在按需充电方式下可以考虑两种情况：

(1)MCV 的任务缓冲池接收到的充电请求为满后随即出发为这一系列需要充电的节点进行服务；

(2)MCV 在收到充电请求后随即出发为需要充电的电单车进行充电，在充电的过程中动态选择下一个充电节点。

针对(1)中的情况：

对 MCV 设定缓冲池阈值，MCV 从接收到第一个充电请求开始计数，一直到事先设定好的阈值之后，MCV 随即开始一轮充电。设 MCV 缓冲池的最大值N_{max} 与最小值N_{minthr}。N_{maxth} 是指最多可以为几个电单车服务，由于 MCV 在移动过程中所耗能量要远大于用于充电的能量，因此N_{maxth} 相当于 MCV 最远可以移动的距离，$N_{maxthr}=\frac{E_M}{q_M}$。$N_{minth}$ 指最少可以为几个电单车进行充电，可以设置一个固定值，阈值计算过程如下：

$$\gamma_i=\frac{n_i}{\sum_{i=1}^{S}n_i}(\text{参数})$$

其中，S 为不同服务站，n_i指一段采样时间内，服务站接收到的充电请求总数，对于服务站 i 所涵盖的回路中，其区域内 MCV 的阈值缓冲池大小为：

$$N_{thr}(i)=\lceil\gamma_i(N_{maxth}-N_{minthr})\rceil+N_{minthr}$$

针对(2)中的情况：

对电单车设置两个能量阈值，E_L和E_T，$E_L<E_T$，当电单车的剩余能量低于E_T时，电单车发送充电请求给 MCV，当电单车剩余能量低于E_L时，发送 ALERT 信息给 MCV，表明自己的能量已经达到警示值，同时停止工作，变为静态点。MCV 在接收到第一个 ALERT 信息之后开始一轮充电。由于E_L要比E_T重要，所以将E_T设定为固定值，E_L通过动态变化求解，通过数值仿真求解，利用单位时间内电单车的平均能量消耗 average energy consumption rate(AECR)与平均电单车失效率，这里的失效含义定义为完全没有电量，average failure probability(AFP)两个量以求取E_L。通过数值仿真求取E_L变化过程中 AECR 与 AFP 变化情况，可以得到两个近似线性关系：

$$\text{AECR}(E_L)\approx A_1E_L+B_1$$
$$\text{AFP}(E_L)\approx A_2E_L+B_2$$

利用数值仿真中求解A_1，A_2，B_1，B_2

通过最小化 AECR 和 AFP 的值作为目标，引入权重因子 w，采用最小二乘法优化方法中的平方和公式进行优化，如下：

$$\text{f(w)}=(AECR)^2+w(AFP)^2$$

最优E_L是能够最小化上述目标函数的值，其值如下：

$$E_L(w)=\frac{A_1B_1+wA_2B_2}{(A_1)^2+w(A_2)^2}$$

此处需要说明电单车的工作充电机制：

当电单车的能量达到一定阈值下限后，向服务站发送充电请求，服务站准备调度 MCV 出发为请求充电电单车进行充电

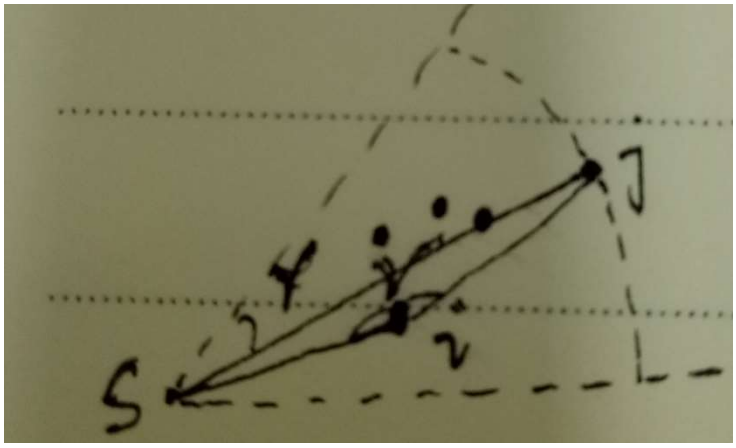
- 针对(1)情况：MCV 在准备出发时为所有将要服务的电单车发送 CONFIRM 信息，接收到 CONFIRM 信息的所有电单车将停止使用，因此在接收到 CONFIRM 信息之前的电单车仍然可以被使用；
- 针对(2)情况：MCV 每次在确定好将要为之服务的下一个电单车后会给它发送 CONFIRM 信息，接收到 CONFIRM 信息的电单车将停止使用。第一种情况是 CONFIRM 一对多发送，第二种情况是 CONFIRM 一对一发送。

