目录

[1　引言 2](#_Toc470410307)

[2　启发式算法的基本概念和发展历史 2](#_Toc470410308)

[2.1　启发式算法的基本概念 2](#_Toc470410309)

[2.2　启发式算法的发展历程 3](#_Toc470410310)

[3　现代启发式算法的理论与方法 5](#_Toc470410311)

[3.1　现代启发式算法的分类 5](#_Toc470410312)

[3.2　现代启发式算法机制、特点及框架 5](#_Toc470410313)

[4　组合优化问题与方法 7](#_Toc470410314)

[4.1　组合优化问题 7](#_Toc470410315)

[4.2　组合优化问题的求解方法 9](#_Toc470410316)

[5　GCP算例与禁忌搜索实现 9](#_Toc470410317)

[5.1　GCP问题描述 9](#_Toc470410318)

[5.1.1　解的初始结构与领域选择 10](#_Toc470410319)

[5.1.2　目标函数的选择与构造 10](#_Toc470410320)

[5.2　禁忌搜索求解 11](#_Toc470410321)

[5.2.1　构造禁忌表及规则 11](#_Toc470410322)

[5.2.2　算法实现框架 12](#_Toc470410323)

[5.3　算例结果 12](#_Toc470410324)

[参考文献 13](#_Toc470410325)

**1　引言**

在计算机科学，人工智能以及数值优化领域，启发式算法主要指一类目的在于快速地获得特定问题上质量足够好的结果的算法。其中，快速和足够好是相对于完备算法而言的。在各种优化问题中，大规模实例上在短时间内获得最优解是不现实的，在这种情况下通常需要使用启发式算法来进行问题求解和完成优化任务。

**2　启发式算法的基本概念和发展历史**

**2.1　启发式算法的基本概念**

对于启发式算法（Heuristic Algorithms），有着不同的定义。一种定义为，一个基于直观或经验的构造的算法，对优化问题的实例能给出可接受的计算成本（计算时间、占用空间等）内，给出一个近似最优解，该近似解于真实最优解的偏离程度不一定可以事先预计；另一种是，启发式算法是一种技术，这种技术使得在可接受的计算成本内去搜寻最好的解，但不一定能保证所得的可行解和最优解，甚至在多数情况下，无法阐述所得解同最优解的近似程度。

有时候人们会发现在某些特殊情况下，启发式算法会得到很坏的答案或效率极差，然而造成那些特殊情况的数据结构，也许永远不会在现实世界出现。因此现实世界中启发式算法很常用来解决问题。而启发式算法的计算量都比较大，所以启发式算法伴随着计算机技术的发展，取得了巨大的成就。

启发式算法往往被应用于优化问题。对于优化问题的解决方法，有着确定方式（Exact Method）和近似（Approximate Method）方式两种方法。根据优化问题的条件和限制，可以描绘出一个解集空间（Solution Space）。确定方式的算法，只可用在解集空间很小的问题，对于NP hard问题，可行时间内在这个空间中找到全局最优解（Global Optimum）的可能性很小，几乎不可能。因此需要使用近似方法在有限时间内寻找一个近似最优解。近似方法又分为两种近似算法（Approximate Algorithms）和启发式算法（Heuristic Algorithms）。近似算法通常可得到一个有质量保证的解。然而启发式算法通常可找到传统解决问题的经验中找到寻求一种面向问题的策略，之后用这种策略在可行时间内寻找一个相对比较好的解，对解的质量有些情况并没有保证。

**2.2　启发式算法的发展历程**

随着计算理论的发展，启发式算法也在快速的发展之中。上世纪40年代由于实际的需要，人们已经提出了一些解决实际问题快速有效的启发式算法。自50年代中期创立了仿生学，许多科学家从自然界中寻求新的用于人造系统的灵感。一些科学分别独立地从生物进化的机理中发展出了适合于现实世界复杂问题优化的模拟进化算法，如进化规则、进化策略等，进化规则和进化策略的发展过程见图1、图2所示。



