求解多旅行商问题的改进分组遗传算法

王勇臻* 陈 燕 于莹莹
(大连海事大学交通运输管理学院 大连 116026)

摘 要：该文针对总路径长度最小的多旅行商问题，提出一种改进分组遗传算法。在该算法中，设计了一种有序分组编码，采用新编码方式的个体与多旅行商问题有效解之间具有一一对应的关系。为了减少算法的运行时间，根据编码的特点构造了一种快速交叉算子。同时，结合贪婪算法和 2-opt 算法设计了一种新的局部搜索算子，以提高算法的收敛精度。实验结果分析表明，所提算法能够有效地解决多旅行商问题，具有可靠的全局收敛性，较高的计算效率。

关键词：分组遗传算法；多旅行商问题；编码；2-opt 算法
中图分类号：TP18 **文献标识码**：A **文章编号**：1009-5896(2017)01-0198-08
DOI: 10.11999/JEIT160211

Improved Grouping Genetic Algorithm for Solving Multiple Traveling Salesman Problem

WANG Yongzhen CHEN Yan YU Yingying
(*Transportation Management College*, *Dalian Maritime University*, *Dalian* 116026, *China*)

Abstract: In order to solve the total-path-shortest Multiple Traveling Salesman Problem (MTSP), an improved grouping genetic algorithm is proposed. This algorithm employs a new encoding scheme called ordered grouping encoding, which makes the adjusted individuals corresponding one by one to valid solutions of MTSP. According to the features of the encoding scheme, a fast crossover operator is constructed for the sake of reducing the running time of the algorithm. For enhancing its local search ability, the algorithm combines the greedy algorithm and the 2-opt algorithm to design a new local search operator. The comparison of results shows that the proposed algorithm can solve MTSP effectively and has an excellent search performance no matter in computing efficiency or convergence precision.

Key words: Grouping Genetic Algorithm (GGA); Multiple Traveling Salesman Problem (MTSP); Encoding; 2-opt algorithm

1 引言

多旅行商问题(Multiple Traveling Salesman Problem, MTSP)是对经典旅行商问题(TSP)的推广，即给定 n 个城市，m 个旅行商从同一(或不同)城市出发，分别走一条旅行路线，使得每个城市有且仅有一个旅行商经过(除出发城市)，同时花费最小[1,2]。相较于 TSP, MTSP 具有更广泛的工程背景，如车辆路径规划[3]、应急物资配送[4]、无人机覆盖搜索[5]和不合格品控制[6]等，已被证明属于 NP-hard 问题。所以，如何快速、有效地解决 MTSP 具有很高的实际应用价值。

MTSP 本质上属于分组问题，其求解过程包含了城市的分组优化以及组内遍历次序优化两个环节[7]。因此，相应的启发式算法大多计算复杂，并且过于依赖问题本身的特征，容易发生早熟收敛[1,8]。近年来，人们从生物机理中得到启发，提出了多种进化算法应用于 MTSP，如遗传算法[9,10]、蚁群算法[11,12]、蜂群算法[13]和杂草入侵算法[14]。而编码方式作为进化算法的核心，直接影响着 MTSP 的求解性能，常见的有单染色体、双染色体和组合染色体 3 种编码[4,9]。然而，这些编码方式将两个优化环节融合在一起，不但使得进化算子通常难以设计，而且算法运行过程中容易产生大量的冗余解[9,10]。随着研究的深入，文献[9]提出了一种分组遗传算法(GGA-SS)用于求解 MTSP，其基本思想是

收稿日期：2016-03-07；改回日期：2016-07-22；网络出版：2016-10-09
*通信作者：王勇臻 kuadmu@163.com
基金项目：国家科技支撑计划(2014BAH24F04)，国家自然科学基金(71271034)
Foundation Items: The National Key Technology Research and Development Program of the Ministry of Science and Technology of China (2014BAH24F04), The National Natural Science Foundation of China (71271034)

先将各旅行商的路径编码为 m 个组，然后按照一定规则对组执行交叉、变异操作，计算结果优于早前算法。在此基础之上，文献[14]通过借鉴不同的生物智能，分别提出了 3 种群体进化算法(ABCFC, ABCVC 和 IWO)，并且在多数实例上得到了目前最优的计算结果。但是，上述算法仍然存在两类问题：解空间存在冗余，以及缺乏有效的进化算子。

本文提出了一种改进分组遗传算法(IGGA-SS)，设计了一种有序分组编码，将解空间进一步缩小了 $m!$ 倍，并基于该编码构造了一种快速交叉算子，以减少算法的运行时间。同时，结合贪婪算法和 2-opt 算法设计了一种新的局部搜索算子，以进一步提高算法的收敛精度。实验结果分析表明，所提算法可以有效地解决 MTSP，计算结果比目前同类算法给出的结果更优。

2　GGA-SS 求解 MTSP

2.1 GGA-SS 核心步骤

(1)编码： GGA-SS 将各旅行商的路径编码为 m 个组，且不考虑组的排列，其解空间大小为 $n!C_{n-1}^{m-1}$，如图 1 所示($n=12$，$m=3$)。