按需充电模式下的充电可调度性决策条件

按需充电方式下路径可调度性的充分必要条件：(1) $p_c \times \eta > p_{sum}$；(2) $v_M > \frac{max_{1\le i\le n} D\times q_c\times\eta\times p_i}{q_c\times\eta - p_{sum}}$；(3)$E_M \ge p_{sum} \times \frac{D\times q_C}{V_M\times(q_C\times\eta - p_{sum})} + D \times q_M$

电单车选择性插入算法

情况描述：当 MCV 选定电单车N_i作为目标充电节点后，在移动至N_i的路途上收到电单车N_j发出的 ALERT 信息，这说明N_j也急需充电。由于 MCV 在工作时的能量消耗主要集中于移动能耗以及充电能耗，而充电能耗要远小于移动能耗，所以在考虑电单车选择性插入时主要考虑移动距离。



说明：当 MCV 从 S 处出发要行驶向 j 处为之服务，中途遇到 i 发出请求，判断是否为 i 服务的标准是判断 si 与 ij 之间的夹角是否接近于 180 度，因为

$$d_{sj} = \sqrt{d_{si}^2 + d_{ij}^2 - 2d_{si}d_{ij}\cos\gamma} \approx \sqrt{d_{si}^2 + d_{ij}^2 - 2d_{si}d_{ij}} = d_{si} + d_{ij}$$

因此如果原始是选择从电单车 i-j 进行充电，中途遇到 k，则构建 i-k 以及 k-j 的真实路径，可以得到路径长度，原始路径长度 i-j 的也可以得到，构建三角形 ikj，已知三边长度，可以得到任意两边的夹角，判断该夹角是否大于某个阈值角度，如果大于就可以插入，反之就不能插入。

电单车选择充电算法

在(1)情况下，每次是已经确定为哪几个节点进行充电，可以直接采用周期性充电方式来做。在(2)情况下，可能某一时刻有多个电单车为 MCV 发送充电请求，MCV 每次在确定好将要为之服务的下一个电单车后会给它发送 CONFIRM 信息，接收到 CONFIRM 信息的电单车将停止使用，而其它发送了充电请求的电单车在发送充电请求后仍可局部使用，当剩余能量在达到E_L后停止使用，固定在当下位置。

目标：在第二种情况下时对于一系列发送充电请求的电单车确定为哪一个电单车进行服务。

需要考虑的因素：发送充电请求的电单车在过去一段时间内的使用频率 k（这是由于在某一时刻接收到的充电请求可能某一电单车是经过了一个月才需要充电，某一电单车是 3 天就需要充电了，所以考虑充电频率是很必要的）；发送请求的电单车距离 MCV 的距离 d；剩余能量e_i；

MC 出发点 depot 部署与充电回路分配

假设：已经构建好充电回路，每个 depot 有 x 个 charging pile，每个 charging pile 可服务的回路条数是 Y 条，那么一个 depot 至多可服务 $x \cdot Y$ 条充电回路。

Depot 就是 S 位置，其初始位置集合 $D = \{d_1, d_2, ...d_n\}$

1）对于每个d_i，构造集合P(d_i)表示d_i可以服务的充电回路个数；
2）对于P(d_i)和P(d_j)，如果P(d_i) ∈ P(d_j)，那么可以删除d_i，用d_j来代替。
优化问题需满足条件：
1）每个充电回路至少可由一个 MC depot 进行负责；
2）每个充电回路只能分配给一个 depot。

第一步：贪心列选择

目的：选择最小数量的列以保证每一行的总和大于等于 1，在这种情况下表明充电回路至少可以由一个 depot 服务。

步骤：将每一列所有元素相加，称之为该 depot 所对应的权重，该权重值越高表明这个 depot 的服务能力越强；2）选择具有最大权重值的一列，移除该列以及该列所覆盖的行，然后更新剩余的行和列；3）再次计算权重值，并重复上述步骤。

第二步：贪心充电回路分配

目的：上一步得到的是每个回路可用的 depot 待选集合，第二步主要用于将每个充电回路分配给一个最佳的 depot 服务。

步骤：属于 depot d_i的充电回路定义为 load(d_i)，这称之为 depot d_i的负载，当一条充电回路P_j可以由多个 depot 所服务时，需要一个优先方法来选择最佳的 depot 分配给P_j。

