



第四章 禁忌搜索(II)

Tabu Search, TS



学习内容

- 引言
- 示例
- 算法流程
- 算法特点
- 算法收敛性
- 算法设计
- 并行**TS**算法
- 算法应用



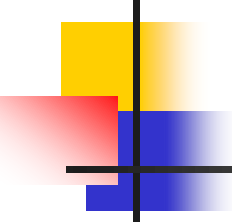
TS设计环节

- 初始解和适配值函数;
- 邻域结构和禁忌对象;
- 候选解选择;
- 禁忌表及其长度;
- 藐视准则;
- 集中搜索和分散搜索策略;
- 终止准则。



适配值函数

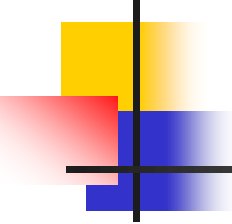
- 适配值函数用于对搜索状态进行评价，进而结合禁忌准则和藐视准则来选取新的当前状态。
- 显然，目标函数直接作为适配值函数是比较容易理解做法。当然，目标函数的任何变形都可作为适配值函数，譬如对极小化问题可将状态 x 的适配值 $f(x)$ 定义为 $M - c(x)$ 或 $e^{-ac(x)}$ ，其中 M 为一足够大正数， $c(x)$ 为目标值， $a > 0$ 。

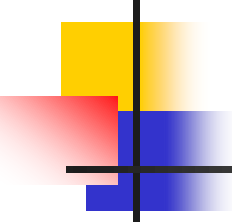
- 
- 若目标函数的计算比较困难或耗时较多，如一些复杂工业过程的目标函数值需要一次仿真才能获得，此时可采用反映问题目标的某些特征值来作为适配值，进而改善算法的时间性能。当然，选取何种特征值要视具体问题而定，但必须保证特征值的最佳性与目标函数的最优性一致。



禁忌对象

- 所谓禁忌对象，就是被置入禁忌表中的那些变化元素，而禁忌的目的则是为了尽量避免迂回搜索而多探索一些有效的搜索途径。归纳而言，禁忌对象通常可选取状态本身或状态分量或适配值的变化等。

- 
- 以状态本身或其变化作为禁忌对象是最为简单、最容易理解的途径。具体而言，当状态由 x 变化到状态 y 时，将状态 y (或 $x \rightarrow y$ 的变化)视为禁忌对象，从而在一定条件下禁止了(y 或 $x \rightarrow y$ 的变化)的再度出现。

- 
- 例：五城市TSP问题，设禁忌长度为3，以SWAP为邻域操作(解的起点固定为1)，规定 $x \notin N(x)$ ，候选解选当前解邻域解集中的最佳4个解，问题的距离矩阵如下：

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



第1步

- 当前状态为(1 2 3 4 5; 45), 其中45为目标值。此时禁忌表 $H = \emptyset$, 候选解集为{(1 3 2 4 5; 43)、(1 4 3 2 5; 45)、(1 2 5 4 3; 59)、(1 2 3 5 4; 44)}, 则选取(1 3 2 4 5; 43)为新的当前解, 并令 $H = \{(1 3 2 4 5; 43)\}$ 。



第2步

- 当前解为(1 3 2 4 5; 43), $H = \{(1\ 3\ 2\ 4\ 5; 43)\}$, 候选解集为 $\{(1\ 3\ 2\ 5\ 4; 43)$ 、 $(1\ 4\ 2\ 3\ 5; 44)$ 、 $(1\ 2\ 3\ 4\ 5; 45)$ 、 $(1\ 3\ 5\ 4\ 2; 58)\}$, 则选取(1 3 2 5 4; 43)为新的当前解, 并令 $H = \{(1\ 3\ 2\ 4\ 5; 43)$ 、 $(1\ 3\ 2\ 5\ 4; 43)\}$ 。



第3步

- 当前解为(1 3 2 5 4; 43), $H = \{(1\ 3\ 2\ 4\ 5; 43), (1\ 3\ 2\ 5\ 4; 43)\}$, 候选解集为 $\{(1\ 3\ 2\ 4\ 5; 43), (1\ 2\ 3\ 5\ 4; 44), (1\ 5\ 2\ 3\ 4; 45), (1\ 4\ 2\ 5\ 3; 58)\}$, 则选取(1 2 3 5 4; 44)为新的当前解, 并令 $H = \{(1\ 3\ 2\ 4\ 5; 43), (1\ 3\ 2\ 5\ 4; 43), (1\ 2\ 3\ 5\ 4; 44)\}$ 。



