需求可拆分的汽车零部件循环取货路径优化研究*

朱 玲, 吴 迪
(上海交通大学 中美物流研究院, 上海 200030)

摘 要: 为了降低汽车零部件入厂物流的总成本,针对汽车零部件入厂物流的循环取货路径规划问题,提出将供应商集货需求拆分配送的改进数学模型,以最小化运输和库存总成本为目标函数,并且满足时间窗和车辆容量的限制。通过设计禁忌搜索算法求解,并根据模型的特点对初始解及邻域搜索方法进行改进。最后应用实验数据验证算法有效性,并与其他算法对比,结果表明通过需求拆分可以节约运输成本并提高车辆装载率。

关键词: 零部件入厂物流; 循环取货; 需求可拆分; 时间窗; 禁忌搜索算法
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2013)06-1647-05
doi:10.3969/j.issn.1001-3695.2013.06.011

Research on automobile parts milk-run routing with split deliveries

ZHU Ling, WU Di
(*Sino-US Global Logistics Institute*, *Shanghai Jiao Tong University*, *Shanghai* 200030, *China*)

Abstract: The milk-run mode has been widely used in the automobile parts supply. The key to the milk-run is to seek the optimal paths for the vehicles. This paper developed a milk-run model with time windows and split deliveries. The objective function was to minimize the transportation and inventory cost. In order to solve this model, this paper proposed a tabu search algorithm. According to the specificity of the model, it designed the tabu search algorithm with special treatments on initial solution and neighborhood search. In the end, it used some computational experiments to testify the validity and efficiency of the model and the algorithm. The results indicate that the transportation cost can be saved and the vehicle loading rate can be improved by this model and algorithm.

Key words: parts supply logistics; milk-run; split deliveries; time windows; tabu search algorithm

0 引言

准时生产方式(just in time,JIT)是由日本丰田汽车公司在20世纪50年代初提出,经20几年逐步形成的一种先进生产方式。其核心思想是消除浪费、降低成本,增加利益。而其中浪费的主要原因是库存造成的。因此JIT也可以说是一种追求无库存或者最小库存的生产系统。

但是汽车制造厂一味地追求零库存而减小零部件供应的订货批量,将会引起订货频次的增加,从而有可能导致高昂的运输成本;相反,为了降低运输成本而减小订货频次、增加订货批量,又将会导致库存成本的升高。这两者都将导致高昂的物流总成本,不符合JIT思想,为企业带来巨大浪费。因此,一种以最小化库存和运输总成本为目标的循环取货模式得到了企业和学者的广泛关注。

循环取货(milk-run)是指汽车制造商用一辆货车从制造厂或区域分拨中心(regional distribution center,RDC)出发,沿事先设计好的路线,在每天固定的时间窗到指定的多个零部件供应商处提取零部件,然后返回制造厂或区域分拨中心的供货模式。这种供货模式具有小批量、多频次以及准时性的特点[1],克服了供货批量与频次的矛盾。循环取货模式通常由制造厂委托第三方物流(third-party logistics,TPL)来具体实施。例如上海通用汽车与安吉天地物流合作,使得运输成本节约30%,送货准时率、正确率都明显提高[2]。

循环取货模式应用的关键是取货路径的优化。文献[3]设计了改进的启发式节约算法来求解milk-run最优路径。文献[1]提出了结合扫描法和禁忌搜索法的两阶段求解算法,将车辆路径问题转换为多个旅行商问题,但是其在建立车辆路径优化模型时,只考虑了运输成本作为目标函数,而没有考虑库存成本。文献[4]引入了线边最大库存的限制,但是该模型限定了每一家供应商只能由一条取货路径访问,这一限定虽然大大降低了问题的复杂度,但是与现实情况并不相符;并且当一家供应商的零部件可以被分配到多条路径中运输时,即需求可拆分时,运输成本将会显著下降[5],车辆装载率也将提高。文献[6]建立了需求可拆分车辆路径问题的数学模型,并设计禁忌搜索算法求解,但是该问题没有考虑时间窗的约束。文献[7]提出了两阶段算法求解需求可拆分的车辆路径问题,但是仅针对单车场、单车型且无时间窗约束。文献[8]研究了需求可拆分的开放车辆路径问题,即车辆最终不返回车场,因此也不符合循环取货的要求。

由于目前尚没有研究针对带有时间窗约束的需求可拆分的循环取货路径问题,因此本文将针对汽车零部件循环取货路径优化问题,以需求可拆分为前提,建立以最小化运输和库存总成本为目标函数,满足时间窗及容量约束的数学模型,并设计改进禁忌搜索算法求解。

收稿日期: 2012-10-18; **修回日期**: 2012-11-29 **基金项目**: 国家自然科学基金重大项目(71132006)

作者简介:朱玲(1987-),女,天津人,硕士,主要研究方向为物流供应链(zhuling1987@hotmail.com);吴迪(1954-),男,辽宁沈阳人,教授,博导,主要研究方向为运营与物流管理.

