## 变量

可见时间窗（VTW）是卫星对地面目标成像的一段时间。

观察窗（OW）定义了成像过程的持续时间。

由于卫星与地面控制中心的联系有限，指令无法真实的上传。该算法采用离线调度的方式，在指定的时间点前对请求进行收集、分析和调度，并将生成的调度上传到卫星上。

问题的输入是指用户的观测请求和预定卫星的能力。令是任务的集合，其中是要调度的任务的数量。这里，请求被认为是任务。对于每个请求，我们定义

,任务优先级，衡量的是通过完成任务能获得的收益

,用户要求的最低图像质量，

,任务的指定角度，俯仰角

,请求的持续时间，决定了图像的大小

,任务图像应该被获取的时间

卫星的能力参数定义如下：

,存储器的最大容量，用总观察时间(s)来定义

,光学相机的锥角，与视野范围相关

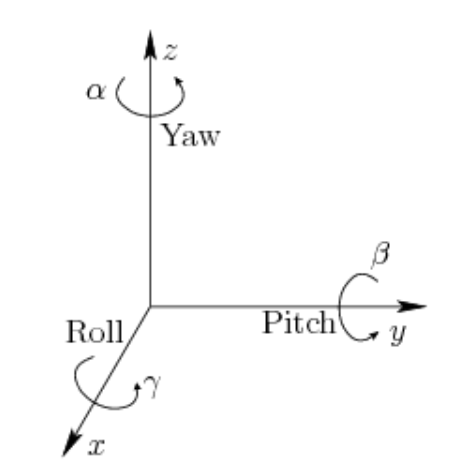
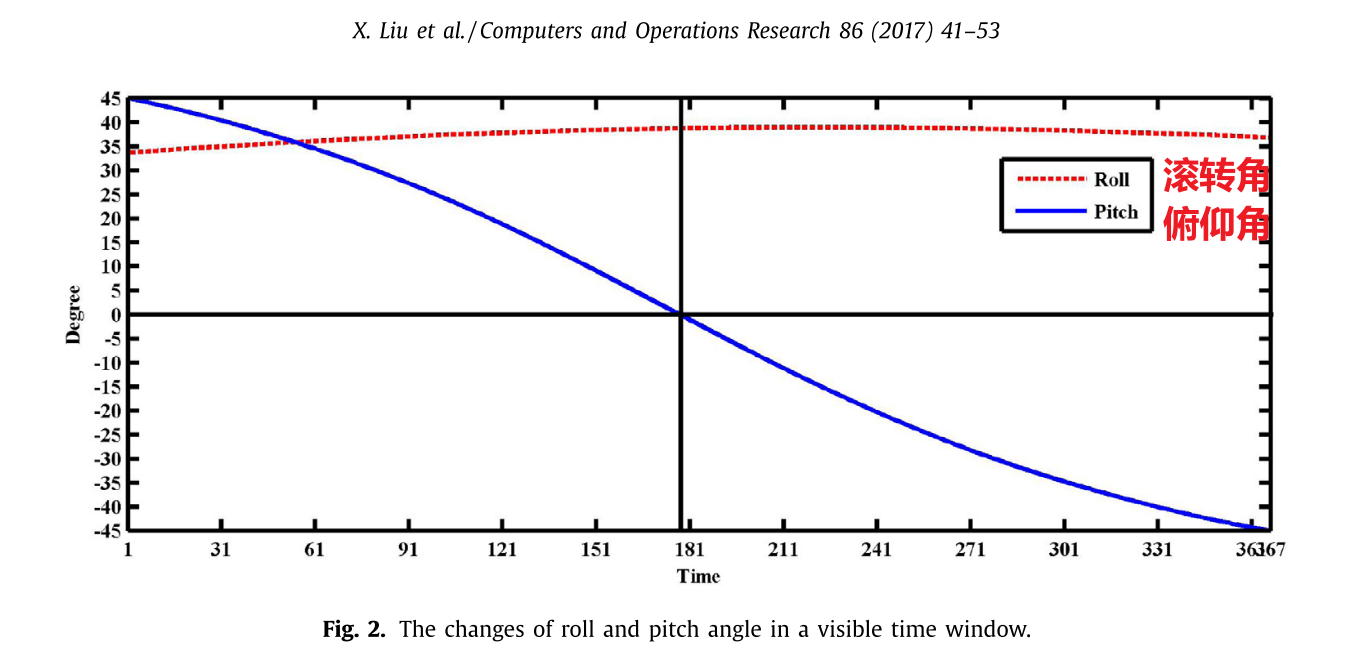
,相机的焦距，影响分辨率

