2 启发式优化算法基础

这个世界喜欢优化(Optimization)。商人寻求低风险高回报的投资；消费者喜欢质量好价格低的产品；工厂老板希望降低成本提高产量；国家领导人期望既节约资源又能让经济快速发展。甚至在我们吃饭、穿衣服、走路、说话、生产、消费的每个时刻，我们都在无意中进行着某种选择，期望选用某种最优方案。优化潜伏在这个世界的每个时刻每个角落，以至于大数学家欧拉说“Nothing at all takes place in the Universe in which some rule of maximum or minimum does not appear[33]. ”(宇宙中每个事物都是某种规则下的最大值或最小值)。正因为如此，许许多多的科学研究和生产实践的问题，都最终落实为优化问题。为这些问题设计好的优化算法，能够提高生产效率，减少资源消耗；能够给我们带来更优质的商品、更高效的设计、更科学的规划；能够创造巨大的经济效益和社会价值。

在各种类型的优化问题中，存在着一类非常重要的困难的优化问题：NP难度的优化问题。它们广泛出现在科学研究和生产实践的各个领域，是各自领域里的核心问题和基础性问题。但是，关于计算复杂性理论的研究经验表明，对于NP难度的优化问题，可能根本不存在高效率的精确完备算法。迄今为止，人们所能找到的求解NP难度问题的完备算法都是指数型的，只适用于规模较小的问题实例。对于实际应用中的稍大规模的问题实例，其计算时间动辄达到天文数字，无法满足实际需要。为了现实的求解那些具有重要实际价值的大规模的复杂优化问题，人们把目光转向了非完备的启发式优化算法。在理论上，启发式优化算法不能保证找到全局最优解，甚至无法保证找到可行解。但是，许多实际的研究经验表明，好的启发式优化算法往往能够在合理计算时间内找到满足实际需求的高质量的解方案，能够又快又好的解决问题。在求解许多来自实际应用的大规模复杂优化问题时，启发式优化算法已成为研究人员们的首选（甚至唯一）方法。

启发式优化算法具有一些非常有趣的性质。第一、启发式优化算法很神秘。一方面，设计启发式优化算法本质上没有固定的套路，许多优秀算法的得来靠的是人们在长期求解问题时积累的尚未想清楚的感觉和经验。另一方面，启发式优化算法的运行时表现(run-time behavior)复杂多变，很难分析和把握。第二、启发式优化算

法很有效。在许多经典NP难度问题以及实际应用问题上的研究经验都表明，启发式优化算法能够求解大规模的复杂的问题实例，能够在较短时间内找到优度非常接近理论下界的解方案。第三、大部分启发式优化算法都非常简洁、天然、富于人情味。许多最先进的启发式优化算法都是基于一些简单的、天然的、富于人情味的思路。有的算法来自于模拟一些自然现象，譬如说金属的退火过程、生物的进化过程、蚂蚁寻找食物的过程、等等；也有的算法来自于人们在上千年的生产实践中积累的经验和感觉，譬如说求解矩形Packing和圆形Packing问题的占角策略、求解工件车间调度问题的各种优先指派规则、等等。

在最近的几十年里，关于启发式优化算法的研究在持续的推进。一方面，各式各样的新的算法设计思路和算法分析方法不断涌现，使启发式优化算法在求解经典问题上的能力越来越强。另一方面，在科学研究和生产实践的各个领域涌现了越来越多的新的复杂的优化问题，为启发式优化算法领域提出了新的挑战。可以预见，在今后相当长一段时间内，启发式优化算法将会在计算机科学、人工智能和运筹学领域占据越来越重要的地位，在求解各类复杂的实际优化问题上产生越来越显著的经济效益和社会价值。

本章将简要介绍关于启发式优化算法的基础知识。首先将给出优化问题和最优解的严格定义，并介绍关于启发式优化算法的两个重要理论背景——计算复杂性理论和没有免费的午餐定理。然后，将介绍若干经典的启发式优化算法设计思路，包括局部搜索的概念以及禁忌搜索、模拟退火等经典算法模式。最后，将介绍启发式优化算法的评价方法——基于Benchmark的经验分析方法。

2.1 优化问题与最优解

在我们日常生活中，存在着许许多多形形色色的优化问题，它们出现在不同的领域，面对不同的对象，具有不同的特性。但是从数学上讲，所谓优化，即是要从一系列的候选解中找出某个评价标准下的“最优”解。在最一般情况下，优化问题可被定义为许多问题实例(instances) 的集合，每个问题实例有二元组(S, f)给出，其中S是候选解的集合(即解空间)，$f : S \to R$是评价函数。求解实例(S, f)即是要在S中找到一个元素s，使得f在S上取最小值。在很多情况下，也称$s \in S$为解、决策变量，称f为目标函数、评估函数、打分函数。

优化问题有许多种类型。根据解空间S和目标函数f的不同特性，优化问题可分成：离散优化、连续优化、以及离散和连续混合优化。根据问题的求解难度，优化问题可分成容易的优化问题和困难的优化问题。本文主要面对两类困难的优化问题，即具有NP难度的组合优化和全局优化问题。

典型的NP难度的组合优化问题包括: 合取范式可满足性问题(Satifiability Problem,SAT)、旅行商问题(Traveling Salesman Problem,TSP)、加工调度问题(Scheduling Problem)、0-1背包问题(Knapsack Problem)、装箱问题(Bin Packing Problem)、图着色问题(Graph Coloring Problem)、聚类问题(Clustering Problem)等等。

典型的NP难度的全局优化问题包括：蛋白质折叠问题(Protein Folding Problem)、设施布局问题(Facility Location Problem)、飞行器轨迹规划问题(Trajectory Planning)、开普勒问题(The Kepler Conjecture)、物质结构优化问题(Stucture Geometry Optimization Problem)、圆形Packing问题(Circle Packing Problem)等等。

在求解组合优化与全局优化问题时，会面临一个突出的矛盾，即局部最优解和全局最优解之间的矛盾。在很多情况下，寻找一个局部最优解比较容易，但是寻找全局最优解非常困难。为了准确描述局部最优解和全局最优解，我们给出如下定义。

定义 2.1 (**全局最优解** Global Optimum)： 一个解$s^* \in S$是全局最优解当且仅当s^*在f上的取值小于等于S中任何其它元素在f上的取值，即$\forall s \in S, f(s^*) \leq f(s)$。

局部最优解的概念和解空间的邻域结构有关。在最一般情况下，邻域可被如下定义。

定义 2.2 (**邻域** Neighborhood)： 邻域函数N是一个映射，$N : S \to 2^S$，它将S中的每个元素s映射到S的一个子集$N(s) \subseteq S$上。若$s' \in N(s)$，则称s'是s的一个**邻居(neighbor)**。

针对不同类型的问题，可以定义不同的邻域结构。在连续优化问题中，邻域采用数学分析中的经典定义。

定义 2.3： 在连续优化问题中，解s的邻域定义为以s为中心以$\epsilon(\epsilon > 0)$为半径的一个球，$N(s) = \{s' \in R^n | \ \|s' - s\| < \epsilon\}$，其中$\|s' - s\|$是$s$和$s'$的欧氏距离。

在组合优化问题中，邻域与施加在解上的动作(move)相关。在很多时候，我们对解s施加某个动作mv，使其轻微的改变，得到一个相邻的解s'。这个从一个解到达它的相邻解的过程可被形式化的描述为：$s' = s \oplus mv$。此时，解s的邻域可被定义为通过对s施加某种动作能够达到的相邻解的集合。

