文章编号:1671 - 6833(2015)03 - 0120 - 05

求解 VRPSDP 的变邻域混合遗传算法

马 欢[1], 张建伟[1], 赵进超[2], 陈 明[1]

(1. 郑州轻工业学院 软件学院,郑州 河南 450002; 2. 郑州轻工业学院 计算机与通信工程学院,郑州 河南 450002)

摘 要: 针对卸装一体化车辆路径问题,提出一种结合变邻域下降搜索和遗传算法的混合启发式算法(GA_VND). 利用随机生成的初始种群,通过遗传算法的交叉变异操作生成弱可行解种群,选择其中的最优值作为变邻域深度搜索的初始解. 在变邻域深度搜索的过程中通过两种不同的局部搜索算子对解进行局部搜索和迭代优化. 通过对 54 个算例的求解,仿真结果表明 GA_VND 更新了 54 个已知最好解中的 8 个,表明了该算法是解决卸装一体化车辆路径问题的一种有效方法.

关键词: 卸装一体化; 车辆路径问题; 变邻域下降搜索; 遗传算法; 组合优化

中图分类号: TP301 **文献标志码**: A doi:10.3969/j.issn.1671 - 6833.2015.03.026

0 引言

卸装一体化车辆路径问题(Vehicle Routing Problem with Simultaneous Delivery and Pickup, VRPSDP)普遍存在于物流运输中,该问题的解决对于减少物流成本,提高物流效率具有重要意义. 由于 VRPSDP 同 VRP 问题都属于 NP 难题[1],因此目前国内外对于 VRPSDP 的研究主要集中在各种元启发式算法和混合式的启发算法上,在元启发式算法方面:国内外的专家学者多采用遗传算法[2]、蚁群算法[3]、粒子群算法[4]、模拟退火算法[5]等来解决这一问题. 在混合启发式算法上,苏孟洛等[6]提出结合粒子群算法和变邻域下降搜索的混合粒子群算法;Y. M 等[7]提出结合禁忌搜索和蚁群搜索算法的混合算法;Zachariadis 等[8]提出导引式局部搜索和禁忌搜索相结合的混合算法. 这些算法虽然都取得了一定成果,但其求解质量仍有一定的改进空间.

将遗传算法和变邻域下降算法相结合,提出一种混合启发式算法 GA_VND 用于求解 VRPSDP 问题. 利用 54 个基准测试算例进行实验,并与文献[8]以及文献[9]中的算法求解结果比较,验证本文算法的有效性.

1 问题描述

1.1 问题描述

定义设有向带权图 $G=(V,A,C)$,其中 $V=\{i \mid i=0,1,\cdots,n\}$ 是节点集合,节点 0 为出发地节点,$(1\sim n)$为客户地节点;$A=\{i,j \mid i,j\in V\}$ 表示弧集,$\boldsymbol{C}=\{c_{ij} \mid (i,j)\in A\}$ 为权重矩阵,c_{ij}表示节点 i 到节点 j 的距离. 每个客户节点 i 都有卸货需求 d_i 和装货需求 p_i,运输车辆的最大载货量为 Q. VRPSDP 则可以描述为:①每辆车都从 0 点出发,服务若干客户后返回 0 点,形成一个解 S;②每个客户都仅被服务一次,而且只能由某一车辆提供服务;③每个车辆的载重量都不超过 Q;④所有客户的装卸货需求都不超过 Q;⑤总的运输距离 $f(S)$最小.

1.2 解的可行性定义

对于 VRPSDP 的解 $S=\{r_j \mid j=1,2,\cdots,k\}$,其中 $r_j=\{i \mid i\in[1,n]\}$,表示车辆 j 的路径,k 为解中最大的车辆数. S 是强可行解的充要条件:车辆在访问过 r_j 上所有客户后载重都不超过 Q. 对于解 S,$\exists r_i\in S$ 不满足约束条件,则解 S 是一个弱可行解;$\forall r_i\in S$ 不满足约束条件,则解 S 是一个不可行解.

