温度迭代的总次数和被接受的次数，

当迭代数超过 L 时，若接受次数同总次数的比率不小于 R 时，在这一温度不再迭代而开始温度下降，否则，一直迭代到上限步数。

同样可以用拒绝次数为指标类似上面的讨论得到一些控制迭代步数的规则。

（3）概率控制法

以概率 1 跳出任何一个局部最优解是概率控制法的一个基本思想。设状态 i 是一个局部最优状态，$p_{ii}(t)$ 为 i 状态一步转移不动的概率，n 步保持不动的概率是 $p_{ii}^{n}(t)$。于是温度 t 时状态 i 不转移的平均次数

$$\tilde{N}_i(t)=\sum_{n=1}^{\infty} np_{ii}^{n}=\frac{p_{ii}}{(1-p_{ii})^2}\le\frac{1}{(1-p_{ii})^2}。$$

当产生概率为(3.1.6)和接受概率为(3.1.7)时，

$$p_{ii}(t)=1-\frac{1}{|N(i)|}\sum_{\substack{j\in N(i)\\ j\neq i}}\min\{1,\exp\left(-\frac{f(j)-f(i)}{t}\right)\}。$$

于是

$$\tilde{N}_i(t)\le\frac{1}{(1-p_{ii})^2}=\frac{|N(i)|^2}{\min\{1,\exp\left(-\frac{f(j)-f(i)}{t}\right)\}^2},$$

近似估计

$$\tilde{N}_i(t)\approx\frac{|N(i)|^2}{\min\left\{1,\exp\left(-\frac{f_{\max}-f_{\min}}{t}\right)^2\right\}},\qquad(3.5.15)$$

其中，$f_{\max}$，$f_{\min}$ 分别为整个状态空间的最大和最小费用值。从(3.5.15)的估计可以看出：当温度较高时，跳出局部最优的平均次数为 $O(|N(i)|^2)$，这与迭代长度规则（1）的估计基本吻合；当温度变低时，跳出局部最优的平均次数增加非常快，增加的速度与 $e^{\frac{1}{t}}$ 同阶。由此可以看出，在实际应用中，(3.5.15)也是不适用的。实际应用中，迭代长度规则（1）和（2）比较容易实现。

算法的终止原则

模拟退火算法从初始温度开始，通过在每一温度的迭代和温度的下降，最后达到终止原则而停止。尽管有些原则有一定理论的指导，终止原则大多是直观的。下面分类讨论。

（1）零度法

模拟退火的最终温度为零。因而最为简单的原则是：给定一个比较小的正数 ε，当温度 $t_k\le\varepsilon$ 时，算法停止。表示已经达到最低温度。

（2）循环总数控制法

这一简单原则在时齐算法的温度下降方法（2）中已经提及，即总的温度下降次数为一定值 K，当温度迭代次数达到 K 时，停止运算。这一原则可分为两类，一类是整个算法的总迭代步数为一定数，它表示各温度时马氏链迭代数（内循环）的总和为一个给定的数。这样很容易计算算法的计算复杂性，但需要合理分配内循环的长度和温度下降的次数。另一类是内循环的次数由迭代长度规则（2）或（3）决定，温度下降次数（外循环）为一个定值。这样的控制法对估计算法的复杂性有一定的困难。解决的方法一是通过理论的估计，二则是类似迭代长度规则（2）给出每一温度的迭代长度上限。

（3）基于不改进规则的控制法

在一个温度，在给定的迭代次数内没有改进当前的局部最优解，则停止运算。模拟退火

的一个基本思想是跳出局部最优解。直观的结论是在较高的温度没能跳出局部最优解，则在低的温度跳出最优解的可能也比较小。由此产生上面的停止原则。

（4）接受概率控制法

该方法与终止原则（3）相同的思想。给定一个指标 $\chi_f > 0$ 是一个比较小的数，在给定温度，除局部最优解，其他状态的接受概率都小于 χ_f 时，停止运算。实现终止原则（3）或（4）时，记录当前局部最优解，给定一个固定的迭代次数，在规定的迭代次数里没有离开局部最优解或每一次计算的接受概率都小于 χ_f，则在这个温度停止计算。

（5）邻域法

若采用定理 3.3.6 的发生概率和(3.1.7)的接受概率，且设 f_0 和 f_1 分别为一个邻域内的局部最优和次最优值，当满足

$$\exp(-\frac{f_1 - f_0}{t}) < \frac{1}{N} \tag{3.5.16}$$

时（其中 N 为邻域的大小），局部最优到次优的接受概率满足(3.5.16)，而从局部最优到其他费用更高的状态的接受概率更小。直观的想法是邻域中每次至少有一个状态被接受，但(3.5.16) 满足时，除局部最优解以外状态的接受概率都小于邻居的平均数，此时可以认为从局部最优解转移到其他状态的可能性很小，因此停止。通过(3.5.16)可得终止温度

$$t_f \le \frac{f_1 - f_0}{\log N}。 \tag{3.5.17}$$

（6）Lundy 和 Mees 方法

Lundy 和 Mees[11]从概率的意义给出一个判定方法。给定充分小的正数δ和ε，在达到终止温度应该满足

$$P(X(k) = i \wedge f(i) > f_{OPT} + \varepsilon \mid t = t_f) < \delta,$$

其中∧表示逻辑的“与”。由平稳分布所具有的性质，在温度 t，$P(X(k) = i) = v_i(t)$，可以近似为

$$P(X(k) = i \wedge f(i) > f_{OPT} + \varepsilon \mid t = t_f) \approx \sum_{i: f(i) > f_{OPT} + \varepsilon} v_i(t)$$

$$< (|D| - 1)\exp(-\frac{\varepsilon}{t}) < \delta,$$

直接推导可得终止温度

$$t_f \le \frac{\varepsilon}{\log(|D| - 1) - \log\delta}。 \tag{3.5.18}$$

（7）Aarts 和 Van Laarhoven 法

Aarts 和 Van Laarhoven[3]在(3.1.6)和(3.1.7)的条件下，用概率分析的方法给出终止温度的精细估计，他们的基本思想是根据(3.5.4)期望费用

$$\langle f(t) \rangle = \sum_{i \in D} f(i) v_i(t)$$

的函数特性(定理 3.5.1)，

$$\frac{d\langle f(t) \rangle}{dt} = \frac{1}{t^2} \sum_{i \in D} \{ f(i) - \langle f(t) \rangle \}^2 v_i(t)。$$

在终止温度 t_f，由泰勒(Taylor)展开的第一项，费用均值近似为

$$< f(t_f) > - f_{OPT} \approx t_f * \left. \frac{d\langle f(t) \rangle}{dt} \right|_{t = t_f}。 \tag{3.5.19}$$

当(3.5.19)的右端充分小的时候，则左端也较小，可以认为已达到要求。用数学公式描述为：

对给定的$\varepsilon>0$，当

$$\left.\frac{d\langle f(t)\rangle}{dt}\right|_{t=t_f}\frac{t_f}{\langle f(t_0)\rangle}<\varepsilon, \tag{3.5.20}$$

或

$$\left.\frac{d\langle f(t)\rangle}{dt}\right|_{t=t_f}\frac{t_f}{\langle f(t_0)\rangle-\langle f(t_f)\rangle}<\varepsilon, \tag{3.5.21}$$

时，计算停止。(3.5.20)和(3.5.21)可以用样本均值和样本矩近似替代

$$\frac{\langle f^2(t_f)\rangle-\langle f(t)\rangle^2}{t_f\langle f(t_0)\rangle}<\varepsilon, \tag{3.5.22}$$

$$\frac{\langle f^2(t_f)\rangle-\langle f(t)\rangle^2}{t_f(\langle f(t_0)\rangle-\langle f(t_f)\rangle)}<\varepsilon。 \tag{3.5.23}$$

当满足(3.5.22)或(3.5.23)时，计算停止。用样本值近似

$$\frac{\overline{f^2(t_f)}-\overline{f(t)}^2}{t_f\overline{f(t_0)}}<\varepsilon, \tag{3.5.24}$$

$$\frac{\overline{f^2(t_f)}-\overline{f(t)}^2}{t_f(\overline{f(t_0)}-\overline{f(t_f)})}<\varepsilon。 \tag{3.5.25}$$

理论上是用一个马尔可夫链描述模拟退火算法的变化过程，因此具有全局最优性。实际应用中的模拟退火算法是一个启发式算法。它有诸多的参数需要调整，如起始温度、温度下降的方案、固定温度时的迭代长度及终止规则等，这样需要人为地调整。人为的因素，如对问题的了解、参数和规则的搭配等，造成计算结果的差异。解决这个矛盾的方法主要通过大量的数值模拟计算，从中选择比较好的参数搭配。

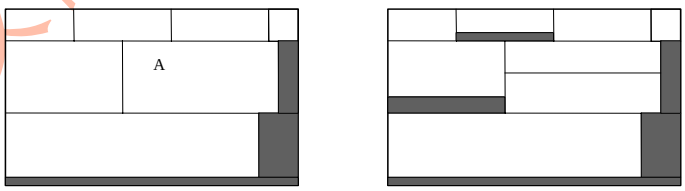
3.6 应用案例——下料问题

下料问题(cutting stock)存在诸如玻璃、钢板、木材、纸张和制衣等的裁剪问题中。它们共同的特点是原料的尺寸大于需求的尺寸，而需求的品种尺寸可以不相同，最终的目标是在满足需求的前提下，使得边角废料最小。在以上所提出的问题中，它们都有二维参数。原料有其长和宽度，需求的品种同样有长和宽的要求。本节讨论的问题产生于新闻界的锌版下料问题。一个印刷厂负责若干家报刊的印刷工作，它要将锌版按报刊版面的不同要求下料。每一块锌版原料的长和宽已知，所有锌版原料具有同一尺寸。报刊版面则有不同的长宽。

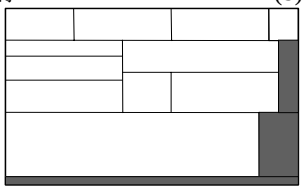
我们首先介绍常见的几种下料方法。在不混淆的前提下，我们称锌版原料为原料。

（1）直线切割(guillotine cutting)

这种切割保持切割线平行于原料的一个边线，对截开的料继续采用这个规则，一直到满足需求为止。参考图 3.6.1 的三类直线切割法。



(a)2 级切割 (b)3 级切割



(c)一般切割

图 3.6.1 直线切割分类

三种分类的主要区别：2 级切割先将原料按一个方向切割后，再对切割后的条材与原切割方向垂直切割；3 级切割是在 2 级切割后的料上再垂直于第二次切割方向切割；多于 3 级切割的直线切割为一般切割。2 级切割、3 级切割和一般切割的区别见图 3.6.1(a)(b)(c)中 A 区域所示。图中黑色部分表示剩余废边角料。

更一般的套裁方式如图 3.6.2

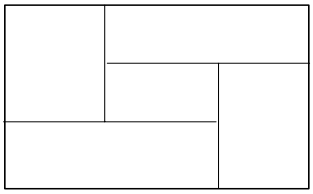


图 3.6.2 套裁方式

直观可知，套裁节约用料。平面上的套裁问题是二维装箱问题，是一维装箱问题的推广，因此，这个优化问题是 NP 难。文[13]中采用模拟退火算法求解下料套裁问题，文中把下料的位置用两个坐标表示，如原料的左下角和右上角的坐标分别为 $(x,y),(x+a,y+b)$，其中 a 和 b 分别表示原料的长和宽。第一个矩形（原材料）为 $R1=[(x,y),(x+a,y+b)]$，所有的下料点都从矩形的左下角开始。

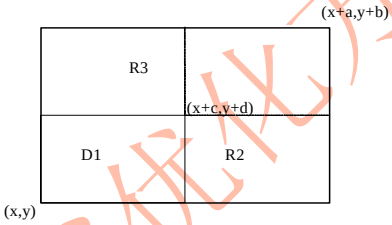


图 3.6.3 下料运算图

若所下 D1 的长宽尺寸分别为 c 和 d，则下料后的剩余矩形为，如图 3.6.3,

$$\begin{aligned}&[(x,y),(x+a,y+b)]-[(x,y),(x+c,y+d)]\\&=\{[(x+c,y),(x+a,y+b)],[(x,y+d),(x+a,y+b)]\},\end{aligned}\tag{3.6.1}$$

按逆时针记录矩形，则有

$$R2=[(x+c,y),(x+a,y+b)],$$
$$R3=[(x,y+d),(x+a,y+b)]。$$

R2 和 R3 有相交部分。任意两个矩形[(x1,y1),(x2,y2)]和[(x3,y3),(x4,y4)]，相交部分的计算公式为

$$\begin{aligned}&Com([(x1,y1),(x2,y2)],[(x3,y3),(x4,y4)])\\&=[(\max\{x1,x3\},\max\{y1,y3\}),(\min\{x2,x4\},\min\{y2,y4\})]。\end{aligned}\tag{3.6.2}$$

图 3.6.3 中 R2 和 R3 相交部分为：

$$Com(R2,R3)=[(x+c,y+d),(x+a,y+b)]。$$

若在 R2 或 R3 中一块中去掉这块公共部分，则将 R2 和 R3 分解成两个互不相交（边界除外）的矩形，这时可以再按相同的方法选择一块矩形从左下角下料。

$$R3-com(R2,R3)=[(x,y+d),(x+c,y+b)]，\tag{3.6.3}$$

$$R2-com(R2,R3)=[(x+c,y),(x+a,y+d)]。\tag{3.6.4}$$

第一次下料后，出现两块矩形。若再进行一次下料，必须从中选定一块。当选中一块后，将未选用的块用(3.6.3)或(3.6.4)减去公共部分，如图 3.6.3 中 R3 的$[(x,y+d),(x+a,y+b)]$块未被选中，则相交部分不再归属这个矩形，用（3.6.3）计算即可。因为下一次还以矩形为原料继续下料，因此，重复(3.6.1)-(3.6.4)可以得到余料的坐标。

在第 K 次切割完成后，一共剩余 K+1 个矩形，其中，K－1 个除边界外互不相交，有两块有公共相交部分。第 K+1 次切割从 K+1 个矩形中选出哪一个稍后讨论。选出的矩形有两种可能性。第一是从不相交的 K-1 个矩形中选出，此时，相交的那两块将重合部分归面积大的那个矩形，因此又分为两个不重合的矩形。第二是从相交的一对矩形中选择一个，原有重合部分归属被选择上的那个矩形，因此有重合部分的料又分为两个不重合的矩形。在任何一个矩形下一次料，由图 3.6.3 知，必然余下两块有重合部分的矩形。

采用上述的方法，每次下料前，都有若干个独立的矩形和一对带有部分重合的相交矩形。以上的规则可以重复运用。为了以后的方便，我们将需求料称为产品，余下的每一个矩形称为矩形原料。

如果将产品的下料顺序看成问题的一个解，现在必须给出如何从诸多矩形中选择一个，及从何处下料。当这些问题解决后，就可以计算余料的大小，即目标值。优化问题就是确定一个最优的下料序。用模拟退火求解时，其邻域的结构可以模仿 TSP 中的 2-opt 来构造，交换两个产品的加工顺序。于是问题解的形式和邻域结构已经得到。

选择矩形原料下料位置和目标函数计算的思想类似于例 2.3.9（2.3.3 节）的替代目标函数的启发式构造。在给定的一个下料序后，用一种启发式方法安排下料，在产品安排完后，合计余料的值，计为目标值。

可以选择的矩形原料必须满足产品的需求，但可能会出现多块原料满足产品要求的情况，从中选择的启发式规则如下。

（1）给定比较小的正数ε，当产品的长或宽同原料的长或宽不超过ε，选择满足这样条件的任一块矩形下料。有这样的可能，原料宽同产品长的差不超过ε，如图 3.6.4(a)；其二是原料长同产品长的差不超过ε，如图 3.6.4(b)；图 3.6.4 中的灰色部分表示产品的下料位置和大小。

(a) (b)

图 3.6.4 产品与原料吻合

（2）如果不存在（1）的原料，比较原料的长宽之和同产品长宽之和的差值的大小，从中选差值最小的原料。

在多次下料后，有些原料因为尺寸太小而失去使用价值，此时再记录这样小的矩形原料只会增加内存而无任何好处。因此，应该删除这样的矩形原料。对给定的数ε>0，在已下 K 块产品后，若余下的 K+1 块矩形原料中有长或宽不超过ε的，则删除这些原料块。

目标函数采用余料和同总用料的比率。由此可以估计目标值的上界是 100%，即所有的料被浪费，下界是 0%，表示无浪费。于是，可以估计出 (3.5.1)上下界的差不超过$\Delta=1$。起始温度用(3.5.1)计算，$t_0=K\Delta$，其中 K=100。

温度的下降采用 Lundy 和 Mees 的(3.5.13)

$$t_{k+1} = t_k (1+\beta t_k)^{-1},$$

其中，

$$\beta = \frac{t_0 - t_f}{M t_0 t_f},$$

t_f 为一给定的值，M 是温度可下降的最大次数。在文[13]中，作者采用非时齐算法，即每一温度只迭代一步。因此，总的迭代次数为 M。

计算的数据采用随机产生和香港地区一印刷厂的实际数据两类。模拟退火的参数分别为：起始温度 $t_0 = 100$ ， $t_f = 10$ ， M=10,000。除了最多迭代 M 步的停止准则以外，再加上如果边角余料之和不超过开始时原料的 5%，则停止计算。在此，我们仅采用随机产生数据的计算结果。

数据产生的参数是：原料分两种尺寸：400×200，400×400，产品料的种类从 5--35 个种类，每一类产品料对两种尺寸各生成 50 个随机数据组。产生的原则是每组数据一定有余料为零的最优解。整个计算是在主频 33HZ 的 486 上实现。结果见表 3.6.1 和表 3.6.2：

表 3.6.1 原料 400×200 的模拟结果

产品料种类	测试数据组数	平均计算时间（单位：秒）	达到最优解的组数	平均的迭代步数
5	50	3.8	50	8
10	50	13.0	50	15
15	50	42.0	50	303
20	50	101.5	50	868
25	50	405.2	46	857
30	50	413.0	44	866
35	50	445.3	41	943

表 3.6.2 原料 400×400 的模拟结果

产品料种类	测试数据组数	平均计算时间（单位：秒）	达到最优解的组数	平均的迭代步数
5	50	2.3	50	4
10	50	9.3	50	12
15	50	44.0	50	230
20	50	221.0	47	522
25	50	389.3	45	855
30	50	432.1	41	941
35	50	442.0	42	921

从表 3.6.1 表 3.6.2 可以看出，达到最优解的组数相对较高，算法停止时的平均迭代步数没有超过所给的上限 M，但计算时间随产品料种类数增加的增加速度较快。因此，可以说算法的计算效果还是相当好的。

图 3.6.5 表示一个 20 个产品种类在 400×400 的原料的最优下料图，余料率为 0%。

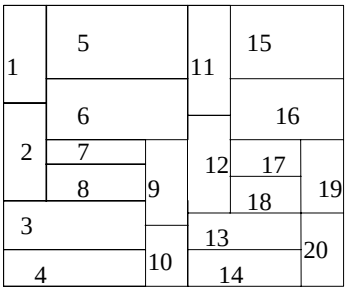


图 3.6.5 400×400 料 20 个产品的最优下料图

图 3.6.6 描述用模拟退火计算图 3.6.5 实例时，迭代步数同余料率之间的关系，其中迭代步数的每一格表示 50 个迭代步数。

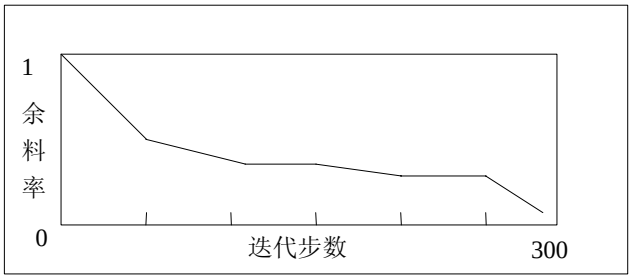


图 3.6.6 余料率同迭代步数关系

从图 3.6.6 可以看出：以平均的意义，随迭代的步数增加，余料率逐步下降，在 300 时基本达到零余料率。

练习题

1. 对非能量最低的状态，证明：存在 $t>0$，使得其波兹曼概率在$(0,t)$单调升。

2.若 $v_j = \lim_{n\to\infty} p_{ij}^{(n)} = \lim_{n\to\infty} P(X(n)=j \mid X(0)=i)(j\in D)$ 的极限有限，且状态空间有限，则证明：

$$v_j = \sum_{i\in D} v_i p_{ij}^{(n)} = \sum_{i\in D} v_i p_{ij}。$$

3.时齐马氏链的一步转移矩阵为 $P=(p_{ij}), p_{ij}=\Pr(X(k+1)=j \mid X(k)=i)$。它的 n 部转移概率为

$$p_{ij}^{(n)} = \Pr(X(n)=j \mid X(0)=i)。$$

它的矩阵形式为 $P^{(n)}=(p_{ij}^{(n)})$。证明：$P^{(n)}=P^n$。

4.若发生概率

$$G_{ij}(t)=\begin{cases}\dfrac{1}{L}, & \forall j\in N(i),\\ 0, & j\notin N(i),\end{cases}$$

其中 L 为一个正数。接受概率按(3.1.6)定义。证明：当邻域的定义使得状态空间连通时，由此定义的时齐马氏链有平稳分布。写出平稳分布的表达式。

5.若接受概率满足定理 3.3.3 的条件，发生概率按文[3]中的定义

$\exists |D|\times|D|$矩阵 Q 满足

$$\forall i, j \in D : Q_{ij} = Q_{ji};$$

$$\forall i \in D, j \in N(i) : G_{ij} = \frac{Q_{ij}}{\sum_{l \in D} Q_{il}},$$

证明：它的平稳分布存在且为：

$$v_i(t) = \frac{(\sum_l Q_{il}) A_{i_0 i}(t)}{\sum_{j \in D} (\sum_l Q_{jl}) A_{i_0 j}(t)}, \forall i \in D。$$

6. 若发生概率按练习 4 定义，文[4]中给出接受概率为

$$A_{ij}(t) = (1 + \exp(-\frac{f(j) - f(i)}{t}))^{-1}。$$

证明：它的平稳分布存在且为

$$v_i(t) = \frac{\exp(-\frac{f(i) - f_{OPT}}{t})}{\sum_{j \in D} \exp(-\frac{f(j) - f_{OPT}}{t})}, \forall i \in D。$$

7.验证

$$v_i(t_k) = \frac{\exp(-\frac{f(i) - f_{OPT}}{t_k})}{\sum_{j \in D} \exp(-\frac{f(j) - f_{OPT}}{t_k})}, \forall i \in D$$

满足(3.4.3)。

8.若 $A(t)$和 $G(t)$满足定理 3.3.6 的条件，且 $A(t)$和 $G(t)$分别满足(3.1.5)和(3.1.6)，则当 $i = i_{OPT}$ 时，平稳分布 $v_i(t)$ 为 t 的单调减函数。

9.在时齐马氏链平稳分布存在的条件下，若 $A(t)$是(3.1.6)的形式，证明：

$$\frac{d}{d \log t} \langle f(t) \rangle = \frac{\langle f^2(t) \rangle - \langle f(t) \rangle^2}{t}。$$

10. 你对 3.6 节下料问题的模拟退火算法有什么样的评价？按照你的观点实现模拟退火算法并同 3.6 节的算法比较。

11. 比较禁忌搜索、模拟退火算法在下料问题的应用效果。

12. 用模拟退火求解 TSP，同禁忌搜索算法比较它们的计算效率。

13.对你感兴趣的优化问题，尝试模拟退火的计算效果。

参考文献

1. Metropolis N, Rosenbluth A, Rosenbluth M et al. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 1953, 21:1087~1092
2. Kirkpatrick S, Gelatt Jr C D, Vecchi M P. Optimization by simulated annealing. Science, 1983, 220:671~680
3. Aarts E H L, van Laarhoven P J M. Simulated Annealing: Theory and Application. Dordrecht: D Reidel Publishing Company, 1987
4. Ackley D H, Hinton G E, Seinowski T J. A learning algorithm for Boltzmann machines. Cognitive Science, 1985, 9:147~169

5. Seneta E. Non-negative Matrices and Markov Chains. 2nd Edition. New York: Springer Verlag, 1981
6. Isaacson D, Madsen R. Markov Chains. New York: Wiley, 1976
7. Geman S, Geman D. 1984, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In: IEEE Proc. Pattern Analysis and Machine Intelligence, PAMI-6, 1984. 721~741
8. Anily S, Federgruen. Probabilistic Analysis of Simulated Annealing Methods. Graduate School of Business, Columbia University, New York, Preprint, 1985
9. Hajek B. Cooling Schedules for Optimal Annealing, Working Paper.(MOR86), 1986
10. 盛骤. 概率论与数理统计. 第二版. 北京：高等教育出版社, 1990
11. Lundy M, Mees A. Convergence of an annealing algorithm. Mathematical Programming, 1986, 34:111~124
12. Huang M D, Romeo F, Sangiovanni-Vincentelli A L. An efficient general cooling schedule for simulated annealing. In: Proceeding of IEEE Int. Conference on Computer-Aided Design. Santa Clara, 1986. 381~384
13. Lai K K, Chan J W M. Developing a simulated annealing algorithm for the cutting stock problem, Computers Ind Engng 32(1), 1997. 115-127

第四章 遗传算法

Holland 教授在 20 世纪 70 年代初期首先提出遗传算法(genetic algorithms)这个概念并使起发展起来。1975 年，Holland 出版了第一本比较系统论述遗传算法的专著《Adaptation in Natural and Artificial Systems[1]》。本章 4.1 节首先介绍遗传算法的基本框架及一些基本性质，继而在 4.2 和 4.3 节给出模板定理和马尔可夫收敛性结论，然后在 4.4 节讨论遗传算法实现的技术问题，4.5 节给出遗传算法同模拟退火算法的结合，最后通过在约束批量模型的应用了解算法实现过程。

4.1 遗传算法

“适者生存”揭示了大自然生物进化过程中的一个规律：最适合自然环境的群体往往产生了更大的后代群体。遗传算法主要借用生物进化中“适者生存”的规律，因此有必要在介绍遗传算法之前，先简单了解生物进化的基本过程，生物进化的基本过程可见图 4.1.1。

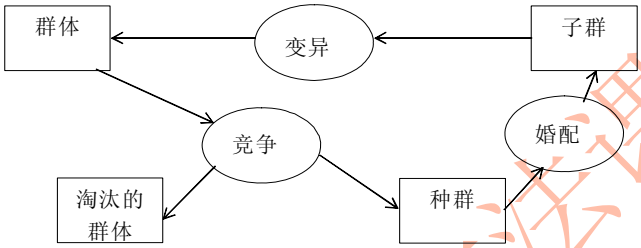


图 4.1.1 生物进化循环图

以这个循环圈的群体(population)为起点，经过竞争后，一部分群体被淘汰而无法再进入这个循环圈，而另一部分则成为种群(reproduction)。优胜劣汰在这个过程中起着非常重要的作用，这在自然界显得更加突出，因为自然环境的恶劣和天敌的侵害，使得大自然中的很多动物的成活率非常低，即使在成活群体中，还要通过竞争产生种群。种群通过婚配的作用产生子代群体（简称子群）。进化的过程中，可能会因为变异而产生新的个体。综合变异(mutation)的作用，子群体成为新的群体。在新的一个循环过程中，新的群体将替代旧的群体而成为循环的开始。

以人类进化为例，人类群体在自然竞争、淘汰后，形成了繁殖后代的群体，即成人群体。群体的每一个体是一个人，每个人包含有 46 条染色体(chromosome)，组成 23 对同源染色体(参考[2]，P.8)。男女性的结合使得对应的 23 对染色体优胜劣汰再产生 23 对染色体，因而形成一个新的生命。以性别染色体为例（见图 4.1.2），母亲的性染色体中由 X(1)和 X(2)基因(gene)组成，父亲的性染色体由 X(3)和 Y 基因组成。两个染色体的结合是这两对基因竞争后的产物，竞争的结局可能是{X(1),X(3)}（女），{X(1),Y}（男），{X(2),X(3)}（女），{X(2),Y}（男），这一对新的结合物为子代的一对性染色体。每一条染色体是由特定的基因组成。在自然进化中，会出现某些基因的变异，形成一个新的个体。

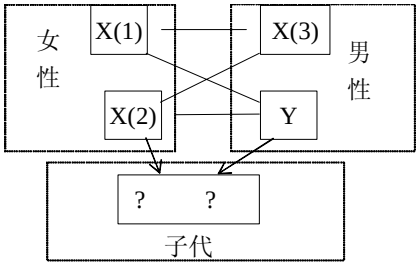


图 4.1.2 性别染色体的竞争

遗传算法主要借鉴了生物进化的一些特征，它的一些主要生物进化特征体现为：

（1）进化发生在解的编码上。这些编码按生物学的术语称为染色体。由于进行了编码，优化问题的一切性质都通过编码来研究。编码和解码是遗传算法的一个主题。

（2）自然选择规律决定哪些染色体产生超过平均数的后代。遗传算法中，通过优化问题的目标而人为地构造适应函数以达到好的染色体产生超过平均数的后代。

（3）当染色体结合时，双亲的遗传基因的结合使得子女保持父母的特征。

（4）当染色体结合后，随机的变异会造成子代与父代的不同。

遗传算法包含以下的主要处理步骤。首先是对优化问题的解编码，此处，我们称一个解的编码为一个染色体，组成编码的元素称为基因。编码的目的主要是用于优化问题解的表现形式和利于之后遗传算法中的计算。第二是适应函数的构造和应用。适应函数基本上依据优化问题的目标函数而定。适应函数确定以后，自然选择规律是以适应函数值的大小决定的概率分布来确定哪些染色体适应生存，哪些被淘汰。生存下来的染色体组成种群，形成一个可以繁衍下一代的群体。第三是染色体的结合。双亲的遗传基因结合是通过编码之间的交配(crossover)达到下一代的产生。新一代的产生是一个生殖过程，它产生了一个新解。最后是变异。新解产生过程中可能发生基因变异，变异使某些解的编码发生变化，使解有更大的遍历性。表 4.1.1 列出了生物遗传基本概念在遗传算法中作用的对应关系。

表 4.1.1 生物遗传概念在遗传算法的对应关系

生物遗传概念	遗传算法中的作用
适者生存	在算法停止时，最优目标值的解有最大的可能性被留住
个体(individual)	解
染色体(chromosome)	解的编码（字符串，向量等）
基因(gene)	解中每一分量的特征（如各分量的值）
适应性(fitness)	适应函数值
群体(population)	选定的一组解（其中解的个数为群体的规模）
种群(reproduction)	根据适应函数值选取的一组解
交配(crossover)	通过交配原则产生一组新解的过程
变异(mutation)	编码的某一个分量发生变化的过程

最优化问题的求解过程是从众多的解中选出最优的解。生物进化的适者生存规律使得最具有生存能力的染色体以最大的可能性生存。这样的共同点使得遗传算法能在优化问题中应用。我们以一个简单的例子来理解表 4.1.1。

例 4.1.1 用遗传算法求解 $f(x)=x^2, 0\le x\le 31, x$ 为整数的最大值。

一个简单的表示解的编码是二进制编码，即 0,1 字符串。由于变量的最大值是 31，因此可以采用 5 位数的二进制码。如

10000→16　　11111→31　　01001→9　　00010→2,

以上的六位字符串称为染色体，每一个分量称为基因，每个基因有两种状态 0 或 1。模拟生

物进化，首先要产生一个群体，可以随机取四个染色体组成一个群体，如 x_1=(00000), x_2=(11001), x_3=(01111), x_4=(01000)。群体有 4 个个体。适应函数可以依据目标函数而定，如适应函数 fitness(x)= $f(x)=x^2$。于是

fitness(x_1)=0, fitness(x_2)=25, fitness(x_3)=15, fitness(x_4)=8。

定义第 i 个个体入选种群的概率为

$$p(x_i)=\frac{fitness(x_i)}{\sum_j fitness(x_j)}。$$

于是，适应函数值大的染色体个体其生存概率自然较大。若群体中选四个个体成为种群，则极有可能竞争上的是 x_2=（11001），x_2=（11001），x_3=（01111），x_4=(01000)。再若它们结合，采用如下的交配方式，称为简单交配

$$\begin{array}{l} x_2=(11|001) \\ x_3=(01|111) \end{array} \longrightarrow \begin{array}{l} y_1=(11|111) \\ y_2=(01|001) \end{array}$$

$$\begin{array}{l} x_2=(110|01) \\ x_4=(010|00) \end{array} \longrightarrow \begin{array}{l} y_3=(110|00) \\ y_4=(010|01) \end{array}$$

一组交换第二个位置以后的基因，另一组交换第三个位置以后的基因，得到 y_1，y_2，y_3 和 y_4。若 y_4 的第一个基因发生变异，则变成 y_4=(11001)。□

通过例 4.1.1，我们可以将求解组合优化问题的遗传算法简化地描述为

遗传算法

STEP1 选择问题的一个编码；给出一个有 N 个染色体的初始群体 POP(1)，t:=1；

STEP2 对群体 POP(t)中的每一个染色体 $pop_i(t)$ 计算它的适应函数

$$f_i = fitness(pop_i(t))；$$

STEP3 若停止规则满足，则算法停止；否则，计算概率

$$p_i=\frac{f_i}{\sum_{j=1}^{N} f_j}, i=1,2,\cdots,N, \qquad (4.1.1)$$

并以概率分布(4.1.1)从 POP(t)中随机选一些染色体构成一个种群

$$NewPOP(t+1)=\{pop_j(t) \mid j=1,2,\cdots M\}；$$

［注］ $NewPOP(t+1)$ 集合中可能重复选 POP(t)中的一个元素，如例 4.1.1 中的 x_2 就选取两次。

STEP4 通过交配，得到一个有 N 个染色体的 $CrossPOP(t+1)$；

STEP5 以一个较小的概率 p，使得一个染色体的一个基因发生变异，形成 $MutPOP(t+1)$；t:=t+1，一个新的群体 $POP(t)=MutPOP(t)$；返回 STEP2。

种群的选取方式(4.1.1)称为轮盘赌。以下再通过例 4.1.2 理解遗传算法。

例 4.1.2 用遗传算法求 max $f(x)=1-x^2, x\in[0,1]$。

对连续变量求解，要解决的一个问题是编码。假设解的误差要求是 1/16，则可以采用四位二进制编码。对应关系为

$$(abcd)\leftrightarrow\frac{a}{2}+\frac{b}{4}+\frac{c}{8}+\frac{d}{16}。$$

若计算的一步如表 4.1.2

表 4.1.2 遗传算法中的一步计算

x 值	旧群体 POP(1)	适应函数值 f	概率分布 p	NewPOP	交配位	CrossPOP	是否变异	MutPOP	x 值
1/16	0001	0.996	0.318	0001	00|01	0000	N	0000	0
1/4	0100	0.938	0.299	0100	01|00	0101	Y	1101	13/16
3/16	0011	0.965	0.308	0001	000|1	0001	N	0001	1/16
7/8	1110	0.234	0.075	0011	001|1	0011	N	0011	3/16
	平均值=	0.7833,	交配概率 p	=1,		变异概率	=0.02		

在表 4.1.2 中，NewPOP 只选择四个染色体，交配方法采用随机选择一对染色个体按交配位进行交叉，如 0001 和 0100 的交配位随机选在位置 2，经过交配产生 0000 和 0101 两个个体。在算法中，0001 和 0011 的交配位随机选在位置 3，经过交配产生 0001 和 0011 两个个体。这样的交配称为简单交配。交配后的染色体组成 CrossPOP。在 CrossPOP 中可能有些染色体会产生变异。简单的变异是 CrossPOP 中每一个基因都以同样一个概率变异。如 0101 第一位的基因发生变异。变异后，得到 MutPOP，也就是新的群体。□

上面的例子是简单遗传算法计算的一步。简单遗传算法可以理解为：求解的问题是极大目标函数的优化问题；采用 0-1 二进制编码；*POP*(*t*)中的染色体个数是一个常数并为偶数；初始群体随机选取；适应函数为目标函数；按轮盘赌选取染色体个数同 *POP*(*t*)相同的种群；交配按例 4.1.2 的常规交配方法——对染色体按随机位交换后的基因；染色体中的每一个基因都以相同的概率变异。

以上只对遗传算法有一个直观的了解，一些技术问题将在以后的各节中讨论。归结起来有如下主要因素有待研究。

第一，解的编码和解码。遗传算法的基础工作之一是解的编码，只有在编码之后才可能有其他的计算。例 4.1.1 和例 4.1.2 采用二进制编码，后续章节还会介绍其他一些其他的基于求解问题的编码方式。编码和解码是相对应的。算法的最后一个工作是通过解码得到问题的一个解。

第二，初始群体的选取和计算中群体的大小。一般采用随机产生初始群体或通过其他方法先构造一个初始群体。通过其他方法构造的初始群体可能会节省进化的代数，但也可能过早地陷入局部最优群体中。我们称过早地陷入局部最优群体中的现象为早熟(premature)现象。群体中个体的个数称为群体的维数。群体的维数越大，其代表性越广泛，最终进化到最优解的可能性越大。群体的维数越大，造成计算时间越长，这不是我们希望的。群体的维数常常采用一个不变的常数，在一些应用中，群体的维数可以采用同遗传代数有关的变量，以使算法更有效。

第三，适应函数的确定。一般情况，适应函数同目标函数相关，以保证较优的解有较大的生存机会，也可以采用替代函数，这将在后续部分讨论。

第四，三个算子。遗传算法的三个算子是：种群选取、交配和变异或称突变。新群体产生中的一个主要问题是如何选取种群。上面的遗传算法中已经介绍，种群是以一个概率分布——轮盘赌的形式选择个体而产生的。种群选定后需考虑它的交配规则，交配规则较多，这在后续部分将详细讨论，如双亲遗传法，主个体规则，单亲遗传规则等。还需考虑交配位的选取、交配概率及产生后代等一些细节问题。交配使得算法能够搜索更多的解，但受基因的限制，只能搜索已有的所有基因组合。变异是扩大染色体选择范围的一个手段，可以得到一些新的基因。通常，遗传算法实现变异的方法是赋予每一个基因一个相对比较小的变异概率。过小的变异概率使得解有一定的局限性，遍历性较差。较大的变异概率使得进化的随机性增大，也不容易得到稳定的解。因此，不可忽略变异概率如何确定这一过程。

遗传算法的优越性可以简单地归结为以下三条。

第一，遗传算法适合数值求解那些带有多参数、多变量、多目标和在多区域但连通性较差的 NP 难优化问题。对多参数、多变量的 NP 难优化问题，通过解析求解或是计算求最优解的可能性很小，主要依赖于数值求解。遗传算法是一种数值求解的方法，具有普适性，对目标函数的性质几乎没有要求，甚至都不一定要显式地写出目标函数，因此用遗传算法求解优化问题不足为奇。遗传算法所具有的特点是记录一个群体，它可以记录多个解而不同于局部搜索、禁忌搜索和模拟退火仅仅是一个解，这多个解的进化过程正好适合于多目标优化问题的求解。

第二，遗传算法在求解很多组合优化问题时，不需要有很强的技巧和对问题有非常深入的了解。如排序(scheduling)、路线调度(routing)问题、布局(layout)问题等。遗传算法在给这些问题的决策变量编码后，其计算过程是比较简单的，且可以较快得到一个满意解。

第三，遗传算法同求解问题的其他启发式算法有较好的兼容性。如可以用其他的算法求初始解；在每一群体，可以用其他的方法求解下一代新群体。

遗传算法也不可避免的有它的不足。首先存在编码不规范及表示不准确等问题。如例 4.1.2 中，是否有更好的编码方式表示连续变量就值得研究。对 TSP 问题，解可以用城市的一个序列表示，如果采用这种编码形式，例 4.1.2 的交配规则又无法采用。

例 4.1.3 假设 8 个城市的 TSP 问题，两个染色体为(14562387)和(56327841)，这两个交配后，得到

$$\begin{array}{l}(145623|87)\\(563278|41)\end{array} \longrightarrow \begin{array}{l}(14562341)\\(56327887)\end{array}$$

交配后的两个后代已不是 TSP 的一个可行解。□

于是，交配规则同问题的编码紧密相连。

其次，单一的遗传算法编码不能全面地将优化问题的约束表示出来。这个缺点类似于模拟退火算法 3.5.1 中有关解形式的讨论。考虑约束的一个方法就是对不可行解采用罚值的方法，毫无疑问，计算的时间必然增加。

最后，是否能保证收敛到最优解？这是一个理论问题。我们将在下一节讨论这个问题。

本章在没有特殊说明时所讨论的内容都是基于组合优化问题

$$\begin{aligned} z &= \max f(x) \\ s.t. & g(x) \geq 0, , \\ & x \in D, \end{aligned}$$

其中 D 是有限离散定义域。

4.2 模板理论

模板理论(schema theorem)是基于 4.1 节的简单遗传算法，主要从一种结构的角度介绍遗传算法收敛性。这种结构，在此称为模板(schema)，在遗传算法的进化过程中，有较大的概率遗传[3]。另一种类似模拟退火算法的基于马氏链的理论研究将在 4.3 节介绍[4]。简单遗传算法的主要特征有：群体和种群的维数相等，为一个偶数维，且不随代数的变化而变化；适应函数直接选用目标函数；种群中的个体通过轮盘赌(4.1.1)的方式选取；种群中的一对个体采用随机交配位的方式产生一对子代；每一个基因有相同的变异概率。

为了易于理解，我们将染色体称为向量。一个给定的向量结构，包括关心的分量所在位置和值，称为该向量的一个模板。如对下面的两个染色体，1010001 和 1111010，它们有共同的基因和结构 1*1*0**，其中，标记*的位置表示我们不关心这个位置的取值。称 H=1*1*0**为一个模板。具有相同模板的两个染色体预示着具有一些共同的性质。对于 n

维的向量，如果采用二进制编码，共有 3^n 个模板。从遗传学的角度，生物体的代代遗传，使得某些基因和模板得到生存，生存的一个原因是这些基因和模板有很强的适应能力。

若向量的各分量取值 0、1 或*，一个模板中 0 或 1 所占的位置称为模板位置，*所占位置称为非模板位置。模板的长度(length)定义为：从第一个模板位置到最后一个模板位置的所有分量个数减 1。如 H=1*1*0**的长度为 4，其含义是从模板位置 1 到最大的模板位置 5 中间有 4 个位置的空隙。模板长度记为 $\delta(H)$。模板的阶数(order)定义为：模板位置对应的分量个数。如 1*1*0**的阶数为 3。记成 $o(H)$。若染色体 Y 在 H 的模板位置上对应的分量相同，则称 Y 具有 H 模板。记 $G(t)$是第 t 代群体，即第 t 代染色体集合，

$$T(H,G(t))=\{Y\in G(t)\,|\,\text{Y 具有模板 H}\}, \tag{4.2.1}$$

其中 t 称为遗传的第 t 代。$T(H,G(t))$为模板 H 所包含的 G(t)中的所有染色体集合。

在下面的结论中，假设群体的规模不变，即遗传的过程中，每一代的群体中染色体数相同。由于遗传算法的三个算子分别为：种群选取、交配和变异，下面三个引理分别针对这三种情况独立考虑模板的变化。

引理 4.2.1 在生殖过程中，若一个体被选入种群 NewPOP(t+1)的概率为(4.1.1)，且种群的规模同群体相同，则模板 H 所包含的染色体在 t+1 时刻的期望数为

$$E_1(H,t+1)=f(H,t)N(H,t), \tag{4.2.2}$$

其中 $N(H,t)$为 t 时刻 $T(H,POP(t))$所包含的染色体数，

$$f(H,t)=\frac{\sum\limits_{Y\in T(H,POP(t))}\dfrac{fitness(Y)}{|T(H,POP(t))|}}{\sum\limits_{Y\in POP(t)}\dfrac{fitness(Y)}{|POP(t)|}}。 \tag{4.2.3}$$

证明：由(4.1.1)，个体 $pop_i(t)$ 被选中的概率为：

$$p_i=\frac{f_i}{\sum\limits_{j=1}^{N}f_j},i=1,2,\cdots,N,$$

H 模板的每一个染色体被选中的平均概率为

$$\frac{1}{|T(H,POP(t))|}\sum_{Y\in T(H,POP(t))}\frac{fitness(Y)}{\sum\limits_{X\in POP(t)}fitness(X)}。$$

而种群 NewPOP(t+1)的个体数还是$|POP(t)|$，故含 H 模板的染色体数为

$$\frac{|POP(t)|}{|T(H,POP(t))|}\sum_{Y\in T(H,POP(t))}\frac{fitness(Y)}{\sum\limits_{X\in POP(t)}fitness(X)},$$

所以，(4.2.2)成立。□

引理 4.2.2 若在 t 时刻，模板 H 的长度为 $\delta(H)$，采用简单交配方法，即随机选一个交配位，交换位后基因，交配的概率是 p_c，则在 t+1 时刻模板 H 保留下来的概率为

$$p_1(H,t+1)\ge 1-\frac{p_c\delta(H)}{n-1}(1-p(H,t)), \tag{4.2.4}$$

其中，p(H,t)表示 t 时刻模板 H 出现的概率。

证明：模板 H 的变化是由交配产生的。由于交配位后的基因交换，因此选择第一个模板位到最后一个模板位中的任何一个位置为交配位都可能产生模板的变化，所以，模板发生改变的最大概率是 $p_c\dfrac{\delta(H)}{n-1}$。假设交配的双方有相同的模板，则交配后模板不变。一方不具有同 H 相同模板的概率为 1－p(H,t)。使模板 H 改变的最大概率为

$$p_c \frac{\delta(H)}{n-1}(1-p(H,t)),$$

在交配的过程中，会出现交配的双方不具有 H 模板，但交配后却具有 H 模板，故有

$$p_1(H,t+1) \geq 1-p_c \frac{\delta(H)}{n-1}(1-p(H,t)) \text{。} \square$$

经过简单的交配，引理 4.2.2 指出模板中的模板位置和元素保持不变的最小概率。同样下面的引理 4.2.3 也是专门讨论模板的变化。

引理 4.2.3 假设模板 H 在 t 时刻存在的概率为 p(H,t)，经过简单变异，则有

$$p_2(H,t+1) \geq 1-p_m o(H), \tag{4.2.5}$$

其中，p_m 为每个基因变异概率，$o(H)$ 为 H 的阶。

证明：变异没有改变 H 模板的概率为

$$(1-p_m)^{o(H)} \geq 1-p_m o(H) \text{。} \square$$

通过上面三个引理，从数学理论得到种群选取、交配和变异后，一个模板 H 的变化概率。由此，可以综合下面定理。

定理 4.2.1（模板定理----Schema theorem) 假设群体在 t 时刻时含有模板 H 的染色体个数为 N(H,t)，经过满足引理 4.2.1 的种群选取、满足引理 4.2.2 的以概率 p_c 的交配和满足引理 4.2.3 以概率 p_m 的变异，则在 t+1 时刻，群体中具有 H 模板的染色体数的期望值为

$$E(H,t+1) \geq \{1-\frac{p_c\delta(H)}{n-1}(1-p(H,t))-p_m o(H)\} f(H,t) N(H,t) \text{。} \tag{4.2.6}$$

如果

$$f(H,t) \geq \{1-\frac{p_c\delta(H)}{n-1}(1-p(H,t))-p_m o(H)\}^{-1}, \tag{4.2.7}$$

则从概率意义来说，每代中具有 H 模板的染色体个数将随代数 t 的增加而增加。

证明：遗传算法一代的遗传由种群选取、交配和变异三个算子组成。三个事件的综合效果，使新的一代群体中具有模板 H 的染色体数的期望值由引理 4.2.1、引理 4.2.2 和引理 4.2.3 得到

$$\begin{aligned} E(H,t+1) &= E_1(H,t+1)p_1(H,t+1)p_2(H,t+1) \\ &\geq \{1-\frac{p_c\delta(H)}{n-1}(1-p(H,t))\}\{1-p_m o(H)\} f(H,t)N(H,t) \\ &\geq \{1-\frac{p_c\delta(H)}{n-1}(1-p(H,t))-p_m o(H)\} f(H,t)N(H,t). \quad \square \end{aligned}$$

(4.2.7)的右端是一个不小于 1 的数。由此可以推断，要使具有 H 模板的染色体个数随代数 t 增加，必须要求(4.2.3)的 $f(H,t)$ 不小于 1。直观的解释是：若要具有 Y 模板的染色体个数代代增加，必须使得它们的平均适应性不小于 1。要使(4.2.6)的右端尽量的大，可以控制的参数有：使 $\delta(H)$ 尽量小，使 $o(Y)$ 尽量小。在平均适应性相同的前提下，新的一代群体中，具有低阶、长度短模板的染色体期望数不低于那些具有高阶、长度长模板的染色体期望数。低阶、长度短且平均适应性不低于 1 的模板一般称之为“积木块”(building block)。

