5 改进交叉算子求解 TSP 问题的研究

5.1 引言

用遗传算法求解 TSP 问题的关键就是交叉算子的设计。通过大量的实践发现，一个好的算子往往能够使群体快速的向最优方向靠拢，反之，则不能达到比较好的效果。虽然从理论上讲，通过采取一定的方法，譬如，选取初始种群数量足够大，迭代的代数接近无穷大时一定能获得最优解，但实际操作显然是不可能的。因此，对遗传算法求解 TSP 问题的交叉算子进行研究和改进，快速地找到近似最优解，无论在学术上还是在应用上都具有十分重要的意义。

5.2 常用交叉算子分析

本文的研究主要针对基于路径表示的交叉算子。人们常用的基于路径表示的交叉算子主要有：部分匹配交叉(PMX)、顺序交叉(OX) 、循环交叉(CX) 、插入交叉(IX) 、两交换启发式(HGA)等算子。

部分匹配交叉(PMX)、顺序交叉(OX) 、循环交叉(CX)这三种算子是最基本的交叉算子，它们使用起来比较简单，而且不会产生非法的路径。但是它们没有充分考虑 TSP 问题自身所具有的特点，忽视了父代中边的邻接状况，使得运算效果不理想。许多的交叉算子都是在它们的基础上设计出来的。例如：陈国良等在文献[22]中采用的类 OX 算子，梁艳春等在文献[23]中设计的单点交叉映射算子和单点顺序交叉算子。这些算子虽然做了一些改进的工作，但效果仍不十分明显。

插入交叉(IX)和两交换启发式(HGA)算子虽然对邻接状况进行了考虑，在交叉的过程中每一步的结果都是在母体的范围内使用贪婪策略进行选择，得到的序列在母体的范围是一种局部最优点列，但是，最后一个城市都没有选择的余地，也就没有考虑染色体最后一个基因和第一个基因之间的距离，同时它们对邻接状况的评价也具有一定的局限性，因此难以保证其选择的优越性。

因此，为克服上述算子的局限性，通过进一步研究，本文提出了两种能够有效利用局部信息，并且能很好的继承父代优秀基因的交叉算子。通过实例仿真，取得了令人满意的效果。

5.3 顺序插入交叉算子

5.3.1 顺序插入交叉算子设计

假设有 n 个城市 $1,2,\cdots n,$ 染色体编码定义为推销员所经过的城市的顺序号，如

染色体 $X:(x_1,x_2,\cdots x_n)$ ，将上述染色体看成一个环，染色体中 x_n 的下一个城市为 x_1。

定义 1：设有三个城市 a,b,c，定义三角距离差函数 $G(a,b,c)$ 如下:

$$G(a,b,c)=d(a,b)+d(b,c)-d(a,c)$$

其中 $d(x,y)$ 表示城市 x 到城市 y 的距离。

在遗传算法中，本文构造适应度函数为 $z=(Length(X))^{-1}$，其中 $Length(X)$ 表示染色体 X 所代表的回路的长度。

顺序插入交叉算子的基本设计思想：对于两个待交叉的染色体，将一个作为基本染色体，另一个作为参考染色体，按照在参考染色体中城市间的邻接顺序，利用三角距离差函数作为评价标准，运用贪婪策略思想，逐步改变基本染色体的编码顺序，然后将基本染色体与参考染色体互相交换角色，改变参考染色体的编码顺序，从而使经过交叉得到的两个子代染色体的适应度提高。

设待交叉的双亲： A: $(a_1,a_2,\cdots a_n)$， B: $(b_1,b_2,\cdots b_n)$，设计顺序交叉算子如下：

① 首先，我们将父体 A 作为基本染色体，将父体 B 作为参考染色体，在 B 中随机找到一个参考城市 b_i $(i=1,2,\cdots n)$；

② 在染色体 A 中找到城市 $a_j=b_i$， $a_m=b_{i+1}$，然后对函数值 $G(a_j,b_{i+1},a_{j+1})$ 与 $G(a_{m-1},a_m,a_{m+1})$ 进行比较，如果 $G(a_j,b_{i+1},a_{j+1})\geq G(a_{m-1},a_m,a_{m+1})$，则染色体 A 保持不变，否则，在染色体 A 中删除城市 a_m，然后在 a_j 与 a_{j+1} 之间插入 b_{i+1}，得到新的染色体 A'；

