文章编号:1007－6735(2019)04－0344－06　　DOI: 10.13255/j.cnki.jusst.2019.04.007

求解 P 中位问题的混合蝙蝠算法

王婷婷，　张惠珍
（上海理工大学 管理学院，上海 200093）

摘要：根据 P 中位问题的数学模型及其具体特征，重新定义了蝙蝠位置与位置之间的减法操作算子、速度与位置之间的加法操作算子和可行化函数，引入了遗传算法中交叉的思想对当前解进行局部搜索，提出了求解该问题的混合蝙蝠算法。通过对多个 P 中位算例进行测试，并将测试结果与其他算法进行比较，验证了该混合蝙蝠算法求解 P 中位问题的可行性与有效性。

关键词：P 中位问题；蝙蝠算法；可行化函数；交叉
中图分类号：TP 301.6　　**文献标志码：**A

Hybrid Bat Algorithm for Solving a P-Median Problem

Wang Tingting,　Zhang Huizhen
(*Business School, University of Shanghai for Science and Technology, Shanghai 200093, China*)

Abstract: Based on the mathematical model and specific features of a P-median problem, the subtraction operator between the location and the location of bats, the addition operator between the velocity and location as well as the feasible function were redefined. The idea of crossover in genetic algorithm was introduced in order to perform a local search on the current solution and a hybrid bat algorithm (HBA) was proposed for the problem. By some tests on the P-median problem and comparisons with other algorithms, the computational results show that the hybrid bat algorithm is feasible and efficient for solving P-median problems.

Keywords: *P-median problem*; *bat algorithm*; *feasible function*; *crossover*

设施选址问题是运筹学中的经典问题之一，在生产、生活及物流等方面都有着非常广泛的应用，如工厂、仓库、急救中心、消防站及物流中心的选址等。而 P 中位问题（P-median problem，PMP）就是一种很常见的设施选址问题，该问题最初由 Hakimi[1-2]在 1964 年提出的，目的是要在给定没有容量限制的 m 个候选设施集中选择 $p(p<m)$ 个作为开放设施，使得所有顾客到这些中位点的距离之和达到最小。

目前，用于求解此问题的方法多种多样，如动态规划方法[3]、拉格朗日松弛算法[4-5]等经典求解方法。但是，这些经典方法仅对小规模的问题有效,而当求解问题的规模较大时，其计算复杂度也将极大地增加，乃至无法求解。由于 Garey 和

收稿日期: 2018−03−06
第一作者: 王婷婷（1994−），女，硕士研究生．研究方向：运筹学、智能优化．E-mail：1293801837@88.com
通信作者: 张惠珍（1979−），女，副教授．研究方向：运筹学、智能优化．E-mail：zhzzywz@163.com

Johnson[6]在 1979 年通过计算复杂性理论已经证明了该问题是 NP-hard 问题，因此，除非 P=NP，否则不存在多项式时间算法求解该问题。所以，绝大多数学者都致力于启发式算法的研究，以期得到问题的满意解或近似最优解。许多传统的优化算法已被用来求解 PMP，如局部搜索算法[7-8]、随机搜索算法[9]等。此外，人们也提出了多种现代启发式算法，如模拟退火算法[10]、遗传算法[11-13]及粒子群算法[14]等。这些启发式算法能够生成较好的最优解或近似最优解，特别是以遗传算法为代表的仿生算法，以其自组织和自适应等良好特征，被广大学者研究并应用到组合优化问题的求解过程中。而本文正是将蝙蝠算法这种具有良好优化性能和发展前景的仿生算法用于求解 PMP，以期能够克服已有算法的局限性，获得较好的优化性能。

蝙蝠算法（bat algorithm，BA）是 Yang[15]在 2010 年提出的一种新型启发式算法，这是一种搜索全局最优解的有效方法。自提出以来，该算法已经被证明可以应用于多种实际问题的求解。Nakamura 等[16]提出了一种二进制蝙蝠算法来解决特征选择问题；李枝勇等[17]在元胞自动机原理和基本蝙蝠算法的基础上提出了一种元胞蝙蝠算法用于 0-1 规划问题的研究；Rizk-Allah 等[18]给出了一种二元蝙蝠算法实现 0-1 背包问题的求解。但是，至目前尚未有文献将 BA 应用于 PMP 的求解。因此，本文基于 BA 求解离散问题的局限性，结合 PMP 的具体特征，重新定义了 BA 的操作算子，提出了求解 PMP 的混合蝙蝠算法（hybrid bat algorithm，HBA），并与其他智能算法如遗传算法[13]和粒子群算法[14]进行比较，不仅验证了该混合蝙蝠算法用来求解 PMP 的有效性，而且拓宽了 BA 的应用领域。

