在这道题目中，有两个主要的对称

（1）所有相同类型的资源都是对称的，例如land pad1和land pad2并没有区别，假设有一家无人机需要降落，那么选择land pad1或land pad2对最后运送订单的选择并没有影响。这里分两种情况，第一，使用同种资源且时间段相同的无人机所使用的同种资源的编号是对称的，第二，资源充沛的时候无人机可以选择一种资源的任意编号，所以我们可以增加约束：

constraint forall (d1,d2 in DRONE，a in { LAND, INSPECT, FULLSERVICE, RECHARGE, PACK }where(d1<d2)/\

(start[d1,enum\_next(ACTION,a)]=start[d2,enum\_next(ACTION,a)])/\(start[d1,a]=start[d2,a]))

(resource[d1,a]< resource[d2,a]);

还有一种办法是：规定在没有资源争夺的时候，编号小的无人机优先使用编号小的资源。

（2）相同状态的无人机是对称的。假设有两架无人机Drone1和Drone2，这两架无人机在到达时间和电量相同的情况下，假设要派送的订单为x1，x2，那么考虑下面两种情况：

Drone1派送x1，Drone2派送x2

Drone1派送x2，Drone2派送x1

这两种情况得到的结果是一样的，也就是说相同状态的无人机是对称的。所以我们可以增加约束：

constraint forall(d1,d2 in DRONE where(d1<d2)/\

(arrival[d1]!=arrival[d2])/\

(charge[d1]!=charge[d2])/\

(order[d1]!=0)/\order[d2]!=0)

(lex\_lesseq([order[d1]],[order[d2]]));

或者我们可以规定编号较小的无人机送编号较小的订单

constraint forall(d1,d2 in DRONE where(d1<d2)/\

(arrival[d1]!=arrival[d2])/\(charge[d1]!=charge[d2]/\(order[d1]!=0)/\order[d2]!=0))

(order[d1]<order[d2]);

使用gecode6.3.0运行，得到的结果汇总在下表中：见表格

得到的结论：

优化飞机对称性，当飞机数量远远小于订单数量时且多数飞机状态相同时，程序的运行速度明显提升，很快就得到了最优解，飞机数量接近或等于订单数量且多数飞机状态不相同时，运行速度无影响，由于多了一条限制，求解还会变慢。当资源充沛时，优化使用资源的对称性，程序的运行速度明显提升，当资源紧缺时，解决对称性并没有提升多少效率。

总结，当对称性十分普遍时，必须考虑解决对称性，这可以大幅提高模型的效率。

敏感性分析

跑不出来的见表格stageG

策略一： 加大资源数量。这对于一部分输入来说是可行的，比如drone3，drone6，drone7，drone8这些，由于资源问题，一开始并没有得到满足要求的解。

策略二：对于所有飞机保持高电量，比如drone1中将飞机3号的电量设为10，这样他就不需要快充了，利润会变为28。

策略三：更具策略二思路，还可以做的调整就是延长最长的时间限制，时间延长以后，所有的飞机都不要快充，快充的成本就降低了。

策略四：和上面的类似，可以将货物的重量降低，这样也可以避免充电，就有可能减少快充的成本。

策略五：加大时间轴的做法还有一种思路，就是减少打包的时间，变相加大了时间轴。

策略六：使得所有的订单都在很早的时间被准备好，使得所有的飞机一开始就能返航。这样子问题就变成了挑选ndrones个最贵的订单。