例 4.2.1 用遗传算法求解 $z=\max\limits_{\substack{0\leq x\leq 31\\ x\text{为整数}}} x^2$。观察模板 H1=1****，H2=*10**和 H3=1***0 的变化情况。

表 4.2.1 遗传算法第一步数值结果

No.	群体	x	适应函数 f(x)	概率分布	$\frac{f(i)}{\sum_{i=1}^{4}\frac{f(i)}{4}}$	随机模拟选取出现次数
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
合计			1170	1.00	4.00	4.0
平均			293	0.25	1.00	1.0
最大			576	0.49	1.97	2.0

模板特性（生殖前）

模板	群体	具有同模板染色体	$\sum_{i\in T(H,t)}\frac{f(i)}{\lvert T(H,t)\rvert}$	$\frac{\sum_{i\in T(H,t)} f(i)/\lvert T(H,t)\rvert}{\sum_{i\in POP(t)} f(i)/\lvert POP(t)\rvert}$
H1	1****	2,4	469	1.6
H2	*10**	2,3	320	1.09
H3	1***0	2	576	1.97

按表 4.2.1 轮盘赌模拟选取种群，在种群中（见表 4.2.1）11000 出现两次，01101 和 10011 各出现一次。由引理 4.2.1 的(4.2.2)，种群中具有 H1 模板的染色体期望值为

$$E_1(H1,t+1)=f(H1,t)N(H1,t)=\frac{469}{293}\times 2\approx 3.20\text{。}$$

同样的方法可计算 $E_1(H2,t+1)=2.18$，$E_1(H3,t+1)=1.97$。种群中具有 H1，H2 和 H3 同模板的染色体实际出现数（见表 4.2.2(b)）分别为 3，2 和 2。在选定种群后，采取两个染色体随机匹配，随机选一个位置进行位置后的基因交换，交配的概率为 1，变异的概率为 0，所得结果如图 4.2.2(a)。观察新群体中模板 H1，H2 和 H3 的情况，结果见表 4.2.2(c)。它们的理论期望数由(4.2.6)给出，计算 H1 的期望数，因为 $\delta(H1)$ =0，所以

$$E(H1,t+1)\geq f(H1,t)N(H1,t)=3.20\text{。}$$

计算 H2 的期望数时，$\delta(H2)$ =1, p(H2,t)=2/4，所以

$$E(H2,t+1)\geq\{1-\frac{1}{4}(1-0.5)\}f(H2,t)N(H2,t)=1.90\text{。}$$

计算 H3 的期望数时，$\delta(H3)$ =4，p(H3,t)=1/4，所以

$$E(H3,t+1)\geq\{1-\frac{4}{4}(1-0.25)\}f(H3,t)N(H3,t)=0.49\text{。}$$

表 4.2.2(a) 遗传算法第二步数值结果

种群体	匹配（随机选）	交配位（随机选）	新群体	x 值	f(x)
0110|1	2	4	01100	12	144
1100|0	1	4	11001	25	625
11|000	4	2	11011	27	729
10|011	3	2	10000	16	256
合计					1754
平均					439
最大					729

表 4.2.2(b) 模板特性（种群）

模板	理论期望数	实际数	同模板染色体
H1	3.20	3	2,3,4
H2	2.18	2	2,3
H3	1.97	2	2,3

表 4.2.2(c) 模板特性（生殖后）

理论期望数	实际数	同模板染色体
3.20	3	2,3,4
1.90	2	2,3
0.49	1	4

其中，匹配是从种群中选择一个匹配的对象，如表 4.2.2(a)种群中个体 1 选择个体 2 为匹配对象，则表示他们两个成为交配的父母。同模板染色体是具有相同模板的染色体的缩写。上面的数值模拟结果反映了定理 4.2.1 的结论。长度短的模板适应能力较强，而长度长的适应能力较低。□

在一个有 m 个染色体，每个染色体的编码长度为 n 的群体中，一代遗传可生存的模板有多少？可以这样粗略的估计：可生存的模板一定有较大的出现概率，给以一个固定的概率值 p_s，由于交配使得长度大的模板不易生存，再由引理 4.2.2，

$$1-\frac{p_c\delta(H)}{n-1}(1-p(H,t))\geq 1-\frac{p_c\delta(H)}{n-1}\geq p_s,$$

则要求

$$\delta_s=\delta(H)\leq\frac{(n-1)(1-p_s)}{p_c}, \tag{4.2.8}$$

由此考虑长度较短的模板，考虑染色体中长度不大于 δ_s 的模板。这样长度的模板数量的一个下限为 $2^{\delta_s}(n-\delta_s)$，上式的估计可以用下面的例子这样解释：一个九个基因的染色体*********，如果仅考虑长度不大于 3 的模板，如$$$******，$表示模板中的一个基因，*表示不关心的基因，将三个$$$一体逐位后移至少有 $2^3(9-3)$ 个不同长度不超过 3 的模板。加之在群体中有 m 个染色体，故长度小于 δ_s 的模板出现数的下界为

$$m2^{\delta_s}(n-\delta_s)。 \tag{4.2.9}$$

若群体维数 $m=2^{\frac{\delta_s}{2}}$，则长度小于 δ_s 的模板出现数为 $O(m^3)$。这是 Goldberg[3]估计的一个结果。在给予一个合适的群体维数后，它隐含着遗传算法的一次进化中平行处理 $O(m^3)$ 种不同的模板。这一性质一般被称为遗传算法的隐式并行性(implicit parallelism)

删除的内容: 只
删除的内容: 小
删除的内容: 小
删除的内容: 则
删除的内容: *
删除的内容: 则

4.3 马尔可夫链收敛分析

本节所讨论的内容基于简单遗传算法。第三章 3.2 节已经介绍马氏链的一些基本性质。为了进一步分析遗传算法的收敛性，再引入下面的定义和性质。

定义 4.3.1 一个方阵 $A\in R^{n\times n}$ 称为：

(1)非负 $(A\geq 0)$，若 $a_{ij}\geq 0,\forall i,j\in\{1,2,\cdots,n\}$ 成立；

(2)全正 $(A>0)$，若 $a_{ij}>0,\forall i,j\in\{1,2,\cdots,n\}$ 成立；

(3)本原，如果 A 非负且存在一个整数 $k\geq 1$ 使得 $A^k>0$；

(4)可约，若 A 非负且经过相同的行和列初等变换得到如下形式

$$\begin{pmatrix} C & 0 \\ R & T \end{pmatrix}$$

其中，C 和 T 皆为方阵；

(5)不可约，如果 A 非负且不可约；

(6)随机，若 A 非负且 $\sum_{j=1}^{n} a_{ij} = 1, \forall i \in \{1,2,\cdots,n\}$；

(7)稳定，若 A 是一个随机阵且所有行相同，即每一列中的元素全部相同；

(8)列容，若 A 是一个随机阵且每一列中至少有一个正数。

下面的引理是遗传算法收敛性分析的基础。

引理 4.3.1 若 C、M 和 S 是随机阵，其中 M>0 和 S 列容，则 CMS 为随机阵且 CMS>0。

证明：记 A=CM 和 B=AS。因为 C 随机，则 C 的每一行中至少有一个正元素。由此

$$a_{ij} = \sum_{k=1}^{n} c_{ik} m_{kj} > 0, \forall i, j \in \{1,2,\cdots,n\},$$

即得 A>0。类似因为 S 是列容，得到

$$b_{ij} = \sum_{k=1}^{n} a_{ik} s_{kj} > 0, \forall i, j \in \{1,2,\cdots,n\}。$$

CMS 为随机阵的证明留给读者。□

引理 4.3.2 若 P 是本原随机阵，则 P^k 收敛到一个全正稳定随机阵

$$P^{\infty} = \lim_{k\to\infty} P^k = (1,1,\cdots,1)^T (p_1, p_2, \cdots, p_n),$$

其中 $(p_1, p_2, \cdots, p_n)^T$ 唯一满足，

$$(p_1, p_2, \cdots, p_n)P = (p_1, p_2, \cdots, p_n), \ \sum_{i=1}^{n} p_i = 1 \text{ 和 } p_j = \lim_{k\to\infty} p_{ij}^{(k)} > 0。$$

即 $(p_1, p_2, \cdots, p_n)^T$ 是矩阵 P^T 的特征值为 1 且每一个分量为正数的特征向量，且满足 $\sum_{i=1}^{n} p_i = 1$。

证明：我们借助定理 3.2.5 来证明这个结论。感兴趣的读者可利用本章练习题 1,2,3 证明。因为 P 为本原随机阵，将 P 看成时齐马氏链的一步转移矩阵，故有 $\forall i, j \in \{1,2,\cdots,n\}$，存在 k 满足 $p_{ij}^{(k)} > 0$。由定理 3.2.6 知马氏链是不可约的。再由 $P^k > 0$ 得到 $P^l > 0, l \geq k$，和周期性的定义，马氏链中的任何一个状态都是非周期的。由定理 3.2.5 得到以下结论：存在平稳分布 $(p_1, p_2, \cdots, p_n)^T$ 满足

$$(p_1, p_2, \cdots, p_n)P = (p_1, p_2, \cdots, p_n), \ \sum_{i=1}^{n} p_i = 1 \text{ 和 } p_j = \lim_{k\to\infty} p_{ij}^{(k)} > 0。$$

再由 $\left(p_{ij}^{(k)}\right) = P^k$（练习题 6）推出，

$$P^{\infty} = \lim_{k\to\infty} P^k = (1,1,\cdots,1)^T (p_1, p_2, \cdots, p_n)。□$$

引理 4.3.3 若 P=$\begin{pmatrix} C & 0 \\ R & T \end{pmatrix}$ 的随机阵，其中可约阵中 C 为一个 $m \times m$ 全正随机阵，R 每一行中至少有一个非零元素，则

$$P^{\infty} = \lim_{k\to\infty} P^k = \lim_{k\to\infty} \begin{pmatrix} C^k & 0 \\ \sum_{i=0}^{k-1} T^i R C^{k-i} & T^k \end{pmatrix} = \begin{pmatrix} C^{\infty} & 0 \\ R_{\infty} & 0 \end{pmatrix},$$

是一个稳定的随机矩阵，满足

$$P^{\infty} = \lim_{k\to\infty} P^k = (1,1,\cdots,1)^T (p_1, p_2, \cdots, p_n),$$

$$\sum_{i=1}^{n} p_i = 1 \text{ 和 } p_j = \lim_{k\to\infty} p_{ij}^{(k)} \geq 0,$$

其中， $p_j > 0(1 \le j \le m), p_j = 0(m+1 \le j \le n)$， R_∞ 表示 $\sum_{i=0}^{k-1} T^i R C^{k-i}$ 的极限。

证明：首先证明 P 有一个特征值 1，而其余特征值的模都小于 1。分三种情况分别讨论。

情况 1： $P = (p_{ij})$ 的所有特征值的模都不超过 1。假设不然，存在一个特征值 $\|\lambda\| > 1$。考虑特征多项式 $A = \lambda I - P = (a_{ij})$，有

$$\begin{aligned} \|a_{ii}\| - \left(\sum_{j=1, j\neq i}^{n} \|a_{ij}\| \right) &= \|\lambda - p_{ii}\| - \sum_{j=1, j\neq i}^{n} \|p_{ij}\| \\ &\ge \|\lambda\| - \|p_{ii}\| - (1 - p_{ii}) \\ &= \|\lambda\| - 1. \end{aligned}$$

因此，可知 A 可逆（练习 1），与λ是特征值矛盾。

情况 2: $\|\lambda\| = 1$ 中只有 $\lambda = 1$ 是一个特征值。设 P 的一个特征值为 $\lambda = e^{i\theta}$。当 $\theta \neq 2k\pi$，k 为整数时，

$$\begin{aligned} \|a_{ii}\| - \left(\sum_{j=1, j\neq i}^{n} \|a_{ij}\| \right) &= \|\cos\theta + i\sin\theta - p_{ii}\| - \sum_{j=1, j\neq i}^{n} \|p_{ij}\| \\ &= \sqrt{p_{ii}^2 - 2p_{ii}\cos\theta + 1} - (1 - p_{ii}) \\ &= \sqrt{(1-p_{ii})^2 + 2p_{ii}(1-\cos\theta)} - (1 - p_{ii}) \\ &> 0. \end{aligned}$$

同第一种情况，当 $\theta \neq 2k\pi$ 时 $\lambda = e^{i\theta}$ 不是 P 的特征值。明显，特征值 $\lambda = 1$ 的一个特征向量为 $(1,1,\cdots,1)^T$。

情况 3：由 $P^k = \begin{pmatrix} C^k & 0 \\ \sum_{i=0}^{k-1} T^i R C^{k-i} & T^k \end{pmatrix}$，因为 R 每一行中至少有一个非零元素，推出 T 的每一行和严格小于 1。考虑特征多项式 $\lambda I - T$，当 $\|\lambda\| \ge 1$ 时，$\lambda I - T$ 是主对角占优，$\lambda I - T$ 可逆（练习 1）。由此得到 $\|\lambda\| < 1$。得到 $T^k \to 0 (k \to \infty)$（练习 3）。

综合上面三种情况，由 $|\lambda I - P| = |\lambda I - C||\lambda I - T|$，得到 P 的一个特征值为 1 且为 C 的特征值，其余特征值的模小于 1。由代数学的基本结论，P 的约当标准型为

$$Q^{-1} P Q = diag(D_0, D_1, \cdots, D_s),$$

其中， D_0 为对应特征值 1 的约当标准型， $D_1, \cdots, D_s$ 为对应其他模小于 1 的约当标准型。

证明 $\lambda = 1$ 的代数重数是 1，即 $D_0 = 1$。假设 $\lambda = 1$ 的代数重数大于 1，则至少存在一个与 $(1,1,\cdots,1)^T$ 线性无关的特征向量 $x = (x_1, x_2, \cdots, x_n)^T$。假设 x 中的前 m 个分量 $\{x_1, x_2, \cdots, x_m\}$ 中最小的元素为 $x_i (1 \le i \le m)$。由

$$\begin{pmatrix} C & 0 \\ R & T \end{pmatrix} (x_1, x_2, \cdots, x_n)^T = (x_1, x_2, \cdots, x_n)^T$$

得到

$$c_{i1}(x_1 - x_i) + c_{i2}(x_2 - x_i) + \cdots + c_{im}(x_m - x_i) = 0;$$

再加上 C 为全正随机阵，得到 $x_1 = x_2 = \cdots = x_m$。

由于同一个特征值的特征向量形成线性子空间，选 $x = (x_1, x_2, \cdots, x_n)^T$ 的第一个分量 $x_1 = 1$，因而 $x_1 = x_2 = \cdots = x_m = 1$。再由

$$\begin{pmatrix} C & 0 \\ R & T \end{pmatrix}\left((x_1,x_2,\cdots,x_n)^T-(1,1,\cdots,1)^T\right)=\begin{pmatrix} 0 \\ T\left((x_{m+1}\cdots,x_n)^T-(1,1,\cdots,1)^T\right) \end{pmatrix}$$
$$=\begin{pmatrix} 0 \\ (x_{m+1}\cdots,x_n)^T-(1,1,\cdots,1)^T \end{pmatrix},$$

得到

$$T\left((x_{m+1}\cdots,x_n)^T-(1,1,\cdots,1)^T\right)=(x_{m+1}\cdots,x_n)^T-(1,1,\cdots,1)^T,$$

所以，T 有特征值 1 的特征向量，与情况 3 的结论矛盾。因此，$\lambda=1$的代数重数是 1。再由

$$P^k=\begin{pmatrix} C^k & 0 \\ \sum_{i=0}^{k-1}T^iRC^{k-i} & T^k \end{pmatrix}=Qdiag(D_0^k,D_1^k,\cdots,D_s^k)Q^{-1},$$

得到

$$P^k=Qdiag(D_0^k,D_1^k,\cdots,D_s^k)Q^{-1}\to Qdiag(1,0,\cdots,0)Q^{-1},k\to\infty,$$

$P^k(k\to\infty)$ 极限存在。记

$$P^\infty=\lim_{k\to\infty}P^k=\lim_{k\to\infty}\begin{pmatrix} C^k & 0 \\ \sum_{i=0}^{k-1}T^iRC^{k-i} & T^k \end{pmatrix}=\begin{pmatrix} C^\infty & 0 \\ R_\infty & 0 \end{pmatrix}。$$

由

$$P^\infty=\begin{pmatrix} C^\infty & 0 \\ R_\infty & 0 \end{pmatrix}=Qdiag(1,0,\cdots,0)Q^{-1},$$

$$rank(Qdiag(1,0,0,\cdots,0)Q^{-1})=rank\begin{pmatrix} C^\infty & 0 \\ R_\infty & 0 \end{pmatrix}=1。$$

再由引理 4.3.2，存在唯一的$(p_1,p_2,\cdots,p_m)$满足$C^\infty=(1,1,\cdots,1)^T(p_1,p_2,\cdots,p_m)$，$\sum_{i=1}^m p_i=1$和$p_j>0(1\le j\le m)$。

综合得到，存在唯一$(p_1,p_2,\cdots,p_n)$满足，$\sum_{i=1}^n p_i=1$，$p_j=\lim_{k\to\infty}p_{ij}^{(k)}\ge 0$和$p_j>0(1\le j\le m),p_j=0(m+1\le j\le n)$，使得

$$P^\infty=\lim_{k\to\infty}P^k=(1,1,\cdots,1)^T(p_1,p_2,\cdots,p_n)。\square$$

现在研究简单遗传算法的马氏链表示。记所有个体形成的空间为Ω，个体用 X 表示，$X\in\Omega$。简单遗传算法是将一个维数（规模）为 N 的群体看成一个状态。再记简单遗传算法的马氏链状态空间为 G，简单遗传算法的状态空间维数是 $M=|G|=|\Omega|^N$。

例 4.3.1(续例 4.1.2) 由于群体选用 4 个染色体，则表 4.1.2 中{(0001),(0100),(0011),(1110)}为一个状态，{(0000),(0101),(0001),(0011)}也为一个状态。染色体（个体）空间的维数是2^4，所以简单遗传算法的状态空间维数是 M=$|\Omega|^N=(2^4)^4$。□

无论如何，只要染色体空间的维数有限，简单遗传算法的状态空间维数也是有限并记为 M。遗传算法由三个算子种群选取、交配和变异组成。由于遗传是循环往返的，为了证明容易，我们按交配、变异和种群选取顺序考虑。很直观交配、变异和种群选取使得一个群体（状态）变化到另一个群体（状态），这也就是一个有限状态马氏链。下面就按这三个算子分别讨论。

交配

记交配概率矩阵 $C=\left(c_{ij}\right)_{M\times M}$，其中 c_{ij} 为状态 i 经过交配变为状态 j 的概率。一个状态经过交配运算变化到另一个状态，由全概公式

$$\sum_{j=1}^{M} c_{ij}=1。$$

C 为一个随机矩阵。

变异

每个基因有相同的变异概率 $p_m>0$，一个群体状态 i 经过变异到达 j 的概率记为 m_{ij}，对应的概率矩阵为 $M=\left(m_{ij}\right)_{M\times M}$。比较两个群体状态 i 和 j 中个体顺序相同和位置相同的基因，记有相同基因的位置总数记为 H_{ij}。如两个状态

$$s_i=\{(1000),(0101),(1100)\}, s_j=\{(1010),(1110),(0101)\},$$

则 H_{ij}=6。由于采用染色体的每个基因有相同的变异概率 $p_m>0$，状态 i 变到状态 j 的概率 $m_{ij}=p_m^{Nn-H_{ij}}(1-p_m)^{H_{ij}}>0$，其中 n 为个体的编码长度，N 为群体的规模。因此 M 是全正矩阵。

种群选取

简单遗传算法经过交配和变异后得到一个群体（状态），通过种群选取变化到另一个状态。记种群选取的转移矩阵为 $S=\left(s_{ij}\right)_{M\times M}$。仅考虑一个状态 i 经过种群选取不变的概率 s_{ii}。假设状态 i 由 N 个染色体 $\{X_1,X_2,\cdots,X_N\}$ 组成，由种群选取的轮盘赌方法，一个染色体 X_j 被选中的概率为

$$\frac{fitness(X_j)}{\sum_{k=1}^{N} fitness(X_k)}>0,$$

状态 i 经过种群选取不变的概率

$$s_{ii}\geq\prod_{j=1}^{N}\left(\frac{fitness(X_j)}{\sum_{k=1}^{N} fitness(X_k)}\right)>0。$$

所以，S 是列容的。

定义 4.3.2 假设优化问题要求目标函数 $f(x)$达到最大。令

$$Z_t=\max\{f(X_j(t))|j=1,2,\cdots,N\},$$

其中 $Y(t)=(X_1(t),X_2(t),\cdots,X_N(t))$ 是第 t 代遗传后得到的一个种群体。一个遗传算法收敛到全局最优，当且仅当

$$\lim_{t\to\infty}\Pr(Z_t=f^*)=1,$$

其中，$f^*=\max\{f(X)|X\in\Omega\}$。

定理 4.3.1 若变异概率 $0<p_m<1$，交配概率 $0\leq p_m\leq 1$，则简单遗传算法不收敛到全局最优值。

证明：设状态 $Y=(X_1,X_2,\cdots,X_N)$ 满足 $\max\{f(X_j)|j=1,2,\cdots,N\}<f^*$，$p_Y^t$ 是遗传算法第 t 代在 Y 状态的概率，则有 $\Pr\{Z_t=f^*\}\leq 1-p_Y^t$。由交配矩阵 C、变异矩阵 M

和种群选取矩阵 S 的性质和引理 4.3.1，知 CMS 为全正矩阵。再由引理 4.3.2，$\lim_{t\to\infty} p_Y^t = p_Y^\infty > 0$，因此，$\lim_{t\to\infty}\Pr(Z_t = f^*) \le 1 - p_Y^\infty < 1$。□

定理 4.3.1 从概率的意义说明简单遗传算法不收敛到全局最优。只要对简单遗传算法做一点改动，记录前面各代遗传的最优解并存放在群体的第一位，这个染色体只起一个记录的功能而不参与遗传运算。此时，第 t 代进化时，马氏链状态的形式改进为

$$\left(X_{best}(t-1), X_1(t), \cdots, X_N(t)\right),$$

其中 $X_{best}(t-1)$ 表示遗传到 $t-1$ 代的最优染色体。改进遗传算法收敛到全局最优解，且状态空间维数是 M=$|\Omega|^{N+1}$。

由于第一位的染色体不参与遗传运算，将第一位染色体相同的改进群体归于一类。它们具有简单遗传算法相同的交配矩阵 C，变异矩阵 M 和种群选择矩阵 S。记改进简单遗传算法的交配、变异和种群选择转移阵分别为 C^+、M^+和 S^+，则按第一位染色体相同的改进群体归一类有

$$C^+ = diag(C, C, \cdots, C),\quad M^+ = diag(M, M, \cdots, M),\quad S^+ = diag(S, S, \cdots, S),$$

其中各对角阵中分别有|Ω|个 C、M 和 S。

在改进群体的状态空间中，除了交配、变异和种群选取三个算子以外，最后一个记忆最优染色体的状态变化等价于状态 $W_1 = (Z_1, X_1, \cdots, X_N)$ 到 $W_2 = (Z_2, X_1, \cdots, X_N)$ 的变化，一步转移概率为

$$\Pr(W_2 \mid W_1) = \begin{cases} 1, & Z_2 = \arg\max\{f(X_1), \cdots, f(X_N)\}\text{且} \\ & f(Z_2) > f(Z_1), \\ 1, & \max\{f(X_1), \cdots, f(X_N)\} \le f(Z_1), Z_2 = Z_1, \\ 0, & \text{其它.} \end{cases} \tag{4.3.1}$$

$\arg\max\{f(X_1), \cdots, f(X_N)\}$ 表示 $\{f(X_1), \cdots, f(X_N)\}$ 中最大函数值对应的第一个染色体。如果将所有状态按第一位染色体目标值从好到坏的顺序排列，则(4.3.1)可以表示为一步转移概率矩阵

$$U = \begin{pmatrix} U_{11} & & & \\ U_{21} & U_{22} & & \\ \vdots & \vdots & \ddots & \\ U_{|W|,1} & U_{|W|,2} & \cdots & U_{|W|,|W|} \end{pmatrix},$$

其中 U_{ij} 是一个 $|\Omega|^N \times |\Omega|^N$ 矩阵，表示按第一位目标值排序第 i 位和第 j 位的所有状态由(4.3.1)定义的一步转移概率。U_{11} 的每一列至少有一个 1。

定理 4.3.2 如果改进简单遗传算法按交配、变异、种群选取之后更新当前最优染色体（解）的进化循环过程，则收敛于全局最优。

证明：如果将第一位的染色体按目标值从好到坏的顺序排列，这个进化过程马氏链的一步转移概率为

$$P^+ = C^+M^+S^+U = \begin{pmatrix} CMS & & & \\ & CMS & & \\ & & \ddots & \\ & & & CMS \end{pmatrix} \begin{pmatrix} U_{11} & & & \\ U_{21} & U_{22} & & \\ \vdots & \vdots & \ddots & \\ U_{|W|,1} & U_{|W|,2} & \cdots & U_{|W|,|W|} \end{pmatrix}$$

$$= \begin{pmatrix} CMSU_{11} & & & \\ CMSU_{21} & CMSU_{22} & & \\ \vdots & \vdots & \ddots & \\ CMSU_{|W|,1} & CMSU_{|W|,2} & \cdots & CMSU_{|W|,|W|} \end{pmatrix}.$$

注：本证明中假设只有一个最优解。当最优解不唯一时，可以同样用U_{11}表示第一个染色体目标值为最优的所有状态按(4.3.1)定义的矩阵，同样可以得到以下相同的结论。

带格式的：缩进：首行缩进： 0 磅

由 *CMS* 全正和U_{11}每一列至少有一个 1，推出$CMSU_{11}$为全正矩阵。记

$$R = \begin{pmatrix} CMSU_{21} \\ CMSU_{31} \\ \vdots \\ CMSU_{|\Omega|,1} \end{pmatrix},\ T = \begin{pmatrix} CMSU_{22} & & \\ & \ddots & \\ CMSU_{|\Omega|,2} & & CMSU_{|\Omega|,|\Omega|} \end{pmatrix}。$$

由矩阵 P^+ 第一位的染色体按目标值从好到坏的顺序排列，由(4.3.1)得到所有$U_{ij}, i \geq j$为非零矩阵。再由 *CMS* 全正，得到 R 中的每一行至少有一个非零元。利用引理 4.3.3 得知，不包含最优染色体的状态在马氏链的极限分布中的概率为 0，且有包含最优染色体的状态极限分布和为 1，定理得证。□

定理 4.3.3 如果改进简单遗传算法按交配、变异后就更新当前最优染色体，之后再进行种群选取的进化循环过程，则收敛于全局最优。

证明留给读者。

本节采用马氏链理论对改进简单遗传算法进行了分析。文[5]对编码是十进制、多种变异形式用马氏链理论分析了它们的收敛性，得到类似结果。

4.4 实现的技术问题

遗传算法同禁忌搜索和模拟退火算法一样，主要问题之一是算法如何实现等技术问题。实现中各种技术问题是这些算法中的热门题目。下面所讨论的一些技术方法有些是借助于直观，有些则有一定的理论。无论如何，在下面的讨论中，我们不依赖于理论地给出技术细节。

4.4.1 编码

编码是遗传算法中的基础工作之一。0，1 是一种比较直观和常规二进制码，我们称这一类码为常规码，与人类的染色体成对结构类似。这种编码方法使得算法的三个算子构造比较简单。诸如前面介绍的简单遗传算法的交配、变异非常简单。对一些优化问题有其表示简单和直观的优越性，如 0-1 背包问题。

例 4.4.1 0-1 背包问题为

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} a_i x_i \le b,$$

$$x_i \in \{0,1\}.$$

问题的解是一个 0-1 向量，1 表示装包，0 表示不装。于是，可以按 $(x_1, x_2, \cdots, x_n)$ 的取值形成一个自然编码。这样的编码很直观且易于遗传算法的应用。□

采用 0-1 码可以精确地表示整数，如例 4.1.1 的编码。如要用 0-1 编码精确表示 a 到 b 所有整数，只需编码长度 n 满足 $\frac{b-a}{2^n} < 1$，即 $n > \log_2(b-a)$。

连续变量也可以采用二进制编码，但需要考虑精度。对给定的区间[a,b]，采用二进制编码长为 n，任何一个变量

$$x = a + a_1 \frac{b-a}{2} + a_2 \frac{b-a}{2^2} + \cdots + a_n \frac{b-a}{2^n}, \tag{4.4.1}$$

对应一个二进制码 $a_1 a_2 \cdots a_n$。二进制码与实际变量的最大误差为 $\frac{b-a}{2^n}$。

常规码在表示有些组合优化问题时会显得无效或不方便。如 TSP 问题，用的是图表示法。选择一个弧，对应的分量为 1，否则为 0。这样一个 n 城市 TSP 的解可用一个 $n \times (n\text{-}1)$ 的 0-1 向量表示。虽然用常规码简单，但是，当每两个节点有弧相连，二进制编码的所有解的个数为 $2^{n(n-1)}$。当确定一个城市为始终点时，我们可用城市间顺序的排列来表示可行解，可行解的个数有$(n\text{-}1)!$，如果是对称 TSP，可行解的个数有 $\frac{(n-1)!}{2}$。采用这种 0,1 编码，在计算的过程中只有极小的比例 $\frac{(n-1)!}{2^{n(n-1)}}$ 为可行解，大量的计算浪费。针对 TSP，一个自然的表示方法是 n 个城市的排列。例 4.1.3 告知常规的交配对这种编码失效。于是，有必要讨论这种非常规的编码方式及其交配方法。

除常规的 0-1 编码外，其他的非 0-1 码称为非常规编码。非常规的编码同问题联系紧密。下面将通过例 1.1.5 约束机器排序问题的编码来理解非常规码。

例 4.4.2 约束机器排序问题

$$\min T$$

$$s.t. \sum_{t=1}^{T} x_{it} = 1, i = 1,2,\cdots,n,$$

$$\sum_{i=1}^{n} d_i x_{it} \le c_t, t = 1,2,\cdots,T,$$

$$x_{it} \in \{0,1\}, i = 1,2,\cdots,n, t = 1,2,\cdots,T.$$

很容易得知，它的特殊情形($c_t = c$)是装箱问题。一个自然的编码是选用决策变量的 0-1 码，是一个 $n \times T$ 的向量。由于 T 是决策变量，因此码的上限是不确定的。简便的处理方法是给定一个充分大的数 K，使得在 K 时段内达到最优解。这样，估计 K 成为问题的关键。估计的太大造成不必要的记忆空间浪费，估计的太小根本达不到最优解。即使估计 K 比较容易解决，常规的 0-1 编码还存在致命的缺点。编码没有考虑能力的约束，对应每一时段只有 0-1 码确定后才知道能力是否满足。因为解空间中大量是不可行解，若对不满足能力约束的解采用罚值的方法，计算时间必定增加。

精确的估计是：每个解用一个 $n \times T$ 的向量表示，解空间中共有 $2^{n \times T}$ 解。若假设每个时

段最多可以加工 P 个产品，则至少有 $2^{(n-P)\times T}-1$ 个解是不可行的。对这些不可行解的计算是浪费时间。当然可以动态控制 T 的大小，但是何时增加或减少 T 同样也是一个难以解决的问题。

针对这个问题，一种有效的编码方法是给产品(1,2,...,n)一个加工序 $(i_1,i_2,\cdots,i_n)$，由加工序按能力约束以最小的剩余能力依次安排时段加工，在加工的过程中不允许改变产品的加工序。如产品需求为($d_1=5,d_2=3,d_3=10,d_4=4$)，前五个时段可提供能力为(3,9,10,5,20)，若按(1,2,3,4)顺序加工，各时段加工的产品为：第一个时段不加工，第二个时段加工产品 1 和 2，第三个时段加工产品 3，第四个时段加工产品 4。很明显，最优解为第一时段加工产品 2，第二时段加工产品 1 和 4，第三时段加工产品 3，最优序为(2,1,4,3)。□

就这两种编码法，第一种虽然直观，但没有考虑问题的特性，在解的构造中没有考虑约束，造成计算中出现不可行解，浪费计算时间。第二种编码方法考虑问题的特性及约束情况，它的表示也非常简单，在计算中无不可行解，因此，不会因为不可行解的处理而浪费时间。同样它同 TSP 的城市序编码一样，存在交配和变异规则确定问题。

编码问题的讨论中包含有这样一种观点：虽然遗传算法，包括模拟退火、禁忌搜索等，是具有通用性的全局最优算法，如果不针对问题设计算法，其计算时间可能是非常大的；可以通过对问题的了解而换取计算时间的节省。这正是目前被一些学者接受的“无免费午餐(参考[6])”观点。这一观点使得我们在应用中尽可能地根据实际问题进行编码。

4.4.2 评价遗传算法的常用方法

遗传算法的模板定理只对模板的进化趋势给予讨论。马氏链收敛分析给出概率 1 收敛的条件。无论如何，理论分析都是无穷代进化的概率结果。用一个方法监察计算中解的变化趋势，可以了解进化的程度以便决定是否继续下去。遗传算法求解最优化问题的一个目的是得到最优解。于是需要了解遗传算法求最优解的功效。为了尽可能地利用信息，常用处理方法和评价法分类讨论如下。

当前最好解(best-so-far)方法。在每一代的进化中，记录下最好的解。通过这个最好解展示算法的效果。这个最好解可以用于不同算法的横向比较。如果知道问题的下界，也可以用最好解与下界的差来衡量算法的功效。

在线(on-line)比较法。用进化过程中的每一个解来了解进化趋势，其计算公式为

$$v^{on-line}(T)=\frac{1}{T}\sum_{t=1}^{T}v(t), \tag{4.4.2}$$

其中，T 为当前计算中群体中出现的染色体总数，$v(t)$为第 t 个染色体的目标值。

与在线比较法类似，有离线(off-line)比较法，其计算公式为

$$v^{off-line}(T)=\frac{1}{T}\sum_{t=1}^{T}v^*(t), \tag{4.4.3}$$

其中，T 为当前计算中所出现的遗传代数，

$$v^*(t)=\max\{v^*(1),v^*(2),\cdots,v^*(t-1),v^*(POP(t))\},$$

$v^*(POP(t))$ 表示 $POP(t)$中的最优目标值。(4.4.2)和(4.4.3)的区别在于(4.4.2)中的 t 是记第 t 个染色体，而(4.4.3)的 t 是第 t 代。

(4.4.3)的另一种理解方法是 $v^*(t)$为第 t 代所得的最优目标值，即

$$v^*(t)=\max_{i\in POP(t)}\{v(i)\}。 \tag{4.4.4}$$

(4.4.4)又提供一种评价进化计算的方法。如果记录当前时刻的最优解，(4.4.4)可以用来比较每一代解的改进情况。

从直观可以看出，当优化问题是最大化目标函数时，随着 T 的增加(4.4.3)应该具有上升

趋势。因为适应函数取决于目标函数，随着一代代的进化使得适应能力强的染色体生存下来，也就是目标值大的状态保留下来，(4.4.3)的平均值上升。

(4.4.2)虽说不同于(4.4.3)，但总的变化趋势是相同的。可能不同的地方是在每一代，(4.4.2)可能有波动，但代与代之间还应该保持上升趋势。

通过(4.4.2)或(4.4.3)或(4.4.4)的监控，可以掌握遗传算法计算的进展情况，以决定是否改进、停止算法。

4.4.3 初始参数的选取和停止原则

群体的规模

在算法的第一步，需要确定群体的规模。依据第 4.2 节模板理论的结果，一个比较好的群体规模为 $m=2^{\frac{\delta_s}{2}}$，其中 δ_s 为满足(4.2.8)的模板长度。这个群体的规模隐含 $O(m^3)$ 个模板的并行计算。从(4.2.8)估计 δ_s 为 $O(n)$，其中 n 为个体的编码长度。于是当编码长度增加时，群体的规模指数增加。若将这一群体规模应用在大规模的实际问题中，可以推断它们群体的规模非常之大，因此带来计算时间的增加，于是遗传算法难以同其他的算法竞争。

经常采用的方法之一是将群体的规模设定为个体编码长度数的一个线性倍数。如 m 在 n 和 $2n$ 之间的一个确定数。

群体规模的选择也可以是变化的。非常直观，当(4.4.4)多个进化代没能改变解的性能，保持现有的群体规模已很难改进解，此时早熟，则可扩大群体的规模。反之，若解的改进非常好，则可以减少群体的规模以便加快计算的速度。

初始群体的选取

大多数学者接受这样的一个观点：初始群体应该随机选取。只有随机选取才能达到所有状态的遍历，因而最优解在遗传算法的进化中最终得以生存。毫无疑问，初始群体的随机选取加大了进化的代数，因而，加大了计算时间。一些学者也提出应该用其他的一些启发式算法或经验选择一些比较好的染色体（种子）作为初始群体。争论的焦点是：种子的选取有一定的偏见和缺乏代表性，因此可能产生早熟而无法求出最优解。这个问题实际上是“无免费午餐”。应用者应针对实际问题而权衡。

终止规则

一个最为简单的停止规则是给定一个最大的遗传代数 MAXGEN，算法迭代代数在达到 MAXGEN 时停止。

第二类方法是给定问题一个上界 UB 的计算方法，当进化中达到要求的偏差度ε时，算法终止，即当 UB－$v^*(t)$<ε时，停止。

删除的内容: 下

删除的内容: L

删除的内容: －LB

第三类规则有一定的自适应性。如(4.4.2)、(4.4.3)和(4.4.4)的评价算法规则，当由它们监控发现算法再进化已无法改进解的性能，此时停止计算。如用(4.4.4)监控到算法 K 代没有进化到一个更好的解，于是算法停止。

最后一类是多种停止终止规则的组合。如第一类、第二类或第三类准则的组合。

4.4.4 进化过程中的技术问题

这是遗传算法应用中讨论最多的内容。主要涉及进化过程中适应函数的确定、种群的选取、交配的及变异方法等问题。下面逐个问题讨论。

适应函数

(1)简单适应函数

简单的适应函数是目标函数的简单变形。若 $f(x)$为目标函数，则适应函数可以取

$$fitness(x) = f(x), \quad \text{优化目标为最大，} \tag{4.4.5}$$

$$fitness(x) = M - f(x), \quad M > c_{\max} \text{ 且优化目标为最小。} \tag{4.4.6}$$

简单适应函数的优点是构造简单，与目标函数直接相关。

值得注意的是：采用简单的适应函数可能使得算法在迭代过程中出现收敛到一些目标值近似的不同染色体，而简单的适应函数已难区别这些染色体。

例 4.4.3 已知优化问题为：$z = \max\limits_{x\in[0.5,1]} f(x) = \max\limits_{x\in[0.5,1]}\{1+\log x\}$。若采取(4.4.1)四位编码且适应函数为优化问题的目标函数。遗传算法进化的某一步为表 4.4.1:

表 4.4.1

x	群体	fitness(x)	概率分布 p_x
1/2	0000	0.689	0.214
5/8	0100	0.796	0.248
21/32	0101	0.817	0.254
23/32	0111	0.857	0.272

此时，从群体中以概率分布选取种群，由于它们四个的概率分布值相差很小，因此哪一个染色体占先也不是非常显著。□

(2) 非线性加速适应函数

例 4.4.4(续例 4.4.3) 从随机模拟来看，表 4.4.1 的适应值改进速度非常慢。若将适应函数用同目标函数具有相同变化趋势，但变化速度更快的

$$fitness(x) = \begin{cases} \dfrac{1}{1-x}, & 0.5 \le x < 1, \\ M > 0, & 1 \le x, \end{cases}$$

替代，其中 M 是一个充分大的数，表 4.4.1 改进为表 4.4.2

表 4.4.2

x	群体	fitness(x)	概率分布 p_x
1/2	0000	3.322	0.161
5/8	0100	4.902	0.237
21/32	0101	5.464	0.264
23/32	0111	6.993	0.338

此时，已有较明显的区别。要构造这样的替代函数需要对问题有深入的了解，知道 x=1 是问题的最优解，这不是非常容易做到的。一个有效且简便的方法是根据已有的信息构造替代函数。如

$$fitness(x) = \begin{cases} \dfrac{1}{f_{\max} - f(x)}, & f(x) < f_{\max}, \\ M > 0, & f(x) = f_{\max}, \end{cases} \tag{4.4.7}$$

其中 M 是一个充分大的数，$f_{\max}$ 是当前的最优目标值。上式将表 4.4.1 改进为表 4.4.3。

表 4.4.3

x	群体	fitness(x)	概率分布 p_x
1/2	0000	5.376	5.376/(46.769+M)
5/8	0100	16.393	16.393/(46.769+M)
21/32	0101	25.000	25/(46.769+M)

23/32	0111	M	M/(46.769+M)

(4.4.7)仅根据已有的计算信息得到。在(4.4.7)中，存在的问题是 M 的选取，M 决定当前最优解的继承性。□

(4.4.7)给出适应函数的一种求解方法。例 4.4.4 适应函数构造中，M 有很重要的作用。选取 M 的策略是：初始迭代时，M 同第一大与第二大目标差值的倒数尽量接近以避免早熟，后期迭代中逐步扩大差距。也可以在早期迭代中用简单的适应函数，而在后期用这类加速的方法。

(3) 线性加速适应函数

(4.4.7)的思想进一步系统化得到线性加速适应函数。线性加速适应函数为

$$fitness(x) = \alpha f(x) + \beta, \tag{4.4.8}$$

其中α，β按下方程确定

$$\begin{cases} \alpha \dfrac{\sum_{i=1}^{m} f(x_i)}{m} + \beta = \dfrac{\sum_{i=1}^{m} f(x_i)}{m} \\ \alpha \max\limits_{1\le i\le m}\{f(x_i)\} + \beta = M \dfrac{\sum_{i=1}^{m} f(x_i)}{m} \end{cases}, \tag{4.4.9}$$

其中所有的 $x_i(i = 1,2,\cdots,m)$ 为当前代群体中的染色体。(4.4.9)的第一个方程表示平均值变换后不变，第二个方程表示将当前最优值放大到平均值的 M 倍。选取 M 的策略是：当目标值相差较大时，M 不要过大，以便遗传的随机性；当遗传的一个群体目标值接近时，逐步扩大 M。

由(4.4.9)解得

$$\alpha = \frac{\dfrac{\sum_{i=1}^{m} f(x_i)}{m}(M-1)}{\max\limits_{1\le i\le m}\{f(x_i)\} - \dfrac{\sum_{i=1}^{m} f(x_i)}{m}},$$

$$\beta = \frac{\sum_{i=1}^{m} f(x_i)}{m}\left[\frac{\max\limits_{1\le i\le m}\{f(x_i)\} - M\dfrac{\sum_{i=1}^{m} f(x_i)}{m}}{\max\limits_{1\le i\le m}\{f(x_i)\} - \dfrac{\sum_{i=1}^{m} f(x_i)}{m}}\right]。$$

(4) 排序适应函数

将同一代群体中的 m 个染色体按目标函数值从小到大排列，直接取分布概率为

$$p(i) = \frac{2i}{m(m+1)},\ 1 \le i \le m, \tag{4.4.10}$$

这样，避开了对目标函数进行线性、非线性等加速适应函数的早熟可能，使每一代当前最优解以最大的概率 2/(m+1)遗传。

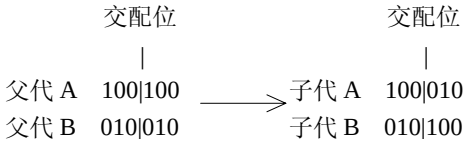
加速适应函数的思想同第三章模拟退火算法的接受概率思想相似，都希望开始时每一个状态有比较大的选取性，随着计算的进行，逐渐拉开目标值不同所对应状态的档次。这种相似之处使得遗传算法和模拟退火算法很好的结合，在 4.5 节将专门介绍这个内容。

交配规则

遗传算法中，交配规则较多，此处只介绍一些比较常用的规则。

(1) 常用方法——双亲双子

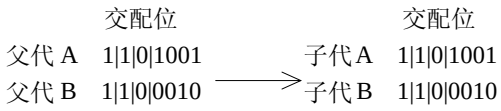
在双亲确定后，这种方法以一个随机位进行位之后的所有基因对换。对换后形成两个后代。简单的示例如下：



这是一种非常简单的方法。4.2 节的理论就是在这种交配方法下得到的。

(2) 变化交配法(string-of-change crossover)

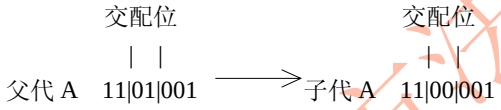
对于某些双亲，采用常规方法可能造成父代与子代完全相同，势必影响收敛速度和搜索范围。如



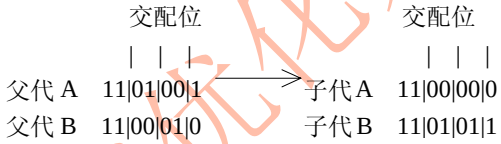
选交配位在 1,2,3 的交配无任何变化。因此，应该避开这三个位置。可以采用的方法是从头开始先比较它们的相同的基因，从不同基因位置按常规方法随机选交配位。比较上面示例得 sssdsdd，其中 s 表示相同，d 表示不同。交配位可以在第四到第七位随机选取。

（3）多交配位法(multi-point crossover)

随机选择多个交配位，双亲以一个交配位到下一个交配位基因相互替代和下一个交配位到再下一个交配位不变这样交叉形成两个新的后代。如有两个交配位

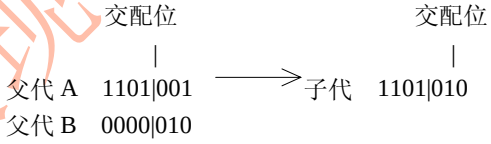


和有三个交配位



（4）双亲单子法

这一方法使得一对双亲只有一个后代。一类是从常规交配法的两个后代中随机选一个，另一类则是根据优胜劣汰从两个后代中选一个好的。如



同样，变化交配法和多交配位法也可以选择双亲单子法。

（5）显性遗传法(dominance)

对双亲中的基因，有些是具有优超关系的，这些基因必将遗传到下一代。例 4.1.1 可用来解释这种现象。例 4.1.1 中，其他位不变的情况下，任何一位的 1 永远比同位的 0 要好，于是 1 优超 0。优超法的运算如下示例

父代 A 1101001 ⟶ 子代 1101011
父代 B 0000010

Goldberg 和 Smith[7]将这种方法成功地应用在背包问题中。

以上方法都是双亲遗传法，还有一大类为单亲遗传法。

(6) 单亲遗传法

单亲遗传的一个特点是只有一个父代，下一代的产生通过单亲自身的基因变化。如选定一个单亲，随机选两个基因位置，将两个位置的元素进行交换，见下面的交换：

交换位 交换位
| | ⟶ | |
父代 1 1 0 1 0 0 1 子代 1 0 0 1 0 1 1

也可以选定两个交配位，通过交配位之内的基因倒排得到子代。如

交配位 交配位
| | ⟶ | |
父代 1 |1010| 01 子代 1 | 0101 | 01

单亲遗传法使得染色体中的基因取值受到限制，如上例中的 1 的个数为 4 个，0 的个数为 3 个，因此限制搜索的范围。在使用单亲遗传时应加大变异的概率。单亲遗传法可以同双亲遗传法结合使用。

以上讨论都局限于常规码，对例 4.1.3 和例 4.4.2 中讨论的非常规码又如何处理？非常规码对上面讨论的一些交配方式失效，如对常规双亲双子、变化交配法、多交配位、单子法和优超法等失效。于是，有必要讨论非常规码的交配问题。

（7）非常规码的交配方法

此处只讨论 n 个整数{1,2,⋯,n}排列的非常规码，这种编码形式经常应用在 TSP 和约束机器排序等问题中。一些优化问题可能有其他的非常规编码形式，它们的交配方法可根据本节介绍的思想构造。

法 1：`非常规码的常规交配法。随机选一个交配位，两个后代交配位之前的基因分别继承双亲的交配位之前基因，交配位之后的基因分别按对方基因顺序选取不重基因。如

交配位 交配位
| |
父代 A 213|4567 ⟶ 子代 A 213|4567
父代 B 431|2567 子代 B 431|2567

子代 A 是从父代 A 的交配位前取 213，然后以父代 B4312567 依顺序选不重基因 4,5,6,7。

法 2：不变位法。随机产生一个同染色体有相等维数的不变位向量，每一分量随机产生 0 或 1，其中 1 表示不变，0 表示变。变化的方式按法 1 处理。如随机产生这样的一个向量

1 0 0 1 1 0 0

则交配的变化情况为

不变位
* ** * **
父代 A 2134657 ⟶ 子代 A 2314657
父代 B 4312567 子代 B 4132567

其中，子代 A 的形成首先将父代 A 的第 1、4、5 位不变，第 2、3 位的变化按法 1 从父代 B 按顺序取与不变位不同的基因 3、1，4、5 位不变使得 6、7 位分别选 5、7。同样规则得子代 B。