(2)交叉算子： GGA-SS 采用一种两阶段交叉算子：

阶段 1：从两个父代中随机选择一个，根据式(1)计算各组的 r_k，选择 r_k 值最大的组并复制给子代。

$$r_k = z_k / n_k \tag{1}$$

其中，z_k 表示第 k 组的路径长度，n_k 表示第 k 组包含的城市个数。然后将该组包含的城市从两个父代中删除，反复执行阶段 1 直到子代中包含 m 个组。

阶段 2：将阶段 1 中未分配的城市逐个插入到子代中，且保证每次插入都使得总路径长度增加最少。

(3)变异算子：将父代中每个城市以概率 p_{cp} 复制给子代，未分配的城市采用阶段 2 进行处理。

(4)互斥执行：每一次迭代产生一个随机数 $\text{rand}\in[0,1)$，若小于阈值 p_c 则执行交叉算子，否则执行变异算子。

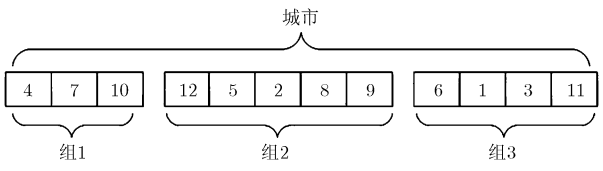


图 1 GGA-SS 编码示意图

(5)稳态保留：新生成的子代与当前种群进行比较，若唯一则替换当前最差个体，否则丢弃该子代。

2.2 GGA-SS 存在的问题

通过分析 GGA-SS 主要计算过程可知，其存在 3 个问题：

(1)编码不考虑组的排列，存在冗余。

(2)在阶段 1 反复计算父代中各组的 r_k，耗费了大量的时间。

(3)阶段 2 本质上是贪婪算法，虽然在一定程度上能够得到满意解，但是随着问题规模的增大容易陷入局部极值。

3　IGGA-SS 求解 MTSP

3.1 有序分组编码

由于 GGA-SS 编码不考虑组的排列，以图 1 为例，考虑如下编码方案：

{{4,7,10}, {6,1,3,11}, {12,5,2,8,9}},
{{4,7,10}, {12,5,2,8,9}, {6,1,3,11}},
{{6,1,3,11}, {4,7,10}, {12,5,2,8,9}},
{{6,1,3,11}, {12,5,2,8,9}, {4,7,10}},
{{12,5,2,8,9}, {4,7,10}, {6,1,3,11}},
{{12,5,2,8,9}, {6,1,3,11}, {4,7,10}}

对于 MTSP，组本身所代表的旅行商没有任何意义，以上 6 个编码表达了同一个 MTSP 有效解。为了避免这种冗余，本文提出一种有序分组编码。上述例子中，假设 3 个组{4,7,10}, {12,5,2,8,9}和{6,1,3,11}的路径长度分别是 100, 110 和 120，计算可得 $r_1=100/4=25$，$r_2=110/5=22$ 和 $r_3=120/5=24$。r_k 反映了一个组属于最优解的可能性，r_k 越小则其属于最优解的概率越大[9]。然后将 3 个组按照 r_k 进行升序排列，则{{12,5,2,8,9}, {6,1,3,11}, {4,7,10}}是有序分组编码后的唯一个体。

显然，有序分组编码的个体与 MTSP 有效解之间是一一对应的关系，其解空间大小是 $n!C_{n-1}^{m-1}/m!$，较 GGA-SS 编码缩小了 $m!$ 倍，有利于减小算法的复杂度，对大规模 MTSP 的求解具有重要意义。

3.2 快速交叉算子(阶段 1)

GGA-SS 在阶段 1 反复计算父代中各组的 r_k，导致算法耗时严重。针对该缺陷，受文献[15]启发，本文基于有序分组编码提出一种快速交叉算子。设两个父代 O_1=$\{P^{1,1},P^{2,1},\cdots,P^{m,1}\}$，$O_2$=$\{P^{1,2},P^{2,2},\cdots,P^{m,2}\}$，子代 $O_{\text{new}}=\{P^1,P^2,\cdots,P^m\}$，具体过程如下：

(1)令 $P^i=\varnothing$，$i=1,2,\cdots,m$，$j=1$；

(2)产生一个随机数 $\text{rand}_j\in[0,1)$，若 $\text{rand}_j<0.5$ 则 $P^j=P^{j,1}$，否则 $P^j=P^{j,2}$；

(3)将 P^j 中的城市从两个父代中删除，令 $j=j+1$，若 $j\le m$ 则转步骤(2)，否则终止。

图 2 是快速交叉算子示意图。可见，该算子并未反复计算父代中各组的 r_k，提高了算法的计算效率。同时，该算子可以有效地保留父代中的部分分组，而基于有序分组编码，使得更具潜力的组优先得到保留。对于该阶段未分配的城市，将通过 3.3 节的局部搜索算子(阶段 2)进行处理。

3.3 局部搜索算子(阶段 2)