定义 2.4： 在组合优化问题中，设$\mathcal{T}(s)$为所有可以施加于解s的动作的集合，s的邻域即是通过对s施加$\mathcal{T}(s)$中的某个动作可以达到的解的集合，即$N(s) = \{s \oplus mv \mid mv \in \mathcal{T}(s)\}$。

在给定解空间S的邻域结构后，局部最优解的概念可被如下定义。

定义 2.5 (局部最优解 Local optimum)： 在某个给定的邻域函数N下，若解s在f上的取值小于等于它的每个邻居，即$\forall s' \in N(s)$有$f(s) \leq f(s')$, 则称s是局部最优解。

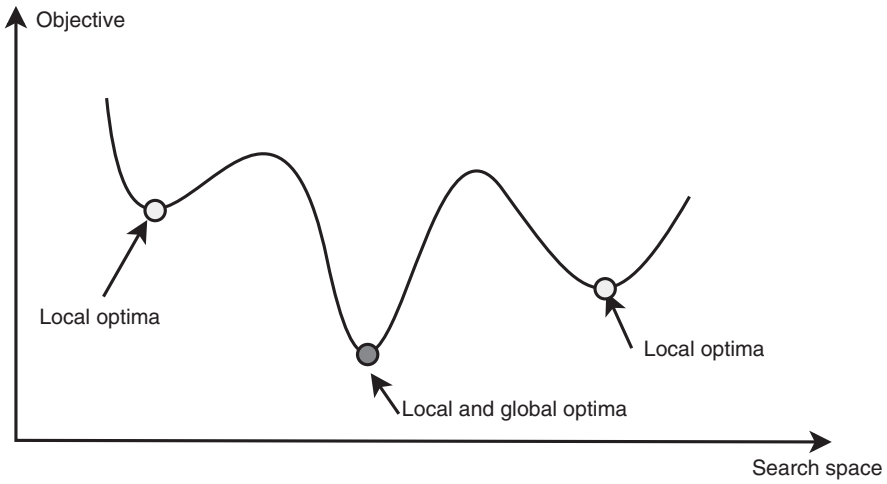

图 2-1 局部最优解与全局最优解

图2-1给出了解空间中局部最优解和全局最优解的示意图。值得注意的是，局部最优解的概念与具体的邻域结构密切相关。在一个邻域函数N_1下的局部最优解并不一定是另一个邻域函数N_2下的局部最优解。

2.2 理论背景

关于求解NP难度的优化问题，有两个非常重要的理论背景：计算复杂性理论[10–12]和没有免费的午餐定理(No Free Launch Theorem)[13,14]。计算复杂性理论中的

研究经历告诉我们，对于NP难度问题，可能根本不存在那种精确完备而又快速高效的求解算法。为了现实的求解这些问题，我们应该在解的优度以及算法的完备性上做出妥协。没有免费的午餐定理告诉我们，优化算法的通用性与算法的效率之间是有矛盾的。一个算法的适用范围越广，往往效率越差。在设计高效率的求解算法时，应该具体问题具体分析，将与问题本质特性密切相关的信息融入到算法中去。

2.2.1 计算复杂性理论

计算复杂性理论主要研究哪些问题是容易计算的，哪些问题是难计算的。问题的计算复杂度与求解该问题的算法的计算复杂度密切相关。一般而言，所谓算法的计算复杂度指的是算法的时间复杂度，即从算法开始执行到终止所经历的计算步数。算法的时间复杂度一般表示为问题规模n的函数$f(n)$。按照算法的时间复杂度，可以把算法分成两类。计算时间可以用多项式限界的算法称为多项式时间算法；而计算时间用指数函数限界的算法则称为指数时间算法。一般认为，多项式时间算法是高效的，而指数时间算法是低效的。因为指数时间算法通常只能用于求解规模很小的问题实例，随着问题规模的增长，其计算时间会迅速增长到让人无法接受的程度。

问题的计算复杂度定义为能精确求解该问题的最好的算法的计算复杂度。在计算复杂性理论中，人们认为，如果一个问题存在多项式时间算法能够求解则该问题是容易计算的。反之，如果求解该问题的最好算法的也是指数时间的，那么认为该问题是难求解的。

在计算复杂性理论中通常把常见的问题分成两类：P类问题和NP类问题。所谓P类问题，指的是确定型的图灵机能够在多项式时间之内求解的问题。而NP类问题，指的是非确定型的图灵机能够在多项式时间之内求解的问题。因此，一个问题属于P，则对其存在多项式时间的精确型求解算法，因此可以认为它是易于求解的；而如果一个问题属于NP，则对其不一定存在多项式时间的精确型求解算法，那么它有可能是难解的。

在19实际70年代，人们发现，在NP类问题中存在某种特殊类型的问题，如果对其存在多项式型的求解算法，则NP类问题中的所有问题都可以在多项式时间之内求解。这种类型的问题称为NP完全问题。第一个被发现的NP完全问题是合取范式的可满足性问题(SAT)。著名的Cook定理即是说：SAT $\in$ P 当且仅当P=NP。

说问题H是NP难度的(NP hard)当且仅当存在某个NP完全问题L能够在多项式时间之内归约到问题H。根据这个定义，如果某个NP难度问题能够在多项式时间内求解，则一定存在某个NP完全问题能够在多项式时间求解，更进一步，根据NP完全问题的特性，则NP类问题中的所有问题都可以在多项式时间之内求解，即P=NP。

P是否等于NP是计算机科学中的著名的尚未解决的难题。经过几十年的研究，人们普遍猜测P≠NP。根据这一猜测，对于NP难度问题，根本不存在多项式时间的精确型求解算法。对于日常生活中出现的一些经典的重要的NP难度问题，人们投入了大量的精力进行研究，迄今为止，所能找到的最好的精确型求解算法的时间复杂度都是指数型的。

计算复杂性理论中的相关研究经历启发我们，对于NP难度问题，有很大的可能性根本不存在人们所希求的那种又快又好精确完备的求解算法。为了现实的求解日常生活中出现的一些重要的NP难度问题，**我们应该在解的优度和算法的完备性上作出妥协。即，不要求必须找到全局最优解，能找到满足实际需求的高质量的解就可以了；不要求算法能够快速求解所有情形的问题实例，能够高效率的求解实际应用中出现的绝大多数情形的问题实例就可以了**。

计算复杂性理论给我们的启发与实际的研究经验是相吻合的。虽然从理论上讲，精确完备的求解NP难度问题是非常困难的。但是，在实际应用中，我们经常能够为一些困难的NP难度问题找到又快又好的启发式求解算法。利用这些算法，能够在绝大多数情况下为实际问题找到满足实际需要的解决方案。

2.2.2 没有免费的午餐定理

没有免费的午餐定理由Wolpert 和Macready 在1995年正式给出[13]，其原文描述为：“*We show that all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.*”(我们证明，所有的寻找函数最优值的算法在所有可能的优化问题上的平均效率是一样的。具体的，如果算法A在某些问题上的性能超过算法B，粗略的说，那么必然在同样多的其它问题上，算法B的性能超过算法A。)

对于本文而言，没有免费的午餐定理主要给了我们两方面的启发。第一、**算法求解问题的效率和算法的通用性之间是有矛盾的**。一个最通用的算法，往往也是效率最低下的算法。因此，在评价某一个算法的时候，不能因为算法的通用性不够强，而指责算法的价值较小。特别是在求解NP难度的优化问题的时候，如果某个算法在某一个具体问题的某些算例上有非常显著的效果，那么这个算法就已经有很大的价值。第二、**研究求解NP难的优化问题的时候，不应该去追求那种“一劳永逸”的通用型算法，而应该具体问题具体分析**。在设计求解算法的过程中，不能只是简单的套用某些通用的算法模板，而应该花精力去研究具体问题的特性。如果能够将问题的一些本质特性抓住，并将相关的感觉和经验形式化，融入到算法设计中去，往往能够得到一个针对性的高效率的求解算法。

