# 文档

[min-sum 1](#_Toc2030283677)

[初始解 1](#_Toc1610092156)

[Vns算法 1](#_Toc377430045)

[min-max 4](#_Toc1947556724)

[初始解 4](#_Toc640112694)

[Vns算法 4](#_Toc1628460235)

## min-sum

min-sum的入口文件在ccvrp\_md.cc中，在main函数中指定了数据所在文件夹。

### 初始解

算法采取Regret Insertion算法生成初始解。

### Vns算法

Vns算法包括Local Search、Shake、Local Search的循环。在经历WHILE\_NUM（vns.cc line 22，当前值8）次循环搜索结果没有改进后算法停止，过程中获取的min-sum值为最终结果。

#### Neighbour Reduction

##### 基础

在Local Search过程中会涉及到将某个节点移动到另一个位置，针对排列组合过多的情况，这里采取了Neighbour Reduction。当选择了一个节点，只有其将要插入的位置前后两个节点中有节点与此节点的距离较近（属于其Neighbour）才可能插入此位置，否则此转换不成立。每个节点的Neighbour的由离此节点距离最近的5%节点构成（5%在vns.cc，line 130可设置）。

##### Potential Depots

Neighbour Reduction还考虑到Potential Depots的问题。一个节点只能被插入到这个节点的potential depots出发的路径中。Potential depots的确定由每个depot到此节点的相对距离决定。距离最近的depot一定是potential depot，而其余depots按距离短至长的顺序，如果节点到前一个depot的距离除以节点到当前depot的距离大于0.7（vns.cc line 190可设置）且前一个depot属于potential depots，则当前depot也是potential depot，否则当前depot不是potential depot。

##### Potential Depots Extend

除了基本的potential depots外，这里还引入了基于平面图形“前、后”概念的potential depots。如果两个depot与节点的路径夹角小于等于30°（vns.cc line 227可设置），则只有离节点最近的depot才是potential depot。

#### Local Search

Local Search有五个算子：

1. TwoSwap：任意交换路径中（同一路径或者不同路径）的两个节点；
2. TwoOptSwap：在路径排列中，给定两个节点，翻转这两个节点之间的路径；
3. RelocationMove：将路径中任意节点插入到另一位置；
4. ArcNodeMove：将路径中某个位置的连续两个节点移动到另一位置；
5. ArcNodeSwap：将路径中某个位置的连续两个节点与路径中另一个节点互换位置。

Local Search过程会依次选择两个路径中的位置，执行上述某个算子得到新的路径排列，如果此排列符合路径要求，则是原路径的Neighbour。

另外，Local Search中会进行depot的search，即将某条路径起始和结尾的depot替换为其它depot，如果替换后的结果比替换前要好，则将当前路径改为替换后的路径。（武汉数据将不会执行depot search）

Local Search时会依次搜索以上五个算子的neighbours，搜索时当找到第search\_better\_depth（vns.cc line 267，当前值为 3000/(客户数\*仓库数) 至 9000/(客户数\*仓库数) 之间的一个随机数）个improve则将其作为下一次搜索起始路径，如果所有neighbours都搜索完而没有找到第search\_better\_depth个improve，这赋值为所找到的improve中的最好值。

如果连续max\_no\_improve次（vns.cc line 445，当前值为6至14之间的一个随机数）Local Search（每个算子算一次）都没有找到更好的解，则Local Search停止，搜索到的最佳解即为当前解。

#### Shake

Shake有三个算子： 1. TwoSwap：任意交换路径中（同一路径或者不同路径）的两个节点； 2. TwoOptSwap：在路径排列中，给定两个节点，翻转这两个节点之间的路径； 3. RelocationMove：将路径中任意节点插入到另一位置。

Shake将从路径中随机选择两个位置执行上述算子。

一次Shake操作会连续运行shake\_times（vns.cc，line 447，当前值为2至min(路径数, 10)之间的随机数）次随机选择的算子。

Shake后会对shake后的路径做Local Search，这个过程会循环shake\_max\_no\_improve（vns.cc，line 446，当前值为3至14之间的随机数）遍，如果过程中找到了更好的结果，则计数清零，重新开始计数。

## min-max

min-max的入口文件在ccvrp\_md\_minmax.cc中，在main函数中指定了数据所在文件夹。

### 初始解

算法采取Regret Insertion算法生成初始解。

### Vns算法

Vns算法包括Local Search、Shake、Local Search的循环。在经历WHILE\_NUM（vns\_minmax.cc line 19，当前值18）次循环搜索结果没有改进后算法停止，过程中获取的min-max值为最终结果。

#### Neighbour Reduction

##### 基础

在Local Search过程中会涉及到将某个节点移动到另一个位置，针对排列组合过多的情况，这里采取了Neighbour Reduction。当选择了一个节点，只有其将要插入的位置前后两个节点中有节点与此节点的距离较近（属于其Neighbour）才可能插入此位置，否则此转换不成立。每个节点的Neighbour的由离此节点距离最近的5%节点构成（5%在vns\_minmax.cc，line 160可设置）。

##### Potential Depots

Neighbour Reduction还考虑到Potential Depots的问题。一个节点只能被插入到这个节点的potential depots出发的路径中。Potential depots的确定由每个depot到此节点的相对距离决定。距离最近的depot一定是potential depot，而其余depots按距离短至长的顺序，如果节点到前一个depot的距离除以节点到当前depot的距离大于0.7（vns\_minmax.cc line 228可设置）且前一个depot属于potential depots，则当前depot也是potential depot，否则当前depot不是potential depot。

#### Local Search

Local Search过程会选择cost最大的子路径中的节点将其优化到另一条路径，使得最大cost的子路径cost变小。当最长路径优化没有效果后，会尝试第二长路径优化，然后再次进行最长路径优化。这个过程会循环执行search\_better\_while（vns\_minmax.cc line 268可设置，当前值为10）次。

另外，Local Search中会进行depot的search，即将某条路径起始和结尾的depot替换为其它depot，如果替换后的结果比替换前要好，则将当前路径改为替换后的路径。（武汉数据将不会执行depot search）

如果连续max\_no\_improve次（vns\_minmax.cc line 507，当前值为6至14之间的一个随机数）Local Search（每个算子算一次）都没有找到更好的解，则Local Search停止，搜索到的最佳解即为当前解。

#### Shake

Shake将从最长子路径中随机选择一个节点并将其放入到起始点是此节点potential depot的任意路径中。

一次Shake操作会连续运行shake\_times（vns\_minmax.cc，line 509，当前值为2至min(路径数, 10)之间的随机数）次。

Shake后会对shake后的路径做Local Search，这个过程会循环shake\_max\_no\_improve（vns\_minmax.cc，line 446，当前值为3至14之间的随机数）遍，如果过程中找到了更好的结果，则计数清零，重新开始计数。