基于超启发式算法的选址-路径问题研究

王万良,徐　昶,赵燕伟,朱文成
(浙江工业大学 计算机科学与技术学院,浙江 杭州 310023)

摘要:为了降低物流配送过程中车辆的碳排放,采用具有良好通用性的超启发式算法对低碳选址路径问题进行求解。将蛙跳算法作为超启发式算法的高层选择策略,并在蛙跳算法中提出了基于最长公共子序列的相似度计算方式代替原有的相似度计算,而采用动态规划的方法对个体间的最长公共子序列进行计算。实验结果表明:提出的相似度计算方式能更直观地反映个体之间的相似性,具有良好的通用性,并且在低碳选址-路径问题上获得更优秀的解。

关键词:超启发式算法;选择策略;蛙跳选择;最长公共子序列
中图分类号:TP183　　**文献标志码**:A　　**文章编号**:1006-4303(2019)06-0604-07

Research on location-routing problem based on hyper-heuristic algorithm

WANG Wanliang, XU Chang, ZHAO Yanwei, ZHU Wencheng
(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: In order to reduce the carbon emission of vehicles, the hyper-heuristic algorithm with good universality is used to solve the low-carbon location-routing problem. This paper uses leapfrog algorithm as the top choice strategy of hyper-heuristic algorithms, and proposes the similarity calculation method based on longest common subsequence instead of the original similarity calculation in leapfrog algorithm, and uses the dynamic programming to calculate the longest common subsequence between individuals. The experimental results show that the new similarity calculation method can more intuitively reflect the similarities between individuals, and it makes hyper-heuristic algorithm get better solutions in location-routing problem.

Keywords: hyper-heuristic algorithm; selection strategy; leapfrog algorithm; longest common subsequence

选址-路径问题(Location-routing problem, LRP)将选址分配问题(Location allocation problem, LAP)与经典车辆路径问题(Vehicle routing problem, VRP)进行组合研究,属于经典的 NP 难问题。随着全球气候变暖,低碳问题得到了越来越多人的重视,而物流业是碳排放的主要来源之一,因此对于低碳物流模型的研究日趋增多。在模型上,曹剑东等[1-2]以总成本最小化为优化目标研究了考虑碳排放因素的车辆路径问题(Vehicle routing problem, VRP)。在求解算法上,基本以遗传算法、禁忌搜索等启发式算法居多,但这些算法不具有良好的通用性,无法适用于新研究出来的模型。

超启发式算法[3](Hyper-heuristic algorithms)是近年才发展起来的一种新型的启发式算法,可以简单阐述为"寻找启发式算法的启发式算法",该算法具有良好的通用性,可用于求解排课问题[4]、流水车间调度问题[5]和装箱问题[6]等约束较少的组合优化问题。超启发式算法的研究主要集中在选择策

收稿日期:2018-06-06
基金项目:国家自然科学基金资助项目(61572438)
作者简介:王万良(1957—),男,江苏高邮人,教授,博士,博士生导师,研究方向为计算机控制与智能自动化、智能调度等,E-mail:zjutwwl@zjut.edu.cn。

略、接收策略和底层算子三部分。文献[7]研究了针对旅行锦标赛问题的基于蚁群的超启发式算法，采用蚁群算法来管理和操纵 LLH 以获得新的启发式算法，每只蚂蚁均构造一个新的启发式算法。Marshall 等[8]将语法演化应用在超启发式算法中，使其产生良好的解，通过仿真实验，对 40 个著名的车辆路径问题进行仿真，取得了不错的结果。国内外关于超启发式算法在低碳 LRP 模型中的应用研究甚少，笔者将基于蛙跳算法的选择策略用于启发式算法对低碳 LRP 问题进行求解，并提出了基于最长公共子序列的相似度计算方式。

1　问题描述

1.1　低碳 LRP 模型

本实验采用的模型[9]为多个配送中心使用单一类型车辆为每个客户点配送货物，不考虑时间窗、车辆最大行驶距离等约束条件。此问题描述为：有 n 个客户点需要配送货物，有 M 个仓库点进行发送货物，每个仓库都只有一种类型的车，车的载重都为 Q_k。每个客户点都由某一个仓库点进行货物发送，每一辆车都从仓库出发，前往每一个客户点去送货，直至送完所有货物，最后返回仓库。具体详情见表 1。

