**Computer Science and Application** 计算机科学与应用, 2020, 10(4), 760-766
Published Online April 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.104079

Hans汉斯

# A Hybrid Particle Swarm Optimization for Solving Nash Equilibrium

**Huaqin Li**

College of Science, Guizhou Institute of Technology, Guiyang Guizhou
Email: lhq305@163.com

Received: Apr. 4th, 2020; accepted: Apr. 19th, 2020; published: Apr. 26th, 2020

## Abstract

**At present, many intelligent algorithms have been used for solving the Nash equilibrium. They have their own advantages and disadvantages. To overcome local optimum, it designs a hybrid Particle Swarm Optimization for solving Nash equilibrium, by combining the crossover operator in Genetic Algorithm with the basic Particle Swarm Optimization. Experiments show that the algorithm designed is effective, and it is superior to the immune algorithm, immune Particle Swarm Optimization, and basic Particle Swarm Optimization.**

## Keywords

**Nash Equilibrium, Particle Swarm Optimization, Crossover, Hybrid**

# 一种混合粒子群算法求解Nash平衡

**黎华琴**

贵州理工学院理学院，贵州 贵阳
Email: lhq305@163.com

收稿日期：2020年4月4日；录用日期：2020年4月19日；发布日期：2020年4月26日

## 摘 要

**目前，已有多种智能算法应用于求解Nash平衡，这些算法各有优缺点，为避免粒子群算法在求解Nash平衡时陷入局部最优，本文将遗传算法中的杂交算子引入基本粒子群算法中，设计了一个求解博弈Nash平衡的混合粒子群算法。实验表明，设计的算法具有较好的性能，优于免疫算法，免疫粒子群算法与基本粒子群算法。**

文章引用: 黎华琴. 一种混合粒子群算法求解 Nash 平衡[J]. 计算机科学与应用, 2020, 10(4): 760-766.
DOI: 10.12677/csa.2020.104079

## 关键词

**Nash平衡，粒子群算法，杂交，混合**



## 1. 引言

粒子群优化(Particle Swarm Optimization, PSO)算法是由 Kennedy 和 Eberhart 于 1995 年率先提出的，源于对鸟群或鱼群捕食过程的模拟。目前，粒子群优化算法已经在多个领域中得到应用[1] [2] [3] [4] [5]。

Nash 证明了 $n$ 人非合作博弈 Nash 平衡的存在性[6]，但迄今未有通用算法来求解所有的博弈模型。Nash 平衡的求解，最终可归结为求解一个约束优化问题，要得到此优化问题的准确解通常是很难的。然而在许多实际问题中，我们必须得到此优化问题的一个具体的解。一般采用某些数值方法求其近似，但数值方法通常要求可导、连续等条件。另一类方法是采用智能算法，这些算法直接对结构对象进行操作，不要求可导、连续等条件，它们具有内在的并行性和较好的寻优能力，可以自动调整搜索方向，从而较好地得到优化问题的解。目前，已有多种智能算法用来求解 Nash 平衡，包括遗传算法[7]、启发式搜索算法[8]、免疫算法[9]、粒子群算法[5]及一些改进的粒子群算法[10] [11]等。

注意到一方面，求解优化问题时，智能算法后期收敛速度较慢，且容易陷入局部最优；另一方面，粒子群优化算法容易实现，将之作为博弈演化模型，能够比较合理地解释局中人的行为，即局中人是理性的，其目标是最大化自己的收益。本文以粒子群算法为基础，引入遗传算法的杂交算子，设计了一个求解 Nash 平衡的混合粒子群优化算法。实验表明，本文算法的收敛速度优于目前已有的一些算法。

## 2. 问题描述

设在一个 $n$ 人非合作博弈 $G$ 中，局中人 $i(1\le i\le n)$ 的纯策略集 $S_i=\{s_1^i,s_2^i,\cdots,s_{m_i}^i\}$，其中，$m_i$ 是局中人 $i$ 的纯策略的个数，局中人 $i$ 的支付函数为 $P_i$。称 $\boldsymbol{x}^i=(x_1^i,x_2^i,\cdots,x_{m_i}^i)$ 为局中人 $i$ 的一个混合策略，其中 $\boldsymbol{x}^i$ 满足 $x_k^i\ge 0(1\le k\le m_i)$，$x_1^i+x_2^i+\cdots+x_{m_i}^i=1$，即纯策略集 $S_i$ 上的一个概率分布。记局中人 $i$ 的混合策略集记为 $X_i$。博弈的混合策略组合可记为 $\boldsymbol{x}=(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^n)$。在此混合策略组合下，局中人 $i(1\le i\le n)$ 的期望支付 $E_i(x)=\sum_{j_1=1}^{m_1}\sum_{j_2=1}^{m_2}\cdots\sum_{j_n=1}^{m_n}P_i(s_{j_1}^1,s_{j_2}^2,\cdots,s_{j_n}^n)\cdot x_{j_1}^1x_{j_2}^2\cdots x_{j_n}^n$。