③ 依次在参考染色体 B 中取参考城市 $b_{i+1},\cdots,b_n,b_1,\cdots b_{i-1}$，重复步骤②，直至最终生成子代个体 A_1；

④ 再将父体 B 作为基本染色体，将父体 A 作为参考染色体，采用上述步骤①②③得到子代个体 B_1。

需要指出的是，顺序插入交叉算子是在参考城市一定的情况下，对基本体整体按照局部最优进行插入的。

事实上，我们在基本体 A 中插入参考体 B 中某一个城市 b_{i+1} 时，此时基本体 A 中的三条边：

$$a_ja_{j+1},\quad a_{m-1}a_m,\quad a_ma_{m+1}$$

将被替换为另三边

$$a_jb_{i+1},\quad b_{i+1}a_{j+1},\quad a_{m-1}a_{m+1},$$

因为

$$G(a_j,b_{i+1},a_{j+1})<G(a_{m-1},a_m,a_{m+1}),$$

即

$$d(a_j,b_{i+1})+d(b_{i+1},a_{j+1})+d(a_{m-1},a_{m+1})<d(a_j,a_{j+1})+d(a_{m-1},a_m)+d(a_m,a_{m+1})$$

所以

当插入城市b_{i+1}之后得到基本体A'的回路长度将变小，从而经过交叉操作之后，所得到的后代将不会劣于其各自的基本体。

同时，由于本文的交叉算子充分考虑了父代染色体中城市的邻接状况，从而能使子代个体很好地继承父代优秀的基因。

以$n=8$的 TSP 问题为例，假设城市间的距离用下面的矩阵表示：

$$\begin{pmatrix} 0 & 5 & 12 & 26 & 24 & 20 & 8 & 24 \\ 5 & 0 & 24 & 21 & 20 & 21 & 8 & 26 \\ 22 & 24 & 0 & 34 & 36 & 5 & 30 & 5 \\ 26 & 21 & 34 & 0 & 6 & 30 & 26 & 38 \\ 14 & 20 & 36 & 6 & 0 & 32 & 23 & 40 \\ 20 & 21 & 5 & 10 & 32 & 0 & 27 & 9 \\ 8 & 8 & 30 & 26 & 23 & 27 & 0 & 31 \\ 24 & 26 & 5 & 38 & 10 & 9 & 31 & 0 \end{pmatrix}$$

设待交叉的双亲：

A: 3 7 1 5 4 2 6 8

B: 1 3 8 4 7 5 6 2

经过本文提出的顺序插入交叉算子运算，得到两个子代个体：

A_1: 5 4 7 2 1 3 8 6

B_1: 3 8 2 7 1 5 4 6

可以计算得：

$Length(A)=156, Length(B)=115, Length(A_1)=72, Length(B_1)=111$。

显然，子代个体A_1, B_1都比它们的父代染色体所表示的路径长度小，并且从中我们看出它们很好地继承了父代优秀的基因。

5.3.2 实例仿真

为了验证本文提出的顺序插入交叉（OIC）算子，我们将本文提出的交叉算子同时与插入交叉（IX）算子和两交换启发式（HGA）交叉算子进行比较，并设计了相应的算法5.1、算法5.2和算法5.3，其中三个算法的选择算子均采用轮转法，变异算子采用对换变异，算法5.1为插入交叉算子，算法5.2为两交换启发式交叉算子，算法5.3为顺序插入交叉算子。通过比较求解标准的30、50和75个城市的TSP问题，得到了表5.1所示的结果。部分遗传参数分别为：交叉概率为0.8；变异概率为0.2；