最大横滚角、最大俯仰角、最大偏航角，共同确定卫星的移动性

调度问题输入的是卫星和目标的VTW，VTW时间不等（几十秒到两三分钟） 表示任务的所有可见时间窗口的集合，表示任务的第个时间窗口。

:时间窗口的开始和结束时间，

，指时刻的瞬时夹角，是时间窗口中的时刻。（卫星与目标之间的相对位置是变化的，导致卫星必须进行调整以捕获目标，视角因此随时间变化）



为计划任务的集合，任何的，确定一个时间窗口来观察，并确定何时执行观察（VTW比OW时间长得多，OW对应VTW的一部分）

是任务的观测窗口，开始时间，观测长度，图像角度，结束时间，根据和，可以从其角度序列中检测出瞬时图像角度，从而得到，

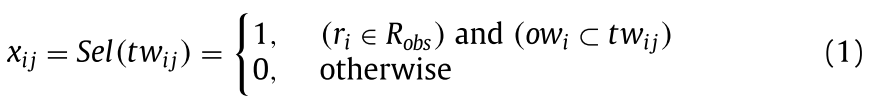
如果被调度且观测窗口为，可以记作当且仅当和同时成立。

引入轨道的概念，太阳同步轨道是光学卫星最常用的轨道，定义了时间间隔，该间隔从卫星飞出地球影子开始，到再次进入地球影子结束。只有卫星处于阳光下才能进行观测。根据卫星轨道轨迹的预测结果，将调度范围划分为许多轨道，代表调度范围的第k个轨道，是在该轨道上观察到的任务集合，一个调度范围内的所有轨道构成一个集合

## 数学公式

超额订阅问题（请求的数量>可以服务的数量），基本目标是最大化总收益

二元决策变量 表示是否选择来自任务的可见时间窗口，如果全为0，则任务失败



目标函数（这里单个任务分了多个TW，而我们的单个任务只有一个TW）

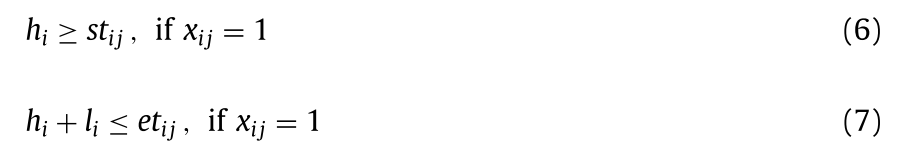


约束条件

每个请求最多被观察一次



OW必须被包括在VTW内（观测必须在任务对应的可见时间窗口内完成）



任务与任务之间必须存在转换时间 此处固定为

任务消耗的存储量必须小于卫星最大存储容量

调度可以按照观测的开始时间排序的观测窗口的集合来表示

## 解决方案

必须要做两个决策：

1. 选择可以完成的任务，（文章中需要决定哪个VTW作为任务的候选时间窗口，但是我们的情况是一个任务只有一个VTW，所以决定VTW的过程就自定省略了），通过约束最大化收益来实现选取
2. 为每个确定的VTW选择适当的观察时间

两个决策是动态耦合的。

## ALNS结构

Step01 初始化所有参数，生成一个可行的初始解，并计算其奖励，将赋给当前解以及最优解，以及

Step02 从当前解移动到其邻域中的新解。选择破坏因子从当前解中移除一些任务，产生破坏了的解，之后选择修复因子通过向被破坏的解中添加为调度的任务来修复被破坏的解，生成新的当前解。

Step03 若新的当前解优于旧的当前解，, 然后将其与当前最优解比较。如果优于当前最优解，则使用新的当前解覆盖最优解，

若新的当前解差于旧的当前解，则使用模拟退火算法验收标准来确定是否接受新的当前解

Step04 更新操作因子的权重：

Step05 如果满足停止条件，则输出当前最佳解，否则回到Step02

### 解的初始化

因为ALNS对初始解并不敏感，所以使用贪婪启发式算法构造一个初始解。

为了创建卫星的观测序列时间线，我们首先按照收益的降序和开始时间的升序对所有的VTW排序，收益作为主要顺序。遍历排序后的VTW列表，尝试将每一个VTW插入到当前的调度中，当列表中的所有VTW都被访问后，我们得到了一个观察窗口的时间顺序，它定义了问题的初始解。