2.3 经典的启发式优化算法

对于NP难度的优化问题，存在三种类型的求解算法[34]：精确算法(Exact algorithm)、近似算法(Approximate algorithm)和启发式算法(Heuristic algorithm)。精确算法的优点在于它能保证找到问题的全局最优解；缺点在于算法的时间复杂度一般是指数型的，对于规模稍大的问题实例，计算时间长到无法忍受。精确算法适用于问题规模很小的情形，或者是问题本身有特殊结构的情形。近似算法的优点在于它能保证在一个合理的计算时间内找到一个质量有保障的解。它能通过理论证明，最终找到的解的优度与全局最优解的优度相差在某个范围之内。它的缺点在于，一般而言，找到的解的优度不高，理论上保障的解的优度离实际需求经常相差较远。近似算法适用于必须在合理计算时间内找到有严格质量保障的解的情形。启发式算法的优点在于它在大多数情形下能够在合理计算时间内找到质量满足实际需求的解方案；它的缺点在于它对最终找到的解的质量甚至可行性没有任何理论上的保障。图2-2给出了各种类型的优化方法的分类[34]。

图2-3给出了一些比较有名的启发式优化算法的发展图谱[34]。其中，1947年Dantzig为线性规划问题提出的单纯形法(Simplex method)可被认为是第一个局部搜索算法[36]。1971年，Edmonds首次在求解组合优化问题上提出贪心构造方法(Greedy heuristic)[35]。图中列出的其它的启发式优化算法包括：ACO (ant colonies optimization)、AIS (artificial immune systems)、BC(bee colony)、CA(cultural algorithm-

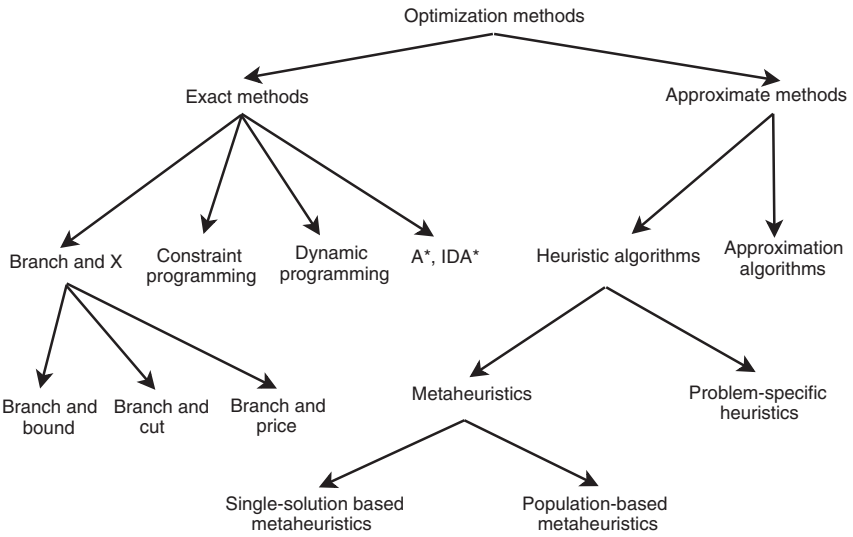

图 2-2 各种类型的优化方法

s)、CEA (coevolutionary algorithms)、CMA-ES (covariance matrix adaptation evolution strategy)、DE (differential evolution)、EDA (estimation of distribution algorithms)、EP (evolutionary programming)、ES(evolution strategies)、GA (genetic algorithms)、GDA (great deluge)、GLS (guided local search)、GP (genetic programming)、GRASP (greedy adaptive search procedure)、ILS (iterated local search)、NM (noisy method)、PSO (particle swarm optimization)、SA (simulated annealing) 、SM (smoothing method)、SS (scatter search)、TA(threshold accepting)、TS (tabu search)和VNS (variable neighborhood search)。

本节主要介绍几种非常具有代表性的启发式优化算法。首先给出局部搜索算法的基本概念，并介绍三种经典的局部搜索算法：单调下降的局部搜索算法、模拟退火算法和禁忌搜索算法。接着介绍对局部搜索算法的改进算法——迭代局部搜索算法。最后介绍群体算法(Population-based algorithm)中非常有代表性、在许多问题上获得成功的Memetic算法。

2.3.1 局部搜索算法的基本概念

在启发式优化算法中，局部搜索(Local search)是最基本的搜索模式。在局部搜索的每一步，对当前解做局部的轻微的改变，使得搜索从当前解移动到一个新的相

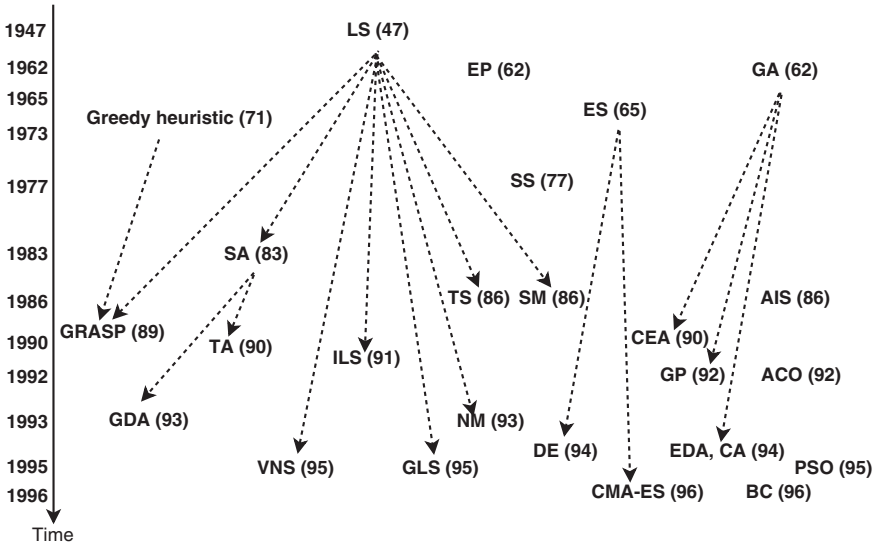

图 2-3 启发式优化算法图谱

邻解。通过不断的从一个解移动到相邻解，搜索能够探测解空间中的许多区域。在这个探测过程中，一般还会引入对每个解进行评价的函数，并要求从当前解尽量移动到一个优度更好的解。通过不断的进行这种变好的移动，搜索能够找到越来越好的解。整个局部搜索过程不断地进行下去，直至达到某种停止标准。局部搜索的思想非常简单天然，但是在这个思想上诞生了许多当前最先进的启发式优化算法，其中包括：单调下降的局部搜索算法、模拟退火算法(Simulated Annealing)[38]、禁忌搜索算法(Tabu Search)[39]、迭代局部搜索算法(Iterated Local Search)[26]、动态局部搜索算法(Dynamic Local Search)[32]、变邻域搜索算法(Variable Neighborhood Search)[27]、蚁群算法(Ant Colony Optimization)[37]、GRASP(Greedy Randomized Adaptive Search Procedure)[40]等等。

在设计几乎所有的局部搜索算法时都会面临三个重要的问题：(1)如何设计合适的邻域结构? (2)如果设计高效率的邻域评估策略? (3)如何让搜索在疏散性和集中性之间达到合适的平衡? 下面分别对这三个问题进行介绍。