1　P 中位问题

假设给定设施集 $M=\{1,2,\cdots,m\}$，客户集 $N=\{1,2,\cdots,n\}$，对于任意给定的 $i\in M$，$j\in N$，d_{ij} 表示设施 i 与客户 j 之间的距离，所有的候选设施都没有容量限制，要求在 m 个候选位置点中选择 p 个位置作为开放设施（中位点），并且要求每个客户必须选择一个且只能选择一个开放设施来满足其需求，同时使总距离最小。PMP 的数学模型可被描述为

$$\min z=\sum_{i=1}^{m}\sum_{j=1}^{n}d_{ij}x_{ij} \tag{1}$$

$$\sum_{i=1}^{m}x_{ij}=1,\quad \forall j\in N \tag{2}$$

$$x_{ij}\leqslant y_i,\quad \forall i\in M,j\in N \tag{3}$$

$$\sum_{i=1}^{m}y_i=p,\quad \forall i\in M \tag{4}$$

$$x_{ij}\in\{0,1\},\forall i\in M,j\in N \tag{5}$$

$$y_i\in\{0,1\},\forall i\in M \tag{6}$$

式中：z 为目标函数，即总距离；p 为允许开放的设施数目；x_{ij} 表示客户 j 是否由设施 i 服务，如果客户 j 由设施 i 服务，则 x_{ij}=1，否则，x_{ij}=0；y_i 表示设施 i 是否开放，如果设施 i 开放，则 y_i=1，否则，y_i=0。

2　基本蝙蝠算法

蝙蝠是一种神奇的动物，它靠自身的回声定位系统来探测猎物，避免障碍物，在黑暗中找到它们的栖息地，而 BA 是一种基于蝙蝠的回声定位系统而产生的智能启发式算法，将蝙蝠映射为问题空间中的可行解，将搜索和优化的过程模拟成蝙蝠寻找猎物的过程，利用适应度函数值的大小来衡量蝙蝠所处位置的优劣，将蝙蝠的优胜劣汰过程类比为搜索和优化过程中用好的可行解替代较差的可行解的迭代过程。

2.1　基本蝙蝠的搜索过程

BA 的搜索过程实质上是通过频率的变化实现对蝙蝠速度的控制，从而更新蝙蝠的位置。假设在 d 维空间中，蝙蝠 i 在 t 时刻下的速度和位置分别为 v_i^t 和 x_i^t，则该蝙蝠通过改变频率 f_i 来调整自身位置与当前所处位置最好的蝙蝠之间的距离，从而更新自身新位置 x_i^{t+1}，向当前求解的最好解靠拢，则蝙蝠 i 在 t+1 时刻下的速度和位置更新公式定义为

$$f_i=f_{\min}+(f_{\max}-f_{\min})\beta \tag{7}$$

$$v_i^{t+1}=v_i^t+\left(x_i^t-x^*\right)f_i \tag{8}$$

$$x_i^{t+1}=x_i^t+v_i^{t+1} \tag{9}$$

式中：$\beta\in[0,1]$是服从均匀分布的随机数；$f_{\min}$ 和 $f_{\max}$ 为频率的最小值和最大值；v_i^t 和 v_i^{t+1} 为 t 和 t+1

时刻蝙蝠 i 的速度；x^*表示当前求得的最好解；x_i^t 和 x_i^{t+1} 分别为 t 和 t+1 时刻蝙蝠 i 的位置。

2.2 基本蝙蝠的优化过程

蝙蝠的优化过程即局部搜索过程，在当前求解的最好解集中选择一个解 x_{old}，那么，每只蝙蝠在该最好解附近按照随机游走法则产生局部新解 x_{new}。

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t \tag{10}$$

式中：ε是属于[−1, 1]的 p 维随机向量；A^t 表示 t 时刻当前代蝙蝠种群音量的平均值。

当蝙蝠找到猎物时，音量就会降低，同时脉冲发生率逐渐提高。具体按照式(11)和式(12)对音量和脉冲发生率进行更新。

$$A_i^{t+1} = \alpha A_i^t \tag{11}$$

$$r_i^{t+1} = r_i^0[1-\exp(-\gamma t)] \tag{12}$$

式中：α 和 γ 是常量；A_i^t 和 A_i^{t+1} 分别为 t 和 t+1 时刻蝙蝠 i 的音量；r_i^0 为初始脉冲发生率；r_i^{t+1} 为 t+1 时刻蝙蝠 i 的脉冲发生率。