在非常规码的交配中，变异也不能同常规码一样只是 0 或 1 的变化，这样的方法在非常规码中不再用。由于遗传算法是生物进化的模拟，因此，需要变异这一功能。可以采用位置交换的方法实现这一功能。其基本思想是 2-opt 法。可以用常规编码时上面第(6)条的交

配方法产生变异。

种群的选取和交配后群体的确定

种群由适应函数对应的概率分布以轮盘赌确定。在实际模拟中所关心的模板可能出现实际选取数与引理 4.2.1 的理论期望数的偏差。此时遗传算法的整体收敛性将引起偏差，于是可用监控的方法使得种群中所关心的结构在给定的范围内。

在种群的选取中，如用常用的交配方法，种群中随机选取的染色体数同群体 POP(t)的维数相等。假设群体 POP(t)维数为 $m=2k$，则以概率分布随机选取的 m 个染色体随机结成 k 对，在交配概率为 1 的前提下生成 m 个后代。

在进化的过程中，有多个因素与群体的规模相关。第一是种群 NewPOP(t+1)的选取，可以选择小于 POP(t)规模的种群，这样才能体现选择最优的染色体组成种群。第二是交配过程。交配概率不一定为 1，有些父母就不一定有子女。在采用其他的交配方式时，如双亲单子这样的方式，就无法保证有 m 个后代。于是，产生用多少个新的染色体去替代旧群体中的染色体的问题。

即使以概率 1 交配产生 m 个后代，则用新产生的 m 个后代（变异后）替代原有的 m 个父代会产生无法将最优解保持到下一代。如父代染色体为

(0000), (1010), (1001), (1101)，

其中假设(0000)为最优解，则任何两个染色体结合都无法保证最优解遗传到下一代。基于上面的原因，种群选取的染色体个数可以不同于群体的个数，其次是交配后的子代不一定要求全部替换旧群体。由此产生一个替换问题。针对这个问题，常见的方法是：

（1）种群的选取、交配和变异用常规的方法，只是在 MutPOP(t+1)中选最优的 L 个染色体替换 POP(t)中最差的 L 个染色体；

（2）选择种群中染色体的个数只是群体的一个比例，此时采用常规的交配方法，交配概率为 1，交配后的子代同 POP(t)中的染色体通过筛选组成 POP(t+1)；

（3）采用一些非常用的交配方法，用交配、变异后的子代同 POP(t)通过筛选组成 POP(t+1)。记替换率为

$$G(t+1)=1-\frac{POP(t)\cap POP(t+1)}{m}。\tag{4.4.11}$$

替换率是替换问题的一个重要参数。$G=0$ 时为零替换，即 $POP(t)=POP(t+1)$, 遗传一代后没有一个染色体变化。无任何进展的进化等于计算的重复。因此，我们期望替换率 $G>0$。100%（$G=1$）的替换使得新群体和旧群体的染色体相重率降低。直观看计算效果应该好。但有可能使得当前最优解无法遗传下去。由 4.3 节的理论，我们希望当前最优解保存下去。

低替换率产生过多地重复计算适应函数值，使得搜索的范围扩展较慢。它的优点是使得某些关心的染色体得以保留。Whitley[8]通过计算模拟说明替换率的大小与评价函数的关系，说明 100%替换率并不是最好的。

在群体或种群中有时会出现相同的染色体。相同的染色体造成适应函数的重复计算，但同时也是适应能力的一种表现，有扩大下一代相同染色体生存的可能性。对这个问题不必作太多的处理。

100%的替换会出现当前最优解的遗失，一种保持的策略是使得上一代的当前最优解强行遗传到下一代。

本节主要介绍遗传算法中应注意的问题和各种处理方法。我们没有讨论它们之间的关系和实际应用的最佳搭配。我们坚持“无免费午餐”的观点，参考[9]。因此用遗传算法求解优化问题时，应尽可能地了解问题的本身结构，针对问题给出算法设计，决不能无目的的模拟计算。

4.5 遗传模拟退火算法

第 4.1 节中提到遗传算法的兼容性，本节简单介绍模拟退火算法同遗传算法的一种结合算法。在模拟退火时齐算法中，每两个温度之间的状态点是无关的。从理论上来看，随着迭代的步数增加，任何一个温度的马氏链都渐渐不依赖起点状态，且在每一个状态的概率服从平稳分布。遗传算法强调的是两代之间的进化关系，但其交配有可能使最好解遗失。结合这两个算法的特点，构成下面的遗传模拟退火算法。

遗传模拟退火算法

STEP1 给定群体规模 MAXPOP，k:=0；初始温度 $t_k := t_0$，群体 POP(k)；

STEP2 若满足停止规则，停止计算；否则，对群体 POP(k)中每一个染色体 $i \in POP(k)$ 的邻域中随机选状态 $j \in N(i)$，按模拟退火中的接受概率

$$A_{ij}(t_k) = \min\{1, \exp(-\frac{f(j) - f(i)}{t_k})\}, \tag{4.5.1}$$

接受或拒绝 j，其中 $f(i)$为状态 i 的目标值；这一阶段共需 MAXPOP 次迭代选出新群体 NewPOP1(k+1)；

STEP3 在 NewPOP1(k+1)中计算适应函数

$$f_i(t_k) = \exp\{-\frac{f(i) - f_{\min}}{t_k}\}, \tag{4.5.2}$$

其中，$f_{\min}$ 是 NewPOP1(k+1)中的最小值；由适应函数决定的概率分布从 NewPOP1(k+1)中随机选 MAXPOP 个染色体形成种群 NewPOP2(k+1)；

STEP4 按遗传算法的常规方法进行交配得到 CrossPOP(k+1)；再变异得到 MutPOP(k+1)；

STEP5 $t_{k+1} := d(t_k)$，$k := k+1$，POP(k)=MutPOP(k)，返回 STEP2。

遗传模拟退火算法中，STEP2 的群体选择较遗传算法的选择范围要大，用 $\sum_{i=1}^{MAXPOP} N(i)$ 取代遗传算法中的 MAXPOP。但它并不是简单地随机选取，而是采用(4.5.1)的公式。这里具有模拟退火的第一个特征。适应函数的加速特性已在 4.4.4 节讨论，(4.5.2)是一个非常好的加速适应函数。当温度较高时加速性不明显，当温度较低时加速性非常明显。这正是我们需要的，也是模拟退火的第二个特征。其他计算步骤同遗传算法。

4.6 应用案例——生产批量问题

本节介绍遗传算法在生产批量问题中的应用。生产批量问题是企业生产和管理中一个常见的问题。主要考虑最优的生产批量使得生产费用、生产准备费用和库存费用综合指标最小。按传统将企业内部的生产管理按图 4.6.1 分为三个层次。第一个层次为企业的发展战略和长期规划，是一个企业为了生存和参加市场竞争对企业发展所规划的一个方向。这个层次是战略性的。第二个层次是企业中、短期计划。主要体现为根据人力、物力的能力对产品需求，包括订货、计划库存等。第三个层次是生产计划的具体实现是对已安排的生产计划具体落实到人和机器。最下一层的具体安排可能发现能力的不足，

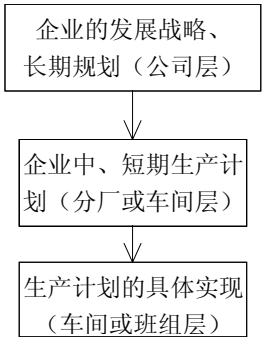


图 4.6.1

此时，可采取一些补救的方法，如人力不足可以加班加点弥补，在最下层内部解决；也可以反馈到上层对生产计划甚至发展规划进行更改。

物料需求计划(MRP-Material Requirement Planning)是在不考虑能力约束的前提下，根据产品需求按生产的工艺流程 BOM(Bill Of Materials)进行物料的计划管理。它是利用主生产计划（由第二层决定）来决定需要哪些材料及需要哪些零部件来完成生产计划。MRPII(Manufacturing Resources Planning)是在 MRP 的基础上于七十年代发展起来，它主要在考虑了市场的需求、企业的生产能力和对用户的服务水平等诸多因素的前提下来安排企业的生产。

在 MRPII 中，第二层的主要问题之一是生产计划中的约束批量问题[10]。在满足能力约束和生产平衡的条件下，能力约束批量问题是考虑生产费用、库存费用和生产准备(set-up)费用综合目标的优化问题。它的数学模型为

$$Min\ \ Cost=\sum_{i=1}^{n}\sum_{t=1}^{T}\left\{s_iY_{it}+c_ix_{it}+h_iI_{it}\right\} \tag{4.6.1}$$

$$s.t.\ \ I_{i,t-1}+x_{it}-I_{it}=d_{it}+\sum_{j\in S(i)}r_{ij}x_{jt},\quad i=1,2,\cdots,n,t=1,2,\cdots,T, \tag{4.6.2}$$

$$\sum_{i=1}^{n}(a_{kit}x_{it}+A_{kit}Y_{it})\le c_{kt},\quad k=1,2,\cdots,K;t=1,2,\cdots,T, \tag{4.6.3}$$

$$Y_{it}=\begin{cases}1, & x_{it}>0,\\ 0, & x_{it}=0,\end{cases}\quad i=1,2,\cdots,n;t=1,2,\cdots,T, \tag{4.6.4}$$

$$I_{i0}=I_{iT}=0,I_{it}\ge 0,x_{it}\ge 0,\ i=1,2,\cdots,n;t=1,2,\cdots,T, \tag{4.6.5}$$

其中

T：计划时段，如一年的 12 个月，一周 5 天等；n：产品的品种数；x_{it}：为产品 i 在时段 t 的生产批量，是决策变量；I_{it}：为产品 i 在时段 t 的库存量，是决策变量；d_{it} 为时段 t 时产品 i 的需求量；s_i 为生产产品 i 时的生产准备费用，这一费用是由于机器更换加工产品而一次性发生的；c_i 为单位产品 i 的生产费用；h_i 为单位产品 i 的库存费用；a_{kit} 为 t 时段单位产品 i 占用资源 k 的量；A_{kit} 为 t 时段产品 i 的生产准备占用资源 k 的量；c_{kt} 为 t 时段资源 k 的可提供量；r_{ij} 为 j 产品的生产过程中对 i 产品的需求量，如一个部件直接需要若干个零件，称 j 为 i 的直接后继；$S(i)$ 为所有 i 的直接后继集合。

在上面的能力约束批量模型中，生产准备费用、生产费用和库存费用只与产品有关而与时段无关，可以更为复杂地考虑与时段有关的模型。从下面的介绍中可以看出，这些费用与时段有关的模型研究的难度更大。

(4.6.1)为目标函数，它要求生产准备、库存、生产和加班费用的综合最小。方程(4.6.2)是物流的平衡方程，当时段生产的量和上时段的库存量应该满足外部和内部后继产品的需求及本时段库存的需要。(4.6.3)是能力约束方程。(4.6.4)标记产品 i 在时段 t 是否生产，1 表示生产。(4.6.5)表示生产批量不是负数和不允许有欠货。

生产中常用的生产结构有：串联型（图 4.6.2a）、组装型（图 4..6.2b）和一般型（图 4.6.2c）。

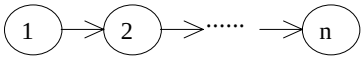


图 4.6.2a 串联型

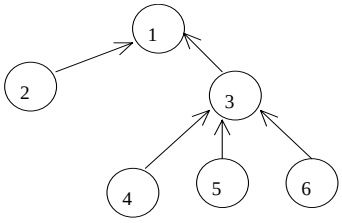


图 4.6.2b 组装型

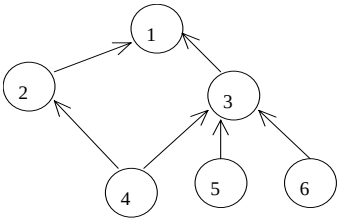


图 4.6.2c 一般型

一个产品按工序：原材料——成型——精加工，是一个典型的串联型生产。组装型生产以汽车制造、飞机制造业为典型，是由零件、部件到成品的一个生产过程。一般型生产包含前两种并比它们更为广泛。在多数企业生产中都会遇到这样的情况，比较典型的是化工企业，投入几种原材料会产生出几种产品。如炼油过程，从原油变成汽油、柴油和煤油等。

产品的生产结构主要体现在(4.6.2)的内部需求关系 r_{ij}。约束批量问题中，当约束(4.6.3)的 A_{kit} 不全为零，求解(4.6.2)到(4.6.5)的一个可行解是 NP 完全[11]。去掉约束(4.6.3)的多层无约束批量问题仍然是 NP 难[12]。于是，我们尝试用遗传算法求解。

(1) 编码

遇到的第一个问题是编码。由于约束批量问题是一个混合整数规划问题，决策变量 I_{it}, x_{it} 为连续变量，Y_{it} 为 0-1 整数变量。于是需要研究变量之间的关系和性质。首先研究的是无能力约束批量问题：去掉能力约束(4.6.3)。

定理 4.6.1 无能力约束批量问题(4.6.1)、(4.6.2)、(4.6.4)和(4.6.5)有满足下面条件的最优解

$$x_{it} I_{it-1} = 0 \text{。} \tag{4.6.6}$$

证明：按产品生产的阶段性，由(4.6.2)和(4.6.5)产品 i 的生产量为一个常数

$$\sum_{t=1}^{T} x_{it} = \sum_{t=1}^{T} (I_{i,t-1} + x_{it} - I_{it}) = \sum_{t=1}^{T} (d_{it} + \sum_{j \in S(i)} r_{ij} x_{jt})$$

$$= \sum_{t=1}^{T} d_{it} + \sum_{j \in S(i)} r_{ij} \sum_{t=1}^{T} x_{jt}, \ i = 1,2,\cdots,n.$$

如在生产的最后一个阶段，假设 i=1 为最终产品，则

$$\sum_{t=1}^{T} x_{1t} = \sum_{t=1}^{T} d_{1t} \text{；}$$

在到数第二个阶段，假设产品标号为 i=2 且为第一个产品提供中间产品，则

$$\sum_{t=1}^{T} x_{2t} = \sum_{t=1}^{T} (d_{2t} + r_{21} x_{1t}) = \sum_{t=1}^{T} d_{2t} + r_{21} \sum_{t=1}^{T} d_{1t} \text{。}$$

再由(4.6.1)知三项费用中的生产费用 $\sum_{i=1}^{n} \sum_{t=1}^{T} c_i x_{it} = \sum_{i=1}^{n} c_i \sum_{t=1}^{T} x_{it}$ 为常数。因此，只需讨论生产准备费用和库存费用。

假设无能力约束批量问题有一个最优解满足 $x_{it} I_{it-1} > 0$，表示在 t-1 时段末有库存而第 t 时段还要生产，则可以重新安排生产：在 t 时段之前新的批量 x_{it}^* 满足

$$x_{it}^* = x_{it} + I_{it-1}, \ \sum_{s=1}^{t-1} x_{is}^* = \sum_{s=1}^{t-1} x_{is} - I_{it-1} \text{。}$$

在之后保持原来批量不变。新的批量满足(4.6.2)、(4.6.4)和(4.6.5)且库存费用不增、生产准备费不增和生产费不变，总体目标值不增，但此时 $x_{it} I_{it-1} = 0$。同样方法可以证明定理成立。□

由定理 4.6.1，产品的生产满足：当 $Y_{it}=0$ 时 $x_{it}=0$；当 $Y_{it_1}=1, Y_{it_2}=1,\ 1\le t_1<t_2\le T$ 且 $Y_{it}=0, t_1<t<t_2$ 时，则

$$x_{it_1}=\sum_{t=t_1}^{t_2-1}(d_{it}+\sum_{j\in S(i)}r_{ij}x_{jt})。\tag{4.6.7}$$

上面的讨论给出 0-1 变量 Y_{it} 同生产批量的关系。根据 Y_{it} 任何一个时段要么不生产，要么按(4.6.7)生产足以满足下一个生产准备时段前的所有产品需求。对无能力约束批量问题可以用 Y_{it} 对应的 0-1 编码采用遗传算法求解。对有能力约束的批量问题如何采用上面已经得到的结果？

处理的方法之一是针对能力可以通过其他方式，如加班或租借等。此时可以去掉能力约束(4.6.3)，可以用一个第 t 时段第 k 种资源的加班量 O_{kt} 去松弛(4.6.3)的约束，但在目标(4.6.1)中体现。

$$Min\ \ Cost=\sum_{i=1}^{n}\sum_{t=1}^{T}\{s_iY_{it}+c_ix_{it}+h_iI_{it}\}+\sum_{k=1}^{K}\sum_{t=1}^{T}\rho_kO_{kt}\ ,\tag{4.6.8}$$

其中，

$$O_{kt}=\max\left\{\sum_{i=1}^{n}(a_{kit}x_{it}+A_{kit}Y_{it})-c_{kt},\ 0\right\},$$
$$k=1,2,\cdots,K;t=1,2,\cdots,T,$$

ρ_k 为增加单位资源 k 的费用。这样的处理方法有其应用背景，一些企业经常在能力不足的情况下通过租借或是加班加点扩大生产能力。由于增加 O_{kt}，能力约束已转移到目标函数中考虑，于是无能力约束批量问题的编码可以继续使用。

另一种处理方法是针对能力不可改变而设计。它还是利用无能力约束批量问题的结果，对给定的 0-1 编码 Y_{it}，按(4.6.7)进行批量生产。可能出现产生能力不足，此时用批量模型中经常使用的“移动”启发式方法[13]。移动启发式方法的基本思想是：首先按（4.6.8）确定每个时段的生产批量。第二，从最后一个时段 T 到第一个时段逆顺序检查是否满足能力约束。满足约束的条件下继续向前检查。不满足能力约束的条件下，将该时段的加工多余部分前移。当然这里面存在前移到哪一个时段的技术问题。如考虑将超出能力的生产量按最近时段有能力空闲前移。第三，为目标值的计算。若将所有不满足能力约束的时段都前移至可行，此时虽说按(4.6.4) $Y=\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$ 的生产批量同移动后的生产批量不同，但我们还是将移动后的可行解对应原有的编码 $\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$，由此按移动后的解求出目标值作为 $\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$ 的一个评价值，记为 $v(Y)$。若无法将所有不满足能力约束的时段都前移至可行，此时认为无可行解，目标值定义为一个充分大的数。

(2) 适应函数

由于有能力约束批量问题的目标是极小目标函数，适应函数与群体各目标值的关系是：给定一个群体 $\{Y^1,Y^2,\cdots,Y^{Maxpop}\}$、常数 $\varepsilon>0$ 和一个 0-1 码 $Y=\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$，适应函数为

$$fitness(Y)=\underset{1\le i\le Maxpop}{Max}\ v(Y^i)-v(Y)+\varepsilon,\tag{4.6.9}$$

其中，在可以加班模型和不可加班模型中 $v(Y)$分别对应各自的目标值 Cost(Y)和评价值。(4.6.9)中的ε越小，具有最大目标值（评价值）的状态被选取为种群的概率越小。

(3) 实现的参数选取及计算结果

由上面的讨论，每一个解编码为

$$Y^{g,j}=(Y_{11}^{g,j},Y_{12}^{g,j},\cdots,Y_{1T}^{g,j},Y_{21}^{g,j},Y_{22}^{g,j},\cdots,Y_{2T}^{g,j},\cdots,Y_{n1}^{g,j},Y_{n2}^{g,j},\cdots,Y_{nT}^{g,j}),$$
$$j=1,2,\cdots,Maxpop;g=1,2,\cdots,Maxgen,$$

。

群体的规模为 Maxpop，一共迭代 Maxgen 代。在解的编码中为避免不可行解假设

$$d_{i1}>0,Y_{11}^{g,j}=Y_{21}^{g,j}=\cdots=Y_{n1}^{g,j}=1。$$

批量由(4.6.7)确定。在能力不可以加班时，再用“移动”的方法修正批量。

根据(4.6.9)计算适应函数，用概率分布

$$fitness(Y^{g,j})=\frac{fitness(Y^{g,j})}{\sum_{i=1}^{Maxpop}fitness(Y^{g,i})},$$

选种群。种群的规模同群体的规模相同。

采用双亲双子交配法，交配概率 p_c=0.6。采用交配后代直接替代双亲，而没有交配的双亲直接遗传到下一代，也就是交配后群体规模同旧群体规模相同，覆盖率为 60%。再以 p_m=0.033 的概率变异。

对 n=7,T=6,K=1 用十组随机模拟数据比较模拟退火、禁忌搜索和遗传算法在有能力约束、无加班和组装型生产模型的应用效果，发现计算时间和结果相差不是很大。在可加班、一般型生产模型中，对一个 N=21、T=6、K=2 的实际问题进行求解。在主频为 66HZ 的 PC-486 上用近十分钟得到满意解。

练习题

1.方阵 A 主对角占优的定义为：

$$\|a_{ii}\|>\sum_{j=1,j\neq i}^{n}\|a_{ij}\|,1\le i\le n,$$

其中||x||表示 x 的模。证明：主对角阵占优的方阵可逆。

2.若 A 为随机矩阵，则 A 除有一个值为 1 的特征根，其他特征值的模均小于 1。当 A 为全正的随机矩阵时，特征值 1 的代数重数是 1。

3. 对约当矩阵 $A=\begin{pmatrix}\lambda&1&0&\cdots&0\\0&\lambda&1&\cdots&0\\0&0&\lambda&\cdots&0\\&\cdots&&\cdots&\\0&0&0&\cdots&\lambda\end{pmatrix}$，当$\|\lambda\|<1$时，证明：$\lim_{k\to\infty}A^k=0$。

4. 若 P 为全正随机矩阵，则有 P^k 收敛到一个全正稳定随机阵

$$P^{\infty}=(1,1,\cdots,1)^T(p_1,p_2,\cdots,p_n),$$

其中，

$$(p_1,p_2,\cdots,p_n)P=(p_1,p_2,\cdots,p_n)。$$

5. 若 A 为本原随机矩阵，则有 P^k 收敛到一个全正稳定随机阵

$$P^{\infty}=(1,1,\cdots,1)^T(p_1,p_2,\cdots,p_n),$$

其中，

$$(p_1,p_2,\cdots,p_n)P=(p_1,p_2,\cdots,p_n)。$$

6. 若 P 为时齐马氏链的一步转移矩阵，证明：k 步转移矩阵为$\left(p_{ij}^{(k)}\right)=P^k$。

7.证明：定理 4.3.3。

8. 若 C、M 和 S 是随机阵，其中 M>0 和 S 列容，证明 CMS 为随机阵.

9. 对感兴趣的组合优化问题，比较禁忌搜索、模拟退火和遗传算法的计算效果。

参考文献

1. Holland J H. Adaptation in Natural and Artificial Systems. MIT Press, 1975
2. 徐维衡主编. 医学遗传学基础. 北京医科大学、中国协和医科大学联合出版社, 1993
3. Goldberg D E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, 1989
4. Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Networks, 1994, 5:96~101
5. 佘春峰. 十进制遗传算法的理论分析及其应用. 清华大学硕士论文, 1998
6. Wolpert D H, Macready W G. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1997, 1:67-82
7. Goldberg D E, Smith R E. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 1987. 59~68
8. Whitley D. 1987, Using reproductive evaluation to improve genetic search and heuristic discovery. In: Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 1987. 108~115
9. Xie J, Xing W. Incorporating domain-specific knowledge into evolutionary algorithms. Beijing Mathematics, 1998, 4:131~139
10. 谢金星. 生产计划和调度：数学模型及算法. 清华大学博士论文, 1995
11. Maes J, MaClain J O, Van Wassenhove L N. Multilevel capacitated lot-sizing complexity and LP-based heuristics. European Journal of Operational Research, 1991, 53:131~148
12. 谢金星，姜启源，邢文训 et al. 能力受限的批量问题的数学模型与算法新进展. 运筹学杂志，1996, 15(1), 1~12
13. Clark R, Armentano V A. A heuristic for a resource-capacitated multi-stage lot-sizing problem with lead-time. Journal of Operational Research Society, 1995, 46:1208~1222

第五章 蚁群优化算法

蚁群优化算法(ant colony optimization algorithms)是一种分布式智能模拟算法，基本思想是模仿蚂蚁依赖信息素进行通信而显示出的社会行为。它是一种随机的通用试探法，可用于求解各种不同的组合优化问题，具有通用性和鲁棒性，是基于总体优化的方法。

蚂蚁是自然界中常见的一种生物，“蚂蚁搬家，天要下雨”之类的民谚反映出人们对蚂蚁智能性有较高的评价。在我们的记忆中，一提起蚂蚁，总可以回想起孩提时代专心致志观察成群结队的蚂蚁搬运食物的情形。随着近代仿生学的发展，人们开始对蚁群的这种相对弱小、功能并不强大的个体是如何完成复杂工作(如寻找到食物的最佳路径并返回等)进行研究，这种似乎微不足道的小东西越来越多地受到学者们的关注。1992 年Dorigo[1]在他的博士论文中首先提出了一种全新的蚁群系统(ant system)启发式算法，在此基础上一种很好的优化算法逐渐发展起来。

本章首先在 5.1 节介绍蚁群优化算法的基本概念和算法结构，在 5.2 节建立算法的数学模型及分析算法的收敛性，5.3 节给出实现的一些技术细节，5.4 介绍一个应用案例——医学诊断的数据挖掘。

5.1 蚁群优化算法的概念

蚁群优化算法是一种受自然界生物行为启发而产生的“自然”算法，产生于对蚁群行为的研究。蚁群中的蚂蚁以“信息素”(pheromone)为媒介，间接异步地相互联系。这是蚁群优化算法的最大的特点。蚂蚁在行动(寻找食物或者寻找回巢的路径)中，会在它们经过的地方留下一些化学物质，称之为“信息素”。这些物质能被同一蚁群中后来的蚂蚁感受到，并作为一种信号影响后者的行动，具体表现在后到的蚂蚁选择有这些物质的路径的可能性比选择没有这些物质的路径的可能性大得多。后到者留下的信息素会对原有的信息素进行加强，并循环下去。这样，经过蚂蚁越多的路径，后到蚂蚁选择这条路经的可能性就越大。由于在一定的时间内，越短的路径会被越多的蚂蚁访问，因而积累的信息素也就越多，在下一个时间内被其他的蚂蚁选中的可能性也就越大。这个过程会一直持续到所有的蚂蚁都走最短的那一条路径为止。

通过图 5.1.1 简单地了解蚂蚁的运动过程。假设一个蚂蚁外出寻找食物，蚂蚁从 A 点出发，行走速度相同，食品在 D 点，蚂蚁可能行走的路线如图 5.1.1。由于无法预知道路中间的情况，蚂蚁出发时会随机选择 ABD 或 ACD 中的一条。假设初始每条路线上分别分配一个蚂蚁，每单位时间行走一步。当行走 9 个单位时间后，为图 5.1.1 中上半部分的情形，已经有一个蚂蚁到达 D 点，行走的路线为 ABD，而行走 ACD 路线的蚂蚁才到达 C 点。当行走 18 个单位时间后，走 ABD 的蚂蚁已经回到 A 点，而行走 ACD 的蚂蚁到达 D 点。如果蚂蚁每经过一处都留下大小为 1 点的信息素，这时 ABD 路线的第一点聚集 2 点，而 ACD 路线的第一点聚集 1 点。在行走 36 个单位时间后，这两点的信息素变化分别为 4 和 2，比值为 2：1 ，ACD 路线的蚂蚁返回 A 点。

如果按比值的比例，蚁群决定 ABD 路线派两个蚂蚁而 ACD 路线上派一个蚂蚁。在每个蚂蚁再各行走 36 个单位时间后，ABD 和 ACD 路线的第一点各累计 12 和 4，比值为 3：1。如果再按比值分配蚂蚁数量，则 ABD 路线分配 3 个蚂蚁，而 ACD 路线分配一个蚂蚁。按原有的模式重复 36 个单位时间，ABD 和 ACD 路线的第一点信息素各累计 24 和 6，比值为 4：1。如此重复下去，可以发现 ABD 和 ACD 路线的第一点信息素的比值会越来越大，最后的极限是所有的蚂蚁只选择 ABD 路线。

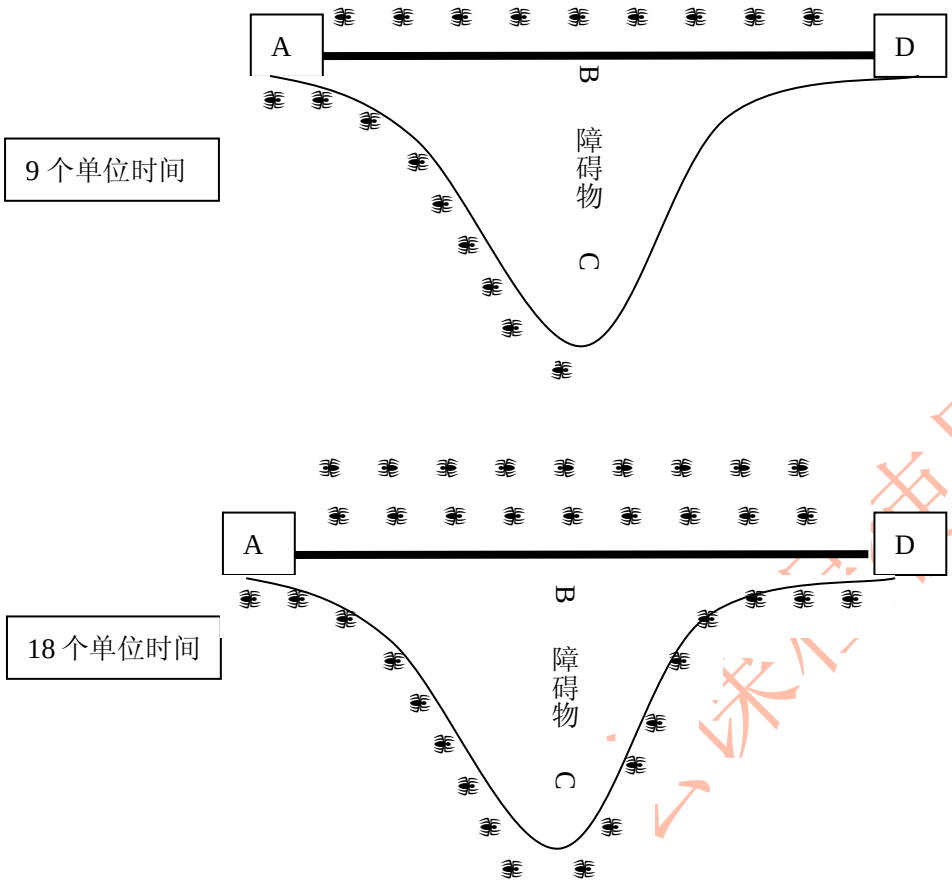


图 5.1.1 简化的蚂蚁寻物的过程

在自然界中,蚁群的这种寻找路径的过程表现为一种正反馈的过程。正基于此种行为,产生了人工蚁群的寻优算法。我们把只具备了简单功能的工作单元视为“蚂蚁”,那么上述寻找路径的过程可以用于解释人工蚁群的寻优过程。人工蚁群和自然界蚁群的相似之处在于,两者优先选择的都是含“信息素”浓度较大的路径。这在两种情况下,较短的路径上都能聚集相对较多的信息素。两者的工作单元(蚂蚁)都是通过在其所经过的路径上留下一定信息的方法进行间接的信息传递。人工蚁群和自然界蚁群的区别在于,人工蚁群有一定的记忆能力。它能够记忆已经访问过的节点。另外,人工蚁群在选择下一条路径的时候并不是完全盲目的,而是按一定的算法规律有意识地寻找最短路径(如在 T S P 问题中,可以预先知道下一个目标的距离)。

为了更好地了解蚁群优化算法,本章所有的符号和算法设计以 TSP 为基础,其他应用则需读者对问题研究后,给出算法的改进。TSP 可以简单地用 n 个城市的一个有向图 G=(N,A)表示,其中,$N=\{1,2,\cdots,n\}$,$A=\{(i,j)\mid i,j\in N\}$,和城市间的距离 $D=(d_{ij})_{n\times n}$ 表示,目标函数为 $f(W)=\sum_{l=1}^{n} d_{i_{l-1}i_l}$,其中,$W=(i_1,i_2,\cdots i_n)$ 为城市(1,2,…,n)的一个排列,$i_{n+1}=i_1$。

仿效自然界的蚁群行为,蚁群优化算法中人工蚂蚁的行为可以描述如下。m 只蚂蚁在 TSP 图中相邻的节点间移动,从而协作异步地得到问题的解。每只蚂蚁的一步转移概率由图中每条边上的两类参数决定:(1)信息素值,也称为信息素痕迹,是蚁群的“记忆”信息;(2)可见度,也就是先验值。信息素的更新由两种操作组合完成的:挥发,一种全局的减少弧上信息素值的办法,是模仿自然界蚁群的信息素随时间挥发的过程。还有就是增强,给评价值“好”的弧增加信息素值。

它们的移动是通过运用一个随机决策原则来实现,这个原则运用所在节点存储的信息,

计算出下一步可达节点的概率，通过这个概率分布实现一步移动。通过这种移动，蚁群建立的解会越来越接近最优解。当一个蚂蚁找到的一个解，或者在它的找寻的过程中，这个蚂蚁会评估这个解(或者是解的一部分)的优化程度，并且在相关连接的信息素痕迹中保存对解的评价信息。这些信息素的信息对蚂蚁未来的搜索有指导意义。

基于图的蚁群优化算法[2](GBAS: graph-based ant system)可以简单地描述如下：

STEP 0 对 n 城市的 TSP 问题，$N=\{1,2,\cdots,n\}$, $A=\{(i,j)\mid i,j\in N\}$，城市间的距离 $D=(d_{ij})_{n\times n}$，为 TSP 图中的每一条弧(i,j)赋信息素痕迹初值 $\tau_{ij}(0)=1/|A|$，假设有 m 只蚂蚁在工作，所有的蚂蚁从同一城市 i_0 出发。k:=1。当前最好解 $W=(1,2,\cdots,n)$。

STEP 1 （外循环）如果满足算法的停止规则，停止计算并输出计算得到的最好解。否则，让蚂蚁 s 从起点 i_0 出发，用 L(s)表示蚂蚁 s 行走的城市集合，初始 L(s)为空集，$1\le s\le m$。

STEP 2 （内循环）按蚂蚁 $1\le s\le m$ 的顺序分别计算。当蚂蚁在城市 i，若 $L(s)=N$ 或 $\{l\mid (i,l)\in A, l\notin L(s)\}=\varnothing$ 完成第 s 只蚂蚁的计算。否则，若 $L(s)\neq N$ 且 $T=\{l\mid (i,l)\in A, l\notin L(s)\}-\{i_0\}\neq\varnothing$，则以概率

$$p_{ij}=\begin{cases}\dfrac{\tau_{ij}(k-1)}{\sum_{l\in T}\tau_{il}(k-1)}, & j\in T,\\ 0, & j\notin T.\end{cases} \tag{5.1.1}$$

到达 j，$L(s)=L(s)\cup\{j\}$，i:=j；

若 $L(s)\neq N$ 且 $T=\{l\mid (i,l)\in A, l\notin L(s)\}-\{i_0\}=\varnothing$，到达 i_0，$L(s)=L(s)\cup\{i_0\}$，i:= i_0；重复 STEP2。

STEP 3 对 $1\le s\le m$，若 $L(s)=N$，按 L(s)中城市的顺序计算路径长度；若 $L(s)\neq N$，路径长度是一个充分大的数。比较 m 只蚂蚁中的路径长度，记走最短路径的蚂蚁为 t。若 $f(L(t))<f(W)$，则 W:=L(t)。用(5.1.2)对 W 路径上的弧信息素痕迹加强，对其他弧的信息素痕迹挥发，

$$\tau_{ij}(k)=\begin{cases}(1-\rho_{k-1})\tau_{ij}(k-1)+\dfrac{\rho_{k-1}}{|W|}, & (i,j)\text{为}W\text{的一条弧},\\ (1-\rho_{k-1})\tau_{ij}(k-1), & \text{其他}.\end{cases} \tag{5.1.2}$$

得到新的 $\tau_{ij}(k)$，k:=k+1，重复 STEP1。

在 STEP3 中，挥发因子 ρ_k 对于某固定的 K≥1，满足

$$\rho_k\le 1-\frac{\log k}{\log(k+1)}, k\ge K, \tag{5.1.3}$$

并且

$$\sum_{k=1}^{\infty}\rho_k=\infty。 \tag{5.1.4}$$

在上面描述的蚁群优化算法中，(5.1.1)为蚂蚁的搜寻过程，即以信息素决定的概率分布选择下一个访问的城市。算法还包括两个其他的过程，由(5.1.2)体现，称之为信息素痕迹的挥发(evaporation)过程和增强(reinforcement)过程。信息素痕迹的挥发过程就是每个连接上的信息素痕迹的浓度自动逐渐减弱的过程，由 $(1-\rho_k)\tau_{ij}(k-1)$ 表示。信息素痕迹的挥发过程主要用于避免算法太快的向局部最优区域集中。采用这种实用的遗忘方式有助于搜寻区域的扩展。增强过程是蚁群优化算法的一个可选部分，用于实现由单个蚂蚁无法实现的集中行动。在(5.1.2)中，增强过程体现在观察蚁群(m 只蚂蚁)中每只蚂蚁所找到的路径并

且将蚂蚁所找到最短路径上的弧上保存额外的信息素。增强过程中进行的信息素更新被称为离线的信息素过程。在 STEP3 中，除非蚁群发现了一个更好的解，否则，蚁群的永远记录第一个最好解。

很容易验证，(5.1.2)满足 $\sum_{(i,j)\in A}\tau_{ij}(k)=1,\forall k\geq 0$。

例5.1.1 四城市非对称TSP如图5.1.1

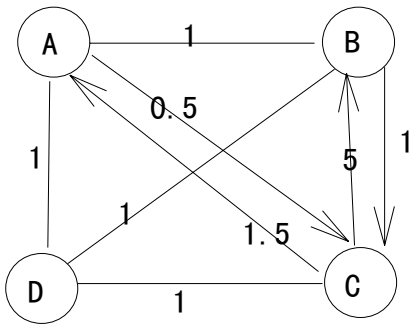


图5.1.1 四城市TSP

其中，没有标示箭头的弧为双向弧，即往返距离相同。距离矩阵为

$$D=(d_{ij})=\begin{bmatrix}0 & 1 & 0.5 & 1\\ 1 & 0 & 1 & 1\\ 1.5 & 5 & 0 & 1\\ 1 & 1 & 1 & 0\end{bmatrix}。$$

假设蚁群中有 4 个蚂蚁，所有的蚂蚁都从城市 A 出发，挥发因子 $\rho_k=1/2,k=1,2,3$，观察 GBAS 算法的计算过程。

首先有一个初始的记忆表，记录信息素痕迹。图 5.1.1 中一共有 12 条有向弧

$$\tau(0)=(\tau_{ij}(0))=\begin{bmatrix}0 & 1/12 & 1/12 & 1/12\\ 1/12 & 0 & 1/12 & 1/12\\ 1/12 & 1/12 & 0 & 1/12\\ 1/12 & 1/12 & 1/12 & 0\end{bmatrix}。$$

按(5.1.1)计算，每一个到达下一个城市的可能性均等，行使完算法 STEP2，假设蚁群的四只蚂蚁行走的路线分别为：

第一只：$W_1:A\to B\to C\to D\to A$，$f(W_1)=4$，

第二只：$W_2:A\to C\to D\to B\to A$，$f(W_2)=3.5$，

第三只：$W_3:A\to D\to C\to B\to A$，$f(W_3)=8$，

第四只：$W_4:A\to B\to D\to C\to A$，$f(W_4)=4.5$。

当前最优解（实际为全局最优解）为W_2，STEP3 的信息更新为

$$\tau(1)=(\tau_{ij}(1))=\begin{bmatrix}0 & 1/24 & 1/6 & 1/24\\ 1/6 & 0 & 1/24 & 1/24\\ 1/24 & 1/24 & 0 & 1/6\\ 1/24 & 1/6 & 1/24 & 0\end{bmatrix}。$$

结束一个循环。如果再对蚁群循环一次，因为W_2为全局最优解，由 STEP3 的信息素痕迹更新规则，无论四只蚂蚁行走的路线如何，STEP3 的信息更新为

$$\tau(2)=\left(\tau_{ij}(2)\right)=\begin{bmatrix} 0 & 1/48 & 5/24 & 1/48 \\ 5/24 & 0 & 1/48 & 1/48 \\ 1/48 & 1/48 & 0 & 5/24 \\ 1/48 & 5/24 & 1/48 & 0 \end{bmatrix}。$$

在这一步，信息素痕迹更新不依赖蚁群所行走的路线，主要原因是W_2已经为最优解，而 GBAS 算法只记录第一个最优解，因此，一旦得到一个全局最优解，信息素痕迹更新将不依赖蚁群的后面行为。再重复一次循环，得到

$$\tau(3)=\left(\tau_{ij}(3)\right)=\begin{bmatrix} 0 & 1/96 & 11/48 & 1/96 \\ 11/48 & 0 & 1/96 & 1/96 \\ 1/96 & 1/96 & 0 & 11/48 \\ 1/96 & 11/48 & 1/96 & 0 \end{bmatrix}。\square$$

归结为理论问题，可以发现，(5.1.1)使得蚂蚁以一定的概率转移，整个信息素痕迹的更新在 STEP3 的(5.1.2)完成，并随每个 k 变化。如果记第 k 次（外）循环结束后的信息素痕迹为$\tau(k)=\left(\tau_{ij}(k)\,|\,(i,j)\in A\right)$，当前最优解为$W(k)$。于是，第 k 次外循环前，信息素痕迹和当前最优解为$\{\tau(k-1),W(k-1)\}$，经过第 k 次外循环后，得到$\{\tau(k),W(k)\}$，且由于(5.1.1)的随机性，$\{\tau(k-1),W(k-1)\}$到$\{\tau(k),W(k)\}$的转移同样具有随机性。这是一个马尔可夫过程。我们将在 5.2 节中专门讨论蚁群优化算法的马尔可夫过程的收敛性。

一般的蚁群优化启发式算法同 GBAS 有相同的框架，主要由三个组成部分：蚁群的生成和活动，信息素的挥发，增强行为。主要体现在(5.1.1)和（5.1.2）的变化。算法进程结构并不限定这三个行为发生的顺序或者是否同步，也不要求它们以一种完全平行、独立的形式来运行。这样给算法设计者以充分的自由来安排这三个行为的顺序。这些将在 5.3 节的技术细节中讨论。5.4 节将介绍蚁群优化算法的应用。

5.2 算法模型和收敛性分析

蚁群优化算法的每步迭代对应随机变量

$$X_k=(\tau(k),W(k)),k=0,1,\cdots,$$

其中，$\tau(k)$为信息素痕迹，在$R^{|A|}$中取值；$W(k)$为 n 城市的一个排列，最多有$n!$个状态。注意到第 s 只蚂蚁在第 k 轮的转移只由$\tau(k-1)$决定，进一步，这个蚂蚁行走的路径和$W(k-1)$一起，共同决定了$W(k)$，再通过信息素的更新原则可以进一步得到$\tau(k)$。也就是说，X_{k+1}的变化仅由X_k决定，而与之前的状态无关。这是一个典型的马尔可夫过程，这个马尔可夫过程是非时齐的。

定义 5.2.1 一个马尔可夫过程$\{X_k,k=0,1,\cdots\}$，对任给的$\varepsilon>0$满足

$$\lim_{k\to\infty}\Pr\left\{\left|X_k-X^*\right|<\varepsilon\right\}=1,$$

则称马尔可夫过程X_k依概率 1 收敛到X^*。

5.2.1 GBAS 算法的收敛性分析

定理 5.2.1[3] 满足(5.1.1)-(5.1.4)的 GBAS 马尔可夫过程$X_k=(\tau(k),W(k)),k=0,1,\cdots$依概率 1 收敛到$X^*=(\tau^*,W^*)$，其中$W^*$为一条最优路径，$\tau^*$定义如下：

$$\tau^*_{ij}=\begin{cases}\dfrac{1}{|W|}, & (i,j)\text{为}W^*\text{的一条弧},\\ 0, & \text{其他}.\end{cases} \tag{5.2.1}$$

证明 注意蚁群优化算法的 STEP3，一旦达到一个全局最优路径，由 $f(L(t))<f(W)$，我们只记录第一个最优解。因此证明分三个部分。第一证明以概率 1 到达一个最优路径，第二部分证明(5.2.1)成立，最后得到依概率 1 收敛到一个最优路径。

首先证明依概率 1 到达一个最优路径。考虑任何一条固定的最优路径W^*，令 F_k 为蚁群中某只蚂蚁在第 k 次外循环后首次走到最优路径W^*的事件，$\overline{F}_k$ 表示仅第 k 次外循环没有走到W^*的事件，前 k-1 次可能走过路径W^*。W^*永远不会被走到的事件为 $\overline{F}_1\cap\overline{F}_2\cap\cdots$。由条件概率公式，我们有

$$\begin{aligned}&\Pr(\overline{F}_1\cap\overline{F}_2\cap\cdots)\le\\ &\prod_{k=1}^{\infty}\Pr(\text{第}k\text{次循环蚁群没有走到}W^*\mid\text{第}i<k\text{次循环蚁群没有走到}W^*)\\ &=\prod_{k=1}^{\infty}\Pr(\text{前}k\text{次循环蚁群一次也没有走到}W^*),\end{aligned} \tag{5.2.2}$$

现在，对于任何一条固定的弧（i,j），在第 k 次循环结束后，其上的信息素值的一个下界可以如下给出:：考虑最坏的情形，（i,j）上的信息素一直处于挥发状态，而从来没有受过增强。不失一般性的，令 K≥2 则，对于 $k\ge K$，由(5.1.1)和(5.1.3)

$$\begin{aligned}&\tau_{ij}(k)=\prod_{l=1}^{k-1}(1-\rho_l)\tau_{ij}(1)\\ &\ge\prod_{l=1}^{K-1}(1-\rho_l)\prod_{l=K}^{k-1}\frac{\log l}{\log(l+1)}\tau_{ij}(1)\\ &=\prod_{l=1}^{K-1}(1-\rho_l)\frac{\log K}{\log k}\tau_{ij}(1)=\frac{1}{\log k}\prod_{l=1}^{K-1}(1-\rho_l)\log K\tau_{ij}(1)\end{aligned} \tag{5.2.3}$$

因为$\prod_{l=1}^{K-1}(1-\rho_l)\log K\tau_{ij}(1)$为常数，令$\Delta=\prod_{l=1}^{K-1}(1-\rho_l)\log K\tau_{ij}(1)$，

由于对于某固定的节点，其后续节点最多只有 n-1 个，而且所有弧上的信息素都是小于或者等于 1 的，由此可知

$$p_{ij}(k)\ge\frac{\Delta}{(n-1)\log k},k\ge K.$$

对于蚁群中的一只固定的蚂蚁，它在第 k ($k\ge K$)次循环走到路径$W*$的概率为

$$\prod_{(i,j)\in W^*}p_{ij}(k)\ge\left(\frac{\Delta}{(n-1)\log k}\right)^{|W^*|}。 \tag{5.2.4}$$

对一个蚁群，蚂蚁的个数至少为 1，显然，(5.2.4)也是一个蚁群到达$W*$的下界。(5.2.4)右手边的部分与循环的次数是无关的，所以对于 k=1,2,3…,都是成立的，

$$\Pr(\text{前}k\text{次循环蚁群一次也没有走到}W^*)\le1-\prod_{(i,j)\in W^*}p_{ij}(k)\le1-\left(\frac{\Delta}{(n-1)\log k}\right)^{|W^*|},$$

则(5.2.2)中右边部分的一个上界为

$$\prod_{k=K}^{\infty}\left(1-\left(\frac{\Delta}{(n-1)\log k}\right)^{|W^*|}\right), \tag{5.2.5}$$

对(5.2.5)取对数，可以得到

$$\sum_{k=K}^{\infty}\log\left(1-\left(\frac{\Delta}{(n-1)\log k}\right)^{|W^*|}\right)\le-\sum_{k=K}^{\infty}\left(\frac{\Delta}{(n-1)\log k}\right)^{|W^*|}=-\infty,$$

由于 $\sum_k(\log k)^{-L}$ 对于任意的正整数 L 来说是一个发散的级数，则(5.2.5)式右边为零，即(5.2.2)式右边项也是取零的（请读者证明，练习 1），这也就证明了事件 $F_1\cup F_2\cup\cdots$ 的概率是取 1 的。

下面证明(5.2.1)成立。由上面的证明，随机过程 $X_k=(\tau(k),W(k)),k=0,1,\cdots$ 以概率 1 到达一条最优路径。当某条最优路径被首次走到之后，用 W^* 定义为这条路径。我们可以继续类似上面的计算，当 $f(L(t))<f(W)$ 时才替换 W，即只记录第一个最优解。再重复计算时，只有 W^* 上的弧受到增强，其他弧只进行信息素的挥发。

