假设子问题的最优值为，如果则表示存在不满足公式（5）的运输路径，将这条路径加入到限制主问题的运输路径集合中，继续迭代；如果，则表示限制主问题已经达到最优解。

上述模型中虽然保证了各个节点之间流平衡的关系，但是可能会导致出现子回路的情况出现，图3所示：

C

D

A

S

B

1

1

1

3

3

图3 子回路案例

S-A-B-S为一条回路，C-D-C为一条回路，不满足车辆从S出发，遍历所有路径的要求，因此需要生成subtour-elimination约束来排除这条路线。

车辆路径问题的执行逻辑为：

步骤1解子问题

步骤2判断是否有子回路。

步骤3若无子回路则输出；若有子回路则进行步骤4

步骤4若有子回路，生成subtour-elimination约束，返回步骤1

判断是否有子回路的方法为检验从起点出发是否能走过所有有出流的节点，首先生成简化路线图，例如图3对应简化图如下

C

D

A

S

B

图4 子回路简化图

生成节点集合Node。从S出发，探索其后置节点，将探索过的节点放入集合Visited，直到某一节点的后置节点全部存在于集合Visited，例如图4 集合Visited最后为{S, A, B}，集合Node为{S, A, B, C, D}。若集合Visited等于集合Node，则无子回路；若不等则有子回路。

上图判断出有子回路，则需要添加subtour-elimination约束，如下：

其中为此循环的解。添加此约束，则在下次求解中排除这条含有子回路的车辆路径。

反复循环，直到找到一个不含子回路的路径则得到现阶段最优路径。