收稿日期:2015 - 01 - 19;**修订日期**:2015 - 03 - 04

基金项目:国家自然科学基金资助项目(61403349);国家级大学生创新创业训练计划项目(201310462018)

作者简介:马欢(1981 -),男,河南孟州人,郑州轻工业学院讲师,主要研究方向为信息处理,智能与信息系统,E - mail:mahuanresearch@126.com.

2　GA_VND 算法

2.1　遗传算法搜索全局生成强可行解的步骤

STEP1:最近邻法生成遗传算法初始强可行解 S_0. 步骤为:①从未被选择的客户节点集合中选择距离出发节点最近的节点,开始一条新路径;②从未被选择的客户节点集合选择距离路径上最后一个节点最近的节点;③若加入节点后路径满足强可行解约束条件,则返回第 2 步,否则返回第 1 步.

STEP2:将 STEP1 生成的强可行解作为父代染色体,通过交叉生成多个子代染色体,即构造了多个可行解. 步骤如下:①将该强可行解存入子代种群中;②从父代染色体上随机选取客户节点作为交叉节点;③进行染色体交叉和变异运算生成新的染色体. 随机选择交换和插入两种方式进行交叉标点.

$$V_1 = \{j \mid c(i,j) < \alpha\} \tag{1}$$

$$V_2 = \{j \mid \ |c(i,0) - c(j,0)| < \alpha\} \tag{2}$$

上述公式需满足 $i \neq j, i, j \in V$ 和 $i, j \neq 0$. $\alpha = f(S_0)/n$, S_0 为 STEP1 生成的强可行解. 变异操作采用贪婪倒位变异的方式:随机选择变异点 i,在不包含节点 i 以及节点 i 的左右两个节点 *i_left* 和 *i_right* 的集合中选择距离节点 i 最近的节点 j,将 *i_right* 和 j 之间的节点逆向排列. 例如:解为(1,2,3,4,5,6,7,8,9),随机选择节点 4,距离节点 4 最近的节点为 8,倒位变异解后为(1,2,3,4,8,7,6,5,9). 通过交叉和变异生成的染色体不一定满足强可行解的约束条件,因此需要将其进行转化为强可行解. ④判断当前生成的染色体总数是否到达种群上限,是则判断新种群中是否有优于父代染色体的子代染色体,有则选择其中所有优于父代染色体的子代染色体进入 STEP3,否则返回 STEP2,并扩大距离参数 α.

STEP3:利用 Chen J. F.[10] 的方法将解集中的弱可行解转化为强可行解. 方法如下:扫描解中的每一个客户节点,如果满足强可行解的约束条件,则继续扫描,否则记录该节点,直到节点全部扫描完. 扫描完成后将上次扫描中被记录的节点从路径中删除后再按照逆序重新加入路径,即可得到一个强可行解. 选择转换后的强可行解集中的最优解作为变邻域下降搜索的初始解.

2.2　变邻域下降搜索局部最优解

变邻域下降搜索将从 2.1 中获得的强可行解作为初始解,随机选择一种邻域结构进行局部搜索,当到达迭代次数上限或者连续无改进迭代次数到达上限时结束搜索. 这里选择两种邻域结构:插入(insert)和 2 - opt. 插入是将解 S 中的客户 i 从当前位置 l_i 移动到另一个位置 l_j,从而产生一个新解. 位置 l_i 和 l_j 上的节点属于解 S 中的任意路径. 2-opt 是对于解 S 中的同一路径上位于 l_i 和 l_j($l_i < l_j$)上的两个客户,将 $l_i + 1$ 位置上的节点与位于 l_j 上的节点交换位置,并将 $l_i + 1$ 到 l_j 之间的节点按逆序排列. 两种结构如图 1 所示:

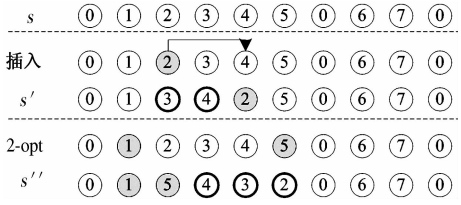

图 1　邻域结构:插入和 2-opt

Fig. 1　Neighborhood structure: insert and 2-opt

两种邻域结构的随机选择可以结合两种邻域不同的寻优能力,同时也可以在一定程度上扩展算法的搜索空间.