3 混合蝙蝠算法

BA 与蚁群算法、粒子群算法等其他群体智能算法类似，是利用蝙蝠的回声定位特性来模拟蝙蝠的捕食猎物行为而建立的仿生算法。它的速度和位置更新步骤有些类似于标准粒子群优化，在一定程度上，BA 可以看作是标准粒子群优化和强化的局部搜索的平衡结合，其平衡受音量和脉冲发生率的控制。BA 的提出是用来求解连续域的函数优化问题，不能直接用来求解组合优化问题，因此，本文根据 PMP 的具体特征和 BA 的基本思想，提出了一种适用于求解 PMP 的混合蝙蝠算法。

3.1 编码方式

PMP 的主要任务是在 m 个候选位置点中选择 p 个位置作为中位点，以这些中位点来服务客户，而仅当每个客户都被分配到离其最近的中位点时才能保证总距离最小，则可认为 PMP 的求解是通过搜索不同组合的 p 个中位点来寻找问题最优解。因而本文采用基于中位数的方式对个体进行编码，每只蝙蝠对应一个 p 维向量，每个维度上的值为[1，m]内的随机整数，该 p 维向量内每一维度上的值都是唯一的，且没有大小顺序。例如，对于一个 m=5，p=3 的 PMP，与蝙蝠 i 对应的编码为 x_i = (5 2 3)，即设施 2，3，5 为中位点。

3.2 相关定义

考虑到 BA 求解离散问题的局限性，为了将 BA 成功运用到 PMP 的求解中，并且有较好的运算效率，根据 PMP 模型的具体特征，设计了针对求解该问题的相关操作算子。

定义 1 速度的更新。假设 t 时刻下蝙蝠 i 的位置为 $x_i^t = (x_{i1}^t, x_{i2}^t, \cdots, x_{ip}^t)$，当前所处位置最好的蝙蝠的位置为 $x^* = (x_1^*, x_2^*, \cdots, x_p^*)$，则蝙蝠 i 的速度 $v_i^{t+1} = x^* - x_i^t$。其中，v_i^{t+1} 是由所有属于 x^* 且不属于 x_i^t 的元素组合而成的集合。当 x^* 和 x_i^t 相同时，则随机产生一个 p 维向量代替其中的 x^*，进行相同的速度更新操作。

定义 2 位置的更新。蝙蝠 i 的位置为 $x_i^{t+1} = Fea(x_i^t + x_i^{t+1})$，可分为两部分。第一部分：$x_i^{t+1'} = x_i^t + v_i^{t+1}$，其中，$x_i^{t+1'}$是由所有属于$x_i^{t+1'}$或$v_i^{t+1}$的元素组合而成的集合；第二部分：$x_i^{t+1} = Fea(x_i^{t+1'})$，其中，$Fea(\,)$函数是可行化函数。由于集合$x_i^{t+1'}$的长度超过 p，因此，需根据适应度值的大小逐次递减该向量中的一个元素，直至集合$x_i^{t+1'}$的长度等于 p，但同时属于 x^*和 x_i^t 的元素应始终保持在 $x_i^{t+1'}$中，不得进行删减操作。

定义 3 局部搜索。基于遗传算法中交叉的思想，随机选择一个已开放的设施 O_1 和另一个未开放的设施 C_1，交换 O_1 和 C_1 的位置，若产生的新解的目标函数值优于当前求得的最好解的目标函数值，则更新当前求得的最好解；否则，保持不变。

例如，假设存在一个 m=5，p=3 的 PMP，距离矩阵

$$\boldsymbol{D} = \begin{bmatrix} 0 & 424 & 569 & 254 & 275 \\ 424 & 0 & 391 & 452 & 298 \\ 569 & 391 & 0 & 167 & 528 \\ 254 & 452 & 167 & 0 & 456 \\ 275 & 298 & 528 & 456 & 0 \end{bmatrix}$$

