# 多周期产品配送问题的自适应大邻域搜索算法

摘要：摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要摘要。

关键词：产品配送；商品装载；整数规划；列生成算法

中图分类号：F224.3 文献标识码：A

# Models and Algorithms for Product Distribution in a Two-Echelon Inventory System

**Abstract:** Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract Abstract.

**Keywords:** product distribution; goods loading; integer programming; column generation

## 引言

## 问题描述

如图 1所示，本研究旨在为某大型电动车零售企业拟定一份最优的产品配送方案，从而实现快速需求响应，并降低配送成本。该企业的两级供应链系统由多家生产基地与经销商组成：在供给端，不同型号的SKU需完成制造流程并登记入库，当该企业接收到需求端的订单后，会在某特定时间内协调各生产基地的库存情况，并委托第三方物流（3PL）完成配送。每个生产基地的库存分为两部分：在仓库存和计划生产库存。前者是指仓库中的现有SKU数量，可以随时装载发货；后者是指根据事先确定的排产计划，等待产品制造完成之后，登记入库的SKU数量，此时就转为在仓库存。该系统呈现“多对多”供应结构特征：某种SKU可以由多家生产基地同时提供，且单一生产基地可以服务多个经销商，单一经销商也可由多个生产基地共同供货。基于行业实际运作特征：设定以下核心条件：（1）出于地理位置和长途运输成本的限制，禁止生产基地之间的横向转运；（2）第三方物流负责多车型车队的装载——配送集成优化，车型选择关联固定启用成本；（3）决策目标为最小化零售企业支付的3PL总费用。

问题建模需满足以下约束条件：（1）需求覆盖约束，各经销商对某种SKU的需求须由关联生产基地的出货量完全满足；（2）供给能力约束，在每个周期内单个生产基地的SKU出货量不得超过其库存持有量；（3）装载容量约束，车辆装载SKU的总体积不得超过车型容量的上限；（4）最小起运量约束，车辆装载SKU的体积低于标准阈值时触发惩罚成本；（5）存储能力约束，在每个周期内每家生产基地拥有的SKU数量不得超过其存储能力限制。为简化问题复杂度并确保模型的可解性，引入如下假设：（1）供给充足性，在每个周期内任意SKU在基地层面的总库存量不低于经销商需求总量；（2）一维度量假设，SKU体积与车辆容量均采用线性标度度量；（3）资源无限性，3PL可以调度的车辆数量无上限，车型选择仅受成本驱动。