令 K 为 W^* 首次被走到的循环次数。随机过程 $X_k=(\tau(k),W(k)),k=0,1,\cdots$ 将在第 K+1 轮循环中，满足信息素更新原则（5.1.2），我们很容易归纳证明，对于 $(i,j)\in W^*$，并且 r=1,2……

$$\tau_{ij}(K+r)=\prod_{l=K}^{K+r-1}(1-\rho_l)\tau_{ij}(K)+\frac{1}{|W^*|}\sum_{l=0}^{r-1}\rho_{K+l}\prod_{q=l+1}^{r-1}(1-\rho_{K+q}) \tag{5.2.6}$$

(在这里，当 $J=\varnothing$ 时，约定 $\prod_{j\in J}a_j=1$)。由(5.1.4)知道级数 $\sum\rho_l$ 是发散的，也就是说 $\prod_{l=1}^{\infty}(1-\rho_l)=0$（练习 1）。由此，当 $(i,j)\notin W^*$ 时，.在第 K 轮模拟之后，这条弧永远不会受到增强，所以

$$\tau_{ij}(K+r)=\prod_{l=K}^{K+r-1}(1-\rho_l)\tau_{ij}(K)\to 0,r\to\infty, \tag{5.2.7}$$

即 $(i,j)\notin W^*$ 弧上的信息素之和将趋于零。

研究(5.1.2)，可以用归纳的方法证明 $\sum_{(i,j)\in A}\tau_{ij}(k)=1,\forall k$ 成立。观察(5.2.6)右端的第二项，发现与(i,j)弧无关，再由(5.2.7)，可知 $\tau_{ij}(K+r),r\to\infty$ 的极限存在，且所有的极限值之和等于 1。对于所有的 $(i,j)\in W^*$，

$$\lim_{r\to\infty}\tau_{ij}(K+r)=\frac{1}{|W^*|}。 \tag{5.2.8}$$

也就得到了 $\lim_{l\to\infty}X_l=(\tau^*,W^*)$。

综合前两部分的讨论，特别的当 X_n 首次到达最优解 W^* 之后，由上面的讨论易知，对于 $(i,j)\in W^*$，其(5.1.1)的转移概率 $p_{ij}(l)\to 1$，也就是说 $X_k=(\tau(k),W(k)),k=0,1,\cdots$ 依概率 1 收敛到 $X^*=(\tau^*,W^*)$。 □

5.2.2 其他算法及收敛性分析

在 GBAS 中(5.1.1)保持不变，Stützle 和 Hoos[4]将信息素的挥发和增强(5.1.2)变更为

$$\tau_{ij}(k):=\begin{cases}\max\{(1-\rho)\tau_{ij}(k-1)+\rho/|W|,\tau_{\min}(k-1)\}, & (i,j)\text{为}W\text{的一条弧},\\ \max\{(1-\rho)\tau_{ij}(k-1),\tau_{\min}(k-1)\}, & \text{其他},\end{cases} \tag{5.2.9}$$

其中，$0<\rho<1$，$\tau_{\min}(k-1)$ 为实数，给出一个 MAX-MIN 蚁群优化算法。

MAX-MIN 蚁群优化算法同 GBAS 的区别在于挥发系数不随时间变化，但给出一个下界 $\tau_{\min}(k-1)$ 控制信息素痕迹的挥发。

定理 5.2.2 在算法 MAX-MIN 中，令

$$\tau_{\min}(k)=\frac{c_n}{\log(k+1)},k\geq 1,$$

其中 $\lim\limits_{k\to\infty}c_k>0$，则定理 5.2.1 的结论同样成立。

定理的证明与定理 5.2.1 是类似的。感兴趣的读者请参考[3]或自己证明（练习 2）。

蚂蚁转移概率(5.1.1)更一般规则由储存在每个节点的路由表数据结构 $A_i=\{a_{ij}\mid(i,j)\in A\}$ 决定，即转移概率为

$$p_{ij}=\begin{cases}a_{ij}(k-1)\Big/\sum\limits_{l\in T}a_{il}(k-1), & j\in T,\\ 0, & j\notin T.\end{cases} \tag{5.2.10}$$

其中，$A_i(k-1)=\{a_{ij}(k-1)\mid(i,j)\in A\}$ 取决于三部分因素，T 为从 i 可以直接到达的节点集合。第一部分为每个节点的信息素痕迹 $\tau_{ij}(k-1)$ 和预见度 $\eta_{ij}(k-1)$。第二部分为每个蚂蚁自身的记忆表中储存着的自身的历史信息。第三部分为问题的约束条件。常见的蚁群的路由表由下式求得：

$$a_{ij}(k-1)=\begin{cases}\dfrac{\tau_{ij}{}^{\alpha}(k-1)\eta_{ij}{}^{\beta}(k-1)}{\sum\limits_{l\in T}\tau_{il}{}^{\alpha}(k-1)\eta_{il}^{\beta}(k-1)}, & j\in T,\\ 0, & j\notin T.\end{cases} \tag{5.2.11}$$

式中 α——残留信息的相对重要程度，β——预见值的相对重要程度。α、β 体现了相关信息痕迹和预见度对蚂蚁决策的相对影响。

信息素痕迹 $\tau_{ij}(k-1)$——k-1 时刻连接城市 i 和 j 的路径上残留信息的浓度（相当于通过该支路的所走蚂蚁所留下的气息）。为了避免残留信息过多而引起的残留信息淹没全局求解，在一个蚁群的每一只蚂蚁完成对所有 n 个城市的访问后（也即一个循环结束后），必须对残留信息进行更新处理。模仿人类记忆的特点，对旧的信息进行削弱，同时，必须将最新的蚂蚁访问路径的信息加入 $\tau_{ij}(k-1)$。记（i, j）弧信息素在第 k-1 个循环的变化为 $\Delta\tau_{ij}(k-1)$，则信息素保留的计算为

$\tau_{ij}(k)=\tau_{ij}(k-1)+\Delta\tau_{ij}(k-1)$。

完成信息素的保留过程后，将要进行信息素的挥发过程。挥发采用

$$(1-\rho)\tau_{ij}(k)$$

的形式。其中 ρ ∈（0，1）称为信息素的衰退系数。1- ρ —残留信息被保留的部分，为了防止信息的无限累积，ρ 必须小于 1。

(5. 2. 11)的预见度 $\eta_{ij}(k-1)$ 是城市 i 转移到城市 j 的启发信息。该启发信息由待解决的问题给出。可通过一定的计算实现。在 T S P 问题中一般取 $\eta_{ij}(k-1)=1\Big/d_{ij}$（$d_{ij}$ 表示城市 i，j 间的距离）。$\eta_{ij}(k-1)$ 在这里可以称为先验知识，即若 i 与 j 间的距离短，则

$\eta_{ij}(k-1)$较大，$a_{ij}(k-1)$也较大。在(5.2.11)中，如果α=0，那么距离越近的城市被选中的概率增大；如果β=0，那么只有信息素在起作用。

设计不同的信息素痕迹保留、挥发和概率转移方式，都将带来一个理论问题，是否有类似定理 5.2.1 和 5.2.2 的结论，依概率 1 收敛到最优解？

5.3 技术问题

理论结果固然重要，其对算法的指导性是显而易见的，但算法总是在可接受的计算时间和费用的前提下实现。因此，不可避免地需要确定可控参数的技术问题。

5.3.1 解的表达形式与算法的实现

首先讨论解的表达形式。前两节都是基于 TSP 问题研究蚁群优化算法。解的形式就是所有城市的一个排列，信息素痕迹按每个弧纪录。TSP 的解是一城市的顺序，但作为一个闭圈，谁排在第一位并不重要。一般的以顺序作为解的优化问题，谁排在第一就很重要。这一类问题在应用蚁群优化算法时，只需建立一个虚拟的始终点，蚁群穴居在这个虚拟点，就可以简单地将 TSP 的解法推广，应用到诸多的优化问题，如第二章中的车间作业排序问题，第三章的下料问题。这些应用问题有一个共同的特点，解以一个顺序表示。对于 TSP 问题，寻找最短 TSP 回路是算法的目标，而对于一般的组合优化问题，算法 STEP 3 中判断句是否有 $L(s)\neq N$ 就需要根据实际问题进行修改。通过下面的例子来理解修改的基本原则。

例5.3.1 0-1背包问题的解顺序表达形式与算法实现。设有一个容积为b的背包，n个尺寸分别为$a_i\ \ (i=1,2,...,n)$，价值分别为$c_i\ \ (i=1,2,...,n)$的物品，0-1背包问题的数学模型为：

$$\max\sum_{i=1}^{n}c_i x_i$$

$$s.t.\ \sum_{i=1}^{n}a_i x_i\le b$$

$$x_i\in\{0,1\},\ \ j=1,...,n.$$

如果它的解是顺序表达形式$(0,i_1,i_2,\cdots,i_n)$，其中$(i_1,i_2,\cdots,i_n)$为$(1,2,\cdots,n)$的一个排列，如何实现蚁群优化算法的计算？

首先建立一个有向图：$G=(V,A)$，其中$V=\{0,1,2,\cdots,n\}$，$A=\{(i,j)\mid\forall i,j\in V\}$，A 中一共有 n(n+1)条弧。初始信息素痕迹定义为$\tau_{ij}=\dfrac{1}{n(n+1)}$。设第 s 只蚂蚁第 k 步所走的路线为$L(s)=(0,i_1,i_2,\cdots,i_k)$，表示蚂蚁从 0 点出发，顺序到达$i_1,i_2,\cdots,i_k$。第 k+1 步按(5.1.1)的方式行走选择$i_{k+1}$。若$\sum_{j=1}^{k+1}a_{i_j}\le b$，则$L(s)=(0,i_1,i_2,\cdots,i_k,i_{k+1})$，否则，此蚂蚁不再继续行走，退回起点。

对蚁群的 m 只蚂蚁重复上面过程，比较 m 只蚂蚁中的装包值$\sum_{\substack{i\in L(s)\\ i\neq 0}}c_i, s=1,2,\cdots,m$，并记具有最大装包值的蚂蚁为 t。在 GBAS 的 STEP3，修改为：若$f(L(t))>f(W)$，则替换当前最好解 W:=L(t)。用(5.1.2)对 W 路径上的弧信息素痕迹加强，对其他弧的信息素挥发。这样可以实现蚁群优化算法的计算。

如此进行蚁群优化算法计算会发现，最后得到的最优解不一定是一个包含所有 n 个产品

的闭环。

算法中，蚁群记录了三个信息，第一个为信息素痕迹 τ_{ij}，第二个是行走路线 $L(s)=(0,i_1,i_2,\cdots,i_k)$，第三个为问题的约束条件，既需要判断 $\sum_{j=1}^{k+1}a_{i_j}\le b$ 是否满足，以确定是否将 i_{k+1} 加入。 □

5.3.2 每一节点的记忆信息和系数的确定

GBAS 算法中蚂蚁转移概率由(5.1.1)给出。在一般的蚁群优化算法中，其基本思想完全一样，就是以概率转移到下一个节点。算法中主要有三部分信息需要记忆。第一部分信息是存在每个节点的路由表数据结构 $A_i=\{a_{ij}\mid(i,j)\in A\}$，由此决定(5.2.10)转移概率

$$p_{ij}=\begin{cases}a_{ij}(k-1)\Big/\sum_{l\in T}a_{il}(k-1), & j\in T,\\ 0, & j\notin T.\end{cases}$$

其中 T 可以看成节点 i 的邻域。$A_i(k-1)=\{a_{ij}(k-1)\mid(i,j)\in A\}$ 按(5.2.11)定义为

$$a_{ij}(k-1)=\begin{cases}\dfrac{\tau_{ij}{}^{\alpha}(k-1)\eta_{ij}{}^{\beta}(k-1)}{\sum_{l\in T}\tau_{il}{}^{\alpha}(k-1)\eta_{il}^{\beta}(k-1)}, & j\in T,\\ 0, & j\notin T.\end{cases}$$

第二部分需要记忆的信息为每个蚂蚁的记忆表中储存着的自身的历史信息。这一部分主要由算法中的 $L(s)$ 记忆，表示蚂蚁已经行走过的节点。

第三部分为问题的约束条件。在 GBAS 算法中，T 集合表示满足约束条件的候选集。在例 5.3.1 中由判别条件 $\sum_{j=1}^{k+1}a_{i_j}\le b$，则 $L(s)=(0,i_1,i_2,\cdots,i_k,i_{k+1})$ 来实现记忆功能。

α —残留信息的相对重要程度和 β —预见值的相对重要程度体现了相关信息痕迹和预见度对蚂蚁决策的相对影响。Dorigo 在求解 TSP 问题时，推荐参数的最佳设定如下：α =1，β =5， ρ =0.5。

5.3.3 蚁群的大小和停止规则

一般应用中，蚁群中蚂蚁的个数 m 是固定数，不超过 TSP 图中的节点数。终止条件主要有三类。第一类为一个给定的外循环最大数目，表明已经有足够的蚂蚁工作；第二类为当前最优解连续 K 次相同而停止的规则，其中 K 是一个给定的整数，表示算法已经收敛，不需要继续了；第三类为目标值控制规则，给定优化问题（目标最小化）的一个下界和一个误差值，当算法得到的目标值同下界之差小于给定的误差值时，算法终止。

5.3.4 在线和离线信息素修改

信息素痕迹的更新可分为离线和在线两种方式。离线方式，也称为同步更新方式，的主要思想是在若干只蚂蚁完成 n 个城市的访问后，统一对残留信息进行更新处理。信息素在线更新，也称为异步更新，则蚂蚁每行一步，马上回溯并且更新行走路径上的信息素。

对于离线方式的信息素更新，进一步可以细分为单蚂蚁离线更新和蚁群离线更新两种方式。蚁群更新方式是在蚁群中的 m 只蚂蚁全部完成 n 个城市的访问（即第 k 次蚁群循环）后，统一对残留信息进行更新处理，即

$$\tau_{ij}(k)=\tau_{ij}(k-1)+\Delta\tau_{ij}(k-1),$$

其中 $\tau_{ij}(k)$ 为第 k-1 次循环后的信息素痕迹值。单蚂蚁更新是在第 s 只蚂蚁完成对所有 n 个

城市的访问后，进行路径回溯，更新行走路径上的信息素，同时释放分配给它的资源。更新公式为

$$\tau_{ij}(s+1)=\tau_{ij}(s)+\Delta\tau_{ij}(s),$$

第 s+1 只蚂蚁根据 $\tau_{ij}(s+1)$ 重新计算的(5.2.11)出行。

在 TSP 的应用中，蚁群优化算法根据信息素痕迹更新方式的不同可以分为不同的算法。当采用离线方式，且 $\Delta\tau_{ij}(k-1)$ 或 $\Delta\tau_{ij}(s)$ 为

$$\Delta\tau_{ij}(t)=\begin{cases}\dfrac{Q}{|W|}, & (i,j)\in W,\\ 0, & (i,j)\notin W.\end{cases}$$

时，其中 W 为 t 循环中 m 只蚂蚁所行走的最佳路线或第 t 只蚂蚁所行走的一条路径，Q 为一个常数。这样的算法称为蚁环算法(ant-cycle algorithm)，特点是行走的路径越短对应保存的信息素的值就越大。

GBAS 算法就是信息素离线更新的典型算法。在信息素离线更新算法中，蚁群中蚂蚁的先后出行顺序没有相关性，前面出行的蚂蚁不影响后面出行蚂蚁的行为，但每次循环需要记录 m 只蚂蚁的行走路径，以便最后比较选取最好路径。同蚁群离线方式比较，单蚂蚁离线更新方式的一个优点是记忆信息相对较少，只需记录第 s 只蚂蚁行走的路径，并通过信息素更新后，释放该蚂蚁的所有记录信息。第 s+1 只蚂蚁根据 $\tau_{ij}(s+1)$ 重新计算的(5.2.11)出行。单蚂蚁离线方式等价于蚁群离线方式中每个蚁群只有一只蚂蚁。

信息量记忆更小的是信息素在线更新方法，蚂蚁每行一步，马上回溯并且更新行走路径上的信息素。信息素更新为

$$\tau_{ij}(k+1)=\tau_{ij}(k)+\Delta\tau_{ij}(k),$$

其中，k 为蚂蚁行走的第 k 步。

蚁量算法(ant-quantity algorithm)的信息素更新为 $\Delta\tau_{ij}(k)=\dfrac{Q}{d_{ij}}$，Q 为常量，$d_{ij}$ 表示 i 到 j 的距离，即信息浓度会因为城市距离的减小而增大。蚁密算法(ant-density algorithm) 信息素更新为 $\Delta\tau_{ij}(k)=Q$。

通过上述三种方法模拟计算比较，蚁环算法的效果最好，这是因为它用的是全局信息，而其余两种算法用的是局部信息。蚁环离线更新方法很好地保证了残留信息不至于无限累积，如果路径没有被选中，那么上面的残留信息会随时间的推移而逐渐减弱，这使算法能“忘记”不好的路径。

完成信息素的保留过程后，将要进行信息素的挥发过程。在蚁群的启发式算法的进程中，挥发过程是可选的。当采用的时候，它主要达到全局信息的集中。

5.4 应用案例——医学诊断的数据挖掘

蚁群优化算法在 TSP 的应用结果[5]非常令人鼓舞。自蚁群优化算法被成功地运用于 TSP 问题，虽然与已经发展完备的一些算法(如遗传算法等)比较起来，基本蚁群优化算法计算量比较大，而且效果也并不一定更好，但是它的成功运用还是激起了人们对蚁群优化算法的极大兴趣，并吸引了一批研究人员从事蚁群优化算法的研究。

5.4.1 数据挖掘背景介绍

数据挖掘（DM: date mining）和知识发现(KD: knowledge discovery)技术是指从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中提取隐含的、未知的、潜在的、

有用的信息的过程。数据挖掘一种公认的定义是：就是从大型数据库的数据中提取人们感兴趣的知识，这些知识是隐含的、事先未知的、潜在有用的信息，提取的知识可表示为概念（concepts）、规则（rules）、规律（regularities）、模式（patterns）等形式。此定义把数据挖掘的对象仅定义为数据库中的数据。广义地讲：数据挖掘是在一些事实或观察的集合中寻找模式的决策支持过程。也就是说，数据挖掘地对象不仅是数据库，也可以是文件系统，或其他任何数据集合中的元素。

近年来随着数据库和计算机网络的广泛应用，加上使用先进的自动数据生成和采集工具，人们所拥有的数据量急剧增大。数据的迅速增加与数据分析方法的相对滞后之间的矛盾越来越突出，人们也希望能够在对已有的大量数据分析的基础上进行科学研究、商业决策或者企业管理，但是目前所拥有的数据分析工具很难对数据进行深层次的处理，使得人们只能望“数”兴叹。数据挖掘正是为了解决传统分析方法的不足，并针对大规模数据的分析处理而出现的。数据挖掘从大量数据中提取出隐藏在数据之后的有用的信息，它被越来越多的领域所采用，并取得了较好的效果，为人们的正确决策提供了很大的帮助。

数据挖掘具有如下特点。第一，挖掘对象是海量的、复杂的各种类型的数据。这些源数据可能是残缺、充斥噪音的“脏”数据。第二，挖掘的结果是潜在的、未知的、多样性的（发现的知识可以是多种形式的）。第三，挖掘方法是不确定的、数据挖掘方面没有所谓最好的技术或通用的技术，因此，问题不是一种方法比另一种方法更好，而是哪一种更适合所要解决的问题。最后，技术的综合性。数据挖掘融入了人工智能技术、数据库技术、数理统计技术、可视化技术等技术和哲学、逻辑学等学科的知识。

数据挖掘的方法分类有多种，这是仁者见仁、智者见智的问题，都有道理。比较常见的有两种分法，一种是六分法：（1）信息论法（2）集合论法（3）仿生物法（包括神经网络和遗传算法）（4）公式发现（5）统计分析方法（包括相关分析和回归分析、聚类分析、差异分析、因子分析、判别分析等）（6）其他挖掘技术（例如：模糊论方法、可视化技术、地理信息系统、分形系统等）；另外一种是两分法：一类是统计型，常用的技术有概率分析、相关性、聚类分析和判别分析等；另一类是人工智能中的机器学习型，通过训练和学习大量的样品集得出需要的模式或参数。数据挖掘的应用中，最终的目标都是发现有价值的知识和信息，有共同的思路和步骤，但也存在很大的差异和区别。

随着计算机技术迅速发展，数据库技术也得到了广泛的应用。现在的数据库(data base)系统可以高效地实现数据的录入、查询、统计计算等功能。如果用数据库管理系统来存储数据，用机器算法来分析数据，挖掘数据背后的知识，这样就产生了数据库中的知识发现(即KDD)和数据挖掘。KDD 和 DM 是指识别出存在于数据库中有效的、新颖的、具有潜在效用的、最终可理解的模式的非平凡过程。

数据挖掘的任务分成很多种，包括数据描述(characterization)、数据分类(classification)、数据关联(association)、区别（判别）分析(discrimination)、数据回归、数据聚类(clustering)、数据预测(prediction)。

我们这里要用蚁群优化算法解决的是数据分类这样一个问题：我们预先定义一组类，然后把数据系中的每一个数据根据该数据的属性，归入这些类中的一个。

比如说病情诊断：在医院里，病人感觉不适，要求医生进行检查。医生对一些以前的状况进行询问并进行一些必要的检查，可以得到很多方面的参数。比方对于 SARS 病情，可以得到的一些相关参数：体温，咳嗽，肺部 X 照射是否有阴影，以及各种症状持续时间，病人前一段时间接触过何种人群等等，这些大量的数据比较容易得到，如何根据这些大量的病人的病情参数来判断是否确实为 SARS 病症并不是一件很简单的事情，这就是我们的数据分类工作要做的事情。

我们要对数据进行分类，首先要有进行分类的规则，我们把判别规则表示为如下形式：

IF < conditions > THEN < class >

其中判别规则的前导部分（IF 部分）是一个条件项集合的与运算。条件项是一个只有<属性，

关系符，属性取值>三个元素的简单的条件。不同的条件项用逻辑运算符连接，一般是 AND。所以规则的前导部分就是一个由许多简单的条件逻辑连接而成的复合条件句。符合这个条件的数据将被纳入规则后部的 class 中。

从规则有意义和可理解性来考虑，简单条件不能太多，而属于规则的数目也不能太多。

5.4.2 蚁群优化算法的实现[6]

通过对蚁群优化算法作适当的改进，得到了有关医学诊断数据挖掘的蚁群优化算法，下面介绍蚁群优化算法的几个关键的地方。

问题数学模型化，将问题的解对应于蚂蚁行走的路，规则表示为如下形式：

IF< term1 AND term2 AND … > THEN < class >

规则的 IF 部分由很多项目（term）组成，每个项目是一个三元组<属性，关系符，属性取值>三个元素的简单的条件，其中，属性表示为项目的名称，如属性可为性别、体温、年龄等；在本应用中关系符取等号；属性取值表示项目的属性值。如果属性取值是连续的，在程序运行前，必须先对其作离散和分级预处理。

例如一个单位开运动会，分组方法是：

IF(性别=男)AND(年龄≤40) AND(年龄≥18) AND(单位=数学系) THEN (男甲)

IF(性别=男)AND(年龄≤50) AND(年龄>40)THEN (男乙)

上面的例子中，可以将年龄离散化为[18, 40]用 1 表示，(40，50]用 2 表示，（50，60]用 3 表示，则上面两句可以重新写成

IF(性别=男)AND(年龄=1) AND(单位=数学系) THEN (男甲)

IF(性别=男)AND(年龄=2)THEN (男乙)

还需要注意的是判别 IF 句的项目数不一定相同，第一句中用了三个判别条件，而第二句中用两个判别条件。如果将这两个语句中作为规则在同一个单位应用的话，则发现第一个语句的中出现冗余，即不需要(单位=数学系)。于是，我们的目的就是在一个规则中使用最少的判别语句。

对给定的一种病，如非典型肺炎，病情诊断时，可以得到的项目三元数据组数据可能很多，我们的目的是用尽量少的项目数据（三元数据组）来确诊病情。因此，采用蚁群优化算法达到最优的选择。

例 5.4.1 如何画出用于蚁群优化算法的非典型肺炎症状图？目前非典型肺炎(SARS)主要通过临床诊断，假设参考的因素有：发热、胸部透视阴影、传染病流行病学史，加上目前新开发的一些检测方法得到的数据，还有一些其他的参考指标，再加上职业等一系列参考因素，列表如表 5.4.1。

表 5.4.1 非典型肺炎病例调查项目

属性	发热	职业	胸部阴影	流行病学史
值	（0，1，2，3）	（0，1，2）	（0，1，2）	（0，1）
说明	0：不考虑该属性 1：37.5 度以下 2：37.5-38.5 度 3：38.5 度以上	0：不考虑该属性 1：医务人员 2：其他人员	0：不考虑该属性 1：无 2：有	0：不考虑该属性 1：无 2：有

观察表 5.4.1，我们发现属性之间没有直接相关性。因此很难将每一个属性看成图的一个节点。

假设每种属性只能出现在一个条件项中，不允许出现诸如：“IF(性别=男)AND(性别=女)”。这样，将每个属性复制一次分别作为一个子图的起点和终点，并将每个属性值看成一个中间节点，可以把蚂蚁走过的路和条件的选取联系起来，表 5.4.1 对应结构为图 5.4.1。其

中，蚂蚁在始终节点出发，每行走一步可以增加一个属性，也可以绕过这个属性，最后返回始终节点。一个蚂蚁行走 k（k≤4）步，得到一条路径为 route[k]={0,2,1,0}。它的含义是：诊断一共有四个属性，第一个和第四个属性不考虑，第二个属性考虑 2（其他人员），第三个属性考虑 1（胸部无阴影），在图 4.4.1 中用黑虚线表示。对应的判别句，也是一个规则，为

IF(职业=其他人员)AND(胸部阴影=无)THEN(非典型肺炎)。

这个判别规则肯定不符合非典型肺炎的诊断标准，因此一定要给一个非常差的评价函数值。

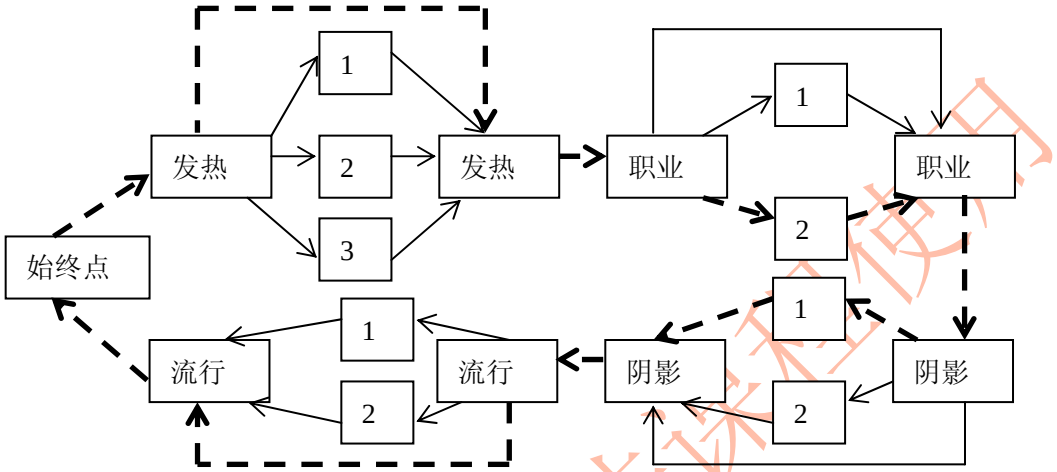


图 5.4.1 非典型肺炎诊断图结构

蚂蚁可以每次选择一个结点作为它的行进方向，选择一条到达该结点评价函数值最小的路。此处的费用最小值就是诊断的最小误诊率。这样，解就和一条连接部分结点的路联系起来了。□

例 5.4.1 描述了解的表示形式。假设 a 表示属性的总个数，第 i 个属性的取值域中可取 b_i 个数值。在例 5.4.1 中，$a=4$，$b_1=3$，$b_2=b_3=b_4=2$。于是，一只蚂蚁的行走 k 步的路线可以表示为

$$route[k]=(y_1,y_2,\cdots,y_a), \tag{5.4.1}$$

其中，y_i 在属性 i 的值域空间外加一个 0 取值，$y_i=0$ 表示规则中不包含属性 i，$(y_1,y_2,\cdots,y_a)$ 的最大取值个数为 $(b_1+1)\times(b_2+1)\times\cdots\times(b_a+1)$。

当确定一条蚂蚁的行走路线，解的评价函数定义为

$$Q=\frac{TP}{TP+FN}\cdot\frac{TN}{FP+TN}, \tag{5.4.2}$$

其中：

TP：(true positives) 属于该类的待检验数据被判断为该类的数目；

TN：(true negatives) 不属于该类的待检验数据被判为不属于该类的数目；

FP：(false positives) 属于该类的待检验数据被判为不属于该类的数目；

FN：(false negatives) 不属于该类的待检验数据被判为属于该类的数目。

可以看出，Q 的数值一定在[0,1]之间，而且 Q 的数值越接近 1，则表面规则对该类的判断越准确。(5.4.2)中，$\frac{TP}{TP+FN}$ 称为敏感度(sensitivity)，代表该规则判断为属于该类的待检验数据中确实属于该类的比例；$\frac{TN}{FP+TN}$ 称为明确度(specificity)，代表该规则判断为不属于该类的待检验数据中确实不属于该类的比例。这是对一个完整的规则的评价函数，如果需要对一个条件项（三元组）进行评价，那么就是对加入条件项前后规则的性能进行比较。

这个评价是针对有训练样本进行的，说明了本节介绍的蚁群算法是一种需要预先进行学习的智能算法。

从例 5.4.1 看到，蚂蚁每走一步，一个属性加入到路线当中，但这个属性可以不考虑，即对应项的值为 0，也可以是这个属性的具体值。按蚁群优化算法的随机性，对应一个转移概率，转移概率定义为

$$P_{ij}=\frac{\eta_{ij}\tau_{ij}(t)}{\sum_{q=0}^{b_i}\eta_{iq}\tau_{iq}(t)},i=1,2,\cdots,a,\ j=0,1,\cdots,b_i, \tag{5.4.3}$$

其中：η_{ij} 表示每个条件项的启发式参数值（预见值），这个参数的计算依据是信息理论，在后续部分讨论；$\tau_{ij}(t)$ 表示第 i 个属性的第 j 个取值在 t 时刻的信息痕迹量。这个参数在算法的执行过程中不断更新，每次的更新量与该条件项是否包括在蚂蚁行走的路线上有关，如果包含在路线上，则信息量增加，否则信息量减少；改变量与当前路线的性能有关，性能越好，变化量越大。

(5.4.3)与[6]的定义略有不同。在[6]中，蚂蚁不是按属性的固有顺序行走，可以先到任何一个属性项，这样不需要虚拟一个 j=0 的属性值，转移概率为

$$P_{ij}=\frac{x_i\eta_{ij}\tau_{ij}(t)}{\sum_{p=1}^{a}x_p\sum_{q=1}^{b_p}(\eta_{pq}\tau_{pq}(t))},i=1,2,\cdots,a,\ j=1,2,\cdots,b_i, \tag{5.4.4}$$

如果属性 i 的任何一个条件项都不在规则中，x_i 置 1，表示可以对该条件项选择，否则置 0。这样可以保证当前蚂蚁选择所有下一条件项的概率和为 1。

每只蚂蚁走完后，将根据它搜索到的路径（规则）的性能优劣来更新所有节点的信息量，增加它走过路的信息量，同时减少没有选择到的路的信息量。这样，下一只蚂蚁将在新的信息量分布下开始它的规则构建（寻路）。

算法开始的时候，设置：

$$\tau_{ij}(t=0)=\frac{1}{\sum_{i=1}^{a}(b_i+1)},i=1,2,\cdots,a;\ j=0,1,\cdots,b_i, \tag{5.4.5}$$

在[6]中，

$$\tau_{ij}(t=0)=\frac{1}{\sum_{i=1}^{a}b_i},i=1,2,\cdots,a;\ j=1,2,\cdots,b_i, \tag{5.4.6}$$

也就是说，对于所有的可以选择的条件项，信息量都是均匀分布的。一旦一个规则建立后，即蚂蚁走出一条路径，具体的更新过程如下。第一步：

$$\tau_{ij}(t+1)=\tau_{ij}(t)+\tau_{ij}(t)\cdot Q,\forall(i,j)\in R, \tag{5.4.7}$$

其中，$0\le Q\le 1$，R 是当前规则（路径）中所有包含的条件项。这一步使得被选中的三元组的信息量增加。第二步：

$$\tau_{ij}(t+1)=\frac{\tau_{ij}(t)}{\sum_{i=1}^{a}\sum_{j=0}^{b_i}\tau_{ij}(t)},i=1,2,\cdots,a;b=0,1,\cdots,b_i, \tag{5.4.8}$$

在[6]中，

$$\tau_{ij}(t+1)=\frac{\tau_{ij}(t)}{\sum_{i=1}^{a}\sum_{j=1}^{b_i}\tau_{ij}(t)},i=1,2,\cdots,a;b=1,2,\cdots,b_i, \tag{5.4.9}$$

这一步使得所有三元组的信息量之和为 1，并且减少了没有被选中的三元组的信息量,相当于信息素痕迹挥发过程。

每一个条件项都有一个相关的 η_{ij} 的值，这个值为条件项的信息熵量度。根据信息理论，如果 A_i 是第 i 个属性，而 V_{ij} 是该属性值域中的第 j 个取值，那么这个条件项的熵值为：

$$H(W|A_i=V_{ij})=-\sum_{w=1}^{K}(P(w|A_i=V_{ij})\cdot\log_2 P(w|A_i=V_{ij}))$$

这里 W 是一个类属性，K 是类的数目，举例来说，如果给定的是病人数据，则可能包含多种疾病，K 就是疾病的种类数，$P(w|A_i=V_{ij})$ 表示 $A_i=V_{ij}$ 数据属于第 w 类的经验概率。因为我们是采用训练集的方式，所以这个概率可以通过对训练集数字统计，然后得到 η_{ij}：

$$\eta_{ij}=\frac{\log_2 K-H(W\mid A_i=V_{ij})}{\sum_{i=1}^{a}x_i\sum_{j=0}^{b_i}(\log_2 K-H(W\mid A_i=V_{ij}))} \tag{5.4.10}$$

这里的 a、x_i 和 b_i 与（5.4.4）的含义相同。

我们对这个式子作一个直观的理解：$H(W|A_i=V_{ij})$ 越大，说明 $A_i=V_{ij}$ 条件项对类的不确定度越大，这个属性 A_i 取 V_{ij} 在各个类分布的更加均匀，所以 $A_i=V_{ij}$ 对类的确定的贡献很小，于是就希望以比较小的概率取到这个条件项，所以在 η_{ij} 的表达式中要用 $\log_2 K-H(W\mid A_i=V_{ij})$ 作分子，并且用所有项的和作分母，使得概率和为 1。$H(W|A_i=V_{ij})$ 越小，$A_i=V_{ij}$ 被选中的可能越小。

这个启发式函数有两个极限情况：如果属性 A_i 取 V_{ij} 的数据在训练集中没有出现，那么 $H(W|A_i=V_{ij})$ 被置最大值 $\log_2 K$，从而使得 η_{ij} 为 0，这个条件项不会被选中。如果所有的训练集中属性 A_i 取 V_{ij} 的数据都属于同一个类，那么 $H(W|A_i=V_{ij})$ 被置为 0，从而使得 η_{ij} 为最大，这样这个条件项以最大的概率被选中。

算法的特点是每次把属性和属性值成对作为一个整体来计算其信息熵，熵参数一直在信息更新，贯穿始末。因为不断的反馈和重建，使用信息痕迹量参数和信息熵共同作用，使得蚂蚁算法更具鲁棒性。

算法描述：

```
TrainingSet={所有训练数据}:
DiscoveredRuleList=[];                    //初始规则集为一个空集
    WHILE(TrainingSet>Max_uncovered_cases)
        t=1;
        j=1;
        由(5.4.5)赋予相同的初始信息素;
      REPEAT
          第 t 只蚂蚁 Ant(t)从一个空规则集开始，每次加入一个属性项，逐步建立规则集 R(t) ;
          更新 R(t);
```

在 Ant(t)走过的路线上信息素痕迹增强(根据 R(t)的比例)且在没有走过的路线上减少 信息素痕迹（挥发）；
IF(R(t) 与 R(t-1) 相同) //收敛性检验
Then j=j+1;
ELSE j=1;
END IF
t=t+1;
UNTIL (t>=No_of_ants) OR (j>=No_rules_converg)
对所有的蚂蚁，选择 R(t)中最好的规则 Rbest;
将规则 Rbest 添加到 DiscoveredRuleList;
TrainingSet=TrainingSet-{Rbest 已正确识别的病例数据};
END WHILE

其中，三个参数为 Max_uncovered_cases， No_of_ants 和 No_rules_converg，分别表示，没有诊断或没有正确诊断集中样本数量，蚂蚁的数量和当前最优解不变化的最大迭代数。

这是一个异步算法，由于算法每次迭代只派出一个蚂蚁，因此蚂蚁数量（No_of_ants）是 REPEAT 次数最大数目，表示已经有足够的蚂蚁工作；连续 n（n=No_rules_converg）次得到相同的规则（解），表示算法已经收敛，不需要继续了。

在将规则 Rbest 添加到 DiscoveredRuleList 语句中，外加一个参数 Min_cases_per_rule，表示每一个诊断规则至少要正确诊断 Min_cases_per_rule 个病例。这表示诊断规则的通用性。涵盖的数据太少，分类就没有意义了。

5.4.3 具体实现和数据结果分析

这里我们有六种病例的一组测试数据，数据来源见表 5.4.2，这些数据中，有连续的属性，也有离散的属性，所以要先对连续属性作离散化。

表 5.4.2 病例数据来源

数据库	病例数	离散属性种类	连续属性种类	数据库种类
Ljubljana breast cancer	282	9	0	2
Wisconsin breast cancer	683	0	9	2
Tic-tac-toe	958	9	0	2
Dermatology	366	33	1	6
Hepatitis	155	13	6	2
Cleveland Heart disease	303	8	5	5

对于以上数据，蚁群优化算法的关键参数设置为：Max_uncovered_cases=10，No_of_ants=3000，No_rules_converg=10，Min_cases_per_rule=10。这些参数的选取没有经过任何优化，从而得到一组结果，与一个著名的分类算法 CN2 进行比较，计算结果见表 5.4.3 和表 5.4.4。

表 5.4.3 诊断准确度比较（均值域标准差）

数据库	蚁群优化算法诊断精度	CN2 诊断精度
Ljubljana breast cancer	75.28±2.24	67.69±3.59
Wisconsin breast cancer	96.04±0.93	94.88±0.88

Tic-tac-toe	73.04±2.53	97.38±0.52
Dermatology	94.29±1.20	90.38±1.66
Hepatitis	90.00±3.11	90.00±2.50
Cleveland heart disease	59.67±2.50	57.48±1.78

在诊断准确度上，有四组数据得到了比 CN2 明显好的结果，一组数据相同，一组数据结果比 CN2 要差。

在诊断规则的确定上，CN2 一旦确定一个诊断规则，则将这个规则确定，并将这个规则可以诊断的病例删除。然后开始新的诊断规则的确定，蚁群优化算法采用同样的逻辑。在诊断规则精简性能上，计算结果见表 5.4.4。

表 5.4.4 诊断规则数（均值域标准差）

数据库	蚁群优化算法诊断规则数	CN2 诊断规则数
Ljubljana breast cancer	7.10±0.31	55.40±2.07
Wisconsin breast cancer	6.20±0.25	18.60±0.45
Tic-tac-toe	8.50±0.62	39.70±2.52.
Dermatology	7.30±0.15	18.50±0.47
Hepatitis	3.40±0.16	7.20±0.25
Cleveland heart disease	9.50±0.92	42.40±0.71

在诊断规则的精简上，蚁群优化算法相对 CN2 有及其明显的优势。总体而言，蚁群优化算法的结果优于 CN2。

计算复杂度： 一般来说，计算复杂度为：$O\left(r\cdot z\cdot\left[k\cdot a+k^3\cdot n\right]+a\cdot n\right)$，最坏情况下，有：$O\left(r\cdot z\cdot a^3\cdot n\right)$，其中 n 为数据数目，a 为属性数目，z 为蚂蚁数目，r 已经发现的规则数目。

蚁群优化算法来源于对自然界蚂蚁寻找从蚁巢到食物的最短路径并找到回巢路径方法的研究。它是一种并行算法，所有“蚂蚁”（工作单元）独立行动，没有监督机构；它是一种合作算法，每一只“蚂蚁”选择路径时，有残留信息的路径被选中的可能性要比没有残留信息的路径大得多；它是一种鲁棒算法，因为只要对算法作小小的修改，就可以运用于别的组合优化问题。

蚁群优化算法的历史还不太长，它的发展远没有形成完整的理论体系，许多理论问题和实际运用问题还有待于逐步解决。但是可以预料，随着研究的深入，蚁群优化算法将给我们展示一个分布式和网络系统的优秀寻优算法。

练习题

1. 若 $0\le a_k<1$，$\sum_{k=1}^{\infty}a_k=\infty$，证明：$\prod_{k=1}^{\infty}\left(1-a_k\right)=0$。

2. 基于 GBAS 算法，Stützle 和 Hoos[4]给出一个 MAX-MIN 蚁群优化算法。他们定义信息素的挥发和增强为

$$\tau_{ij}(k):=\begin{cases}\max\left\{(1-\rho)\tau_{ij}(k-1)+\rho/|W|,\tau_{\min}(k-1)\right\}, & (i,j)\text{为}W\text{的一条弧},\\ \max\left\{(1-\rho)\tau_{ij}(k-1),\tau_{\min}(k-1)\right\}, & \text{其他},\end{cases}$$

其中，$0<\rho<1$，$\tau_{\min}(k-1)$为实数。令 $\tau_{\min}(k)=\dfrac{c_n}{\log(k+1)},k\geq 1,$

其中$\lim\limits_{k\to\infty}c_k>0$，证明 MAX-MIN 算法以概率 1 收敛全局最优解。

3．尝试用 GBAS 和 MAX-MIN 求解 TSP 问题，并比较它们的计算效果。

参考文献

[1] Dorigo M. Optimization, learning and natural algorithms. PhD Thesis, Department of Electronics, Politecnico di Milano, Italy, 1992

[2] Gutjahr WJ. A graph-based ant system and its convergence. Future Generation Computing Systems, 2000, 16:873-888

[3] Gutjahr WJ. ACO algorithms with guaranteed convergence to the optimal solution. Information Processing Letters, 2002, 82:145-153

[4] Stützle T, Hoos HH. MAX-MIN ant system. Future Generation Computing Systems, 2000, 16:889-914

[5] Stützle T, Dorigo M. ACO algorithms for the traveling salesman problem. In: Miettinnen K et al(eds). Evolutionary algorithms in engineering and computer science. John Wiley & Sons, 1999.

[6] Parpinelli R S, Lopes H S, Freitas A A. Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computing, 2002, 6:321-332.