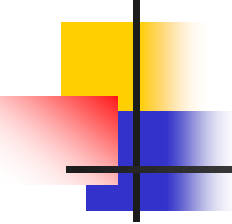
第4步

- 当前解为(1 2 3 5 4; 44), $H = \{(1 3 2 4 5; 43), (1 3 2 5 4; 43), (1 2 3 5 4; 44)\}$, 候选解集为 $\{(1 3 2 5 4; 43), (1 5 3 2 4; 44), (1 2 3 4 5; 45), (1 2 4 5 3; 58)\}$, 则选取(1 5 3 2 4; 44)为新的当前解, 并令 $H = \{(1 3 2 5 4; 43), (1 2 3 5 4; 44), (1 5 3 2 4; 44)\}$ 。



第5步

- 当前解为(1 5 3 2 4; 44), $H = \{(1\ 3\ 2\ 5\ 4; 43), (1\ 2\ 3\ 5\ 4; 44), (1\ 5\ 3\ 2\ 4; 44)\}$, 候选解集为 $\{(1\ 5\ 4\ 2\ 3; 43), (1\ 2\ 3\ 5\ 4; 44), (1\ 5\ 3\ 4\ 2; 44), (1\ 5\ 2\ 3\ 4; 45)\}$, 则选取(1 5 4 2 3; 43)为新的当前解。

- 
- 状态的变化包含了多个状态分量的变化，因此以状态分量的变化为禁忌对象将扩大禁忌的范围，并可减少相应的计算量。譬如，对置换问题，**SWAP**操作引起的两点互换意味着状态分量的变化，这就可作为禁忌对象；对高维函数优化问题，则可将某一维分量本身或其变化作为禁忌对象。
 - 对于上述例子：



第1步

- 当前状态为(1 2 3 4 5; 45), $H = \emptyset$, 候选解集为{(1 3 2 4 5; 43)、(1 4 3 2 5; 45)、(1 2 5 4 3; 59)、(1 2 3 5 4; 44)}, 则选取(1 3 2 4 5; 43)为新的当前解, 并令 $H = \{(2\ 3)\}$ 。



第2步

- 当前解为(1 3 2 4 5; 43), $H=\{(2\ 3)\}$, 候选解集为{(1 3 2 5 4; 43)、(1 4 2 3 5; 44)、(1 2 3 4 5; 45)、(1 3 5 4 2; 58)}, 则选取(1 3 2 5 4; 43)为新的当前解, 并令 $H=\{(2\ 3)、(4\ 5)\}$ 。



第3步

- 当前解为(1 3 2 5 4; 43), $H = \{(2\ 3), (4\ 5)\}$, 候选解集为{(1 3 2 4 5; 43)、(1 2 3 5 4; 44)、(1 5 2 3 4; 45)、(1 4 2 5 3; 58)}, 此时候选解集中有多个状态被禁忌, 因此选取(1 5 2 3 4; 45)为新的当前解, 并令 $H = \{(2\ 3), (4\ 5), (3\ 5)\}$ 。



禁忌长度和候选解

- 禁忌长度和候选解集的大小是影响**TS**算法性能的两个关键参数。
- 所谓禁忌长度，即禁忌对象在不考虑藐视准则情况下不允许被选取的最大次数(通俗些，可视为对象在禁忌表中的任期)，对象只有当其任期为**0**时才被解禁。候选解集则通常是当前状态的邻域解集的一个子集。
- 在算法的构造和计算过程中，一方面要求计算量和存储量尽量少，这就要求禁忌长度和候选解集的尽量小；但是，禁忌长度过短将造成搜索的循环，候选解集过小将容易造成早熟收敛，即陷入局部极小。