种群个数为60。程序分别运行10次。

由表5.1可以看出顺序插入交叉算子在各种性能上都超过了插入交叉算子和两交换启发式交叉算子，并且在进化代数较少的情况下，能得到比较好的计算结果。图5.1、图5.2、图5.3分别是算法5.1、5.2、5.3所求得最优回路。

表5.1 三种算法求解TSP问题的结果
Table5.1 The results of 3 algorithms for TSP

算法	城市个数	进化代数	最优回路长度	最差回路长度	平均回路长度	标准差
5.1	30	200	830.7028	983.8409	926.2559	50.4568
5.2	30	200	439.4261	481.7334	456.5686	13.8828
5.3	30	200	423.7406	443.1785	428.0542	6.8410
5.1	50	500	1105.0471	1232.2534	1197.2366	59.0732
5.2	50	500	455.3299	5000.7331	470.4657	15.5938
5.3	50	500	431.3452	451.1738	440.3928	5.9080
5.1	75	1000	1941.9276	2059.5783	2020.3484	44.5709
5.2	75	1000	580.5929	664.2999	607.4263	24.7964
5.3	75	1000	564.7203	595.0924	585.0293	14.2613

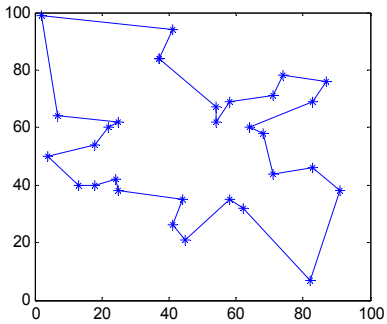

图 5.1 算法 5.3 所求的 30 城市最优回路
Fig5.1 The best circuit of 30 cities with algorithm 5.3

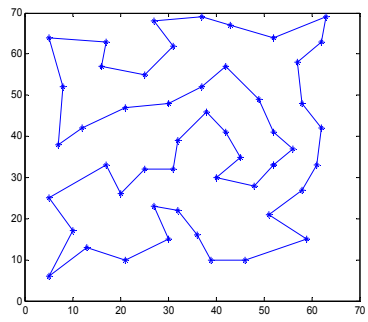

图 5.2 算法 5.3 所求的 50 城市最优回路
Fig5.2 The best circuit of 50 cities with algorithm 5.3

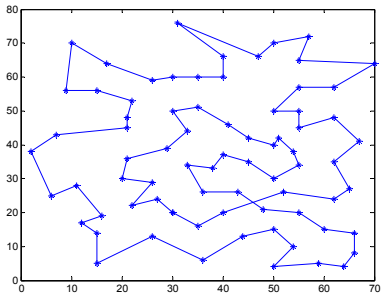

图 5.3 算法 5.3 所求的 75 城市最优回路
Fig5.3 The best circuit of 75 cities with algorithm 5.3

5.4 动态顺序插入交叉算子

5.4.1 动态顺序插入交叉算子的设计

定义1'：设 A 是一回路，b 是该回路中的一个城市 ，a 是城市 b 的前邻城市，c 是城市 b 的后邻城市，定义三角距离差函数 $G(b,A)$ 如下:

$$G(b,A)=d(a,b)+d(b,c)-d(a,c)$$

其中 $d(x,y)$ 表示城市 x 到城市 y 的距离。

定义 2：设有下列路径 $(a,x_1,x_2,\cdots x_m,b)$，$m\neq 0$ 定义广义三角距离差函数如下：

$$H(a,x_1,x_2,\cdots x_m,b)=d(a,x_1)+\sum_{i=1}^{m-1}d(x_i,x_{i+1})+d(x_m,b)-d(a,b)。$$

动态顺序插入交叉算子的基本设计思想：对于两个待交叉的染色体，将一个作为基本染色体，另一个作为参考染色体，按照在参考染色体中城市间的邻接顺序，利用三角距离差函数和广义三角距离差函数作为评价标准，运用贪婪策略思想，通过随机动态地调整交叉步长，逐步改变基本染色体的编码顺序，从而使经过交叉得到的子代染色体的适应度提高。