为了克服贪婪算法容易早熟收敛的缺陷，本文通过结合 2-opt 算法来引入新的信息，以维持种群的多样性，提出一种新的局部搜索算子，具体过程如下：

首先产生随机数 $\text{rand}\in[0,1)$，若大于阈值 p_d 则使用贪婪算法；否则，将未分配的城市逐个随机插入到子代中任意组，然后使用 2-opt 算法[16](如图 3 所示)对各组路径进行优化。完成分配之后，计算各组的 r_k，并按升序进行排列。

该算子既结合贪婪算法和 2-opt 算法来提高收敛精度，又引入了随机性保证种群不早熟收敛，平衡了算法的开发性和探索性。

3.4 算法描述

在 GGA-SS 主要计算过程基础之上，下面给出 IGGA-SS 的求解步骤：

步骤 1　算法和问题参数初始化；

步骤 2　根据有序分组编码初始化种群 $P(t)$，令 $t=1$，$c=0$；

步骤 3　产生随机数 $\text{rand}\in[0,1)$，若 $\text{rand}<p_c$ 则转步骤 4，否则转步骤 5；

步骤 4　执行快速交叉算子，转步骤 6；

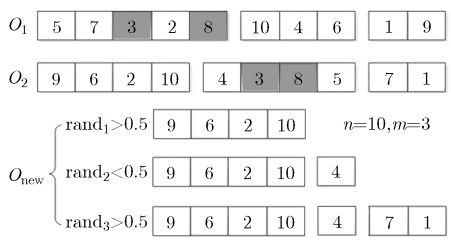


图 2 快速交叉算子示意图

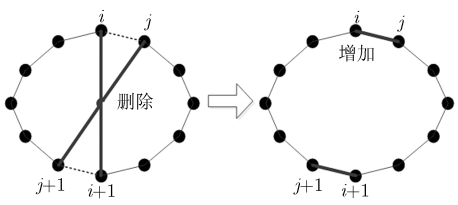


图 3 2-opt 算法示意图

步骤 5　执行变异算子；

步骤 6　执行局部搜索算子；

步骤 7　执行稳态保留，令 $c=c+1$，若 $c==P_{\text{size}}$ (种群规模)则令 $c=0$ 并转步骤 8，否则转步骤 3；

步骤 8　令 $t=t+1$，若 $t<\text{MaxGen}$ (最大迭代次数)则转步骤 3，否则终止算法。

3.5 时间复杂度分析

(1)贪婪算法的时间复杂度 $O(T_1)=O((n-n_{un})+(n-n_{un}+1)+\cdots+(n-1))$。其中，$n_{un}$ 表示未分配的城市个数，令 $n_{un}=\alpha\cdot n,\ \ 0<\alpha<1$，则 $O(T_1)=O(n_{un}\cdot(n-n_{un}+n-1)/2)=O((2\alpha-\alpha^2)\cdot n^2-\alpha\cdot n)$。

(2)2-opt 算法的时间复杂度 $O(T_2)=O(n_{un}+n_1^2+n_2^2+\cdots+n_m^2)$。其中，$n_1+n_2+\cdots+n_m=n$。最坏的情况下，有 1 个组包含 $n-(m-1)$ 个城市，其余 $m-1$ 个组各包含 1 个城市，$O(T_2)=O(\alpha\cdot n+(n-m+1)^2)$。

(3)计算各组的 r_k 并排序的时间复杂度 $O(T_3)=O(m\log m+n)$。

(4)局部搜索算子的时间复杂度 $O(T_4)=O((1-p_d)\cdot T_1+p_d\cdot T_2+T_3)$。

(5)选择父代的时间复杂度记作 $O(T_5)$。

(6)阶段 1+阶段 2 的时间复杂度 $O(T_6)=O(2T_5+n+T_4)$。

(7)变异算子的时间复杂度 $O(T_7)=O(T_5+n+T_4)\approx O(T_6)$。

(8)稳态保留的时间复杂度 $O(T_8)=O(P_{\text{size}}\cdot m)$。

综上所述，IGGA-SS 迭代一次的时间复杂度为

$$O\left(P_{\text{size}}\cdot\left(p_c\cdot T_6+(1-p_c)\cdot T_7+T_8\right)\right)\approx O\left(P_{\text{size}}\cdot(T_7+T_8)\right)$$

4 实验结果与分析

本文采用 Java 语言编写程序实现算法，并在一台配置为 Inter(R) Core(TM) i7-3770 CPU @3.40 GHz 的 PC 上运行程序。参考文献[9,14]的做法，使用 TSPLIB 中距离对称的实例(设置不同的旅行商数目)进行数值实验。

为了便于控制选择压力，本文采用等级选择方法选取父代[17]，如式(2)，式(3)所示。其中，PS(i)表示种群中第 i 个个体被选中的概率，SP 表示选择压力，$1\le\text{SP}\le 2$，SP 越大则选中最优个体的概率越大。

$$P(i)=\frac{1}{MS}\left(\text{SP}-2\times(\text{SP}-1)\times\frac{i-1}{MS-1}\right) \tag{2}$$

$$\text{PS}(i)=\sum_{j<=i}P(j),\quad i=1,2,\cdots,MS \tag{3}$$

4.1 参数设置及其影响

IGGA-SS 包含 4 个主要参数：SP，p_c，p_{cp}和p_d。从 TSPLIB 中选取 kroA100 实例($m=5$)采用实验设计方法(Design Of Experiment，DOE)[18]研究参数对算法性能的影响。各参数均取 4 个水平，如表 1 所示，说明如下：

