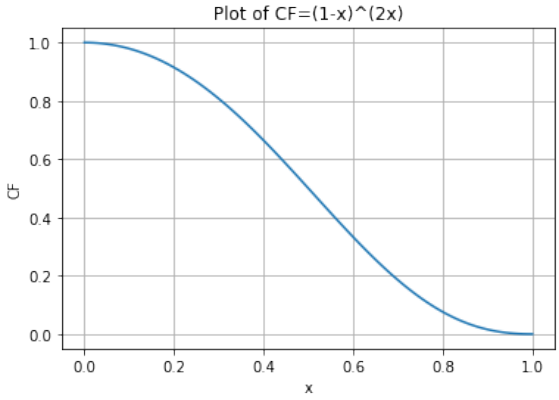

图 2-3　CF 变化图

当他们返回交换保姆时，按照保姆交换标准，很久没觅食的成员会成为保姆。一旦保姆被交换，他们就不会回到先前觅食的地方，以避免过度放牧。

侦察兵会发现一个新的觅食地点，并通知雌性首领，带领家族到达新的地点。

可以看出，侏儒猫鼬算法中种群的主体部分（阿尔法小组或者侦察兵小组）一次迭代中会进行两次更新行为。

显而易见，侏儒猫鼬算法在进行每一步的更新时，使用了类似于梯度（如公式2.18）的更新方式，因此该算法只能解决连续形式的问题，不能解决离散形式的问题，需要本论文将其改造为像基因算法一样的离散型算法。

$$X_i^{t+1} = X_i^t + \text{phi} \times \text{peep} \times (X_i^t - X_{peep}) \tag{2.18}$$

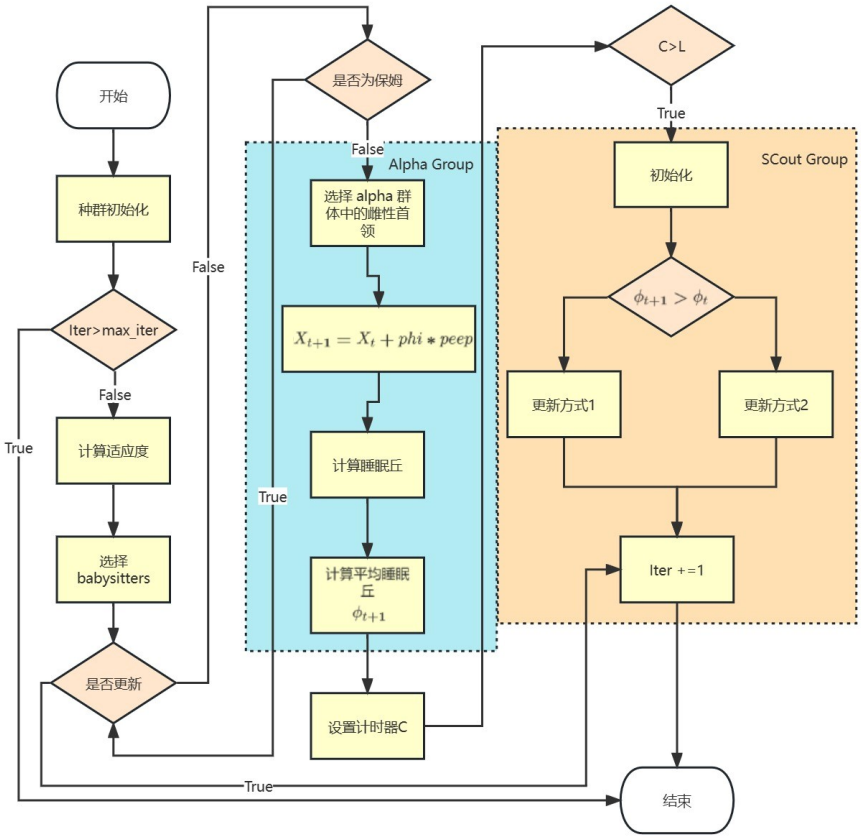

图 2-4　侏儒猫鼬算法流程图

2.4　本章小结

本章详细介绍了将算法修改为离散形式的相关知识，包括数学规划和数学规划描述形式本章所介绍的内容将为后续离散算法的提出提供借鉴的基础。

表 3-1　本文符号说明

参数	解释
T	资源共享时间
C_{it}	CPU 容量
r_i	车辆 i 的工作负载
r'_{jt}	资源请求者 j 需要的资源
M_{it}	每秒百万次指令数（MIPS）
θ_i	估计参数
E_{it}^{blnc}	车辆 i 消耗的能量
E_i^{rec}，E_i^{send}	车辆 i 接收或发送的能量