第六章 人工神经网络

James[1]在 1890 年的《心理学(Psychology(Briefer Course))》一书中这样描述神经网络的基本原理：大脑皮层每一点的活力是由其他点势能释放的综合效能产生。这一势能同下面的因素有关：第一，相关其他点的兴奋次数；第二，兴奋的强度；第三，与其不相连的其他点所接受的能量。

他的这一原理一直沿用至今。神经网络的研究分为两个派别。一个派别主要包括生物学家、物理学家和心理学家，他们研究的主要目的是给出大脑活动的精细模型和描述。另一派别主要包括工程技术人员，他们关心的是怎样利用神经网络的基本原理，来构造解决实际问题的算法，使得这些算法具有有趣和有效的计算能力。本章内容属于后者。我们称后者为神经网络的工程应用研究，或称为人工神经网络。

神经网络的基本原理是构造人工神经网络模型的一个基本依据。1943 年 McCulloch 和 Pitts[2]建立了第一个人工神经网络模型，后被扩展为“认知(perceptron)”模型。认知模型的一个功效可以用来解决简单的分类问题。1969 年 Minsky 和 Papert[3]在《认知论(Perceptrons)》一书中指出：认知模型无法解决最为经典的“异或(XOR——exclusive-or)”问题。这个结论使得人工神经网络陷入一次危机。产生这次危机的原因可以归结为以下主要原因：第一，人们对人工神经网络的认识不足和认识上的错误；第二，人们过高的期望和夸大；第三，宣传中的错误引导。实际上，Minsky 和 Papert 主要研究的是单隐含层的认知网络模型，他们证明了简单的线性感知功能是有限的。这个结论不应该对人工神经网络进行全面的否定。

二十世纪八十年代，Hopfield[4,5]将人工神经网络成功地应用在组合优化问题，McClelland 和 Rumelhart[6]构造的多层反馈学习算法成功地解决了单隐含层认知网络的“异或”问题及其他的识别问题。他们的突破使得人工神经网络成为新的研究热点。

本章介绍人工神经网络(artificial neural networks)的两类主要模型：前向神经网络和反馈神经网络。6.1 节介绍人工神经网络得基本概念，6.2 节介绍单层前向神经网络，6.3 节介绍多层前向神经网络，6.4 节介绍竞争学习神经网络，6.5 节介绍反馈性 Hopfield 神经网络。

6.1 人工神经网络的基本概念

图 6.1.1 是生物学中神经网络的简图。

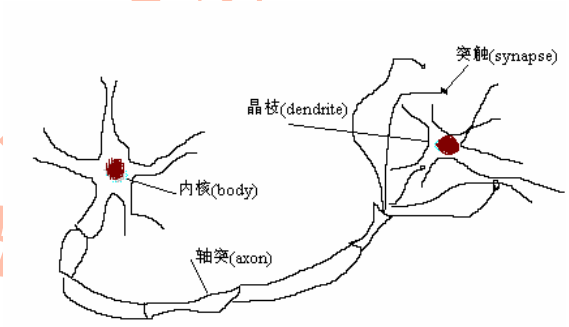


图 6.1.1 生物学中神经网络简图

大脑的一个重要成分是神经网络。神经网络由相互关联的神经元组成。每一个神经元由内核(body)、轴突(axon)和晶枝(dendrite)组成。晶枝形成一个非常精密的“毛刷”环绕在内核周围。轴突可以想象为一根又长又细的管道，其终点分为众多细小分支，将内核的信息传递给其他内核的晶枝。这些细小分支的头，即那些又长又细管道的终点，称为突触

删除的内容:

(synapse)，它们的主要工能是接触其他内核的晶枝。

一个神经元根据晶枝接收到的信息，通过内核进行信息处理，再通过它所控制的突触传送给其他的神经元。神经元可分为“抑制”性的或“兴奋”性的两种。当一个神经元的晶枝接收的兴奋性信息累计超出某一值时，神经元被激活并传递出一个信息给其他神经元，这个值称为阈值(threshold)。这种传递信息的神经元为“兴奋”性的。第二种情况是神经元虽然接收到其他神经元传递的信息，但没有向外传递信息，此时，称这个神经元为“抑制”性的。

简单模拟上面原理的人工神经网络是 McCulloch—Pitts 认知网络。假设一个神经元通过晶枝接收到 n 个信息，McCulloch—Pitts 认知网络由图 6.1.2 表示。

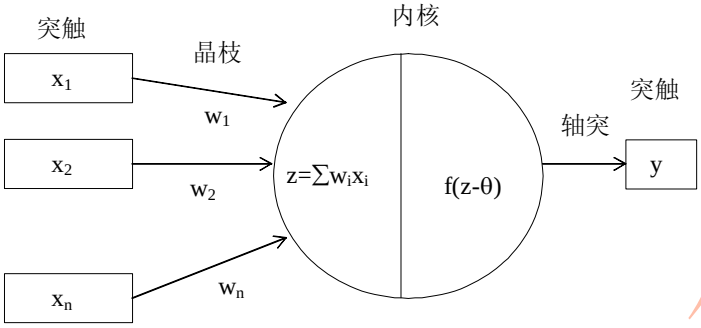


图 6.1.2 McCulloch—Pitts 网络

在图 6.1.2 中，w_i 为关联权，表示神经元对第 i 个晶枝接收到信息的感知能力。f 称为输出函数或激活函数(activation function)，$y = f(z-\theta)$ 为输出神经元的输出值。McClulloch—Pitts 输出函数定义为

$$y = f(z-\theta) = \mathrm{sgn}(\sum_{i=1}^{n} w_i x_i - \theta), \tag{6.1.1}$$

其中，

$$\mathrm{sgn}(x) = \begin{cases} 1, x \geq 0, \\ 0, \text{其他}, \end{cases} \tag{6.1.2}$$

$\theta \geq 0$ 时，θ 称为阈值；$\theta < 0$ 时，$-\theta$ 称为神经元的外部刺激值；一般称 θ 为神经元的激励值。

从方程(6.1.1)可以看出，当 $w_i (i = 1,2,\cdots,n)$ 和 θ 为给定值时，对一组输入 $(x_1, x_2, \cdots, x_n)^T$，很容易计算得到输出值。我们的想法就是对给定的输入，尽可能使(6.1.1)的计算输出同实际值吻合。这就要求确定参数 $w_i (i = 1,2,\cdots,n)$ 和 θ。

人工神经网络的建立和应用可以归结为三个步骤：网络结构的确定，关联权的确定和工作阶段。

网络结构的确定 主要内容包含：网络的拓扑结构和每个神经元激活函数的选取。

拓扑结构是一个神经网络的基础。前向型人工神经网络的特点是将神经元分为层，每一层内的神经元之间没有信息交流，信息由后向前一层一层地传递。反馈型神经网络则将整个网络看成一个整体，神经元相互作用，计算是整体性的。

激活函数的类型比较多，主要有阶跃函数（6.1.2），线性函数

$$f(x) = ax + b. \tag{6.1.3}$$

其中，a 和 b 是实常数。

S(Sigmoid)函数(见图 6.1.3)：

删除的内容: Sigmiod

$$f(x) = \frac{1}{1+e^{-x}}, \tag{6.1.4}$$

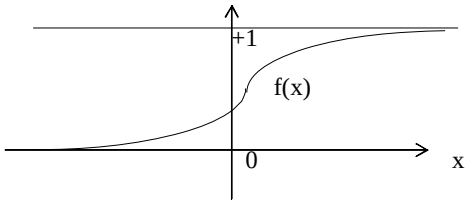


图 6.1.3 S 形函数

在识别和归类问题中，如果采用阶跃函数，当输出值为 1 时，我们可以肯定地说出输入的归类。阶跃函数的缺点是数学性质较差，如在 0 点不光滑。Sigmoid 函数弥补了这一方面的不足，使得函数值在(0,1)区间连续变化。Sigmoid 函数又称 S 形函数。

为了取得其他区间的函数输出，需对阶跃函数或 S 形函数进行简单的修改。如需要{-1,+1}的输出，阶跃函数为 $2\,\mathrm{sgn}(x)-1$，S 形函数为 $2\,f(x)-1$。

关联权和θ的确定 权和θ是通过学习（训练, train）得到，分为有指导学习和无指导学习两类。在已知一组正确的输入输出结果的条件下，人工神经网络依据这些数据，调整并确定权数 w_i 和θ，使得网络输出同理想输出偏差尽量小的方法称为有指导学习。在只有输入数据而不知输出结果的前提下，确定权数和θ的方法称无指导学习。在学习过程中，不同的目标函数得到不同的学习规则。

工作阶段(simulate) 在权数和θ确定的基础上，用带有确定权数的神经网络去解决实际问题的过程称为工作。

当然，学习和工作并不是绝对地分为两个阶段，他们相互相承。可以通过学习、工作、再学习、再工作的循环过程，逐渐提高人工神经网络的应用效果。

图 6.1.4 是前向型人工神经网络的计算流程。第一个阶段如图 6.1.4(a)描述，它的主要步骤是：在选择网络结构模型和学习规则后，根据已知的输入和理想输出学习数据，通过学习规则确定神经网络的权数。尤如一个医学院的学生，通过教科书中病例的发病症状和诊断结果，来学习诊断。第二个阶段如图 6.1.4(b)描述，它的主要步骤是：根据第一个阶段确定的模型和得到的权数和θ，在输入实际问题的输入数据后，给出一个结论。尤如一个医学院的毕业生，在遇到病人后，根据医学院学到的诊断方法，给病人一个诊断。

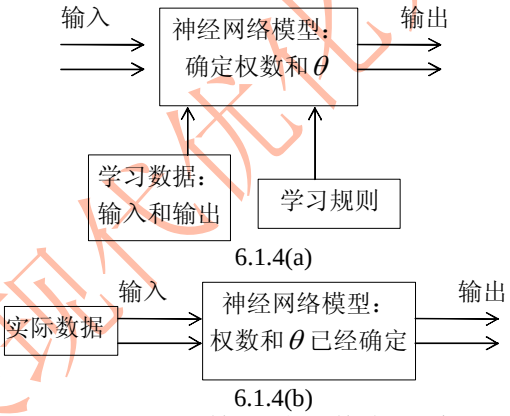


图 6.1.4 人工神经网络计算过程示意图

依据神经网络的层数和激活函数的类型，前向型神经网络还可以进一步细分。下面就从简到难的情况来逐一介绍前向型神经网络。

6.2 单层前向神经网络

单层前向神经网络(single layer feed-forward neural networks)的模型结构如图 6.2.1

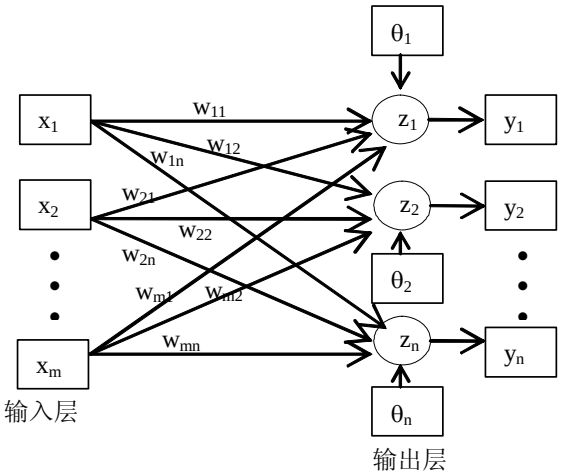


图 6.2.1 单层前向神经网络

在单层前向神经网络中，只有输入层和输出层。输入层中的每一个元素表示轴突，而输出层中的每一个元素代表一个神经元的内核，由此得名为单层神经网络。它的特点是只有一个输入层和一个输出层。单层前向神经网络中各层中的元素无权数相连。输出层通过对输入层的输入数据和同理想输出结果的比较，确定其权数和θ。最后，用确定了权数和θ的神经网络应用到实际问题。这类神经网络主要在识别问题中有较强的能力。

对于有指导学习情况，记输入变量为 $X=(x_1,x_2,\cdots,x_m)^T$，输出结果为 $Y=(y_1,y_2,\cdots,y_n)^T$，激励值为 $\theta=(\theta_1,\theta_2,\cdots,\theta_n)^T$，$w_{ij}$ 为输入 i 同神经元 j 的权数，第 j 个神经元所接收到的值为

$$z_j=\sum_{i=1}^{m}w_{ij}x_i,,\quad y_j=f_j(z_j-\theta_j),\quad j=1,2,\cdots,n。\tag{6.2.1}$$

用矩阵表示为

$$W=\begin{bmatrix}w_{11}&w_{12}&\cdots&w_{1n}\\w_{21}&w_{22}&\cdots&w_{2n}\\\cdots&\cdots&\cdots&\cdots\\w_{m1}&w_{m2}&\cdots&w_{mn}\end{bmatrix},\quad Z=W^TX-\theta,\quad Y=(y_1,y_2,\cdots,y_n)^T。\tag{6.2.2}$$

6.2.1 线性网络

最为简单的一类神经网络是线性网络。可以讨论多种激活函数。一类最为简单的激活函数是线性函数(6.1.3)，称该神经网络为简单线性网络。

满足(6.1.3)的一种简单情况（其中 a=1,b=0）是

$$Y=W^TX\ 。\tag{6.2.3}$$

每一个神经元满足

$$y_j=(w_{1j},w_{2j,}\cdots,w_{mj})\begin{pmatrix}x_1\\x_2\\\vdots\\x_m\end{pmatrix}。\tag{6.2.4}$$

从简单线性神经网络(6.2.4)可以看出，其每个神经元的输出只与其相关的权数和输入有关，因此，n 个神经元是相互独立的，所以可以采用单个神经元研究其神经网络的特性。

例 6.2.1 (6.2.3)决定的单层线性网络无法识别“异或问题”(XOR——exclusive-or)。异或问题是逻辑中最为简单的一种运算关系，两个逻辑变量 x_1, x_2 的运算及结果 y 满足

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

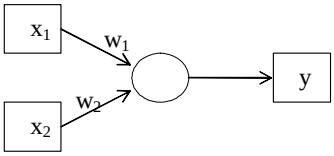


图 6.2.1 异或问题网络

图 6.2.1 是求解两变量一个输出的单层线性神经网络。如果这个神经网络可以求解异或问题，则必须满足 $y = w_1x_1 + w_2x_2$。求解这个方程得到

$$\begin{cases} w_2 = 1, \\ w_1 = 1, \\ w_1 + w_2 = 0. \end{cases}$$

这是一个矛盾方程，故无法确定权使得单层线性神经网络来识别异或问题。□

虽说单层线性神经网络有例 6.2.1 的致命缺陷，但多数研究者仍然继承其研究方法。为了达到神经网络输出同理想输出最为接近，一个易于理解的学习规则为平均最小二乘最小(LMS—least mean squares)。设一组学习样本为

$$\{(X(1), D(1)), (X(2), D(2)), \cdots, (X(J), D(J))\},$$

其中，$X(t) = (x_1(t), x_2(t), \cdots, x_m(t))^T$ 和 $D(t) = (d_1(t), d_2(t), \cdots, d_n(t))^T$。目标函数为

$$F(W) = \frac{1}{J}\sum_{t=1}^{J}(Y(t) - D(t))^T(Y(t) - D(t)), \qquad (6.2.5)$$

其中，$Y(t) = W^T X(t)$。

$$\frac{\partial F(W)}{\partial W} = \frac{1}{J}\frac{\partial}{\partial W}\sum_{t=1}^{J}(W^T X(t) - D(t))^T(W^T X(t) - D(t))$$

$$= \frac{2}{J}(X(1), X(2), \cdots, X(J))(Y(1) - D(1), Y(2) - D(2), \cdots, Y(J) - D(J))^T.$$

记 $\overline{X} = (X(1), X(2), \cdots, X(J))$，$\overline{Y} = (Y(1), Y(2), \cdots, Y(J))$ 和 $\overline{D} = (D(1), D(2), \cdots, D(J))$，W 的一个驻点满足

$$\frac{\partial F(W)}{\partial W} = 2(\overline{X}\overline{X}^T W - \overline{X}\overline{D}^T) = 0, \qquad (6.2.6)$$

只要输入 $X(t) = (x_1(t), x_2(t), \cdots, x_m(t))^T (t = 1,2,\cdots, J)$ 的向量秩为 m，我们就可以通过解析求解，得到（6.2.6）的最优解

$$W = (\overline{X}\overline{X}^T)^{-1}\overline{X}\overline{D}^T. \qquad (6.2.7)$$

当 $X(t) = (x_1(t), x_2(t), \cdots, x_m(t))^T (t = 1,2,\cdots, J)$ 的向量秩小于 m 时，同样可以解方程组（6.2.6），解得一个解。

当学习样本一个个地到达，如自适应线性（Adaline，Adaptive Linear）网络，每到达一个样本，系统将学习并更新权数。设 $X = (x_1, x_2, \cdots, x_m)^T$ 和 $D = (d_1, d_2, \cdots, d_n)^T$ 分别是随机选取的一对学习输入输出向量，$Y = W^T X$ 是实际输出向量。由(6.2.5)

$$F(W) = \sum_{j=1}^{n}\left(\sum_{l=1}^{m} w_{lj}x_l - d_j\right)^2。 \qquad (6.2.8)$$

进而

$$\frac{\partial F(W)}{\partial w_{ij}} = 2x_i(\sum_{l=1}^{m} w_{lj}x_l - d_j) = 2(y_j - d_j)x_i \text{。}$$

F(W)下降的方向为

$$\delta w_{ij} = -\frac{\varepsilon}{2}\frac{\partial F(W)}{\partial w_{ij}} = \varepsilon(d_j - y_j)x_i, i = 1,2,\cdots,m; j = 1,2,\cdots,n, \qquad (6.2.9)$$

其中，ε为学习效率。于是权数的变化形式为

$$W(t+1) = W(t) + \delta W(t) = W(t) + \varepsilon_t\left((d_j(t) - y_j(t))x_i(t)\right)_{m\times n}. \qquad (6.2.10)$$

(6.2.7)和(6.2.10)都是由平均最小二乘规则得到的权数变化公式，它们的区别是：(6.2.7)将所有的学习样本看成一个总体，一次计算出权数值，使得学习数据的计算输出与理想输出平方和偏差最小；(6.2.10)每学习一对输入和对应的理想输出，都要对权数进行修改。

(6.2.9)中的 $y_j(t)$ 表示第 t 组学习数据的输出，它是一个已知的值。是在学习了 $t-1$ 组数据后，得到 W(t)，而后由 $Y(t) = W^T(t)X(t)$ 确定，$d_j(t)$ 是期望的目标输出。

LMS 应用在 Adaline 得到的（6.2.10）称为 Widrow-Hoff 学习规则，也称 Hebb 规则。因此，LMS、Hebb 规则、Adaline 规则有时混称。它们可以统称为误差修正规则。

实际上，(6.2.9)是函数 F(W)下降的方向。在什么样的条件下，可以保证（6.2.10）迭代收敛且为(6.2.5)的局部最优解 W*？Widrow 和 Stearns[7]给出收敛的充分条件为

$$\sum_{t=1}^{\infty}\varepsilon_t = \infty, \qquad \sum_{t=1}^{\infty}\varepsilon_t^{\,2} < \infty \text{。} \qquad (6.2.11)$$

很明显，$\varepsilon_t = \frac{1}{t}$ 满足(6.2.11)的条件。

定理 6.2.1 当 $X = (X(1), X(2),\cdots, X(m))$ 为标准正交向量时，若 W(0)=0，通过学习规则(Hebb 规则的变形)

$$\delta w_{ij}(t) = \varepsilon_t d_j(t)x_i(t), i = 1,2,\cdots,m; j = 1,2,\cdots,n,$$

其中，$D = (D(1), D(2),\cdots, D(m))$ 为理想输出，学习 m 次，则由(6.2.3)决定的单层线性神经网络可以区分这 m 个向量，且对输入 $X(t) = (x_1(t), x_2(t),\cdots, x_m(t))^T (t = 1,2,\cdots,m)$，认知为 $\varepsilon_t D(t)$。

证明：在学习 m 次以后，由 $Y(t) = W^T(t)X(t)$ 和(6.2.9)得到

$$\delta W(t) = (\varepsilon_t x_1(t), \varepsilon_t x_2(t),\cdots,\varepsilon_t x_m(t))^T (d_1(t), d_2(t),\cdots, d_n(t)),$$
$$t = 1,2,\cdots,m,$$

再由(6.2.10)和 W(0)=0，得

$$W(m+1) = W(m) + \delta W(m) = \sum_{t=1}^{m}\delta W(t) \text{。}$$

重新认知 X(t)，有

$$Y = W^T(m+1)X(t) = \left(\sum_{\tau=0}^{m}\delta W(\tau)\right)^T X(t)$$

$$= \sum_{\tau=1}^{m}(d_1(\tau), d_2(\tau),\cdots, d_n(\tau))^T$$

$$(\varepsilon_\tau x_1(\tau), \varepsilon_\tau x_2(\tau),\cdots,\varepsilon_\tau x_m(\tau))(x_1(\tau), x_2(\tau),\cdots, x_m(\tau))^T$$

$$= \varepsilon_t D(t).$$

由此，证明结论。□

定理 6.2.1 说明了 Hebb 规则对正交的向量组有非常好的识别性能。这是它的长处。若

识别的向量不具有正交性，则 Hebb 规则的认知性就不一定很好。例 6.2.1 就说明了问题。

由前面的讨论，单层线性前向神经网络层的多个神经元的研究可以按一个神经元研究。求解满足(6.2.5)的 W 取决所有可能的样本。通过所有的样本计算求解(6.2.5)是不经济的，也是不现实的。对单层线性神经网络可以通过概率分析求解 W。只讨论一个神经元的情形。

设输入 $X=(X_1,X_2,\cdots,X_m)^T$ 和理想输出 d 为随机变量，LMS 的期望模型为

$$F_E(W)=E(d-Y)^2,$$

其中 $Y=W^TX$ 。变形为

$$F_E(W)=E(d-W^TX)^2=E(d^2-2W^TXd+W^TXX^TW)$$
$$=E(d^2)-2W^TE(dX)+W^TE(XX^T)W.$$

这是一个二次型，最小值存在的充分条件为 $E(XX^T)$ 正定，其中

$$E(XX^T)=\left(E(X_iX_j)\right)_{m\times m}。$$

$E(XX^T)$ 正定的一个充分条件是：$X_iX_j(i\neq j)$ 相互独立，$E(X_i)=0,i=1,2,\cdots,m$ 和 $E(X_i^2)\neq 0,i=1,2,\cdots,m$。

在 $E(XX^T)$ 正定的条件下，由

$$\frac{\partial F_E}{\partial W}=\begin{pmatrix}\frac{\partial F_E}{\partial w_1}\\ \frac{\partial F_E}{\partial w_2}\\ \vdots\\ \frac{\partial F_E}{\partial w_m}\end{pmatrix}=-2E(dX)+2E(XX^T)W=0,$$

得

$$W^*=E(XX^T)^{-1}E(dX)。\qquad(6.2.12)$$

有多个神经元的情况下，理想输出为 $D=(d_1,d_2,\cdots,d_n)^T$。由(6.2.12)第 j 个神经元的权为

$$W^*_j=\begin{pmatrix}w_{1j}\\ w_{2j}\\ \vdots\\ w_{mj}\end{pmatrix}=E(XX^T)^{-1}E(d_jX),j=1,2,\cdots,n。$$

在大多数情况下，求解 $E(XX^T)$ 是比较难实现的。解决的办法是数值求解（6.2.10）。

6.2.2 非线性网络

当激活函数选非线性函数时，神经网络称为非线性神经网络。常见的非线性函数为(6.1.2)和(6.1.4)。

符号函数是阶跃函数，在物理中容易实现。在识别和归类问题中，当输出值为 1 时，我们可以肯定地说出输入的归类。它的缺点是数学性质较差，如在零点不光滑。S(Sigmoid)形函数弥补了这一方面的不足，使得函数值在(0,1)区间连续变化。Sigmoid 函数又称 S 形函数。从符号函数和 Sigmoid 函数及神经网络结构可以看出，前向单层非线性神经网络非常适合分类和判定问题，如以输出值 0 和 1 表示分类问题或以输出为“否”和“是”的判定问题。由于符号函数和 S 形函数本身所具有的特性，在实际应用中多采用符号函数，因为这个函数的值有明确的 0 和 1 归类；理论分析中常采用后一个函数，因为这个函数的数学

性质非常好。

先研究符号函数的性质。从下例可以看到，符号函数决定的单层非线性神经网络同样无法实现 XOR 功能。

例 6.2.2 采用符号函数的单层非线性神经网络同样无法实现 XOR 功能。从例 6.2.1 得到

x_1	x_2	z	期望输出
0	0	0	0
0	1	w_2	1
1	0	w_1	1
1	1	w_1+w_2	0

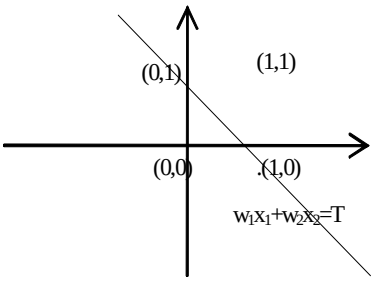


图 6.2.2 $w_1x_1+w_2x_2=T$ 分割平面图

$w_1x_1+w_2x_2=T$ 将平面分成两部分（见图 6.2.2）。如果具有区分功能，则满足(0,0)和(1,1)在一半平面内的同时，(0,1)和(1,0)在另一半平面内，即要求

$$(0,0),(1,1)\in H1=\{(x_1,x_2)\mid w_1x_1+w_2x_2<T\},$$

且

$$(0,1),(1,0)\in H2=\{(x_1,x_2)\mid w_1x_1+w_2x_2\geq T\},$$

或反之。但满足这样要求

$$\begin{cases} w_1+w_2<T, \\ w_2\geq T, \\ w_1\geq T, \end{cases}$$

的 w_1,w_2,T 是不存在的，因此，结论成立。□

以单个神经元单一随机样本讨论学习规则。符号函数决定的神经网络的学习规则同线性网络的学习规则相同。设给定初始 W(0)和激励 $\theta(0)$，学习规则是

$$W(t+1)=\varepsilon_t X(t)\{D(t)-Y(t)\}^T+W(t)。 \qquad (6.2.13)$$

删除的内容:

其中，$X(t)=(x_1(t),x_2(t),\cdots,x_m(t))^T$，$D(t)=(d_1(t),d_2(t),\cdots,d_n(t))^T$，$Y(t)=W^TX(t)$。从(6.2.13)可以得到，

(1) 当 $d_j(t)=y_j(t)$ 时，表明第 j 个神经元对第 t 组输入 X(t)达到了理想输出，此时，$w_{ij}(t+1)=w_{ij}(t),i=1,2,\cdots,m$，即与神经元 j 相关的权数不修改。此时神经元 j 的激励值也不修改；

(2) 当 $d_j(t)=1,y_j(t)=0$ 时，表明第 j 个神经元对第 t 组输入 X(t)没有达到理想输出，此时有

$$z_j(t)=\sum_{i=1}^{m}w_{ij}(t)x_i(t)<\theta_j(t),$$

于是(6.2.13)修改权数使得

$$z_j(t+1)=\sum_{i=1}^{m}w_{ij}(t+1)x_i(t)=\varepsilon_t\sum_{i=1}^{m}x_i^2(t)+\sum_{i=1}^{m}w_{ij}(t)x_i(t)\geq z_j(t),$$

激励值修改为

$$\theta_j(t+1)=\theta_j(t)-\varepsilon_t; \tag{6.2.14}$$

（3）当$d_j(t)=0, y_j(t)=1$时，表明第 j 个神经元对第 t 组输入 X(t)没有达到理想输出，此时有

$$z_j(t)=\sum_{i=1}^{m}w_{ij}(t)x_i(t)\geq\theta_j(t),$$

于是(6.2.13)修改权数使得

$$z_j(t+1)=\sum_{i=1}^{m}w_{ij}(t+1)x_i(t)=-\varepsilon_t\sum_{i=1}^{m}x_i^2(t)+\sum_{i=1}^{m}w_{ij}(t)x_i(t)\leq z_j(t),$$

将激励值修改为

$$\theta_j(t+1)=\theta_j(t)+\varepsilon_t。 \tag{6.2.15}$$

从理论来考虑上述学习规则的收敛性。符号函数$(f_1(u_1(t)), f_2(u_2(t)),\cdots, f_n(u_n(t)))^T$中

$$u_j(t)=\sum_{i=1}^{m}w_{ij}(t)x_j(t)-\theta_j(t)。 \tag{6.2.16}$$

在学习过程中，W(t)和阈值$\theta(t)=(\theta_1(t),\theta_2(t),\cdots,\theta_n(t))^T$都是待确定的参数。记广义权和输入为

$$\overline{W(t)}=(W(t),\theta(t))^T,\quad \overline{X(t)}=(X(t),-1)^T,$$

(6.2.16)变形为

$$u(t)=(u_1(t),u_2(t),\cdots,u_n(t))^T=\overline{(W(t))}^T\,\overline{X(t)}。 \tag{6.2.17}$$

这是一个 n+1 个神经元的单层神经网络问题，其阈值为 0。

如果给定的分类问题或判定问题是线性可分的，即如图 6.2.4 所示，存在一条直线（平面）将平面（空间）分为两部分，使得理想输出为 1 和 0 的输入点分别在各自一半的部分中。

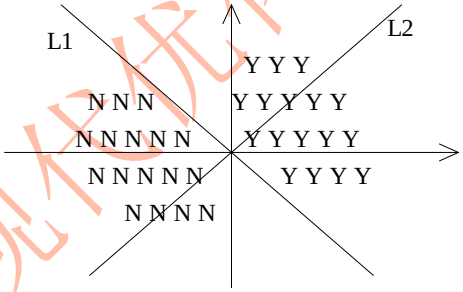


图 6.2.4 线性可分示意图

(6.2.17)过原点，由上面(1)、(2)和(3)的讨论，当已有的权数使得神经网络不能区分“Y”和“N”两部分时，即如图 6.2.4 中的直线（平面）L2：$u(t)=W^T(t)X-\theta(t)$时，从理论上来说，只要按一定的条件，用(6.2.13)和(6.2.14)或(6.2.15)修改广义权和适当选取 ε_t，L2 最终逼近 L1。为了便于理解，以输出为一维的情形时，修改广义权的办法是：先以理想输出为 1 的输入数据学习，使得神经网络可以识别理想输出为 1 的数组。在学习的过程中要求适当选取 ε_t 使得直线 L1 不能变化太快，以避免将已经识别的数据重新不能识别，且至少使一个没有识别的数组得以识别。在理想输出为 1 的输入数据得以识别后，再用类似的方

法，识别理想输出为 0 的输入数据。由此可以有限步识别线性可分问题。总结上面的讨论，得到这样的结论：如果给定的分类问题或判定问题是线性可分的且 ε_t 选取合适，则(5.14)的学习规则使得神经网络可以识别这些问题。

上面讨论了一层线性与非线性神经网络，下面将介绍一个多层神经网络的模型、学习规则和算法。

6.3 多层前向神经网络

一般的多层前向神经网络(multi-layer feed forward neural networks)的结构如图 6.3.1 所示。它们有两层以上（包括两层）的神经元，第 i 层的神经元的输出为第 i+1 层神经元的输入，但在一层之内的神经元是没有输入输出关系。除输入和输出层以外的神经元层称为隐层。记输入层为第 0 层，则输出层所在的层数为神经网络的层数。

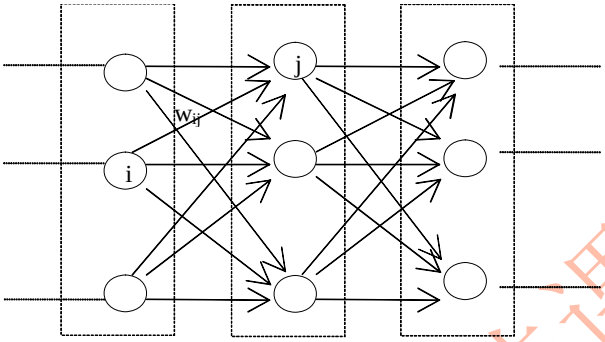


图 6.3.1 多层前向神经网络结构示意图

记第 $k-1$ 层到第 k 层的权数矩阵为

$$W_k = \left(w_{ij}^{\ k}\right)_{n_{k-1}\times n_k},$$

其中，n_k 表示第 k 层神经元个数。输入向量为 $X = (x_1, x_2, \cdots, x_{n_0})^T$。

若多层神经网络的每一层都采用(6.2.3)的激活函数输出，称为多层简单线性前向网络，第 n 层的输出为

$$Z_n = (W_n)^T (W_{n-1})^T \cdots (W_1)^T X \text{ 。}$$

因此，多层简单线性前向网络同(6.2.3)等价，无法区别 XOR 问题。

6.3.1 非线性神经网络

采用非线性激活函数的神经网络，它区别异或问题的效果又如何？

例 6.3.1 设两层前向神经元网络如图 6.3.2，

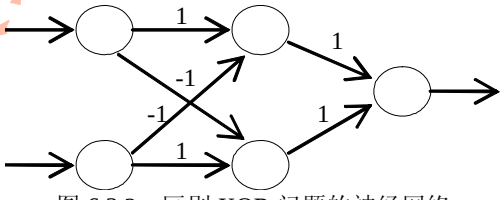


图 6.3.2 区别 XOR 问题的神经网络

激活函数采用符号函数，所有阈值为一个充分小的正数ε。我们讨论图 6.3.2 的神经网络是否能够区别 XOR 问题。

删除的内容: 0

由

$$W_1=\begin{pmatrix}1 & -1\\ -1 & 1\end{pmatrix},\ W_2=\begin{pmatrix}1\\ 1\end{pmatrix},$$

当 $X=\begin{pmatrix}1\\0\end{pmatrix}$ 时，

$$Z_1=W_1^T\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\-1\end{pmatrix},\ Y_1=f(Z_1-\begin{pmatrix}\varepsilon\\ \varepsilon\end{pmatrix})=\begin{pmatrix}1\\0\end{pmatrix},$$
$$Z_2=W_2^TY_1=1,\ Y_2=f(Z_2-\varepsilon)=1.$$

当 $X=\begin{pmatrix}0\\1\end{pmatrix}$ 时，

$$Z_1=W_1^T\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}-1\\1\end{pmatrix},\ Y_1=f(Z_1-\begin{pmatrix}\varepsilon\\ \varepsilon\end{pmatrix})=\begin{pmatrix}0\\1\end{pmatrix},$$
$$Z_2=W_2^TY_1=1,\ Y_2=f(Z_2-\varepsilon)=1.$$

当 $X=\begin{pmatrix}1\\1\end{pmatrix}$ 时，

$$Z_1=W_1^T\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix},\ Y_1=f(Z_1-\begin{pmatrix}\varepsilon\\ \varepsilon\end{pmatrix})=\begin{pmatrix}0\\0\end{pmatrix},$$
$$Z_2=W_2^TY_1=0,\ Y_2=f(Z_2-\varepsilon)=0.$$

当 $X=\begin{pmatrix}0\\0\end{pmatrix}$ 时，

$$Z_1=W_1^T\begin{pmatrix}0\\0\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix},\ Y_1=f(Z_1-\begin{pmatrix}\varepsilon\\ \varepsilon\end{pmatrix})=\begin{pmatrix}0\\0\end{pmatrix},$$
$$Z_2=W_2^TY_1=0,\ Y_2=f(Z_2-\varepsilon)=0.$$

可以看出，由阈值为 0 的符号函数决定的图 6.3.2 神经网络能够区别 XOR 问题。□

虽说采用符号函数的多层前向神经网络有很强的分类功能，如对 XOR 问题的识别，但因没有很好的学习方法，故在实际问题中应用较少。而采用线性函数的多层前向神经网络，由于同单层线性神经网络的功效等价，因此，无法解决复杂问题的分类或识别问题。

S 形或线性函数有很好的函数特性，其效果又近似符号函数，因此现主要讨论采用 S 形和线性函数的前向多层神经网络的学习方法。

假设一个 K 层神经网络每一层的输入输出关系如下：从第 0 层到第一层的原始输入向量、权矩阵、接收值向量和输出向量及它们之间的关系分别为

$$X=\begin{pmatrix}x_1\\x_2\\\vdots\\x_{n_0}\end{pmatrix},\ W_1=\left(w_{ij}^1\right)_{n_0\times n_1},\ Z_1=\begin{pmatrix}z_1^1\\z_2^1\\\vdots\\z_{n_1}^1\end{pmatrix}=W_1^TX,\ Y_1=\begin{pmatrix}y_1^1\\y_2^1\\\vdots\\y_{n_1}^1\end{pmatrix},\qquad(6.3.1)$$

第 k-1 层到第 k 层的权矩阵、接受值向量和输出向量及它们之间的关系分别为

$$W_k=\left(w_{ij}^k\right)_{n_{k-1}\times n_k},\ Z_k=\begin{pmatrix}z_1^k\\z_2^k\\\vdots\\z_{n_k}^k\end{pmatrix}=W_k^TY_{k-1},\ Y_k=\begin{pmatrix}y_1^k\\y_2^k\\\vdots\\y_{n_1}^k\end{pmatrix},\tag{6.3.2}$$

其中，$y_i^k=f_i(z_i^k),i=1,2,\cdots,n_i,k=1,2,\cdots,K$。

与（6.2.17）结论相同的方法，将激励值看成一个神经元的权数，神经元的输入输出可以写成(6.3.1)和(6.3.2)的形式。

类似 6.2 节对单层线性网络学习规则的讨论方法，我们先讨论单样本学习规则。学习的原则是：确定 W，使得

$$F(W)=(D-Y_K)^T(D-Y_K)\tag{6.3.3}$$

最小，其中，$D=(d_1,d_2,\cdots,d_{n_K})^T$ 为理想输出。分输出层、隐层及输入层对学习规则讨论。

一、输出层

对输出层的每一个神经元，由(6.3.3)得

$$\frac{\partial F}{\partial w_{ij}^K}=-2(d_j-y_j^K)\frac{\partial y_j^K}{\partial w_{ij}^K},$$
$$\frac{\partial y_j^K}{\partial w_{ij}^K}=\frac{dy_j^K}{dz_j^K}y_i^{K-1},\qquad i=1,2,\cdots,n_{K-1},j=1,2,\cdots,n_K.\tag{6.3.4}$$

二、隐层及输入层

对第 k 层($k\le K-1$)的一个神经元 j，当 k=K−1 时，

$$\begin{aligned}\frac{\partial F}{\partial w_{ij}^{K-1}}&=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{\partial y_{l_K}^K}{\partial w_{ij}^{K-1}}\\&=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\frac{\partial z_{l_K}^K}{\partial w_{ij}^{K-1}}\\&=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\sum_{l_{K-1}=1}^{n_{K-1}}w_{l_{K-1}l_K}^K\frac{\partial y_{l_{K-1}}^{K-1}}{\partial w_{ij}^{K-1}}\\&=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}w_{jl_K}^K\frac{dy_j^{K-1}}{dz_j^{K-1}}y_i^{K-2},\end{aligned}\tag{6.3.5}$$

$$i=1,2,\cdots,n_{K-2};j=1,2,\cdots,n_{K-1}.$$

图 6.3.3 给出(6.3.5)各参数的关系。

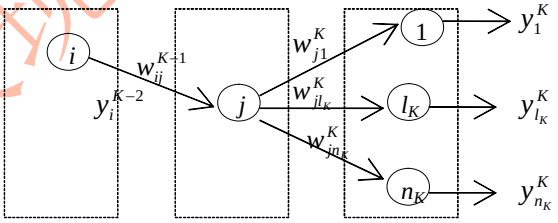


图 6.3.3　(6.3.5)的参数关系

当 $k\le K-2$ 时，有

$$\frac{\partial F}{\partial w_{ij}^{k}}=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^{K})\frac{dy_{l_K}^{K}}{dz_{l_K}^{K}}\sum_{l_{K-1}=1}^{n_{K-1}}w_{l_{K-1}l_K}^{K}\frac{dy_{l_{K-1}}^{K-1}}{dz_{l_{K-1}}^{K-1}}\cdots$$

$$\sum_{l_{k+2}=1}^{n_{k+2}}w_{l_{k+2}l_{k+3}}^{k+3}\frac{dy_{l_{k+2}}^{k+2}}{dz_{l_{k+2}}^{k+2}}\sum_{l_{k+1}=1}^{n_{k+1}}w_{l_{k+1}l_{k+2}}^{k+2}\frac{dy_{l_{k+1}}^{k+1}}{dz_{l_{k+1}}^{k+1}}w_{jl_{k+1}}^{k+1}\frac{dy_{j}^{k}}{dz_{j}^{k}}y_i^{k-1},\qquad(6.3.6)$$

$$i=1,2,\cdots,n_{k-1};\ j=1,2,\cdots,n_k,$$

其中，输入层 k=0 时 $y_i^0=x_i$，$1\le i\le n_0$。

由于(6.3.4)、(6.3.5)和(6.3.6)比较难以记忆，可以用矩阵的形式表示。记

$$B_{K+1}=\begin{pmatrix}d_1-y_1^K\\ d_2-y_2^K\\ \vdots\\ d_{n_K}-y_{n_K}^K\end{pmatrix},\ W_{K+1}=I(\text{单位阵}),$$

则(6.3.4)为

$$F(W)=(B_{K+1})^T B_{K+1}。\qquad(6.3.7)$$

记

$$B_K=diag(\frac{dy_1^K}{dz_1^K},\frac{dy_2^K}{dz_2^K},\cdots,\frac{dy_{n_K}^K}{dz_{n_K}^K})W_{K+1}B_{K+1},$$

其中，$diag(a_1,a_2,\cdots,a_n)$ 表示以 $a_1,a_2,\cdots,a_n$ 为对角元素的 n 阶对角阵。(6.3.4)改写为

$$\left(\frac{\partial F(W)}{\partial w_{ij}^K}\right)_{n_{K-1}\times n_K}=-2\begin{pmatrix}y_1^{K-1}\\ y_2^{K-1}\\ \vdots\\ y_{n_{K-1}}^{K-1}\end{pmatrix}(B_K)^T.\qquad(6.3.8)$$

记

$$B_{K-1}=diag(\frac{dy_1^{K-1}}{dz_1^{K-1}},\frac{dy_2^{K-1}}{dz_2^{K-1}},\cdots,\frac{dy_{n_{K-1}}^{K-1}}{dz_{n_{K-1}}^{K-1}})W_K B_K,$$

则(6.3.5)改写成

$$\left(\frac{\partial F(W)}{\partial w_{ij}^{K-1}}\right)_{n_{K-2}\times n_{K-1}}=-2\begin{pmatrix}y_1^{K-2}\\ y_2^{K-2}\\ \vdots\\ y_{n_{K-1}}^{K-2}\end{pmatrix}(B_{K-1})^T.\qquad(6.3.9)$$

记

$$B_k=diag(\frac{dy_1^k}{dz_1^k},\frac{dy_2^k}{dz_2^k},\cdots,\frac{dy_{n_k}^k}{dz_{n_k}^k})W_{k+1}B_{k+1}$$

则(6.3.6)改写为

$$\left(\frac{\partial F(W)}{\partial w_{ij}^k}\right)_{n_{k-1}\times n_k}=-2\begin{pmatrix}y_1^{k-1}\\ y_2^{k-1}\\ \vdots\\ y_{n_{k-1}}^{k-1}\end{pmatrix}(B_k)^T.\qquad(6.3.10)$$

当第 t 个学习样本值 X(t)输入后，可以依次得到

$$Y_1(t)=(y_1^1(t),y_2^1(t),\cdots,y_{n_1}^1(t))^T,$$
$$Y_2(t)=(y_1^2(t),y_2^2(t),\cdots,y_{n_2}^2(t))^T,$$
$$\cdots$$
$$Y_K(t)=(y_1^K(t),y_2^K(t),\cdots,y_{n_K}^K(t))^T,$$

此时，用(6.3.8)(6.3.9)(6.3.10)和已知的 W(t)可以得到相应的值，用带有参数 t 形式的 $B_k(t)$ 和 $\left(\frac{\partial F(W)}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k}$ 表示对每一个样本 X(t)的计算值，按 F(W)下降的方向确定权数的学习规则为

$$W_k(t+1)=W_k(t)+\delta W_k(t),\ k=K,K-1,\cdots,1, \tag{6.3.11}$$

其中，

$$\delta W_k(t)=-\frac{1}{2}\varepsilon_t\left(\frac{\partial F(W)}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k}=\varepsilon_t\begin{pmatrix}y_1^{k-1}(t)\\ y_2^{k-1}(t)\\ \vdots\\ y_{n_{K-1}}^{k-1}(t)\end{pmatrix}(B_k(t))^T, \tag{6.3.12}$$

ε_t 为第 t 步的学习效率。

从(6.3.11)的学习规则看到这样一个规律：W_k 的修正是从最后一层神经元的权数 W_K 开始，反向递推修正 K－1 层的 W_{K-1}，…，一直修正到第一层的权数 W_1。由此将(6.3.11)学习规则称为反推(BP)学习规则(Back Propagation)。采用 S 形函数的前向多层神经网络有下面的反推学习算法。这是一个有监督的学习算法。

反推学习算法

STEP1 选定学习的数组 $\{X(t),D(t)\},t=1,2,\cdots,J$，随机确定初始权矩阵 W(0);

STEP2 用学习数据 X(t)计算 $Y_1(t),Y_2(t),\cdots,Y_K(t)$；

STEP3 用(6.3.8)(6.3.9)(6.3.10)计算，反向修正 W(t)，修正公式为(6.3.11)；直到学习完所有的数组。

当激活函数 $f(x)=\frac{1}{1+e^x}$ 时，

$$\frac{dy_j^k}{dz_j^k}=f(z_i^k)(1-f(z_j^k)),\ i=1,2,\cdots,n_{k-1},\ j=1,2,\cdots,n_k, \tag{6.3.13}$$

代入(6.3.8)(6.3.9)(6.3.10)使得计算得以简化。

BP 算法弥补了符号函数在实际应用中难以确定权数的不足，使得具有很强识别功能的前向多层神经网络得以应用，但应用 BP 算法中需要注意以下主要问题：

第一，激活函数为 S 形函数或线性函数。由线性网络的讨论，如果网络中的全部神经元都取线性函数，那么，同线性网络没有任何区别。

第二，当激活函数为 S 形函数时，输出值只能趋近 0 或 1。于是，在神经网络的工作期，判定一个数组的归类问题就需要给出一个分界值，如 0.5，而不是绝对的 1 或 0。

第三，W(0)的选取最好是随机的。当 W(0)的所有权数相等时，由(6.3.2)的 $Z_k=(W_k)^TY_{k-1},k=1,2,\cdots,K$ 推出：无论什么样的输入 X，Z_1 中的全部分量相同；当第一层所有神经元的激活函数取同一个函数时，Y_1 的全部分量相同；当每一层的神经元选相同激活函数时，由于 Z_k 的各分量相同，所以 $\frac{dy_1^k}{dz_1^k}=\cdots=\frac{dy_{n_k}^k}{dz_{n_k}^k}$，再由(6.3.8)(6.3.9)(6.3.10)的

运算得 $\left(\frac{\partial F(W)}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k}$ 中的所有元素相等。由此得到的修正权对很多问题无法识别。

第四，算法的全局最优性，即算法在什么样的条件下收敛？能否收敛到区局最优点？

只要学习效率 ε_t 选取合适，(6.3.8)、(6.3.9)和(6.3.10)决定的下降算法，类似(6.2.11)，收敛到局部最优，参考[8]。为了避免局部最优解的一个改进方法是将(6.3.11)学习规则中的 $\delta W_k(t)$ 修正为

$$\delta W_k(t) = -\frac{1}{2}\varepsilon_t\left(\frac{\partial F(W)}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k} + \alpha_t \delta W_k(t-1), \tag{6.3.14}$$

(6.3.14)右端的第二项相当于一个势能“惯性”，式中的 α_t 决定于权数两次变化的相互关系。

第五，对于多样本学习，即一次是用一组样本学习，此时的学习规则是使

$$F(W) = \frac{1}{J}\sum_{t=1}^{J}(D(t)-Y_K(t))^T(D(t)-Y_K(t))$$

最小。由于 $F(W)=\sum_{t=1}^{J}F_t(W)$, 其中

$$F_t(W) = (D(t)-Y_K(t))^T(D(t)-Y_K(t)),$$

所以学习规则是用 J 个样本的(6.3.12)累加来修正权数。

6.3.2 BP 算法的一个示例

为了说明 BP 算法的计算过程，用图 6.3.4 的共有 6 个神经元的两层前向神经网络[9]去产生图 6.3.5 的理想输出。

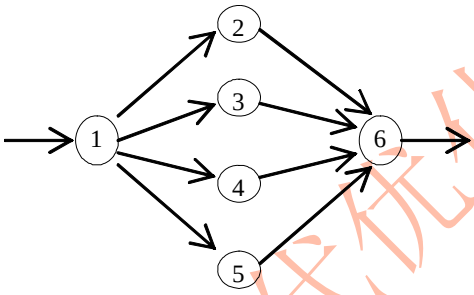


图 6.3.4 6 个神经元的两层神经网络

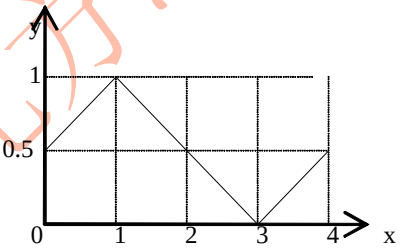


图 6.3.5 理想输出

假设每个神经元的激励值为 0，权矩阵随机选取为

$$w_{12}=0.2, w_{13}=0.3, w_{14}=0.4, w_{15}=0.5,$$

$$w_{26}=0.5, w_{36}=0.2, w_{46}=0.1, w_{56}=0.4.$$

学习数据如表 6.3.1，按输入变量每 0.05 为一个分割点，共有 80 个数组。

表 6.3.1 学习数据

输入值	输出值	输入值	输出值	输入值	输出值
0.000	0.500	1.800	0.600	2.900	0.050
0.050	0.525	1.850	0.575	2.950	0.025
0.100	0.550	1.900	0.550	3.000	0.000
0.150	0.575	1.950	0.525	3.050	0.025
0.200	0.600	2.000	0.500	3.100	0.050
…	…	2.050	0.475	…	…

0.900	0.950	2.100	0.450	3.800	0.400
0.950	0.975	2.150	0.425	3.850	0.425
1.000	1.000	2.200	0.400	3.900	0.450
1.050	0.975	2.250	0.375	3.950	0.475
1.100	0.950	…	…	4.000	0.500
…	…	…	…		

激活函数采用

$$f(x)=\frac{1}{1+e^{-x}}.$$

(6.3.12)的学习效率 ε_t=1。从表 6.3.1 的开始对每一个数据进行学习。

第一次学习时：

$$x_1=0,$$
$$z_2=z_3=z_4=z_5=0,$$
$$y_i=f(z_i)=\frac{1}{2},i=2,3,4,5,$$
$$z_6=\frac{1}{2}(0.5+0.2+0.1+0.4)=0.6,$$
$$y_6=0.6457.$$

反推确定第二层权数变化：

$$B_3=(0.5-0.6457)=(-0.1457),$$
$$B_2=\frac{dy_6}{dz_6}B_3=0.6457\times(1-0.6457)\times(-0.1457)=-0.0333.$$

梯度矩阵

$$\left(\frac{\partial F(W)}{\partial w_{ij}^2}\right)_{6\times1}=-2\begin{pmatrix}y_2\\y_3\\y_4\\y_5\end{pmatrix}(B_2)^T=-2\begin{pmatrix}-0.0167\\-0.0167\\-0.0167\\-0.0167\end{pmatrix},$$

由(6.3.11)的学习规则得

$$\begin{pmatrix}w_{26}\\w_{36}\\w_{46}\\w_{56}\end{pmatrix}:=\begin{pmatrix}0.5\\0.2\\0.1\\0.4\end{pmatrix}-\frac{1}{2}\left(\frac{\partial F(W)}{\partial w_{ij}^2}\right)_{6\times1}=\begin{pmatrix}0.4833\\0.1833\\0.0833\\0.3833\end{pmatrix}.$$

反推确定第一层的权数变化:

$$B_1=diag(0.5\times(1-0.5),0.25,0.25,0.25)\begin{pmatrix}w_{26}\\w_{36}\\w_{46}\\w_{56}\end{pmatrix}(B_2)^T$$
$$=diag(0.25,0.25,0.25,0.25)\begin{pmatrix}0.5\\0.2\\0.1\\0.4\end{pmatrix}(-0.0333)^T=\begin{pmatrix}-0.0042\\-0.0017\\-0.0001\\-0.0003\end{pmatrix}.$$

$$\left(\frac{\partial F(W)}{\partial w_{ij}^1}\right)_{1\times 4} = -2x(B_1)^T = -2x\begin{pmatrix}-0.0167\\-0.0167\\-0.0167\\-0.0167\end{pmatrix} = 0,$$

再由(6.3.11)的学习规则得

$$(w_{12}, w_{13}, w_{14}, w_{15})^T = (0.2, 0.3, 0.4, 0.5)^T.$$

通过这一个样本的学习，再输入这个样本，则有

$$z_6 = 0.5(0.4833 + 0.1833 + 0.0833 + 0.3833) = 0.5666,\quad y_6 = 0.6380,$$

比原有的 0.6457 下降 5.3%。以表 6.3.1 的数据，按顺序依次进行学习，当神经网络采用(6.3.14)的学习规则，$\alpha_t = \varepsilon_t = 0.5$，表 6.3.1 的数据重复学习 2000 次，可以满足输出值同理想输出值的平方和均值不超过 0.001，终止学习。此时，再重新将表 4 的数据输入，其计算结果有如图 6.3.6 的示意图形。