图1 进化规则的发展过程



图2 进化策略的发展过程

50年代启发式算法的研究逐步繁荣起来。随后，人们将启发式算法的思想和人工智能领域中的各种有关问题的求解的搜索方法相结合，提出了许多启发式的搜索算法。其中贪婪算法和局部搜索等算法受到人们的关注。60年代随着人们对数学模型和优化算法的研究越来越重视，发现以前提出的启发式算法速度很快，但是解得质量不能保证。虽然对优化算法的研究取得了很大的进展，但是较大规模的问题仍然无能为力（计算量相对当时的计算设备还是太大）。

70年代计算复杂性理论的提出。NP完全理论告诉我们，许多实际问题不可能在合理的时间范围内找到全局最优解。发现贪婪算法和局部搜索算法速度快，但解不好的原因主要是他们只是在局部的区域内找解，得到的解不能保证全局最优性。由此必须引入新的搜索机制和策略，才能有效地解决这些困难问题，这就导致了超启发式算法（Meta-Heuristic Algorithms）的产生。其中John Henry Holland（*Adaptation in Natural and Artificial Systems1975*）模拟地球上生物进化规律提出了遗传算法（Genetic Algorithm），它的与众不同的搜索机制引起了人们再次引发了人们研究启发式算法的兴趣，从而掀起了研究启发式算法的热潮，遗传算法的主要发展过程见图3。



图3 遗传算法的发展过程

80年代以后模拟退火算法（Simulated Annealing Algorithm），人工神经网络（Artificial Neural Network），禁忌搜索（Tabu Search）相继出现。最近，演化算法（Evolutionary Algorithm）， 蚁群算法（Ant Algorithms）， 拟人拟物算法，量子算法等油相继兴起，掀起了研究启发式算法的高潮。

**3　现代启发式算法的理论与方法**

**3.1　现代启发式算法的分类**

　　现代启发式算法的各种具体实现方法是相对独立提出的，相互之间有一定的区别。从历史上看，现代启发式算法主要有：模拟退火算法（SA）、遗传算法（GA）、列表搜索算法（TS）、进化规划（EP）、进化策略（ES）、蚁群算法（ACA）、人工神经网络（ANN）。图4给出了一种基于其方法的现代启发式算法的分类方案。



图4 基于其方法的现代启发式算法分类

这里讨论在算法流程上具有较大相似性的一类算法，如模拟退火算法、遗传算法、列表搜索算法和蚁群算法。模拟退火算法是Monte Carlo迭代求解的一种全局概率型搜索算法，具有区别于常规算法的搜索机制和特点，它是借鉴了热力学的退火原理建立起来的。遗传算法是借鉴“优胜劣汰”生物进化与遗传思想而提出的一种全局性并行搜索算法。进化规划和进化策略不像遗传算法注重父代和子代遗传细节而侧重父代与子代表现行为上的联系（强调物种层的行为变化）。列表搜索是一种具有记忆功能的全局逐步优化算法。蚁群算法是受到人们对自然界中真实的蚁群集体行为研究成果的启发而提出的一种基于种群的模拟进化算法，属于随机搜索算法。

**3.2　现代启发式算法机制、特点及框架**

模拟退火、遗传算法、列表搜索算法和蚁群算法的机制及特点如表1所示。从表1可看出各种现代启发式算法在优化机制方面存在一定的差异，但在优化流程上却具有较大的相似性，均是一种“领域搜索”结构。算法都是从一个或一组初始解出发，在算法的关键参数的控制下通过领域函数产生若干领域解，按接收准则（确定性、概率性或混沌方式）更新当前状态，而后按关键参数修改准则调整关键参数。如此重复上述搜索步骤直到满足算法的收敛准则，最终得到问题的优化结果。