VTW的长度总是不小于OW长度，所以要在VTW中选择时间开始观测。有两种策略，一个是最早的开始时间，即选择一个可见时间窗口的开始时间作为观测的开始时间；另一个是基于图像质量的。事实上，当卫星经过目标的正上方时获得的图像是最好的。因此，观测的开始时间将选择在其可见时间窗口的中间。

### 破坏算子

破坏方法从给定的解中选择移除q个时间窗口，将其保存在任务库Q中，Q中的任务将是下次插入的候选任务。将所有未排定的任务保存在一个集合中。和Q中包含的任务将被重新插入以产生一个新的解决方案。

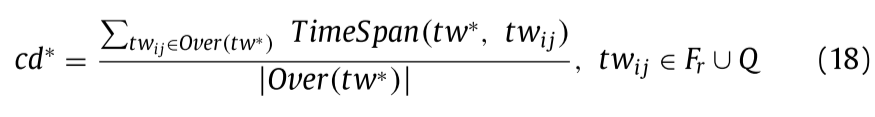
破坏方法一共有六种。

**随机移除：**给定的解中随机选择q个任务并删除。

**最小收益优先移除：**（贪婪启发式）迭代的移除收益最低的任务，直到任务库满为止

**最高成功率优先移除：**（参考论文）论文中讲的是优先移除具有最多VTW的任务（具有VTW的数量越多，说明该任务被接受的可能性越大）但是我们的情况是，一个任务只有一个VTW，所以我想的是优先移除最长的VTW的任务，若所有任务的VTW一样长，则不考虑该方法

**冲突移除：**冲突度，表示一个时间窗口是否与其他时间窗口重叠



当前时间窗，是与重叠的时间窗集合，计算重叠的长度，若值大于0，，则说明有重叠。冲突度是指当前时间窗口与未计划时间窗口冲突的平均时间。此方法优先移除冲突度最高的时间窗。

**集群移除1：**选择任务数量最少的轨道，移除所有任务。若数量不足以填满任务库则再次重复此过程，直到任务库填满。

**集群移除2：**选择利润和最小的轨道，移除所有任务。如果所选轨道中的任务编号不等于|Q|，则应用集群移除1.

### 修复算子

修复算子向被破坏的解中插入新任务来改进解。待插入的候选任务由两部分组成，未安排的任务集合以及已删除的任务集合。

修复方法一共三种：

贪婪插入：对所有候选任务按照利润降序排序，依次满足任务插入。

最小成功率优先插入：论文中讲的是优先安排VTW较少的任务进行插入（VTW越少则被接受的可能性就越小）但是我们的情况是一个任务只有一个VTW，所以我想的是优先选择VTW最短的任务来进行插入，若所有任务的VTW一样长，则忽略此方法。

最小冲突插入：优先选择冲突程度最小的任务进行插入，（按照冲突度升序排列随后一次进行插入）

## 自适应策略设计

目标是为了选择更有效的算子对

### 算子的权重更新（论文写的不详细故弃用）

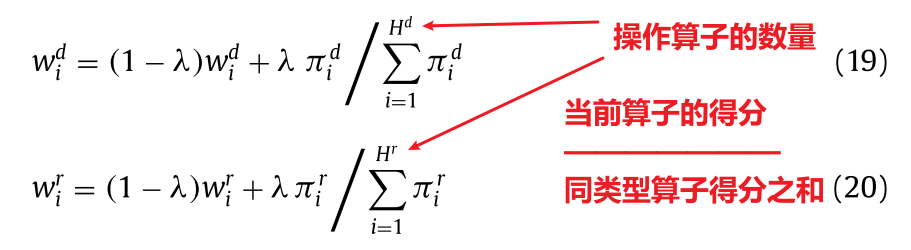
该算法被划分为若干段，每个段的迭代次数为ξ次。操作符在每次迭代时的得分和权重仅在分段结束时更新一次。所有运算符的分数在每个分段结束时重置为其初始值。一旦选择了算子，其分数将根据以下规则增加：