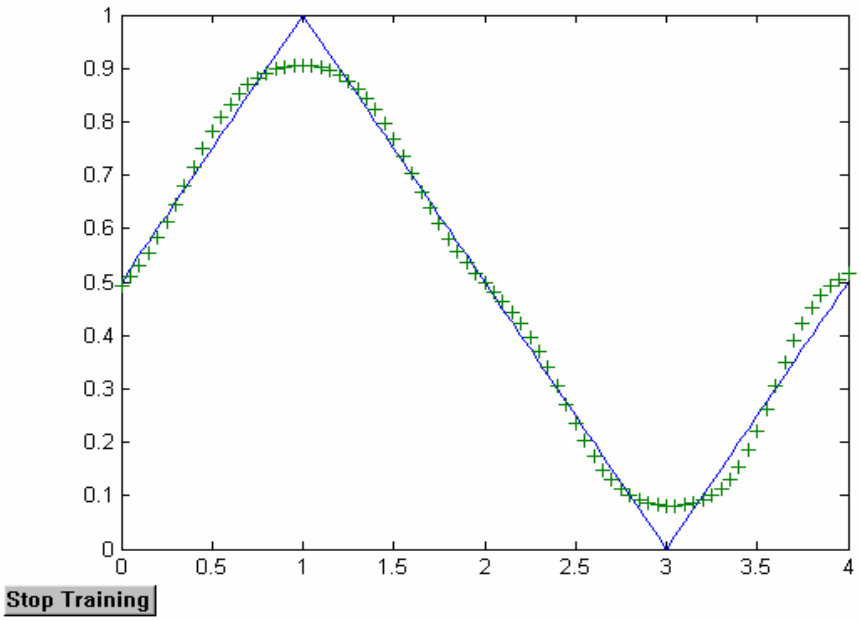


图 6.3.6 学习效果示意图

其中，实线图形为理想输出，‘+’图形为神经网络计算的结果。

前向多层神经网络的实际应用包括很多方面，如手写体的识别，语音识别，文本语音转换，图象识别，生物和医学信号处理等[10]。

6.4 竞争学习神经网络

竞争学习神经网络是无监督神经网络中的一类。给定输入数据，不同于有监督学习中需要对应的理想输出，无监督学习不需要对应的理想输出。也就是说，它的输出结果表明输入数据中的相关或相似性。

对于单层前向网络，一类简单的线性竞争学习模型为

$$y_j = \begin{cases} 1, & \sum_{i=1}^{m} w_{ij}x_i = \max\limits_{1\le k\le n}\left(\sum_{i=1}^{m} w_{ik}x_i\right), \\ 0, & \text{其他}. \end{cases}$$

当输入向量和$W^{(j)}=(w_{1j},w_{2j},\cdots,w_{mj})^T, j=1,2,\cdots,n$单位化以后，被选中神经元的权数向量$W^{(j)}$同输入向量 X 的欧氏空间距离最近。因此，它的学习规则是

$$w_{ij}(t+1)=w_{ij}(t)+\delta w_{ij}(t),$$

其中

$$\delta w_{ij}(t)=y_j(t)(x_i(t)-w_{ij}(t)).$$

竞争学习的最终效果使得相近的输入数据分为同一类。Kosko[11]用下面的一个简单神经网络例子“保持距离”来模拟分类问题。神经网络为图 6.4.1

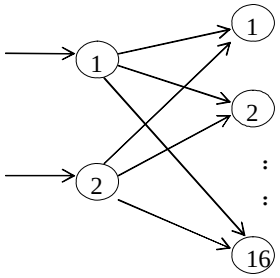


图 6.4.1 “保持距离”神经网络

输入(x_1,x_2)服从[－1,1]的均匀分布，初始权数$W^{(j)}=\begin{pmatrix}w_{1j}\\w_{2j}\end{pmatrix}$在[－1,1]×[－1,1]随机选取。由上面讨论的结果，$W^{(j)}$同输入向量 X 的欧氏空间距离越近，X 通过竞争被归为 j 类的可能性越大。通过 24982 次竞争学习后，竞争学习的模拟输出结果如示意图 6.4.2，其中黑团表示对应的$W^{(j)}$在以后竞争学习的变化区域。

删除的内容: 再

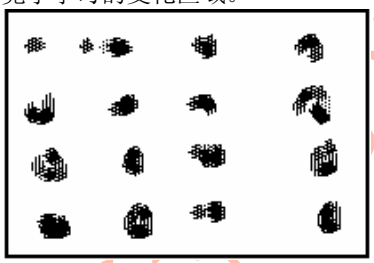


图 6.4.2 24982 次竞争学习的模拟示意图

无监督学习的另一类应用是根据自适应谐振理论（ART）构成的神经网络，关心的读者可以参考文献[10]。

6.5 反馈型神经网络

反馈型神经网络(feed-back neural networks)的一般结构如图 6.5.1，

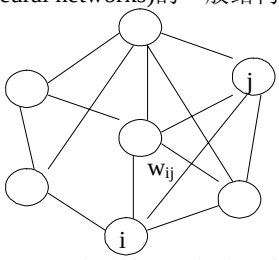


图 6.5.1 反馈型神经网络的一般结构

即神经元之间的信息传递关系不再是从一层到另一层，而是各神经元之间存在着联系。反馈型神经网络有连续和离散系统两类，主要取决于激活函数是采用连续函数还是阶跃函数。

6.5.1 连续系统神经网络

记 t 时刻第 i 个神经元接受到的信息量为 $z_i(t)$，信息量的变化为

$$\begin{aligned}&\frac{dz_i(t)}{dt}=-Az_i(t)+I_i(t)+\sum_{j=1}^{n}w_{ji}y_j(t),\\&y_i(t)=f_i(z_i(t)),\ i=1,2,\cdots,n,\end{aligned}\tag{6.5.1}$$

其中，$y_i(t)$ 为 t 时刻第 i 个神经元的输出；$I_i(t)$ 为 t 时刻外部传递给 i 神经元的信息；A 是一个正数，表示退化系数；w_{ij} 表示两个神经元 i 和 j 相连的权数。(6.5.1)是一个动力系统方程。连续系统神经网络要求(6.5.1)式中函数 $f_i(z_i)$ 有很好的光滑性。下面给出动力系统的一些基本术语和概念。

动力系统微分方程组可以简单记

$$\frac{dz(t)}{dt}=g(z,t),\tag{6.5.2}$$

其中，$z(t)=(z_1(t),z_2(t),\cdots,z_n(t))^T$。

定义 6.5.1 平衡点

如果

$$g(z_e,t)=0,\ \forall t,$$

则称 z_e 是动力系统的平衡点，也称 z_e 为吸引子。

定义 6.5.2 稳定性

满足下列条件的平衡点 z_e 称为稳定点：

任给 $\varepsilon>0$ 和 $t_0\geq 0$，存在 $\delta(\varepsilon,t_0)>0$，当 $t\geq t_0$，$\|z_0-z_e\|<\delta(\varepsilon,t_0)$ 时，有

$$\|z(t;z_0,t_0)-z_e\|<\varepsilon,$$

其中，$z(t;z_0,t_0)$ 表示以 z_0 为起始点，t_0 为起始时间满足(6.5.2)的一条参数曲线。称系统(6.5.2)在点 z_e 稳定。

定义 6.5.3 渐近稳定性

满足下列条件的稳定点 z_e 是渐近稳定点：

任给 $t_0\geq 0$，存在 $\eta(t_0)>0$，当 $\|z_0-z_e\|<\eta(t_0)$ 时，有

$$\lim_{t\to\infty}z(t;z_0,t_0)=z_e。$$

称系统(6.5.2)在点 z_e 渐近稳定。

定义 6.5.4 吸引域

满足下列条件的点 z_0 组成的集合称为平衡点 z_e 的吸引域：

存在 $t_0\geq 0$ 和 $\eta(t_0)>0$，当 $\|z_0-z_e\|<\eta(t_0)$ 时，有

$$\lim_{t\to\infty}z(t;z_0,t_0)=z_e。$$

研究反馈型神经网络的平衡点、平衡点的稳定性和吸引域等的性质的方法之一是研究动力系统和与它对应的 Lyapunov 能量函数的性质[5]。连续动力系统同离散动力系统有很多相似之处[12]，因此连续动力系统是反馈型神经网络研究的一个基础。从物理学的角度来看有这样的现象：随着系统的运动，其储存的能量随时间的增长而衰减，直至趋于能量极小的平衡状态，这类动力系统称为耗散系统。由此原理，反馈型神经网络的研究一般采用稳定性的 Lyapunov 第二方法，或称直接法，它首先给出微分方程和对应的能量函数或广义能量函数，或称 Lyapunov 函数，在研究函数的特性后，可以不用求解系统的运动方程，而给出系统平衡稳定性的信息。

Hopfield 利用模拟电路构造了反馈型人工神经网络的电路模型，建立的能量函数表达式为

$$E(y)=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}y_iy_j+\sum_{i=1}^{n}\frac{1}{R'_i}\int_0^{y_i}f_i^{-1}(y)dy-\sum_{i=1}^{n}I_iy_i,\qquad(6.5.3)$$

其中，$y_i=f_i(z_i)$，f_i 为 Sigmoid 或线性函数，w_{ij} 为电导参数，为 i 和 j 两个神经元的权数，具有对称性 $w_{ij}=w_{ji}$，R_i 是模拟电子线路电阻的大小，z_i 为第 i 个神经元的接受值，I_i 为外部偏置电流输入值，$\sum_{i=1}^{n}\frac{1}{R_i'}\int_0^{y_i}f_i^{-1}(y)dy$ 称为增益项。

(6.5.3)的能量函数对应一个动力系统

$$\begin{cases}\dfrac{dz_i}{dt}=-A_iz_i+\sum_{j=1}^{n}w_{ij}y_j+I_i,\\ y_i=f_i(z_i),i=1,2,\cdots,n,\end{cases}\qquad(6.5.4)$$

其中 $A_i=\frac{1}{R_i'}$，$\frac{1}{R_i'}=\frac{1}{R_i}+\sum_j w_{ij}$，$w_{ij}=w_{ji}$。

很容易验证，(6.5.3)同(6.5.4)有关系

$$-\frac{dz_i}{dt}=\frac{\partial E}{\partial y_i}。\qquad(6.5.5)$$

由(6.5.3)、(6.5.4)和(6.5.5)，得到

$$\begin{aligned}\frac{dE}{dt}&=\sum_{i=1}^{n}\frac{\partial E}{\partial y_i}\frac{dy_i}{dt}=-\sum_{i=1}^{n}\frac{dz_i}{dt}\frac{dy_i}{dt}=-\sum_{i=1}^{n}\frac{dz_i}{dy_i}\left(\frac{dy_i}{dt}\right)^2\\&=-\sum_{i=1}^{n}\frac{df_i^{-1}(y_i)}{dy_i}\left(\frac{dy_i}{dt}\right)^2\le 0.\end{aligned}\qquad(6.5.6)$$

特别当 $\frac{dz_i}{dt}=0(i=1,2,\cdots,n)$ 时，$\frac{dE}{dt}=0$。

定理 6.5.1[13] 若(6.5.3)是正定能量函数（即对 $\forall z\neq z_e$，有 $E(z)>0$），且满足 $\frac{dE(z)}{dt}\le 0$，则系统(6.5.4)的任何一个平衡点 z_e 是稳定的。

判别 Hopfield 神经网络稳定性经常采用的另一个结论是 Lasalle 不变原理。

定理 6.5.2[13] (Lasalle 不变原理) 若 S 是一个有界闭集，动力系统（6.5.2）满足：$z(t_0;z_0,t_0)\in S$，都有 $z(t;z_0,t_0)\in S,t\ge t_0$，且 S 上存在能量函数 $E(z)\in C^1$，使得 $\frac{dE(z)}{dt}\le 0,z\in S$，则 $z(t;z_0,t_0)\to Z_e=\{z\mid\frac{dE(z)}{dt}=0,z\in S\},t\to\infty$，当 Z_e 中只有一个点时，这一点是渐近稳定的。

Lyapunov 第二方法的基本思想是：先给一个动力系统，而后构造对应的 Lyapunov 函数，通过对 Lyapunov 函数的研究来确定动力系统的稳定性。一个系统的 Lyapunov 函数不具有唯一性。

Lyapunov 函数方法研究的是每一点及邻域的变化情况，除非函数有非常好的特性，一般情况下所研究的结论都是局部性的。

从(6.5.3)、(6.5.4)、(6.5.5)和(6.5.6)的推导看出，Hopfield 神经网络模型正是 Lyapunov 第二方法的应用。第一步是根据实际问题构造能量函数，然后利用电路系统的背景建立(6.5.4)的动力系统方程，其次再利用动力系统方程的计算，求解平衡点。(6.5.6)表明，随着时间的推移，能量函数逐渐下降。当(6.5.4)的系统方程组达到平衡点时，(6.5.6)也降到了一个能量函数的极小值点。于是 Hopfield 神经网络在组合优化问题中的计算归结为下面的计算步骤。

Hopfield 神经网络的计算步骤：

STEP1 针对实际的组合优化问题构造能量函数，使得能量函数有好的稳定性，如满足定理 6.5.1 或定理 6.5.2；

STEP2 由能量函数，根据(6.5.5)的关系求解出动力系统方程(6.5.4)；

STEP3 用数值计算的方法求解动力系统方程(6.5.4)的平衡点，具体的计算求解方法可通过自编程序或用现有的软件（如 Matlab），由定理 6.5.1 或定理 6.5.2 知平衡点是否为稳定点或渐近稳定，是否达到局部极小值。

Hopfield 通过一个电路找到了人工神经网络的微分方程组(6.5.4)和能量函数(6.5.3)的对应关系。寻找一个微分方程的 Lyapunov 能量函数并不容易，这也因此成为很多理论工作者研究的一个方向。对(6.5.4)的微分系统，当一些参数条件变化时，研究它的稳定性仍然对实际应用有重要的指导意义。

定理 6.5.1 的结论比较直观，只要对应的能量函数为正定，则任意一个平衡点是稳定的，为局部最优。定理 6.5.2 给出的是一个动力系统在局部区域 S 的渐近稳定性性质。表面上看，它比定理 6.5.1 更一般，但从理论上很难具体地给出 S 这个区域。从应用的角度来看，只要给出一个平衡点的微小扰动区域的任意一点为初始点，动力系统渐近收敛到这个平衡点，那么，可以认为这个点是渐近稳定的。实现的方法是通过数值计算。这样扩大了神经网络的应用范围。

文[14]从理论上研究了连接权不具有对称性和外加偏置电流 I_i 与时间参数 t 有周期输入的人工神经网络，得到该网络周期吸引子的存在性。

6.5.2 Hopfield 人工神经网络在 TSP 中的应用

为了方便神经网络的求解，用一个 $n\times n$ 矩阵表示 TSP 的一个解，第 i 行表示商人到达该城市的顺序，第 i 行由 0,1 数字组成一个 n 维向量，其中只能有一个分量为 1，1 所在的序数表示商人到达的序数。如向量(00010)表示商人第四个访问该城市。

例 6.5.1 四个城市 TSP 的一个解的矩阵表示为

	1	**2**	**3**	**4**
城市 1	0	1	0	0
城市 2	0	0	1	0
城市 3	1	0	0	0
城市 4	0	0	0	1

它表示商人行走的城市顺序为：3-1-2-4。□

从上例中可以看出，每一行、每一列的和各为 1。由此构造一个有 $n\times n$ 个神经元的神经网络。对应(i,j)位置的神经元，接收到的值为 z_{ij}，其输出值为 y_{ij}，激活函数采用 Sigmoid 函数。记两个城市 x 和 y 的距离是 d_{xy}，因为激活函数为 Sigmoid 函数，因此期望每一行的和为 1，也就是希望

$$E_1 = \sum_{u=1}^{n}\sum_{i=1}^{n}\sum_{j\neq i} y_{ui}y_{uj} \tag{6.5.7}$$

越小越好。(6.5.7)的理想状态 $E_1 = 0$ 的含义是每一行最多有一个 1，可以全为 0。同理要求神经元的每一列的关系式

$$E_2 = \sum_{i=1}^{n}\sum_{u=1}^{n}\sum_{v\neq u} y_{ui}y_{vi} \tag{6.5.8}$$

越小越好。

要保证每一行每一列正好有一个 1，理想状态是

$$E_3 = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} - n \right)^2 \tag{6.5.9}$$

为零。

当 E_1, E_2, E_3 三项都达到理想状态也只保证得到 TSP 的一个可行解，而 TSP 的一个重要目标是得到一条费用最小的最短路。于是，当 $d_{uv} = d_{vu}$ 时，考虑使得下列形式

$$E_4 = \sum_{u=1}^{n} \sum_{v \neq u} \sum_{i=1}^{n} d_{uv} y_{ui} (y_{vi-1} + y_{vi+1}), \tag{6.5.10}$$

最小，其中，$y_{u0} = y_{un}, y_{un+1} = y_{u1}$。由神经元的构造，两相邻的列表示商人行走的相邻城市，故 $d_{uv} y_{ui} y_{vi+1}$ 表示城市 u 和 v 之间的距离。由(6.5.10)是对称形式。

最后将能量函数写成

$$E = \frac{A}{2} E_1 + \frac{B}{2} E_2 + \frac{C}{2} E_3 + \frac{D}{2} E_4 + \alpha E_5, \tag{6.5.11}$$

A,B,C,D,α 为非负常数，表示 E_1, E_2, E_3, E_4, E_5 项对能量 E 的贡献大小。按(6.5.3)有

$$E_5 = \sum_{i,j} \int_0^{y_{ij}} f_{ij}^{-1}(y) dy \text{。}$$

由(6.5.5)的关系，

$$\begin{cases} \dfrac{dz_{ui}}{dt} = -\dfrac{\partial E}{\partial y_{ui}} \\ = -\alpha z_{ui} - A \sum_{j \approx i} y_{uj} - B \sum_{v \neq u} y_{vi} - C \left(\sum_{v=1}^{n} \sum_{j=1}^{n} y_{vj} - n \right) - D \sum_{v \neq u} d_{uv} (y_{vi+1} + y_{vi-1}), \\ y_{ui} = f(z_{ui}), \end{cases} \tag{6.5.11}$$

(6.5.11)的系数整理后

$$\begin{cases} w_{ui,vj} = -A\delta_{uv}(1 - \delta_{ij}) - B\delta_{ij}(1 - \delta_{uv}) - C - Dd_{uv}(\delta_{j,j+1} + \delta_{j,i-1}), \\ I_{ui} = nC, \end{cases} \tag{6.5.12}$$

其中，

$$\delta_{ij} = \begin{cases} 1, i = j, \\ 0, i \neq j, \end{cases} \qquad d_{uu} = 0 .$$

于是，(6.5.11)写成(6.5.4)的标准形式。

上面的讨论将求解 TSP 问题归结为 Hopfield 计算常规步骤。由(6.5.11)的构造、(6.5.6)的推导和定理 6.5.1，知(6.5.11)的平衡点是稳定点。存在的问题是参数 α, A, B, C, D 的选取。这些参数的选择是实际计算中的难点之一。D 对应 TSP 问题的目标函数项，应该突出，但同时希望罚值 A、B 和 C 尽可能的大以便求到可行解。实际计算中常常采取多组参数计算比较，从中选择好的解。下例是 Hopfield 给出的神经元初始值和参数选取的规则。

例 6.5.2 Hopfield[4]在求解 10 个城市的 TSP 问题时，数值计算的参数选取如下：

$$\alpha = 1, A = B = D = 500, C = 200 \text{。}$$

Sigmoid 函数为

$$y_{ui} = f(z_{ui}) = \frac{1}{1 + e^{-\frac{2z_{ui}}{\mu_0}}},$$

其中，$\mu_0 = 0.02$。

初始设置 y_{ui} 都相等，使得

$$\sum_{u=1}^{10}\sum_{i=1}^{10} y_{ui} = 10,$$

即期望最终的输出结果使得每一行和每一列只有一个输出非常接近 1。用初始相等的 y_{ui} 可以解出 $z_{00} = -\frac{\mu_0}{2}\ln 9$。给 z_{00} 一个扰动

$$z_{ui} = z_{00} + \delta z_{ui},\ -0.1\mu_0 \le \delta z_{ui} \le 0.1\mu_0,$$

其中，δz_{ui} 为 $[-0.1\mu_0, 0.1\mu_0]$ 均匀分布，为动力系统的初始点开始数值计算。□

Hopfield 的开创性工作是给出求解组合优化问题的一种神经网络方法，但其不足也是显而易见的。首先是参数 A,B,C,D,α 的选取，过多的参数增加了计算的工作量。第二是例 6.5.2 中参数 μ_0 的确定，μ_0 较小时，使得 S 形曲线有较好的识别力，输出值接近 0 或 1，但初始解的范围较小，因此计算收敛的速度慢。当参数 μ_0 较大时，又难以使输出值接近 0 或 1，使得路径的描述产生混淆。第三个不足是它的计算量和收敛的效果。由于它的能量函数(5.38)实际上是罚函数。罚函数势必造成计算量的增加。再者是 A,B,C,D,α 五个参数选取难以突出 TSP 的目标值，过多突出 D 会加大解的不可行性，加大其他参数会忽略目标值。

根据 Sigmoid 函数的特性，当 μ_0 较小时或 y_{ij} 接近 0 或 1 时，y_{ij} 的变化对(6.5.11)的总能量影响不是很大，于是常见作者省略 E_5 这一项，省略后(6.5.11)变为

$$E = \frac{A}{2}E_1 + \frac{B}{2}E_2 + \frac{C}{2}E_3 + \frac{D}{2}E_4. \tag{6.5.13}$$

用神经网络求解 TSP 问题还有其他形式的能量函数，如用 y_{ijk} =1 表示商人的第 k 步行走路线为城市 i 到城市 j，则能量函数：

$$E = \frac{A}{2}\sum_{j=1}^{n}\left(\sum_{i=1}^{n}\sum_{k=1}^{n} y_{ijk} - 1\right)^2 + \frac{B}{2}\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\sum_{k=1}^{n} y_{ijk} - 1\right)^2,$$
$$+\frac{C}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} y_{ijk} - n\right)^2 + \frac{D}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} d_{ij} y_{ijk}$$

其中上式右端第一项表示城市 j 恰好进去一次，右端第二项表示城市 i 恰好出去一次，第三项表示只有一条路径，第四项为行走路线最短。

6.5.3 离散系统神经网络

此处，离散系统神经网络主要指激活函数采用阶跃函数的反馈型神经网络。此时，输出值一般有关系：

$$Y(t+1) = g(Y(t), t),\ t \in I = \{i | i \ge t_0, i \text{ 为整数}\}, \tag{6.5.14}$$

其中，$Y(t) = (y_1(t), y_2(t), \cdots, y_n(t))^T$。采用阶跃函数的反馈神经网络为

$$\begin{cases} y_i(t+1) = f_i\left(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i\right), \\ i = 1, 2, \cdots, n. \end{cases} \tag{6.5.15}$$

其中，f_i 为阶跃函数。

连续动力系统中，定义 6.5.1 的平衡点 z_e 与参数 t 无关，也就使得输出 Y 与 t 无关。于是，在离散动力系统中，可以用输出 Y 的特性定义平衡点。

$$Y_e = g(Y_e, t), \forall t \in I,$$

则称 Y_e 为离散系统的平衡状态。

由于(6.5.15)是一个离散动力系统，系统的不同实施方法对应了不同的收敛性结论，也

对应了不同的算法，如同步和异步两种算法。所谓异步算法，是指每次只调整一个神经元，表达式为

$$\begin{cases} y_i(t+1)=\operatorname{sgn}(\sum_{j=1}^{n} w_{ij}y_j(t)-\theta_i), \\ y_j(t+1)=y_j(t), \quad j\neq i, \end{cases} \tag{6.5.16}$$

它的含义是：在一步迭代中，只有第 i 个神经元发生变化，而其他神经元保持不变。每次被调整的神经元 i 随机选定。

同步算法指同一时刻对所有神经元同时调整，表达式为

$$y_i(t+1)=\operatorname{sgn}(\sum_{j=1}^{n} w_{ij}y_j(t)-\theta_i), i=1,2,\cdots,n\text{。}$$

例 6.5.3 三个神经元的反馈网络的权为

$$W=\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{pmatrix},$$

在 t 时刻，一组输出 $Y(t)=(1,0,1)^T$，每个神经元的阈值都为 3.5。在第 t+1 时刻，同步算法使得每个神经元接受到的值为

$$Z(t+1)=WY(t)-\theta=(2,4,2)^T-(3.5,3.5,3.5)^T,$$

同步算法在 t+1 时刻的输出为

$$Y(t+1)=f(Z(t+1))=(0,1,0)^T\text{。}$$

异步算法的计算为：当 t+1 步迭代选择第一个神经元时，它的接收值为

$$z_1(t+1)=(0,1,2)Y(t)-\theta_1=-1.5,$$

所以，t+1 时刻个神经元的输出为

$$y_1(t+1)=0,\ y_2(t+1)=0,\ y_3(t+1)=1,$$
$$Y(t+1)=(0,0,1)^T\text{。}$$

若 t+2 步迭代选择第二个神经元变化时，

$$z_2(t+2)=(1,0,3)Y(t+1)-\theta_2=(1,0,3)\begin{pmatrix}0\\0\\1\end{pmatrix}-3.5=-0.5,$$

所以

$$y_1(t+2)=0,\ y_2(t+2)=0,\ y_3(t+2)=1,$$
$$Y(t+2)=(0,0,1)^T\text{。}$$

若 t+3 步迭代选择第三个神经元变化时，

$$z_2(t+3)=(2,3,0)Y(t+2)-\theta_3=(2,3,0)\begin{pmatrix}0\\0\\1\end{pmatrix}-3.5=-3.5,$$

所以

$$y_1(t+3)=0,\ y_2(t+3)=0,\ y_3(t+2)=0,$$
$$Y(t+3)=(0,0,0)^T\text{。}$$

虽说异步算法采用上例中相邻的三次迭代达到同步算法对三个神经元都进行了计算，但输出结果(0,0,0)同同步算法的输出结果(0,1,0)的差别是显而易见的。□

由(6.5.15)离散系统所具有的特性，输出的状态数最多为 2^n。连续系统的稳定性和渐进稳定性都同邻域相关，而离散系统则需要重新定义邻域，可以仿造第一章 1.3 节的定义给出

删除的内容: 3

邻域的定义，然后通过直接研究动力系统(6.5.15)来研究稳定性问题。

虽然，连续系统的 Lyapunov 方法研究离散动力系统稳定性的结论不再可用，可以沿用 Hopfield 求解连续神经网络的思想，利用离散动力系统来求解组合优化问题。基本思路为：将组合优化问题对应一个类似(6.5.3)的能量函数，建立一个（6.5.15）的离散动力系统，找出(6.5.15)的平衡点与能量函数极值点的关系。假设组合优化问题对应的能量函数为

$$E(y)=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}y_iy_j+\sum_{i=1}^{n}\theta_iy_i\ , \tag{6.5.17}$$

对应的离散动力系统为（6.5.15）。神经网络方法是通过求（6.5.15）的平衡点，以达到(6.5.17)的全局最优点。如果邻域定义为：

定义 6.5.5 $Y\in\{0,1\}^n$ 的邻域为

$$N(Y)=\{Y+\delta_ie_i|i=1,2,\cdots,n\},$$

其中，e_i 表示第 i 个分量为 1 其他分量为 0 的 n 维向量，

$$\delta_i=\begin{cases}-1, & 当y_i=1,\\ 1, & 当y_i=0,\end{cases}$$

若能量函数在 Y*满足

$$E(Y^*)\le E(Y),\quad \forall Y\in N(Y^*),$$

则称 Y*为邻域 $N(Y^*)$ 的能量极小点，简称能量极小点。

定理 6.5.3 若(6.5.17)中 W 为半正定对称矩阵且满足 $w_{ii}=0,i=1,2,\cdots,n$，则采用同步算法时，使得(6.5.17)随迭代时间能量值非增，且(6.5.15)的平衡点是(6.5.17)的能量极小点。

证明：记(6.5.17)

$$E(t)=-\frac{1}{2}Y^T(t)WY(t)+Y^T(t)\theta,$$

$$\Delta E(t)=E(t+1)-E(t),$$

$$\Delta Y(t)=Y(t+1)-Y(t)=(\Delta y_1(t),\Delta y_2(t),\cdots,\Delta y_n(t))^T,$$

于是，

$$\Delta E(t)=-\Delta Y^T(t)\big(WY(t)-\theta\big)-\frac{1}{2}\Delta Y^T(t)W\Delta Y(t)\ 。 \tag{6.5.18}$$

令

$$Z(t+1)=WY(t)-\theta=\big(z_1(t+1),z_2(t+1),\cdots,z_n(t+1)\big)^T\ 。$$

当 $z_i(t+1)\ge 0$ 时，

$$y_i(t+1)=f(z_i(t+1))=1,$$

得 $\Delta y_i(t)\ge 0$，推出 $\Delta y_i(t)z_i(t+1)\ge 0$。

当 $z_i(t+1)<0$ 时，

$$y_i(t+1)=f(z_i(t+1))=0,$$

得 $\Delta y_i(t)\le 0$，推出 $\Delta y_i(t)z_i(t+1)\ge 0$。

综合上面的讨论，

$$-\Delta Y^T(t)\big(WY(t)-\theta\big)\le 0\ 。$$

由定理中 W 为半正定的条件，当 $\Delta Y(t)\ne 0$ 时，$\Delta E(t)\le 0$。

因此，能量函数值随迭代时间非增。假设 $Y^*=(y_1^*,y_2^*,\cdots,y_n^*)^T$ 为(6.5.15)的平衡点，Y^* 邻域中的任何一点 $Y'=(y_1^*,\cdots,y_{i-1}^*,y_i',y_{i+1}^*,\cdots,y_n^*)^T$，其中 $|y_i^*-y_i'|=1$。因为 Y^* 为平衡点，当 $y_i^*=1$ 时，$z_i^*=\sum_{j=1}^{n}w_{ij}y_j^*-\theta_i\ge 0$，再由 (6.5.18)，

删除的内容: >

$\Delta E = E(Y') - E(Y^*) = \sum_{j\neq i} w_{ij} y_j^* - \theta_i \geq 0$；当 $y_i^* = 0$ 时，$z_i^* = \sum_{j=1}^{n} w_{ij} y_j^* - \theta_i < 0$，再由(6.5.18)，$\Delta E = -(\sum_{j\neq i} w_{ij} y_j^* - \theta_i) > 0$；所以，$Y^*$ 为极小点。□

由于极值点是一个局部性质，那么动力系统(6.5.15)的平衡点可能不是(6.5.17)的全局最优点。从定理 6.5.3 的证明可以看出，动力系统的每次迭代使得能量函数非增。这同邻域搜索算法类似，强烈地依赖初始点。同时，研究离散动力系统的平衡点同能量函数极值点的关系，完全取决于 W 和θ满足的条件。在以下的讨论中，特假定下列条件满足：W 是对称的 n 阶方阵，且 $w_{ii} \geq 0$；阶跃函数为

$$\mathrm{sgn}(x) = \begin{cases} 1, & x \geq 0, \\ 0, & \text{其它.} \end{cases}$$

Hopfield 神经网络异步算法的基本步骤为：

STEP1 任选一个初始状态 $Y(0) \in \{0,1\}^n$；

STEP2 随机选一个神经元，按(6.5.16)更新状态；

STEP3 检验 Y(t)是否为网络的平衡点，是转 STEP4；否转 STEP2;

STEP4 输出 Y(t).

定理 6.5.4 对任意的初始状态，满足上面假设的异步 Hopfield 算法最终收敛到一个平衡状态。

证明：按异步算法任选一个神经元 i，变化

$$\Delta Y(t) = (0,\cdots,0,\Delta y_i(t),0,\cdots,0)^T,$$

于是，(6.5.18)等价于

$$\begin{aligned} \Delta E(t) &= -\Delta y_i(t)\left[\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i\right] - \frac{1}{2} w_{ii} \Delta y_i^2(t) \\ &= -\Delta y_i(t)\left[z_i(t+1) + \frac{1}{2} w_{ii} \Delta y_i(t)\right]. \end{aligned} \tag{6.5.19}$$

按 $\Delta y_i(t)$ 的取值分情况讨论：

CASE1 $\Delta y_i(t)$ =0,即 $y_i(t+1) = y_i(t)$;

CASE2 $\Delta y_i(t)$ =1,即 $y_i(t+1) = 1, y_i(t) = 0$;

CASE3 $\Delta y_i(t)$ =－1,即 $y_i(t+1) = 0, y_i(t) = 1$。

下面按上面的三种情况分别讨论。

CASE1 $\Delta y_i(t)$ =0 推出 $\Delta E(t)$ =0。

CASE2 由 $\Delta y_i(t)$ =1,

$$y_i(t+1) = 1, y_i(t) = 0,\ y_i(t+1) = \mathrm{sgn}(z_i(t+1)),$$

得到

$$z_i(t+1) \geq 0\ 。$$

再由 $w_{ii} \geq 0$ 得 $\Delta E(t) \leq 0$。

CASE3 $\Delta y_i(t)$ =－1 时，$y_i(t+1) = 0, y_i(t) = 1$，由上面阶跃函数的假设，推出 $z_i(t+1) < 0$ 及 $\Delta E(t) < 0$。

采用异步算法，由 $E(t)$的非增性，经过一定的迭代后，将达到 E(t)不再下降。此时有 $\Delta E(t)$ =0。若 $\Delta y_i(t) = 0, \forall i$，则 Y(t)为一个平衡点，否则，由 CASE1 到 CASE3 的讨论，存在 i，使得 $\Delta y_i(t) = 1$ 且

删除的内容: 此处的阶跃函数有别于前面的函数定义，它把 x=0 的点赋值 1。

删除的内容: 新

$$z_i(t+1)+\frac{1}{2}w_{ii}\Delta y_i(t)=0。$$

由此而得到

$$z_i(t+1)=0, w_{ii}=0。$$

此时记

$$H=\{i \mid y_i(t+1)=1, y_i(t)=0, z_i(t+1)=0, w_{ii}=0\},$$

则对 H 中的任何一个神经元再迭代一次，就达到

$$y_i(t+1)=y_i(t+2)=1。$$

对于已经满足平衡的集合 $J=\{j|\Delta y_j(t)=0\}$，经过 t+1 次迭代后，再对 J 中的神经元 j 第 t+2 次迭代是否会出现不平衡？不平衡的情况为

I. $y_i(t)=0, y_i(t+1)=1$， $y_j(t)=y_j(t+1)=0, y_j(t+2)=1;$

II. $y_i(t)=0, y_i(t+1)=1$， $y_j(t)=y_j(t+1)=1, y_j(t+2)=0$。

由 CASE1、2、3 中的讨论，每次迭代中能量函数都是非增和能量函数对任何神经元迭代不再下降的假设，知 II 不可能发生。而 I 的情况又归于 H。因神经元个数有限，最后一定达到全部平衡。□

定理 6.5.5 对异步算法，若阶跃函数修改为：若 $z_i(t+1)=0$ 时，定义 $y_i(t+1)=y_i(t)$，则每个能量极小点为平衡点。

证明：记能量极小点为 Y*，记

$$z_i^*=\sum_{j=1}^{n}w_{ij}y_j^*-\theta_i,\quad y_i(2)=f(z_i^*),\quad \Delta y_i^*=y_i(2)-y_i^*。$$

若为平衡点，则结论成立。当某一个神经元 i 不平衡时，记神经元的能量变化为 ΔE_i，则由(6.5.19)有

$$\Delta E_i=-\Delta y_i^*\left[\sum_{j=1}^{n}w_{ij}y_j^*-\theta_i+\frac{1}{2}w_{ii}\Delta y_i^*\right]$$

$$=-\delta_i\left[z_i^*+\frac{1}{2}w_{ij}\delta_i\right],$$

其中，$\delta_i=\Delta y_i^*$。按定理 6.5.4 证明中讨论的三种情况和当条件 $z_i(t+1)=0$ 时，定义 $y_i(t+1)=y_i(t)$，不平衡的可能有

CASE1 $\delta_i=1$ 时，$y_i^*=0$，则

$$z_i^*+\frac{1}{2}w_{ii}\le 0,$$

由 $w_{ii}\ge 0$ 的条件，推出 $z_i^*\le 0$，进而推出

$$y_i(2)=f(z_i^*)=\begin{cases}0, & 当z_i^*<0,\\ y_i^*, & 当z_i^*=0.\end{cases}$$

于是，$y_i(2)=y_i^*$，与不平衡矛盾。

CASE2 $\delta_i=-1$ 时，$y_i^*=1$，则

$$z_i^*+\frac{1}{2}w_{ii}\delta_i\ge 0,\qquad z_i^*\ge\frac{1}{2}w_{ii}\ge 0,$$

进而推出

$$y_i(2)=f(z_i^*)=\begin{cases}1, & 当z_i^*>0,\\ y_i^*, & 当z_i^*=0.\end{cases}$$

于是， $y_i(2)=y_i^*$，与不平衡矛盾。

综合 CASE1 和 CASE2 的讨论，得到 Y*是一个平衡点。□

虽说能量函数的极小点是一个平衡点，但是否是一个稳定点？从极小点邻域中的一点出发，在什么样的条件下才能回到这个极小点？

定理 6.5.6 激活函数满足定理 6.5.5 的条件且权矩阵对角线满足 diag(W)=(0,0,…,0)时，异步算法网络的所有平衡点为能量函数极小点。

证明：记平衡点 $Y^*=(y_1^*,y_2^*,\cdots,y_n^*)^T$，从 Y*的邻域中任选一点，不妨设为第 i 个神经元变化，则由(6.5.19)能量变化为

$$\Delta E_i=-\delta_i\left[z_i^*+\frac{1}{2}w_{ii}\delta_i\right]=-\delta_i z_i^*。$$

对 Y*邻域中的任何一点 $Y'=(y_1^*,\cdots,y_{i-1}^*,y_i',y_{i+1}^*\cdots,y_n^*)^T$，当 $\delta_i=y_i'-y_i^*=1$时，$y_i^*=f(z_i^*)=0$，知 $z_i^*\le 0$。由此得到 $\Delta E_i\ge 0$。

当 $\delta_i=-1$时，$y_i^*=f(z_i^*)=1$，知 $z_i^*\ge 0$。由此得到 $\Delta E_i\ge 0$。

综合上面两点，平衡点为能量函数极小点。□

比较同步算法定理 6.5.3 与异步算法的定理 6.5.4、定理 6.5.5 和定理 6.5.6 的条件和结论，可以发现，激活函数和 W 的选取决定了离散动力系统是否收敛到平衡点，平衡点与能量函数极小值点的关系等。如定理 6.5.3 给出平衡点是一个能量函数极小值点，但没有证明动力系统一定收敛。定理 6.5.4、定理 6.5.5 和定理 6.5.6 则给出了收敛的条件、平衡点与能量极小点的关系。

正因为存在局部极小点，因此采用数值计算的方法就无法避免落入局部极小。于是相关的吸引子性质研究成为理论研究的热点。在算法的实现上，为了避免局部极小，产生同模拟退火、遗传算法等的结合。

6.5.4 变形算法——TSP 中的应用

变形算法(elastic net method)是反馈型神经网络在组合优化问题中的应用的一种，它来源于 Kohonen[15]的自组织影射模型。在这种算法中，组合优化问题的任一实例数据为网络的输入。在进化过程中，网络修改它的连接权。我们在此介绍变形模型的一个应用——TSP 中的弹性网络[16]——来理解变形算法的基本原理。

假设 TSP 的 n 城市位置为 $X_i=(x_{11},x_{12},\cdots,x_{1m})^T,i=1,2,\cdots,n$，其中 m 为空间的维数。在这个空间中给以 $N>n$ 个临时点 $Y_a=(y_{a1},y_{a2},\cdots,y_{am})^T,a=1,2,\cdots,N$。我们期望将临时点同城市位置匹配使得：每个城市仅同一个临时点匹配且这些临时点形成的闭路满足 $\sum_a|Y_a-Y_{a+1}|$尽量小，其中 $Y_{N+1}=Y_1$。当城市 i 同临时点 a 匹配，则定义一个变量 $s_{ia}=1$，否则 $s_{ia}=0$。于是可以给出下列的能量函数

$$E(s_{ia},Y_a)=\frac{1}{2T}\sum_{ia}s_{ia}|X_i-Y_a|^2+\frac{\gamma}{2}\sum_a|Y_a-Y_{a+1}|^2。\qquad(6.5.20)$$

(6.5.20)的第一项的功能是匹配，需要满足 $\sum_{ia}s_{ia}=n$。当已经匹配且 T 渐近零的时候，(6.5.20)的第二项功能是使得这些临时点形成的闭路尽量小且点的分布尽量均匀。由于 Y_a 和 s_{ia} 是变量和 s_{ia} 求解的难度，用另一个效用能量函数

$$E_{eff}(Y_a)=-T\sum_i\log_{10}\left(\sum_a e^{-|X_i-Y_a|^2/2T^2}\right)+\frac{\gamma}{2}\sum_a|Y_a-Y_{a+1}|^2,\qquad(6.5.21)$$

使其达到最小。

在(6.5.21)中，从右边的第一项看出，当每一个 X_i 都有一个临时点与其匹配时，

$$\sum_a e^{-|X_i-Y_a|^2/2T^2} = \sum_{a\in M} e^{-|X_i-Y_a|^2/2T^2} + \sum_{a\notin M} e^{-|X_i-Y_a|^2/2T^2} \to k, T\to 0,$$

其中，M 表示同 X_i 匹配的临时点集合，k 为同 X_i 匹配的临时点个数。此时，当 T 趋向零时，(6.5.21)右端第一项的值接近零。当存在一个 X_i，没有一个临时点与其匹配时，由 $\sum_a e^{-|X_i-Y_a|/2T^2} \to 0, T\to 0,$ 推出(6.5.21)右端第一项随着 T 的趋向零而增加，因此需要继续优化。(6.5.21)右端的第二项要求 N 个临时点形成的闭路尽量小和尽可能在闭路上均匀分布。(6.5.21)中只有 Y_a 为决策变量。能量函数的梯度为

$$\frac{\partial E_{eff}(Y_a)}{\partial Y_a} = -\sum_i v_{ia}(X_i - Y_a)/T + \gamma(2Y_a - Y_{a+1} - Y_{a-1}),$$

其中，

$$v_{ia} = \frac{\exp\{-\frac{|X_i - Y_a|^2}{2T^2}\}}{\sum_b \exp\{-\frac{|X_i - Y_b|^2}{2T^2}\}}。 \qquad (6.5.22)$$

$E_{eff}(Y_a)$ 对变量 Y_a 的下降梯度为

$$\Delta Y_a = -\eta \frac{\partial E_{eff}(Y_a)}{\partial Y_a}, \eta \ge 0 \text{的参数}。 \qquad (6.5.23)$$

由于 $E_{eff}(Y_a)$ 是一个非线性函数，且具有光滑性。采用梯度下降方向的迭代算法收敛到局部最优要求η满足一定的条件，在此不进一步讨论，关心这一部分内容的读者可参考非线性规划的有关论著，如[17，第二章]。具有变形算法在 TSP 应用——弹性算法的基本变化可以由图 6.5.2 描述。

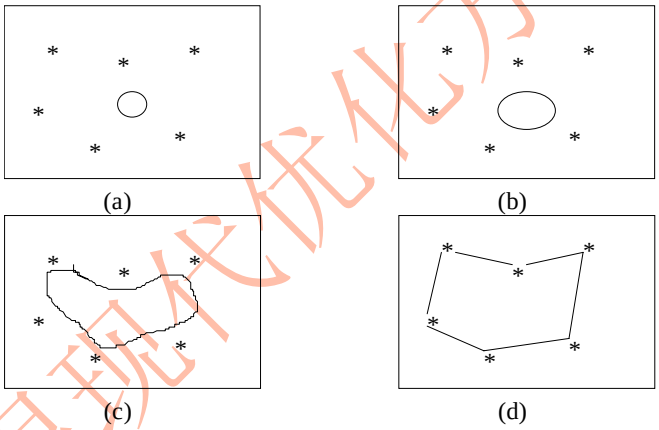


图 6.5.2 弹性算法的基本变化

图 6.5.2 描述了弹性算法计算时变化的四个阶段。图中“*”表示城市的位置。初始时在城市的重心位置的一个小圆周上，所有临时点均匀分布其上，如图 6.5.2(a)所示。经过若干次迭代，这些临时点向各城市靠近，就如同一个橡皮圈被拉向各城市一样，这种变化如图 6.5.2 中的(b)和(c)变化一样。最后，在理想情况下，每个城市都有临时点匹配且其他点按距离均匀分布在这些匹配点连成的线段上，就如同图 6.5.2(d)所示。

TSP 的弹性算法如下：

STEP1 给定 TSP 实例 n 个城市的坐标 $\{X_i | i = 1,2,\cdots,n\}$；

STEP2 选一个比 n 大的多的临时点数 N；

STEP3 选择合适的参数γ、初始温度 T_0 和初始步长参数 η_0；

STEP4 求 $\{X_i|i=1,2,\cdots,n\}$ 的重心；随机将 $Y_a=(y_{a1},y_{a2},\cdots,y_{am})^T, a=1,2,\cdots,N$ 均匀地布在以重心为圆心的一个小圆周上；

STEP5 在算法没有达到停止条件前，重复以下计算：

a. 更新临时坐标 Y_a 为

$$Y_a(T_l)=Y_a(T_{l-1})+\Delta Y_a(T_{l-1}),$$

其中，$\Delta Y_a(T_{l-1})$ 由(6.5.23)计算；

b. l:=l+1; $T_l=\alpha T_{l-1}(0<\alpha\le 1)$。

变形算法为什么可以认为是反馈型神经网络？第一是将 $\{X_i|i=1,2,\cdots,n\}$ 和临时点看成神经元。第二是因为 v_{ia}。它可以看成神经元 i 和 a 的权数，当问题的参数 $\{X_i|i=1,2,\cdots,n\}$ 输入后，这一权数由(5.51)随着 $Y_a=(y_{a1},y_{a2},\cdots,y_{am})^T, a=1,2,\cdots,N$ 的改进一步步地改进。

弹性网络法适合以欧氏距离为费用的 TSP 问题，Peterson[18]将这一算法应用到 TSP 并对大量典型数据进行计算比较，他发现算法的效率较高且临时点的个数也较小 N=2n。

变形算法的特点是它的思想，在此介绍这个算法的目的是希望读者能将这一种思想在其他实际问题中得以应用。

对人工神经网络的总体评价可以归为以下若干方面：第一，提供信息处理的一种新的手段。主要归功于人工神经网络的自适应性、学习能力和大规模的平行计算能力；第二，发展已较为成熟。成熟的三个方面是：（1）模型的建立和数学理论的支持，（2）计算工具的发展，（3）神经生物学的发展和对大脑的认识；第三，人工神经网络发展仍然受到限制。尽管近二十年计算机发展迅速，同时有大量的数学理论支持和发展，但计算机设备在储存、速度和用户柔性方面的现有能力，限制了人工神经网络的发展；第四，有成功应用的案例。人工神经网络在视觉、语言、信号处理和机器人等方面有成功应用的示例。虽说应用的范围有限，但其应用的问题及效果给人们的印象深刻。

人工神经网络的模型要求发展神经网络型计算系统来替代传统的计算机。这种计算系统不再是传统计算机顺序执行命令的运行过程，而是希望对输入进行平行处理；这种计算系统不再是只包含一个或几个复杂的计算设备，而是由众多简单设备有机组成在一起；这种计算系统处理信息时，不再是将信息存储在一个精确的位置上，而是通过神经元的内部相连关系达到信息储存的功能。

在目前的条件下，还只能依赖传统的计算机来模拟人工神经网络的这些功能。数值计算和分析是模拟中的一个重要部分。只有当人工神经网络硬件系统得到发展，才能使目前的人工神经网络更快、更有效地解决更大的实际问题。我们在后面各节中所介绍的模型就是基于这样的原理。