禁忌长度和候选解

- 禁忌长度的选取与问题特性、研究者的经验有关，它决定了算法的计算复杂性。
- 一方面，禁忌长度 t 可以是定常不变的，如将禁忌长度固定为某个数(譬如 $t=3$ 等)，或者固定为与问题规模相关的一个量(譬如 $t=\sqrt{n}$ ， n 为问题维数或规模)，如此实现很方便、简单。
- 另一方面，禁忌长度也可以是动态变化的，如根据搜索性能和问题特性设定禁忌长度的变化区间 $[t_{\min}, t_{\max}]$ (譬如 $[3, 10]$ 、 $[0.9\sqrt{n}, 1.1\sqrt{n}]$ 等)，而禁忌长度则可按某种原则或公式在其区间内变化。当然，禁忌长度的区间大小也可随搜索性能的变化而动态变化。



禁忌长度和候选解

- 一般而言，当算法性能动态下降较大时，说明算法当前的搜索能力比较强，也可能当前解附近极小解形成的“波谷”较深，从而可设置较大的禁忌长度来延续当前的搜索行为，并避免陷入局部极小。大量研究表明，禁忌长度的动态设置方式比静态方式具有更好的性能和鲁棒性，而更为合理高效的设置方式还有待进一步研究。
- 候选解通常在当前状态的邻域中择优选取，但选取过多将造成较大的计算量，而选取过少则容易造成早熟收敛。然而要做到整个邻域的择优往往需要大量的计算，如TSP的SWAP操作将产生 C_n^2 个邻域解，因此可以确定性或随机性地在部分邻域解中选取候选解，具体数据大小则可视问题特性和对算法的要求而定。



藐视准则

- 在禁忌搜索算法中，可能会出现候选解全部被禁忌，或者存在一个优于“**best so far**”状态的禁忌候选解，此时藐视准则将使某些状态解禁，以实现更高效的优化性能。在此给出藐视准则的几种常用方式。
 - 基于适配值的准则。
 - 全局形式(最常用的方式)：若某个禁忌候选解的适配值优于“**best so far**”状态，则解禁此候选解为当前状态和新的“**best so far**”状态；
 - 区域形式：将搜索空间分成若干个子区域，若某个禁忌候选解的适配值优于它所在区域的“**best so far**”状态，则解禁此候选解为当前状态和相应区域的新“**best so far**”状态。
 - 该准则可直观理解为算法搜索到了一个更好的解。



藐视准则

- 基于搜索方向的准则。
 - 若禁忌对象上次被禁时使得适配值有所改善，并且目前该禁忌对象对应的候选解的适配值优于当前解，则对该禁忌对象解禁。
 - 该准则可直观理解为算法正按有效的搜索途径进行。
- 基于最小错误的准则。
 - 若候选解均被禁忌，且不存在优于“**best so far**”状态的候选解，则对候选解中最佳的候选解进行解禁，以继续搜索。
 - 该准则可直观理解为对算法死锁的简单处理。



藐视准则

- 基于影响力的准则。
 - 在搜索过程中不同对象的变化对适配值的影响有所不同，有的很大，有的较小，而这种影响力可作为一种属性与禁忌长度和适配值来共同构造藐视准则。
 - 直观的理解是，解禁一个影响力大的禁忌对象，有助于在以后的搜索中得到更好的解。需要指出的是，影响力仅是一个标量指标，可以表征适配值的下降，也可以表征适配值的上升。譬如，若候选解均差于“**best so far**”状态，而某个禁忌对象的影响力指标很高，且很快将被解禁，则立刻解禁该对象以期待更好的状态。
 - 显然，这种准则需要引入一个标定影响力大小的度量和一个与禁忌任期相关的阈值，无疑增加了算法操作的复杂性。同时，这些指标最好是动态变化的，以适应搜索进程和性能的变化。



禁忌频率

- 记忆禁忌频率(或次数)是对禁忌属性的一种补充,可放宽选择决策对象的范围。譬如,如果某个适配值频繁出现,则可以推测算法陷入某种循环或某个极小点,或者说现有算法参数难以有助于发掘更好的状态,进而应当对算法结构或参数作修改。在实际求解时,可以根据问题和算法的需要,记忆某个状态的出现的频率,也可以是某些对换对象或适配值等出现的信息,而这些信息又可以是静态的,或者是动态的。
- 静态频率信息主要包括状态、适配值或对换等对象在优化过程中出现的频率,其计算相对比较简单,如对象在计算中出现的次数,出现次数与总迭代步数的比,某两个状态间循环的次数等。显然,这些信息有助于了解某些对象的特性,以及相应循环出现的次数等。