**定义 1** [12] 设 $\boldsymbol{x}^*=(\boldsymbol{x}^{1*},\boldsymbol{x}^{2*},\cdots,\boldsymbol{x}^{n*})$ 是 $n$ 人非合作博弈 $G$ 的一个混合策略组合，如果对任意 $i(1\le i\le n)$ 及 $\boldsymbol{x}^i\in X_i$，均有 $E_i(\boldsymbol{x}^*\|\boldsymbol{x}^i)\le E_i(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}^*$ 是博弈 $G$ 的一个混合策略 Nash 平衡，$\{E_i(x^*)\}$ 为对应的平衡结果。这里 $E_i(\boldsymbol{x}^*\|\boldsymbol{x}^i)$ 是将局中人 $i$ 的混合策略 $\boldsymbol{x}^{i*}$ 换成混合策略 $\boldsymbol{x}^i$ 后的期望支付。

注意到局中人 $i$ 的纯策略 $s_k^i\in S_i$ 等同于混合策略 $(0,\cdots,0,1,0,\cdots,0)$，其中第 $k$ 个分量为 1，其余分量为 0。Nash 平衡具备重要性质[5]：$\boldsymbol{x}^*$ 是 $n$ 人非合作博弈 $G$ 的一个混合策略 Nash 平衡的充要条件是：对于每个局中人 $i$ 及每个纯策略 $s_k^i\in S_i$，有 $E_i(\boldsymbol{x}^*\|s_k^i)\le E_i(\boldsymbol{x}^*)$。

特别地，对双矩阵博弈 $\Gamma(\boldsymbol{x},\boldsymbol{y};A,B)$ (简记 $\Gamma(A,B)$)，其中，$\boldsymbol{x},\boldsymbol{y}$ 分别为局中人 1 与局中人 2 的混合策略，$A,B$ 是 $m\times n$ 维支付矩阵，即 $\boldsymbol{x}A\boldsymbol{y}^{\mathrm{T}}$ 与 $\boldsymbol{x}B\boldsymbol{y}^{\mathrm{T}}$ 分别为局中人 1 采取混合策略 $\boldsymbol{x}$、局中人 2 采取混合策

略 $\boldsymbol{y}$ 时，局中人 1 与局中人 2 的期望支付。

本节余下内容，我们将求解 Nash 平衡问题转化为优化问题。

我们给每个混合策略组合赋予一个适应度。对混合策略组合 $\boldsymbol{x}=\left(\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^n\right)$，根据 Nash 平衡的定义与性质，定义 $\boldsymbol{x}$ 的适应度 $f\left(\boldsymbol{x}\right)=\sum_{i=1}^{n}\max_{k=1,2,\cdots,m_i}\left\{E_i\left(\boldsymbol{x}\,\|\,s_k^i\right)-E_i\left(\boldsymbol{x}\right),0\right\}$，这里 $\max_{k=1,2,\cdots,m_i}\left\{E_i\left(\boldsymbol{x}\,\|\,s_k^i\right)-E_i\left(\boldsymbol{x}\right),0\right\}$ 为局中人 $i$ 关于混合策略组合 $\boldsymbol{x}$ 的适应度，$f\left(\boldsymbol{x}\right)$ 为所有局中人关于混合策略组合 $\boldsymbol{x}$ 的适应度之和。由 Nash 平衡的性质，混合策略组合 $\boldsymbol{x}$ 为纳什均衡解当且仅当适应度函数在 $\boldsymbol{x}$ 处取得最小值 0。故博弈的混合组合空间内只有 Nash 平衡点的适应度最小。

特别地，对双矩阵博弈 $\Gamma\left(\boldsymbol{x},\boldsymbol{y};A,B\right)$，其中 $A,B$ 是 $m\times n$ 维支付矩阵， 混合策略组合 $\left(\boldsymbol{x},\boldsymbol{y}\right)$ 的适应度 $f\left(\boldsymbol{x},\boldsymbol{y}\right)$ 如下：

$$f\left(\boldsymbol{x},\boldsymbol{y}\right)=\max\left\{\max_{1\le i\le m}\left\{A_i\boldsymbol{y}^{\mathrm{T}}-\boldsymbol{x}A\boldsymbol{y}^{\mathrm{T}}\right\},0\right\}+\max\left\{\max_{1\le i\le n}\left\{\boldsymbol{x}B_j-\boldsymbol{x}B\boldsymbol{y}^{\mathrm{T}}\right\},0\right\}$$