(1)文献[9]中p_c取 0.8，即更倾向于执行交叉算子，而文献[14]提出的 3 种算法却完全摒弃了交叉算子，折中考虑，设置p_c取 0.6~0.9，间隔为 0.1。

(2)文献[9,14]中p_{cp}取值差异较大，但是总体上取 0.65~1.00，本文p_{cp}取 0.75~0.90，间隔为 0.05。

(3)选择压力 SP 从小到大取 1.2~1.8，间隔为 0.2；同样，偏好阈值p_d取 0.35~0.65，间隔为 0.10。

算法在每种参数组合下独立运行 20 次，设置 Psize 为 100，MaxGen 为 1000, 20 次独立运行所得平均性能 AVG 为评价指标。选择规模为 $L_{16}(4^4)$的正交实验表，正交表和所得 AVG 如表 2 所示。根据正交表，计算各参数的极差和重要程度，如表 3 所示。进而绘制各参数对算法性能的影响趋势，如图 4 所示。

表 1 参数水平

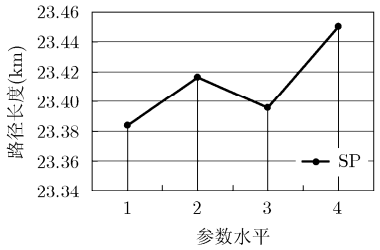
参数	水平1	水平2	水平3	水平4
SP	1.2	1.4	1.6	1.8
p_c	0.6	0.7	0.8	0.9
p_{cp}	0.75	0.8	0.85	0.9
p_d	0.35	0.45	0.55	0.65

表 2 正交表和AVG统计(m)

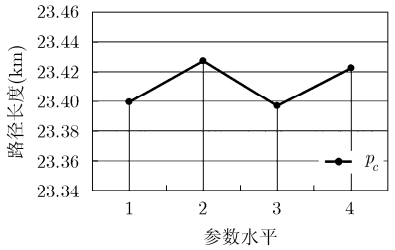
No	SP	p_c	p_{cp}	p_d	AVG
1	1	1	1	1	23340.69
2	1	2	2	2	23450.00
3	1	3	3	3	23417.98
4	1	4	4	4	23327.05
5	2	1	2	3	23381.95
6	2	2	1	4	23506.34
7	2	3	4	1	23349.79
8	2	4	3	2	23428.10
9	3	1	3	4	23417.27
10	3	2	4	3	23410.60
11	3	3	1	2	23360.12
12	3	4	2	1	23394.59
13	4	1	4	2	23457.01
14	4	2	3	1	23342.49
15	4	3	2	4	23459.94
16	4	4	1	3	23540.87

表 3 各参数响应值(m)

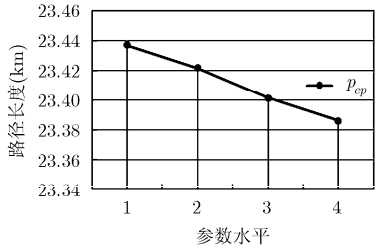
水平	SP	p_c	p_{cp}	p_d
1	23383.93	23399.23	23437.01	23356.89
2	23416.54	23427.36	23421.62	23423.81
3	23395.65	23396.96	23401.46	23437.85
4	23450.08	23422.65	23386.11	23427.65
极差	66.15	30.40	50.89	80.96
等级	2	4	3	1



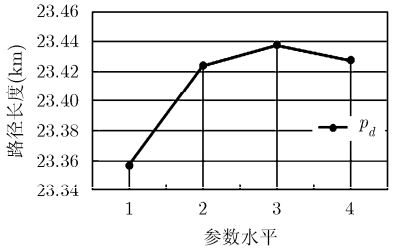
(a) SP对算法性能影响的水平趋势



(b) p_c 对算法性能影响的水平趋势



(c) p_{cp} 对算法性能影响的水平趋势



(d) p_d 对算法性能影响的水平趋势

图 4 各参数对算法性能影响的水平趋势

综合表 3 和图 4 可知：参数 p_d 的极差最大，表明 p_d 对算法性能影响最大，并且 p_d 取值不宜太大，否则容易导致算法收敛缓慢。影响程度其次的是选择压力 SP，较小的 SP 有利于充分使用整个种群所包含的信息。紧接着是 p_{cp}，并且随着其取值的增大，算法性能单调递增。而 p_c 对算法性能的影响最小。

基于上述分析，兼顾算法寻优质量和效率，设置参数如下：$\mathrm{SP}=1.2$，$p_c=0.8$，$p_{cp}=0.90$，$p_d=0.35$。

4.2 算法性能分析

为了验证算法改进的有效性，选取 kroA100 实例并设置 m 分别为 3, 5 和 10，采用 IGGA-SS, GGA-SS 和 IGGA-SS2 进行数值实验，其中 IGGA-SS2 在阶段 2 中使用贪婪算法。