表 1　符号与变量说明

Table 1　Explanation of variable and symbol

符号与变量	含义	取值范围
M	配送中心	$m=\{1,2,\cdots,n\}$
C	客户点	$c=\{1,2,\cdots,n\}$
V	配送车辆	$v=\{1,2,\cdots,n\}$
T_m	配送中心 m 的容量	由具体测试样本确定
Q_k	车辆 k 的载重	由具体测试样本确定
L_c	客户 c 的需求	由具体测试样本确定
d_{ij}	车辆从节点 i 到节点 j 的距离	不定
η	燃油转换系数	2.68
P_0	车辆空载时单位距离燃油消耗	0
P^*	车辆满载时单位距离燃油消耗	1
q_{ijv}^m	仓库 m 的车辆 v 从客户点 i 行驶到客户点 j 时车辆所装在的货物重量	不定
x_{ijv}^m	配送中心 m 车辆 v 从节点 i 到节点 j 时	1
	其他	0
Z_v	车辆 v 的固定碳排放量	由具体测试样本确定

目标函数为

$$\min \psi = \sum_{m\in M}\sum_{i\in C}\sum_{j\in C}\sum_{v\in V}\eta x_{ijv}^m d_{ij} \cdot (P_0 + \frac{P^* - P_0}{Q_k} q_{ij}^m) + \sum_{v\in V} Z_v \tag{1}$$

约束条件为

$$\sum q_{ijv}^m < Q_k \quad i \in \{M,C\},\ j \in \{M,C\},\ \forall v \in V \tag{2}$$

$$\sum_{c\in C} L_c < T_m, \quad \forall m \in M \tag{3}$$

$$\sum_{m\in M}\sum_{v\in V}\sum_{j\in\{M,C\}} X_{ijv}^m = 1, \quad \forall i \in C \tag{4}$$

$$\sum_{i\in C} X_{imv} - \sum_{j\in C} X_{mjv} = 0, \quad \forall m \in M, v \in V \tag{5}$$

$$\sum_{i\in\{M,C\}} y_{ij} - \sum_{m\in\{M,C\}} y_{jm} = L_j, \quad \forall j \in C \tag{6}$$

式(1)是 LRP 模型中车辆碳排放量的计算方法，即所有车辆完成配送任务后所产生的碳排放量，也是本实验的目标函数，是在张春苗等[9]所提计算方法的基础上取消了配送中心开放的固定碳排放量，增加了每一辆车的固定碳排放量，以鼓励减少车辆的使用；式(2)表示车的货物总量总是小于车的载重；式(3)保证每个配送中心访问的顾客总需求小于配送中心的容量；式(4)保证每个客户均被访问一次；式(5)保证每一辆车从配送中心出发最终又回到配送中心；式(6)保证每一个客户点的需求都被满足。