1 模型构建

1.1 设计思路

传统的循环取货路径问题规定每一供应商点只能被一条路径访问,因此,当车辆的剩余容量不能满足某点全部的供货需求时,只能选择不载,由另一路径完成全部供货需求。这就导致了车辆的装载率低、运输成本高。针对这一问题,本文提出了供应商需求可拆分配送的方案。即每个供应商点可允许多条路径访问,当某一路径的剩余容量不能满足该点的全部供货需求时,可以只配送一部分,剩余的需求由另外的路径运载。这一方案将大大提高车辆的装载率,同时降低运输成本。方案示意图如图 1 所示。图中不同的线型分别代表不同的取货路径,几条路径共同访问的点即为需求可拆分点。

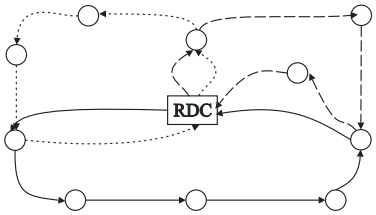

图1 需求可拆分车辆路径示意图

给定有向图 $G=(V,E)$,其中:$V=\{i \mid i=0,1,\cdots,n\}$,$i=0$ 时代表 RDC,$i=1,\cdots,n$ 代表 n 个供应商,且供应商与 RDC 的位置坐标均为已知;E 为有向图中的弧,以 (i,j)($\forall i,j\in V, i\neq j$)表示。另有 $K=\{k \mid k=1,\cdots,m\}$ 代表有 m 条取货路径,每条路径由一辆货车从 RDC 出发途经多个供应商集货最终返回 RDC,所以亦即有 m 辆车,m 是一个待定的变量。每个供应商点的供货需求可以被拆分,由多条路径访问。但在同一路径中,每个节点只能被访问一次。并且假设每个供应商可以同时服务多辆货车,即当有多辆货车同时到达同一供应商时不会出现拥挤和等待。

问题已知以下参数:节点 i 处的最早开始供货的时间为 e_i;最晚开始供货的时间为 l_i;供货需求量为 D_i;货物装载时间为 s_i;路径 k 上 (i,j) 之间的运输成本为 c_{ij}^k,它与运输距离 d_{ij} 成正比;运行时间为 t_{ij}^k;车辆 k 的容量限制为 Q^k;路径 k 零件 i 的单位库存成本为 h_i^k;决策变量为 x_{ij}^k、y_i^k 和 T_i^k。

$$x_{ij}^k=\begin{cases}1 & \text{路径 } k \text{ 的车辆经过}(i,j)\\ 0 & \text{其他}\end{cases}\qquad \forall i,j\in V;\forall k\in K$$

其中:y_i^k 为车辆 k 在 i 点的集货量;T_i^k 为车辆在开始接受服务的时间。

问题的目标函数为最小化运输与库存总成本,同时满足车辆容量、时间窗及供应商需求可拆分的约束条件,求解出最优的循环取货路径。

1.2 数学建模

根据以上分析,建立如下数学模型:

$$\min Z=\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m}c_{ij}^k x_{ij}^k+\sum_{i=1}^{n}\sum_{k=1}^{m}h_i^k y_i^k \tag{1}$$

$$\text{s.t.}\quad \sum_{j=1}^{n}x_{0j}^k=1\quad \forall k\in K \tag{2}$$

$$\sum_{j=1}^{n}x_{j0}^k=1\quad \forall k\in K \tag{3}$$

$$\sum_{i=0}^{n}x_{ip}^k-\sum_{j=0}^{n}x_{pj}^k=0\quad \forall k\in K;\forall p\in V \tag{4}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{m}x_{ij}^k\geqslant 1\quad \forall i\in V \tag{5}$$