(1) 邻域设计。邻域结构是局部搜索算法的基石，没有邻域结构，局部搜索算法无从谈起。邻域结构的设计决定了一个局部搜索算法能够探测到的解空间的区域，决定了算法是否能够找到一个高质量的解。一个好的灵活的邻域结构是设计高效率的局部搜索算法的基础。在为一个具体问题设计邻域结构时，基本的思路是要使得上层的局部搜索算法能够更方便快速的找到高质量的解。

邻域结构与解的表示方式(Solution Representation)息息相关，在更深层次上，也与问题的结构以及约束的强弱有关。有如下三种常见的解的表示方式:

- **二进制串表示**。在这种表示方式下，解空间中的每个解表示为一个由0和1组成的二进制串。在离散优化问题中，二进制表示方法非常流行，因为许多离散优化问题可以天然的用二进制变元建模。典型的例子包括：最大团问题(Max Clique)、SAT/Max-SAT、UBQP(Unconstrained Binary Quadratic Programming)问题等等。在二进制表示方式中，有两种基本的邻域结构：*k-flip* 以及*Swap*。所谓*k-flip*是翻转(flip)解中的$k(k \geq 1)$个变元，使其从0变成1或者从1变成0。而*Swap*则是交换两个取值不同的变元的值。值得注意的是，$Swap$动作可以通过两个*1-flip*动作实现。
- **全排列表示**。在这种表示方式下，解空间中的每个解表示为n个数的全排列。典型的例子包括：旅行商问题(Traveling Salesman Problem, TSP)、调度问题(Flow-Shop/Job-Shop Scheduling)、线性排列问题(Linear Arrangement)等等。在全排列的表示方式中，有两种基本的邻域结构：*Insert*和*Swap*。对于一个排列$\Pi = (\pi_1, \pi_2, \ldots, \pi_n)$, *Insert*动作将排列中的第$i(i = 1, 2, \ldots, n)$个元素$\pi_i$取出放到第$j(j = 1, 2, \ldots, n; j \neq i)$个位置。当$i < j$时，新排列为$\Pi' = (\pi_1, \pi_2, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_i, \pi_j, \pi_{j+1}, \ldots, \pi_n)$;当$i > j$时，新排列为$\Pi' = (\pi_1, \pi_2, \ldots, \pi_{j-1}, \pi_i, \pi_j, \pi_{j+1}, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n)$。*Swap*动作则交换排列中第$i$和第$j$个元素$\pi_i, \pi_j$的位置，新排列为$\Pi' = (\pi_1, \pi_2, \ldots, \pi_{i-1}, \pi_j, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_i, \pi_{j+1}, \ldots, \pi_n)$。
- **整型向量表示**。在这种表示方式下，解空间中的每个解表示为一个整型向量，向量的每个分量的值取自某个给定的定义域。整形向量的表示方法非常适合于表示现实生活中的许多约束满足性问题(Constraint Satifiability Problem, CSP)。在整形向量表示方式中，一个基本的邻域结构是*one-change*。即将整形向量中的某个分量的值从当前值变成另外一个值。

(2)邻域评估。在局部搜索算法中，需要不断地从当前解s移动到邻域中的某个相邻解$s' \in N(s)$。这个移动通常是有偏向性的，即要求s'比s有更好的优度($f(s') < f(s)$)；在许多情况下，甚至要求s只能移动到$N(s)$中优度最好的相邻解上。为了实现这种有偏向性的移动，在局部搜索的每一步都需要评估当前解s的一个或者多个相邻解的优度。在实际计算中，邻域评估占据了局部搜索算法绝大部分的

计算量。如何找到一种快速的评估邻域中每个解的优度的方法，是设计高效率的邻域搜索算法的关键。最简单直接的邻域评估方法是从头算起，即在搜索的每一步直接计算$N(s)$中的每个解s'的优度$f(s')$。在实际经验中，从头算起的方法往往也是最低效的方法。通常有如下三种加速邻域评估的方法。

- **增量更新**。注意到在邻域搜索过程中，当前解s与其相邻解s'的差别很小，而且这种差别是因为某种固定的动作($move$)引起的，因此在很多情况下可以得到一个计算$\Delta f = f(s') - f(s)$的简单函数，比$f(s')$具有小得多的计算量。此时，$\Delta f < 0$，说明s'比s具有更好的优度；Δf越小，说明s'的优度越好。因此在评估s的每个邻居的时候，只要计算容易计算的Δf就可以了，不需要从头计算$f(s')$。这种只计算Δf的策略叫增量更新策略。增量更新策略在图染色、SAT、旅行商问题等许多经典的组合优化问题上得到广泛应用。
- **Δ 表**。所谓Δ表是一种记录当前解s与其每个邻居$s' \in N(s)$的优度偏差$\Delta f = f(s') - f(s)$的数据结构。通过搜索当前解s的Δ表，可以方便的查找s与每个邻居s'的偏差值，从而找到邻域中优度最好的邻居。在局部搜索算法中，通过Δ表，可以实现邻域评估的缓存(Caching)技术。当搜索从一个解s移动到相邻解s'的时候，s'的Δ表并不需要从头开始计算。因为在很多情况下，s的Δ表与s'的Δ表在大部分表项上取值都是相同的。因此，计算s'的Δ表的时候，只要从s的Δ表出发，更新那些少数不同的表项就可以了。在局部搜索过程中，Δ 表是一种非常重要的数据结构，能显著的提高局部搜索的效率。在使用Δ表时，最重要的是分析清楚Δ表的更新策略，即搜索从当前解s移动到相邻解s'之后，Δ 表中哪些表项需要更新，哪些不需要更新。值得指出的是，Δ 表的更新策略是与具体的问题密切相关的，需要具体问题具体分析。在SAT、图染色、最大团等经典组合优化问题上，Δ表是非常重要的加速技术。
- **近似评估**。在有些问题上，评价函数$f(s)$的计算量比较大，而且使用增量更新和Δ表技术并不能显著的提升计算效率。此时可以使用近似评估的方法，即用一个容易计算的简单的函数$f'(s)$来替代$f(s)$。在此种方法下，最关键的问题在于如何找到一个合适的近似评价函数$f'(s)$。在工件车间调度(Job-shop Scheduling)等经典NP难度问题上，近似评估是局部搜索方法中非常重要的加速技术[41,42]。

(3) 集中性和疏散性。在启发式优化算法中，有两个非常重要而又“神秘”的概

念：搜索的集中性(Intensification)和疏散性(Diversification)[39]。到底什么是搜索的集中性和疏散性，迄今为止人们都没有给出明确的严格定义。一般来说，集中性反映的是搜索能够细致深入的探索解空间中某个部分区域的能力；疏散性反映的是搜索能够不断的探索解空间不同的有前途的区域的能力。集中性使得搜索能够从当前解出发逐渐的寻找到优度越来越好的解，保证搜索能达到一定的深度；而疏散性能够防止搜索被限制在解空间中的某个局部区域，保证搜索能覆盖一定的广度。在设计启发式优化算法过程中，需要同时考虑集中性和疏散性两个方面。片面的强调集中性，要么使得搜索陷在解空间中的某个局部区域出不来，要么使得计算的时间过长以至于无法忍受。而片面的强调疏散性则容易使得搜索失去方向，找不到质量较高的解。因此，一个高效率的启发式优化算法需要在集中性和疏散性之间达到一个合理的平衡。

值得注意的是，**疏散性和集中性的概念与中国哲学中的阴阳的概念有相通之处**。在某种意义上，集中性是阳，疏散性是阴。集中性是一种执着的追求，是“天行健，君子以自强不息”；疏散性是一种包容的胸怀，是“地势坤，君子以厚德载物”。