现在有待交叉的双亲：A：$(a_1,a_2,\cdots a_n)$，B：$(b_1,b_2,\cdots b_n)$，结合顺序交叉(OX)算子和贪婪策略，我们设计算子如下：

① 首先，将父体 A 作为基本体，将父体 B 作为参考体，在 B 中随机找到一个参考城市 b_j ($j=1,2,\cdots n$)，设 $m=0$；

② 设 $g=0$，随机产生一个步长整数 u $(u\le n/q)$,其中 $q(2\le q\le n)$ 是一个整数参数（q 的具体取值根据城市数 n 选择），$m\leftarrow m+u$，如果 $m>n$, 则 $u=m-n$，$m=n$。在 A 中计算 $G(b_{j+1},A)$,则 $g\leftarrow g+G(b_{j+1},A)$，在 A 删除城市 b_{j+1}，得到新的 A_1，然后再计算 $G(b_{j+2},A_1)$，$g\leftarrow g+G(b_{j+2},A_1)$，在刚得到的 A_1 中删除 b_{j+2}，得到新的 A_2，…，最后计算 $G(b_{j+m},A_{j+m-1})$，$g\leftarrow g+G(b_{j+m},A_{j+m-1})$，在刚得到的 A_{j+m-1} 中删除 b_{j+m}，得到新的 A_m。

③ 在新得到的 A_m 中找到 $a_i=b_j$，计算 $H(a_i,b_{j+1},\cdots b_{j+m},a_{i+1})$，比较 g 与 $H(a_i,b_{j+1},\cdots b_{j+m},a_{i+1})$，如果 $H(a_i,b_{j+1},\cdots b_{j+m},a_{i+1})<g$,则在 A_m 中 a_i 的后面插入 $b_{j+1},\cdots b_{j+m}$，得到新的染色体 A，否则 A 保持不变。

④ 重复上述步骤②、③，直至 $m=n$，结束，得到最终子代 A。

需要指出的是，动态顺序插入交叉算子是在参考城市一定的情况下，对基本体整体按照局部最优进行插入的，并且本算子在交叉运算过程中，随着动态交叉长度的随机选择，从而使一些较长的优秀基因片段得以保留，提高了算法的收敛速度。

以 $n=9$ 的 TSP 问题为例，假设城市间的距离用邻接矩阵表示如下：

$$\begin{pmatrix} 0 & 26 & 28 & 4 & 37 & 39 & 8 & 33 & 6 \\ 26 & 0 & 5 & 22 & 26 & 24 & 20 & 8 & 24 \\ 28 & 5 & 0 & 24 & 21 & 20 & 21 & 8 & 26 \\ 4 & 22 & 24 & 0 & 34 & 36 & 5 & 30 & 5 \\ 37 & 26 & 21 & 34 & 0 & 6 & 30 & 26 & 38 \\ 39 & 24 & 20 & 36 & 6 & 0 & 32 & 23 & 40 \\ 8 & 20 & 21 & 5 & 30 & 32 & 0 & 27 & 9 \\ 33 & 8 & 8 & 30 & 26 & 23 & 27 & 0 & 31 \\ 6 & 24 & 26 & 5 & 38 & 40 & 9 & 31 & 0 \end{pmatrix}$$

设待交叉的双亲：

A: 1 2 3 4 5 6 7 8 9

B: 1 8 2 7 4 6 3 5 9

经过本文提出的动态顺序插入交叉算子运算，得到两个子代个体：

A_1: 2 3 4 5 6 7 9 1 8

B_1: 7 4 3 2 1 9 8 6 5

可以计算得：

$Length(A)=191, Length(B)=187, Length(A_1)=157, Length(B_1)=156$。

显然，子代个体 A_1, B_1 都比它们的父代染色体所代表的路径长度小，并且它们很好地继承了父代优秀的基因。

5.4.2 实例仿真

为了验证本文提出的动态顺序插入交叉（DOIC）算子，将本文提出的交叉算子同时与插入交叉算子和两交换启发式交叉算子进行比较，并设计了相应的算法 5.4、算法 5.5 和算法 5.6，算法 5.4 为插入交叉算子，算法 5.5 为提出的两交换启发式交叉算子，算法 5.6 为动态顺序插入交叉算子(参数 $q=10$)。其中三个算法的选择算子均采用轮转法，变异算子采用对换变异，并对变异结果执行进化逆转操作。