$$\sum_{k=1}^{m}y_i^k=D_i\quad \forall i\in V \tag{6}$$

$$y_i^k\leqslant D_i\sum_{j=0}^{n}x_{ij}^k\quad \forall k\in K;\forall i\in V \tag{7}$$

$$e_i\leqslant T_i^k\leqslant l_i\quad \forall i\in V \tag{8}$$

$$x_{ij}^k(T_i^k+s_i+t_{ij}^k-T_j^k)\leqslant 0\quad \forall i,j\in V \tag{9}$$

$$\sum_{i=1}^{n}y_i^k\leqslant Q^k\quad \forall k\in K \tag{10}$$

$$x_{ij}^k=\{0,1\}\quad \forall i,j\in V;\forall k\in K \tag{11}$$

$$y_i^k\geqslant 0\quad \forall i\in V;\forall k\in K \tag{12}$$

$$T_i^k\geqslant 0\quad \forall i\in V;\forall k\in K \tag{13}$$

式(1)为目标函数,即最小化运输与库存的总成本;式(2)(3)保证了所有路径均由 RDC 出发,并最终返回 RDC;式(4)为流量守恒约束,即到达某点的车辆数等于离开该点的车辆数;式(5)确保了每一节点至少被一条路径访问;式(6)保证每一点的供货需求得到满足;式(7)确保了需求只能被拆分到访问该点的路径上,并且该路径上被分配的需求量不能超过该点的总需求量;式(8)是取货车辆到达时间窗的约束;式(9)表明如果 (i,j) 两点之间存在路径,则车辆 k 到达 j 点的时间要大于等于车辆从 i 离开的时间加上两点之间运行时间之和;式(10)为车辆容量限制,路径 k 上的集货量不能超过车辆 k 的最大容量;式(11)~(13)为变量的整数性及非负性限制。

2 算法设计

上述模型同时具有带时间窗的车辆路径问题(vehicle routing problem with time windows, VRPTW)及需求可拆分的车辆路径问题(split delivery VRP, SDVRP)的特点,属于 NP-Hard 问题,很难用精确算法求解。因此本文将设计改进的启发式算法,采用改进的最佳插入法[6,9]构造初始解,并用禁忌搜索算法[10]求得进一步优化的解。

2.1 初始解构建

禁忌搜索算法对初始解依赖性较强,本文将设计改进的最佳插入法构建初始解,具体流程如下:

a)令集合 U 为未访问点的集合,判断 U 是否为空集,若是则终止;否则转步骤 b)。

b)选择 U 中距离 RDC 最远的一个供应商点建立从 RDC 出发,经过该节点并返回 RDC 的初始路径,并将该节点从中删除。

c)对 U 中的任一节点 u 分别考察将其插入到现有路径中任意两点 (i,j) 之间时是否满足时间窗约束条件式(8)(9),将满足条件的节点及其对应位置设为集合 U'。

d)对 U' 中的任一节点 u 分别计算将其插入到现有路径中任意时间窗可行位置 (i,j) 之间造成的距离增加值 $c(i,u,j)=d_{iu}+d_{uj}-d_{ij}$,选择具有最小增加值 $c(i_u,u,j_u)$ 的位置 (i_u,j_u) 作为该点的最佳插入位置。对 U' 中所有点的最佳插入位置进行比较,选出具有最小距离增加值的节点 u^*,将其插入到最佳位置为 (i_{u^*},j_{u^*})。

e)考察插入 u^* 后路径的总容量是否满载,否则将 u^* 从中删除,转步骤 c);是则转步骤 f)。

f)考察插入 u^* 后路径的总容量是否超载,否则该路径构建结束,将 u^* 从 U 中删除,转步骤 a);是则拆分节点 u^*,使当前路径刚好满载,构建结束。超出部分 u^* 仍保留 U 中,转步骤 a)。

2.2 禁忌搜索算法设计

2.2.1 解的表示方法

采用需求点与需求量对应排列的方式表示解[6]。用 0 表示 RDC,$1,\cdots,n$ 表示各供应商节点,并在每个节点后以括号注明该

路线在该节点的配送量。例如,解的表示为 0(0)2(5)4(3)3(1)0(0)0(0)5(2)1(3)0(0),则该解包含两条配送路径:0-2-4-3-0 和 0-5-1-0,并且每个供应商点的配送量也已表明。