2.3　GA_VND 算法

首先需要定义相关参数:遗传算法染色体种群数 *max_chom*,最大迭代次数 *max_i*,交叉概率 *pc*,变异概率 *pm*,GA 连续无改进迭代次数 *nobetter_ga_i* = *max_i*/4,VND 连续无改进迭代次数 *nobetter_vnd_k* = $n/2$. GA_VND 算法伪码如下:

1. 初始化参数:*max_chom*, *max_i*, *pc*, *pm*, *nobetter_ga_i*, *nobetter_vnd_k*;

2. 定义变量:迭代次数 $i = 0$;GA 当前无改进迭代次数 $j = 0$; VND 当前无改进迭代次数 $k = 0$;

3. WHILE ($i <$ *max_i*) DO

4. 按照 2.1 节 STEP1 构造初始强可行解 S_i;

5. $S_{best} = S_i$; $\alpha = f(S_0)/n$;

6. WHILE ($j <$ *nobetter_ga_i*) DO

7. $S_0 = S_{best}$; //S_0 为 VND 的初始解

8. 以 S_0 为父代染色体按照 2.1 中的 STEP2 和 STEP3 方法生成新种群;

9. 取得新种群中的最优染色体 S_0^{new}

10. WHILE($k <$ *nobetter_vnd_k*) DO

11. 对于解 S_0^{new} 随机选择一种邻域结构进行局部搜索,得到解 S_k;

12. IF ($f(S_k) < f(S_0^{new})$)

13. $S_0^{new} = S_k$;

14. $k = 0$;

15. ELSE

16. $k++$;
17. End IF
18. End WHILE//输出局部最优解 S_0^{new}
19. IF ($f(S_0^{new}) < f(S_{best})$)
20. $S_{best} = S_0^{new}$;
21. $j = 0$;
22. ELSE
23. $j++$;
24. $\alpha = \alpha \times 1.1$;
25. End IF
26. End WHILE
27. $i++$;
28. End WHILE
29. 输出 S_{best},算法结束;

3 实验结果及分析

3.1 测试用例

为了测试 GA_VND 算法的性能,主要采用两种类型的测试数据进行实验:随机取数的 Dethloff 算例[11]、Salhi 和 Nagy 算例[12]. Dethloff 算例为随机生成的 40 个问题规模均为 50 的算例. 根据客户分布和车辆需求可以分为 SCA3x, SCA8x, CON3x, CON8x. 其中 SCA 算例为[0,100]间均匀分布,CON 算例为[0,200]间非均匀分布. 各客户点的卸货需求 d 为在 [0,100]间取随机值,装货需求 p 为$(k+0.5)*d$,其中 k 在[0,1]间取随机值. Salhi 和 Nagy 算例的问题规模在[50,199]之间,每个问题包含两个算例,其中 CMTxX 为在原 CMTx 算例基础上增加了装卸货需求的算例[10], CMTxY 为和 CMTxX 装卸货需求相反的算例.

GA_VND 算法采用 c#在. net 平台下实现,运行环境为 WIN7 64x,CPU 为 Intel core i5 - 2520M 2.5 GHz,内存为 6G,参数设置为 $max\_chom = 100$;$max\_i = 200$;$pc = 0.5$; $pm = 0.05$; $nobetter\_ga\_i = 50$;Dethloff 算例中 $nobetter\_vnd\_k = 25$. Salhi 和 Nagy 算例中 $nobetter\_vnd\_k = n/2$, CMTxX 和 CMTxY 的问题规模相同,分别为(50,75,100,120,150,199).

3.2 算法的可行性分析

以 Dethloff 算例中的 SCA3 - 0 算例为例,设定最大迭代次数为 200,对 GA_VND 算法和 VND 算法每次迭代求得的当前最优解进行比较,重复运行 10 次取平均值,结果如图 2 所示.