设定 Psize 为 100, MaxGen 为 1000，其余参数取自 4.1 节，3 种算法在每组实验上独立运行 20 次，结果如表 4 所示。可知，IGGA-SS 在全部 3 组实验上均获得了 3 种算法中最好的结果；耗时方面，IGGA-SS2 最少，IGGA-SS 次之。进而绘制 m 对 3 种算法收敛精度、耗时的影响，如图 5，图 6 所示。

表 4 3种算法20次独立运行结果统计

问题	算法	最短路径(m)	平均路径(m)	平均耗时(s)
n=100 m=3	IGGA-SS	**22098.61**	**22359.35**	7.80
	IGGA-SS2	22122.20	23318.31	**1.66**
	GGA-SS	23528.79	25061.29	6.84
n=100 m=5	IGGA-SS	**23139.98**	**23357.31**	7.54
	IGGA-SS2	23242.08	24996.45	**1.77**
	GGA-SS	24479.01	26534.20	7.82
n=100 m=10	IGGA-SS	**27137.11**	**27444.91**	6.91
	IGGA-SS2	27727.41	30537.00	**2.10**
	GGA-SS	28552.83	29798.42	9.67

综合表 4，图 5 和图 6 分析，可得出如下结论：

(1)对比 GGA-SS 和 IGGA-SS2 可知，快速交叉算子可以大幅度减少算法的耗时，计算可得 IGGA-SS2 相较于 GGA-SS 耗时减少了 75.73%～78.28%，并且随着 m 的增大其差距单调递增。但是，由于该算子不重新计算父代中各组的 r_k，将会影响各组保留的优先次序，随着 m 的增大其收敛精度下降比较明显。

进一步地，由 3.5 节可知：交叉算子执行过程中，两者除了首次计算 r_k 的耗时相同之外，GGA-SS 反复计算 r_k 的时间复杂度为

$$O(m_2\lg m_2+n_2+m_3\lg m_3+n_3+\cdots+m_m\lg m_m+n_m)$$

其中，$m_k\le m$，$n_k\le n$，$k=2,3,\cdots,m$。易知，随着 m，n，或者 p_c，亦或 Psize，MaxGen 的增大，这部分耗时累积将单调递增。在本文参数设置下，两者耗时差距已达到 75%以上。

(2)对比 IGGA-SS 和 IGGA-SS2 可知，局部搜索算子可以进一步改善算法的收敛精度，计算可得 IGGA-SS 相较于 IGGA-SS2 收敛精度提高了 4.11%～10.13%，并且随着 m 的增大其差距单调递增，同时，两者耗时差距则单调递减。

(3)对比 IGGA-SS 和 GGA-SS 可知，本文所做改进可以有效地提高算法的收敛精度、减少算法的耗时。计算可得 IGGA-SS 相较于 GGA-SS 收敛精度提高了 7.90%～11.97%。同时，随着 m 的增大 IGGA-SS 耗时单调递减，除了 $m=3$ 之外(14.04%)，IGGA-SS 耗时均少于 GGA-SS(3.58%和 28.54%)。

4.3 与知名算法的对比分析

为了更好地验证 IGGA-SS 求解 MTSP 的性能，引进目前最优秀算法：两种蜂群算法 ABCFC, ABCVC 和杂草入侵算法 IWO 作为比较对象[14]，需要说明的是，这 3 种算法都是 GGA-SS 的变体。选取 eil101, ch150($m=3,5,10$)和 kroB200($m=3,5,10,20$)进行数值实验。设定 Psize 为 100, MaxGen 为 1000，其余参数取自 4.1 节，所对比算法参数取自文献[14], 4 种算法在每组实验上独立运行 20 次，结果如表 5 所示。可知，IGGA-SS 在全部 10 组实验上均获得了 4 种算法中最好的结果，并且耗时最少。