表1 算法机制与特点

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算法类型 | 首次使用者 | 机制 | 优化流程 | 关键参数 | 收敛准则 |
| 模拟退火 | Kirkpartick·S（1983） | 基于Monte Carlo的全局概率型串行搜索优化算法 | 领域搜索 | 初始温度、退温温度、状态产生方式、抽样稳定准则 | 最大迭代步数、最优解连续不变的最大次数、搜索序列对应的平均值与最优值的最小偏差 |
| 遗传算法 | Holland J H(1992) | 基于生物进化与遗传思想的全局性并行优化算法 | 领域搜索 | 种群数目及复制、交叉、变异操作概率 | 同上 |
| 列表搜索 | Glover F  （1989） | 具有记忆功能的全局逐步优化算法 | 领域搜索 | 列表大小、领域函数结构与数量 | 同上 |
| 蚁群算法 | Dorigo M  （1991） | 具有强化学习功能的全局性并行优化算法 | 领域搜索 | 初始残留的信息量、信息量的增量、信息的积累量、启发式因子、信息量的消失因子 | 同上 |

现代启发式算法法与普通的搜索算法一样都是一种迭代算法，但是它们也有很大的区别：1）普通搜索算法是以一个解为迭代的初始值，而现代启发式算法是以一组解（种群Population）为迭代的初始值；2）现代启发式算法需要将问题的优化参数进行编码，映射为可进行启发式操作的数据结构，而普通搜索算法不需要此过程；3）普通搜索算法的搜索策略为确定性的，而现代启发式算法的搜索策略是结构化和随机化的（概率型）；4）现代启发式算法仅用到优化的目标函数值的信息，不必用到目标函数的导数信息，而普通搜索算法的大多数算法需要导数信息；5）现代启发式算法对问题的数学描述不要求满足可微性、性等条件，普通搜索算法对此有着较严格的要求；6）现代启发式算法具有全局优化性能、通用性强且适于并行处理的特点，普通搜索算法不具备这些优点。通过上述的比较可以看出，现代启发式算法的适用范围非常广泛、且算法均已相对成熟、易于修改，特别适用于大规模并行计算。

**4　组合优化问题与方法**

**4.1　组合优化问题**

在计算机科学和很多其他领域，组合优化问题都大量存在，例如人工智能，生物信息学，大规模集成电路设计，物流运筹，网络流量等等。这些问题的目标在于对于一个给定组合问题，从一个有限的合法解集合中找到一个全局最优解。例如，从若干城市序列中选择使巡游成本最低的路径（Traveling Salesman Problem旅行商问题）；对于若干布尔变量进行赋值使得公式为真（可满足问题）；在限定重量范围内选择若干物品使得价值最大化（背包问题）；从颜色集合中选择最少颜色使相邻顶点颜色不同（Graph Coloring Problem图着色问题）等等。这些问题的一个共性是寻找一个有限，离散的变量集合的子集，排序或者赋值，以期最大化或最小化特定目标。下面给出两个典型的组合优化作为实例。

1）旅行商问题：图5显示的是一个典型的旅行商问题（TSP）的实例投影示意图该实例由加拿大University of Waterloo维护。这个实例由美国49603个历史遗址的地理位置为原型。对应的，图6中连线表示已知的最优解。该解由William Cook、Daniel Espinoza、Marcos Goycoolea和Keld Helsgaun共同获得，该解对应路径长度为350201525米，即意味着要遍历所有的49603个历史遗址，至少需要走350201525米。