2.3.2 单调下降的局部搜索算法

单调下降的局部搜索算法可能是最古老最简单的启发式优化方法。它从一个初始解出发，在搜索的每一步，从当前解s走向其邻域$N(s)$中的某个具有更好优度的邻居。当搜索到达某个局部最优解，即其邻域中不存在更好的邻居的时候，算法终止。单调下降的局部搜索算法的流程见算法9：

Input: 初始解s_0

Output: 从s_0出发得到的局部最优解

```
s ← s_0 ;
while s不是局部最优解 do
    从s的邻域N(s)中挑选一个邻居s' 使得f(s') < f(s);
    s ← s' ;
end
return s;
```

算法 1: 单调下降的局部搜索算法

在单调下降的局部搜索算法中，当前解s的邻域中可能存在多个优度更好的邻居(改进邻居)，此时算法要采用某种挑选策略(Pivoting rule), 从多个改进邻居中挑选一个替代当前解。一般而言有三种不同的挑选策略：

- **选改进最大的邻居(Best improvement, Steepest descent)**。即从所有改进邻居中，挑选对当前解改进最大的邻居。此策略的优点在于其挑选到的邻居对当前解的改进是最大的，其缺点在于需要完整的搜索整个邻域。当邻域结构比较大，或者评估每个邻居的优度代价较大的时候，这种方法比较耗时。
- **选第一个遇到的改进邻居(First improvement)**。即在逐个计算当前解的邻居的优度的时候，第一次遇到能改进的邻居之后，就选择该邻居。此策略的优点在于很多时候不需要完整的搜索当前解的整个邻域，其缺点在于找到的改进邻居的优度并不一定是最好的。
- **随机选一个改进邻居(Random selection)**。即从当前解的所有改进邻居中随机选择一个。此策略的优点在于它增强了算法的疏散性，其缺点在于比较耗时。

单调下降的局部搜索算法的优点在于流程简单，收敛快，容易实现。在实际问题中，它经常能够在很短时间内找到质量较高的解方案。在求解复杂约束的问题的时候，单调局部搜索算法因为其简单直接，而成为最适合的下降算法。

单调下降的局部搜索算法的缺点在于，在面对问题地貌特性(Landscape)比较复杂的情形，容易陷入局部最小值的陷阱。值得指出的是，单调局部搜索算法是许多更先进的局部搜索算法的基石。人们为了克服单调下降的局部搜索算法容易陷入局部最小值的特点，提出了许多种改进策略，形成了许多更复杂更高效的局部搜索算法。有以下四种跳出局部最优值陷阱的改进策略：

- **从另一个初始解重新开始单调下降过程**。此策略被用于多初始点方法(Multi-start method)、迭代局部搜索方法(Iterated local search)、GRASP方法等等。
- **接受非改进邻居**。在此种策略中，允许接受非改进的邻居，使得搜索可以逃离局部最优值的陷阱。此策略被用于模拟退火(Simulated Annealing, SA)方法和禁忌搜索(Tabu Search, TS)算法。
- **转换邻域结构**。当搜索转换到一个新的邻域结构之后，当前的局部最优解在新的邻域结构下不再是局部最优解，因此搜索能够继续进行下去。此种策略

被用于变邻域搜索算法(Variable Neighborhood Search, VNS)中。

- **改变评价函数**。改变评价函数后，当前的局部最优解在新的评价函数下不再是局部最优解。此种策略被用于有导向的局部搜索(Guided Local Search, GLS)方法和动态局部搜索(Dynamic Local Search, DLS)方法中。

2.3.3 模拟退火算法

模拟退火可能是最著名的应用最广泛的通用型启发式优化方法，它由Scott Kirkpatrick, C. Daniel Gelatt 和Mario P. Vecchi 在1983提出[38]。顾名思义，模拟退火是模拟金属的退火过程。在冶金术中，使用退火过程是为了增大晶粒的体积，减少晶格中的缺陷。它一般会先将材料加热，然后经特定速率冷却。在正常温度下，材料中的原子停留在使内能有局部最小值的位置，通过加热使其内能增大，原子会离开原来位置而随机在其它位置移动。在缓慢的退火过程中，原子有很大可能性找到内能更低的位置。

模拟退火算法从一个初始解开始，在每一次迭代过程中，随机的试探当前解的某个邻居。如果相邻解比当前解更优，则接受(Accept)相邻解(使用相邻解替代当前解)；如果相邻解比当前解要差，则以一定的概率接受相邻解。接受较差的相邻解的概率有两种因素决定：(1) 相邻解相对于当前解变差的程度。变差的程度越小，接受的概率越大；反之越小。(2) 退火温度。温度越大，则接受变差的相邻解的概率越大，反之越小。在整个搜索过程中，退火的温度从一个初始设定的值开始慢慢变小。在搜索的早期阶段，温度相对较高，容易接受变差的解；到了后期阶段，则基本上不接受变差的解。通过温度的逐渐降低，搜索最终收敛到某个局部最优值。模拟退火算法的流程见算法2。

模拟退火算法是应用非常广泛的启发式优化算法，在许多经典的NP难组合优化与全局优化问题上都取得了显著的效果。它的优点在于算法简单，容易实现，能够为大规模的复杂优化问题找到高质量的解。它的缺点在于，对参数敏感。如果没有找到一个合适的参数，算法的效果往往不尽人意。

在设计模拟退火算法求解具体的NP难优化问题时，要注意以下几点：

- **初始温度**。在模拟退火算法中，温度是控制算法贪心程度的决定性因素。在温度很高的时候，模拟退火等同于随机行走(Random walk)算法；而在温度很

Input: 初始解s_0
Output: 搜索过程中找到的最好解

$s \leftarrow s_0$; /* 产生初始解 */
$T \leftarrow T_0$; /* 设定初始温度 */
repeat
  repeat
    随机选取s的一个邻居s';
    $\Delta E \leftarrow f(s') - f(s)$;
    if $\Delta E \leq 0$ **then**
      接受s', 即$s \leftarrow s'$;
    else
      以概率$e^{\frac{-\Delta E}{T}}$ 接受s' ; /* Metropolis接收准则 */
    end
  until *未达到平衡条件*;
  降低温度T;
until *达到停止标准*;
return 整个搜索过程中找到的最好解;

算法 2: 模拟退火算法

低的时候，模拟退火则等同于单调下降的局部搜索算法。一般情况下，初始温度不能选得太高，否则算法在搜索早期的很长时间内只是在解空间中随机走动。在面对具体的问题时，我们的经验是，把初始温度设置到某个值，使得变差程度不大的邻居有一定的接收概率。

- **固定温度下的迭代步数**。在某个固定的退火温度上，在达到平衡状态之前，退火算法必须要尝试一定数量的移动动作，即搜索必须在某一固定温度下经历一定数量的Metropolis迭代步数。一般而言，在某个固定退火温度下的迭代步数与问题的邻域有关，通常选取为邻域大小$|N(s)|$的常数倍。
- **退火过程**。一般情况下有三种类型的退火方法：(1)线性下降，即$T \leftarrow T - \beta$, 其中β是一个设定的常数。(2)几何下降，机$T \leftarrow \alpha * T$, 其中α是预先给定常数。(3)自适应方法(Adaptive method)，即搜索根据遇到的不同的地貌特性以及在过去一段时间内的行为自适应的调节温度。

- **停止标准**。模拟退火算法的停止标准一般有两种：(1)当温度下降到小于某个值之后，算法停止。(2)当搜索过程中找到的历史最好解经过某个最大的迭代步数仍未得到改进，算法停止。