：生成新的全局最优解

：生成的新解优于旧解，但差于当前最优解

：生成的新解差于旧解，但被接受

：生成的解被拒绝



最开始的迭代中，权重的初始值都为1，所以使用的算子都是随机选择的，在前20次迭代中，需要通过强制选择以确保所有的算子都会被调用

### 权重更新（使用此方法）

每进行一次destroy和repair，都会更新所使用的方法的得分以及权重，得分初始值为1分

：生成新的全局最优解

：生成的新解优于旧解，但差于当前最优解

：生成的新解差于旧解，但被接受

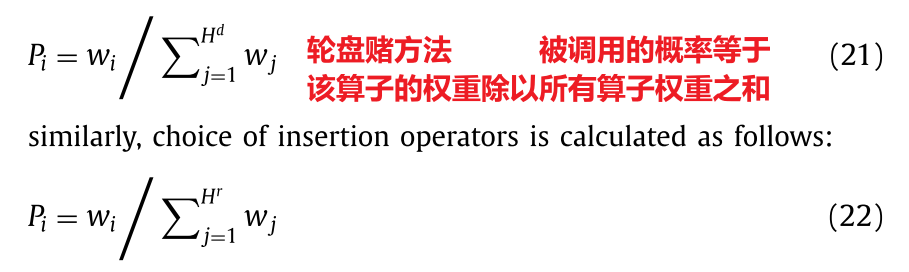
：生成的解被拒绝

[]=

根据得分和使用的次数来更新权重

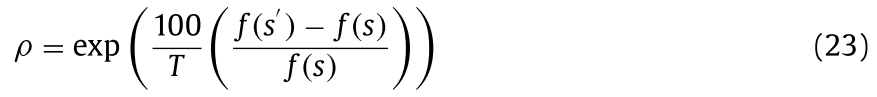
### 算子的选择

每次都要选择破坏算子和修复算子，独立选择，共同工作。



### 接受标准

SA模拟退火



如果，则用新的当前解覆盖旧的当前解

否则，覆盖的概率如上

第n次迭代的温度是

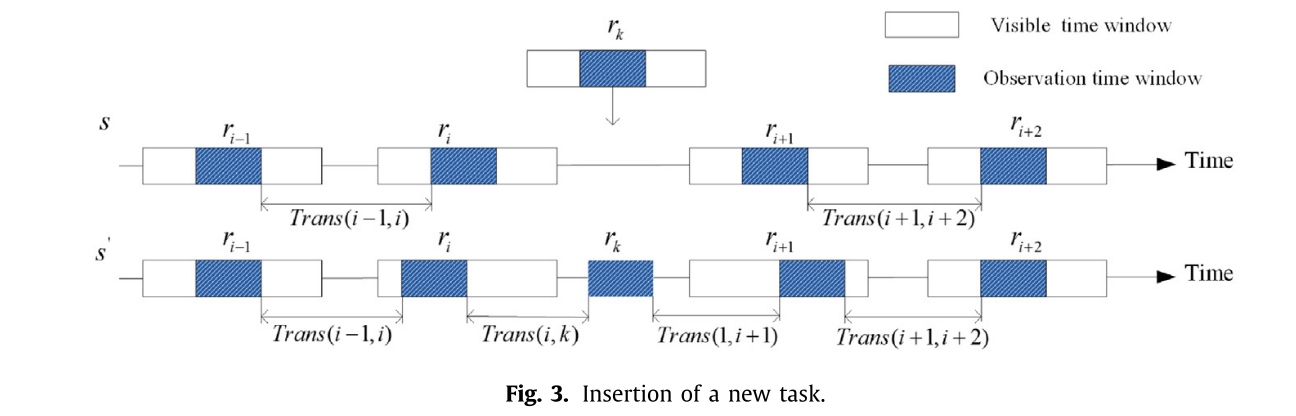
初始温度设置如下



停止标准，满足其中一条即可：1达到最大迭代次数；2最大运行时间；3所有任务都被安排

## 基于时间松弛度的快速插入算法

将候选任务插入到给定解中需要考虑空缺位置前驱任务和后继任务，（因为转换时间的存在，插入任务可能导致过渡时间更改的传播，即为了满足当前任务的插入而向前移动前驱任务，向后移动后继任务，前驱任务的前驱任务、后驱任务的后驱任务又要继续向前移动或向后移动）为了避免这一过程，引入基于时间松弛度的快速插入算法。

