# 带容量约束的供应链物流运输调度问题的双层变邻域蝙蝠算法

戚远航[1],蔡延光[2],蔡 颢[3],杨 亮[1],YAO Yeboah[2]
(1.电子科技大学中山学院计算机学院,广东中山 528402;2.广东工业大学自动化学院,
广东广州 510006;3.奥尔堡大学健康科学与工程系,丹麦奥尔堡 9220)

**摘 要:** 本文考虑了多个供应商、多个制造商和多个零售商的三级供应链物流运输调度,以最大限度地降低采购、加工和运输成本为目标,提出了带容量约束的供应链物流运输调度模型(Capacitated Vehicle Routing Problem in Supply Chain,CVRPSC).进一步地,本文构造了求解CVRPSC的双层变邻域蝙蝠算法(Two-Level Bat Algorithm with Variable Neighborhood Search,TLBAVNS).该算法提出了一种双层蝙蝠位置的定义,引入了相应的蝙蝠算法的更新操作,采用变邻域局部搜索策略加强算法的寻优能力.实验证明:TLBAVNS能在合理的时间内求解CVRPSC;在大部分测试算例中,该算法相对于对比算法均表现出了更强的寻优能力和稳定性.

**关键词:** 供应链;车辆路径问题;蝙蝠算法;邻域搜索
**中图分类号:** TP301 **文献标识码:** A **文章编号:** 0372-2112 (2019)07-1434-09
**电子学报URL:** http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.07.006

# Two-Level Bat Algorithm with Variable Neighborhood Search for Capacitated Vehicle Routing Problem in Supply Chain

QI Yuan-hang[1],CAI Yan-guang[2],CAI Hao[3],YANG Liang[1],YAO Yeboah[2]
(1. *School of Computer Science,University of Electronic Science and Technology of China,Zhongshan Institute,Zhongshan,Guangdong* 528402,*China*;2. *School of Automation,Guangdong University of Technology,Guangzhou,Guangdong* 510006,*China*;
3. *Department of Health Science and Technology,Aalborg University,Aalborg* 9220,*Denmark*)

**Abstract:** Considering a two-level vehicle routing problem in a three-echelon supply chain with multiple suppliers, multiple manufacturers and multiple stockists,with the aim of minimizing the cost of purchasing,production and transportation,this paper proposes a model for the capacitated vehicle routing problem in supply chain (CVRPSC). Further,a two-level bat algorithm with variable neighborhood search (TLBAVNS) is presented to solve CVRPSC. The algorithm proposes a definition of two-level bat position and introduces the corresponding operations of the bat algorithm. In addition,a variable neighborhood local search is proposed to enhance the optimizing capability of TLBAVNS. The experiments have shown that TLBAVNS can effectively solve the instances of CVRPSC within a suitable amount of time and significantly outperforms all the other alternatives in most of the cases.

**Key words:** supply chain;vehicle routing problem;bat algorithm;neighborhood search

## 1 引言

随着经济全球化以及电子商务的快速发展,全球竞争日益激烈加剧,企业必须有效地优化供应链中的生产与配送过程,才能以最低的成本来满足当前严峻的市场需求[1].因此,供应链的优化成为了现代企业管

收稿日期:2018-07-09;修回日期:2018-11-19;责任编辑:孙瑶
基金项目:国家自然科学基金(No.61074147);广东省自然科学基金(No.S2011010005059,No.2016A030313018);广东省教育部产学研结合项目(No.2012B091000171,No.2011B090400460);广东省科技计划(No.2012B050600028,No.2014B010118004,No.2016A050502060);广州市花都区科技计划(No.HD14ZD001);广州市科技计划(No.201604016055);中山市科技计划重点项目(No.2018B1018);中山市重大科技专项(No.2017A1024,No.2017SF0603,No.2016A1028)

理的重要方面,成为了一个非常活跃的研究领域[2]. 物流运输调度是供应链中的重要问题之一,是降低供应链成本、提高服务质量的重要环节[3]. Uday 等[4]表示:产品成本的 30% 是用于进行运输调度,因此合理的物流运输调度可以降低运输成本,从而降低产品价格. 由此可见,对供应链物流运输调度问题进行研究是具有研究价值以及实际意义的.

供应链系统由供应商、制造商、仓库、分销商、零售商以及最终客户等节点组成[5]. 因此根据所考虑供应链节点的数量,供应链物流运输调度问题可以分为单节点供应链物流运输调度问题和集成供应链物流运输调度问题[6]. 单节点供应链物流运输调度问题主要是围绕供应链的某个节点,考虑不同的实际因素,进行针对性优化的问题,而集成供应链物流运输调度问题则是指对多个供应链节点进行优化调度而提出的问题. 其中,越库配送(Cross Docking,CD)[7]和生产路径问题(Production Routing Problem,PRP)[8]是集成供应链物流运输调度问题中典型的三级供应链物流运输调度问题,也是学者们在供应链物流运输调度问题上致力研究的重点和热点.

CD 是针对供应商、转运(库存)中心、零售商的三级供应链物流运输调度问题:商品由收货车辆收集并运送到转运中心,整合分拣后,再由发货车辆根据顾客的需求装载配送到目的地[9,10]. 相对于传统的高成本仓库存储策略,CD 可以有效地提高产品的流动性,同时减少存储成本、存储空间[11]. 因此,CD 在 1995 年由 Rohrer[12]首次提出后,便得到了学者们的关注和研究,并且迅速地在一些高运输成本的行业得到成功的应用[13,14]. 但是,目前对 CD 的研究中,多数学者均只关注供应链的货物整合配送的过程,很少考虑过生产过程对货物配送造成的影响. 另一方面,PRP 在 1994 年由 Chandra 等[15]提出,是针对制造商、转运(库存)中心、零售商的三级供应链物流运输调度问题. 与 CD 不同之处在于,PRP 同时考虑生产、库存和运输调度等环节[16,17]. 但是,PRP 只考虑了生产过程和生产之后的物流运输调度,没有考虑生产之前的物流运输调度,即供应商提供原料到制造商这阶段的物流运输调度,无法全面地体现供应链的特性.