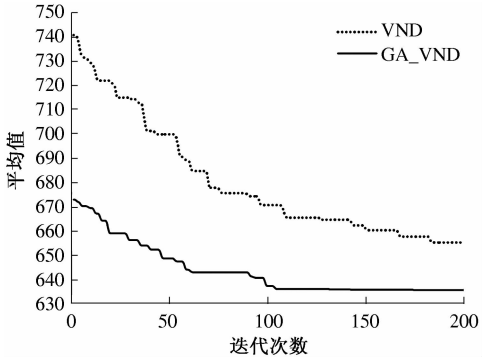

图 2 SCA3 - 0 算例上每次迭代最优解的平均值比较

Fig. 2 Comparison of the optimal solution's average value by every iterated based on SCA3 - 0 examples

可以看出 GA_VND 算法中由 GA 算法求得的初始可行解的存在,使得 GA_VND 算法可以从一个较优的初始解开始搜索. 陷入局部最优时扩展 GA 交叉范围的机制和 VND 仅变换邻域相比,GA_VND 算法可通过变化初始解来重新求解,跳出当前局部最优,避免了 VND 算法随机初始解求解的局限性.

3.3 Dethloff 算例上的最优解比较

为了验证 GA_VND 算法的有效性,这里将遗传算法、Zachariadis 提出的禁忌搜索和 Guided Local Search 的混合算法 TS_GLS[8]、GA_VND 算法应用于 Dethloff 算例进行计算. 表 1 描述了在 Dethloff 算例上 GA 求得的最优解、TS_GLS 求得的最优解和 GA_VND 运行 10 次取平均值的比较,其中,L 为最优路径长度,t 为计算消耗时间,单位为 s.

表 1 Dethloff 算例上的结果比较

Tab. 1 Comparison of the solutions based on Dethloff

算例	GA		TS_GLS		GA_VND	
	L	t_a/s	L	t_b/s	L	t_a/s
SCA3 - 0	646.41	0.77	636.06	2.83	635.62	1.05
SCA3 - 1	704.38	0.85	**697.84**	2.12	**697.84**	0.99
SCA3 - 2	681.18	0.88	**659.34**	2.58	**659.34**	1.01
SCA3 - 3	691.32	1.01	**680.04**	3.13	**680.04**	1.12
SCA3 - 4	718.95	0.91	**690.50**	2.68	**690.50**	1.03
SCA3 - 5	671.23	0.85	**659.90**	2.56	**659.90**	1.09
SCA3 - 6	656.12	0.78	**651.09**	4.40	**651.09**	0.95
SCA3 - 7	666.87	0.86	**659.17**	2.98	**659.17**	1.03
SCA3 - 8	746.42	0.91	**719.47**	3.98	**719.47**	1.13
SCA3 - 9	699.16	0.79	**681.00**	3.86	**681.00**	1.09
SCA8 - 0	969.93	0.94	**961.50**	3.21	**961.50**	1.15
SCA8 - 1	1055.23	1.28	**1 050.20**	3.55	**1 049.65**	1.42
SCA8 - 2	1051.13	1.73	**1 044.48**	4.67	**1 044.48**	1.85
SCA8 - 3	1010.96	1.21	**983.34**	3.29	**983.34**	1.39