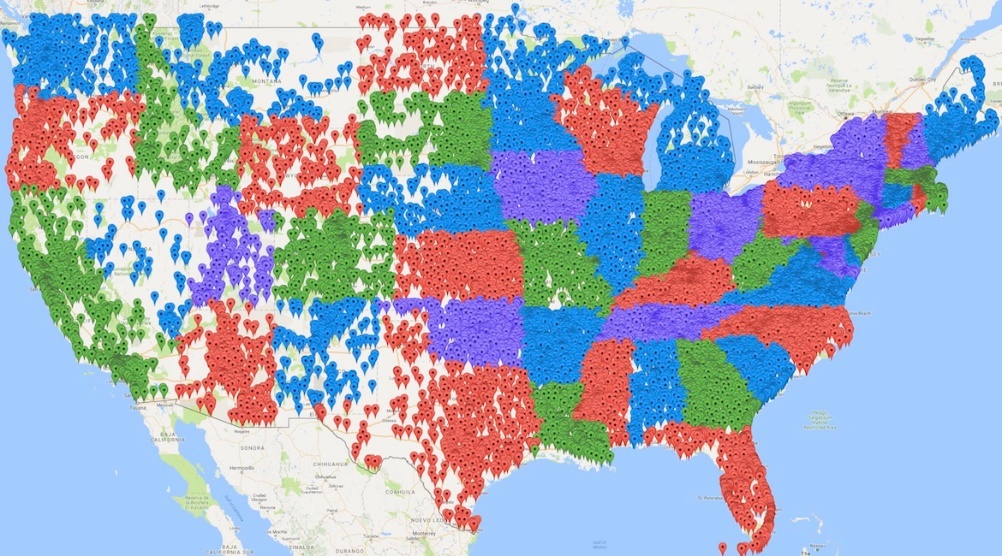


图5 以美国历史遗址为原型的TSP实例

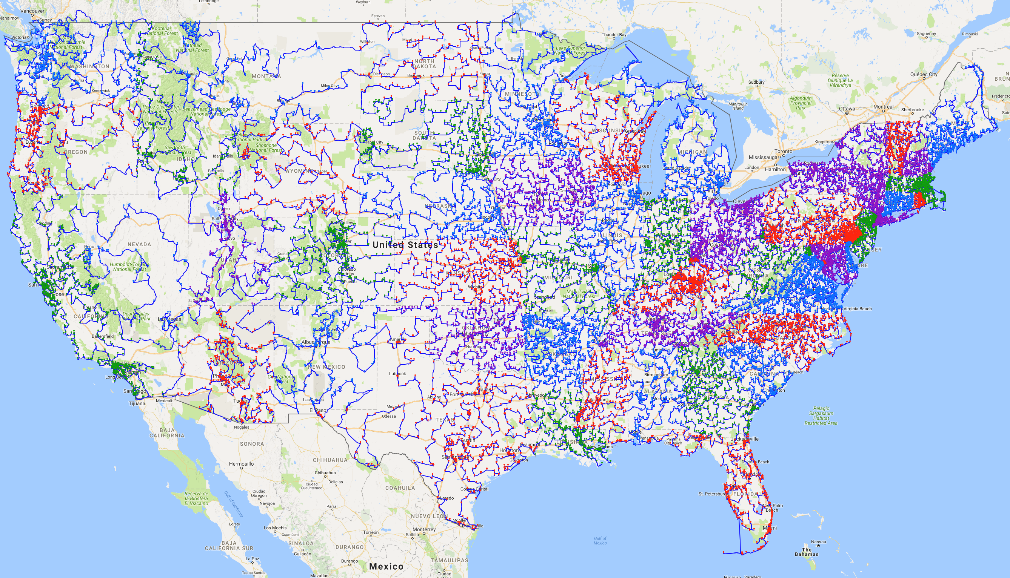


图6 US history TSP

TSP问题可以形式化的描述如下：

给定带图其中图的顶点集合为，且，边集为，边的权重函数为，中一条环路就是一条不重复访问中所有顶点的Hamilton环路，记为，其中且对于，，我们记为中所有环路的集合。

定义；

目标是寻找环路，使得。

不难看出，对于一个规模为*n*的TSP实例，总共有数量级的环路，因为每个城市的排序都能构成一个环路，即是说该实例有数量级的合法解。随着实例中城市数量的增加，合法解的个数以阶乘级速度增加。