3.2　问题建模

在本节中，本论文将问题分为两种情况来建模，一种是任务可分情况（3.3.1 节），另一种是任务不可分情况（3.3.2 节）。当任务可分的情况下，问题能够直接求解数学规划的方式解决；当任务不可分时，不能通过求解数学规划的方式来解决。

3.2.1　任务可分情况

当车辆在进行某些同质任务时，可以将大任务任意的拆成一些小任务，因为任务的粒度太小，因此可以近似看成任何一个任务可以随意拆分然后分配到任意的车辆上，也就是任务可分的情况。

在任务可分时，根据本地留存的高优先级任务的大小又分为：高于平均任务（大任务），和低于平均任务大小（小任务）这两种情况。

（1）小任务情况

当本地任务低于平均任务大小（小任务）时，该问题较为简单，直接将所有任务平均分配到车辆上就能解决。可以看出，该问题等价于：首先将固定值 C 切制成 n 个数字，使得的他们的三次方之和最小。设切分的数分别为 $a_1, a_2, \cdots, a_n$，根据均值不等式：$\sqrt[r]{\frac{\sum_{i=1}^{n} a_i^r}{n}} \geq \frac{\sum_{i=1}^{n} a_i}{n}$，可得：$\frac{\sum_{i=1}^{n} a_i^3}{n} \geq \left(\frac{\sum_{i=1}^{n} a_i}{n}\right)^3$，其中当 $a_1 = a_2 = ... = a_n = \frac{C}{n}$ 时，等号成立。

因此，要使 $\sum_{i=1}^{n} a_i^3$ 最小，需要让 $\sum_{i=1}^{n} a_i$ 的值尽可能平均分配给 n 个数，即令 $a_1 = a_2 = \cdots = a_n = \frac{C}{n}$。