1.2　算子介绍

根据超启发式算法的特点，底层启发式算子属于问题域，由应用领域的专家所提供。在低碳 LRP 模型中，根据文献[10]，笔者采用的底层启发式算子为

LLH1：2-opt 算子。选择一条路径，客户数量为 N，从前 $N-1$ 个客户中随机选举一个客户，将这个客户点和其后面相邻的客户点进行交换。

LLH2：or-opt 算子。选择一条路径，客户数量为 $N(N>2)$，随机选取相邻的两个客户点，将这两个客户点随机插入到该路径的其他位置。

LLH3：shift 算子。选择两条路径，从第一条路径中，随机选取一个客户点，将它插入到第二条路径合适的位置。

LLH4：Interchange。选择两条路径，随机从两条路径中各选取一个客户点进行交换。

LLH5：配送中心的 shift 算子。选择一条路径，给这条路径设置一个新的配送中心。

LLH6：配送中心的 Interchange 算子。选择两条路径，交换它们的配送中心。

LLH7：Location-based Radial Ruin。选择一条

路径，随机选举其中一个客户点，将此客户点作为基准客户，根据每个客户和基准客户的欧氏距离来重新生成解。

LLH8：Interchange*。与之前的 Interchange 算子一样，只不过此算子只接受改进解。

LLH9：shift*。与之前的 shift 算子一样，只不过此算子只接受改进解。

LLH10：2-opt*。与之前的 2-opt 算子一样，只不过此算子只接受改进解。

2 基于蛙跳算法的选择策略

选择式超启发算法根据反馈机制来源不同，可以分为不学习、离线学习和在线学习 3 种，而在线学习机制又可分为基于元启发、基于强化学习和基于函数选择 3 种选择方法，笔者使用了蛙跳算法（SFLA）来选择底层启发式算法，是在线学习机制中基于元启发选择方法的一种。SFLA 模拟青蛙寻找食物的过程，是一种新型群智能进化算法，它是由 Eusuff 和 Lansey[11] 于 2003 年首次提出，并应用在求解水资源网络的管径选择和管网扩张问题中。笔者提出一种基于 SFLA 的上层选择策略，将蛙跳算法应用在选择式超启发，其框架如图 1 所示，在原有的基本框架上，加入了 SFLA 算法用于低层启发式算法的选择，同时根据应用算子后的结果反馈于 SFLA 算法。