由此可见,在供应链物流运输调度问题中,很少学者在考虑制造商加工过程的基础上,同时考虑供应商到制造商以及制造商到零售商的两层物流运输调度. 为了提高供应链的整体效率,本文考虑了供应商、制造商和零售商的三级供应链,考虑制造商生产因素的两层物流运输调度优化,提出了带容量约束的供应链物流运输调度模型(Capacitated Vehicle Routing Problem in Supply Chain,CVRPSC). 然而,对于 CVRPSC 这种组合优化问题,精确算法不能在合理的时间内得到最佳的解决,学者们常常启发式和元启发式算法来进行求解[18]. 基于本人之前的研究,离散蝙蝠算法(Discrete Bat Algorithm,DBA)在求解物车辆路径问题(Vehicle Routing Problem,VRP)及其相关问题时具有较强的寻优能力、较高的鲁棒性和较少的时间耗费[19,20]. 因此,在此基础上,本文设计了一种双层变邻域蝙蝠算法(Two-Level Bat Algorithm with Variable Neighborhood Search,TLBAVNS)求解 CVRPSC.

## 2　物流运输调度模型

### 2.1　问题描述

CVRPSC 分为两层物流运输调度,供应商到制造商的物流运输调度和制造商到零售商的物流运输调度,包含多个供应商、多个制造商和多个零售商. 其中,不同供应商的最大原料供应量、车辆的最大载重量以及配送成本不相同;不同制造商加工商品的成本、车辆的最大载重量以及车辆的配送成本不相同;不同零售商的商品需求不相同. 而 CVRPSC 的配送流程为:供应商根据零售商的需求提供原料,并通过供应商的车辆把原料配送到制造商进行加工;待原料加工成商品后,通过制造商的车辆按零售商的需求把商品运送到零售商. CVRPSC 配送流程示意图如图 1 所示,其包含了 4 个供应商,2 个制造商和 6 个零售商,共涉及 6 台供应商车辆和 3 台制造商车辆. CVRPSC 的目标在于确定如何采购供应商的原料,选择进行加工的制造商,确定供应商和制造商车辆的运输调度,并最大限度地降低采购、加工和运输的总成本.

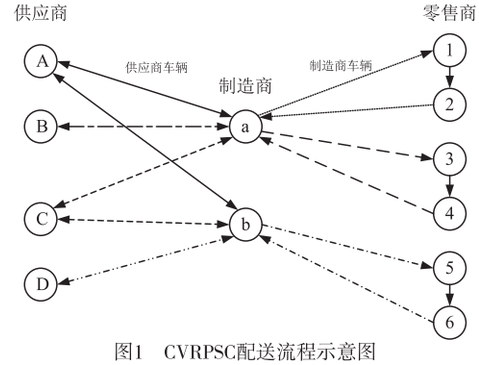


图1　CVRPSC配送流程示意图

### 2.2　基本假设

本文所提模型的基本假设如下所示.

(1)供应商提供原料的总量不能超过其最大的供应量.

(2)供应商的车辆一次性按需把原料配送到对应的制造商,配送完毕后车辆返回原供应商.

(3)加工前的原料和加工后的商品重量不变.

(4)制造商的车辆的起点和终点相同,且每台车辆最多只能配送商品一次,车辆不能重复使用.

(5)制造商车辆所配送的商品总重量不能超过该车辆的最大载重量.

(6)零售商的商品需求必须被满足且只能由一个的供应商、一个制造商和一台制造商车辆完成.

### 2.3 数学模型

#### 2.3.1 数学变量

$S$:供应商的数量.

$M$:制造商的数量.

$P$:零售商的数量.

$s$:供应商的索引.

$m$:制造商的索引.

$p$:零售商的索引.

$k$:制造商车辆的索引.

$e_s$:供应商 $s$ 的原料的最大供应量.

$c_s$:供应商 $s$ 的原料的单位重量的成本.

$b_s$:供应商 $s$ 的车辆配送原料的单位距离的配送成本.

$n_m$:制造商 $m$ 所拥有的车辆的数量.

$c'_m$:制造商 $m$ 把原料加工成商品的单位重量的加工成本.

$b'_{mk}$:制造商 $m$ 的车辆 $k$ 配送商品的单位距离的配送成本.

$u_{mk}$:制造商 $m$ 的车辆 $k$ 的最大载重量.

$d_{sm}$:供应商 $s$ 到制造商 $m$ 的距离.

$d'_{mp}$:制造商 $m$ 到零售商 $p$ 的距离.

$d''_{pp'}$:零售商 $p$ 到零售商 $p'$的距离.

$o_p$:零售商 $p$ 的商品需求(零售商 $p$ 需购买的原料的重量).

$z_{sm}$:如果供应商 $s$ 使用供应商车辆为制造商 $m$ 配送原料,$z_{sm}=1$;否则,$z_{sm}=0$.

$y_{mkp}$:如果零售商 $p$ 是制造商 $m$ 的车辆 $k$ 第一个经过的销售商,$y_{mkp}=1$;否则,$y_{mkp}=0$.

$y'_{mkpp'}$:如果制造商 $m$ 的车辆 $k$ 配送完零售商 $p$ 后直接配送零售商 $p'$,$y'_{mkpp'}=1$;否则,$y'_{mkpp'}=0$.

$y''_{mkp}$:如果零售商 $p$ 是制造商 $m$ 的车辆 $k$ 最后一个经过的销售商,$y''_{mkp}=1$;否则,$y''_{mkp}=0$.

$g_{smkp}$:如果制造商 $m$ 的车辆 $k$ 配送给给零售商 $p$ 的商品是由供应商 $s$ 提供的原料加工完成的,$g_{smkp}=1$;否则,$g_{smkp}=0$.