2.2.2　解的评价方法

以目标函数作为解的评价函数,在满足约束条件的基础上,目标函数的值越小,则解的质量越高。

2.2.3　邻域搜索方法

文献[6]提到如果两条配送路线中存在两个共同访问点,则可通过对共同访问点进行合并操作,减少到一个,从而得到一个更优的解。

邻域搜索主要有线路内的搜索和线路间的搜索两种。本文采用 or-opt 交换法以及 λ-interchange 法进行邻域搜索。具体方法如下:

a)or-opt 交换法用于线路内的搜索。即在一条路径上选取一点 i,将其插入到同一路径的其他两点之间,且要满足时间窗等约束。由于该线路的总体需求量没有改变,因此容量限制总是可以被保证。

b)λ-interchange 法用于线路间的搜索。本文将采用 λ-interchange 的 1-0 交换法和 1-1 交换法,并根据需求可拆分的特点进行一定的改进。

1-0 交换法即将一条路径 k_1 上的某一点插入到另一路径 k_2 的最优位置,同时要满足时间窗及容量等约束。针对需求可拆分问题,具体可分为以下四种情况:

情况 1　$i \in k_1, i \notin k_2$ 且 k_2 路径的剩余容量 $Q_R^{k_2}$ 大于等于 i 点在路径 k_1 上的集货需求量 $y_i^{k_1}$,此时将 i 从路径 k_1 中删除,插入到路径 k_2 中,并更新 $y_i^{k_2} = y_i^{k_1}, y_i^{k_1} = 0$。

情况 2　$i \in k_1, i \notin k_2$ 且 $Q_R^{k_2} < y_i^{k_1}$,此时将 i 插入到路径 k_2 中,并且将其需求拆分为 $y_i^{k_2} = Q_R^{k_2}, y_i^{k_1} = y_i^{k_1} - Q_R^{k_2}$。

情况 3　$i \in k_1 \cap k_2$ 且 $Q_R^{k_2} \geqslant y_i^{k_1}$,此时将 i 从路径 k_1 中删除,并更新 $y_i^{k_2} = y_i^{k_1} + y_i^{k_2}, y_i^{k_1} = 0$。

情况 4　$i \in k_1 \cap k_2$ 且 $Q_R^{k_2} < y_i^{k_1}$,此时无法进行插入操作。

1-1 交换法即将一条路径 k_1 上的某一点 i 与另一路径 k_2 上的某点 j 的位置进行交换,同时需满足各项约束条件。针对需求可拆分问题,具体可分为以下四种情况:

情况 5　$i \in k_1, i \notin k_2, j \notin k_1, j \in k_2$ 且 $Q_R^{k_1} + y_i^{k_1} \geqslant y_j^{k_2}, Q_R^{k_2} + y_j^{k_2} \geqslant y_i^{k_1}$,则将 i、j 两点完全交换,$y_i^{k_2} = y_i^{k_1}, y_j^{k_1} = y_j^{k_2}$,如图 2 所示。

情况 6　$i \in k_1, i \notin k_2, j \notin k_1, j \in k_2$ 且 $Q_R^{k_1} + y_i^{k_1} < y_j^{k_2}, Q_R^{k_2} + y_j^{k_2} > y_i^{k_1}$,则将 i 从 k_1 删除,插入到 k_2 中,$y_i^{k_2} = y_i^{k_1}$,将 j 插入到 k_1,其需求拆分为 $y_j^{k_1} = y_i^{k_1}, k_j^{k_2} = y_j^{k_2} - y_i^{k_1}$,如图 3 所示。

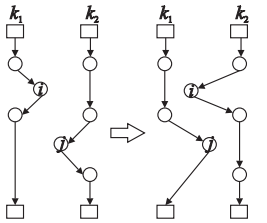

图2　交换法情况5

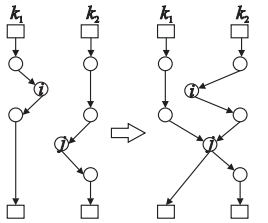

图3　交换法情况6

情况 7　$i \in k_1 \cap k_2, j \notin k_1, j \in k_2$ 且 $y_i^{k_1} \leqslant y_j^{k_2}$,此时将 i 从路径 k_1 删除,则 $y_i^{k_1} = 0, y_i^{k_2} = y_i^{k_1} + y_i^{k_2}$。并将 j 插入到路径 k_1 中,将其需求拆分为 $y_j^{k_1} = y_i^{k_1}, y_j^{k_2} = y_j^{k_2} - y_i^{k_1}$,如图 4 所示。