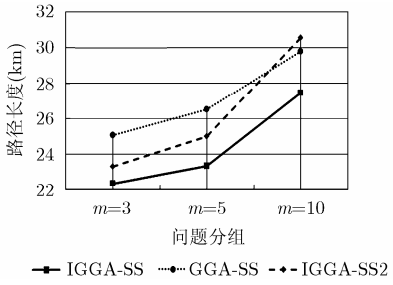


图 5 m 对 3 种算法收敛精度的影响

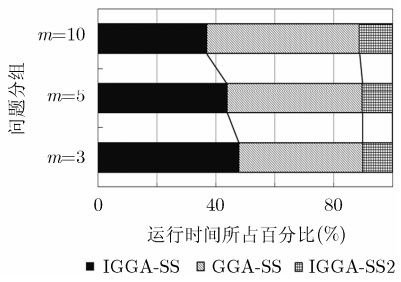


图 6 m 对 3 种算法耗时的影响

表 5　4种算法20次独立运行结果统计

算法	问题	最短路径(m)	平均路径(m)	平均耗时(s)	问题	最短路径(m)	平均路径(m)	平均耗时(s)
IGGA-SS		**665.3037**	**674.4504**	**8.25**		**6610.34**	**6801.01**	**17.18**
ABCFC	n=101	698.4237	726.3776	22.66	n=150	7349.17	7663.54	47.06
ABCVC	m=3	694.0849	722.7571	20.38	m=3	7273.57	7714.09	43.41
IWO		665.5774	678.4148	40.89		6881.91	7073.49	96.68
IGGA-SS		**687.43**	**700.00**	**7.68**		**6680.29**	**6855.11**	**16.73**
ABCFC	n=101	749.98	776.26	22.02	n=150	8096.99	8308.51	46.26
ABCVC	m=5	746.54	767.27	20.05	m=5	7767.54	8229.51	42.36
IWO		692.94	715.89	40.96		7073.02	7270.23	92.36
IGGA-SS		**783.48**	**795.45**	**6.72**		**7360.80**	**7654.32**	**15.37**
ABCFC	n=101	905.51	944.09	22.91	n=150	9565.58	10088.52	47.13
ABCVC	m=10	877.91	916.72	20.94	m=10	8707.89	9839.96	44.01
IWO		848.46	858.06	43.43		9036.57	9267.15	98.91
IGGA-SS		**30061.64**	**31151.47**	**30.87**		**31024.77**	**31755.84**	**29.72**
ABCFC	n=200	33060.08	34279.17	83.47	n=200	34776.23	36126.51	77.74
ABCVC	m=3	33176.58	34126.91	71.88	m=5	35225.53	36829.47	71.29
IWO		31945.02	32653.52	150.51		32926.41	33913.12	149.92
IGGA-SS		**33644.94**	**34608.00**	**27.47**		**42400.17**	**43277.96**	**25.09**
ABCFC	n=200	40821.31	43099.15	77.18	n=200	56743.36	60207.58	82.32
ABCVC	m=10	40324.05	42321.22	70.78	m=20	50691.85	54118.86	76.86
IWO		36959.99	41861.45	149.49		53690.68	55996.97	166.53

为了检验 4 种算法计算结果的差异在统计上是否显著，本文进行了单因素方差分析[19]。表 6 是 4 种算法的计算差异性对比，其中 <,=,> 分别表示行所代表算法计算结果劣于、无区别和优于列所代表算法。从表 6 可知，IGGA-SS 在全部 10 组实验上计算结果均优于 ABCFC 和 ABCVC，在 9 组实验上计算结果优于 IWO，整体性能最优；IWO 整体性能仅次于 IGGA-SS；而 ABCFC 与 ABCVC 整体性能相近。

从表 5 中 4 种算法耗时对比可见，IGGA-SS 的耗时远少于其余 3 种算法，计算可得 IGGA-SS 相较于 ABCFC, ABCVC 与 IWO 耗时分别减少了 61.77%~70.67%, 57.05%~67.91%和 79.49%~84.93%。由 3.5 节与文献[14]可知，4 种算法迭代一次的时间复杂度均为 $O($ Psize $\cdot$(变异算子+子代保留))，且 MaxGen 均取 1000。同时，O(变异算子) 的最高次幂都是 n 的 2 次幂，且 O(变异算子) $\gg O$(子代保留)，因此耗时差距主要源于 Psize 取值不同。对比可知，IGGA-SS 中 Psize 取 100，而 ABCFC, ABCVC 和 IWO 分别取 250, 250 和 300，远大于 IGGA-SS。这也说明，在种群规模较小的情况下，IGGA-SS 依然可以得到比其余 3 种算法更高的收敛精度。

图 7 是 4 种算法收敛曲线对比。可知，IGGA-SS 的收敛速度最快，几乎是垂直收敛，并且迭代过程中始终处于其余 3 种算法的下方。ABCFC 在迭代前期收敛速度较快，然而容易早熟收敛。IWO 和 ABCVC 虽然没有陷入局部极值，但是收敛速度与 IGGA-SS 存在明显差距。