#### 2.3.2 目标函数

式(1)为目标函数,其目标在于最小化三级供应链物流运输调度过程中采购、加工和运输的总成本. 其中,式(1)中的第一项代表采购供应商原料的成本,第二项代表供应商配送原料到制造商的配送成本,第三项代表制造商加工成本,第四项和第五项代表制造商车辆配送商品的配送成本.

$$\begin{aligned}\min Z = &\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P} g_{smkp}o_p c_s + \\ &2\times\sum_{s=1}^{S}\sum_{m=1}^{M} z_{sm}d_{sm}b_s + \\ &\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P} g_{smkp}o_p c'_m + \\ &\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P}(y_{mkp}+y''_{mkp})d'_{mp}b'_{mk} + \\ &\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P}\sum_{p'=1}^{P} y'_{mkpp'}d''_{pp'}b'_{mk}\end{aligned} \tag{1}$$

#### 2.3.3 约束条件

$$\sum_{p=1}^{P} y_{mkp}\leqslant 1, m=1,2,\cdots,M, k=1,2,\cdots,n_m \tag{2}$$

$$\sum_{p=1}^{P}\sum_{p'=1}^{P'} y'_{mkpp'}\leqslant\delta\sum_{p=1}^{P} y_{mkp}, \tag{3}$$

$$m=1,2,\cdots,M, k=1,2,\cdots,n_m, \exists\delta\in R^+$$

$$y_{mkp}+\sum_{p'=1}^{P'} y'_{mkp'p}=\sum_{p'=1}^{P'} y'_{mkpp'}+y''_{mkp}, \tag{4}$$

$$m=1,2,\cdots,M, k=1,2,\cdots,n_m, p=1,2,\cdots,P$$

$$\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m} g_{smkp}=1, R_i \tag{5}$$

$$z_{sm}=\min\left\{1,\sum_{k=1}^{K}\sum_{p=1}^{P} g_{smkp}\right\}, \tag{6}$$

$$s=1,2,\cdots,S, m=1,2,\cdots,M$$

$$\sum_{m=1}^{M}\sum_{p=1}^{P} g_{smkp}o_p\leqslant e_s, \tag{7}$$

$$s=1,2,\cdots,S, k=1,2,\cdots,n_m$$

$$\sum_{s=1}^{S}\sum_{p=1}^{P} g_{smkp}o_p\leqslant u_{mk}, \tag{8}$$

$$s=1,2,\cdots,S, k=1,2,\cdots,n_m$$

其中,式(2)确保每台制造商车辆最多只能配送一次货物;式(3)确保制造商车辆必须从制造商出发后才能去配送零售商;式(4)确保制造商车辆在配送过程的连续性;式(5)确保每个零售商的商品需求只由一个供应商、一个制造商和一台制造商车辆完成;式(6)确保了供应商可一次性把原料配送到对应的制造商;式(7)确保供应商提供的原料不能超过其最大量;式(8)确保制造商车辆所配送的货物不超过车辆的最大载重量.

## 3 双层变邻域蝙蝠算法

蝙蝠算法(Bat Algorithm,BA)是由微型蝙蝠回声定位原理演变而成[21]:蝙蝠 $i$ 在位置 $x_i$ 以速度 $v_i$ 随意飞行,以频率 $f_i$ 和响度 $A_i$ 去搜索猎物,并根据目标物与自己的距离自动调节频率,并在靠近猎物时调整发射脉冲的频度 $R_i$. 近年来,在一些离散优化领域,BA 已得到了成功的应用,如背包问题[22]和无人机路径规划[23]

等. 本文提出的 TLBAVNS 的主要结构如下所述.

**step1**　初始化蝙蝠种群(种群大小为 $Q$),随机生成双层蝙蝠位置 $x_i$、双层蝙蝠速度 $v_i$、双层蝙蝠频率 $f_i$、响度 $A_i$ 和发射频度 $R_i$,计算每个蝙蝠的适应度,经过适应度比较后得出全局最优蝙蝠位置,$i=1,2,\cdots,Q$.

**step2**　应用蝙蝠更新操作来更新蝙蝠的位置、速度和频率.

**step3**　如果 $rand(\,)>R_i$,使用蝙蝠变异更新蝙蝠位置.

**step4**　应用变邻域局部搜索策略更新蝙蝠位置.

**step5**　如果蝙蝠位置的适应度更优且 $rand(\,)<A_i$,接受当前的蝙蝠位置,更新 $R_i$ 和 $A_i$.

**step6**　更新全局最优蝙蝠位置.

**step7**　重复 step2 到 step7 直到达到最大迭代次数 *MIT*.

### 3.1　双层蝙蝠位置

基于本人之前的研究[19,20],针对 CVRPSC 的特点,本文提出一种双层蝙蝠位置的定义. 蝙蝠 $i$ 的位置为

$$x_i=[x_i^1,x_i^2] \tag{9}$$

$$x_i^1=[x_{i1}^1,x_{i2}^1,\cdots,x_{iW}^1] \tag{10}$$

$$x_i^2=[x_{i1}^2,x_{i2}^2,\cdots,x_{iW'}^2] \tag{11}$$

(1)第一层 $x_i^1$ 的维度为 $W=P+S-1$,其是一个整数序列$[1,2,\cdots,W]$的一种置换,用来确定零售商的所需商品的原料由哪个供应商提供.

①$x_{ij}^1=1,2,\cdots,W(1\leqslant j\leqslant W)$分别代表着零售商的序号,以此确保每个零售商所需商品的原料均有供应商提供.

②$x_{ij}^1>P(1\leqslant j\leqslant W)$负责把 $x_i^1$ 分割成 $S$ 段. 从左往右,第 $s$ 段代表着供应商 $s$ 所负责的零售商的集合,$s=1,2,\cdots,S$.

(2)第二层 $x_i^2$ 的维度为 $W'=P+\sum_{m=1}^{M}n_m-1$,其是一个整数序列$[1,2,\cdots,W']$的一种置换,用来确定零售商的所需商品由哪个制造商加工,同时被哪台制造商车辆进行运送.

①$x_{ij}^2\leqslant P(1\leqslant j\leqslant W')$代表着零售商的序号,以此确保每个零售商所需商品均有制造商加工及运送.