2）0-1背包问题：背包问题是另外一种另外一种经典组合优化问题。该问题可以描述为：给定一组物品，每种物品都有自己的重量和价格，在限定的总重量内，如何进行选择，以使得物品的总价格最大化。背包问题及其变种在商业，组合数学，计算复杂性理论、密码学都有着广泛的应用背景。背包问题可以形式化地描述为：

给定*n*种物品，其中物品*j*的总量为，价格为，背包的最大称重量为W。如果每个物品只能选择0个或者1个，则该问题被称为0-1背包问题，该问题的目标在于最大化目标函数：

同时

**4.2　组合优化问题的求解方法**

作为求解组合优化问题的方法，历来的研究方法可以分为两类。其中第一类目的在于获得全局最优解。这类方法通常对搜索空间（所有合法解构成的集合）进行穷尽搜索。例如枚举法对每一个合法解进行质量评估，并保留具有最优目标函数值的解作为输出。分支限界搜索算法同样对解空间进行树形完备搜索，不过区别在于在搜索的过程中进行剪枝操作。如果在搜索过程中发现一个子树不可能存在全局最优解，则该子树上的解不会被评估。此外，完备算法还有算法等。

虽然这类算法可以保证当算法结束时能够得到全局最优解。然而，前面提到，很多组合优化问题的合法解规模随问题规模增长呈指数级速度增长。由此产生的问题通常形象地被称为组合爆炸问题。因此，在这类问题的大规模实例上使用完备搜索是不切实际的。在这种情况下，另一类算法应运而生。这类算法的目的在于在有限时间内获得满意质量解。这类算法中，按照发展顺序又可以细分为传统启发式算法以及元启发式算法。

其中，传统启发式算法的例子包括贪心算法，动态规划算法，爬山法等等。这类算法通常原理和实现方法简单，然而对应的缺点通常是质量无法令人满意，很容易陷入局部最优解等等。为了解决这些问题，在上世纪年代左右，开始出现一类算法，这类算法能够在一定程度上避免经典启发式算法的一些问题，即之前所介绍的模拟算法、遗传算法、禁忌搜索算法、蚁群算法。

**5　GCP算例与禁忌搜索实现**

**5.1　GCP问题描述**

根据图论研究，图着色问题大致可以分类为图的顶点着色、图的边着色、图的关联着色、图的全着色、还有图的混合型着色等，其中边着色、关联着色等问题都能转化成图的顶点着色问题来进行解决，而图着色方法中较为关键的一种着色方法是顶点着色，这也是最普遍的着色方法。图的顶点着色问题与现实生活息息相关，其关系着丰富的科学研究和复杂的工程应用，比如，课程表的安排、通信领域、作业调度等问题。图的顶点着色问题是经典的组合优化问题之一，怎么样来求解对图G着色的k值，是一个NP完全问题。在此利用禁忌搜索来解决图顶点着色问题。

5.1.1　解的初始结构与领域选择

禁忌搜索算法是一种启发式的图着色算法，其初始解的好坏对搜索的性能影响很大，需要根据问题而定，或随机给出，或事先用其他算法给出。为了数据结构的简化，我们设顶点集合为有序，各代表一个顶点，如对于顶点集V，编号设为。对于颜色集C，用各来表示。这样便可用一个行向量作为解的数据结构，即：

对应关系为

其中作为各顶点的编号，依次对应各个顶点，向量中各分量的下标对应所着色，着色相同的顶点即在同一划分集。

适当的移动规则的设计，是取得高效的搜索算法的关键。移动策略是算法从一个邻域空间到达另一个邻域空间的过程，也是产生新的初始解的方式，针对不同的问题可以采用不同的移动策略。鉴于初始解的数据结构，本文采用分量变化形式，即对于顶点*v*，其颜色值从变为，其中与属于颜色集。对于领域，每一个解S的邻居由那些满足上面的变化且只有一个分量变化的解组成，每一个分量可以选择*k*种颜色中的一种，那么对每一个解S的邻域由那些满足有且只有一个分量变化的解组成，即有*n\*k*个邻居。