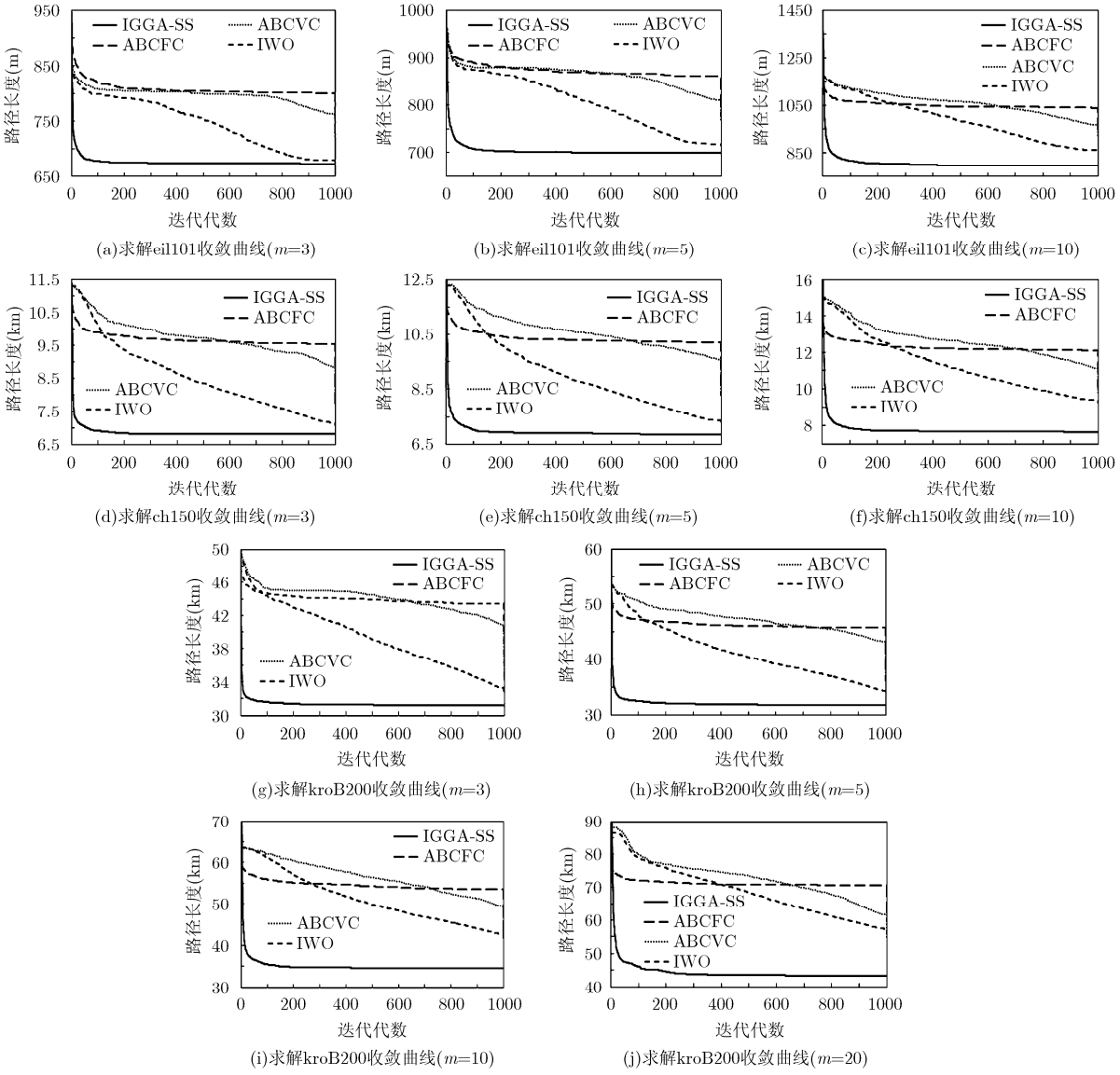
表 6　4 种算法的计算差异性对比

算法	IGGA-SS			ABCFC			ABCVC			IWO		
	<	=	>	<	=	>	<	=	>	<	=	>
IGGA-SS	–	–	–	0	0	10	0	0	10	0	1	9
ABCFC	10	0	0	–	–	–	2	7	1	8	2	0
ABCVC	10	0	0	1	7	2	–	–	–	6	4	0
IWO	9	1	0	0	2	8	0	4	6	–	–	–

综上所述，IGGA-SS 在收敛速度、精度以及耗时方面均优于所对比的 3 种算法。

5 结束语

本文提出了一种改进分组遗传算法，用于求解总路径长度最小的 MTSP。实验结果分析表明，IGGA-SS 具有比最新用于解决该问题的 ABCFC, ABCVC 和 IWO 更优的性能。今后工作仍需进行更多的数值实验和对算法的效率作进一步改进，并将所提算法应用于解决港口自动调度这类大规模 MTSP。



(a)求解eil101收敛曲线(m=3) (b)求解eil101收敛曲线(m=5) (c)求解eil101收敛曲线(m=10)

(d)求解ch150收敛曲线(m=3) (e)求解ch150收敛曲线(m=5) (f)求解ch150收敛曲线(m=10)

(g)求解kroB200收敛曲线(m=3) (h)求解kroB200收敛曲线(m=5)

(i)求解kroB200收敛曲线(m=10) (j)求解kroB200收敛曲线(m=20)

图 7 4 种算法收敛曲线对比

参 考 文 献

[1] SOYLU B. A general variable neighborhood search heuristic for multiple traveling salesmen problem[J]. *Computers & Industrial Engineering*, 2015, 90(11): 390–401. doi: 10.1016/j.cie.2015.10.010.

[2] KOTA L and JARMAI K. Mathematical modeling of multiple tour multiple traveling salesman problem using evolutionary programming[J]. *Applied Mathematical Modelling*, 2015, 39(12): 3410–3433. doi: 10.1016/j.apm.2014.11.043.

[3] 谢秉磊, 李颖, 刘敏. 带临时补充点的融雪剂撒布车辆路径问题[J]. 系统工程理论与实践, 2014, 34(6): 1593–1598. doi: 10.12011/1000-6788(2014)6-1593.