情况 8　$i \in k_1 \cap k_2, j \in k_1 \cap k_2$ 且 $y_i^{k_1} \leqslant y_j^{k_2}$,则将拆分点合并,将 i 从路径 k_1 删除,则 $y_i^{k_1} = 0, y_i^{k_2} = y_i^{k_1} + y_i^{k_2}$。$j$ 点位量不变,但需求量变为 $y_j^{k_1} = y_i^{k_1}, y_j^{k_2} = y_j^{k_2} - y_i^{k_1}$,如图 5 所示。

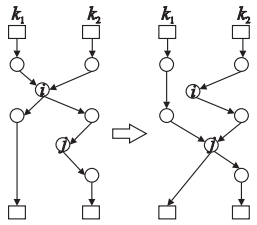

图4　交换法情况7

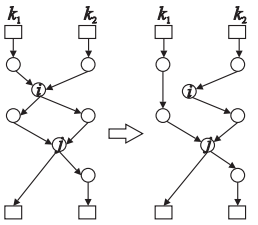

图5　交换法情况8

2.2.4　禁忌对象及禁忌长度

针对以上的邻域搜索方法,建立以解的向量分量为禁忌对象的禁忌表,具体如表 1 所示。

表 1　禁忌对象表

交换方法	禁忌对象
or-opt	$(i,j,j+1)$
1-0 交换	(k_i,i,k_j)
1-1 交换	(k_i,i,k_j,j)

对以上三类禁忌对象均设禁忌长度为 $t = \sqrt{n}$,其中 n 为问题中的节点数。

2.2.5　特赦规则

采用基于评价值的规则,即当出现一个解 x^{now} 的评价值与已出现的最好解 x^{best} 的评价值相比时 $E(x^{now}) < E(x^{best})$,则即使从 x^{best} 向 x^{now} 的变化是被禁忌的,此时也可以解禁。

2.2.6　终止规则

采用双重终止规则,即当总的迭代次数达到某一充分大的数 N,或者在某一给定的步数 L 之内目前最好解没有改进,则算法终止。

3　算例分析

众多研究 VRPTW 问题的学者均通过 Solomon[9] 标准测试数据集来对其设计的启发式算法进行测试。由于需求可拆分且带有时间窗约束的循环取货路径问题目前没有标准测试数据集,因此本文也将采用 Solomon 测试数据来验证需求可拆分算法的可行性。

Solomon 实验中的目标函数只考虑距离的长短,为了便于对照,本文实验将对模型中的参数做如下设置:$c_{ij}^k = d_{ij}, h_i^k = 0$,即也只以距离作为目标函数。为方便计算,另假设货物的体积、重量均为同一规格,则车辆容量仅以货物件数为单位。

本文实验采用 Java 语言编写模拟程序,使用 Eclipse 开发环境。硬件配置如下:CPU 采用 Intel Core i3 M380 @ 2.53 GHz;内存容量为 2 GB,内存频率为 DDR3 1333 MHz。

Solomon 标准测试数据是由一个 RDC 和 100 个供应商点组成的。通过设置各点不同的坐标和时间窗又可分为六大类型的若干数据集。R_1、R_2 类中供应商点的坐标全部为随机生成,C_1、C_2 类中供应商点具有一定的区域聚集性,RC_1、RC_2 类则是混合了随机点和聚集点两种类型。并且 R_1、C_1、RC_1 类中 RDC 的开放时间窗较窄,因此每条路径上被服务的集货点较少,车辆就不得不返回 RDC;而 R_2、C_2、RC_2 的 RDC 开放时间窗较宽,因此可以允许车辆访问更多的集货点后再返回。每个类型均由若干数据集构成,如 R_1 包含 R101、R102、R103 等,各数据集中点的坐标相同,但各点时间窗不同。

实验将通过对比使用本文拆分算法求解 Solomon 标准测试数据得到的最优解与目前已知的使用启发式算法得到的最佳近似解(不可拆分的情况下)及文献[11]所设计的需求可拆分的启发式算法求得的最优解来验证本文算法的有效性。对比结果如表 2 所示,表中用粗体字表示出使用本文算法得到的