蝙蝠 i 在当前时刻 t 下的位置为 x_i^t = (3 1 4)，当前所处位置最好的蝙蝠的位置为 x^* = (1 4 5)，则蝙蝠 i 的速度为 v_i^{t+1} = (5)。第一部分所得到的 $x_i^{t+1'}$ = (3 1 4 5)，再按照第二部分进行可行化，因为，元素 1 和 4 同时属于x^*和x_i^t，则应始终保持在$x_i^{t+1'}$中，不能进行删减，可得到 2 种不同的组合$\begin{bmatrix} 1 & 4 & 5 \\ 3 & 1 & 4 \end{bmatrix}$，比较相应的适应度值$\begin{bmatrix} 0+298+167+0+0 \\ 0+391+0+0+275 \end{bmatrix} = \begin{bmatrix} 465 \\ 666 \end{bmatrix}$，其中，适应度值 465 较小，则$x_i^{t+1}$ = (1 4 5)。对当前所处位置最好的蝙蝠x^* = (1 4 5)进行局部搜索时，随机选择一个已开放的设施 O_1=1，一个未开放的

设施 C_1=3，则产生的局部新解 x_{new} = (3 4 5)，计算该新解的适应度值为 552，大于当前所处位置最好的蝙蝠的适应度值 465，则不更新当前的最好蝙蝠。若该适应度值小于当前所处位置最好的蝙蝠的适应度值时，则进行更新。

3.3　混合蝙蝠算法的设计

混合蝙蝠算法的步骤：

步骤 1　各参数初始化。蝙蝠种群大小 K；最大迭代次数 N_{max} (N_iter = 1)；音量 A；脉冲发生率 r；距离矩阵 $\boldsymbol{D}$；适应度函数 $f(x)$。

步骤 2　随机初始化种群，找出当前所处位置最好的蝙蝠 x^*。

步骤 3　按照定义 1 和定义 2 更新当前蝙蝠的速度 v_i^{t+1} 和位置 x_i^{t+1}。

步骤 4　产生随机数 *rand* 1，判定 *rand* 1>r_i 是否成立，如果成立，则根据定义 3 执行局部搜索。

步骤 5　产生随机数 *rand* 2，判定 *rand* 2>A_i 且 $f(x_i) < f(x^*)$是否同时成立，如果成立，则接受这个新解，并更新 A，r 和当前求得的最好解。

步骤 6　判断是否达到预设的最大迭代次数 N_{max} 或时间限制 3 600 s，若达到，则输出当前求得的最好解；否则，N_iter++，转至步骤 3。

4　数值实验

为了验证混合蝙蝠算法的求解性能，在 Intel(R)Core(TM)CPU@3.60 GHz 处理器、7.87 GB 内存、操作系统为 64 位 Windows 10 的实验环境下，应用 Matlab R2016a 编程实现了求解 PMP 的混合蝙蝠算法，对运筹学数据库 OR-Library 中的 40 个 PMP 算例进行仿真实验，并与文献[13]中的遗传算法(genetic algorithm，GA)和文献[14]中的粒子群算法(novel discrete particle swarm optimization，NDPSO)进行对比。求解这些算例的混合蝙蝠算法参数设置情况为：最大迭代次数 N_{max}=100，蝙蝠音量 A=0.9，脉冲发生率 r=0.25，音量衰减系数 a=0.95，脉冲发生率增加系数 g=0.05。同时考虑到测试算例大小的不同和运行时间的限制，根据开放中位点的大小来设置蝙蝠种群的数目，当中位点 p<50 时，种群规模 K 为 15；否则，种群规模为 7。文献[13]通过 C++编程实现了求解 PMP 的遗传算法，其中，最大迭代次数为 150 000，交叉概率为 0.8，变异概率为 0.4，种群规模为 100；文献[14]通过 C 编程实现了粒子群算法，其中，最大迭代次数为 1 000，学习因子为 0.5，种群规模设置为算例中设施数目的 2 倍。