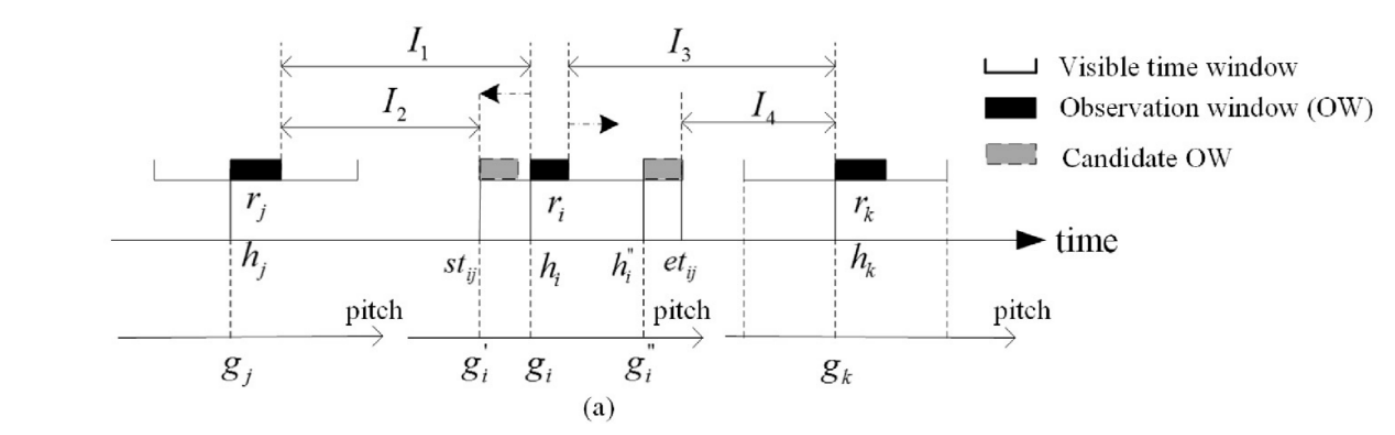


### 时间松弛度的计算

需要确定一个任务能不能插入的衡量标准。

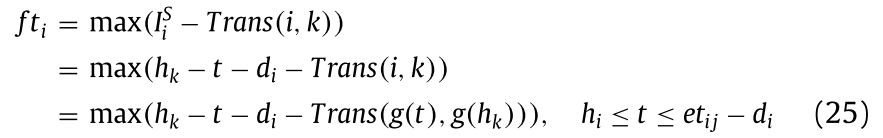
向前时间松弛FTS：衡量的是从路线的第一个顶点出发的出发时间可以向前移动多少而不会导致路线变得不可行。

可以类似：为保证计划调度的可行（为了避免过渡时间更改的传播），计算每个任务可以在其VTW中可以移动的最大长度。分为向前向后，



将移动到其VTW的末端，时间由变为如果，那么该移动可以被接受。(向后移动为啥是个人感觉应该是) 其中

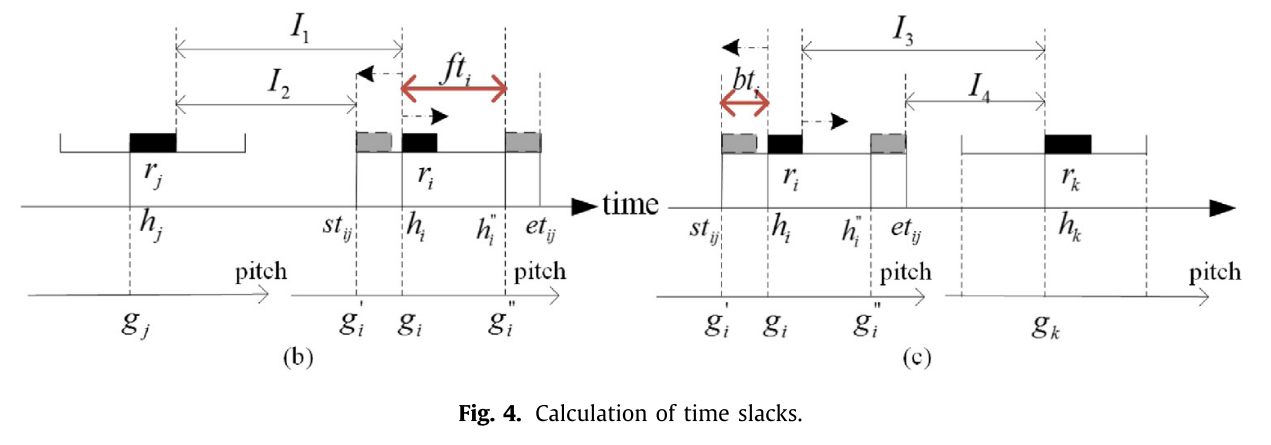
用表示任务和后继任务之间的时间间隔，计算方式为K任务的开始时间减去I任务的结束时间。