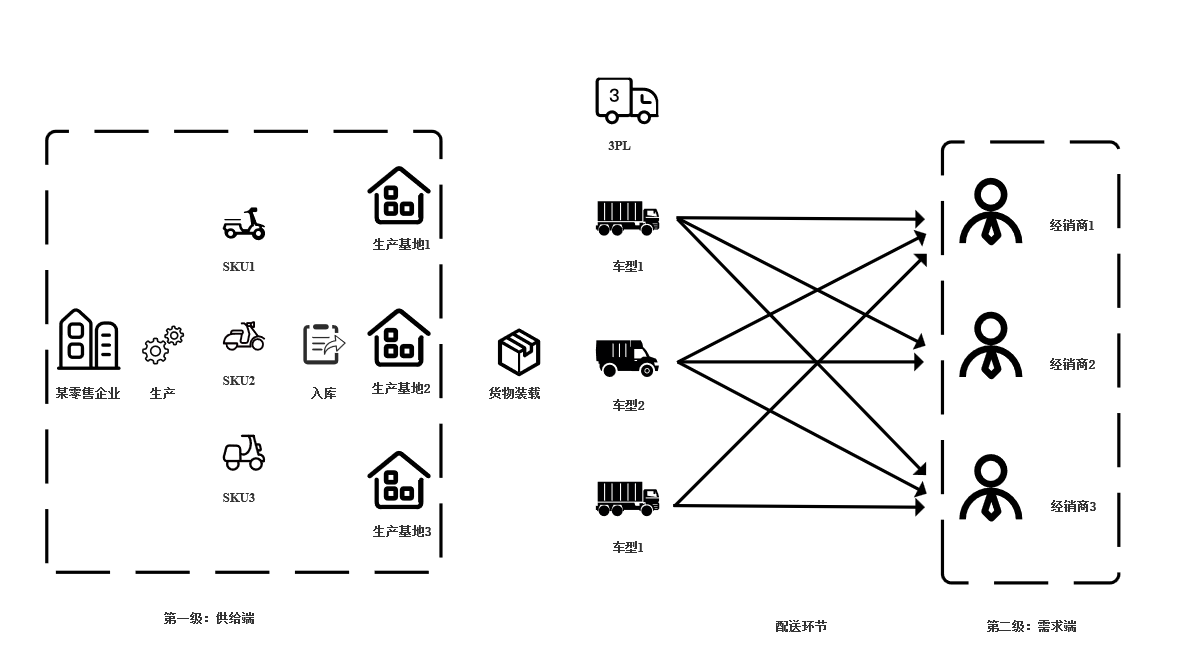


图 1问题示意图

## 模型与算法

### 符号描述

集合与下标：**为生产基地集合，表示生产基地编号；为经销商集合，表示经销商编号；为SKU集合，表示SKU编号；为3PL提供的车辆集合， 表示车辆编号；为当前规划周期，表示期初，为最大规划周期。

问题参数：为SKU的单位体积； 在周期内，生产基地的SKU的生产数量，其中； 为经销商对SKU的需求数量；为车辆的使用成本；为车辆的最大装载体积；为车辆的最小起运体积；为生产基地的存储能力上限。

常量：表示当车辆的实际装载体积低于最小起运量时，仍然使用车辆的惩罚系数；为足够大的正整数。

辅助变量：表示在周期内是否使用车辆，若满足则，否则，其中；表示在周期内，车辆的实际装载体积是否低于最小起运量，若满足则，否则，其中；表示在周期内车辆的低载量标识，如果车辆的实际装载体积低于最小起运量，则，否则，如果在周期内不使用车辆，则，其中。

决策变量：表示在周期内，如果使用车辆配送从生产基地到经销商的商品，则，否则，其中；表示生产基地中SKU的库存转移到周期的数量，其中，表示生产基地中SKU的期初库存；表示在周期内，从生产基地运往经销商的车辆装载SKU的商品数量，其中。

### 数学模型

基于上述符号名称及其含义，建立如下的数学模型：





















式为目标函数，表示最小化所有车辆的使用成本以及不满足最小起运量的惩罚成本；式表示车辆的使用状态；式表示只能将商品装载到已经使用的车辆上；式为最小起运量约束；式表示三种辅助变量应满足的关系；式为装载容量约束；式为需求覆盖约束；式为供给能力约束；式为存储能力约束。

## 算法设计

本文的研究问题属于NP-hard问题，使用精确算法和Gurobi等商业优化求解器难以在可接受时间内求解大规模实例。而自适应大规模邻域搜索（Adaptive Large Neighborhood Search, ALNS）算法作为一种相对新颖的元启发式算法，具备设计灵活性高、搜索过程多样化、求解效率快、可移植性强等优点[1]，根据破坏和修复算子的历史表现动态调整邻域选择机制，极大增强了算法解决复杂优化问题的能力。因此，本研究依据问题特点设计了相应的ALNS算法，其伪代码如所示。

|  |
| --- |
| 算法1：ALNS算法 |
| 输入：可行解，破坏算子的集合，修复算子的集合   1. 最优解，新获得的解 2. 重复下述步骤直至满足终止条件： 3. 使用分段轮盘赌的机制分别从和中选择一个破坏算子和一个修复算子 5. If 模拟退火接受准则接受新解, then 6. End if 7. If 当前解的目标函数值优于最优解的目标函数值, then 8. End if 9. 更新算子选择方案 10. 返回最优解 |

### 解的表达方式

由于该问题的复杂性，无法采用传统的二进制编码或者整数编码表示可行解，而是需要使用面向对象编程的思想，分别设计SolutionState类和Vehicle类。其中，Vehicle类需要与特定的生产基地、经销商、车型、周期相关联，并且确保该类的每一个对象都有唯一的车辆编号，其作用是记录每一辆车中装载的SKU种类和数量。而SolutionState类需要维护由Vehicle类的对象构成的列表以及决策变量的取值。因此，该问题的解可从SolutionState类中获取，表示为，其中为周期，为生产基地，为经销商，为SKU，为车辆编号，为车辆类型，为SKU数量。