假定充电回路P_j可以由 depot d_i进行覆盖，而d_i当前的负载为 load(d_i)，定义优先级函数如下：

$$w_i = (Y - load(d_i)\%Y)\%Y$$

定义|D(P_j)|指可以为充电回路P_j服务的 depot 个数，选择具有最小|D(P_j)|值的先进行最优 depot 选择，这是由于P_j有较小的机率被 depots 覆盖；然后选择该P_j所对应的 depot 中具有最大权值w_i的 depot 分配给充电回路P_j。若两个 depot 具有相同的权重值，那么将P_j分配给距离充电回路较近的 depot。

Algorithm 3 Depot 部署分配问题

输入：充电回路集合 P，待选部署位置集合 D，P(d_i)，D(P_j)
输出：depot 最终部署位置 D3 以及所覆盖充电回路
第一步：贪心列选择
P1 ← P；D1 ← D；D3 = ∅；
While (P1 ≠ ∅)
$D_{max} = argmax_{D_i \in D1}\{|D_i|\}$;
将D_{max}从 D1 中删除；
for $P_j \in D_{max}$
将P_j从 P1 中删除；
依据剩余 P1 更新 D1 中的每个值；(相当于更新每一列所对应总和)
End While
第二步：充电路径分配
P2 ← P；
While(P2 ≠ ∅)
$P_j = argmin_{P_j \in P2}\{|D(P_j)|\}$;
$d_i = argmax_{d_i \in D(P_j)}\{(Y - load(d_i)\%Y)\%Y\}$;
将充电回路P_j分配给 depot d_i；
将已分配 depot 的充电回路从 P2 中删除；
更新所有d_i的负载，即 load(d_i)；
D3 = D3 ∪ d_i；

20： **End While**

充电回路分配

定义：将充电回路集合$\mathrm{P}=\{P_j=(T_j,T_{j-working})|1\le j\le z\}$，其中 z 为集合 P 中包含的充电回路个数，如果可以将这一充电回路集合中的回路分配给一个 MCV 进行充电，则称该 MCV 的元充电周期 G 为所有$P_j\in P$的充电周期的最大公约数，即$\mathrm{G}=\mathrm{GCD}(T_1,T_2,\dots T_z)$，该 MCV 的负载率 ζ 为充电回路集合中所有回路的工作时间之和与 MCV 元充电周期之比：

$\zeta=\frac{\sum_{i=1}^{z}T_{i-working}}{G}$。

充电回路分配算法：

目标：将 P 条构建的充电回路分配给 m 个 MCV 以实现最小化 MCV 个数的目标。

算法核心思想：将 P 条充电回路按充电周期的上限进行排序，然后对每条充电回路判断其是否能够划分给当前的 MCV，能够划分给当前的 MCV 需要满足：1）该回路未分配，分配应满足一覆盖原则；2）元充电周期的 k 倍，即$\mathrm{k}\times\mathrm{G}$应当包含在充电周期的最大与最小值之间；3）应该保证将充电回路分配给 MCV 后，MCV 的负载率不应超过 1。

Algorithm 4 充电回路分配算法

1：**输入**：集合 P 内有若干条充电回路，相关参数
2：**输出**：所需最少 MCV 个数 m，每个 MCV 所负责充电回路
3：将所有充电回路按照周期的最大值从小到大排序并对其标序号；
4：m=0；//初始化 MCV 个数；
5：**for** i=1 to |P| **do**
6: label(P_i)=0；//label 来表示回路P_i是否已经分配 MCV 了，若分配了则 label(P_i)=1，反之为 0；
7：**end for**
8：**for** i=1 to |P| **do**
9： **if** label(P_i)=0 **then**
10： m=m+1；//增加 m 的数量
11： $MCV_m=\{P_i\}$；//将P_i分配给MCV_m；
12： 将G_m的元充电周期设置为第一个分配的回路的充电周期最大值；
13： $\zeta_m=\frac{T_{i-working}}{G_m}$；//计算当前分配回路后 MCV 的负载率；
14： label(P_i)=1；//更改P_i的标签属性；
15： **for** k=i+1 to |P| **do**
16： **if** label(P_k)=0 **then**
17： **if** 存在整数 l 能够保证$l\times G_m$在P_k的充电周期的上下限内 **then**
18： $W=\zeta_m+\frac{T_{k-working}}{G_m}$;
19： **if** $W\le 1$ **then**
20： $\zeta_m=\zeta_m+W$；
21： 将P_k分配给MCV_m；
label(P_k)=1；
22： **end if**
23： **end if**
24： **end if**

```
25：      end if
26：end for
```