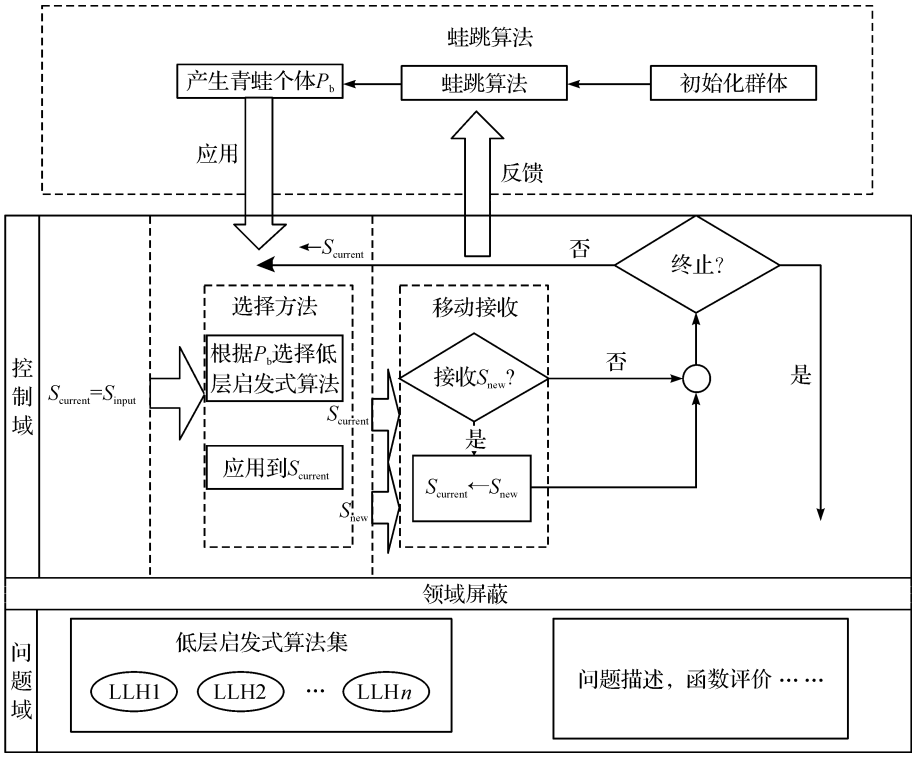


图 1 基于蛙跳算法的选择策略框架

Fig. 1 Framework of hyper-heuristic algorithm in leapfrog algorithm

2.1 算法流程

算法的具体流程为

步骤 1 初始化参数。确定蛙群的数量、种群以及每个种群的青蛙数。

步骤 2 随机产生初始蛙群，计算各个蛙的适应值。

步骤 3 划分种群。

步骤 3.1 随机找寻一个解，计算其他解与它的相似度，按相似度划分种群。

步骤 3.2 重复步骤 3.1，直到种群划分结束。

步骤 4 根据 SFLA 算法公式，在每个族群中进行元进化。

步骤 4.1 对全局最优和局部最优进行随机扰动。

步骤 4.2 对操作后的解计算适应度，如果有更优解出现，则进行更新。

步骤 4.3 重复步骤 4.1 和步骤 4.2，直至达到预设循环次数。

步骤 5 将各个族群进行混合。在每个族群都进行过一轮元进化之后，将各个族群中的蛙重新进行排序和族群划分并记录全局最好解 P_x。

步骤 5.1 利用更新策略对局部最差解进行替换。

步骤 5.2 如果达到迭代次数则停止，转步骤 6；如果没有，则转步骤 3。

步骤 6 输出全局最优解。

算法流程图如图 2 所示。

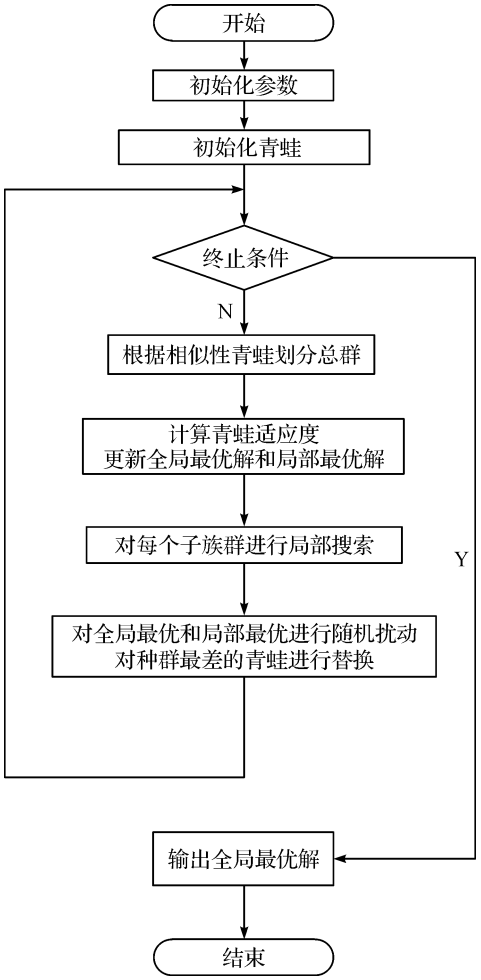


图 2　蛙跳算法流程图

Fig. 2　Flow chart of leapfrog algorithm

2.2　编码解码和算法参数

由于笔者所提及的蛙跳算法是作为选取算子的高层策略，所以搜索到的空间不是车辆路径的解空间，而是算子的组合空间。如图 3 所示，这是一个青蛙个体，是由一串数字组成，每一个数字对应着各自的底层算法，例如数字 1 就代表着 2-opt 算子，即对解执行 2-opt 算子。一个解从最原始的状态进过一个青蛙个体之后，就会产生一个新的解，新的解的目标函数就是该青蛙个体的适应值。与其他优化算法一样，SFLA 也具有一些必要的计算参数：F 为蛙群的数量；m 为族群的数量；n 为族群中青蛙的数量；S_{max} 为最大允许跳动步长；P_x 为全局最好解；P_b 为局部最好解；P_w 为局部最差解；q 为子族群中蛙的数量；LS 为局部元进化次数；SF 为全局思想交流次数等。

3	5	3	6	1	1	4	…	6	8	9	1

图 3　青蛙个体示意图

Fig. 3　Diagram of frog individual

2.3　更新策略

笔者更新策略采用文献[12]提出的方法，具体为对比每个种群内局部最优和局部次优解，如果相同位置上有相同的算子编号，则对该位置进行记录。推测该位置上的算子编号很可能是有利于增加适应度的，即有利于调度的，所以将种群内最差解位置上的算子编号替换为记录的算子编号。该操作如图 4 所示，一共为 4 串数字，也就是 4 个个体，第一行为全局最优解 P_x，第二行为子群最优解 P_b，第三行是子群最差解 P_w，第四行是更新后的 P_w。

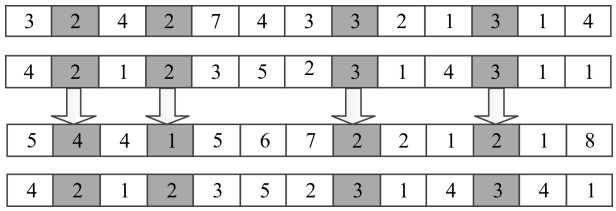


图 4　更新策略示意图

Fig. 4　Diagram strategy of update strategy

2.4　改进的种群相似度计算方法

算法开始时随机初始化总数量的青蛙，即随机生成总数量的具有固定长度的一组数字，每个数字都代表一种底层启发式算子。在常见的 SFLA 算法中[13-16]，青蛙按适应度进行排序，以特定的划分原则进行种群划分。文献[12]提出了一种新的个体相似度定义，其算法是对比两个个体每一位上的数字，如果相同，则相似度加 1。如图 5 所示，图 5 中两个解的相似度为 3。