当本地任务是小任务时，能够保证参与资源共享的车辆，任务的大小可以卸载为平均值。

（2）大任务情况

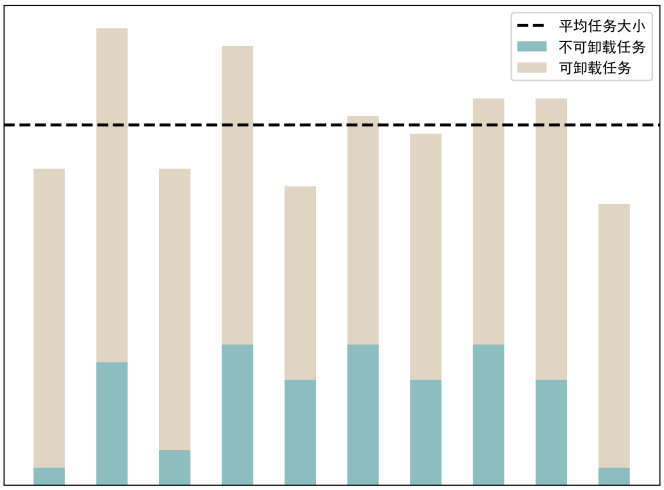

图 3-2　小任务情况

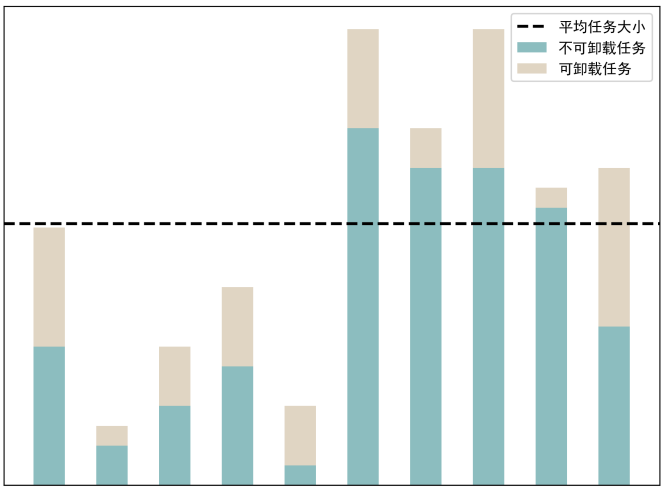

图 3-3　大任务情况

但是当任务高于平均任务大小（大任务）时，该问题需要进行分类的讨论，是否还需要将再给有大任务的汽车分配任务。

该问题等价于：有一个固定值 C，将其切割为 n 个数，要求这 n 个数的三次方最小，已知有 m 个数必须大于 $\frac{c}{n}$，如何切割。

在有限制条件的情况下，该问题可以使用拉格朗日乘数法来求解。设切分后的数为 $a_1, a_2, \cdots, a_n$，且有 m 个数大于 $\frac{c}{n}$。

有以下约束条件：

表 3-2　仿真参数

参数	值/分布	参数	值/分布
f_i	[700, 1900]	r_{it}	U[820, 10000]
v_i	7.683	r'_{it}	U[200, r_{it}]
θ	−4558.52	P_0	0.2
λ	0.00125	W	10 MHz
ΔT	1	SNR	677

由于我们的空间有限，我们假设车辆共享资源的时间为 $T = 10$ 秒和 $\Delta T = 1$ 秒。

如公式（3.3）所示，指令数为 $M_{it} = v_i, \cdot f_{it} + \theta_i$。并且公式 (3.4) 中的 CPU 容量为 $C_{it} = (v_i \cdot f_{it} + \theta_i) \times \Delta T - r_{it}$。基于[57] 中提供的分析，CPU 容量参数 v_i和θ_i 其值分别为 7.683 和 −4558.52。

经过计算，M 的最大值和最小值分别为 819 和 10039。

所有任务执行工作负载所需的计算资源 r_{it} 服从于 U[820，10000] 的均匀分布，并且假设车辆 i 可以卸载给其他车辆的任务计算资源值 r'_{it} 服从于 U[200，r_{it}]，因为某些子任务必须在本地执行。

在本文的实验中，数据的大小从 1MB 到 10MB 不等。我们设定带宽 d 的值为 27Mb/s。

本文选择了 7 种种类的参与资源共享的车辆数目，分别是 10，15，20，25，30，35，40。在算法中，种群数量被设置为 40。而当车辆数量为 35 和 40 时，种群数量被设置为 150。因为当车辆数量为 35 和 40 时，种群的规模太小，算法不会收敛或者不能很好的收敛到最优值附近。

本文将变异概率设置为 0.01，参数用于控制每个个体发生变异的概率。该概率越高，个体发生变异的可能性就越大。一般情况下，变异概率会设置为较大的值，以保持种群的多样性，并防止算法陷入局部最优解。

3.3.2　评价标准

本论文设置了个指标来衡量本论文的实验，分别是，性能指标，公平指标，种群离散程度。在实验中，本论文的性能指标是指节能的百分比。节能的百分比，其定义如下：

$$P = \mathrm{m\,in}\left(100 \cdot \left(1 - \frac{\sum_{i=1}^{N} E_{it}^{blne}}{E_{lc}}\right)\right), t = 1, \ldots, T \tag{3.24}$$

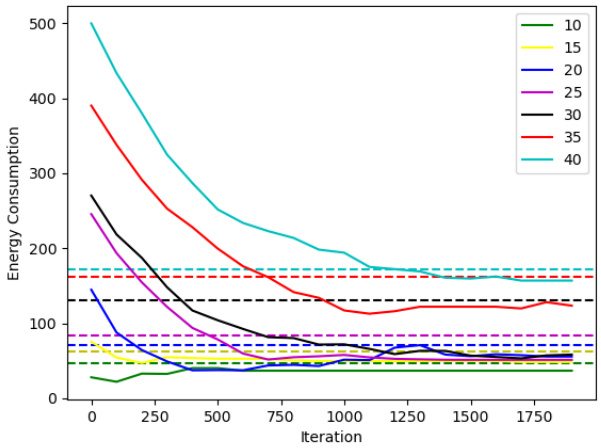

图 3-4　能耗图

其中，E_{loc} 是本地执行任务所需要消耗的能量。为了反映本论文任务分配的公平性，本论文在标准差的基础上定义了公平系数（fairness coefficient，FC）。与标准差一样，数值越小，公平性越高。标准差定义如下：

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}\left(x_i - \mu\right)^2}{n}} \tag{3.25}$$