### 初始解的构造

初始解作为算法迭代的基础，其质量直接影响算法搜索的多样性和效率。高质量的初始解有助于提升算法的性能，本研究采用的初始解生成算法伪代码如所示。

|  |
| --- |
| 算法2：初始解生成 |
| 1. 将经销商的需求降序排列 2. For经销商对SKU的需求 do 3. If , then continue 4. End if 5. 计算已经满足的数量和需求缺口 6. For每家可以提供SKU的生产基地 do 7. For周期todo 8. 根据供给能力约束更新 9. If  or , then continue 10. End if 11. 将车型按照容量降序排列 12. For每种车型 do 13. If , then break 14. End if 15. 尝试使用这种车型的车辆装载经销商需要的SKU 16. 更新已经满足的数量和 17. End for 18. End for 19. End for 20. 更新并记录仍未满足的需求 21. End for 22. 如果存在仍未满足的需求，则调用需求优先修复算子 |

### 破坏算子设计

在ALNS算法中，破坏算子通过有针对性地破坏当前解，驱动算法探索新的解空间，并在适应性机制下与修复算子协作，实现全局优化的目标。本研究依据问题特点和相关文献，分别设计了以下6种破坏算子。

1. 随机移除算子

该算子移除当前解中一定比例的车辆，目的是引入随机性，使搜索更加多样化。

1. 最差移除算子

该算子移除当前解中空闲体积最大的辆车辆，其中是当前解中使用的车辆总数，目的是避免车辆装载能力的浪费。

1. 不可行解移除算子

该算子应用在不可行解上，筛选出周期结束时，变量的SKU，找到满足该条件的SKU装载的车辆，从当前解中移除这些车辆，目的是保证解的可行性。

1. 剩余库存移除算子

该算子筛选出周期结束时，每家生产基地中剩余库存数量最多的SKU，找到满足该条件的SKU装载的车辆，移除当前解中满足该条件的辆车辆，其中是当前解中满足该条件的车辆总数，从而降低生产基地库存过多的风险。

1. Shaw移除算子

该算子最早被Shaw设计用于解决取送货（Pickup and Delivery Problem, PDP）问题，其设计思想后来被Ropke和Pisinger引入到ALNS算法中，进一步提升了该算法的求解效率。该算子的核心思想是通过移除当前解中与某些元素相关性较高的部分来改变解的结构，从而避免局部最优。本文采用的该算子的伪代码如所示。

|  |
| --- |
| 算法3：Shaw移除算子 |
| 1. 从当前解中随机选择一辆车辆作为种子车辆 2. 计算当前解中其他车辆与种子车辆的相关性 3. 将除种子车辆外的其他车辆按照相关性降序排列 4. 移除与种子车辆最相关的车辆，其中为已知的参数，是当前解中使用的车辆总数 5. 移除种子车辆 |

1. 路径移除算子

该算子随机选择一对生产基地与经销商的组合，将其视为一条路径，从当前解中找到该路径上的所有车辆，移除当前解中满足该条件的辆车辆，其中是当前解中位于路径上的所有车辆的数量。

### 修复算子设计

在ALNS算法中，修复算子通过重建可行解以及优化解的质量的方式，与破坏算子协作驱动迭代搜索，进而增强了该算法的搜索多样性。本研究依据问题特点和相关文献，分别设计了以下6种修复算子。

1. 贪心修复算子

该算子为每种尚未满足的经销商需求计算优先级，按照优先级从高到低的顺序，依次选择合适的生产基地和车型满足该需求。本文采用的该算子的伪代码如所示。

|  |
| --- |
| 算法4：贪心修复算子 |
| 1. For经销商对SKU的需求 do 2. 计算尚未满足的需求及权重 3. 计算所有可以提供SKU的生产基地在所有周期内可以提供的库存总量 4. 计算权重 5. 计算优先级，其中 6. End for 7. 将每种尚未满足的经销商需求按照优先级降序排列 8. For每种尚未满足的需求 do 9. If , then continue 10. End if 11. 找到所有可以为经销商提供SKU的生产基地的集合 12. For  do 13. For周期todo 14. 根据最新的库存水平计算生产基地在周期可以提供的SKU的数量 15. If or , then continue 16. End if 17. 将所有车型按照其容量降序排列 18. For每种车型 do 19. 计算当前车型的车辆可以装载SKU的最大数量 20. 计算实际需要装载的SKU数量 21. If , then continue 22. End if 23. 将实际需要的SKU数量装载到当前车型的车辆中 24. 更新和库存水平 25. If , then break 26. End if 27. End for 28. If , then break 29. End if 30. End for 31. If , then break 32. End if 33. End for 34. End for |