本文算法所采用的进化逆转操作为：对于染色体 A，随机产生一个1到 n 的排列 $(a_1, a_2, \cdots a_n)$，然后依次逆转染色体 A 中第 a_i 个基因与第 a_{i+1} 个基因之间的次序 $(i=1,2,\cdots,n-1)$，直至所得到的新染色体比原染色体优秀，则接受此染色体，进化逆转操作结束。通过比较求解标准的30、50和75个城市的TSP问题(交叉率为0.9 ，变异率为0.2),得到了表5.2所示的结果。程序分别运行10次。

表5.2 三种算法求解TSP问题的结果

Table5.2 The results of 3 Algorithms for TSP

算法	城市个数	种群规模	进化代数	最优回路长度	最差回路长度	平均回路长度	标准差
5.4	30	60	50	426.5505	475.4081	452.2828	15.3845
5.5	30	60	50	430.1980	441.4743	437.6097	5.1485
5.6	30	60	50	423.7406	423.7406	423.7406	0
5.4	50	100	200	446.0063	462.7418	454.2096	6.7968
5.5	50	100	200	453.2835	463.5302	457.9728	4.8596
5.6	50	100	200	427.8552	430.1540	428.3997	0.8892
5.4	75	100	500	595.8852	620.9469	607.1974	10.0411
5.5	75	100	500	569.0307	582.7187	575.6571	6.1126
5.6	75	100	500	543.4474	551.8352	547.6151	3.2714

由表 5.2 可以看出本文所提出的动态顺序插入交叉算子，在各种性能上都超过了插入交叉算子和两交换启发式交叉算子，并且在进化代数较少的情况下，能得到比较好的计算结果。

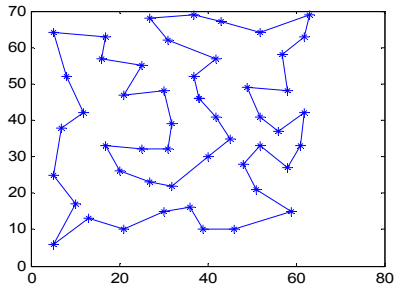

图 5.4 算法 5.6 所求的 50 城市最优回路

Fig5.4 The best circuit of 50 cities with algorithm 5.6

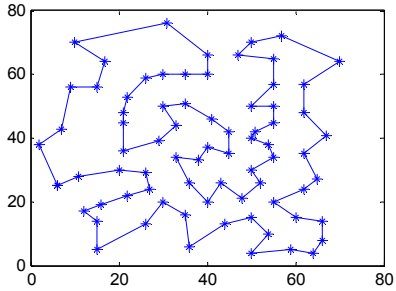

图 5.5 算法 5.6 所求的 75 城市最优回路

Fig5.5 The best circuit of 75 cities with algorithm 5.6

同时，在求解 75 个城市时，算法 5.6 所得到的最优解 543.4474 ，与许多文献中所提供的最优解比较，仍是一个比较优的解。

5.5 进一步分析

5.3 和 5.4 两节分别介绍了顺序插入交叉算子与动态顺序插入交叉算子的设计思想和方法，并且通过实例仿真，验证了它们的有效性。由于在实例仿真中，分别使用了不同的遗传参数以及是否使用逆转操作，从而使得这两种算子的优劣很难辨别。为了使大家对这两种算子有更好的了解，为此设计了算法 5.7 和算法 5.8，两种算法均使用相同的遗传参数，并且不使用逆转操作，来求解标准的 30、50 和