FC 的定义如下：

$$FC = \max \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(E_{it}^{\text{blnc}} - \overline{E}\right)^2}}{\overline{E}}, t = 1, \ldots, T \tag{3.26}$$

其中，t 是迭代的次数，$\bar{E}$ 是所有车辆能耗的平均值。

3.3.3　实验结果

在图 3-4中，虚线是任务没有负载出去而在本地执行所需要消耗的能量。本论文可以看到，当车辆数为 10 时，该算法迭代几十轮就可以得到很好的结果。当车辆数为 20 时，需要 350 次迭代才能得到一个较好的结果。而当车辆数量为 30 时，能耗在 750 次迭代之前变化较大，而在 750 次迭代之后变化就会明显变小。也就是说，能源消耗的减少率在 750 次迭代前较快，而在 750 次迭代后，能源消耗的减少率就会降低。当车辆数为 40 时，几乎没有节能效果。

可以得出结论，随着车辆数量的增加，迭代的次数也在增加。因此，在边缘服务器上运行该算法时，需要合理地调整迭代次数，因为车辆数量的变化会显著的影响算法的运行时间。

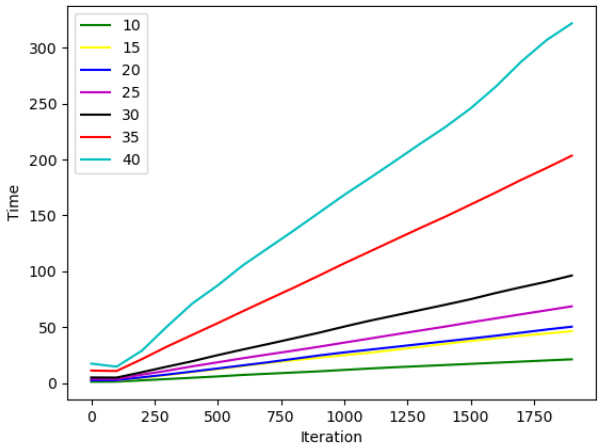

图 3-5　时间图

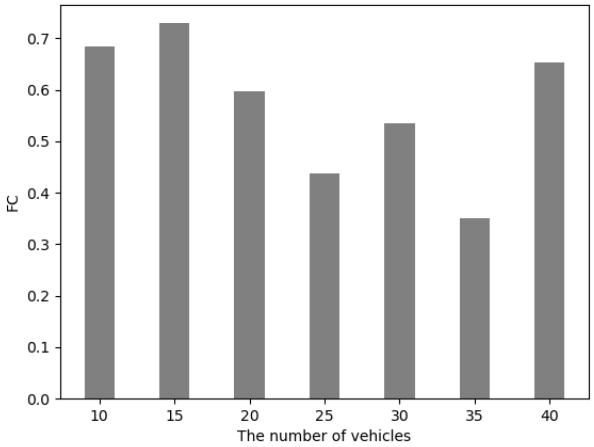

图 3-6　平衡因子 FC 值

在图 3-5中，经过 100 次迭代以后，所有数目运行时间和迭代次数呈线性关系。然而，当迭代次数固定时，算法的运行时间和车辆之间的关系不是线性的，而是非线性的，参与资源共享的车辆的数越多，潜在的分配方案越多，算法的运行时间增加的越多。

如图 3-6所示，随着车辆的增加，车辆能量消耗的平衡首先增加，然后减少。当车辆数目为 30 和 40 时，能耗的平衡系数反而增加。

如图 3-7所示，当车辆数量在 10 到 25 辆之间时，本论文的系统可以节省 20%以上的能量消耗，但是当车辆数量为 30 辆时，只有 17% 的能量消耗。但当车辆数为 30 时，只能节省 17%，而当车辆数为 40 时，几乎可以节省 20%。当车辆数量为