表 1 不仅给出了 40 个 PMP 测试算例相应的设施数和中位数，而且给出了这些算例的已知最优解和应用混合蝙蝠算法 10 次运行测试求得的最好解，并给出最好解与已知最优解之间的 *Gap* 值，*Gap*=(本研究算法所求得的最好解−已知最优解)/已知最优解×100%。其中，GA 和 NDPSO 求得的最好解和 *Gap* 值分别来自文献[13-14]。另外，表 1 中还给出了 HBA 和 NDPSO 的运行时间，由于 GA 的运行时间在原文献中未提及，这里就不加以考虑。除此之外，表 1 的最后还给出了所有算例已知最优解和 3 种算法求得解的平均值、*Gap* 的平均值和运行时间的平均值。

由表 1 可知，HBA 在求解 40 个 PMP 算例时，其中 30 个算例可以达到已知最优解，另外 10 个算例求得的最好解尽管稍逊于已知最优解，但其最大 *Gap* 值仅为 0.654%；而 GA 仅可以求得 7 个算例的最优解，NDPSO 可以求得 16 个算例的最优解。另外，从 3 种算法的平均 *Gap* 值来看，GA 的平均 *Gap*=3.231% 最大，而 HBA 的平均 *Gap*=0.06% 最小，非常接近已知最优解，因此，HBA 在与 GA 和 NDPSO 相比时，该算法在求解精度上具有明显的优势。

为了进一步分析 HBA 在求解 PMP 时的有效性和计算复杂度，给出了一个解得最优解的算例 pmed14 和另一个解得近优解的算例 pmed15 的求解迭代过程图，如图 1 和图 2 所示。由于算例的规模不同而导致运行迭代次数不同，因此，这里以运行时间来作相关分析。图 1 在迭代到 265 s 左右、图 2 到 1 400 s 左右时，2 个算例的计算结果还在发生改变。但是，从运行总时间来看，HBA 相对于 NDPSO 而言，需要耗用较多的时间，这是本文的不足之处。总的来说，HBA 在求解这些规模不同的 PMP 算例时能够求得较好的实验结果，与已知最优解的误差较小，且绝大多数算例结果都可以达到已知最优解，故进一步验证了蝙蝠算法在求解 PMP 上的可行性与有效性。