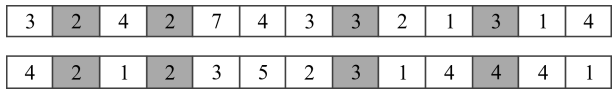


图 5　相似度计算示意图

Fig. 5　Diagram of similarity computation

相比于传统的个体相似度计算方式，笔者提出了基于最长公共子序列的相似度算法，将该相似度算法应用在超启发式算法的上层策略中是本研究的主要创新点。所谓的最长公共子序列（LCS：longest-common-subsequence problem）是一个在一个序列集合中（通常为两个序列）用来查找所有序列中最长子序列的问题。一个数列，如果分别是两个或多个已知数列的子序列，且是所有符合此条件序列中最长的，则称为已知序列的最长公共子序列。例如以下两串编码“123456”和“234567”，根据文献[12]的相似度计算，它们的相似度为 0，然而事实上它们其实十分相似，只是前者比后者一开始先调用了 1 次“1”算子，后者比前者最后多调用了 1 次“7”算子。根据最长公共子序列的相似度计算，前者有

子串“23456”和后者的子串“23456”是完全一样的，所以它们的相似度为 5，可以明显地看到采用公共最长子序列进行相似度计算能更好地还原两只青蛙的相似性。

因 LCS 问题的具有最优子结构性质[17]，最长公共子序列计算公式为

$$C[i,j]=\begin{cases}0 & \text{当 } i=0 \text{ 或 } j=0\\ C[i-1,j-1]+1 & \text{当 } i,j>0 \text{ 且 } x_i=y_j\\ \max(C[i,j-1],C[i-1,j]) & \text{当 } i,j>0 \text{ 且 } x_i\neq y_j\end{cases}$$

根据上述的递归公式和初值，有如下伪代码和实现：

输入 两个青蛙个体

输出 两个个体的相似度

步骤 1 function LCS(x, y, i, j)

步骤 2 if ($x[i]=y[j]$)

步骤 3 then $c[i, j]$=LCS(x, y, $i-1$, $j-1$)+1

步骤 4 else $c[i, j]$=max{LCS(x, y, i, $j-1$),LCS(x, y, $i-1$, j)}

步骤 5 return $c[i, j]$

因此采用最长公共子序列的算法对图 5 的两个数组进行相似度计算，结果如图 6 所示。最长公共子序列为“4232314”，所以两只青蛙的相似度为 7。

3	2	4	2	7	4	3	3	2	1	3	1	4

4	2	1	2	3	5	2	3	1	4	2	4	1

图 6 最长公共子序列相似度计算示意图

Fig. 6 Diagram of similarity computation base on longest common subsequence

根据新的个体相似度计算方法，进行种群划分：

步骤 1 随机找到一个解，计算该解与其他解的相似度，按相似度从高到低选取 n 个解，划分成一个种群，其中 n=青蛙总数(F)/种群个数(m)。

步骤 2 在剩余解中随机找寻一个解，利用步骤 1 的方法进行种群划分，直至种群划分结束。

3 数值仿真与分析

3.1 实验设计和参数分析

为了验证算法的实用性，对基准样例进行仿真实验，包括了 5 个配送中心，20 个客户点，其中每个配送中心的坐标、库存量已知，每个客户点的坐标、需求量已知，配送中心的车辆载重已知，样例详细信息见网址[18]。基于蛙跳算法上层选择策略的超启发式算法，采用 Cplusplus 编程，运行在 CPU 为 Intel Core i7，4.0 GHz，RAM 为 8 G 内存的计算机平台上，考虑到蛙跳算法很多参数可能会对性能造成影响，根据文献[12]的结论，采用如下具体参数：青蛙总数 Population＝200，种群数量 Group＝20，个体长度 Individual＝20，迭代次数 Maxloop＝20。因采用新的更新策略，所以蛙跳算法中的最大允许跳动步长($S_{\max}$)等参数将不存在。