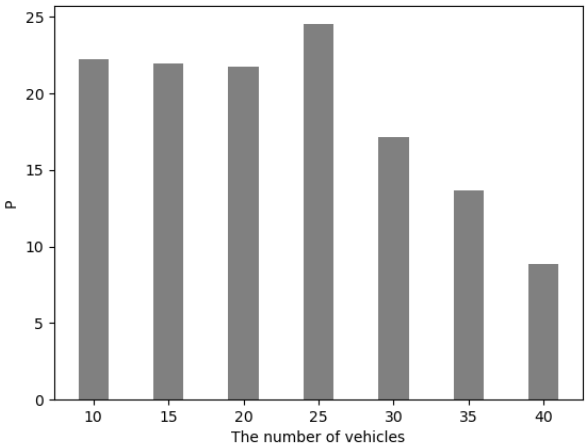

图 3-7　性能度量 P 值图

40 辆时，几乎没有节省能量。

综合考虑性能测量、算法运行时间以及平衡因子，选择 25 辆汽车参与资源共享是非常合适的。

3.4　本章小结

本章所研究的是车联网领域的任务卸载和分配的方案，通过能耗优先的原则设计出一套卸载算法。首先，本章详细介绍了车联网中任务卸载的系统模型，并对其中的要素进行了说明，包括系统场景、车辆计算模型及车辆能耗模型。接着，作者将问题建模为数学规划形式，提出了一种基于粒子群算法的卸载与分配策略，然后设计了相关评价标准，并对实验结果进行了分析。评价标准表明，该算法能够很好地提高资源利用率，同时也兼顾了公平性。

然而，在研究过程中也发现了一些缺陷，比如，基因算法存在着编码和解码的过程，需要进行交叉和变异操作，这导致了计算量太大和计算时间过长等问题。因此，在下一步的研究中，本论文将引入新的智能算法，并将其改进为适用于离散问题的算法，以加快算法的运行时间。

表 4-1　实验环境

名称	型号或版本	名称	型号或版本
处理器	Intel(R) Core(TM) i7-7700HQ	Python	3.8
显卡	NVIDIA GeForce RTX 1050	Numpy	1.19
RAM	8.0GB	Matplotlib	3.3
操作系统	Windows10 专业版		

距离等[58] [59]。但是相同的是，距离越远代表着种群多样性越丰富，反之，距离越小则代表着的种群多样性越差。

第二种方法是基于频率的测量方法（frequency-based measurement，FBM)。在 FBM 中，通过统计具有代表性、有优势的个体的频率[60]或者平均基因频率[61]来评价种群多样性。此外，熵度量是属于 FBM 的另一种方法，因为熵（公式4.1）使用了基因出现的概率/适应度。

$$S = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{4.1}$$

其中，(S) 表示熵，(n) 表示系统中可能的状态数，(p_i) 表示第 (i) 个状态发生的概率。

在本文中，本论文使用了一种度量方法，即将种群中每个解的各个维度的方差之和的二分之一次方作为多样性 D 的衡量标准。下面是计算多样性 D 的公式：

$$D = \sqrt{\sum_{j=1}^{m} D_j^2},\ D_j = \frac{1}{n}\sum_{i=1}^{n}\left(x_{ij} - \overline{x}_j\right)^2 \tag{4.2}$$

其中，m 是解的维度，n 是种群的数量，x_{ij} 是是第 i 个粒子的第 j 个维数。

（2）算法运行时间。算法的运行时间可以直接反映出算法的性能，较短的运行时间通常代表着更高效的算法。通过比较不同算法的运行时间，可以客观评估它们的性能优劣。对于需要实时响应的应用，算法的运行时间直接影响用户体验。短时间内得到结果可以提高用户满意度和使用体验。

4.4.4　实验结果及分析

在图 4-5中，绿色是没有更新的解所对应的函数值，黄色柱状图是更新解以后所对应的变量值，可以看出，如果仅仅只是交换两个变量值的更新方式，更新和不更新没有什么显著的区别。