5　结　论

针对 P 中位问题的特点以及蝙蝠算法的寻优

表 1 OR-Library 算例运行结果

Tab.1 Operating results of the examples from OR-Library

算例			不同算法求解结果			*Gap*/%			CPU 计算时间/s	
名称	(m, p)	最优解	GA	NDPSO	HBA	GA	NDPSO	HBA	NDPSO	HBA
pmed1	(100, 5)	5 819	5 819	5 819	5 819	0	0	0	0.56	10.407
pmed2	(100, 10)	4 093	4 105	4 099	4 093	0.293	0.147	0	21.60	35.425
pmed3	(100, 10)	4 250	4 254	4 250	4 250	0.094	0	0	2.02	33.300
pmed4	(100, 20)	3 034	3 063	3 034	3 034	0.956	0	0	11.84	23.267
pmed5	(100, 33)	1 355	1 392	1 356.5	1 355	2.731	0.111	0	33.79	83.100
pmed6	(200, 5)	7 824	7 824	7 824	7 824	0	0	0	1.37	21.205
pmed7	(200, 10)	5 631	5 642	5 631	5 631	0.195	0	0	18.72	75.837
pmed8	(200, 20)	4 445	4 565	4 445	4 445	2.700	0	0	21.06	44.709
pmed9	(200, 40)	2 734	2 871	2 740.5	2 734	5.011	0.238	0	119.44	114.975
pmed10	(200, 67)	1 255	1 364	1 256.5	1 255	8.685	0.120	0	105.80	892.650
pmed11	(300, 5)	7 696	7 702	7 696	7 696	0.078	0	0	2.04	35.846
pmed12	(300, 10)	6 634	6 683	6 634	6 634	0.739	0	0	5.13	121.260
pmed13	(300, 30)	4 374	4 555	4 374	4 374	4.138	0	0	42.93	919.253
pmed14	(300, 60)	2 968	3 151	2 972.9	2 968	6.166	0.165	0	144.05	343.853
pmed15	(300, 100)	1 729	1 858	1 733.7	1 730	7.461	0.272	0.058	150.24	1 685.085
pmed16	(400, 5)	8 162	8 162	8 162	8 162	0	0	0	6.36	58.613
pmed17	(400, 10)	6 999	7 037	7 000.2	6 999	0.543	0.017	0	57.98	213.733
pmed18	(400, 40)	4 809	4 944	4 817.5	4 809	2.807	0.177	0	150.48	930.483
pmed19	(400, 80)	2 845	3 020	2 863.7	2 846	6.151	0.657	0.035	158.84	3 683.183
pmed20	(400, 133)	1 789	1 982	1 805.4	1 789	10.788	0.917	0	293.66	3 691.753
pmed21	(500, 5)	9 138	9 192	9 138	9 138	0.591	0	0	4.24	81.148
pmed22	(500, 10)	8 579	8 653	8 579	8 579	0.863	0	0	33.00	269.001
pmed23	(500, 50)	4 619	4 797	4 650.6	4 619	3.854	0.684	0	155.87	3 148.572
pmed24	(500, 100)	2 961	3 155	2 989.6	2 965	6.552	0.966	0.135	394.75	3 648.983
pmed25	(500, 167)	1 828	2 028	1 845.3	1 835	10.941	0.946	0.383	727.81	3 783.171
pmed26	(600, 5)	9 917	9 917	9 917	9 917	0	0	0	20.79	96.876
pmed27	(600, 10)	8 307	8 343	8 307.9	8 307	0.433	0.011	0	74.44	200.144
pmed28	(600, 60)	4 498	4 636	4 539.6	4 499	3.068	0.925	0.022	260.62	3 589.889
pmed29	(600, 120)	3 033	3 253	3 062	3 044	7.254	0.956	0.363	708.85	3 814.693
pmed30	(600, 200)	1 989	2 231	2 007.5	2 002	12.167	0.930	0.654	1 972.30	3 813.389
pmed31	(700, 5)	10 086	10 086	10 086.1	10 086	0	0.001	0	53.87	154.176
pmed32	(700, 10)	9 297	9 394	9 297.4	9 297	1.043	0.004	0	77.96	463.450
pmed33	(700, 70)	4 700	4 910	4 744	4 700	4.468	0.936	0	451.27	3 638.058
pmed34	(700, 140)	3 013	3 250	3 042.2	3 025	7.866	0.969	0.398	1 303.58	3 662.543
pmed35	(800, 5)	10 400	10 400	10 400	10 400	0	0	0	25.38	194.359
pmed36	(800, 10)	9 934	9 980	9 945.5	9 934	0.463	0.116	0	113.79	625.870
pmed37	(800, 80)	5 057	5 317	5 104.7	5 062	5.141	0.943	0.099	948.08	3 634.414
pmed38	(900, 5)	11 060	11 060	11 060	11 060	0	0	0	29.19	185.163
pmed39	(900, 10)	9 423	9 471	9 423	9 423	0.509	0	0	111.10	774.579
pmed40	(900, 90)	5 128	5 358	5 177.8	5 140	4.485	0.971	0.234	1 393.01	3 747.834
平均值		5 535.3	5 635.6	5 545.8	5 537	3.231	0.304	0.06	255.2	1 313.606

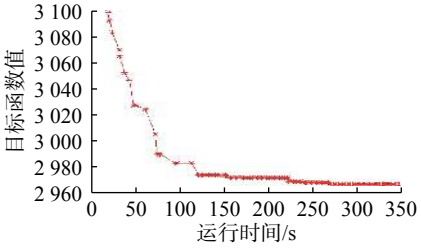

图 1 HBA 求解 pmed14 的最优迭代图

Fig. 1 Optimal iterative graph of pmed14 solved by HBA

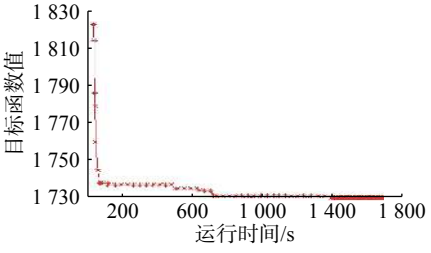

图 2 HBA 求解 pmed15 的最优迭代图

Fig. 2 Optimal iterative graph of pmed15 solved by HBA

机制，重新定义了原有蝙蝠算法的操作算子，并提出了一种可行化函数。同时在局部搜索部分引入了遗传算法的交叉思想，设计了一种针对求解该问题的局部搜索方式，提出了求解该问题的混合蝙蝠算法。