②$x_{ij}^2>P(1\leqslant j\leqslant W')$负责把 $x_i^2$ 分割成 $\sum_{m=1}^{M}n_m$ 段,$j=1,2,\cdots,W'$. 从左往右,顺序为$\left[\sum_{i=1}^{m-1}n_i,\sum_{i=1}^{m}n_i\right]$的子段是属于制造商 $m$ 的车辆,$m=1,2,\cdots,M$. 进一步地,属于制造商 $m$ 的 $n_m$ 个子段,从左往右,顺序指定为制造商 $m$ 的车辆 $k$ 的配送路径,$k=1,2,\cdots,n_m$.

(3)因为 $x_i^1$ 确定了零售商的所需商品的原料由哪个供应商提供,同时 $x_i^2$ 又确定了零售商的所需商品由哪个制造商负责运输,因此易得出每个供应商的原料应该提供给哪个制造商加工.

### 3.2　适应度函数

通过双层蝙蝠位置的编解码可以得到供应商、制造商和零售商之间的配送路径,但无路径可能会违反式(7)和式(8)的约束. 进一步地,为了更好的控制算法的寻优方向,本文使用了一种惩罚机制和向量机制相结合的适应度改进策略[20]. 式(1)使用惩罚和向量机制后得到蝙蝠 $i$ 的适应度函数为:

$$G_i=\{G_{i1},G_{i2},G_{i3}\} \tag{12}$$

$$G_{i1}=\sum_{s=1}^{S}\max\{0,\sum_{m=1}^{M}\sum_{p=1}^{P}g_{smp}o_p-e_s\} \tag{13}$$

$$G_{i2}=\sum_{m=1}^{M}\sum_{k=1}^{n_m}\max\{0,\sum_{s=1}^{S}\sum_{p=1}^{P}g_{smkp}o_p-u_{mk}\} \tag{14}$$

$$G_{i3}=Z \tag{15}$$

其中,$G_{i1}$ 是违反式(6)约束的惩罚处理,$G_{i2}$ 是违反式(7)约束的惩罚处理,$G_{i3}$ 是原目标. 因为适应度函数采用向量的形式表达,因而可以采用基于向量的比较机制进行适应度的比较[20].

### 3.3　双层蝙蝠的更新操作

蝙蝠算法的迭代更新主要由蝙蝠位置更新操作、蝙蝠变异操作,蝙蝠响度和发射频度更新操作等 4 部分组成. 而蝙蝠位置更新操作需要使用蝙蝠速度和频率,因此首选需定义双层蝙蝠位置对应的双层蝙蝠速度和双层蝙蝠频率.

(1)蝙蝠 $i$ 的速度为

$$v_i=[v_i^1,v_i^2] \tag{16}$$

$$v_i^1=[v_{i1}^1,v_{i2}^1,\cdots,v_{iW}^1] \tag{17}$$

$$v_i^2=[v_{i1}^2,v_{i2}^2,\cdots,v_{iW'}^2] \tag{18}$$

其中,$v_{ij}^1(j=1,2,\cdots,W)$是整数且 $1\leqslant v_{ij}^1\leqslant W$;$v_{ij}^2(j=1,2,\cdots,W')$是整数且 $1\leqslant v_{ij}^2\leqslant W'$.

(2)蝙蝠 $i$ 的频率为

$$f_i=[f_i^1,f_i^2] \tag{19}$$

其中,$f_i^{'}=[f_i^{'1},f_i^{'2}]$,$f_r$ 是随机生成且 $f_r\in[0,1]$,频率影响因子 $\theta^1>1,\theta^2>1$.

由式(9)的定义以及编解码策略可知,第一层蝙蝠位置和第二层蝙蝠位置在算法的运行过程中是相对独立的,只是在求解适应度值时才会建立他们的关系. 由此易知,与双层蝙蝠位置相对应的速度、频率,其第一层和第二层信息也是独立的. 因此,本文令 $x_i^1$、$v_i^1$ 和 $f_i^1$ 对应一个独立 DBA,$x_i^2$、$v_i^2$ 和 $f_i^2$ 对应另一个独立 DBA,再结合 DBA[20]的更新操作,可得出本文算法相应的更新操作.

### 3.4　变邻域局部搜索策略

启发式算法独立求解 VRP 时容易出现陷入局部最

优、求解不稳定等问题[24]. 因此许多学者常常将启发式算法和局部搜索算法相结合,利用局部搜索算法的特性来加强算法的局部搜索能力、提高算法的收敛性能[25]. 变邻域局部搜,索算法是一种采用多个不同的邻域结构进行搜索的局部搜索算法. 近年来,变邻域局部搜索算法与启发式算法混合后应用于 VRP 取得较为理想的求解效果[26~28]. 例如,Jabir 等[26]结合蚁群算法和变邻域局部搜索算法求解多车场绿色 VRP; Sicilia 等[27]混合禁忌搜索算法和变邻域局部搜索算法求解广义 VRP; Xiao 等[28]提出了一种基于变邻域局部搜索的模拟退化算法求解带容量约束的 VRP. 因此,本文引入 2-Opt 搜索、0-1 搜索和 1-1 搜索[29,30],结合双层蝙蝠个体的定义,提出了一种变邻域搜索策略(Variable Neighborhood Search, VNS)来加强算法的局部搜索能力.

VNS 具体步骤如下:首先进行 0-1 搜索,当无法改进解时,则使用 1-1 搜索;如果 1-1 搜索能继续改进解,则退回到 0-1 搜索. 而 2-Opt 搜索是指对某台制造商车辆的配送路径进行 2-Opt 搜索,其主要应用在第二层蝙蝠位置的 0-1 搜索以及 1-1 搜索中[20].

VNS 为了避免搜索方向发生过大的扰动,在局部寻优的过程中,某一层蝙蝠位置改变时,另一层蝙蝠位置不改变. 而本文的 VNS 只是按照一定的规则来交换蝙蝠位置的分量的顺序,得到的蝙蝠位置仍然符合 3.1 节蝙蝠位置的编码定义,可以通过解码得到一个完整的 CVRPSC 路径.

## 4 实验与分析

### 4.1 实验算例

因为 CVRPSC 模型没有标准算例,本文按规律随机生成一些测试算例来评估算法. 测试用例主要有 4 组,每组随机生成 5 个算例,一共 20 个测试算例,分别为 P01, P02, …, P20. 其算例的规模和参数的取值范围如表 1、表 2 所示.