禁忌频率

- 动态的频率信息主要记录从某些状态、适配值或对换等对象转移到另一些状态、适配值或对换等对象的变化趋势，如记录某个状态序列的变化。显然，对动态频率信息的记录比较复杂，而它所提供的信息量也较多。常用的方法如下：
 - 记录某个序列的长度，即序列中的元素个数，而在记录某些关键点的序列中，可以按这些关键点的序列长度的变化来进行计算。
 - 记录由序列中的某个元素出发后再回到该元素的迭代次数。
 - 记录某个序列的平均适配值，或者是相应各元素的适配值的变化。
 - 记录某个序列出现的频率等。



禁忌频率

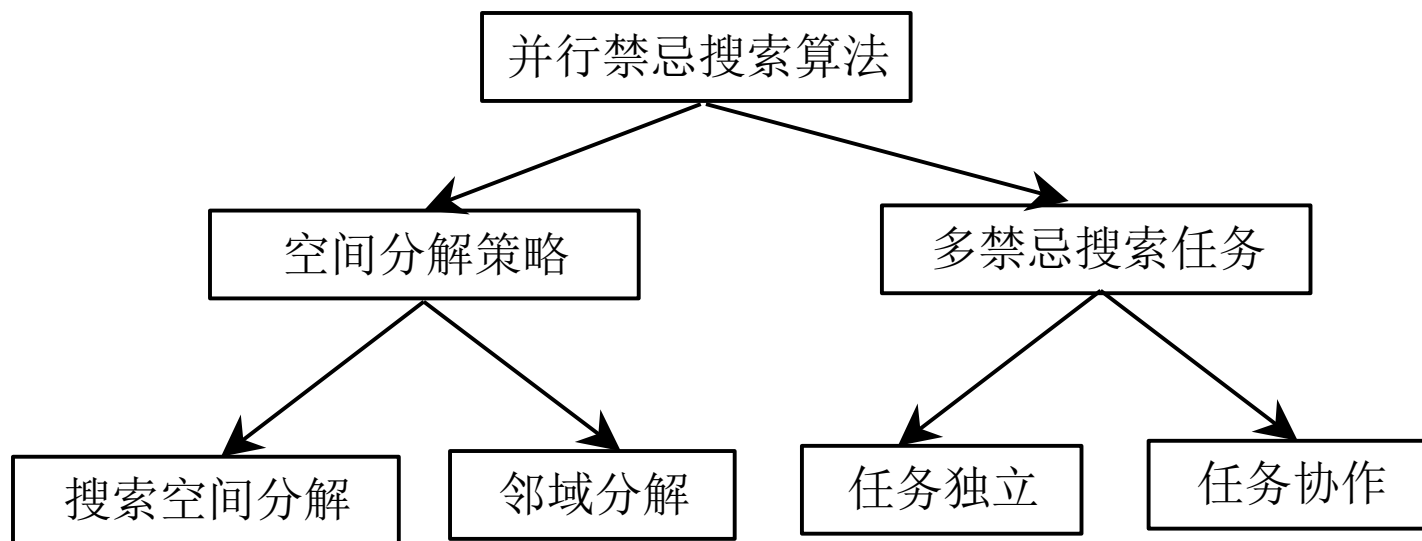
- 上述频率信息有助于加强禁忌搜索的能力和效率，并且有助于对禁忌搜索算法参数的控制，或者可据此对相应的对象实施惩罚(如前文示例)。譬如，若某个对象频繁，则可以增加禁忌长度来避免循环；若某个序列的适配值变化较小，则可以增加对该序列所有对象的禁忌长度，反之则缩小禁忌长度；若最佳适配值长时间维持下去，则可以终止搜索进程而认为该适配值已是最优值。
- 此外，许多改进的禁忌搜索算法还根据频率等信息在算法中增加集中搜索(intensification)和分散搜索(diversification)机制，以增强算法的搜索质量和效率。其中，集中搜索机制强调算法对优良区域的重点搜索，如基于最优或次优状态进行重新初始化或进行多步搜索，加强对应取得最优状态的算法参数的被选取概率等；分散收缩机制则强调拓宽搜索范围，尤其是那些未探索的区域，这与增强遗传算法种群多样性有些类似。显然，集中搜索和分散搜索在某些层面上是矛盾的，而两者对算法性能都有很大影响，好的TS算法应当具有合理平衡集中搜索与分散搜索的能力，这也是许多论文实现对禁忌搜索算法性能改进的突破口。