2.3.4 禁忌搜索算法

禁忌搜索算法是启发式优化算法领域最著名的局部搜索算法之一，由Glover和Hansen在80年代分别独立提出[39]。禁忌搜索算法最显著的特征在于它能通过接受非改进邻居来逃离局部最优解的陷阱，并通过使用禁忌表禁忌最近经过的解或者执行的动作使得搜索不在一个小范围内兜圈子。经过几十年的发展，禁忌搜索在许多复杂的组合优化和全局优化问题上取得了巨大的成功。它经常能够在较短时间之内，为大规模的复杂的问题找到优度接近或者达到最优的解。在工件车间调度、图染色、最大团、约束可满足性问题等经典NP难度问题上，禁忌搜索算法是当前最好的局部搜索算法。

禁忌搜索算法的流程见算法3。其流程类似于最陡下降的局部搜索算法。它从一个初始解开始，在搜索的每一步使用当前解的邻域中未被禁忌的优度最好的邻居替换当前解。这个过程不断的迭代下去，直到达到某个停止条件。与最陡下降的局部搜索算法相比，禁忌搜索算法主要有以下四点不同:

- **禁忌表(Tabu list)**。在禁忌搜索过程中，使用禁忌表记录搜索最近执行的动作或者最近经过的解。如果一个解或者达到解的动作在禁忌表中，则该解处于禁忌状态。在搜索过程中，不允许使用被禁忌的解替换当前解。在禁忌搜索算法中，对一个解或者动作的禁忌是有一定时间期限的。一般把一个刚刚做过的动作或者刚刚经历过的解放入禁忌表，经过一定的迭代次数之后，再将这个解或者动作从禁忌表中取出。一个解或者动作从进入禁忌表到出禁忌表所经历的迭代步数叫做这个解或者动作的**禁忌期限**(Tabu tenure)。使用禁忌表的一个最直接的好处在于它使得搜索在陷入局部最优值的时候仍然能继续前进，又能避免搜索在一个小范围内陷入死循环。
- **解禁规则(Aspiration criterion)**。 在大部分的禁忌搜索算法中，禁忌表中存放的是搜索最近执行的动作。这种策略使得一些从未经过的优度较好的解可能被禁忌。通过使用解禁规则，让那些被禁忌的优良解有被选择的机会。一个

广泛使用的解禁规则是，如果某个被禁忌的解的优度比历史上找到的最好解的优度还要好，则认为解禁该解。

- **接受非改进邻居**。在最陡下降的局部搜索算法中，不接受非改进邻居。在禁忌搜索算法中，当搜索达到某个局部最优值的时候，此时所有邻居的优度都比当前解优度要差，但是搜索仍然从那些未被禁忌的邻居中挑选一个优度最好的邻居来替代当前解，并继续迭代过程。
- **停止标准**。在最陡下降的局部搜索算法中，当搜索达到局部最优值的时候算法停止。但是在禁忌搜索算法中，当搜索达到局部最优解的时候，它能够通过接受非改进邻居来越过局部最优值的陷阱达到某个更有前景(more promising)的地方。在禁忌搜索算法中，一般采用如下两种停止标准中的某种：(1)在搜索过程中，如果发现经过某个设定的迭代步数之后，找到的历史最好解仍未被改进，则搜索终止。这个设定的迭代步数叫做**搜索深度**。(2)当整个禁忌搜索过程所经历的迭代步数超过了某个设定的最大迭代步数，则搜索终止。

Input: 初始解s_0

Output: 搜索过程中找到的最好解

```
1 s ← s_0 ;                                   /* 产生初始解 */
2 初始化禁忌表;
3 repeat
4     找出s的邻域中未禁忌或符合解禁规则的优度最好的邻居s' ;
5     s ← s' ;
6     更新禁忌表;
7 until 达到停止标准;
8 return 整个搜索过程中找到的最好解;
```

算法 3: 禁忌搜索算法

在禁忌搜索算法中，对解或者动作的禁忌期限是决定搜索性能的一个关键参数。禁忌期限的大小决定了禁忌搜索算法的跳坑能力以及搜索解空间中某个区域的细致程度。一般而言，设定更长的禁忌期限使得搜索能够越过更高的局部最优值陷阱的壁垒。如果禁忌期限的设定的较小，则有可能算法仍被限制在解空间中的一个局部区域中；禁忌期限设置得太大则会使得邻域中的大部分解都被禁忌，从而失去了很

多找到优良解的机会。因此，禁忌期限不能太大，也不能太小，必须设定在某个合适的值。在许多问题上，一个合适的禁忌期限的大小与邻域的大小$|N(s)|$有很强的关系。一个参考经验是，将禁忌期限的大小取为邻域大小的$\frac{1}{4}$到$\frac{3}{4}$之间。一般有三种设定禁忌期限的策略：

- **静态的**。即将禁忌期限设定成某个固定值。实际的经验表明，这种策略过于死板，当问题的解空间具有复杂的地貌特性的时候，搜索不能灵活应对。
- **动态的**。即在搜索过程中动态的设定禁忌期限。一个通常的做法是，首先将禁忌期限设定一个挑选区域$[T_{min}, T_{max}]$，在每一次设定禁忌期限的时候，在这个区域随机取一个值。在很多问题上的经验表明，这种策略比静态的设定禁忌长度更好。
- **自适应的**。在这种策略中，通过观察搜索的行为特征而有针对性的设定禁忌期限。譬如说，如果看到某些解在搜索过程中经常重复出现，则感觉到可能搜索被陷在某个大的局部最优的陷阱中去了，因此加大禁忌的期限，使得搜索能够从这个陷阱中越出。在另外一些时候，如果看到解的搜索轨迹非常凌乱，则感觉到可能是因为禁忌期限设置的过大使得不能细致的搜索某一片区域，此时则应该减小禁忌期限。如何根据观察搜索的行为动态的设定禁忌期限是一个很深刻的重要的问题。使用这种策略自适应的设定禁忌期限的禁忌算法一般被称为**自适应禁忌搜索(Reactive Tabu Search)**[28]。

在禁忌搜索算法中还可以引入中期记忆(Mid-term memory)和长期记忆(Long-term memory)来调控算法的集中性和疏散性。中期记忆主要基于近因记忆(Recency memory), 它记忆搜索在最近的一段时间内，找到的优度较好的解所具有个公共的属性。在搜索过程中的某些时候，通过将这些属性固定，使得搜索被限制在优度较好的解的附近区域。通过中期记忆一般能够增强算法的集中性。长期记忆主要基于频率记忆(Frequency memory), 它通过记录在搜索历史中某些属性重复出现的次数，来区分解空间中频繁搜索的区域和很少经历过的区域。在搜索过程中，通过强制性的将解引入那些很少搜索过的区域，来增强算法的疏散性。

禁忌搜索已经被成功的用于求解许多困难的组合优化和全局优化问题。在今天，人们能不断的发现它的新的应用。相对于单调下降的局部搜索算法而言，禁忌搜索具有跳坑能力，找到的解的优度通常比单调下降的局部搜索算法要好。相对于模拟退火算法而言，禁忌搜索算法一般更稳定。在使用禁忌搜索算法求解实际的困难的

组合优化问题时，关键在于为问题设计合适的邻域结构，以及为禁忌搜索算法中禁忌期限的设置找到一个合适的灵活的策略。

2.3.5 迭代局部搜索算法

迭代局部搜索算法(Iterated Local Search)[26]是对局部搜索算法的一种改进。它在局部搜索算法陷入解空间中某个局部最优的陷阱中的时候，使用扰动算子(Perturbation operator)来对局部最优解进行扰动，使得搜索从局部最优值的陷阱中跳出到达某个更有前景的地方。虽然迭代局部搜索算法的流程非常简单，但是在求解许多经典的NP难度问题，譬如说旅行商问题，调度问题，Max-SAT等问题上，迭代局部搜索算法取得了巨大的成功，是当前最先进的(state-of-the-art)启发式优化算法。