**表 1 算例的规模**

| Group | $S$ | $M$ | $n_m$ | $P$ |
|---|---|---|---|---|
| G1 | 3 | 2 | 3 | 10 |
| G2 | 4 | 3 | 3 | 20 |
| G3 | 6 | 3 | 4 | 30 |
| G4 | 10 | 5 | 5 | 50 |

**表 2 算例参数的取值范围**

| 参数 | 取值范围 | 参数 | 取值范围 |
|---|---|---|---|
| 点坐标 | rand (0,500) | $u_{mk}$ | rand(1,3) ×10 |
| $e_s$ | rand (20,40) | $c'_m$ | rand(1500,1800) |
| $c_s$ | rand(3000,3200) | $b'_{mk}$ | rand(3,5) |
| $b_s$ | rand(10,15) | $o_p$ | rand(1,10) |

### 4.2 实验环境与算法参数设置

TLBAVNS 是在本人之前研究的 DBA[19,20] 的基础上设计而成,其每层算子均对应一个独立 DBA. 虽然 TLBAVNS 为双层结构,但是其算子的内部设计以及算法的总体架构、逻辑关系均与 DBA 相同. 因此,本文参考之前工作中的 DBA 参数实验[20]来设置 TLBAVNS,使 TLBAVNS 具有较好的寻优能力,参数设置为: $Q = 30$, $L = 100$, $MIT \leqslant 2000$, $\theta^1 = W$, $\theta^2 = W'$, $\alpha = 0.999$, $\gamma = 0.001$. 本文算法的所有实验均在同一实验环境中进行,其中 CPU 主频为 2.30GHz,内存为 4GB,操作系统为 32 位 Windows 7,编程语言为 C ++.

### 4.3 实验结果与分析

**实验 1** 为了准确地分析 TLBAVNS 求解 CVRPSC 的有效性,本实验选用 P01 算例的实验结果进行具体实例分析. P01 算例的信息如表 3、表 4 和表 5 所示,TLBAVNS 求解 P01 的实验结果如表 6 所示.

**表 3 P01 的供应商信息**

| 供应商 | X 坐标 | Y 坐标 | 单位配送成本 | 单位原料成本 | 最大供应量 |
|---|---|---|---|---|---|
| SUP1 | 280 | 29 | 11 | 3001 | 29 |
| SUP2 | 295 | 178 | 12 | 3043 | 40 |
| SUP3 | 270 | 57 | 13 | 3066 | 36 |

**表 4 P01 的制造商信息**

| 制造商 | X 坐标 | Y 坐标 | 单位配送成本 | 单位加工成本 | 车辆数 | 车辆信息 | |
|---|---|---|---|---|---|---|---|
| | | | | | | 序号 | 最大载重量 |
| MAN1 | 464 | 98 | 3 | 1511 | 3 | VEH1 | 20 |
| | | | | | | VEH2 | 30 |
| | | | | | | VEH3 | 20 |
| MAN2 | 482 | 276 | 4 | 1744 | 3 | VEH4 | 30 |
| | | | | | | VEH5 | 30 |
| | | | | | | VEH6 | 20 |

**表 5 P01 的零售商信息**

| 零售商 | X 坐标 | Y 坐标 | 商品需求 |
|---|---|---|---|
| STO1 | 191 | 275 | 9 |
| STO2 | 306 | 102 | 1 |
| STO3 | 304 | 461 | 5 |
| STO4 | 457 | 139 | 2 |
| STO5 | 100 | 19 | 10 |
| STO6 | 162 | 380 | 5 |
| STO7 | 496 | 128 | 6 |
| STO8 | 173 | 222 | 2 |
| STO9 | 177 | 339 | 5 |
| STO10 | 225 | 449 | 10 |

**表 6　P01 的最优解**

| 名称 | 集合/路径信息 | 原料供应量 | 原料成本 | 车辆所用载重量 | 加工成本 | 路径长度 | 配送成本 |
|---|---|---|---|---|---|---|---|
| 供应商所负责的零售商集合 | SUP1:STO6,STO10,STO4,STO8,STO5 | 29 | 87029 | | | | |
| | SUP2:STO3,STO1,STO2,STO9,STO7 | 26 | 79118 | | | | |
| 制造商到零售商的配送路径 | VEH1:MAN1 - >STO7 - >MAN1 | | | 6 | 9066 | 86 | 258 |
| | VEH2:MAN1 - >STO4 >STO3 - >STO10 - >STO6 - >STO9 - >STO8 - >MAN1 | | | 29 | 43819 | 1045 | 3135 |
| | VEH3:MAN1 - >STO2 - >STO5 - >STO1 - >MAN1 | | | 20 | 30220 | 976 | 2928 |
| 供应商到制造商的配送路径 | SUP1 - > MAN1 | | | | | 392 | 4312 |
| | SUP2 - > MAN1 | | | | | 372 | 4464 |
| 成本总和 | | | 166147 | | 83105 | | 15097 |
| | | 264349 | | | | | |

从表 6 可以看出:

(1)TLBAVNS 的求解 P01 的最优解为 264349. 在该最优解中:SUP1 提供的原料量为 29,SUP2 提供的原料量为 26;使用了 MAN1 的 3 台车辆,VEH1 载重量为 6,VEH2 的载重量为 29,VEH3 载重量为 20. 该最优解满足 CVRPSC 的供应量约束和最大载重量约束,其是一个可行解.

(2)在 P01 的最优解中,没有选择 SUP3 的原料和配送. 因为 SUP3 的单位配送成本和单位原料成本均是三个供应商中最贵的,SUP1 和 SUP2 的最大供应量的和已经满足了所有零售商的货物需求量,此外,相对于 SUP1 和 SUP2,SUP3 与制造商们的距离这方面也没有表现出太大的优势,所以应优先选择 SUP1、SUP2,符合现实生活中的选择决策表现.

(3)在 P01 的最优解中,没有选择 MAN2 进行加工和配送. 因为 MAN2 的单位加工成本和单位配送成本比较贵,而距离供应商也比较远,所以应优先选择 MAN1,也符合现实生活的选择决策表现.