1. 局部搜索修复算子

该算子使用贪心修复算子获得可行解，尝试为解中的每辆车辆替换新的车型，只有在新车型足够容纳货物、满足最小起运量、成本更低的情况下，才更换为新车型的车辆。在完成替换操作后，评估新解的目标函数值是否有所改进，若有改进，则接受该变更，否则继续使用原车型。本文采用的该算子的伪代码如所示。

|  |
| --- |
| 算法5：局部搜索修复算子 |
| 1. 使用贪心修复算子获得可行解，且 2. While do 4. 计算替换车型之前的目标函数值 5. For可行解中的每辆车辆 do 6. 计算车辆中装载SKU的体积以及车辆容量 7. For每种车型do 8. 计算新车型是否能够容纳所有SKU 9. 计算所有SKU的体积是否不低于新车型的最小起运量 10. 计算新车型是否为成本更低的较小车型 11. If andand, then 12. 将所有SKU重新装载到新车型的车辆中，并计算新目标函数值 13. If , then 15. Else 16. 撤销替换操作 17. End if 18. End if 19. End for 20. End for 21. End while |

1. 库存平衡修复算子

该算子将每家生产基地按照周期结束时的库存水平降序排列，从库存水平最高的生产基地开始，为该基地供货的每家经销商随机选择一种车型，尝试使用该车型的车辆满足其尚未完成履约的需求，若能成功装载，则更新其需求。本文采用的该算子的伪代码如所示。

|  |
| --- |
| 算法6：库存平衡修复算子 |
| 1. 计算尚未完成履约的需求以及周期结束时的库存水平 2. 将生产基地和周期的组合按照降序排列 3. For每对组合 do 4. 获取生产基地可以供货的所有经销商的集合 5. For  do 6. 随机选择一种车型 7. 获取经销商所有尚未完成履约的需求 8. 使用车型的车辆尝试装载 9. If装载成功, then 更新 10. End if 11. End for 12. End for |

1. 紧急订单修复算子

该算子和贪心修复算子的计算步骤基本一致，不同点在于贪心修复算子必须确保获得解，能够完全满足所有经销商的每种需求，因此在车辆成功装载后，还需要检查条件是否成立，若成立，则需要尝试其他车型。而紧急订单修复算子不需要这种尝试，该算子允许在一定程度上违反需求覆盖约束，从而增加解的多样性。

1. 不可行解修复算子

该算子筛选出周期结束时，变量的SKU，只要存在满足该条件的SKU，检查当前解中的每一辆车辆，减少车辆中装载的该SKU的数量，直至。如果车辆中未装载任何SKU，则从当前解中移除该车辆。本文采用的该算子的伪代码如所示。

|  |
| --- |
| 算法7：不可行解修复算子 |
| 1. 筛选出满足条件的SKU构成的集合 2. While do 3. For当前解中的每一辆车辆 do 4. For SKUdo 5. If SKU装载在当前车辆中, then 6. 获取当前车辆中装载的SKU的数量 7. 计算应该减少的SKU的数量 9. If , then 从当前车辆中移除SKU 10. End if 11. If 当前车辆中未装载任何SKU, then 从当前解中移除该车辆 12. End if 14. If , then 15. End if 16. Break 17. End if 18. If , then break 19. End if 20. End for 21. End for 22. End while |

1. 需求优先修复算子

该算子优先考虑需求覆盖约束，确保所有经销商的每种需求都能得到满足， 在第一阶段优先使用较大车型和SKU的库存总量最多的生产基地，尝试满足尚未完全履约的需求，在第二阶段更新并遍历所有可以提供SKU的生产基地，使用最小车型的车辆装载货物。本文采用的该算子的伪代码如所示。