5.1.2　目标函数的选择与构造

对于图顶点着色问题，目标函数的构造是一个难点。根据图着色问题的定义，对一个正常点着色，各个顶点与其相邻的顶点所着颜色不同，即在各个顶点与顶点之间，有边相连的顶点不在同一个划分集中，而没有边相连的顶点尽量在一个划分集中。由此可推导，如果有：

其中表示顶点集中包含的边数。那么选取目标函数为：

如果有，则是对于那些非正常点着色的可行解。在进行禁忌搜索时，每次从领域中以目标函数值最小为依据来选取解。在禁忌搜索完毕时，给出目标函数值最小的解，若为0则得到一个正常点着色方案，若不为0，我们可以得到一个较好的方案。

候选解集是为了方便找到下一个最优解，一般由邻域中的邻居组成，最为常用的方法是从当前解生成的邻域中选取若干个对象作为候选解集，并计算这些对象的目标函数或评估其价值，选出最佳的入选对象，也可以随机选取的方法实现部分邻居的选取。

**5.2　禁忌搜索求解**

5.2.1　构造禁忌表及规则

禁忌表在禁忌搜索算法中，是用来保存历史上近期的若干次迭代的记录，并对最近访问对象的禁忌状态作标记，在一次迭代次数内，算法不允许访问禁忌表中的对象，用这种策略来避免禁忌搜索陷入局部最优解而影响全局搜索能力。根据禁忌搜索算法初始解的数据结构，将禁忌表设为的矩阵，为禁忌表的长度，其第一列储存分量所在下标，对应图中的某点，第二列储存顶点的颜色值，即，第三列储存用来交换的颜色值。对于禁忌表长度，在实际运用时可以通过实验来比较选取该值。

特赦准则是与禁忌表相互对立统一的策略，有时算法运行中发现，在禁忌表中存在一个禁忌状态，其移动后可得到比历史最优解更好的解，此时可以把这个禁忌对象从禁忌表中解禁出来，特赦准则是对禁忌对象的一个例外。对于特赦准则，其设置的好坏直接影响进行全局最优化时的效果。考虑当前解、对应的候选集中的最优解若它被禁而同时它对应的目标函数值满足了，那么我们有理由将它解禁，因此得到更好的解。