XIE Binglei, LI Ying, and LIU Min. Vehicle routing problem with temporary supplementary points for spreading deicing salt[J]. *Systems Engineering-Theory & Pratice*, 2014, 34(6): 1593–1598. doi: 10.12011/1000-6788(2014)6-1593.

[4] 刘明, 张培勇. 求解多旅行商问题的新混合遗传算法: 以应

急物资配送为例[J]. 系统管理学报, 2014, 23(2): 247–254.
LIU Ming and ZHANG Peiyong. New hybrid genetic algorithm for solving the multiple traveling salesman problem: An example of distribution of emergence materials[J]. *Journal of System & Management*, 2014, 23(2): 247–254.

[5] ANN S, KIM Y, and AHN J. Area allocation algorithm for multiple UAVs area coverage based on clustering and graph method[J]. *IFAC-PapersOnLine*, 2015, 48(9): 204–209. doi: 10.1016/j.ifacol.2015.08.084.

[6] KIRALY A, CHRISTIDOU M, CHOVAN T, *et al*. Minimization of off-grade production in multi-site multi-product plants by solving multiple traveling salesman problem[J]. *Journal of Cleaner Production*, 2016, 111: 253–261. doi: 10.1016/j.jclepro.2015.05.036.

[7] KASHAN A H, AKBARI A A, and OSTADI B. Grouping evolution strategies: an effective approach for grouping problems[J]. *Applied Mathematical Modelling*, 2015, 39(9): 2703–2720. doi: 10.1016/j.apm.2014.11.001.

[8] BEKTAS T. The multiple traveling salesman problem: An overview of formulations and solution procedures[J]. *Omega*, 2006, 34(3): 209–219. doi: 10.1016/j.omega.2004.10.004.

[9] SINGH A and BAGHEL A S. A new grouping genetic algorithm approach to the multiple traveling salesperson problem[J]. *Soft Computing*, 2009, 13(1): 95–101. doi: 10.1007/s00500-008-0312-1.

[10] YUAN S, SKINNER B, HUANG S, *et al*. A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms[J]. *European Journal of Operational Research*, 2013, 228(1): 72–82. doi: 10.1016/j.ejor.2013.01.043.

[11] LIU W, LI S, ZHAO F, *et al*. An ant colony optimization algorithm for the multiple traveling salesmen problem[C]. Proceedings of IEEE Conference on Industrial Electronics and Applications, ICIEA, Xi'an, China, 2009: 1533–1537. doi: 10.1109/ ICIEA.2009.5138451.

[12] NECULA R, BREABAN M, and RASCHIP M. Performance Evaluation of Ant Colony Systems for the Single-depot Multiple Traveling Salesman Problem[M]. Springer International Publishing, 2015: 257–268. doi: 10.1007/ 978-3-319-19644-2_22.

[13] XUE M, WANG T, and MAO S. Double evolutsional artificial bee colony algorithm for multiple traveling salesman problem[C]. Preoeedings of MATEC Web of Conferences, EDP Sciences, 2016: 44. doi: 10.1051/ matecconf/20164402025.

[14] VENKATESH P and SINGH A. Two metaheuristic approaches for the multiple traveling salesperson problem[J]. *Applied Soft Computing*, 2015, 26: 74–89. doi: 10.1016/ j.asoc.2014.09.029.

[15] 韩丽霞, 王宇平, 兰绍江. 基于有序划分编码的图着色算法[J]. 电子学报, 2010, 38(1): 146–150.
HAN Lixia, WANG Yuping, and LAN Shaojiang. Graph coloring algorithm based on ordered partition encoding[J]. *Acta Electronica Sinica*, 2010, 38(1): 146–150.

[16] HELSGAUN K. General k-opt submoves for the Lin–Kernighan TSP heuristic[J]. *Mathematical Programming Computation*, 2009, 1(2/3): 119–163. doi: 10.1007/s12532-009-0004-6.

[17] ALIJLA B O, WONG L P, LIM C P, *et al*. A modified intelligent water drops algorithm and its application to optimization problems[J]. *Expert Systems with Applications*, 2014, 41: 6555–6569. doi: 10.1016/j.eswa.2014.05.010.

[18] WANG S, WANG L, LIU M, *et al*. An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem[J]. *International Journal of Production Economics*, 2013, 145(1): 387–396. doi: 10.1016/j.ijpe.2013.05.004.

[19] 王军强, 郭银洲, 崔福东, 等. 基于多样性增强的自适应遗传算法的开放式车间调度优化[J]. 计算机集成制造系统, 2014, 20(10): 2479–2493. doi: 10.13196/j.cims201410016.
WANG Junqiang, GUO Yinzhou, CUI Fudong, *et al*. Diversity enhancement-based adaptive genetic algorithm for open-shop scheduling problem[J]. *Computer Integrated Manufacturing Systems*, 2014, 20(10): 2479–2493. doi: 10.13196/j.cims201410016.

王勇臻：　男，1990 年生，博士生，研究方向为智能计算、数据挖掘等.

陈　燕：　女，1952 年生，教授，研究方向为管理科学与决策、知识管理、数据仓库与数据挖掘等.

于莹莹：　女，1987 年生，博士生，研究方向为数据挖掘、信息检索等.