笔者提出的核心创新点就是 SFLA 算法中的相似度计算的改进，由原来的个体每一位相比较改为基于最长公共子序列的相似度计算。为了验证笔者提出的相似度计算方法的有效性，设计了两个对比算法。

1) SFLA：最基本的蛙跳算法，相似度即个体的适应度。划分总群即按相似度从高到低依次分配到每个子群。

2) SFBE(SFLA base each dimension)：由 2.4 节所描述的根据每个个体每一位上是否相等来计算相似度。

3) SFBL(SFLA base longest common subsequence)：笔者提出的基于最长公共子序列的相似度计算。

3.2 结果分析

首先对 SFLA 算法、SFBE 算法和 SFBL 算法在解决低碳 LRP 模型的性能进行比较，通过收敛代数和目标函数收敛情况这两个方面进行分析。

3.2.1 收敛代数分析

3 种算法对 LRP 模型求解的收敛代数见表 2，取运行 10 次的平均收敛代数，这里迭代次数不设置为 20，而是采用连续 5 次迭代不出现更优秀个体作为终止迭代条件。

表 2 SFLA，SFBE 和 SFBL 的收敛代数

Table 2 Convergence times of SFLA，SFBE and SFBL

运行数/次	SFLA	SFBE	SFBL
1	109.0	85.0	60.0
2	102.0	76.0	64.0
3	94.0	74.0	69.0
4	92.0	78.0	63.0
5	96.0	66.0	58.0
6	88.0	80.0	73.0
7	104.0	79.0	61.0
8	91.0	74.0	66.0
9	99.0	68.0	60.0
10	93.0	71.0	70.0
平均值	96.8	75.1	64.4
方差	43.3	33.2	24.7

由表 2 可以看出：SFLA 算法平均迭代次数最多，为 96.8；其次是 SFBE 算法，平均迭代次数为 75.1；最后是 SFBL 算法，平均迭代次数最少为 64.4。说明 SFBL 算法收敛速度最快，效率更高，在求解更大规模的样例和模型时具有更大优势；同时，从迭代次数的方差来看，SFBL 的方差最小为 24.7，说明 SFBL 算法相比于另外两个算法更稳定。

3.2.2　目标函数分析

为了更清楚地比较 3 个算法在求解 LRP 模型上的最终效果，把 3 个算法目标函数的碳排放随着迭代次数的曲线图放在一起比较，见图 7。由图 7 可知：SFBL 算法的末端值最低，碳排放为 165.353 kg，因此 SFBL 算法最终的解质量最优，说明 SFBL 算法更有效，能够避免早熟现象；从 3 个算法的曲线来看，前期 SFBL 算法下降趋势最明显，其次是 SFBE 算法，最后是 SFLA 算法，说明了 SFBL 收敛速度最快。

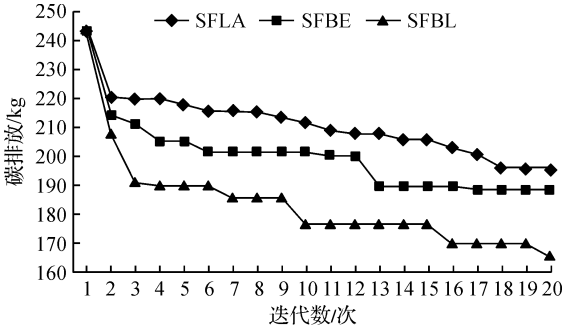
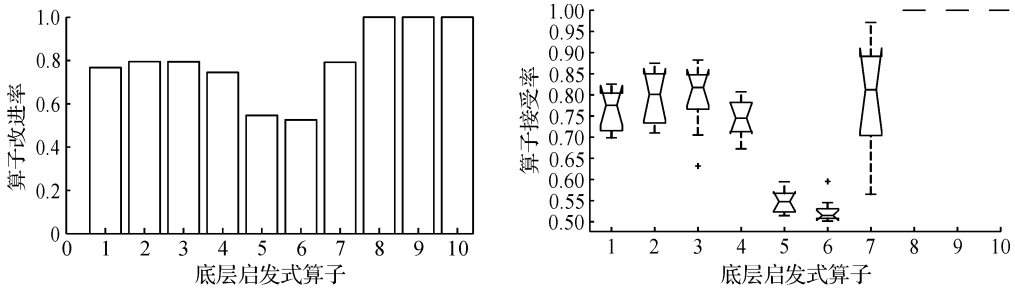