实验部分求解多个测试算例，并将混合蝙蝠算法的计算结果与文献中遗传算法和粒子群算法的计算结果进行对比，测试结果表明，混合蝙蝠算法在确保种群多样性的基础上，具有较好的全局优化性能，能够有效求解 P 中位问题，验证了该算法在求解质量上的优势。但是，与其他改进的智能算法相比，本文提出的混合蝙蝠算法在求解速度上还存在明显的不足，这将是进一步的研究方向。

参考文献：

[1] HAKIMI S L. Optimum locations of switching centers and the absolute centers and medians of a graph[J]. Operations Research, 1964, 12(3): 450–459.

[2] HAKIMI S L. Optimum distribution of switching centers in a communication network and some related graph theoretic problems[J]. Operations Research, 1965, 13(3): 462–475.

[3] HRIBAR M, DASKIN M S. A dynamic programming heuristic for the p-median problem[J]. European Journal of Operational Research, 1997, 101(3): 499–508.

[4] SENNE E L F, LORENA L A N. Lagrangean/surrogate heuristics for p-median problems[M]//LAGUNA M, GONZÁLEZ VELARDE J L. Computing Tools for Modeling, Optimization and Simulation. Boston: Springer, 2000: 115-130.

[5] BELTRAN C, TADONKI C, VIAL J P. Solving the p-median problem with a semi-lagrangian relaxation[J]. Computational Optimization and Applications, 2006, 35(2): 239–260.

[6] GAREY M R, JOHNSON D S. Computers and intractability: a guide to the theory of NP-completeness[M]. San Francisco: W. H. Freeman, 1979: 45-76.

[7] YURI K, TATYANA L, EKATERINA A, et al. Large neighborhood local search for the p-median problem[J]. Yugoslav Journal of Operations Research, 2005, 15(1): 53–63.

[8] RESENDE M G C, WERNECK R F F. On the implementation of a swap-based local search procedure for the p-median problem[C]//Proceedings of the 5th Workshop on Algorithm Engineering and Experiments. Baltimore: SIAM, 2003: 119-127.

[9] ANTAMOSHKIN A N, KAZAKOVTSEV L A. Random search algorithm for the p-median problem[J]. Informatica, 2013, 37(3): 267–278.

[10] AL-KHEDHAIRI A. Simulated annealing metaheuristic for solving p-median problem[J]. International Journal of Contemporary Mathematical Sciences, 2008, 3(25/28): 1357–1365.

[11] ALP O, ERKUT E, DREZNER Z. An efficient genetic algorithm for the p-median problem[J]. Annals of Operations Research, 2003, 122(1/4): 21–42.

[12] LI X, XIAO N C, CLARAMUNT C, et al. Initialization strategies to enhancing the performance of genetic algorithms for the p-median problem[J]. Computers & Industrial Engineering, 2011, 61(4): 1024–1034.

[13] KRÖMER P, PLATOŠ J. New genetic algorithm for the p-median problem[M]//PAN J S, SNASEL V, CORCHADO E S, et al. Intelligent Data analysis and Its Applications, Volume II. Cham: Springer, 2014: 35-44.

[14] SEVKLI M, MAMEDSAIDOV R, CAMCI F. A novel discrete particle swarm optimization for p-median problem[J]. Journal of King Saud University:Engineering Sciences, 2014, 26(1): 11–19.

[15] YANG X S. A new metaheuristic bat-inspired algorithm[M]//GONZÁLEZ J R, PELTA D A, CRUZ C, et al. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Berlin, Heidelberg: Springer, 2010: 65-74.

[16] NAKAMURA R Y M, PEREIRA L A M, COSTA K A, et al. BBA: a binary bat algorithm for feature selection[C]// Proceedings of the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images. Ouro Preto, Brazil: IEEE, 2012: 291-297.

[17] 李枝勇, 马良, 张惠珍. 0-1 规划问题的元胞蝙蝠算法[J]. 计算机应用研究, 2013, 30(10): 2903–2906.

[18] RIZK-ALLAH R M, HASSANIEN A E. New binary bat algorithm for solving 0-1 knapsack problem[J]. Complex & Intelligent Systems, 2018, 4(1): 31–53.

（编辑：石　瑛）