续表 1

算例	GA		TS_GLS		GA_VND	
	L	t_a/s	L	t_b/s	L	t_a/s
SCA8 – 4	1092.01	1.69	**1 065.49**	2.68	**1 065.49**	1.87
SCA8 – 5	1043.52	1.86	**1 027.08**	4.50	**1027.08**	2.15
SCA8 – 6	975.08	1.82	**971.82**	2.67	**971.82**	2.08
SCA8 – 7	1075.81	1.62	**1 052.17**	4.32	**1 051.28**	1.91
SCA8 – 8	1079.84	1.59	**1 071.18**	3.43	**1 071.18**	1.78
SCA8 – 9	1086.48	1.24	**1 060.50**	4.12	**1 060.50**	1.35
CON3 – 0	628.66	1.90	**616.52**	3.89	**616.52**	2.01
CON3 – 1	575.39	1.64	**554.47**	2.97	**554.47**	1.81
CON3 – 2	541.92	1.53	**518.00**	3.32	**518.00**	1.75
CON3 – 3	597.02	1.87	**591.19**	2.78	**591.19**	2.11
CON3 – 4	609.62	2.10	**588.79**	3.12	**588.79**	2.39
CON3 – 5	574.24	1.82	**563.70**	3.45	**563.70**	1.95
CON3 – 6	511.88	1.55	**499.05**	2.98	**499.05**	1.81
CON3 – 7	592.10	2.01	**576.48**	2.40	**576.48**	2.26
CON3 – 8	538.62	0.86	**523.05**	5.02	**523.05**	1.15
CON3 – 9	600.08	1.73	**578.25**	3.14	**578.25**	1.90
CON8 – 0	866.32	1.34	**857.40**	3.40	**857.40**	1.51
CON8 – 1	755.87	1.94	**740.85**	3.73	**740.85**	2.09
CON8 – 2	742.64	1.16	**712.89**	2.87	**712.89**	1.38
CON8 – 3	826.00	1.52	**811.07**	3.82	**811.07**	1.81
CON8 – 4	780.03	1.77	**772.25**	2.98	**772.25**	1.90
CON8 – 5	761.07	1.62	**754.95**	5.76	**754.95**	1.85
CON8 – 6	703.98	2.15	**678.92**	4.00	**678.92**	2.30
CON8 – 7	841.63	1.35	**811.96**	2.46	**811.96**	1.59
CON8 – 8	778.27	1.11	**767.53**	4.21	**767.53**	1.21
CON8 – 9	835.27	1.38	**809.00**	3.87	**809.00**	1.49

注:t_a 表示处理器为 i5 微机,主频为 2.5 GHz 上的运行时间;t_b 表示处理器为 Pentium IV 微机,主频为2.4 GHz 上的运行时间.

通过对表 1 的 4 组 40 个测试算例进行计算,其中的已知最好解用粗体标出. 通过遗传算法的和 GA_VND 算法的结果比较可以看出:混合算法 GA_VND 的求解质量明显高于遗传算法的,通过 TS_GLS 算法和 GA_VND 算法的结果比较可以看出 GA_VND 算法更新了 Dethloff 算例中的 SCA3 – 0、SCA8 – 1 和 SCA8 – 7 在 TS_GLS 中的已知最好解,说明 GA_VND 算法是可行有效的.

3.4　Salhi 和 Nagy 算例上的最优解比较

为了进一步验证 GA_VND 算法的有效性,采用在 Salhi 和 Nagy 算例上将 GA_VND 算法运行 10 次取平均值和 TS_GLS 算法以及 C. Yu 提出的 AIS 算法[9]两种算法求得的最优解进行对比,结果如表 2 所示.

通过 TS_GLS 算法和 GA_VND 算法以及 AIS 算法计算的最优解结果相比较,可以看出:在平均路径长度方面,GA_VND 算法较 TS_GLS 算法减少了 0.46,较 AIS 算法减少了 18.73;在最优解的质量上,GA_VND 算法较 TS_GLS 算法最大提高了 2.55%,最小提高了 0.65%. 较 AIS 算法最大提高了 5.35%,最小提高了 0.27%,进一步说明了 GA_VND 算法是可行有效的.

表 2　Salhi 和 Nagy 算例上的结果比较

Tab. 2　Comparison of the solutions based on Salhi and Nagy

算例	TS_GLS		ALS		GA_VND	
	L	t_a/s	L	t	L	t_b/s
CMT1X	469.80	2.89	472	—	**466.77**	1.15
CMT1Y	469.80	3.85	472	—	**466.77**	0.99
CMT2X	684.21	7.42	**682**	—	684.21	4.85
CMT2Y	684.21	8.02	688	—	**666.75**	5.13
CMT3X	721.27	11.62	728	—	**715.51**	8.90
CMT3Y	**721.27**	13.53	728	—	**721.27**	9.85
CMT12X	**662.22**	11.80	664	—	**662.22**	9.15
CMT12Y	**662.22**	7.59	671	—	663.50	8.70
CMT11X	**838.06**	17.78	886	—	846.23	21.15
CMT11Y	837.08	14.26	877	—	**830.04**	25.87
CMT4X	**852.46**	27.75	883	—	**852.46**	40.31
CMT4Y	**852.46**	31.20	867	—	862.28	39.22
CMT5X	**1 030.55**	51.67	1 081	—	1 035.51	91.60
CMT5Y	**1 030.55**	58.81	1 073	—	1 036.14	90.85
平均值	751.15		769.42		750.69	