解优于其他算法的解。

表 2 原始 Solomon 数据三种算法最优解对比

数据集	已知最优解	文献[11]最优解	本算法最优解
R101	1645.79	1648.96	1658.23
R102	1486.12	**1506.08**	**1506.08**
R103	**1292.68**	**1276.58**	**1269.29**
R104	1007.24	**1059.42**	**1024.47**
R105	1377.11	**1448.89**	**1382.33**
R106	1251.98	**1282.50**	**1271.75**
R107	1104.66	**1229.47**	**1113.48**
R108	960.88	962.33	980.83
R109	**1194.73**	**1241.21**	**1185.18**
R110	1118.59	**1212.11**	**1127.30**
R111	**1096.72**	**1098.37**	**1093.97**
R112	982.14	**1000.12**	**985.01**
R201	**1252.37**	**1236.93**	**1236.61**
R202	**1191.70**	**1138.91**	**1085.45**
R203	**939.54**	911.87	**937.87**
R204	**825.52**	**844.13**	**797.69**
R205	994.42	**1035.57**	**1024.96**
R206	906.14	**990.42**	**945.68**
R207	**893.33**	**909.18**	**885.06**
R208	726.75	**766.37**	**740.39**
R209	909.16	**966.11**	**959.42**
R210	939.34	**950.99**	**946.10**
R211	**892.71**	**833.47**	**874.43**
C101	828.94	828.94	829.30
C102	828.94	840.30	942.18
C103	**828.06**	**834.56**	**828.06**
C104	824.78	855.30	876.22
C105	828.94	828.94	900.22
C106	828.94	828.94	881.01
C107	828.94	828.94	896.43
C108	828.94	828.94	891.23
C109	828.94	828.94	879.07
C201	**591.56**	**591.56**	**591.56**
C202	591.56	591.56	593.28
C203	591.17	591.17	626.66
C204	590.60	**612.51**	**594.64**
C205	**588.88**	**588.88**	**588.88**
C206	588.49	588.49	591.42
C207	588.29	588.29	600.29
C208	588.32	588.35	599.64
RC101	**1696.94**	**1743.32**	**1671.94**
RC102	**1554.75**	**1564.88**	**1512.81**
RC103	1261.67	1303.40	1303.98
RC104	1135.48	**1203.85**	**1183.64**
RC105	**1629.44**	**1629.84**	**1595.07**
RC106	**1424.73**	**1459.59**	**1419.63**
RC107	1230.48	**1282.58**	**1267.19**
RC108	1139.82	**1268.03**	**1186.84**
RC201	1406.91	1375.94	1436.18
RC202	**1367.09**	**1283.78**	**1270.38**
RC203	1049.62	**1142.61**	**1117.80**
RC204	798.41	**867.72**	**840.39**
RC205	**1297.19**	**1345.87**	**1292.71**
RC206	1146.32	1121.40	1153.97
RC207	1061.14	1045.55	1063.53
RC208	828.14	**987.41**	**949.68**

通过观察表 2 发现，在 56 组测试数据中，本文算法有 37 组最优解优于文献[11]的最优解或目前已知最好非拆分解，这充分证明了本文算法的有效性，特别是由于文献[11]也采用了拆分的方法，通过对比说明了本文拆分算法优于文献[11]的拆分算法。但是只对比本算法与目前已知最优非拆分算法的解就会发现，虽然本文算法在某些数据集合上更有优势，但是仍有大部分不及目前已知最优解。这是因为 Solomon 标准测试数据中设置的车辆容量较大，而时间窗则较窄，因此使用本文算法计算时往往不能发挥出需求可拆分的特点，车辆大多在装载率较低时就由于时间窗的限制不得不中止服务。

针对这一问题，将 Solomon 数据中的车辆容量缩小至原始的四分之一，再分别使用本文拆分算法和 Solomon 不可拆分算法对改进的数据求解，对比两种算法的最优解，证明在容量有限的情况下，可拆分算法得到的解更优，对比结果见表 3 所示。另在表 3 中第二列列出原始车辆容量限制时各学者采用各种不可拆分算法求得的已知最优解，与容量缩小后两种算法得到的最优解进行对比。表中用粗体字标示使用本文算法得到的解优于其他算法的解。