练习题

1.前向神经网络适合哪些组合优化问题？这些问题可否用反馈型神经网络处理？举例说明。

2.反馈型神经网络适合求解什么样的组合优化问题？它在哪些方面比前向型神经网络有优越性？举例说明。

3.按例 5.5 的各参数设定实现 Hopfield 神经网络算法在 TSP 的应用，是否能得到 Hopfield 等[4]相类似的结果？

4.比较 Hopfield 神经网络和变形算法在 TSP 的应用效果。

5.就你关心或实际遇到的组合优化问题用神经网络方法求解。试同其他诸如禁忌搜索、

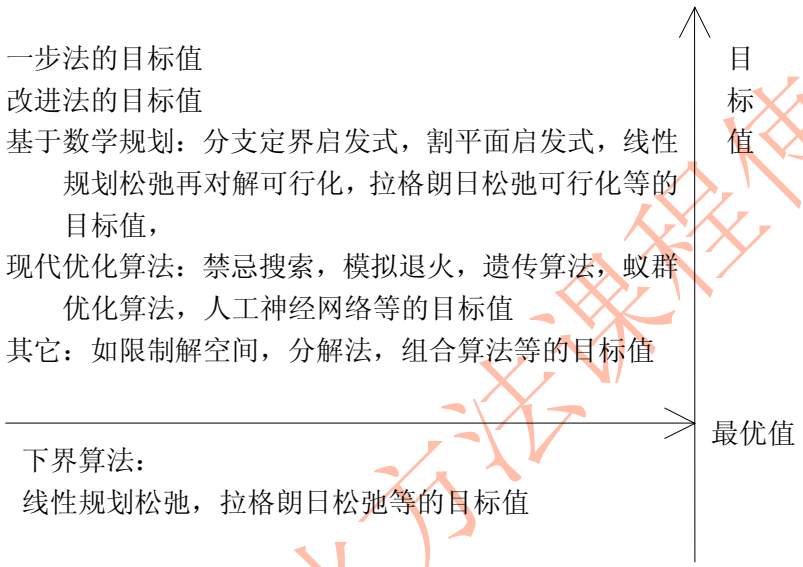
模拟退火、遗传算法等进行计算结果比较。

参考文献

1. James W. Psychology(Briefer Course). New York: Holt, 1890
2. McCulloch W, Pitts W. A logical calculus of the ideas imminent in nervous activity. Bulletin of Mathematical Biophysics, 1943, 5:115~133
3. Minsky M, Papert S. Perceptrons. Cambridge, MA: MIT Press, 1969
4. Hopfield J, Tank D. 'Neural' computation of decisions in optimization problems. Biological Cybernetics, 1985, 52:141~152
5. Hopfield J, Tank D. Computing with neural circuits: a model. Science, 1986, 233:625~633
6. McClelland J, Rumelhart D. Explorations in Parallel Distributed Processing. Cambridge, MA: MIT Press, 1988
7. Widrow B, Stearns S. Adaptive Signal Processing. Englewood: Prentice Hall, 1985
8. Rumelhart D, Hinton G E, Williams R J. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge: MIT Press, 1986
9. Wu B. An introduction to neural networks and their applications in manufacturing. Journal of Intelligent Manufacturing, 1992, 3:391~403
10. 杨行峻，郑君里. 人工神经网络. 高等教育出版社, 1992
11. Kosko B. Neural Networks and Fuzzy Systems. Englewood: Prentice Hall, 1992
12. Hopfield J. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Science, USA, 1984, 81:3088~3092
13. Lasalle J P. The stability of dynamical systems. Philadelphia, SIAM, 1976
14. 李铁成，王铎. 一类带有周期输入的人工神经网络的渐近性质. 高校应用数学学报，1997, 12(Ser A): 25~28
15. Kohonen T. Self-Organization and Associative Memory. Springer, 1984
16. Durbin R, Willshaw D. An analogue approach to the TSP using an elastic net method. Nature, 1987, 326:689~691
17. 袁亚湘. 非线性规划数值方法. 上海科学技术出版社, 1993
18. Peterson C. Parallel distributed approaches to combinatorial optimization problems--benchmark studies on TSP. Neural Computation, 1990, 2:261

第七章 拉格朗日松弛算法

当一个组合优化问题被判定为 NP 完全或 NP 难时，解决这个问题的常用方法是构造启发式算法，求尽量接近最优解的可行解。这些算法包括第二章至第六章的局部搜索算法、禁忌搜索、模拟退火、遗传算法、蚁群优化算法和人工神经网络等等。以极小优化目标函数为例，这些算法给出最优值的上界，第一章的 1.4 节给出这些算法的目标值同最优目标值关系的示意图如下：



评价算法好坏的一个标准是考察它所计算的目标值同最优目标值的差别。由于组合优化问题的难度，求解最优值有时是非常困难的。解决这个难点的一个有效方法是通过计算下界，用上界和下界的差来评价算法。拉格朗日(Lagrange)松弛算法就是求解下界的一种方法。由于拉格朗日松弛算法的实现比较简单和有比较好的性质，它不仅可以用来评价算法的效果，同时可以用在其他算法中，以提高算法的效率。拉格朗日松弛算法包含两部分内容：一方面是提供下界，另一方面则演变为拉格朗日松弛启发式算法。

本章 7.1 节介绍一些数学规划松弛的方法，7.2 节给出拉格朗日松弛的理论，7.3 节进一步讨论拉格朗日松弛的适用模型和更一般的结论，7.4 节讨论拉格朗日松弛算法，7.5 节给出一个应用案例——能力约束单机排序问题。

7.1 基于规划论的松弛方法

在此仅以整数规划为基础讨论，可在混合整数规划问题中作相应的讨论。整数规划的数学模型为

$$\begin{aligned} z_1 &= \min c^T x \\ s.t.\ & Ax \geq b \\ & x \in Z_+^n, \end{aligned} \tag{7.1.1}$$

其中，决策变量 x 为 n 维列向量，c 为 n 维列向量，A 为 $m \times n$ 矩阵，b 为 m 维列向量；系数 c，A 和 b 取整数，Z_+^n 表示非负整数集合。

1．线性规划松弛

在(7.1.1)中将整数变量约束松弛为实数，就可以得到

$$\begin{aligned} z_{LP} &= \min c^T x \\ s.t.\ Ax &\geq b \\ x &\in R_+^n . \end{aligned} \tag{7.1.2}$$

称(7.1.2)为(7.1.1)的线性规划松弛。线性规划松弛扩大了整数规划的可行解区域。若记

$$S = \left\{ x \in Z_+^n \mid Ax \geq b \right\},\quad S' = \left\{ x \in R_+^n \mid Ax \geq b \right\},$$

则有 $S \subseteq S'$，于是得到结论：

定理 7.1.1 $z_{LP} \leq z_1$。

定理 7.1.1 说明线性规划松弛得到整数规划的一个下界。可以通过单纯形算法或多项式时间的内点算法[1]，求得(7.1.2)的线性规划的最优解。

当 S 中的一个解 x_0 满足 $c^T x_0 = z_{LP}$ 时，推出 x_0 为(7.1.1)的最优解。作为求解整数规划问题启发式算法的一部分，线性规划松弛适用于整数规划问题中决策变量是比较大的整数。对这样的问题，启发式算法的两阶段为：第一阶段将整数规划问题松弛为线性规划问题，求解线性规划问题的最优解；第二阶段将线性规划的最优解按四舍五入或类似的原则整数化，同时考虑解的可行。

2．对偶规划松弛方法

线性规划(7.1.2)的对偶形式为

$$\begin{aligned} z_{DP} &= \max y^T b \\ s.t.\ A^T y &\leq c \\ y &\in R_+^n , \end{aligned} \tag{7.1.3}$$

其中，决策 y 是一个 m 维列向量。

(7.1.2)和(7.1.3)都为线性规划问题，它们的计算方法相同，且由对偶理论得到 $z_{DP} = z_{LP} \leq z_1$。至于采用(7.1.2)或(7.1.3)中哪一个求(7.1.1)的下界，需比较哪一个计算简单。无论如何，单纯形算法和内点算法在实际应用中都可能因为耗时过大而不能满足要求。如在一个循环计算的算法中，每一次循环需要求解一个线性规划问题，当循环的步数较大时，这样无论用单纯形算法还是内点算法都会感到计算时间过多，可能无法满足计算时间的要求。

3．代理(surrogate)松弛法

当(7.1.1)的约束过多时，代理松弛法是通过一个约束

$$\sum_{j=1}^{n} \left(\sum_{k=1}^{K} a_{i_k j} \right) x_j \geq \sum_{k=1}^{K} b_{i_k}$$

替代(7.1.1)中的 K 个约束

$$\sum_{j=1}^{n} a_{i_k j} x_j \geq b_{i_k}, k = 1,2,\cdots,K \text{。}$$

极端的情况可以用一个约束

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} \right) x_j \geq \sum_{i=1}^{m} b_i$$

松弛约束 $Ax \geq b$。代理松弛法保证目标函数、整数变量约束不变，且因约束的减少造成计算量的减少。

4．拉格朗日松弛方法

拉格朗日松弛方法的基本原理是：将造成问题难的约束吸收到目标函数中，并使得目标函数仍保持线性性，使得问题容易求解。产生对它的研究兴趣主要基于下面的原因：

第一，一些组合优化问题是 NP 难，除非 P=NP，否则在现有的约束条件下不存在求最优解的多项式时间算法。但在原有的问题中减少一些约束后，求解问题的难度就大大的减少，

使得减少一些约束后的问题在多项式时间内求得最优解。由此，将这些减少的约束称为难约束。对于线性整数规划问题，将难约束吸收到目标函数后，问题又变的容易求解。这时解的质量完全依赖于吸收到目标函数时所选取的参数。

例 7.1.1 集合覆盖问题(The set covering problem)

设 $A=(a_{ij})_{m\times n}$，所有元素 $a_{ij}\in\{0,1\}$，且每一列对应一个费用 $c_j(j=1,2,\cdots,n)$。$a_{ij}=1$ 表示第 j 列覆盖第 i 行。集合覆盖问题是以最小的费用选择一些列使得这些列覆盖所有的行。它的数学模型为

$$z_{SC}=\min\sum_{j=1}^{n}c_jx_j \tag{7.1.4}$$

$$(SC)\qquad s.t.\sum_{j=1}^{n}a_{ij}x_j\geq 1,\quad i=1,2,\cdots,m, \tag{7.1.5}$$

$$x_j\in\{0,1\},\quad j=1,2,\cdots,n. \tag{7.1.6}$$

集合覆盖问题是NP难[2]。若将(7.1.5)松弛，可得优化问题

$$z_{LRSC}(\lambda)=\min\sum_{j=1}^{n}c_jx_j+\sum_{i=1}^{m}\lambda_i(1-\sum_{j=1}^{n}a_{ij}x_j)$$

$$s.t.\qquad x_j\in\{0,1\},j=1,2,\cdots,n,$$

$$\lambda\geq 0.$$

记

$$d_j=c_j-\sum_{i=1}^{m}\lambda_ia_{ij},$$

则松弛后的模型为

$$z_{LRSC}(\lambda)=\min\sum_{j=1}^{n}d_jx_j+\sum_{i=1}^{m}\lambda_i$$

$$s.t.\qquad x_j\in\{0,1\},j=1,2,\cdots,n, \tag{7.1.7}$$

$$\lambda\geq 0.$$

(7.1.7)很容易求得最优解

$$x_j^*=\begin{cases}1, & 若d_j\leq 0,\\ 0, & 其它.\end{cases}$$

最优值为

$$z_{LRSC}(\lambda)=\sum_{j=1}^{n}d_jx_j^*+\sum_{i=1}^{m}\lambda_i.$$

从松弛和求解的过程中看出：对给定的 $\lambda\geq 0$，满足(7.1.5)和(7.1.6)的一个可行解自然满足(7.1.7)的约束，因此，$z_{LRSC}(\lambda)\leq z_{SC}$ 且 $z_{LRSC}(\lambda)$ 是 SC 问题的一个下界。若(7.1.7)的一个可行解不满足约束(7.1.5)时，即存在 i，使得 $\sum_{j=1}^{n}a_{ij}x_j<1$，可以通过调节 λ_i，使其增大而惩罚解的不可行性。于是 $z_{LRSC}(\lambda)$ 同 z_{SC} 的差距依赖于 $\lambda\geq 0$ 的选取。还可以看出松弛后的最优解非常容易得到，只需判别 d_j 的正负号。□

第二，实际的计算结果证实拉格朗日松弛方法所给的下界相当不错，且计算时间可以接受。同时，可以进一步利用拉格朗日松弛的基本原理构造基于拉格朗日松弛的启发式算法。

由上面四种松弛方法，可以给予松弛一个定义：

定义 7.6.1 问题

(RP) $$z_R = \min_{x \in S_R} z_R(x)$$

满足下列两条性质时，RP 称为整数规划(7.1.1)的一个松弛(relaxation)：

(1) 可行解区域兼容： $S \subseteq S_R$；

(2) 目标函数兼容： $cx \geq z_R(x),\ \forall x \in S$

其中，S_R 表示一个解集合，$z_R(x)$ 为实函数。

定理 7.1.2 若 RP 无可行解，则(7.1.1)也无可行解；若(7.1.1)有可行解，则 $z_1 \geq z_R$。

证明：当 RP 无可行解时，由可行解区域兼容性，$S = \varnothing$。当(7.1.1)可行时，(7.1.1)的最优解为 RP 的一个可行解，所以 $z_1 \geq z_R$。□

7.2 拉格朗日松弛理论

理论告示我们，如果一个整数规划问题可以在多项式时间内求得最优解，没有必要用更复杂的算法去求解。当面对一个 NP 难的整数规划问题时，除非 P=NP，构造多项式时间的最优算法已不可能。本章是在整数规划问题为 NP 难的前提下讨论它的松弛方法。为了适合拉格朗日松弛方法的讨论，将整数规划问题 IP 描述为

(IP)
$$\begin{aligned} z_{IP} &= \min c^T x \\ s.t.\ \ & Ax \geq b \quad (\text{复杂约束}) \\ & Bx \geq d \quad (\text{简单约束}) \\ & x \in Z_+^n, \end{aligned}$$

其中，(*A*,*b*)为 *m*×(*n*+1)整数矩阵，(*B*,*d*)为 *l*×(*n*+1)整数矩阵。记 IP 的可行解区域为

$$S = \{x \in Z_+^n \mid Ax \geq b, Bx \geq d\}。$$

在 IP 模型中，$Ax \geq b$ 为复杂约束的名称来自于：如果将该项约束去掉，则 IP 可以在多项式时间求到最优解，即假定

$$\begin{aligned} & \min c^T x \\ s.t.\ & Bx \geq d \quad (\text{简单约束}) \\ & x \in Z_+^n, \end{aligned} \tag{7.2.1}$$

可在多项式时间内求得最优解。

对给定的 $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^T \geq 0$，IP 对λ的拉格朗日松弛(在不对λ的取值产生混淆时，简称为 LR)定义为：

(LR)
$$\begin{aligned} z_{LR}(\lambda) &= \min c^T x + \lambda^T (b - Ax) \\ s.t.\ \ & Bx \geq d \\ & x \in Z_+^n. \end{aligned}$$

LR 的可行解区域记为 $S_{LR} = \{x \in Z_+^n \mid Bx \geq d\}$。

定理 7.2.1 LR 同(7.2.1)有相同的复杂性，且若 IP 的可行解区域非空，则

$$\forall \lambda \geq 0 \Rightarrow z_{LR}(\lambda) \leq z_{IP}。$$

证明：令

$$g(x, \lambda) = c^T x + \lambda^T (b - Ax),$$

则

$$g(x, \lambda) = (c^T - \lambda^T A)x + \lambda^T b$$

为 *x* 的线性函数。而 $\lambda^T b$ 为常数，又因它们的约束相同，故 LR 同(7.2.1)的复杂性相同。很明显看出 $S \subseteq S_{LR}$ 且

$$\forall \lambda \geq 0, x \in S \Rightarrow c^T x + \lambda^T (b - Ax) \leq c^T x \text{。}$$

由定理 7.1.2 得到 $\forall \lambda \geq 0 \Rightarrow z_{LR}(\lambda) \leq z_{IP}$。□

定理 7.2.1 说明拉格朗日松弛是 IP 的下界，我们的目的是求与 z_{IP} 最接近的下界。于是需要求解

(LD) $$z_{LD} = \max_{\lambda \geq 0} z_{LR}(\lambda) \text{。}$$

问题 LD 称为 IP 的拉格朗日对偶。用下例来理解拉格朗日松弛和对偶等概念。先定义凸集和凸包的概念。

定义 7.2.1 若 $\forall x, y \in D$，满足

$$\alpha x + (1-\alpha) y \in D, \quad 0 \leq \alpha \leq 1,$$

则称集合 D 为一个凸集。

对离散点集合 $Q = \{P_i | i = 1,2,\cdots\}$，它的凸包定义为

$$Con(Q) = \{P = \sum_i \alpha_i P_i | \alpha_i \geq 0 \text{实数}, \sum_i \alpha_i = 1\} \text{。}$$

很容易验证 *Con*(*Q*)为凸集。

例 7.2.1 假设整数规划问题 IP

$$\begin{aligned} z_{IP} = \min & -7x_1 - 2x_2 \\ s.t. \quad & x_1 - 2x_2 \geq -4 \\ & -5x_1 - x_2 \geq -20 \\ & 2x_1 + 2x_2 \geq 7 \\ & x_1 \geq 2 \\ & -x_2 \geq -4 \\ & x \in Z_+^2. \end{aligned} \tag{7.2.2}$$

的第一个约束为复杂约束，那么拉格朗日松弛后的模型 LR 为

$$\begin{aligned} z_{LR}(\lambda) = \min & [-(7+\lambda)x_1 - (2-2\lambda)x_2] - 4\lambda \\ s.t. \quad & -5x_1 - x_2 \geq -20 \quad (l1) \\ & 2x_1 + 2x_2 \geq 7 \quad (l2) \\ & x_1 \geq 2 \quad (l3) \\ & -x_2 \geq -4 \quad (l4) \\ & x \in Z_+^2. \end{aligned} \tag{7.2.3}$$

问题(7.2.3)可以用图解的方法简单求解。图解形式如示意图 7.2.1。

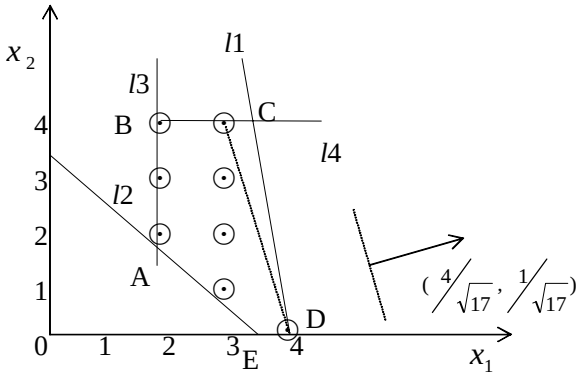


图 7.2.1 (7.2.3)的图解示意

其中，⊙表示整数点，ABCDE 分别表示(7.2.3)的可行解集的极点，图中的四条直线分别代表(7.2.3)四个约束取等号时的直线方程。

当 $\lambda = 0$ 时，目标函数的下降方向是 $(7,2)^T$，图解(7.2.3)的最优解为 $(3,4)^T$，

$z_{LR}(0)=-29$。

当 $\lambda=\frac{1}{2}$ 时，目标函数的下降方向是 $(7.5,1)^T$，图解(7.2.3)的最优解为 $(4,0)^T$，$z_{LR}(\frac{1}{2})=-32$。

当 $\lambda=1$ 时，目标函数的下降方向是 $(8,0)^T$，图解(7.2.3)的最优解为 $(4,0)^T$，$z_{LR}(1)=-32$。

由目标函数可知，$(7+\lambda,2-2\lambda)^T$ 为目标函数的下降方向。当 λ 在 $[0,+\infty)$ 变动时，单位化的方向从 $(\frac{7}{\sqrt{53}},\frac{2}{\sqrt{53}})^T$ 顺时针变化到

$$\lim_{\lambda\to\infty}(\frac{7+\lambda}{\sqrt{53+6\lambda+5\lambda^2}},\frac{2-2\lambda}{\sqrt{53+6\lambda+5\lambda^2}})^T=(\frac{1}{\sqrt{5}},\frac{-2}{\sqrt{5}})^T。$$

于是，只可能在两点 $(3,4)^T$ 和 $(4,0)^T$ 达到最优解。根据图 7.2.1 中的方向变化求得目标值。过 $(3,4)^T$ 和 $(4,0)^T$ 两点的直线方程为 $y+4x=16$，在图 7.2.1 中用虚线表示，它的一个垂直方向是 $\left(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}}\right)^T$。此时求得满足

$$(\frac{7+\lambda}{\sqrt{11+6\lambda+5\lambda^2}},\frac{2-2\lambda}{\sqrt{11+6\lambda+5\lambda^2}})^T=\left(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}}\right)^T,$$

的 $\lambda^*=\frac{1}{9}$。当 $0\le\lambda\le\lambda^*$ 时，最优解为 $(3,4)^T$。当 $\lambda^*\le\lambda$ 时，最优解为 $(4,0)^T$。综合得到

$$z_{LR}(\lambda)=\begin{cases}-29+\lambda, & 当0\le\lambda\le\frac{1}{9},\\ -28-8\lambda, & 当\lambda\ge\frac{1}{9}.\end{cases}\tag{7.2.4}$$

由(7.2.4)解出(7.2.2)的拉格朗日对偶问题目标值为

$$z_{LD}=z_{LR}(\frac{1}{9})=-28\frac{8}{9}。$$

记(7.2.2)的简单约束（即(7.2.3)的约束）的可行解集合为 Q，则

$$Q=\{(2,2)^T,(2,3)^T,(2,4)^T,(3,1)^T,(3,2)^T,(3,3)^T,(3,4)^T,(4,0)^T\}。$$

达到 $z_{LD}=-28\frac{8}{9}$ 的极点是 $\lambda=\frac{1}{9}$，$x=(4,0)^T$ 和 $\lambda=\frac{1}{9}$，$x=(3,4)^T$。同时，由图 7.2.1 直观看出，在 $(4,0)^T$ 和 $(3,4)^T$ 点连接直线上（图 7.2.1 中用虚线表示）的任何一点都有相同的目标值 $z_{LD}=-28\frac{8}{9}$。

考虑复杂约束 $x_1-2x_2\ge-4$，很容易验证：直线 $x_1-2x_2=-4$ 同过 $(4,0)^T$ 和 $(3,4)^T$ 两点的直线相交于 $x^*=(\frac{28}{9},\frac{32}{9})^T$。

综合上面的讨论，$z_{LD}=c^Tx^*$ 且 $x^*\in\{x\in Con(Q)|x_1-2x_2\ge-4\}$。□

下面定理说明例 7.2.1 最后一个结论有普遍性。

定理 7.2.2 若拉格朗日对偶的目标值 z_{LD} 有限，则

$$z_{LD}=\min\{c^Tx|Ax\ge b,x\in Con(Q)\},$$

其中

$$Q=\{x|Bx\ge d,x\in Z_+^n\}。$$

证明：

$$\begin{aligned} z_{LR}(\lambda) &= \min_{x\in Q}\left(c^T-\lambda^T A\right)x+\lambda^T b \\ &= \min_{x\in Con(Q)}\left(c^T-\lambda^T A\right)x+\lambda^T b \\ &= \min_{x\in Con(Q)}[c^T x+\lambda^T(b-Ax)]. \end{aligned}$$

设 $Con(Q)$的极点为 $\{x^k|k\in K\}$，极方向为 $\{r^j|j\in J\}$，则

$$\min_{x\in Q}(c^T-\lambda^T A)x+\lambda^T b=\begin{cases}-\infty, & \text{若存在}j\in J\text{满足}(c^T-\lambda^T A)r^j<0,\\ c^T x^k+\lambda^T(b-Ax^k), & \text{其它},k\in K.\end{cases}.$$

而由 z_{LD} 有限，则有

$$\begin{cases}\text{存在}\lambda\geq 0,\forall j\in J,\text{使得}(c^T-\lambda^T A)r^j\geq 0,\\ z_{LD}=\max\limits_{\lambda\geq 0}z_{LR}(\lambda)=\max\limits_{\lambda\geq 0}\min\limits_{k\in K}\left[c^T x^k+\lambda^T(b-Ax^k)\right].\end{cases}$$

这一结论等价为

$$\begin{aligned} z_{LD}&=\max\eta \\ s.t.\ \ & c^T x^k+\lambda^T(b-Ax^k)\geq\eta, \quad \forall k\in K \\ & (c^T-\lambda^T A)r^j\geq 0, \quad \forall j\in J \\ & \lambda\geq 0. \end{aligned}$$

整理得到

$$\begin{aligned} z_{LD}&=\max\eta \\ s.t.\ \ & \eta+\lambda^T(Ax^k-b)\leq c^T x^k, \quad \forall k\in K \\ & \lambda^T Ar^j\leq c^T r^j, \quad \forall j\in J \\ & \lambda\geq 0. \end{aligned}$$

由线性规划的对偶理论，上式的对偶线性规划为

$$\begin{aligned} z_{LD}&=\min c^T\left(\sum_{k\in K}\alpha_k x^k+\sum_{j\in J}\beta_j r^j\right) \\ s.t.\ \ & \sum_{k\in K}\alpha_k=1 \\ & A\left(\sum_{k\in K}\alpha_k x^k+\sum_{j\in J}\beta_j r^j\right)\geq b\left(\sum_{k\in K}\alpha_k\right) \\ & \alpha_k\geq 0,k\in K;\beta_j\geq 0,j\in J. \end{aligned}$$

最终整理得到

$$\begin{aligned} z_{LD}&=\min_{x\in Con(Q)}c^T x \\ s.t.\ \ & Ax\geq b \\ &=\min\left\{c^T x|Ax\geq b,x\in Con(Q)\right\}. \end{aligned}$$

□

推论 7.2.1 对任给 c，整数规划问题 IP 和拉格朗日对偶 LD 的目标值 $z_{IP}=z_{LD}$ 的充要条件是

$$Con\left(Q\cap\left\{x\in R_+^n|Ax\geq b\right\}\right)=Con(Q)\cap\left\{x\in R_+^n|Ax\geq b\right\}.$$

证明：很容易看出

$$Q\cap\left\{x\in R_+^n|Ax\geq b\right\}\subseteq Con(Q)\cap\left\{x\in R_+^n|Ax\geq b\right\},$$

得到

$$Con\left(Q\cap\left\{x\in R_+^n\mid Ax\geq b\right\}\right)\subseteq Con(Con(Q)\cap\left\{x\in R_+^n\mid Ax\geq b\right\})$$
$$=Con(Q)\cap\left\{x\in R_+^n\mid Ax\geq b\right\}.$$

再由定理 7.2.2，得到

$$z_{IP}=\min_{x\in Con\left(Q\cap\left\{x\in R_+^n\mid Ax\geq b\right\}\right)}c^Tx\leq z_{LD}=\min_{x\in Con(Q)\cap\left\{x\in R_+^n\mid Ax\geq b\right\}}c^Tx\ 。$$

若对任给的 c 满足：$z_{IP}=z_{LD}$，则得到

$$Con\left(Q\cap\left\{x\in R_+^n\mid Ax\geq b\right\}\right)=Con(Q)\cap\left\{x\in R_+^n\mid Ax\geq b\right\}\ 。$$

由此证明了必要性。由上面的推导，充分性的证明明显成立。□

例 7.2.2(续例 7.2.1) 例 7.2.1 中的

$Q=\{(2,2)^T,(2,3)^T,(2,4)^T,(3,1)^T,(3,2)^T,(3,3)^T,(3,4)^T,(4,0)^T\}$。

记

$$S_1=Con(Q\cap\{x\in R_+^n\mid x_1-2x_2\geq -4\}),$$
$$S_2=Con(Q)\cap\{x\in R_+^n\mid x_1-2x_2\geq -4\}。$$

它们的区域图形表示见图 7.2.2。S_1区域为图 7.2.2(a)的虚线所围部分。S_2区域为图 7.2.2(b)的黑粗线所围部分。

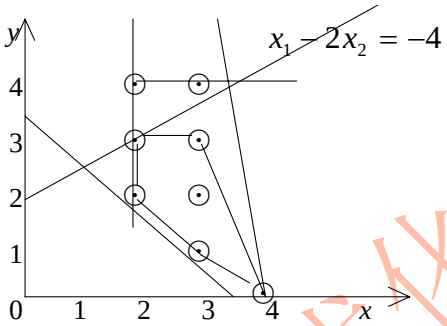


图 7.2.2(a) S_1区域

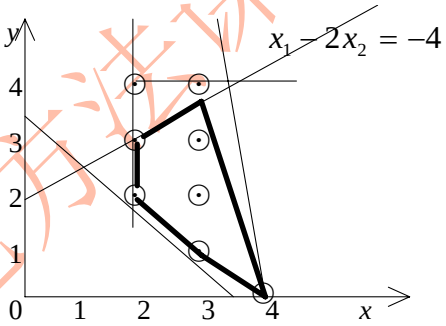


图 7.2.2(b) S_2区域

由推论 7.2.1 可以看出，$z_{IP}-z_{LD}$ 的差值同下面两个因素有关。第一是目标函数中的 c。推论 7.2.1 中的结论是对所有的 c 讨论得到的，但可能存在一个 c 使得 $z_{IP}=z_{LD}$，例 7.2.2 中 c^T=(1,0)时，则，

$$z_{IP}=\min x_1$$
$$s.t.\quad x_1-2x_2\geq -4$$
$$-5x_1-x_2\geq -20$$
$$2x_1+2x_2\geq 7$$
$$x_1\geq 2$$
$$-x_2\geq -4$$
$$x\in Z_+^2.$$

$$z_{LR}(\lambda)=\min[(1-\lambda)x_1+2\lambda x_2]-4\lambda$$
$$s.t.\quad -5x_1-x_2\geq -20$$
$$2x_1+2x_2\geq 7$$
$$x_1\geq 2$$
$$-x_2\geq -4$$
$$x\in Z_+^2.$$

解出 $z_{IP}=2$。$(2,2)^T$和$(2,3)^T$两点的直线方程为 $x=2$，它的一个垂直方向是 $(-1,0)^T$。此时，求满足

$$\left(\frac{\lambda-1}{\sqrt{1-2\lambda+5\lambda^2}},\frac{-2\lambda}{\sqrt{1-2\lambda+5\lambda^2}}\right)^T=(-1,0)^T,$$

的 $\lambda^*=0$。当 $\lambda=\lambda^*=0$ 时，松弛的一个最优解为$(2,2)^T$。于是 $z_{LR}(\lambda^*)=2$。

第二个因素是可行解的区域。由图 7.2.2(a)和图 7.2.2(b)中 S_1 区域和 S_2 区域的不同，存在c使得 $z_{IP}-z_{LD}$ 不为零。例 7.2.1 中，$z_{LD}=-28\frac{8}{9}$，在 $\lambda^*=\frac{1}{9}$ 达到拉格朗日对偶问题的最优值，它的一个最优解是$(4,0)^T$；$z_{IP}=-28$，一个最优解也为$(4,0)^T$。从这个结论可以看出，即使拉格朗日松弛在某一个 λ 求到的最优解为原问题(7.2.2)的可行解，我们不能断言 $z_{IP}=z_{LR}(\lambda)$。□

如果拉格朗日对偶在 $\lambda=0$ 时达到最优值，且它的最优解是 IP 的一个可行解，此时，原问题 IP 同松弛问题 LR 为同一个目标函数，所以有 $z_{IP}=z_{LR}(\lambda)$。

IP 的线性规划松弛为

$$\text{(LP)}\qquad \begin{aligned} z_{LP}&=\min c^Tx\\ s.t.\ \ &Ax\ge b\ \ (\text{复杂约束})\\ &Bx\ge d\ \ (\text{简单约束})\\ &x\in R_+^n. \end{aligned}$$

继续研究例 7.2.1 还可以发现：已经得到 $z_{IP}=-28$，$z_{LD}=-28\frac{8}{9}$。再用图解法，可得到线性规划松弛的最优解为$\left(\frac{36}{11},\frac{40}{11}\right)^T$，最优值为 $z_{LP}=-30\frac{2}{11}$。

定理 7.2.3 若线性规划松弛 LP 存在可行解，则 $z_{LP}\le z_{LD}\le z_{IP}$。

证明：因为

$$\begin{aligned} Q\cap\left\{x\in R_+^n\mid Ax\ge b\right\}&\subseteq Con(Q)\cap\left\{x\in R_+^n\mid Ax\ge b\right\}\\ &\subseteq\left\{x\in R_+^n\mid Bx\ge d\right\}\cap\left\{x\in R_+^n\mid Ax\ge b\right\}, \end{aligned}$$

得到

$$\begin{aligned} Con(Q\cap\left\{x\in R_+^n\mid Ax\ge b\right\})&\subseteq Con(Q)\cap\left\{x\in R_+^n\mid Ax\ge b\right\}\\ &\subseteq\left\{x\in R_+^n\mid Bx\ge d\right\}\cap\left\{x\in R_+^n\mid Ax\ge b\right\}, \end{aligned}$$

再由定理 7.2.2，推出

$$z_{IP}\ge z_{LD}\ge z_{LP}。\ \square$$

由定理 7.2.3 得到，采用拉格朗日松弛对偶后的目标值 z_{LD} 是 IP 的一个下界且不比 z_{LP} 差，因此可能出现差距 $z_{LD}-z_{LP}$。这解释了例 7.2.2 结束时给出的直观结论。

推论 7.2.2 若 $\left\{x\in R_+^n\mid Bx\ge d\right\}$ 的所有极点为整数点，则对任意的 c 有 $z_{LD}=z_{LP}$。

证明：从定理 7.2.3 的证明过程中得到

$$Con(Q)\cap\left\{x\in R_+^n\mid Ax\ge b\right\}\subseteq\left\{x\in R_+^n\mid Bx\ge d\right\}\cap\left\{x\in R_+^n\mid Ax\ge b\right\},$$

当 $\left\{x\in R_+^n\mid Bx\ge d\right\}$ 的所有极点为整数点时，

$$\left\{x\in R_+^n\mid Bx\ge d\right\}=Con(Q),$$

于是

$$Con(Q)\cap\{x\in R_+^n \mid Ax\geq b\}=\{x\in R_+^n \mid Bx\geq d\}\cap\{x\in R_+^n \mid Ax\geq b\},$$

再由定理 7.2.2 得到结论。□

定理 7.2.3 告诉我们，拉格朗日对偶的目标值是 IP 问题的一个下界，那么，这个下界与 IP 的目标值有多大的差距？

定理 7.2.4 $z_{IP}-z_{LD}\leq\varepsilon(\varepsilon\geq 0)$ 的充分必要条件是存在

$$\lambda^*\geq 0 \text{ 和 } x^*\in\{x\in Z_+^n \mid Ax\geq b, Bx\geq d\}$$

满足

$$\lambda^{*T}(b-Ax^*)\geq -\delta_1(\delta_1\geq 0),$$

$$z(\lambda^*,x^*)=c^Tx^*+\lambda^{*T}(b-Ax^*)\leq z_{LR}(\lambda^*)+\delta_2(\delta_2\geq 0),$$

$$\delta_1+\delta_2\leq\varepsilon。$$

证明：充分性：

$$z_{LD}\geq z_{LR}(\lambda^*)\geq z(\lambda^*,x^*)-\delta_2$$

$$\geq cx^*-\delta_1-\delta_2\geq z_{IP}-\delta_1-\delta_2.$$

必要性：

记 x^* 为 IP 的最优解，λ^* 为 LD 的最优解，则有

$$z_{LD}=z_{LR}(\lambda^*)=c^Tx^*+\lambda^{*T}(b-Ax^*)+z_{LR}(\lambda^*)-z(\lambda^*,x^*)$$

$$=z_{IP}+\lambda^{*T}(b-Ax^*)+z_{LR}(\lambda^*)-z(\lambda^*,x^*).$$

由条件 $z_{IP}-z_{LD}\leq\varepsilon(\varepsilon\geq 0)$ 推出

$$\lambda^{*T}(b-Ax^*)+z_{LR}(\lambda^*)-z(\lambda^*,x^*)\geq-\varepsilon,$$

记

$$\lambda^{*T}(b-Ax^*)=-\delta_1,\ z_{LR}(\lambda^*)-z(\lambda^*,x^*)=-\delta_2,$$

则得到

$$\lambda^{*T}(b-Ax^*)=-\delta_1(\delta_1\geq 0),$$

$$z(\lambda^*,x^*)=c^Tx^*+\lambda^{*T}(b-Ax^*)=z_{LR}(\lambda^*)+\delta_2(\delta_2\geq 0),$$

$$\delta_1+\delta_2\leq\varepsilon。□$$

可以用定理 7.2.4 的充分条件来估计一个算法同拉格朗日松弛下界的距离。

例 7.2.3(续例 7.2.1) 当 $\lambda^*=\frac{1}{9}$ 时，$x^*=(4,0)^T$ 为(7.2.2)问题的一个可行解，此时，

$$\lambda^*(b-Ax^*)=\frac{1}{9}(-4-4)=-\delta_1,$$

$$z(\lambda^*,x^*)=c^Tx^*+\lambda^*(b-Ax^*)=-28-\frac{8}{9}=-28\frac{8}{9}=z_{LR}(\lambda^*)+\delta_2,$$

其中，$\delta_2=0$。于是 $\varepsilon=\delta_1+\delta_2=\frac{8}{9}$，故有 $z_{IP}-z_{LD}\leq\frac{8}{9}$。

一般的情况下，无法估计的那么精确，但也可以用定理 7.2.4 的充分条件估计 $z_{IP}-z_{LD}$。如 $\lambda^*=\frac{1}{2}$ 时，$x^*=(4,0)^T$ 为(7.2.2)问题的一个可行解，此时，

$$\lambda^*(b-Ax^*)=\frac{1}{2}(-4-4)=-\delta_1,$$

$$z(\lambda^*,x^*)=c^Tx^*+\lambda^*(b-Ax^*)=-28-4=-32=z_{LR}(\lambda^*)+\delta_2,$$

其中，$\delta_2=-32-z_{LR}(\lambda^*)=-32+28+4=0$。于是 $\varepsilon=\delta_1+\delta_2=4$，故有 $z_{IP}-z_{LD}\leq 4$。

□

从上面的讨论看出，只有给出较好的 λ^* 和 x^*，才能得到 $z_{IP}-z_{LD}$ 较好的估计值。在有多种约束组合可松弛的拉格朗日松弛问题中，我们的目的是选使 $z_{IP}-z_{LD}$ 值最小的松弛。

7.3 拉格朗日松弛的进一步讨论

在前二节中，我们仅就标准的拉格朗日松弛进行了讨论。在实际应用中还可能出现其他的形式。本节针对一些常见的问题分类讨论。

1．等号约束的松弛

将等号约束

$$\sum_{j=1}^{n} a_{ij}x_j = b_i,$$

写成两个标准形式

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \quad 和 \quad \sum_{j=1}^{n}(-a_{ij})x_j \geq -b_i。$$

对应拉格朗日乘子 $\lambda_{i1}\geq 0, \lambda_{i2}\geq 0$。在目标函数中出现

$$\lambda_{i1}(b_i-\sum_{j=1}^{n}a_{ij}x_j)+\lambda_{i2}(-b_i-\sum_{j=1}^{n}(-a_{ij})x_j)=(\lambda_{i1}-\lambda_{i2})(b_i-\sum_{j=1}^{n}a_{ij}x_j)。$$

令 $\lambda_i=\lambda_{i1}-\lambda_{i2}$，则松弛后对应的 λ_i 没有正负号约束。

2．LR 最优解同 LP 最优解的关系

若对给定的 $\lambda\geq 0$，

$$\text{(LR)} \qquad \begin{aligned} z_{LR}(\lambda) &= \min c^T x+\lambda^T(b-Ax) \\ s.t.\quad & Bx\geq d \\ & x\in Z_+^n. \end{aligned}$$

的最优解 $x(\lambda)$ 为 IP 的一个可行解，则有 $z_{LR}(\lambda)\leq z_{IP}$。值得注意的是并不能得到 $c^T x(\lambda)=z_{IP}$，用下例说明这个结论。

例 7.3.1 集合覆盖问题

$$\begin{aligned} \min\ & 2x_1+3x_2+4x_3+5x_4 \\ s.t.\ & x_1+\quad\ x_3\quad\ \geq 1 \\ & x_1+\quad\quad\ x_4\geq 1 \\ & x_2+\ x_3+\ x_4\geq 1 \\ & x_i\in\{0,1\}, i=1,2,3,4, \end{aligned} \tag{7.3.1}$$

直观的结果是最优解为 $x_1=x_2=1, x_3=x_4=0, z_{IP}=5$。拉格朗日松弛三个约束，

$$\begin{aligned} z_{LR}(\lambda)=&\min(2-\lambda_1-\lambda_2)x_1+(3-\lambda_3)x_2 \\ &+(4-\lambda_1-\lambda_3)x_3+(5-\lambda_2-\lambda_3)x_4+\lambda_1+\lambda_2+\lambda_3 \\ s.t.&\ x_j\in\{0,1\}, j=1,2,3,4. \end{aligned} \tag{7.3.2}$$

当 $\lambda_1=\lambda_2=\lambda_3=4$ 时，

$$\begin{aligned} z_{LR}(\lambda)&=\min -6x_1-x_2-4x_3-3x_4+12 \\ s.t.&\ x_j\in\{0,1\}, j=1,2,3,4, \end{aligned}$$

其最优解为：$x_1=x_2=x_3=x_4=1$, $z_{LR}(\lambda)=-2$。该解为(7.3.1)问题的一个可行解，但

$$c^T x = (1,2,3,4)\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = 10 \neq z_{IP}。$$

由此得到：当松弛后问题的一个最优解为原问题(7.3.1)的一个可行解时，并不能得到该解为原问题的最优解。□

当拉格朗日松弛后的一个最优解为原问题 IP 的一个可行解时，在什么样的条件下，该解为 IP 的一个最优解？

定理 7.3.1 $z_{IP} = z_{LD}$ 的充分必要条件是存在 $\lambda^* \geq 0$ 和 x^* 为 IP 的可行解，满足 $\lambda^{*T}(b - Ax^*) = 0$ 且 $z_{LR}(\lambda^*) = z(\lambda^*, x^*)$。

证明：定理 7.2.4 的直接推论。□

当拉格朗日松弛后对应 $\lambda^* \geq 0$ 的一个最优解 x^* 为原问题 IP 的一个可行解时，此时已有 $z_{LR}(\lambda^*) = z(\lambda^*, x^*)$，只需 $\lambda^{*T}(b - Ax^*) = 0$ 时，有 $z_{IP} = z_{LD}$。这是对上面问题的回答。

3．拉格朗日松弛的整数性

定义 7.3.1 若 LR($\forall \lambda \geq 0$)的最优解与其整数约束 $x \in Z_+^n$ 无关，即

$$z_{LR}(\lambda) = \min c^T x + \lambda^T (b - Ax)$$
$$s.t. \quad Bx \geq d$$
$$x \in Z_+^n,$$

与 LR 的线性松弛

$$z_{LRL}(\lambda) = \min c^T x + \lambda^T (b - Ax)$$
(LRL) $$s.t. \quad Bx \geq d$$
$$x \in R_+^n,$$

的最优值相同，则称该问题的拉格朗日松弛具有整数性。

从整数性可以看出，若 LR 具有整数性，则对任意的 $\lambda \geq 0$，最优解在 $\left\{x \in R_+^n \mid Bx \geq d\right\}$ 的一个极点达到且该极点一定是整数点。

例 7.3.2(续例 7.1.1) 例 7.1.1 的集合覆盖问题 SC 的拉格朗日松弛为

(LRSC) $$z_{LRSC}(\lambda) = \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i$$
$$s.t. \quad x_j \in \{0,1\}, j = 1,2,\cdots,n,$$

其中

$$d_j = c_j - \sum_{i=1}^{m} \lambda_i a_{ij}。$$

上面公式的线性规划模型为

(LSC) $$z_{LSC}(\lambda) = \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i$$
$$s.t. \quad 0 \leq x_j \leq 1, j = 1,2,\cdots,n.$$

LRSC 和 LSC 的最优解都为

$$x_j^* = \begin{cases} 1, & 若 d_j \leq 0, \\ 0, & 其它. \end{cases}$$

由此可知 LRSC 具有整数性。□

定理 7.3.2 若 LR 具有整数性，则 $z_{LD} = z_{LP}$。

证明：记

$$Q' = \left\{ x \in R_+^n \mid Bx \geq d \right\},$$

$$z_{LRL}(\lambda) = \min_{x \in Q'} c^T x + \lambda^T (b - Ax),$$

$$z'_{LD} = \max_{\lambda \geq 0} z_{LRL}(\lambda)。$$

同定理 7.2.2 的证明相同，有

$$z'_{LD} = \min\left\{ c^T x \mid Ax \geq b, x \in Con(Q') \right\} = \min\left\{ c^T x \mid Ax \geq b, x \in Q' \right\} = z_{LP}。$$

由 $z_{LRL}(\lambda)$ 定义，存在 $\lambda^* \geq 0$ 使

$$z_{LP} = z'_{LD} = z_{LRL}(\lambda^*)。$$

由整数性得到：

$$z_{LP} = z_{LRL}(\lambda^*) = z_{LR}(\lambda^*) \leq z_{LD}。$$

再由定理 7.2.3 得到 $z_{LP} = z_{LD}$。□

定理 7.3.2 同推论 7.2.2 的结论极为相似，值得注意的是推论 7.2.2 要求 $Q' = \left\{ x \in R_+^n \mid Bx \geq d \right\}$ 的所有极点为整数点，这样可以保证 $z_{LP} = z_{LD}$，说明拉格朗日对偶问题的最优值与 IP 问题的线性规划松弛最优值无差距。但定理 7.3.2 的条件比推论 7.2.2 更弱一些，并不要求 $Q' = \left\{ x \in R_+^n \mid Bx \geq d \right\}$ 的所有极点为整数点，只需对应任一λ，LRL 的最优解为整数点，达到最优目标值 $z_{LRL}(\lambda)$，可从例 7.3.3 中看出。由此得到的结论是 $z_{LP} = z_{LD}$，表示线性规划松弛的最优值同拉格朗日对偶的最优值相同，即采用拉格朗日松弛的效果不比线性规划松弛好。

例 7.3.3 整数规划问题和它的拉格朗日松弛分别为(7.3.3)和(7.3.4)。

$$\begin{aligned} z_{IP} = \min & -7x_1 - 2x_2 \\ s.t. \quad & x_1 - 2x_2 \geq -4 \\ & -4x_1 - x_2 \geq -16 \\ & 2x_1 + 2x_2 \geq 7 \\ & x_1 \geq 2 \\ & -x_2 \geq -4 \\ & x \in Z_+^2. \end{aligned} \tag{7.3.3}$$

$$\begin{aligned} z_{LR}(\lambda) = \min & [-(7+\lambda)x_1 - (2-2\lambda)x_2] - 4\lambda \\ s.t. \quad & -4x_1 - x_2 \geq -16 \\ & 2x_1 + 2x_2 \geq 7 \\ & x_1 \geq 2 \\ & -x_2 \geq -4 \\ & x \in Z_+^2. \end{aligned} \tag{7.3.4}$$

同例 7.2.1 非常相似，将例 7.2.1 中(7.2.2)的第二个约束 $-5x_1 - x_2 \geq -20$ 变更为 $-4x_1 - x_2 \geq -16$，按定理 7.3.2 证明中的符号，

$$\begin{aligned} Q' &= \left\{ x \in R_+^n \mid Bx \geq d \right\} \\ &= \{ x \in R_+^n \mid -4x_1 - x_2 \geq -16,\ 2x_1 + 2x_2 \geq 7,\ x_1 \geq 2,\ -x_2 \geq -4 \}. \end{aligned}$$

用图 7.3.1 表示解集合 Q' 包括：粗黑线内部和边界。

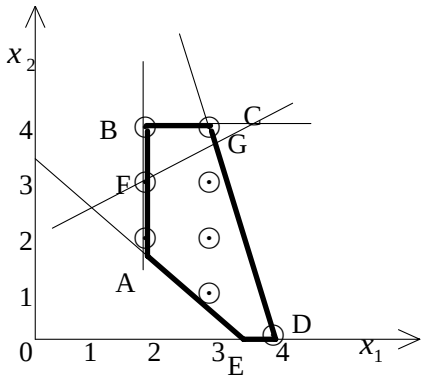


图 7.3.1 Q'集合示意图

很明显，它的极点为A、B、C、D、E，且A=(2,3/2)^{T}和E=(7/2,0) ^{T}不是整数点。因此，由推论 7.2.2 无法得到$z_{LP}=z_{LD}$。

与例 7.2.1 相同的分析，得到$\lambda^*=\frac{1}{9}$。当$0\le\lambda\le\lambda^*$时，最优解为$(3,4)^T$。当$\lambda^*\le\lambda$时，最优解为$(4,0)^T$。而LRL问题在本例的可行解区域为图 7.3.1 的粗黑线内部，同上面相同的讨论，他们对每一个$\lambda\ge 0$的最优值相同，即具有整数性。于是

$$z_{LR}(\lambda)=z_{LRL}(\lambda)=\begin{cases}-29+\lambda, & 当0\le\lambda\le\frac{1}{9},\\ -28-8\lambda, & 当\lambda\ge\frac{1}{9},\end{cases}$$

解得(7.3.3)的拉格朗日对偶问题目标值为

$$z_{LD}=z_{LR}(\frac{1}{9})=-28\frac{8}{9}。$$

线性规划松弛的可行解区域为

$$S'=\left\{x\in R_+^2\middle|\begin{array}{l}x_1-2x_2\ge -4,\ -4x_1-x_2\ge -16,\\ 2x_1+2x_2\ge 7,\ x_1\ge 2,-x_2\ge -4.\end{array}\right\}$$

它的所有极点为图 7.3.1 中的 A、F、G、D 和 E，即极点是

$$\left\{(2,\frac{3}{2})^T,(2,3)^T,(\frac{28}{9},\frac{32}{9})^T,(4,0)^T,(\frac{7}{2},0)^T\right\},$$

于是最优值$z_{LP}=-7\times\frac{28}{9}-2\times\frac{32}{9}=-28\frac{8}{9}=z_{LD}$。验证了定理 7.3.2 的结论。□

在例 7.3.3 的条件下，

$$z_{LD}=z_{LP}=-28\frac{8}{9}\ne z_{IP}=-28,$$

但并不是说$Q'=\left\{x\in R_+^n\middle|Bx\ge d\right\}$的某一个极点不是整数点时，不存在$z_{LD}=z_{LP}=z_{IP}$。