综上所述,TLBAVNS 能有效地求解 CVRPSC,且最优解也符合现实生活的选择决策表现.

**实验 2**　为了更好的深入了解 TLBAVNS 的求解能力,本文使用 4 个算法与其比较. 第 1 个算法是双层蝙蝠算法(Two-Level Bat Algorithm,TLBA),即不加入 VNS 的 TLBAVNS;第 2 个算法是基于 VNS 的启发式算法(Heuristic Algorithm based on Variable Neighborhood Search,HAVNS):HAVNS 以随机种群开始,并通过式(17)进行适应度计算和比较;在往后的迭代中,每层蝙蝠位置分别随机交换该层的两个分量得到一个蝙蝠位置,然后再进行 VNS 搜索得到一个新的蝙蝠位置,如果新的蝙蝠位置优于当前的蝙蝠位置,它将成为新的蝙蝠位置,进一步地,如果优于全局最优蝙蝠位置,则更新全局最优蝙蝠位置;第 3 个算法是粒子群算法(Particle Swarm Optimization,PSO)[31];第 4 个算法是蚁群算法(Ant Colony Optimization,ACO)[32]. 为了确保实验在公平的环境中进行,本文调整 MIT 使 5 个算法的运行时间相同. 本实验选用 20 个算例,实验结果如表 7 所示. 因为测试算例没有已知最优解,而 CVRPSC 的目标函数为式(1),因此本文假设式(1)中的零售商的商品需求不改变,其他变量均由可选择项中的最优值代替,从而得到一个理论最优解 S_L,其计算方式如式(20)所示. 此外,使用 t-检验中的成对双样本均值分析方法来检验 5 个算法之间是否存在显著性差异. 设显著性水平为 0.05,平均差为 0,利用 EXCEL 的"数据分析"功能对 5 个算法的 G_M 和 G_A 进行假设检验,实验结果如表 8 所示.

$$
\begin{aligned}
S\_L = {} & \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P} g_{smkp}o_p \min_s\{c_s\} + 2\times\left(\left\lceil \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P} g_{smkp}o_p / \max_s\{e_s\}\right\rceil\right)\times \min_{s,m}\{d_{sm}\}\times \min_s\{b_s\} + \\
& \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P} g_{smp}o_p \times \min_m\{c'_m\} + 2\times\left(\left\lceil \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P} g_{smkp}o_p / \max_{m,k}\{u_{mk}\}\right\rceil\right)\times \min_{m,p}\{d'_{mp}\}\ \min_{m,k}\{b'_{mk}\} + \\
& \left(p-1-\left(\left\lceil \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{k=1}^{n_m}\sum_{p=1}^{P} g_{smkp}o_p / \max_{m,k}\{u_{mk}\}-1\right\rceil\right)\right)\times \min_{p,p\neq p'}\{d'_{pp'}\}\times \min_{m,k}\{b'_{mk}\} \qquad (20)
\end{aligned}
$$

表 7 TLBAVNS、TBA 和 HAVNS 的实验结果比较

| 算例名称 | | S_L | TLBAVNS | | TLBA | | HAVNS | | PSO | | ACO | | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | G_M | G_A | G_M | G_A | G_M | G_A | G_M | G_A | G_M | G_A | |
| G1 | P01 | 257916 | **2.49** | **2.49** | **2.49** | 2.55 | **2.49** | **2.49** | 2.64 | 2.66 | **2.49** | 2.55 | 8 |
| | P02 | 336726 | **3.97** | **3.97** | 4.00 | 4.57 | **3.97** | **3.98** | 4.62 | 4.70 | 4.01 | 4.55 | 6 |
| | P03 | 337156 | **5.53** | **5.53** | **5.53** | 5.82 | **5.53** | **5.53** | 6.13 | 6.16 | 5.54 | 5.85 | 7 |
| | P04 | 237608 | **4.84** | **4.84** | **4.84** | 4.84 | **4.84** | **4.84** | **4.84** | 4.85 | **4.84** | 4.85 | 8 |
| | P05 | 350821 | **5.66** | **5.66** | **5.66** | 5.79 | **5.66** | **5.66** | 5.73 | 5.75 | **5.66** | 5.81 | 7 |
| G2 | P06 | 530488 | **7.70** | 7.91 | 8.59 | 9.17 | 7.87 | 7.96 | 9.55 | 9.84 | 9.10 | 9.33 | 70 |
| | P07 | 610725 | **7.59** | 8.08 | 10.82 | 11.44 | 7.86 | 8.16 | 12.83 | 14.08 | 11.68 | 11.92 | 45 |
| | P08 | 528354 | **5.19** | 5.57 | 7.38 | 7.85 | 5.20 | **5.43** | 7.92 | 8.57 | 7.59 | 7.92 | 46 |
| | P09 | 513742 | **6.86** | 6.89 | 7.10 | 7.42 | **6.86** | **6.87** | 7.83 | 8.15 | 7.36 | 7.51 | 64 |
| | P10 | 532313 | **4.99** | 5.11 | 5.39 | 5.82 | 5.10 | 5.20 | 5.90 | 6.19 | 5.43 | 5.84 | 64 |
| G3 | P11 | 691414 | **7.86** | 8.41 | 8.83 | 9.33 | 7.91 | **8.31** | 10.33 | 11.04 | 9.24 | 10.64 | 118 |
| | P12 | 721967 | **6.74** | 6.89 | 7.48 | 7.67 | 6.75 | 6.89 | 8.56 | 8.97 | 7.55 | 7.72 | 112 |
| | P13 | 714055 | **5.87** | 6.25 | 7.04 | 7.43 | 6.20 | 6.39 | 8.26 | 8.76 | 7.11 | 7.43 | 115 |
| | P14 | 847848 | **6.13** | 6.26 | 6.66 | 7.75 | 6.16 | 6.30 | 9.59 | 10.57 | 7.40 | 7.82 | 100 |
| | P15 | 710220 | **5.72** | 6.09 | 6.33 | 6.72 | 5.95 | **6.08** | 8.17 | 8.24 | 6.54 | 6.77 | 105 |
| G4 | P16 | 1307289 | **7.72** | 8.16 | 10.27 | 10.87 | 8.08 | 8.24 | 17.48 | 18.07 | 10.85 | 11.16 | 171 |
| | P17 | 1213795 | **8.41** | 8.86 | 9.73 | 10.25 | 8.43 | **8.75** | 14.94 | 15.68 | 9.92 | 10.34 | 187 |
| | P18 | 1382146 | **6.70** | 6.93 | 8.42 | 8.69 | 6.75 | **6.92** | 14.50 | 14.63 | 8.59 | 9.03 | 210 |
| | P19 | 1258913 | **7.38** | 7.84 | 9.27 | 9.81 | 7.66 | 7.87 | 13.93 | 15.79 | 9.72 | 10.03 | 190 |
| | P20 | 1081173 | **7.80** | 7.99 | 9.61 | 10.26 | 7.82 | 8.00 | 15.30 | 15.57 | 10.09 | 10.41 | 208 |