其中，$A_i$ 为矩阵 $A$ 的第 $i$ 行所对应的向量，$B_j$ 为矩阵 $B$ 的第 $j$ 列所对应的向量。

## 3. 混合粒子群算法设计

在粒子群算法中，优化问题的任一可行解都是搜索空间中的一个粒子，每个粒子都有一个由目标函数所确定的适应度。粒子群算法首先初始化一群随机粒子(包括初始位置及速度)，然后这些粒子追随当前的最优粒子在搜索空间中随机搜索，最后通过迭代找到近似最优解[13]。

每个粒子在搜索时，根据自己的速度、自己搜索到的历史最好点与群体内其他粒子的历史最好点，进行位置与速度的更新。记 $x_i$ 为第 $i$ 个粒子的位置，$v_i$ 为第 $i$ 个粒子的速度，$p_i$ 为第 $i$ 个粒子经过的最好点，$p_g$ 为群体内所有的粒子经过的最好点。一般地，粒子的位置和速度都在连续的实数空间内取值。在粒子群算法的迭代中，第 $i$ 个粒子的速度与位置的更新公式如下：

$$v_i^{k+1}=\omega v_i^k+c_1r_1\left(p_i-x_i^k\right)+c_2r_2\left(p_g-x_i^k\right) \tag{1}$$

$$x_i^{k+1}=x_i^k+v_i^{k+1} \tag{2}$$

其中，$v_i^k$ 是第 $i$ 个粒子在第 $k$ 次迭代时的速度，$x_i^k$ 是第 $i$ 个粒子在第 $k$ 次迭代时的位置。$\omega$ 是惯性权重，大小决定了对粒子当前速度继承的多少；$r_1,r_2$ 是 $[0,1]$ 区间内的随机数，$c_1,c_2$ 是学习因子。粒子的速度被限制在一个最大速度 $V_{\max}$ 的范围内。

杂交(交叉)是遗传算法中的一个非常重要的遗传算子，它同时对两个染色体进行操作，组合二者的特性产生新的后代。遗传算法的性能在很大程度上取决于采用的杂交运算的性能。可将杂交算子引入粒子群算法中，基于这一思想，我们设计了一个求解博弈 Nash 平衡的**混合粒子群算法**。

在算法中，粒子就是一个混合策略组合，算法如下：

**Step 1** 确定学习因子 $c_1,c_2$，群体规模 $N$，最大迭代代数 $k_{\max}$，$\omega_{\max}$，$\omega_{\min}$ 及精确度 $\epsilon$。

**Step 2** 随机初始化粒子群及其速度，使每个粒子的第 $i$ 个向量满足

$$x_1^i+x_2^i+\cdots+x_{m_i}^i=1,\ x_j^i\ge 0,\ j=1,2,\cdots,m_i,i=1,2,\cdots,N$$

每个粒子速度的第 $i$ 个向量满足 $v_1^i+v_2^i+\cdots+v_{m_i}^i=0,i=1,2,\cdots,N$。

**Step 3** 计算各个粒子的适应度，并对粒子按照适应度大小进行排序，适应度小的一半粒子直接进入下一代 $\mathrm{NextG}_1$ (另一半记为 $\mathrm{NextG}_2$)，对适应度大的一半粒子进行杂交，杂交之后先对粒子作归一化处理，使之满足 $x_1^i+x_2^i+\cdots+x_{m_i}^i=1,x_j^i\ge 0,j=1,2,\cdots,m_i,i=1,2,\cdots,N$；并与 $\mathrm{NextG}_2$ 形成粒子池，计算粒子池

中粒子的适应度，选择适应度小的一半粒子与 $\text{NextG}_1$ 形成下一代粒子 NextG 。

**Step 4** 计算当前粒子群 $P_k$ 中粒子的适应度值及 $p_i\left(i=1,\cdots,N\right)$，并把 $p_g$ 作为记忆粒子存入记忆库中。

**Step 5** 判断是否满足 $f\left(p_g\right)<\epsilon$ 或 $k\geq k_{\max}$。 若满足，则停止并输出迭代次数 $k$, $p_g$ 以及 $p_g$ 的适应值; 否则，继续。

**Step 6** 利用公式 $\omega=\omega_{\max}-k\cdot\dfrac{\omega_{\max}-\omega_{\min}}{k_{\max}}$ 计算惯性权重 $\omega$ 。

**Step 7** 按公式(1)与(2)更新 $P_k$ 中粒子的速度与位置。

**Step 8** 依次检验第 $i$ 个粒子 $x_i^{k+1}$ 是否属于 $x_i^{k+1}>0$。否则，计算控制步长 $\alpha_i$ (此技术来自文献[11])，并令 $x_i^{k+1}=x_i^k+\alpha_i\cdot v_i^{k+1}$，然后作归一化处理，使得 $x_i^{k+1}$ 回到混合策略组合空间中，形成新一代粒子群体 $P_{k+1}$。然后转到第 3 步。