4　结论

卸装一体化车辆路径问题是一个有着实际应用需要且广泛存在的问题,针对这一问题设计了一种结合遗传算法和变邻域下降搜索的混合启发式算法,将遗传算法的全局搜索能力和变邻域下降搜索算法的局部搜索能力相结合,有效地提高了求解质量. 通过对 Dethloff 算例和 Salhi 和 Nagy 算例共 54 个算例的求解,验证了 GA_VND 算法优于单一的遗传算法,且更新了 54 个算例中的 8 个当前最优解,表明了本文算法的有效性.

参考文献:

[1]　叶春明, 王科峰, 李永林. 同时送取货车辆路径问题算法研究综述[J]. 计算机应用研究, 2013, 30(2): 334 – 340.

[2]　龙磊, 陈秋双, 华彦宁, 等. 具有同时集送货需求的车辆路径问题的粗粒度并行遗传算法[J]. 系统仿真学报, 2009, 21(7): 1962 – 1968.

[3]　GAJPAL Y, ABAD P. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery

and pickup[J]. Computers & Operations Research, 2009, 36(12): 3215 - 3223.

[4] 吴斌, 蔡红, 樊树海, 等. 双倍体差分进化粒子群算法在 VRPSDP 中的应用研究[J]. 系统工程理论与实践, 2010, 30(3): 520 - 526.

[5] 王超, 穆东. 物料配送和废旧产品回收的 VRPSDP 问题的并行模拟退火算法[J]. 北京交通大学学报, 2014, 38(6): 19 - 26.

[6] 苏孟洛, 杨宏安, 孙启峰. 求解卸装一体化的车辆路径问题的混合粒子群算法[J]. 中国制造业信息化:学术版, 2012, 41(9): 52 - 56.

[7] YOUSEFIKHOSHBAKHT M, DIDEHVAR F, RAHMATI F. A combination of modified tabu search and elite ant system to solve the vehicle routing problem with simultaneous pickup and delivery[J]. Journal of Industrial and Production Engineering, 2014, 31(2): 65 - 75.

[8] ZACHARIADIS E E, TARANTILIS C D, KIRANOUDIS C T. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service[J]. Expert Systems with applications, 2009, 36(2): 1070 - 1081.

[9] YU C, LAU H Y K. AIS-based Algorithm for Solving Vehicle Routing Problem with Simultaneous Pick-up and Delivery (VRP - SPD)[J]. Journal of Traffic and Logistics Engineering, 2013, 1(2): 174 - 178

[10] CHEN J F, WU T H. Vehicle routing problem with simultaneous deliveries and pickups[J]. Journal of the Operational Research Society, 2006, 57(5): 579 - 587.

[11] DETHLOFF J. Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick - up[J]. OR-Spektrum, 2001, 23(1): 79 - 96.

[12] SALHI S, NAGY G. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling[J]. Journal of the Operational Research Society, 1999, 50(10):1034 - 1042.

A Hybrid Genetic and Variable Neighborhood Descent Algorithm for Vehicle Routing Problem with Simultaneous Delivery and Pickup

MA Huan[1], ZHANG Jian-wei, ZHAO Jin-chao[2], CHEN Ming[1]

(1. Software Engineering College, Zhengzhou University of Light Industry, Zhengzhou 450002, China; 2. School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China)

Abstract: This paper proposes a hybrid heuristic algorithm combing variable neighborhood descent search with genetic algorithm (GA_VND) to solve vehicle routing problem with simultaneous delivery and pickup. By the use of the initial populations generated randomly, the weak feasible solutions are produced by the crossover and mutation operators of genetic algorithm. And then, the best of them was selected as initial solution of variable neighborhood descent algorithm. Finally, in the process of the variable neighborhood descent search, two different neighborhood structures are used to search the locally optimal solution. The simulation results show that GA_VND can update 8 better solutions in the 54 best known solutions, which illustrates that GA_VND is an effective method for vehicle routing problem with simultaneous delivery and pickup.

Key words: vehicle routing problem; simultaneous delivery and pickup; variable neighborhood descent; genetic algorithm; NP-hard