表 3 容量减小 Solomon 数据各算法最优解对比

数据集	已知最优解	非拆分最优解	拆分最优解
R101	1645.79	2863.23	2236.85
R102	1486.12	2861.47	2122.04
R103	1292.68	2947.33	2024.39
R104	1007.24	2790.17	2007.78
R105	1377.11	2854.03	2141.34
R106	1251.98	2993.46	2028.05
R107	1104.66	2849.03	2003.18
R108	960.88	2575.83	2019.79
R109	1194.73	2876.44	2014.10
R110	1118.59	3211.77	1980.16
R111	1096.72	3237.93	2026.51
R112	982.14	3001.95	1992.18
R201	1252.37	2643.44	1258.57
R202	1191.70	2471.69	1084.67
R203	939.54	2087.55	937.29
R204	825.52	1591.33	826.61
R205	994.42	2506.96	1069.13
R206	906.14	2470.16	997.01
R207	893.33	2101.92	919.40
R208	726.75	1674.79	806.38
R209	909.16	2902.48	952.70
R210	939.34	2843.38	946.55
R211	892.71	2474.21	863.25
C101	828.94	3540.86	2499.37
C102	828.94	3644.83	2560.18
C103	828.06	3576.76	2606.11
C104	824.78	3269.72	2612.44
C105	828.94	3776.40	2617.03
C106	828.94	4231.64	2617.70
C107	828.94	3659.14	2745.92
C108	828.94	4213.43	2664.84
C109	828.94	3918.14	2653.10
C201	591.56	1788.83	1061.38
C202	591.56	2400.84	1078.20
C203	591.17	2262.89	1070.84
C204	590.60	1939.73	1039.55
C205	588.88	2264.84	1093.50
C206	588.49	3391.49	1085.72
C207	588.29	2437.99	1072.05
C208	588.32	3027.65	1077.60
RC101	1696.94	3696.59	3048.91
RC102	1554.75	3729.39	2923.42
RC103	1261.67	3831.73	2900.87
RC104	1135.48	3547.61	2908.67
RC105	1629.44	4259.61	2959.59
RC106	1424.73	3942.54	2933.61
RC107	1230.48	4186.38	2870.57
RC108	1139.82	3724.04	2925.70
RC201	1406.91	3097.78	1481.02
RC202	1367.09	3036.28	1337.67
RC203	1049.62	2448.80	1045.28
RC204	798.41	1965.54	993.18
RC205	1297.19	3515.41	1231.58
RC206	1146.32	2838.27	1261.42
RC207	1061.14	3215.15	1191.89
RC208	828.14	2943.04	857.00

对比车辆容量缩小至四分之一后分别使用不可拆分和可拆分两种算法得到的最优解,可发现使用本文拆分算法明显优于 Solomon 不可拆分算法,证明在容量有限的情况下,可拆分算法可以更好地利用车辆的剩余空间,达到减小运输成本的目的。

另外,对比车辆容量缩小后使用本文拆分算法求得的最优解和在原始容量时通过不可拆分算法求得的已知最优解,可发现在计算 R202、R203、R211、RC202、RC203 和 RC205 数据集时,使用本文拆分算法得到最优解甚至优于四倍车辆容量时的已知最优解,而对其他 R_2 和 RC_2 系列数据集,本文拆分算法的最优解亦非常接近四倍车辆容量时的已知最优解。这是因为 R_2 和 RC_2 系列数据集的 RDC 开放时间窗较宽,车辆可以访问更多的集货点后再返回 RDC,此时需求可拆分的优势可以得到充分发挥。

对比本文拆分算法在 R_1、C_1、RC_1、R_2、C_2、RC_2 各子集上的表现,发现在 R_2 和 RC_2 子集上的表现最佳,原因如前述;而在 C_1 和 C_2 子集上表现稍差,分析原因为 C 系列的数据具有区域聚集性,本文算法并未对其做特殊处理,应有更有效的算法针对具有区域聚集性特点的数据。

4　结束语

本文针对汽车零部件供货物流的循环取货路径规划问题,提出了具有时间窗的、供应商需求可拆分配送的方案,并依此思路建立数学模型,设计了改进的启发式算法求解。通过采用 Solomon VRPTW 问题的标准测试数据验证了本文拆分算法的有效性。并且通过求解缩小车辆容量的 Solomon 数据,验证了在容量有限的情况下,本文拆分算法相较不可拆分算法,可以更好地利用车辆剩余空间,减小运输成本。特别是在时间窗较宽的情况下,本文拆分算法优越性显著。另外,本文为计算方便假设了货物为均一规格、车辆容量以货物件数为单位。在实际情况中,这一假设并不合理,此时可以考虑以货物和车辆的体积或质量为单位进行计算。同时,本文仅考虑了单一 RDC、单一车型的情况,并且假设了供应商可以同时服务多辆货车,即当有多辆货车同时到达同一供应商处时不会出现拥挤和等待。在后续研究中,可以加入多个 RDC、多种车型及供应商服务能力有限、车辆排队等候的情况,进一步考察此时需求可拆分算法的表现。

参考文献:

[1] 张坤,江海容. 汽车零部件循环取货车辆路径优化研究[J]. 物流科技,2009,32(2):69-72.