S_L:理论最优解;G_M:最优解相对误差,G_M(%) = (S_M - S_L)/S_L × 100,S_M 为算法独立运行 10 次后获得的最优解;G_A:平均解相对误差,G_A(%) = (S_A - S_L)/S_L × 100,S_A 是算法独立运行 10 次后获得的平均解;Time:运行时间,单位为 s.

表 8 t-检验:成对双样本均值分析

| t-检验 | P 值 | |
|---|---|---|
| | G_M | G_A |
| TLBAVNS vs TLBA | **0.00013** | **0.00001** |
| TLBAVNS vs HAVNS | **0.00255** | 0.65961 |
| TLBAVNS vsPSO | **0.00012** | **0.00009** |
| TLBAVNS vs ACO | **0.00008** | **0.00001** |

从表 7、表 8 可以看出:

(1)TLBAVNS 求解所有算例的 G_M 均不大于其他 4 个算法,说明了 TLBAVNS 的寻优能力不差于其他 4 个算法. TLBAVNS 的平均解远远优于 TLBA、PSO、ACO,略优于 HAVNS. 对于个别算例,HAVNS 的平均解甚至优于 TLBAVNS,但是 TLBAVNS 的最优解却更优. 由此可见,VNS 极大地提高 TLBAVNS 的稳定性,而 TLBA 也赋予了 TLBAVNS 更强的全局寻优能力. 特别地,在 P16 算例中,TLBAVNS 的 G_M 和 G_A 为 7.72% 和 8.16%,比 HAVNS 的降低了 0.36% 和 0.08%,远远优于 TLBA 的 10.27% 和 10.87%、PSO 的 17.48% 和 18.07%、ACO 的 10.85% 和 11.16%.

(2)在假设检验中,$P > 0.05$ 表示差异性不显著;$0.01 < P < 0.05$ 表示差异性显著;$P < 0.01$ 表示差异性极显著. 显而易见,在 G_M 的 t 检验中,TLBAVNS 相对于其他 4 算法均表现出显著的差异性;而在 G_A 的 t 检验中,除了 HAVNS,TLBAVNS 相对于 TLBA、PSO、ACO 表现出极显著的差异性.

综上所述,虽然 TLBAVNS 求解个别算例的结果不如对比算法,但是整体而言,TLBAVNS 表现出了更强的寻优能力和稳定性,与对比算法的 G_M 也存在极显著的差异.

## 5 结论

本文基于三级供应链的情况,考虑了两层的物流运输调度,提出了 VRPSCM 模型,并构造了求解 VRPSCM 的 TLDBAVNS. TLDBAVNS 提出了双层蝙蝠位置以及相应的更新操作,使用惩罚机制和向量机制相结合的适应度改进策略来控制算法的寻优方向,引入变邻域局部搜索策略加强算法的局部搜索能力. 实验证明:VNS 极大地提高 TLDBAVNS 的稳定性,而 TLDBA 也赋予了 TLDBAVNS 更强的全局寻优能力;在大部分测试算例中,TLDBAVNS 的最优解和平均解均优于对比算法. 在未来的工作中,我们可以尝试考虑供应链的时间约束(如运输时间、加工时间、服务时间等等)来改进模型. 另一方面,我们也可以使用更多的策略来改进 TLBAVNS 算法的求解模型的能力.

### 参考文献

[1] CÓCCOLA M E,ZAMARRIPA M,MÉNDEZ C A,et al. Toward integrated production and distribution management in multi-echelon supply chains[J]. Computers & Chemical Engineering,2013,57(41):78 - 94.

[2] PAPAGEORGIOU L G. Supply chain optimisation for the process industries: Advances and opportunities[J]. Computers & Chemical Engineering, 2009, 33(12): 1931-1938.

[3] SETAK M, HABIBI M, KARIMI H, et al. A time-dependent vehicle routing problem in multigraph with FIFO property[J]. Journal of Manufacturing Systems, 2015, 35(35): 37-45.

[4] Uday M APTE, VISWANATHAN S. Effective cross docking for improving distribution efficiencies[J]. International Journal of Logistics Research & Applications, 2000, 3(3): 291-302.

[5] RAFIE-MAJD Z, PASANDIDEH S H R, NADERI B. Modelling and solving the integrated inventory-location-routing problem in a multi-period and multi-perishable product supply chain with uncertainty: Lagrangian relaxation algorithm[J]. Computers & Chemical Engineering, 2018, 109: 9-22.

[6] POURHEJAZY P, KWON O K. The new generation of operations research methods in supply chain optimization: A review[J]. Sustainability, 2016, 8(10): 1033.

[7] KUMAR A, ZHANG K Y J. A cross docking pipeline for improving pose prediction and virtual screening performance[J]. Journal of Computer-Aided Molecular Design, 2018, 32(1): 163-173.

[8] QIU Y, WANG L, XU X, et al. Formulations and branch-and-cut algorithms for multi-product multi-vehicle production routing problems with startup cost[J]. Expert Systems with Applications, 2018, 98: 1-10.