迭代局部搜索算法包含三个主要模块：局部搜索过程、扰动算子和接受准则。局部搜索过程使得搜索从一个初始解出发达到与其邻近的局部最优解；扰动算子使得搜索跳出局部最优值的陷阱到达某个更有前景的地方；接受准则决定是否从当前的局部最优解移动到新的局部最优解。算法4给出了局部搜索算法的流程。在搜索过程中，通过对初始解s_0使用局部搜索过程得到第一个局部最优解s^*。随后，搜索进入一个迭代过程。在一轮迭代过程中，首先对当前的局部最优解s^*进行扰动得到一个新解t, 然后以t为初始解调用局部搜索过程得到一个新的局部最优解t^*, 最后使用接收准则来决定是否从当前的局部最优解s^*移动到新的局部最优解t^*。整个迭代过程不断地进行下去，直至达到某个停止标准。图2-4给出了迭代局部搜索算法的示意图[34]。

在迭代局部搜索算法中，局部搜索过程可以是单调下降的局部搜索算法，也可以是模拟退火算法或禁忌搜索算法。扰动算子一般是跟具体的问题相关的(problem-specific)。而接收准则一般有以下三种类型：

- **只接受变好的**。在此种接收准则中，只有当新的局部最优解比当前局部最优解更好的时候才接受它。在很多情况下，使用这种简单的接收准则已经能取得非常好的效果。
- **接受变好或稍微变坏的**。此时，接受t^*当且仅当$f(t^*) \leq (1+\epsilon) * f(s^*)$，其中$\epsilon$是一个正的小的实数。

Input: 初始解s_0

Output: 搜索过程中找到的最好解

$s \leftarrow s_0$; /* 产生初始解 */

$s^* \leftarrow$ LocalSearch(s) ;

repeat

  $t \leftarrow$ Perturbation(s^*, $history$) ;

  $t^* \leftarrow$ LocalSearch(t) ;

  $s^* \leftarrow$ AcceptanceCriterion(t^*, s^*, history) ;

until *达到停止标准*;

return 整个搜索过程中找到的最好解;

算法 4: 迭代局部搜索算法

- **类似模拟退火的接收准则**。此时，首先计算$\Delta E = f(t^*) - f(s^*)$，如果$\Delta E \leq 0$，则直接接受$t^*$；否则以概率$e^{-\frac{\Delta E}{T}}$接受$t^*$, 其中$T > 0$是退火温度，在搜索过程中逐渐减少。

迭代局部搜索算法的停止标准一般有以下三种：(1) 经过某个最大的迭代次数之后，找到的最好解仍未被改进，则算法终止。(2)整个迭代局部搜索过程的迭代次数超过了某个最大迭代次数，算法终止。(3) 当接受准则采用的是类似模拟退火的接受准则时，如果退火温度小于某个很小的值，则算法终止。

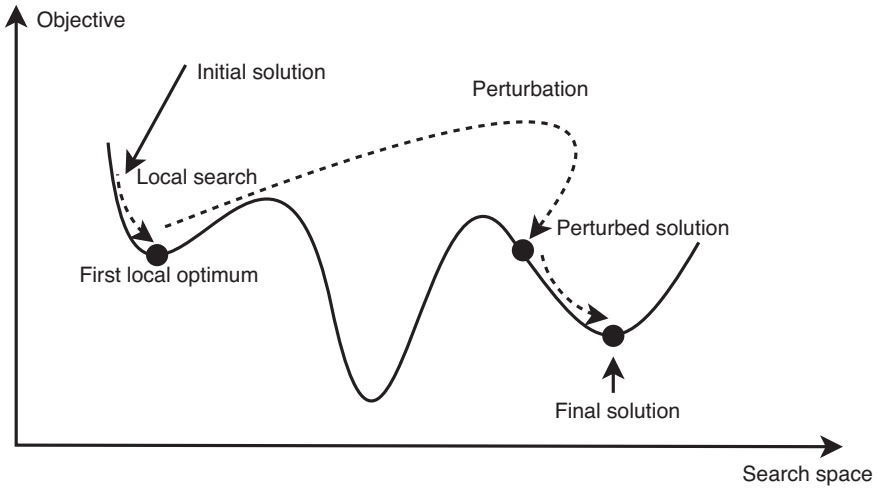

图 2-4 迭代局部搜索算法示意图

在迭代局部搜索算法中，对每个模块的调优以及模块间的配合决定了整个算法的性能。局部搜索过程决定了算法的局部寻优能力。一般而言，局部搜索过程的搜索能力越强，整个算法的效率也越高。扰动过程和接受准则决定了算法的跳坑能力。设计扰动过程时，特别需要注意控制**扰动强度(Perturbation strength)**。如果扰动强度过小，则经过局部搜索过程之后，搜索又回到当前的局部最优解，使得搜索不能越过局部最优的陷阱；如果扰动强度过大，则当前局部最优解的大部分信息都被破坏，使得迭代局部搜索算法的性能和多初始点的局部搜索算法(Multistart)的性能相近，无法体现出迭代局部搜索算法应有的性能。扰动过程应该和接收准则密切配合，使得整个算法在集中性和疏散性之间取得一个良好的均衡。

2.3.6 Memetic算法

Memetic算法是一种基于群体的(Population-based)启发式优化算法。与局部搜索算法相比，Memetic算法最显著的特征在于它在搜索过程中始终维护一个由$n(n \geq 2)$个不同的局部最优解组成的种群。在搜索过程中，算法不断地以种群为输入，使用交叉算符和局部搜索过程产生新解，然后用新解来更新种群。Memetic算法与遗传算法，Scatter Search等经典的群体搜索算法相比，流程更简单。在图染色、图划分、排课表等经典的NP难度问题上，Memetic算法是当前最先进的启发式优化算法[43]。

Memetic算法中包含了四个主要成分：产生初始种群过程、从种群中挑选父解产生子解过程、局部搜索过程以及种群维护过程。在搜索过程中，算法首先调用产生初始种群过程来得到一个初始种群。随后算法进入一个迭代过程，每一轮迭代过程包含三个步骤：首先挑选种群中的$k(k \geq 2)$个解并通过某种组合策略来产生新的解s, 然后以s为输入调用局部搜索过程得到一个新的局部最优解s^*, 最后使用种群维护过程来决定是否将s^*加入种群P中以及是否从P中移除某些解。整个迭代过程不断的进行下去，直至达到某个停止标准。Memetic算法的流程见算法5。

在设计Memetic算法时，一般来说，设计局部搜索过程和初始种群产生过程是比较容易的。局部搜索过程可以选用简单的单调下降的局部搜索算法，也可以选用更精巧的模拟退火算法或者禁忌搜索算法。产生有n个局部最优解的初始种群时，一种简单的做法是随机产生n个初始解，分别调用局部搜索算法，即可得到n个不同的

Input: 种群大小$|P|$
Output: 搜索过程中找到的最好解
$P \leftarrow$ GeneratePopulation($|P|$) ; /* 产生初始种群 */
repeat
  $s \leftarrow$ GenerateChild(P) ; /* 用种群产生子解 */
  $s^* \leftarrow$ LocalSerach(s) ; /* 用局部搜索改进子解 */
  $P \leftarrow$ UpdatePopulation (s, P) ; /* 更新种群 */
until *达到停止标准*;
return 整个搜索过程中找到的最好解;