若两目标值在多次循环中无改进，通常用最大允许迭代次数和目标值下界来避免搜索的过程中陷入死循环。

5.2.2　算法实现框架

第1步：按顺序读取算例，得出生成图的关联矩阵，初始化禁忌表、颜色数K和着色值；

第2步：while（K++），将第1步中得出的已知图的一种顶点着色方案作为禁忌搜索的当前最优解；

第3步：设置禁忌搜索解的最大迭代次数；

第4步：对于候选解集中的领域最好解，判断其是否满足特赦准则。若满足，则更新特赦准则，更新当前解，转第6步，否则继续；

第5步：在候选解集中选择不在禁忌表中的最好解更新为当前解；

第6步：更新禁忌表。

**5.3　算例结果**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Date | Instance | Duration | IterCount | Solution | | 2016-12-21 Wed 23:14:49 | DSJC125.1.col | 0.008 | 2475 | 5 | | 2016-12-21 Wed 23:14:49 | DSJC125.1.col | 0.009 | 2475 | 5 | | 2016-12-21 Wed 23:14:49 | DSJC125.1.col | 0.017 | 2475 | 5 | | 2016-12-21 Wed 23:14:49 | DSJC125.1.col | 0.011 | 2475 | 5 | | 2016-12-21 Wed 23:14:50 | DSJC125.1.col | 0.012 | 2475 | 5 | | 2016-12-21 Wed 23:14:50 | DSJC125.1.col | 0.007 | 1831 | 5 | | 2016-12-21 Wed 23:14:50 | DSJC125.1.col | 0.011 | 1831 | 5 | | 2016-12-21 Wed 23:14:50 | DSJC125.1.col | 0.013 | 1831 | 5 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.068 | 13418 | 8 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.067 | 13418 | 8 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.069 | 13418 | 8 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.067 | 13418 | 8 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.066 | 13418 | 8 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.068 | 13418 | 8 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.064 | 13418 | 8 | | 2016-12-21 Wed 23:14:50 | DSJC250.1.col | 0.07 | 13418 | 8 | | 2016-12-21 Wed 23:16:28 | DSJC250.5.col | 98.11 | 18591519 | 28 | | 2016-12-21 Wed 23:20:32 | DSJC250.5.col | 243.34 | 46717130 | 28 | | 2016-12-21 Wed 23:23:05 | DSJC250.5.col | 153.33 | 28626359 | 28 | | 2016-12-21 Wed 23:24:52 | DSJC250.5.col | 106.991 | 20290584 | 28 | | 2016-12-21 Wed 23:26:38 | DSJC250.5.col | 105.828 | 20146544 | 28 | | 2016-12-21 Wed 23:26:42 | DSJC250.5.col | 4.081 | 766824 | 28 | | 2016-12-21 Wed 23:27:30 | DSJC250.5.col | 47.572 | 9084418 | 28 | | 2016-12-21 Wed 23:28:36 | DSJC250.5.col | 66.225 | 12413645 | 28 | | 2016-12-21 Wed 23:28:37 | DSJC250.9.col | 1.003 | 150476 | 72 | | 2016-12-21 Wed 23:28:40 | DSJC250.9.col | 2.251 | 332513 | 72 | | 2016-12-21 Wed 23:28:41 | DSJC250.9.col | 1.256 | 161960 | 72 | | 2016-12-21 Wed 23:28:42 | DSJC250.9.col | 0.672 | 75453 | 72 | | 2016-12-21 Wed 23:28:42 | DSJC250.9.col | 0.688 | 77929 | 72 | | 2016-12-21 Wed 23:28:43 | DSJC250.9.col | 0.724 | 77929 | 72 | | 2016-12-21 Wed 23:28:45 | DSJC250.9.col | 2.166 | 360401 | 72 | | 2016-12-21 Wed 23:28:46 | DSJC250.9.col | 0.657 | 67752 | 72 | | 2016-12-21 Wed 23:28:55 | DSJC500.1.col | 8.987 | 2218120 | 12 | | 2016-12-21 Wed 23:29:09 | DSJC500.1.col | 14.082 | 3627852 | 12 | | 2016-12-21 Wed 23:30:07 | DSJC500.1.col | 57.369 | 14914371 | 12 | | 2016-12-21 Wed 23:30:37 | DSJC500.1.col | 29.757 | 7855159 | 12 | | 2016-12-21 Wed 23:33:41 | DSJC500.1.col | 184.476 | 48275204 | 12 | | 2016-12-21 Wed 23:34:40 | DSJC500.1.col | 58.715 | 15317624 | 12 | | 2016-12-21 Wed 23:34:49 | DSJC500.1.col | 9.097 | 2166378 | 12 | | 2016-12-21 Wed 23:35:02 | DSJC500.1.col | 12.032 | 2995128 | 12 | |

**参考文献**

1. Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simmulated annealing[J]. science, 1983, 220(4598): 671-680.
2. John H. Holland, Adaptation in natural and artificial systems[J]. 1992.
3. Glover F, Kelly J P, Laguna M. Genetic algorithms and tabu search: hybrids for optimization[J]. Computers & Operations Research, 1995, 22(1): 111-134.
4. Goldberg D E, Holland J H. Genetic algorithms and machine learning[J]. Machine learning, 1988, 3(2): 95-99.
5. Davis L. Handbook of genetic algorithms[J]. 1991.
6. http://www.math.uwaterloo.ca/tsp/index.html