[2] 徐秋华. Milkrun—循环取货方式在上海通用汽车的实践和应用[J]. 汽车与配件,2003(3):21-24.

[3] 王旭,陈栋,王振锋. 汽车零部件 Milk-run 车辆调度优化模型和算法[J]. 计算机应用,2011,31(4):1125-1128,1132.

[4] 汪金莲,蒋祖华. 汽车制造厂零部件入厂物流的循环取货路径规划[J]. 上海交通大学学报,2009,43(11):1703-1709.

[5] JIN Ming-zhou, LIU Kai, BOWDEN R O. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem[J]. International Journal of Production Economics, 2007, 105(1): 228-242.

[6] 孟凡超,陆志强,孙小明. 需求可拆分车辆路径问题的禁忌搜索算法[J]. 计算机辅助工程,2010,19(1):78-83.

[7] 刘旺盛,黄娟. 需求可拆分的车辆路径问题的分段求解[J]. 集美大学学报,2011,16(1):38-44.

[8] 李三彬,柴玉梅,王黎明. 需求可拆分的开放式车辆路径问题研究[J]. 计算机工程,2011,37(6):168-171.

[9] SOLOMON M M. Algorithms for the vehicle routing and scheduling problems with time window constraints[J]. Operations Research, 1987, 35(2): 254-265.

[10] 邢文训,谢金星. 现代优化计算方法[M]. 2 版. 北京:清华大学出版社,2005.

[11] HO S C, HAUGLAND D. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries[J]. Computers & Operations Research, 2004, 31: 1947-1964.

(上接第 1642 页)

参考文献:

[1] STORN R, PRICE K. Differential evolution: a simple an efficient heuristic for global optimization over continuous spaces[J]. Journal Global Optimization, 1997, 11(4): 341-359.

[2] QIN A K, HUANG V L, SUGANTHAN P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Trans on Evolutionary Computation, 2009, 13(2): 398-417.

[3] DAS S, SUGANTHAN P N. Differential evolution: a survey of the state-of-the-art[J]. IEEE Trans on Evolutionary Computation, 2011, 15(1): 4-31.

[4] 杨卫东,姚峰,张明. 基于自适应交叉概率因子的差分进化算法及其应用[J]. 信息与控制,2010,32(2):187-193.

[5] GONG Wen-yin, CAI Zhi-hua, LING C X, *et al*. Enhanced differential evolution with adaptive strategies for numerical optimization[J]. IEEE Trans on System, Man, and Cybernetics, Part B: Cybernetics, 2011, 41(2): 397-413.

[6] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Opposition based differential evolution[J]. IEEE Trans on Evolutionary Computation, 2008, 12(1): 64-79.

[7] 池元成,方杰,蔡国飙. 中心变异差分进化算法[J]. 系统工程与电子技术,2010,32(5):1105-1108.

[8] 暴励,曾建潮. 一种双种群差分蜂群算法[J]. 控制理论与应用,2011,28(2):266-272.

[9] 吴燕玲,卢建刚,孙优贤. 基于免疫原理的差分进化[J]. 控制与决策,2007,22(11):1309-1312.

[10] 任子武,伞冶. 实数遗传算法的改进及性能研究[J]. 电子学报,2007,35(2):269-274.

[11] 刘朝华,章兢,李小花,等. 免疫协同微粒群进化算法的永磁同步电机多参数辨识模型方法[J]. 自动化学报,2012,38(10):1698-1708.

[12] GE Hong-wei, SUN Liang, LIANG Yan-chun, *et al*. An effective PSO and AIS-based hybrid intelligent algorithm for Job-Shop scheduling[J]. IEEE Trans on Systems, Man, and Cybernetics, Part A: System and Humans, 2008, 38(2): 358-368.