算法 5: Memetic算法

局部最优解。也可以采用更细致精巧的做法，譬如说考虑种群的多样性。Memetic算法中比较难以把握，需要花费较大精力的两个模块是：从种群中挑选父解产生子解过程以及种群维护过程。值得指出的是，组合种群中的若干个父解得到子解的策略一般是跟问题相关的，需要保证在子解中能够继承若干父解的某些优良的特性，又能保证子解和父解之间有一定的多样性。而考虑种群维护策略时，一个关键点在于要保证种群有一定的多样性，防止种群过快收敛。Memetic算法的停止过程一般有三种: (1) 当超过某个最大的迭代次数后，搜索停止。(2)当找到的最好解经过某个最大的迭代次数仍未改进后停止。(3)当度量种群多样性的指标低于某个很小的值，搜索停止。

根据过去几十年人们在启发式优化算法上的研究经验，人们一般认为，基于群体的算法在搜索的疏散性上具有优势，而局部搜索算法在搜索的集中性上具有优势。Memetic算法的特征在于它同时具有了这两方面的优势：它既维护了一个多样性的种群来保证搜索的疏散性，又调用了局部搜索算法将解空间中的局部区域仔细搜索。因此，从算法架构上讲，Memetic算法在权衡搜索的疏散性和多样性之间具有天然的优势。在图染色、排课表等经典问题上的成功应用也证实了它的这个优点。值得注意的是，虽然Memetic算法在其架构上有优势，但是如果不考虑问题的特性，直接盲目的套用Memetic算法，只能得到一个低效的算法。在设计Memetic算法时，最关键的地方还在于将与问题本质结构紧密相关的知识融入到算法中去。

2.4 启发式优化算法的评价

经典的算法评价方法是分析算法的时间复杂度，即算法求解问题所需要的计算步数与问题规模n的函数。算法的时间复杂度越低，算法越好。所以时间复杂度为$O(n)$的算法要好于$O(n^2)$的算法；多项式时间的算法要好于指数时间的算法。经典的算法分析方法适用于分析精确完备的算法。但是对于NP难度问题，当前所有已知的精确算法都具有指数时间复杂度，只能用于求解问题规模很小的实例。启发式算法在很多情况下能够高效率的求解实际生活中出现的各种各样的NP难度问题。但是，经典的算法评价方法并不能用来评价启发式优化算法，主要有以下三个原因：(1) 启发式算法是不完备的，无法保证一定能找到问题的最优解，在很多时候，甚至连解的可行性都无法保证。(2) 启发式算法的具体表现与算例的特性有很强的关系，可能在有些算例上比较好，在另外一些算例下比较差。(3) 启发式算法中一般含有随机选择的策略，所以在不同的时间点运行同一个程序，得到的结果以及经历的计算时间并不相同。对于启发式优化算法需要另外的评价体系。

当前，对启发式优化算法的评价主要是基于经验分析(Empirical analysis)的方法[32,44]。在一些经典的NP难度问题上，有若干权威的学者和研究机构出面，挑选一些有代表性的算例(Instance)作为该问题的基准测试集(Benchmark)。一般而言，基准测试集中既包含从真实应用中产生的有结构的算例，也包含一些随机产生的算例。在具体的评价某个算法时，先将该算法用于求解基准测试集中的每个算例，记录相应的计算时间、找到的解的优度等相关信息。然后通过将该算法在Benchmark基准测试集上的表现与历史上已有的其它算法进行比较，来对算法进行评价。对于启发式优化算法而言，最重要的问题在于算法能否快速地、稳定地找到高质量的解。因此，在评价某个启发式优化算法时，最主要关注算法在以下三个方面的表现。

- **解的优度。** 即算法最终能够找到的解的优度。算法能找到优度越高的解，说明算法的搜索能力越强。在求解旅行商问题、图染色、背包问题、集合覆盖问题等经典NP难度问题上，解的优度是评价算法优劣的最首要和最核心的指标。有两种比较解的优度的方法。第一种是和客观的全局最优解相比，看看偏差有多大，偏差越小越好。但是在很多实际问题上，全局最优解是未知的，此时可以和当前人们用各种算法找到的已知最好解(Best-known solution)进行比较。在学术界，很多学者都认为如果某个算法在某个经典问题上能够找到

比当前已知最好解更好的解，那么该算法就具有非常高的价值。

- **计算时间。** 即算法找到某种优度的解所需要的计算时间。时间越短，说明算法的效率越高。值得指出的是，算法的搜索能力和算法的效率之间有很强的关系。在大部分情况下，算法的效率越高，其搜索能力也越强。
- **鲁棒性(Robustness)。** 关于启发式优化算法的鲁棒性，当前学术界没有明确的定义。一般而言，说一个算法是鲁棒的，是指该算法具有以下方面的特性：(1)在不同的时间运行算法，得到的结果优度变化不大。(2)在不同类型的算例上运行算法，算法的表现变化不大。(3)算法每次寻找到某种优度的解所需要的计算时间的变化不大。

值得注意的是，除了上述三个主要指标外，关于启发式优化算法还有以下一些常见的评价指标：算法是否简单明了、算法是否参数较少容易调校、算法是否容易实现、算法是否容易使用、算法的适用范围、算法能否求解大规模的问题实例等等。

基于基准测试集的分析和评价方法有如下优点[44]：

(1) 能一目了然地显示出真实的好算法，剔除出虚伪的“花架子”算法，引导算法研究在健康诚实的道路上发展。
(2) 简化了算法分析手段，有利于改进老算法，得出高性能的新算法，避免科学家长期的对低性能的算法做劳而无功的分析，浪费众多优秀学者的精力和时间。
(3) 雅俗共赏，东西方世界共赏，符合西方杜威主义的“彻底的实验室精神”，也符合东方普通人的“是骡子是马牵出去遛一遛即知”的正常感情。

2.5 本章小结

本章给出了优化问题及其解的严格定义，并简要介绍了启发式优化算法的理论背景、经典算法设计思路以及评价标准。

第一节用数学的方法对什么是优化问题给出了严格的定义，并定义了几个基本的概念：全局最优解、邻域以及局部最优解。

第二节介绍了启发式算法的两个理论背景：计算复杂性理论以及没有免费的午

餐定理。计算复杂性理论中的经验告诉我们，对于日常生活中经常出现的NP难度问题，可能根本不存在那种完备精确而又快速高效的求解算法，为了现实的求解这些问题，应该在算法的完备性以及解的优度上作出让步。没有免费的午餐定理告诉我们，优化算法的通用性和效率之间是有矛盾的，在为某个NP难度问题设计高效率的启发式优化算法时，应该具体问题具体分析，将与问题本质特性相关的信息融入到算法中去。

第三节介绍了一些经典的启发式优化算法。首先给出了局部搜索算法的基本概念，以及设计高效率的局部搜索算法所面临的三个重要问题：合适的邻域结构、快速的邻域评估方法以及搜索的集中性和疏散性的平衡。接着介绍了几种最著名的启发式优化算法，在局部搜索算法中介绍了单调下降的局部搜索算法、模拟退火算法、禁忌搜索算法以及迭代局部搜索算法；群体算法中介绍了Memetic算法。

第四节介绍了启发式优化算法的评价方法──经验分析方法，即将算法用于求解一系列公开的有代表性的测试算例组成的基准测试集，并记录找到的解的优度及其计算时间。在比较算法的性能时主要关注三个重要指标：解的优度、计算时间以及算法的鲁棒性。