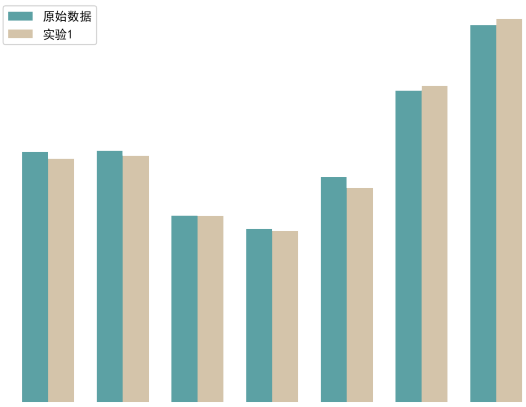

图 4-5　实验一实验结果

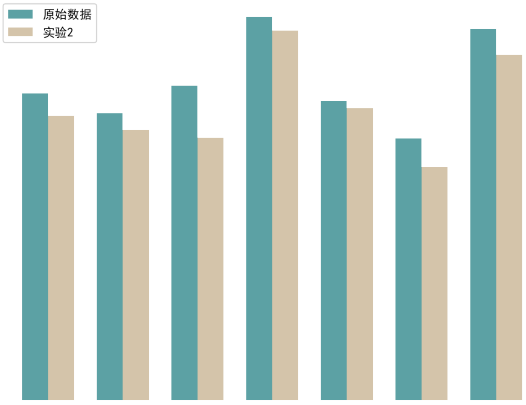

图 4-6　实验二实验结果

在图 4-6中，绿色是没有本地执行所对应的函数值，黄色柱状图是进行交叉操作以后所对应的变量值，可以看出，虽然比实验一效果要好，但是这种交叉这种更新策略效果并不明显，且这几次实验之间的差值比较大，更加说明了变量之间有相关性。

图 4-7 中可以看出，虚线代表基因算法求解问题的时间，而实线则是侏儒猫鼬算法的运行时间。标签前面的数字表示车辆的数量，而同颜色的线代表着相同数量的车辆。值得注意的是，侏儒猫鼬算法的运行时间明显比基因算法的运行时间要短。

当车辆数量为 15 和 20 时，侏儒猫鼬算法的运行时间只有原算法的三分之二；而当车辆数量为 10 和 25 时，则能将运行时间节省约四分之一。因此，从图中可以清晰地看出侏儒猫鼬算法在解决问题时具有更高的效率和优势。

通过观察图 4-8，可以发现基因算法种群差异度一直比较小，侏儒猫鼬算法种

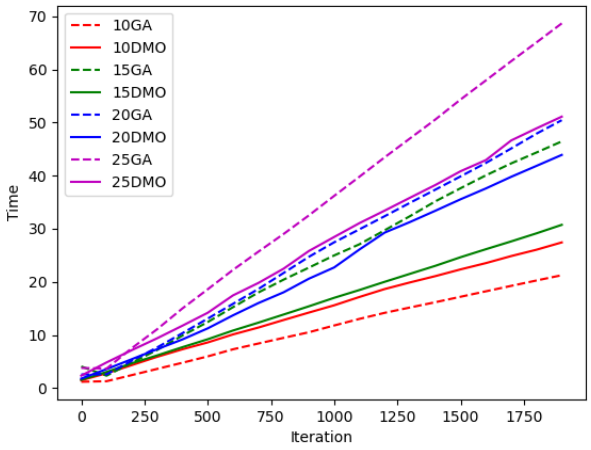

图 4-7　算法运行时间图

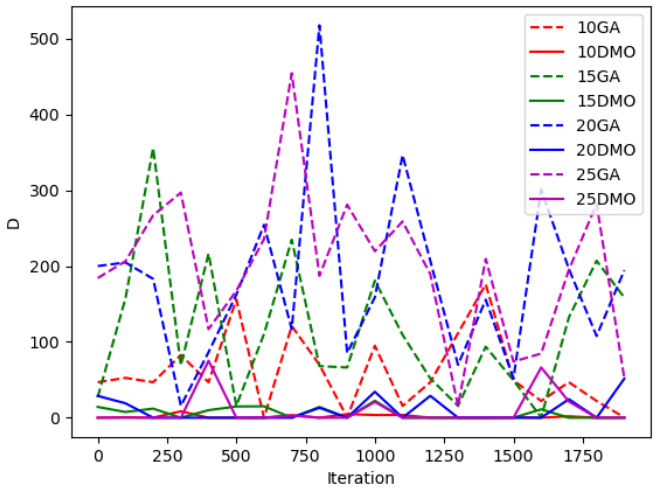

图 4-8　侏儒猫鼬算法种群差异度图

群差异度的变化幅度较大。根据种群差异度的定义，这表明，在算法运行过程中仍在产生新的解，并且新产生的解种群差异度较大。这样一来，侏儒猫鼬算法就能够避免陷入劣解，有可能产生更优质的新解。

4.5　本章小结

本章在上一章的基础上，探索了新的方法来解决第三章提出的问题。本论文解决了基因算法中编码解码过程耗时较长的问题，并通过对侏儒猫鼬算法的改进，将其转变为离散侏儒猫鼬算法。实验结果表明，与基因算法相比，本论文的算法在运行时间上有显著的缩短。