[9] ENDERER F, CONTARDO C, CONTRERAS I. Integrating dock-door assignment and vehicle routing with cross-docking[J]. I Computers & Operations Research, 2017, 88: 30-43.

[10] Golshahi-Roudbaneh A, Hajiaghaei-Keshteli M, Paydar M M. Developing a lower bound and strong heuristics for a truck scheduling problem in a cross-docking center[J]. Knowledge-Based Systems, 2017, 129: 17-38.

[11] WEN M, LARSEN J, CLAUSEN J, et al. Vehicle routing with cross-docking[J]. Journal of the Operational Research Society, 2009, 60(12): 1708-1718.

[12] ROHRER M. Simulation and cross docking[A]. Proceedings of the 1995 Simulation Conference[C]. USA: IEEE, 1995. 846-849.

[13] SHARMA S. Supply chain management: Concepts, practices & implementation, 1/e[J]. Journal of Cellular Biochemistry, 2008, 105(2): 524-533.

[14] YU W, EGBELUB P J. Scheduling of inbound and outbound trucks in cross docking systems with temporary storage[J]. European Journal of Operational Research, 2008, 184(1): 377-396.

[15] CHANDRA P, FISHER M L. Coordination of production and distribution planning[J]. European Journal of Operational Research, 1994, 72(3): 503-517.

[16] LIU S. On the integrated production, inventory, and distribution routing problem[J]. IIE Transactions, 2006, 38(11): 955-970.

[17] QIU Y, QIAO J, PARDALOS P M. A branch-and-price algorithm for production routing problems with carbon cap-and-trade[J]. Omega, 2017, 68: 49-61.

[18] KONUR D, GOLIAS M M. Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach[J]. Transportation Research Part E Logistics & Transportation Review, 2013, 49(1): 71-91.

[19] 戚远航,蔡延光,蔡颢,等. 旅行商问题的混沌混合离散蝙蝠算法[J]. 电子学报,2016,44(10):2543-2547.
QI Yuan-hang, CAI Yan-guang, CAI Hao, et al. Chaotic hybrid discrete bat algorithm for traveling salesman problem[J]. Acta Electronica Sinica, 2016, 44(10): 2543-2547. (in Chinese)

[20] 戚远航,蔡延光,蔡颢,等. 带时间窗的车辆路径问题的离散蝙蝠算法[J]. 电子学报,2018,46(3):672-679.
QI Yuan-hang, CAI Yan-guang, CAI Hao, et al. Discrete bat algorithm for vehicle routing problem with time window[J]. Acta Electronica Sinica, 2018, 46(3): 672-679. (in Chinese)

[21] YANG X S. A new metaheuristic bat-inspired algorithm[J]. Computer Knowledge & Technology, 2010, 284: 65-74.

[22] RIZK-ALLAH R M, HASSANIEN A E. New binary bat algorithm for solving 0-1 knapsack problem[J]. Complex & Intelligent Systems, 2018, 4(1): 31-53.

[23] WANG G G, CHU H C E, MIRJALILI S. Three-dimensional path planning for UCAV using an improved bat algorithm[J]. Aerospace Science & Technology, 2016, 49: 231-238.

[24] 李阳,范厚明. 求解带容量约束车辆路径问题的混合变邻域生物共栖搜索算法[J]. 控制与决策,2018,33(7):1190-1198.
LI Yang, FAN Hou-ming. Hybrid variable neighborhood symbiotic organisms search for capacitated vehicle routing problem[J]. Control and Decision, 2018, 33(7): 1190-1198. (in Chinese)

[25] ISHIBUCHI H, YOSHIDA T, MURATA T. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling[J]. IEEE Transactions on Evolutionary Computation, 2003, 7(2): 204-223.

[26] JABIR E, PANICKER V V, SRIDHARAN R. Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem[J]. Transportation Research Part D Transport & Environment, 2017, 57: 422 – 457.

[27] SICILIA J A, QUEMADA C, ROYO B. An optimization algorithm for solving the rich vehicle routing problem based on Variable Neighborhood Search and Tabu Search metaheuristics[J]. Journal of Computational & Applied Mathematics, 2016, 291(C): 468 – 477.

[28] XIAO Y, ZHAO Q, KAKU I, et al. Variable neighbourhood simulated annealing algorithm for capacitated vehicle routing problems[J]. Engineering Optimization, 2014, 46(4): 562 – 579.

[29] HIERMANN G, PUCHINGER J, ROPKE S, et al. The electric fleet size and mix vehicle routing problem with time windows and recharging stations[J]. European Journal of Operational Research, 2016, 252(3): 995 – 1018.

[30] 周永权, 黄正新, 刘洪霞. 求解 TSP 问题的离散型萤火虫群优化算法[J]. 电子学报, 2012, 40(6): 1164 – 1170.
ZHOU Yong-quan, HUANG Zheng-xin, LIU Hong-xia. Discrete glowworm swarm optimization algorithm for TSP problem[J]. Acta Electronica Sinica, 2012, 40(6): 1164 – 1170. (in Chinese)

[31] ZHANG Y, WANG S, JI G. A comprehensive survey on particle swarm optimization algorithm and its applications[J]. Mathematical Problems in Engineering, 2015, 1: 1 – 38.

[32] SRIKANTH G U, GEETHA R. Task scheduling using ant colony optimization in multicore architectures: a survey[J]. Soft Computing, 2018, 22(15): 5179 – 5196.

## 作者简介

**戚远航** 男, 1993 年 6 月出生, 广东湛江人. 2018 年在广东工业大学获得工学博士学位, 现为电子科技大学中山学院讲师, 从事复杂系统建模与优化、智能规划、运输调度的研究.
E-mail: qiyuanhang77@163.com

**蔡延光(通信作者)** 男, 1963 年 2 月出生, 湖北咸宁人. 1988 年在重庆大学获理学硕士学位, 1996 年在浙江大学获工学博士学位. 现为广东工业大学教授, 博士生导师, 从事复杂网络系统建模、控制与优化、物流控制与优化、智能交通系统、组合优化、智能优化、物联网信息处理与优化控制等方面的研究.
E-mail: caiyg99@163.com