终止准则

- 与SA、GA一样，TS也需要一个终止准则来结束算法的搜索进程，而严格实现理论上的收敛条件，即在禁忌长度充分大的条件下实现状态空间的遍历，这显然是不切合实际的，因此实际设计算法时通常采用近似的收敛准则。常用方法如下：
 - 给定最大迭代步数。此方法简单易操作，但难以保证优化质量。
 - 设定某个对象的最大禁忌频率。即，若某个状态、适配值或对换等对象的禁忌频率超过某一阈值，则终止算法，其中也包括最佳适配值连续若干步保持不变的情况。
 - 设定适配值的偏离幅度。即，首先有估界算法估计问题的下界，一旦算法中最佳适配值与下界的偏离值小于某规定幅度时，则终止搜索。

并行TS算法



参数(初值、禁忌表长度、候选解个数等)设置可相同或不同



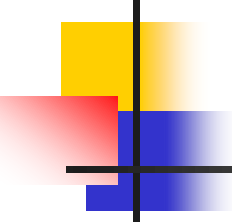
基于空间分解的并行策略

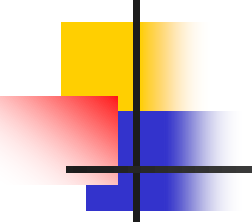
- 搜索空间的分解策略。即通过搜索空间分解将原问题分解为若干子问题，各子问题用不同的禁忌搜索算法进行求解，从而实现并行化。
- 邻域的分解策略。即每一代中用多种方法对邻域分解所得的各子集进行评价，从而实现对最佳邻域解的搜索的并行化。
- 显然，这类基于空间分解的并行策略在实施时对同步的要求很高。



基于多禁忌搜索任务的并行策略

- 顾名思义，这类并行策略中包含多个禁忌搜索算法在运行，而各算法可以使用相同或不同的算法参数(如初值、禁忌表长度、候选解个数等)。同时，各任务可以以不存在通讯的独立方式运行，也可以以协作的方式运行(譬如最优解的共享)。
- 上述分类中，空间或邻域分解策略具有问题依赖性，仅对部分问题适用，而多任务策略比较通用，当然分解策略和多任务策略也可以结合使用。此外，在多处理机情况下，鉴于各任务的数量和定位相对并行机的负荷状态，如静态或动态，又可将并行禁忌搜索分为以下三类。

- 
- 非自适应方式。指任务的数量和定位在编译时就生成的情况，且各任务相应的处理机的定位在算法运行过程中是不变的，即静态调度方案。譬如，根据处理机的个数将邻域分解成相应数目的子集。但是，当各处理机的负荷严重不平衡时，必然造成部分处理机的长时间空闲而影响算法的整体搜索效率。需要指出的是，目前所提出的并行禁忌搜索算法大部分仍属于这类运行方式。

- 
- 半自适应方式。指任务的数量在编译时就固定，而定位却在运行时确定或改变的情况。其目的是提高非自适应并行方式的性能，其手段则是在处理机间动态地重新分配负荷以实现负荷动态平衡。
 - 自适应方式。指任务的生成完全是动态变化的情况，如当处理机空闲时则自动生成任务，而当处理机繁忙时则取消任务。**Talbi等(1998)**提出了一种自适应的并行禁忌搜索算法，算法由并行而独立的子禁忌搜索算法构成，且各子算法基于不同的参数进行初始化，而各串行任务间不需要通讯，并通过对典型QAP问题的高效求解验证了算法的有效性。



TS的具体实现

- 基于TS的组合优化
- 基于TS的函数优化



基于TS的组合优化

- 以Flow-shop为例：
 - 初始解：随机产生，或借助NEH
 - 邻域结构：SWAP等
 - 候选解集：基于SWAP的K个邻域解
 - 禁忌表长度：固定值
 - 藐视准则：best so far解的判断
 - 集中搜索：一定步数后基于最优解重新初始化
 - 分散搜索：一定步数后随机初始化
 - 终止条件：最优解滞留步数



基于TS的函数优化

- 参见文献（注意禁忌判断的区别！）