图 7　目标函数收敛情况比较

Fig. 7　Comparison of objective function convergence

为了解给予蛙跳算法中底层启发式算子对解质量的影响情况，运行 10 次，统计 10 个底层启发式算子的平均改进率，即该算子对解的质量是否起到提升效果，如图 8 所示。



(a) 平均接受率柱状图　　(b) 运行 10 次算子接受率统计图

图 8　算子的平均接受率

Fig. 8　Average acceptance rate of LLH

由图 8 可以看出：10 个底层启发式算子的平均改进率都超过了 0.5，其中局部搜索算子 LLH8，LLH9，LLH10 的改进率为 1，这是由算子本身的性质所决定的，因为算子本身只接受改进解，其余算子从高到低分别为 LLH2，LLH3，LLH7，LLH1，LLH4，LLH5，LLH6，相对而言，由于 LLH5 和 LLH6 是跟配送中心有关的算子，因此改进率比较低。总体而言，10 个底层启发式算子都有大概率能改进解的质量，证明了算子构建的有效性。然后，对于低碳 LRP 问题，3 种算法给出的最优解分别为：SFLA 算法 195.331 kg，SFBE 算法 188.476 kg，SFBL 算法 165.353 kg。这 3 个解对应的路线如表 3 所示。

表 3　最优路线

Table 3　Optimal routes

算法	配送路线
SFLA	D1－13－1－3－10－D1，D5－19－8－12－20－D5，D1－16－2－5－6－D1，D1－15－14－11－17－D1，D1－4－7－9－18－D1
SFBE	D1－9－13－10－3－D1，D2－1－8－11－20－D2，D3－5－16－19－6－D3，D1－15－14－12－17－D1，D1－7－4－18－2－D1
SFBL	D1－13－2－10－3－D1，D1－12－1－18－20－D1，D5－8－5－6－19－D5，D1－15－14－17－11－D1，D3－7－4－9－16－D3

最后，为了验证 3 个算法的通用性，将 3 个算法分别用在 3 种不同规模的 12 个测试样例上进行实验，分别为 20 个客户点，50 个客户点和 100 个客户点，配送中心均为 5 个，每个规模 4 个测试用例，目标函数改为所有车辆行驶距离的总和。

3 个方案，12 个样例的结果如表 4 所示，加粗表示为该样例在 3 个方案中最好的结果。从运行时间方面来讲，因为是毫秒级，3 个算法的复杂度一样，所以运行时间差距不大，总体而言 SFBL 方案运行时间最少。从目标函数而言，SFBL 方案所获得的解质量最高，其次是 SFBE，最后是 SFLA。总体而言，问题规模越大，SFBL 算法能获得最优解的概率就越高。

表 4 SFLA,SFBE 和 SFBL 对比

Table 4 Comparison of SFLA, SFBE and SFBL

样例名称	运行时间/ms			目标函数/km		
	SFBL	SFBE	SFLA	SFBL	SFBE	SFLA
coord20－5－1. dat	43 785	44 010	43 328	355. 683	348. 234	355. 683
coord20－5－1b. dat	53 111	53 754	50 565	157. 245	157. 245	157. 245
coord20－5－2. dat	44 818	43 621	49 935	352. 455	354. 877	352. 455
coord20－5－2b. dat	54 256	52 661	55 953	165. 612	165. 612	165. 612
coord50－5－1. dat	57 553	59 173	67 485	1 335. 700	1 367. 310	1 439. 030
coord50－5－1b. dat	91 011	91 710	106 914	1 127. 770	1 156. 760	1 156. 760
coord50－5－2. dat	57 828	57 508	65 581	1 340. 290	1 340. 290	1 340. 290
coord50－5－2b. dat	82 762	85 989	93 710	1 000. 610	1 012. 230	1 012. 230
coord100－5－1. dat	82 304	85 433	98 403	3 317. 170	3 401. 940	3 401. 940
coord100－5－1b. dat	122 624	126 667	137 791	2 636. 210	2 577. 410	2 636. 210
coord100－5－2. dat	92 729	89 153	103 326	2 612. 770	2 631. 710	2 625. 300
coord100－5－2b. dat	129 556	127 796	149 458	2 093. 790	2 103. 540	2 108. 750
平均值	76 028. 080	76 456. 250	85 204. 080	1 374. 609	1 384. 763	1 395. 959