例 7.3.4(续例 7.3.3) 继续例 7.3.3 的讨论，将(7.3.3)中的约束$x_1-2x_2\ge -4$更改为$x_1-x_2\ge -1$，即模型为：

$$
\begin{aligned}
z_{IP} = \min & -7x_1 - 2x_2 \\
s.t. \quad & x_1 - x_2 \geq -1 \\
& -4x_1 - x_2 \geq -16 \\
& 2x_1 + 2x_2 \geq 7 \\
& x_1 \qquad \geq 2 \\
& \quad -x_2 \geq -4 \\
& x \in Z_+^2.
\end{aligned} \tag{7.3.5}
$$

$$
\begin{aligned}
z_{LR}(\lambda) = \min & [-(7+\lambda)x_1 - (2-\lambda)x_2] - \lambda \\
s.t. \quad & -4x_1 - x_2 \geq -16 \\
& 2x_1 + 2x_2 \geq 7 \\
& x_1 \qquad \geq 2 \\
& \quad -x_2 \geq -4 \\
& x \in Z_+^2.
\end{aligned} \tag{7.3.6}
$$

图 7.3.1 中的 A 点 $(2,\frac{3}{2})^T$ 是 $Q'=\left\{x \in R_+^n \mid Bx \geq d\right\}$ 的一个极点。与例 7.2.1 和例 7.3.3 相同的讨论，拉格朗日松弛问题(7.3.6)目标函数下降的方向从 $(\frac{7}{\sqrt{53}},\frac{2}{\sqrt{53}})^T$ 顺时针变化到

$$\lim_{\lambda\to\infty}(\frac{7+\lambda}{\sqrt{53+10\lambda+2\lambda^2}},\frac{2-\lambda}{\sqrt{53+10\lambda+2\lambda^2}})^T=(\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}})^T。$$

于是，只可能在两点$(3,4)^T$和$(4,0)^T$达到最优解。根据图中方向变化求得目标值。$(3,4)^T$和$(4,0)^T$两点的直线方程为 $y+4x=16$，它的一个垂直方向是 $(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}})^T$。此时求解λ满足

$$(\frac{7+\lambda}{\sqrt{53+10\lambda+2\lambda^2}},\frac{2-\lambda}{\sqrt{53+10\lambda+2\lambda^2}})^T=(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}})^T，$$

得到 $\lambda^*=\frac{1}{5}$。当 $0\leq\lambda\leq\lambda^*$ 时，最优解为$(3,4)^T$。当 $\lambda^*\leq\lambda$ 时，最优解为$(4,0)^T$。于是

$$z_{LR}(\lambda)=\begin{cases}-29, & 当0\leq\lambda\leq\frac{1}{5},\\ -28-5\lambda, & 当\lambda\geq\frac{1}{5}.\end{cases}$$

由此解(7.3.5)，得拉格朗日对偶问题目标值为

$$z_{LD}=z_{LR}(\frac{1}{5})=-29。$$

很容易验证 $x^*=(3,4)^T$ 为 IP 和 LP 的最优解，目标值满足 $z_{LD}=z_{LP}=z_{IP}=-29$。□

4．拉格朗日分解

拉格朗日分解(decomposition)的基本思想是：将整数规划问题 IP 改写成

$$\begin{aligned} z_{IP} &= \min c^T x \\ s.t.\ & Ax \geq b \\ & x = y \\ & By \geq d \\ & x, y \in Z_+^n, \end{aligned} \tag{7.3.7}$$

其中，增加辅助变量 y，使得约束 $x = y$ 耦合 $Ax \geq b$ 和 $Bx \geq d$ 两组约束。采用拉格朗日松弛的方法去掉(7.3.7)中的约束 $x = y$，对任意 λ 得

$$\begin{aligned} & \min c^T x + \lambda^T (x - y) \\ & s.t.\ Ax \geq b \\ & By \geq d \\ & x, y \in Z_+^n. \end{aligned} \tag{7.3.8}$$

由目标函数的线性性，(7.3.8)等价于

$$\begin{aligned} z_{LR1}(\lambda) &= \min c^T x + \lambda^T x \\ s.t.\ & Ax \geq b \\ & x \in Z_+^n, \end{aligned} \tag{7.3.9}$$

和

$$\begin{aligned} z_{LR2}(\lambda) &= \min - \lambda^T y \\ s.t.\ & By \geq d \\ & y \in Z_+^n. \end{aligned} \tag{7.3.10}$$

IP 问题松弛并且分解为(7.3.9)和(7.3.10)两个整数规划问题。它们每一个选 IP 的一部分约束作为自己的约束，使得每一个问题的问题规模减少。更重要的是分解的目的：寻求(7.3.9)和(7.3.10)的整数规划分解，并可以相对简单的求解。这同拉格朗日松弛的原理相同。(7.3.9)和(7.3.10)称为 IP 的拉格朗日分解。

由于拉格朗日分解是拉格朗日松弛的一种特殊情况，很容易验证

$$z_{LR1}(\lambda) + z_{LR2}(\lambda) \leq z_{IP}。$$

它的对偶问题是

$$z_{LD} = \max_{\lambda} z_{LR1}(\lambda) + z_{LR2}(\lambda)。 \tag{7.3.11}$$

有关的理论和计算同拉格朗日松弛相同。

5．增加拉格朗日松弛的约束

有时，拉格朗日松弛使得约束过于宽松，此时增加一些约束可以提高松弛后问题的计算效果。值得注意的是不能因为增加约束而使得问题的复杂性有本质的不同，即拉格朗日松弛问题从一个多项式问题转化为 NP-hard 问题。

例 7.3.5(续例 7.1.1) 当集合覆盖问题松弛所有的约束时，拉格朗日松弛问题为

$$\begin{aligned} z_{LRSC1}(\lambda) &= \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i \\ s.t.\quad & x_j \in \{0,1\}, j = 1,2,\cdots,n, \\ & \lambda \geq 0. \end{aligned} \tag{7.3.12}$$

其中

$$d_j = c_j - \sum_{i=1}^{m} \lambda_i a_{ij}\ ,$$

最优解为

$$x_j^* = \begin{cases} 1, & 若 d_j \le 0, \\ 0, & 其它. \end{cases}$$

从集合覆盖问题的本身可知：任意选定一行，则非零元素对应的列被选用的行覆盖；由此，n 列中存在不超过 m 列的一个列集合（每个列只少有一个 1）覆盖 m 行。增加约束

$$1 \le \sum_{j=1}^{n} x_j \le m,$$

使得(7.3.12)变为

$$\begin{aligned} z_{LRSC2}(\lambda) &= \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i \\ s.t. \quad & 1 \le \sum_{j=1}^{n} x_j \le m, \\ & x_j \in \{0,1\}, j = 1,2,\cdots,n, \\ & \lambda \ge 0. \end{aligned} \tag{7.3.13}$$

不难证明，(7.3.13)是集合覆盖 SC 的一个松弛，且对任意 $\lambda \ge 0$，有 $z_{LRSC1}(\lambda) \le z_{LRSC2}(\lambda) \le z_{SC}$。□

7.4 拉格朗日松弛算法

拉格朗日松弛算法主要包括次梯度优化算法和拉格朗日松弛启发式算法。它们的两个主要应用是给出 IP 问题的下界和构造基于拉格朗日松弛的启发式算法。由前面的讨论可知，每一个λ对应的 $z_{LR}(\lambda)$ 都可以作为 IP 的下界，下界的最佳值为 z_{LD}。求解 z_{LD} 的过程采用类似非线性规划的梯度——次梯度优化算法。进一步可以将次梯度算法扩展为拉格朗日松弛启发式算法。它的两个主要步骤是：第一步是确定一个λ及求解对应 LR 的最优解 x；第二步是当 x 不是 IP 的可行解时，将其可行化。

7.4.1 次梯度优化算法

次梯度优化（Subgradient Optimization）算法的思想与非线性规划的梯度下降思想相同。拉格朗日对偶问题希望 IP 的下界 $z_{LR}(\lambda)$ 尽可能大，于是按 $z_{LR}(\lambda)$ 的上升方向渐渐逼近 z_{LD}。次梯度优化算法就是根据 $z_{LR}(\lambda)$ 本身的分段线性性而构造。

定义 7.4.1 函数 $g: R^m \to R$ 满足

$$\forall x_1, x_2 \in R^m, 0 \le \alpha \le 1: \ g(\alpha x^1 + (1-\alpha)x^2) \ge \alpha g(x^1) + (1-\alpha) g(x^2),$$

则称 g(x)为凹函数。

定理 7.4.1 若 LR 的可行解集合 Q 是有限个整数点的集合，函数

$$z_{LR}(\lambda) = \min\left\{c^T x + \lambda^T (b - Ax) \middle| x \in Q\right\}$$

是凹函数。

证明：由于 Q 是有限个整数点的集合，记这些点为 $x^k (k = 1,2,\cdots,K)$，于是

$$z_{LR}(\lambda) = \min_{1 \le k \le K}\left\{c^T x^k + \lambda^T (b - Ax^k)\right\}。$$

记达到最优值的点为 q，则

$$z_{LR}(\lambda) = c^T x^q + \lambda^T (b - Ax^q)。$$

对 $\forall \lambda^1, \lambda^2$ 及 $\lambda = \alpha\lambda^1 + (1-\alpha)\lambda^2, 0 \le \alpha \le 1$，有

$$\begin{aligned} z_{LR}(\lambda) &= c^T x^q + \lambda^T (b - Ax^q) \\ &= \alpha\left[c^T x + (\lambda^1)^T (b - Ax^q)\right] + (1-\alpha)\left[c^T x + (\lambda^2)^T (b - Ax^q)\right] \\ &\geq \alpha z_{LR}(\lambda^1) + (1-\alpha) z_{LR}(\lambda^2). \end{aligned}$$

因此，$z_{LR}(\lambda)$ 为凹函数。□

定理 7.4.2 函数 g(x) 是凹函数的充分必要条件为 $\forall x^* \in R^m$，存在 $s = (s_1, s_2, \cdots, s_m)^T \in R^m$，使得

$$\forall x \in R^m : g(x^*) + s^T (x - x^*) \geq g(x)。$$

证明：“必要性”：假设 g(x) 为凹函数。对 $H = \{(x,z) | z \leq g(x)\}$ 和 $\forall (x^1, z^1), (x^2, z^2) \in H$，有

$$\alpha(x^1, z^1) + (1-\alpha)(x^2, z^2) = (\alpha x^1 + (1-\alpha)x^2, \alpha z^1 + (1-\alpha)z^2)$$

满足

$$g(\alpha x^1 + (1-\alpha)x^2) \geq \alpha g(x^1) + (1-\alpha) g(x^2) \geq \alpha z^1 + (1-\alpha) z^2,$$

所以，$\alpha(x^1, z^1) + (1-\alpha)(x^2, z^2) \in H$，故 H 为凸集。而 $(x^*, g(x^*))$ 是 H 的一个边界点，于是存在一个过点 $(x^*, g(x^*))$ 和由法方向 s 生成的支撑超平面 $\pi: g(x^*) + s^T (x - x^*)$ 满足

$$g(x^*) + s^T (x - x^*) \geq g(x)。$$

必要性得证。

“充分性”：$\forall x^1, x^2, 0 \leq \alpha \leq 1$, $x^* = \alpha x^1 + (1-\alpha x^2)$，有

$g(x^*) + s^T (x^1 - x^*) \geq g(x^1)$，$g(x^*) + s^T (x^2 - x^*) \geq g(x^2)$，

进一步

$$g(x^*) + s^T (\alpha x^1 + (1-\alpha)x^2 - x^*) \geq \alpha g(x^1) + (1-\alpha) g(x^2),$$

即

$$g(x^*) \geq \alpha g(x^1) + (1-\alpha) g(x^2)。\square$$

图 7.4.1 为定理 7.4.1 结论的一个示意图，$z_{LR}(\lambda)$ 是每一个极点达到最优的分段线性函数曲线

$$z_{LR}(\lambda) = c^T x^q + \lambda^T (b - Ax^q)。$$

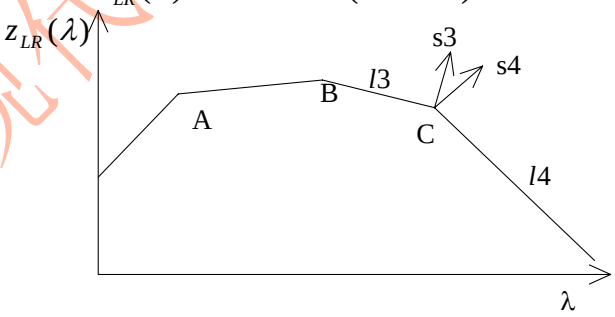


图 7.4.1 $z_{LR}(\lambda)$ 函数曲线示意图

其中，每一条直线表示最优解在某一个极点达到时的目标值。点 C 的两个方向 s3 和 s3 分别垂直于 *l*3 和 *l*4。沿 s3 和 s4 所夹部分的任何一个向量都可以作为满足定理 7.4.2 的方向而建立一个超平面。这样的方向在 $z_{LR}(\lambda)$ 可微时是唯一的，但在诸如 A、B 和 C 点时不是唯一的。光滑函数的梯度法在此不再可行。因此定义广义的梯度——次梯度(subgradient)。

定义 7.4.2 若 $g: R^m \to R$ 为凹函数，在 $x^* \in R^m$ 向量 $s \in R^m$ 满足

$$\forall x \in R^m : g(x^*) + s^T (x - x^*) \geq g(x),$$

则称 $s \in R^m$ 为 *g*(*x*)在 x^* 的一个次梯度。*g*(*x*)在 x^* 的所有次梯度组成的集合记为 $\partial g(x^*)$。

定理 7.4.3 若 $g(x)$为凹函数， x^* 为 $\max\{g(x)|x\in R^m\}$ 最优解的充分必要条件是 $0\in\partial g(x^*)$。

证明：

$$0\in\partial g(x^*)\Leftrightarrow 0(x-x^*)\geq g(x)-g(x^*),\forall x\in R^m \quad \square$$
$$\Leftrightarrow g(x)\leq g(x^*),\forall x\in R^m.$$

定理 7.4.4 设 LR 的可行解集合 Q 由有限个整数点组成，它的极点为 $x^k(k\in K)$，有

$$z_{LR}(\lambda^*)=\min_{k\in K}\{c^Tx^k+\lambda^{*T}(b-Ax^k)\}。$$

记

$$I=\{i|z_{LR}(\lambda)=c^Tx^i+\lambda^{*T}(b-Ax^i)\},\tag{7.4.1}$$

则对任选 $i\in I$，

$$s^i=b-Ax^i\tag{7.4.2}$$

为 $z_{LR}(\lambda)$ 在 λ^* 点的一个次梯度。且

$$\left\{s|s=\sum_{i\in I}\alpha_i s^i,\sum_{i\in I}\alpha_i=1,\alpha_i\geq 0\right\}\in\partial g(\lambda^*),$$

即所有 I 中的满足(7.4.2)的方向凸组合形成 λ^* 点的次梯度子集合。

证明：若 s^i 如(7.4.2)得到，再由(7.4.1)推导出

$$(s^i)^T(\lambda-\lambda^*)=\lambda^T(b-Ax^i)-\lambda^{*T}(b-Ax^i)$$
$$=c^Tx^i+\lambda^T(b-Ax^i)-\left[c^Tx^i+\lambda^{*T}(b-Ax^i)\right]$$
$$\geq z_{LR}(\lambda)-z_{LR}(\lambda^*).$$

所以，(7.4.2)定义的 s^i 为一个次梯度。

类似方法可以得到所有 I 中的满足(7.4.2)的方向凸组合形成 λ 点的次梯度子集合。□

由定理 7.4.1 有关 $z_{LR}(\lambda)$ 的凹性和借助图 7.4.1，若 λ^* 不是 $z_{LR}(\lambda)$ 的最大值点，相交的两个达到目标值的平面 $\pi_1:c^Tx^i+\lambda^T(b-Ax^i)(\lambda\in D_1)$ 和 $\pi_2:c^Tx^j+\lambda^T(b-Ax^j)(\lambda\in D_2)$ 满足 $\lambda^*\in D_1\cap D_2$，且两个平面的法方向交角不超过 90 度，也就是 $\partial z_{LR}(\lambda^*)$ 内的所有次梯度方向夹角不超过 90 度。

定理 7.4.1 推出 $z_{LR}(\lambda)$ 是凹函数。由凹函数的特性，知 $z_{LR}(\lambda)$ 在定义域内的任何一个开集内是连续函数。当 λ^* 是光滑点时，由定理 7.4.1 和定理 7.4.4 知次梯度唯一，是函数 $z_{LR}(\lambda)$ 的梯度方向，因此沿此梯度方向上升最快。在 λ^* 的一个邻域内 $z_{LR}(\lambda)$ 沿此方向上升。

当 λ^* 不是光滑点时，由 $z_{LR}(\lambda)$ 的连续性和(7.4.1)和(7.4.2)，在 λ^* 的一个邻域内 $\lambda=\max\{\lambda^*+\theta s^i,0\}$ 时，其中θ是一个充分小的正数，存在 $j\in I$，使得目标函数

$$z_{LR}(\lambda)=c^Tx^j+\lambda^T(b-Ax^j),$$

于是，由 $\partial z_{LR}(\lambda^*)$ 内的所有次梯度方向夹角不超过 90 度的结论，得到

$$z_{LR}(\lambda)-z_{LR}(\lambda^*)=(\lambda-\lambda^*)^T(b-Ax^j)\geq\theta(b-Ax^i)^T(b-Ax^j)\geq 0,\forall i\in I。$$

上面的讨论已经提示如何构造拉格朗日松弛算法，使得 $z_{LR}(\lambda)$ 逐步上升。基本步骤是：对给定的 λ^* 计算 $z_{LR}(\lambda^*)$；由(7.4.1)和(7.4.2)计算 λ^* 的一个次梯度，若次梯度满足定理 7.4.3 的充分性则达到最优解；否则以这个次梯度寻求 $z_{LR}(\lambda)$ 上升的方向。总结为算法：

次梯度优化算法

STEP1 任选一个初始拉格朗日乘子 λ^1，t=1；

STEP2 对 λ^t，从 $\partial g(\lambda^t)$ 中任选一个次梯度 s^t；若 s^t=0，则 λ^t 达到最优解而停止计算；否则 $\lambda^{t+1}=\max\{\lambda^t+\theta_t s^t,0\}$，$t$：=$t$+1，重复 STEP2。

在 STEP2 中，θ_t 满足

$$\sum_{t=1}^{\infty}\theta_t=\infty,\ \text{且}\theta_t\to 0,t\to\infty。\tag{7.4.3}$$

(7.4.3)是拉格朗日对偶问题收敛的理论结果[3]。由次梯度优化算法可知，$z_{LR}(\lambda)$ 是一个凹函数，代替光滑函数的梯度上升法，它是一个次梯度上升算法。也就有非线性规划收敛的理论结果(7.4.3)。实际计算中，不可能如同(7.4.3)那样迭代无穷多次，因此产生多样的确定 θ_t 的方法和具体执行次梯度优化算法的技术问题。下面就主要的问题分类讨论。

1．θ_t 的选取。

实际计算的目的是尽快得到一个可以接受的下界或是对已经得到的一个解可行化，无论如何，常常采用启发式的方法。其中一类方法是

$$\theta_t=\theta_0\rho^t,\ 0<\rho<1。\tag{7.4.4}$$

这类方法使 θ_t 以指数速度下降，因此迭代次数较少。另一类方法是

$$\theta_t=\frac{z_{UP}(t)-z_{LB}(t)}{\|s^t\|^2}\beta_t,\tag{7.4.5}$$

其中，$0\le\beta_t\le 2$，一般取 $\beta_0=2$。当 $z_{LR}(\lambda)$ 上升时，β_t 不变，当 $z_{LR}(\lambda)$ 在给定的若干步没有变化时，则取其一半。(7.4.5)中的 $z_{UP}(t)$ 为 IP 最优目标值的一个上界。可以用一个可行解的目标值确定，也可以通过估计的方法得到。$z_{UP}(t)$ 可随 t 的变化而逐步修正。$z_{LB}(t)$ 是 $z_{LR}(\lambda^t)$ 的一个下界，一般选取 $z_{LB}(t)=z_{LR}(\lambda^t)$，但有时为了计算简单，只取一个固定值。(7.4.5)的主要思想是用 $z_{UP}(t)-z_{LB}(t)$ 修正变化的速度。

2．停止原则

（1）迭代次数不超过 T。这是一种最为简单的原则。无论解的质量如何，到达迭代步数则停止。由此很容易控制计算的复杂性，但解的质量无法保证。

（2）$s^t=0\in\partial z_{LR}(\lambda^t)$。这是最为理想的状态，由定理 7.4.3，$\lambda^t$ 达到拉格朗日对偶问题的最优解。在实际计算中，由于问题的复杂性和计算机本身的计算误差，这样的结果较难达到，常常用 $\|s^t\|\le\varepsilon$（ε为给定的非负数）来代替。

（3）$z_{UP}(t)=z_{LB}(t)$。在 $z_{UP}(t)$和$z_{LB}(t)$ 可变时，这种情况表示已得到 IP 的最优解，最优值为 $z_{IP}=z_{UP}(t)=z_{LB}(t)$。

（4）λ^t 或目标值 $z_{LR}(\lambda^t)$ 在规定的步数内变化不超过一个给定的值。这时认为目标值不可能再变化，因此，停止运算。

具体应用中，可以采用以上停止原则之一，也可以综合运用。

7.4.2 拉格朗日启发式算法

拉格朗日松弛的一个主要工作是提供 IP 的一个下界。LR 问题的一个比较好的下界所对应的解也应该同 IP 的最优解相近。在这样的逻辑下，产生了拉格朗日启发式算法。拉格朗日松弛启发式算法主要包含两个部分。第一部分就是拉格朗日的次梯度优化计算。由于第一部分得到的 LR 的解不一定为 IP 的可行解，第二部分就是对第一部分得到的解可行化。

例 7.4.1 假设集合覆盖 SC 问题通过拉格朗日松弛得到一个解 $x=(x_1,x_2,\cdots,x_n)^T$，当 x 不是 SC 问题的一个可行解时，即存在 i 使得

$$\sum_{j=1}^{n} a_{ij} x_j = 0$$

时，一个直观的可行化方法是：求 k 满足

$$c_k = \min_{1 \le j \le n} \{ c_j | a_{ij} = 1 \},$$

使得 $x_k = 1$。

重复以上的判别和计算直至所有的行被覆盖。这样得到的算法是基于拉格朗日松弛的启发式算法。所得的解为可行解。□

拉格朗日松弛启发式算法总结为：

第一阶段：次梯度优化。由启发式算法的特性，次梯度优化时不一定要求得到 z_{LD}，这样有很多基于次梯度优化的启发式方法。

第二阶段：可行化。在 LR 的解不可行时，对其可行化。

系数修正法就是拉格朗日松弛启发式算法的一种形式。在给以 λ^0 后，类似次梯度优化算法，系统地调整拉格朗日系数，以改进 LR 的下界。这一方法的优点是计算量较次梯度优化少且每一步使 LR 的下界上升。它的缺点是所得的下界可能比较差，修正的方法依赖于问题本身。我们先从下面的示例了解系数修正法。

例 7.4.2 对集合覆盖问题

$$\begin{aligned} \min\ & 2x_1 + 3x_2 + 4x_3 + 5x_4 \\ s.t.\ & x_1 + x_3 \ge 1 \\ & x_1 + x_4 \ge 1 \\ & x_2 + x_3 + x_4 \ge 1 \\ & x_j \in \{0,1\},\ j = 1,2,3,4. \end{aligned}$$

假设 $\lambda_1 = 1.5, \lambda_2 = 1.6, \lambda_3 = 2.2$.

$$\begin{aligned} z_{LR}(\lambda) = \min\ & \{-1.1x_1 + 0.8x_2 + 0.3x_3 + 1.2x_4\} + 5.3 \\ s.t.\ & x_j \in \{0,1\},\ j = 1,2,3,4. \end{aligned}$$

$z_{LR}(\lambda)$ 的最优解为 $x_1 = 1, x_2 = x_3 = x_4 = 0, z_{LR}(\lambda) = 4.2$。

第三行没有被覆盖，在可覆盖第三行中选费用最小列

$$\delta = \min\{0.8, 0.3, 1.2\} = 0.3,$$

替代

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 1.6 \\ 2.2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 1.5 \\ 1.6 \\ 2.5 \end{pmatrix},$$

得到

$$z_{LR}(\lambda) = \min\{-1.1x_1 + 0.5x_2 + 0.9x_4\} + 5.6 = 4.5。$$

LR 的最优解为 $x_1 = x_3 = 1, x_2 = x_4 = 0$。用算法表示为

STEP0 初始化 λ^0，t=0；

STEP2 计算 $z_{LR}(\lambda^t)$；

STEP3 若所有行被覆盖，停止计算；否则，记 $s_i = 1$ 表示第 i 行没有被覆盖，在没有被覆盖的行中任选一个行 k，计算

$$\delta_k = \min\{d_j | a_{kj} = 1, s_k = 1\},$$

其中 $d_j = c_j - \sum_{l=1}^{m} \lambda_l$；

STEP4 $\lambda_i^{t+1}=\begin{cases}\lambda_k^t+\delta_k, i=k,\\ \lambda_i^t, \quad i\neq k,\end{cases}$ t=t+1,返回 STEP2。

例 7.4.2 不同于次梯度优化算法是迭代步长取为 1。因此它是一种启发式的算法。□

7.5 拉格朗日松弛在能力约束单机排序问题中的应用

柔性制造单元的排序问题是约束单机排序问题的一类，其特点是生产的多样性以满足市场的不同需求，但受生产能力的限制，必须对其生产进行调度，以期达到最优的生产效益。加之每一个顾客对产品的需求量、完成时间等的要求不尽相同，因此我们提出一个加权目标的可拆分约束单机排序问题[4]。

本节首先建立了一个加权目标约束单机排序问题的数学模型，并证明了该问题为NP-hard，然后给出一个简单的启发式算法和一个拉格朗日松弛算法并研究其最优解的性质。最后，对这两个算法的计算结果进行了对比分析。

假设 n 种产品的外部需求为$\{d_i, i=1,2,...,n\}$。加权目标约束单机排序问题的数学模型(WCS)(Weighted Capacitated Single machine scheduling)如下：

$$Z=\min\sum_{i=1}^{n}w_iT_i \tag{7.5.1}$$

$$s.t.\quad \sum_{t=1}^{T_i}x_{it}=d_i,\quad i=1,2,\cdots,n, \tag{7.5.2}$$

$$\sum_{i=1}^{n}\{a_ix_{it}+s_iY_{it}\}\le c_t, t=1,2,\cdots,\max\{T_i|i=1,2,\cdots,n\}, \tag{7.5.3}$$

$$Y_{it}=\begin{cases}1, & 若x_{it}>0,\\ 0, & 其它,\end{cases}\quad T_i=\max\{t|x_{it}>0\}, i=1,2,\cdots,n; t=1,2,\cdots, \tag{7.5.4}$$

$$x_{it}\ge 0, i=1,2,\cdots,n; t=1,2,\cdots, \tag{7.5.5}$$

其中，w_i：产品 i 的权因子，T_i：完成需求 d_i 所需的时段，x_{it}：时段 t 产品 i 的生产批量，a_i：产品 i 的单位产品所占用的能力，s_i：产品 i 的生产准备所占用的能力，c_t：t 时段的可供能力总量。

模型 WCS 中，x_{it}，Y_{it}，T_i 为变量，x_{it} 为决策变量，决定每一产品在各时段的生产批量，Y_{it} 和 T_i 为 x_{it} 的应变量，由(7.5.4)式表示。(7.5.1)为目标函数，使完成所有需求 d_i (i=1, 2, ..., n)所需的时段的加权平均值最小，该目标函数中的权因子是基于用户对产品需求的紧迫性、重要性等因素来决定的。(7.5.2)表示总产量与总需求平衡。(7.5.3)为能力约束方程。由(7.5.4)可以看出，WCS 是一个非线性的混合整数规划问题。

定理7.5.1 WCS的问题复杂性为强NP-hard。

证明: 定义一个 3-partition 问题为：3T 个元素满足

$$\frac{D}{4}<d_i<\frac{D}{2}, i=1,2,\cdots,3T,\ \sum_{i=1}^{3T}d_i=TD,$$

是否能将上面 3T 个元素划分成 T 部分，使得每部分包含三个元素且三元素之和为 D?

按下列条件构造一个判定问题：对给定 T，在 WCS 中，设 w_i=1，a_i=1，s_i=A>D>0，产品需求为 $d_i, i=1,2,\cdots,n$。它们满足

$$n=3T,\ \sum_{i=1}^{3T}d_i=TD,\ \frac{D}{4}<d_i<\frac{D}{2},\ c_t=\text{D+3A},\ \text{D>0}。$$

是否有

$$\sum_{i=1}^{n} w_i T_i \le \frac{3}{2}(1+T)T\ ? \tag{7.5.6}$$

首先，非常容易验证 3-partition问题的任何一个“是”实例是WCS判定问题的一个“是”实例。因为每一时段的能力约束为D+3A，所以最多可以生产 3 种产品。又n=3T，d_i >0，由(7.5.6)，必须在T时段内完成所有的产品加工，这样在T时段内每一时段正好生产 3 个产品，且不允许任一产品拆分，因为任何一个产品拆分必增加一个生产准备占用。于是，WCS的一个“是”实例是 3-partition问题的一个“是”实例。反之，对WCS判定问题的任何一个“是”实例，由它的条件，限定每个时段能且只能生产三个产品，还必须在T个时段内完成。由此证明 3-partition问题多项式转换为WCS的判定问题。已知 3-partition问题是强NP-C[2]，于是该WCS判定问题为强NP-C，因此，优化问题WCS为强NP-hard。□

下面构造一个简单而直观的算法。该算法的基本思想是将{ w_i, i=1, 2, ..., n}中较大权数所对应产品尽可能早的完成，以达到目标(7.5.1)最小。

算法 A

STEP0 S=∅，U={1, 2, ..., n}。从第一个时段 t=1 开始；

STEP1 当$U \neq \varnothing$时，w_{i*}=max{ w_i | i∈ U}，依时段 t 能力约束(7.5.3)情况将 i*尽可能往前安排直到d_{i*}全部生产。可能出现下几种情况：

a) 若 i*的全部需求d_{i*}没有全部生产，且时段 t 的能力足以满足产品 i* 的生产准备占用，则以 t 时段的最大余能力生产产品 i*，产品 i*的未能加工部分到 t +1 时段生产，t=t+1，返回 STEP1。

b) 若 i*的全部需求d_{i*}已全部生产，则 S=S+{i*}。当 S={1, 2, ..., n}时，停止，否则 U=U-S，返回 STEP1。

c) 若 i*的全部需求d_{i*}没有全部生产，且无法在该时段生产，则 U=U-{i*}，返回 STEP1。

STEP2 若 U= ∅，S≠ {1, 2, ..., n}，则 t=t+1，U={1, 2, ..., n}-S，返回 STEP1。

这一算法实际是一种贪婪算法。从直观看，这一算法同背包问题的贪婪算法非常相象。不同的是，WCS 要求所有的外部需求都得完成，而背包问题中在能力不够时，某些物品可以不装。下例表明算法 A 的最坏界为无限。

例7.5.1 设产品数n=6，生产准备占用能力 s_i =5 (i=1, 2, ..., 6)，单位产品占用能力 a_i =1 (i=1, ..., 6)，其他系数如下：

w_1 =1.05，w_2 =1.04，w_3 =1.03，w_4 =1.02，w_5 =1.01，w_6 =1.00，

d_1 =6，d_2 =5，d_3 =4，d_4 =3，d_5 =2，d_6 =1, c_1 =25，c_2 =26，c_t =5+$\frac{1}{2^{t-2}}$ (t >2)。

由算法 A 计算得：在 T=1 时段，生产 d_1 =6，d_2 =5，能力结余 4，因为 s_i =5(i=1, ⋯, 6)，所以无法再安排其他生产。在 T=2，生产 d_3 =4，d_4 =3，d_5 =2，能力还余 2。同 T=1 相同的道理，d_6 =1 只能在 t > 2 的时段安排。因 s_i =5，$\sum_{t=3}^{\infty}\frac{1}{2^{t-2}}=1$，故有 d_6 =1 的完成期 $T_6 \to +\infty$。从例中所给系数可以安排生产计划，在 T=1 和 T=2 两个时段完成所有的外部需求。如 T=1 时生产 d_2 =5、d_4 =3、d_5 =2，T=2 时生产 d_1 =6、d_3 =4、d_6 =1。因此，算法 A 在最坏情况下无界。□

在给出拉格朗日松弛算法之前，先研究 WCS 的拉格朗日松弛理论。从 WCS 的模型中可以看出，能力约束(7.5.3)将各产品联系在一起，因此对约束(7.5.3)进行拉格朗日松弛。因 T_i 是一个变量，是目标函数中需要优化的变量，而拉格朗日乘子的维数为 T= $\max\{T_i | i=1,2,\cdots,n\}$，是一个变量。技术处理的方法是假设 T 充分大，即认为产品可以

在有限的时段内加工完成。

对于充分大的 T，拉格朗日松弛后的数学模型 LRP 是：

LRP

$$Z=\max_{\lambda} Z(\lambda)$$

$$=\max_{\lambda}\min_{X}\{\sum_{i=1}^{n} w_i T_i+\sum_{t=1}^{T}\lambda_t\{\sum_{i=1}^{n}[a_i x_{it}+s_i Y_{it}]-c_t\}\}$$

s.t. (7.5.2), (7.5.4), (7.5.5)和 $\lambda_t \geq 0$。

其中，$X=(x_{it})_{n\times T}$，一个 $n\times T$ 的变量矩阵，$\lambda=(\lambda_1,\lambda_2,\cdots,\lambda_T)$。

由于松弛了(7.5.3)，(7.5.2)、(7.5.4)和(7.5.5)所决定的模型LRP是无能力约束的单机排序问题，对给定充分大的T，各产品的完工时间都可在T时段内完成，即$1\leq T_i \leq T$。因拉格朗日松弛后目标函数的惩罚性，使得T $(1\leq T_i \leq T)$在一个充分大的范围内取值。在技术处理LRP时，T可以从1逐步增加。对给定的T，等价于当$t>T$取 $\lambda_t=0$。

记

$$g(X,T,\lambda)=\sum_{i=1}^{n} w_i T_i+\sum_{t=1}^{T}\lambda_t\{\sum_{i=1}^{n}[a_i x_{it}+s_i Y_{it}]-c_t\}, \tag{7.5.7}$$

则(7.5.7)可以改写为下列形式：

$$g(X,T,\lambda)=$$

$$\sum_{i=1}^{n}[w_i T_i+\sum_{t=1}^{T}\lambda_t a_i x_{it}+\sum_{t=1}^{T}\lambda_t s_i Y_{it}]-\sum_{t=1}^{T}\lambda_t c_t, \tag{7.5.8}$$

再记

$$h(i,X,T,\lambda)=w_i T_i+\sum_{t=1}^{T}\lambda_t a_i x_{it}+\sum_{t=1}^{T}\lambda_t s_i Y_{it}, \tag{7.5.9}$$

由(7.5.7)，(7.5.8)和(7.5.9)可推出

$$\max_{\lambda} Z(\lambda)=\max_{\lambda}\min_{X}\{\sum_{i=1}^{n} w_i T_i+\sum_{t=1}^{T}\lambda_t\{\sum_{i=1}^{n}[a_i x_{it}+s_i Y_{it}]-c_t\}\}$$

$$=\max_{\lambda}\min_{X} g(X,T,\lambda)$$

$$=\max_{\lambda}\{\sum_{i=1}^{n}\min_{X} h(i,X,T,\lambda)-\sum_{t=1}^{T}\lambda_t c_t\}。$$

LRP 可以表示如下：

$$Z=\max_{\lambda}\{\sum_{i=1}^{n}\min_{X} h(i,X,T,\lambda)-\sum_{t=1}^{T}\lambda_t c_t\} \tag{7.5.10}$$

s.t. (7.5.2), (7.5.4), (7.5.5)和 $\lambda_t \geq 0$。

求解 LRP 就可分成两步。首先，当 $\lambda \geqslant 0$ 为定数时求解下子问题：

SUBP

$$\min_{X} h(i,X,T,\lambda)=\min_{X}\{w_i T_i+\sum_{t=1}^{T}\lambda_t a_i x_{it}+\sum_{t=1}^{T}\lambda_t s_i Y_{it}\} \tag{7.5.11}$$

s.t. (7.5.2)，(7.5.4)和(7.5.5)，

得

$$Z(\lambda)=\sum_{i=1}^{n}\min_{X} h(i,X,T,\lambda)-\sum_{t=1}^{T}\lambda_t c_t, \tag{7.5.12}$$

而后，再从所有的 $\lambda \geqslant 0$ 中对(7.5.12)求最大值。

定理 7.5.2 对给定充分大的 T，若 λ_t (t=1,2,...,T)已知且非负，则 SUBP 一定有下列形式的最优解，

$$x_{it}=\begin{cases} d_i, & \text{存在某一} t^*, 0\le t=t^*\le T,\\ 0, & t\ne t^*。\end{cases}$$

证明 设 x_{it}^* (t=1,2,...,T)为 SUBP 的最优解。设 $t=t_1$，t_2，…，t_m 时，$x_{it}^*>0$。令 $\lambda_{t^*}=\min\{\lambda_t \mid x_{it}>0\}$。根据(7.5.11)，很易得

$$\begin{aligned}&\sum_{t=1}^{T}\lambda_t a_i x_{it}+\sum_{t=1}^{T}\lambda_t s_i Y_{it}\\ &\ge \lambda_{t^*}a_i d_i+\lambda_{t^*}s_i Y_{it^*}\\ &\ge \lambda_{t^*}a_i d_i+\lambda_{t^*}s_i,\end{aligned}$$

且由 $w_i T_i\ge w_i t^*$，推出，

$$x_{it}^{**}=\begin{cases} d_i, & t=t^*,\\ 0, & t\ne t^*,\end{cases}$$

为最优解。结论得证。文中后续的拉格朗日算法正是基于此结论，逐步迭代逼近最优解。□

由定理 7.5.2，求解 SUBP 的最优解只需判断

$$\begin{aligned}&w_i t^*+\lambda_{t^*}\{a_i d_i+s_i\}\\ &\le w_i t+\lambda_t\{a_i d_i+s_i\},\qquad 0\le\forall t\le T。\end{aligned}\tag{7.5.13}$$

计算出(7.5.13)的 t^*最多需要 T^2 次比较，所以当 λ 确定时，求 SUBP 的最优解相当容易。此时，可得到{ x_{it}，Y_{it}，i=1, 2, ..., n，t=1, 2, ..., T}。我们还必须求解 λ 使得 $Z(\lambda)$接近或等于 Z。当 $X=\{x_{it}\mid i=1,2,\cdots,n;t=1,2,\cdots,T\}$ 为满足(7.5.13)的最优解时，下列向量 $v(X)$=(v(1, X), v(2, X) , ...,v(T, X))

$$v(t,X)=\{\sum_{i=1}^{n}[a_i x_{it}+s_i Y_{it}]-c_t\}^+,\qquad t=1,2,\cdots,T,\tag{7.5.14}$$

为 LRP 满足(7.4.2)的次梯度(练习题)，其中，$x^+=\begin{cases}x, & x\ge 0,\\ 0, & \text{其它.}\end{cases}$

采用(7.4.5)的迭代步长法和以下两条判停准则：

(1) 给定迭代步数上限；

(2) 给定充分小的 ε >0，$\sum_{t=1}^{T}v^2(t,X^k)<\varepsilon$ 时。

由拉格朗日松弛判停后，所得到的解 X 不一定是 WCS 的可行解。当为 WCS 的不可行解时，需进行解的可行性修改。修改的原则是，若时段 t 的排序超过其能力，尽量将权数大的所对应的产品前移到尚有余能力的时段生产。若前移不行，则尽可能将权数小的产品后移至有能力余量的一个时段生产，以达解的可行。若前两方案都达不到可行解，则以再增加时段来完成。

算法 B

STEP0 T=1，$\lambda^0(1)$=0，Z^*为算法 A 求得的目标值，k=0。

STEP1 解 SUBP，以(7.5.13)分别求出 SUBP 中每一产品的最优解，再以(7.5.14)求次梯度 $v(X)$，最后由(7.4.5)求解 θ_k，$\lambda^{k+1}=\max\{0,\lambda^k+\theta_t v(X)\}$。若不满足判停准则，

(i) 当 $\lambda^{k+1}(T)=0$ 时，k=k+1，返回 STEP1；

(ii) 当 $\lambda^{k+1}(T)>0$ 时，T=T+1，$\lambda^{k+1}(T)=0$，k=k+1，返回 STEP1；若满足判停准则，

到 STEP2。

STEP2 若所求解为 WCS 的可行解，停止。否则，按上面讨论进行可行性处理。

以上算法在 PC-386 采用 Turbo C 编程，分别就 n=30 个产品，将产品需求分为四类，每类各产生 10 个正态分布的数据文件，其正态分布的均值和方差见表 7.5.1。每一时段的能力约束分别为 w_i 与 s_i 均值之和的 1，2，4，6 倍，a_i 都取 1。因此，由四类产品需求及四类能力约束的组合，共 16 组。每组 10 个数据文件，共 160 个数据文件进行验证。

表 7.5.1 数据分类

项目	均值	方差	项目	均值	方差
权因子 w	50	10	产品需求 2	10	2
生产准备 s	10	2	产品需求 3	40	10
产品需求 1	5	1	产品需求 4	100	10

计算中，拉格朗日松弛的判停迭代次数为 200 次，ε=0.01。计算结果见表 7.5.2，其中，比例(a ： b)为能力约束与 s_i 和 d_i 两项均值和的比，Z_A 为算法 A 的计算结果，Z_{LB} 为算法 B 的拉格朗日松弛下界，Z_B 为算法 B 可行化后的最终结果。表中 Z_A，Z_B，Z_{LB} 为对应 10 个数据文件计算结果的平均值，GAP1 和 GAP2 按如下定义

$$GAP1=\frac{Z_B-Z_{LB}}{Z_{LB}},\qquad GAP2=\frac{Z_A-Z_B}{Z_A-Z_{LB}}。$$

表 7.5.2 计算结果的平均值

组数(比例)	Z_A	Z_{LB}	Z_B	GAP1	GAP2
1 (1:1)	35047.40	20322.76	31008.30	0.53	0.27
2 (1:1)	32969.50	20167.81	34853.10	0.73	-0.15
3 (1:1)	25631.60	19267.57	32511.30	0.69	-1.08
4 (1:1)	25018.20	20249.87	28256.20	0.40	-1.89
5 (2:1)	13987.60	10440.72	13747.10	0.32	0.07
6 (2:1)	13073.20	10430.44	13644.60	0.31	-0.22
7 (2:1)	11943.60	10195.26	11924.10	0.17	0.01
8 (2:1)	11383.80	10527.58	11382.20	0.08	0.00
9 (4:1)	6504.60	5646.47	6246.80	0.11	0.30
10 (4:1)	6361.10	5655.45	6247.40	0.10	0.16
11 (4:1)	6100.40	5518.72	5915.40	0.07	0.32
12 (4:1)	6016.90	5675.72	5837.60	0.03	0.53
13 (6:1)	4406.00	4045.23	4309.20	0.07	0.27
14 (6:1)	4376.70	4047.10	4380.80	0.08	0
15 (6:1)	4303.90	3957.02	4188.80	0.06	0.33
16 (6:1)	4255.90	4054.58	4224.00	0.04	0.16

从计算结果可以看出，当每一时段能力平均意义下只能完成一个或两个产品时(1:1 或 2:1 部分)，算法 A 的结果比较好，八组中，有四组结果好于算法 B，一组相同，三组次于算法 B，从直观解释，此时可根据权因子大小排序。但当每时段可完成多类产品时，算法 B 的结果好于算法 A，八组都不次于算法 A，特别是第十二组差值高达 0.53。同时还可以看出，当每时段可完成产品数越多时，拉格朗日松弛算法所得下界 Z_{LB} 和 B 的相对误差变得越小。从计算结果综合来说，拉格朗日松弛算法对能力约束单机排序问题是比较适用的。

练习题

1. 证明：(7.5.14)是 LRP 满足(7.4.2)的次梯度。
2. 与松弛相对应的一个概念是限制。整数规划 IP 的一个限制问题

$$z_R = \min\{z_R(x) | x \in S_r\}$$

满足(a) $S_R \subseteq S = \left\{x \in Z_+^n | Ax \geq b, Bx \geq d\right\}$，(b) $z_R(x) \geq cx$。

(i) 通过什么样的处理方法可以实现 IP 的限制？

(ii) 能得到怎样的结论？

3. 整数规划问题

$$\min 7x_1 + 6x_2 + 2x_3$$
$$s.t. \quad \begin{pmatrix}3\\3\end{pmatrix}x_1 + \begin{pmatrix}3\\1\end{pmatrix}x_2 + \begin{pmatrix}1\\2\end{pmatrix}x_3 \geq b$$
$$x \in Z_+^3.$$

分别用图解的方法就下面条件求上面整数规划问题的拉格朗日对偶最优值。

(i) $b = \begin{pmatrix}5\\4\end{pmatrix}$ 时，拉格朗日松弛第一个约束；

(ii) $b = \begin{pmatrix}5\\4\end{pmatrix}$ 时，拉格朗日松弛第二个约束的值；

(iii) (i)和(ii)的值是否有差别？b 为多少时，(i)和(ii)的值相等且与上面整数规划的目标值相？

4. 整数规划问题

$$\max 2x_1 + 5x_2$$
$$s.t. \quad \begin{pmatrix}4\\1\\1\end{pmatrix}x_1 + \begin{pmatrix}1\\4\\-1\end{pmatrix}x_2 \leq \begin{pmatrix}28\\27\\1\end{pmatrix}$$
$$x \in Z_+^2.$$

对上面整数规划问题按下面各条件分别讨论：

(i) 证明：采用拉格朗日松弛上面整数规划中的任何两个约束，则拉格朗日对偶值等于上面整数规划的线性规划松弛的最优值。

(ii) 寻找一个目标函数使得(i)不成立。

(iii) 证明：拉格朗日松弛任何一个约束的拉格朗日对偶值都是线性规划松弛的改进。

(iv) 将(i)-(iv)用图解法表示出来。

5. 就下面整数规划问题分别松弛两个约束的两种拉格朗日松弛方法

$$\max \sum_i \sum_j c_{ij} x_{ij}$$
$$s.t. \quad \sum_j x_{ij} \leq 1, \quad i \in M$$
$$\sum_i l_i x_{ij} \leq b_j \quad j \in N$$
$$x \in \{0,1\}^{|M| \times |N|}.$$

比较它们在下面个方面的优劣势：

(i) 松弛后的子问题是否容易求解？

(ii) 拉格朗日对偶是否容易求解？

(iii) 拉格朗日对偶值的优劣？

6. 就下面整数规划问题分别松弛两个约束的两种拉格朗日松弛方法

$$\min \sum_{i\in M}\sum_{j\in N} h_{ij}y_{ij} + \sum_{j\in N} c_j x_j$$

$$s.t.\ \sum_j y_{ij} \le a_i,\ \ i \in M$$

$$\sum_i y_{ij} \le b_j x_j,\ \ j \in N$$

$$y_{ij} \le \min(a_i, b_j)x_j,\ \ i \in M, j \in N$$

$$y \in R_+^{mn}, x \in \{0,1\}^{|N|}.$$

讨论拉格朗日松弛各种约束的优劣性。

7. 考虑无约束单机排序问题：n 个工件在一台机器上加工，p_j、r_j 和 w_j 分别是工件 j 的加工时间、最早开工时间和权数。这些是已知参数。记 t_j 为工件 j 的开工时间。极小化加权开工时间的单机排序问题，使 $\sum_{1\le j\le n} w_j t_j$ 最小。若没有最早开工时间的限制，即所有 r_j=0，最优加工顺序是按 $\frac{w_j}{p_j}(j=1,2,\cdots,n)$ 从大到小加工。如何建立这个问题的数学模型和进行拉格朗日松弛以得到问题的一个下界？

8. 考虑第四章 4.6 节的生产批量问题，怎样应用拉格朗日松弛得到问题的一个下界？怎样构造一个基于拉格朗日松弛的启发式算法，以得到问题的一个可行解？

9. 通过数值计算的方法验证对练习 7 的分析结果。

10. 就关心的组合优化问题应用拉格朗日松弛算法。

参考文献

1. Khachian L G. A polynonial algorithm for linear programming. Doklady Akad. Nauk USSR, 1979, 244(5):1093~1096

2. Garey M R, Johnson D S. Computers and Intractibility: A Guide to the Theory of NP-Completeness. Freeman, 1979

3. Fisher M L. The Lagrangian relaxation method for solving integer programming problems, Management Science, 1981, 27(1):1~18

4. Xing W, Zhang J, Jiang Q et al. Capacitated single flexible manufacturing cell with setups: model, complexity and Lagrangean relaxation. In: Ding-Zhu Du et al ed. Operations Research and Its Applications. 1995, 162~170