75 个城市的 TSP 问题(交叉率为 0.9 ,变异率为 0.2), 程序分别运行 10 次，得到了表 5.3 所示的结果。

表5.3 两种算法求解TSP问题的结果
Table5.3 The results of 2 Algorithms for TSP

算法	城市个数	种群规模	进化代数	最优回路长度	最差回路长度	平均回路长度	标准差
5.7	30	30	100	423.7406	450.7255	436.8718	13.5069
5.8	30	30	100	423.7406	455.7809	429.3977	12.9335
5.7	50	50	200	442.6305	467.1312	456.0146	12.4067
5.8	50	50	200	427.9651	450.2448	438.3695	11.2124
5.7	75	100	200	576.5620	594.0809	579.6276	10.2767
5.8	75	100	200	557.3119	574.4345	563.1922	7.7817

图 5.6、图 5.7 分别为顺序插入交叉（OIC）算子和动态顺序插入交叉（DOIC）算子求解 75 个城市的运算过程。从中我们可以看到两种算子均具有较快的收敛速度。而两种交叉算子的区别主要在于动态顺序插入交叉算子能够动态地调整它的交叉步长。

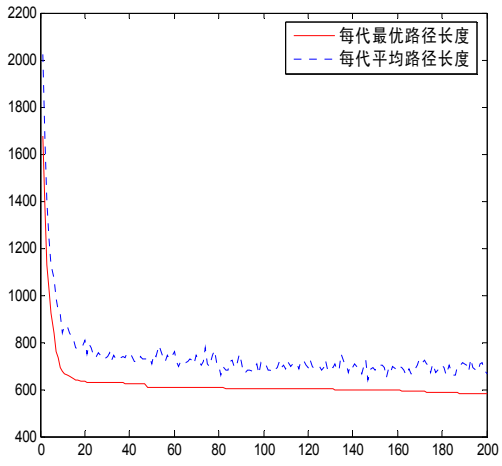

图 5.6 OIC 交叉算子的运算过程
Fig5.6 The operation process of OIC operator

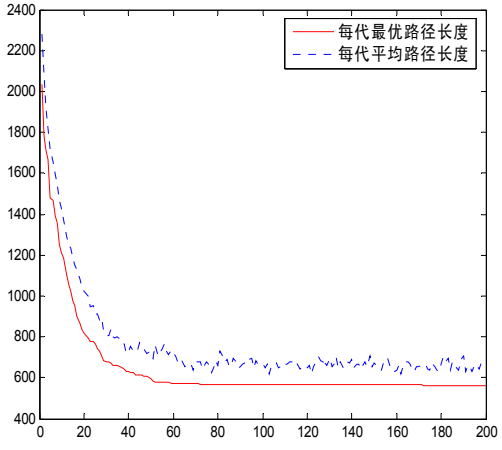

图 5.7 DOIC 交叉算子的运算过程
Fig5.7 The operation process of DOIC operator

并且由表 5.3 可以看出顺序插入交叉算子与动态顺序插入交叉算子在解决小规模的 TSP 问题时的能力是没有很大区别的，但随着城市数目的增加，动态顺序插入交叉算子在解决较大规模的 TSP 的优势将会逐渐显现出来。

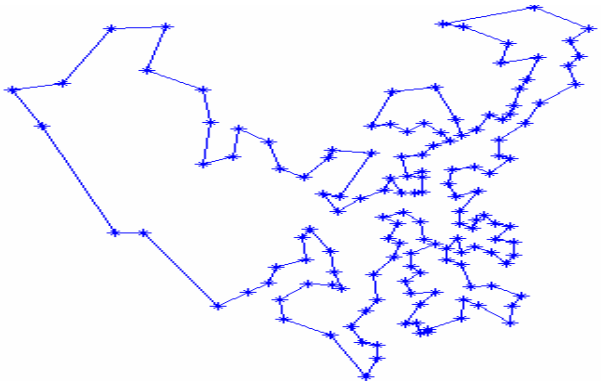

图 5.8 本文算法所求的 CHN144 最优回路
Fig5.8 The best circuit of CHN144 with the algorithm of this article