由此可见:无论是从算法的收敛代数、收敛速度、稳定性和通用性,还是从最优解来看,基于最长公共子序列的相似度计算方式(SFBL 算法)都比其他两个算法要好。

4 结 论

针对低碳 LRP 模型,构建了和问题相关的底层启发式算子,使用蛙跳算法作为选取底层启发算子的高层策略,并提出了基于最长公共子序列的青蛙个体相似度计算方式,最后,将笔者提出的相似度算法与之前的相似度算法还有最原始的蛙跳算法比较,进行仿真实验。实验结果证明了算子构建的有效性,也表明基于最长公共子序列的相似度计算方法(SFBL 算法)在求解低碳 LRP 能够在获得更好解的同时,具有更快的收敛性和稳定性,对跨问题领域求解效果也比较好。

参考文献:

[1] 曹剑东,李家文. 基于顺路原则及位置预估的动态调度策略[J]. 浙江工业大学学报,2015,43(2):197-201.

[2] 赵燕伟,李文,张景玲,等. 多车型同时取送货问题的低碳路径研究[J]. 浙江工业大学学报,2015,43(1):18-23.

[3] 江贺. 超启发式算法:跨领域的问题求解模式[J]. 中国计算机学会通讯,2011,7(3):63-70.

[4] BURKE E K, HYDE M, KENDALL G, et al. A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics[J]. IEEE transactions on evolutionary computation, 2010, 14(6):942-958.

[5] NAREYEK A, SMITH S F, OHLER C M. Integrating local-search advice into refinement search (or not)[C]//Proceedings of the CP 2003 Third International Workshop on Cooperative Solvers in Constraint Programming. Pittsburgh:Carnegie Mellon University,2003:29-43.

[6] BURKE E K, KENDALL G, SOUBEIGA E. A tabu-search hyperheuristic for timetabling and rostering[J]. Journal of heuristics, 2003, 9(6):451-470.

[7] CHEN P C, KENDALL G, BERGHE G V. An ant based hyper-heuristic for the travelling tournament problem[C]//IEEE Symposium on Computational Intelligence in Scheduling. Hawaii:IEEE, 2007:19-26.

[8] MARSHALL R J, JOHNSTON M, ZHANG M. Developing a hyper-heuristic using grammatical evolution and the capacitated vehicle routing problem[C]//Asia-Pacific Conference on Simulated Evolution and Learning. New York:Springer International Publishing, 2014:668-679.

[9] 张春苗,赵燕伟,张景玲,等. 低碳定位——车辆路径问题[J]. 计算机集成制造系统,2017,23(12):2768-2777.

[10] WALKER J D. Design of vehicle routing problem domains for a hyper-heuristic framework[D]. Nottingham:University of Nottingham, 2015.

[11] EUSUFF M M, LANSEY K E. Optimization of water distribution network design using the shuffled frog leaping algorithm[J]. Journal of water resources planning & management, 2003, 129(3):210-225.

[12] 贾凌云,李冬妮,田云娜. 基于混合蛙跳和遗传规划的跨单元调度方法[J]. 自动化学报,2015,41(5):936-948.

[13] EBRAHIMI J, HOSSEINIAN S H, GHAREHPETIAN G B. Unit commitment problem solution using shuffled frog leaping algorithm[J]. IEEE transactions on power systems, 2011, 26(2):573-581.

[14] 陈勇,盛家君,王亚良,等. 基于改进蛙跳算法的可重构装配线调度研究[J]. 浙江工业大学学报,2014,43(3):274-279.

[15] 崔文华,刘晓冰,王伟,等. 混合蛙跳算法研究综述[J]. 控制与决策,2012,27(4):481-486.

[16] 骆剑平,李霞,陈泯融. 基于改进混合蛙跳算法的 CVRP 求解[J]. 电子与信息学报,2011,33(2):429-434.

[17] GNG. 算法导论——最长公共子序列(动态规划)LCS[EB/OL]. [2016-12-19]. https://blog. csdn. net/so_geili/article/details/53737001.

[18] Institut charles Delaunay. Classical Instances for LRP[EB/OL]. [2010-02-04]. http://prodhonc. free. fr/Instances/instances_us. htm.

(责任编辑:朱小惠)