就是说时间间隔减去转换时间就是该任务的空余时间，时间松弛度，由任务的开始时间决定

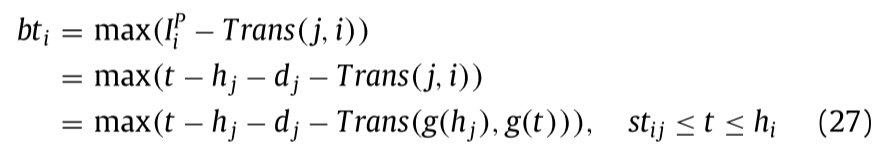
这样计算的前提是 插入任务的位置由后继任务才行，如果没有，则等于 如下图





用表示任务和前驱任务之间的时间间隔，计算方式为

时间间隔减去转换时间就是该任务的空余时间，时间松弛度，由任务的开始时间决定



### 快速插入算法

可以计算出每个任务的时间松弛度，

插入任务时，同一轨道并在插入位置之前的任务可以向左移动，之后的任务可以向右移动，以确保可以插入。

给定一个候选任务，插入到中产生,我们可以计算其累计的向前和前后的松弛度

观测窗口向右的灵活性是由统一轨道的所有后继任务决定的，累计向前松弛度即，所有后继任务松弛度之和，向后松弛度同理。

如果时间间隔加上累积的时间灵活性大于对应的过渡时间，则可以插入观察。？？？？？？？？？能插入的标志到底是什么？？？文章写得不对！！！！

简化插入过程：使因为插入而改动的任务数量达到最小，定义来控制受影响的任务数量。

快速插入：在等待插入的任务自身的VTW中移动任务，以找到合适的开始时间，确保合法。如果没有合适的插入时间，就根据其前驱任务和后继任务的松弛度来移动前驱任务和后继任务。

插入算法：

Step01：对于要插入的任务，选择一个开始时间选择策略，并选择开始时间，如果合法（不违反过渡时间的），直接将插入到解中，插入完成跳转Step04，否则跳转Step02

Step02：根据时间间隔和，前驱任务和后继任务的Trans，在待插入任务的VTW中移动已得到新的合法位置，插入后得到，转到Step4，否则存在至少一个时间间隔不允许插入，转到Step03

Step03：移动的前驱任务和后继任务，为插入营造条件。

Step03.1：固定，并检查，如果只有合法，则跳转Step03.2，如果只有合法，则跳到Step03.3，如果都不合法，则跳到Step03.4

Step03.2：如果成立，即后继任务的时间松弛度大于转换时间减去间隔时间，即后继任务的时间松弛度可以满足欠缺的部分，使用二分法算法在VTW中移动，目标是实现的插入，如果成功插入，转到Step04，失败则返回解，插入结束。

Step03.3：如果,成立，即前驱任务的时间松弛度大于转换时间减去间隔时间，即前驱任务的时间松弛度可以满足欠缺的部分，处理办法与Step03.2类似。

Step03.4： 成立，则同时移动前驱任务和后继任务来达成插入的条件。成功转Step04，否则返回解，插入结束。

Step04：记录插入任务的位置并返回新的解，插入结束

二分法：为了满足插入，必须移动前驱或后继任务，这里举移动前驱任务的例子。需要移动来达成可以插入的条件，确保 等价于,简单地说就是向左移动，来扩大和之间的空隙，使得空隙不小于转换时间就满足了插入条件。

Step01：设为当前当前可行的开始时间。设,可以在其中移动，如果的长度，则跳到Step02，否则返回

Step02：令,计算,如果，返回，如果，则更新,，如果则重复Step02，否则返回，如果,转到Step03

Step03：设，并更新，设, 计算, 如果，返回，如果，则更新,，如果则跳到Step02，否则返回，如果,重复Step03

我的理解：

Step01先判断区间是由具有二分条件，如果没有直接返回

Step02 先取区间中点，区间拿右边，然后判断

如果则中点刚好符合条件直接返回；

如果，则说明有多余空间，判断如果新区间仍然可以二分，则继续二分寻找准确点，若不满足二分条件，则返回；

如果，则说明区间中点不符合插入条件，跳到Step03拿左边区间。

最后确定的值越小（越靠左）说明向左移动的越大，也就越大，也就越有可能不小于，以实现插入。