CHN144是由我国144个城市构成的一个对称的TSP问题。图5.8是利用动态顺序插入交叉算子求解该问题的一次运算结果。显然，所求回路是一个比较好的结果。

以上所解决的TSP问题均是对称的，实际上本文设计的两种交叉算子也可以用来解决非对称的TSP问题。假设如下是非对称的9城市的距离矩阵，显然该TSP问题存在一条最优路径1-3-5-7-8-4-6-9-2。

$$\begin{pmatrix} 0 & 2 & 1 & 4 & 7 & 3 & 4 & 9 & 10 \\ 1 & 0 & 9 & 2 & 2 & 4 & 5 & 2 & 4 \\ 8 & 5 & 0 & 14 & 1 & 2 & 2 & 3 & 2 \\ 4 & 2 & 4 & 0 & 4 & 1 & 5 & 5 & 2 \\ 7 & 6 & 2 & 3 & 0 & 9 & 1 & 15 & 8 \\ 9 & 2 & 2 & 6 & 6 & 0 & 16 & 23 & 1 \\ 8 & 2 & 2 & 5 & 3 & 2 & 0 & 1 & 19 \\ 3 & 8 & 8 & 1 & 6 & 9 & 27 & 0 & 3 \\ 6 & 1 & 6 & 5 & 8 & 4 & 9 & 31 & 0 \end{pmatrix}$$

采用本文提出的动态顺序插入交叉算子，选择算子采用轮转法，变异算子采用对换变异。遗传参数分别为：交叉概率为0.9；变异概率为0.2；种群个数10。程序仅运行到第4代就得到了最优解。

本文提出的顺序交叉算子和动态顺序交叉算子的算法复杂度一样，都是$O(n^2)$。但本文提出的交叉算子在相同的条件下得到了比较好的近似结果，因此，本文提出的改进遗传算法无论在理论上还是在实际应用上都是有价值和有意义的。

5.6 小结

本章对求解TSP问题的常用遗传交叉算子进行了分析和研究，在此基础上，设计了新的交叉算子：顺序插入交叉算子与动态顺序插入交叉算子。将本文所设计的交叉算子应用于TSP问题，并与已有的交叉算子进行了数值实验比较，取得了较为理想的结果，从而验证了本文设计的改进遗传算法的可行性。

6 结束语

6.1 本文工作总结

本文所做的工作主要有：

① 阐述了遗传算法(GA)的基本原理，以及 GA 的数学基础，给出了 GA 的完整的框架，详细地分析了 GA 的编码、适应度函数、遗传操作、参数设置等步骤，并且简单介绍了遗传算法的应用。

② 介绍了旅行商问题的定义、分类、扩展形式、研究难点和应用领域等，其次介绍了旅行商问题的基本算法，其中重点介绍了目前最有发展潜力的神经网络、遗传算法等。

③ 主要介绍了标准遗传算法求解 TSP 问题的基本方法，同时深入研究了几种求解 TSP 问题的改进遗传算法。

④ 分析了在遗传算法求解 TSP 问题中常用的几种交叉算子的主要缺陷，然后设计了两个新的交叉算子：顺序插入交叉算子与动态顺序插入交叉算子。同时，进行了数值仿真实验，并与其它已有的交叉算子进行了比较，从而验证了本文设计的算法是切实可行有效的。

6.2 展望

本文的研究工作取得了一些有价值、有意义的成果，提供了一些具有启发性的思路，实验结果也表明本文提出的改进算法是可行有效的。

由于研究该问题的时间有限，TSP 问题又是一个世界公认的难题，在该研究方向上，今后仍有许多工作可做，主要包括：

① 继续探讨 TSP 问题的具体特性，对编码中的基因组合与解的关系进行严格的数学分析，这就和对人类基因片断与疾病关系的研究类似，因为我们只有知道这种相关性，才能设计出有针对性的高效的遗传算法，快速地找到精确度较高的近似最优解。

② 对于文中提到的改进算法仅是通过实验证明了它的有效性，还可以从理论上严格证明该算法的有效性。

③ 将本文提出的改进遗传算法的基本思想用于解决其它的组合优化问题，如 Job-Shop 问题等。