|  |
| --- |
| 算法8：需求优先修复算子 |
| 1. 计算尚未完成履约的需求并将其降序排列 2. For每种尚未完成履约的需求 do 3. 计算每家生产基地可以提供的SKU的库存总量 4. 将可以提供SKU的生产基地按照降序排列 5. For每家可以提供SKU的生产基地 do 6. If , then break 7. End if 8. For to do 9. If , then break 10. End if 11. 计算当前生产基地在周期内可以提供的SKU的数量 12. If , then 13. 将所有车型按照其容量降序排列 14. For每种车型do 15. If , then break 16. End if 17. 尝试使用当前车型的车辆装载SKU 18. If装载成功, then更新 19. End if 20. End for 21. End if 22. End for 23. End for 24. End for 25. 计算仍未完成履约的需求 26. If , then 27. 将降序排列 28. For每家可以提供SKU的生产基地 do 29. For to do 30. 计算当前生产基地在周期内可以提供的SKU的数量 31. If , then 32. 选择最小车型的车辆尝试装载SKU 33. If 装载成功, then 34. 更新 35. If , then break 36. End if 37. End if 38. End if 39. End for 40. If , then break 41. End if 42. End for 43. End if |

### 解的接受准则

本文采用模拟退火接受准则，假设当前可行解为，获得的新解为，可行解的目标函数值为，新解的目标函数值为，初始温度为，最终温度为，当前温度为，温度的更新步长为。则接受新解的概率为，在迭代过程中使用步长来更新当前温度，即。该准则通过引入受控的随机性，显著增强了ALNS算法的全局搜索能力、收敛效率和鲁棒性，使得该算法能够有效应对复杂的优化问题并适应多种场景。

### 算子选择机制

本文采用分段轮盘赌机制选择算子，该机制是轮盘赌的改进，两者的不同点在于，分段轮盘赌在一个分段内固定权重，仅在段末更新权重，而轮盘赌在每次迭代时都需要及时更新权重。相较于轮盘赌，分段轮盘赌可以避免高权重算子主导搜索的弊端，并且降低了计算开销。

起初，将第个算子的权重设置为1，段长，第个破坏算子的段内累计得分设置为0，第个修复算子的段内累计分得分设置为0。在每次迭代中，依据归一化的当前固定权重选择一对破坏算子和修复算子，分别依据公式和选择破坏算子和修复算子。该算法将选择的算子应用在当前可行解上，产生了一个新的候选解。





其中，和分别表示第个破坏算子和修复算子的权重，和分别表示选择第个破坏算子和修复算子的概率。该候选解经过ALNS算法评估，会得到四种结果，分别是：（1）候选解是全局最优解；（2）候选解优于当前可行解，但不是全局最优解；（3）可以接受候选解；（4）拒绝候选解。每一种结果都会被赋值一个评分，依据候选解的质量评估得分，并将累加到本段的累计得分上，即，。

当迭代次数为段长的整数倍时，同时更新所有算子的权重，破坏算子的权重更新为，修复算子的权重更新为，其中为衰减参数，并且将本段的累计得分清零，即，，进入下一个分段。

## 数值实验

本研究采用Python 3语言和Gurobi 12.0.1软件，实现上述数学模型和ALNS算法。所有数值实验均在Windows 11操作系统环境下完成，硬件配置为AMD Ryzen 7 5700U（1.80GHz）处理器以及16GB内存。

### 实验数据设计

实验数据按规模划分为小、中、大三个层级，每组包含30个独立实例，总计90个实例。实例设计的关键特征包括生产基地的数量、经销商的数量、SKU的种类数量、车辆类型的数量和规划周期，具体取值范围如表 1所示。该表的不同参数设置反映出零售企业经营规模的变化和多车型配置场景。

经销商在所有周期内对SKU的需求总量且服从均匀分布。任意SKU在所有周期内的总供给量满足条件。每种SKU的体积且服从均匀分布。每种车辆类型的容量根据其相对大小分级设定，最小起运量为容量的50%～75%，最小车辆类型的固定使用成本标准化为1，其它车型成本按照容量比例递增。

表 1 不同规模数据集的关键特征

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据集规模 | 生产基地/家 | 经销商/家 | SKU类别/种 | 车辆类型/种 | 周期/天 |
| 小规模 |  |  |  |  | 3 |
| 中规模 |  |  |  |  | 3 |
| 大规模 |  |  |  |  | 3 |

### 结果分析

小规模 900 s、中规模1800 s、大规模 3600 s

## 结论与展望

参考文献

[1] WINDRAS MARA S T, NORCAHYO R, JODIAWAN P, et al. A survey of adaptive large neighborhood search algorithms and applications [J]. Computers & Operations Research, 2022, 146: 105903.DOI:<https://doi.org/10.1016/j.cor.2022.105903>.