关于算法的性能，本文以算法获得 Nash 平衡或其近似解的平均迭代次数来度量。

## 4. 数值例子

本文用文献[9] [10]中给出的 4 个经典的 $2\times2$ 博弈和一个 $3\times2$ 博弈(例 1~5)以及文献[7] [10] [11]中共同给出的一个 $3\times3$ 博弈(例 6)作为算例，用本文给出的混合粒子群算法求解这些博弈。

**例 1** 囚徒困境博弈 $\Gamma\left(A_1,B_1\right)$，其中，$A_1=\begin{pmatrix}-8 & 0\\ -15 & -1\end{pmatrix}$，$B_1=\begin{pmatrix}-8 & -15\\ 0 & -1\end{pmatrix}$；

**例 2** 智猪博弈 $\Gamma\left(A_2,B_2\right)$，其中，$A_2=\begin{pmatrix}1.5 & -0.5\\ 5 & 0\end{pmatrix}$，$B_2=\begin{pmatrix}3.5 & 6\\ 0.5 & 0\end{pmatrix}$；

**例 3** 猜谜博弈 $\Gamma\left(A_3,B_3\right)$，其中，$A_3=\begin{pmatrix}1 & -1\\ -1 & 1\end{pmatrix}$，$B_3=\begin{pmatrix}-1 & 1\\ 1 & -1\end{pmatrix}$；

**例 4** 监察博弈 $\Gamma\left(A_4,B_4\right)$，其中，$A_4=\begin{pmatrix}0 & 50\\ 30 & 30\end{pmatrix}$，$B_4=\begin{pmatrix}-10 & -50\\ 60 & 70\end{pmatrix}$；

**例 5** 博弈 $\Gamma\left(A_5,B_5\right)$，其中，$A_5=\begin{pmatrix}4 & 6\\ 2 & 3\\ 3 & 2\end{pmatrix}$，$B_5=\begin{pmatrix}3 & 2\\ 1 & 6\\ 0 & 8\end{pmatrix}$；

**例 6** 博弈 $\Gamma\left(A_6,B_6\right)$，其中，$A_6=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$，$B_6=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{pmatrix}$。此博弈具唯一解 $\left(\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right),\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)\right)$。

利用本文设计的混合粒子群算法求解上述博弈，例 1~5 中算法的参数设置为：群体规模 $N=10$，学习因子 $c_1=1.4962, c_2=1.4962$，最大迭代次数为 1000，精度设置为 $\epsilon=10^{-2}$，其计算结果如表 1~5 所示。例 6 中精度设置为 $\epsilon=10^{-6}$，其余参数不变，计算结果如表 6。

**Table 1.** Computing result of game $\Gamma\left(A_1,B_1\right)$

**表 1.** 博弈 $\Gamma\left(A_1,B_1\right)$ 的计算结果

| 计算次数 | 迭代次数 | 局中人 1 混合策略 | 局中人 2 混合策略 | 最优适应值 |
|---|---|---|---|---|
| 1 | 1 | (1, 0) | (1, 0) | 0 |
| 2 | 3 | (1, 0) | (1, 0) | 0 |
| 3 | 2 | (1, 0) | (1, 0) | 0 |
| 4 | 2 | (1, 0) | (1, 0) | 0 |
| 5 | 1 | (1, 0) | (1, 0) | 0 |

**Table 2.** Computing result of game $\Gamma(A_2, B_2)$
**表 2.** 博弈 $\Gamma(A_2, B_2)$ 的计算结果

| 计算次数 | 迭代次数 | 局中人 1 混合策略 | 局中人 2 混合策略 | 最优适应值 |
|---|---|---|---|---|
| 1 | 2 | (0.0001, 0.9999) | (1.0000, 0) | 3.2675e−004 |
| 2 | 2 | (0.0025, 0.9975) | (1.0000, 0) | 0.0087 |
| 3 | 2 | (0, 1.0000) | (0.9830, 0.0170) | 0.0085 |
| 4 | 4 | (0, 1.0000) | (0.9963, 0.0037) | 0.0018 |
| 5 | 2 | (0, 1.0000) | (0.9831, 0.0169) | 0.0085 |

**Table 3.** Computing result of game $\Gamma(A_3, B_3)$
**表 3.** 博弈 $\Gamma(A_3, B_3)$ 的计算结果

| 计算次数 | 迭代次数 | 局中人 1 混合策略 | 局中人 2 混合策略 | 最优适应值 |
|---|---|---|---|---|
| 1 | 3 | (0.4993, 0.5007) | (0.5011, 0.4989) | 0.0036 |
| 2 | 3 | (0.5018, 0.4982) | (0.5011, 0.4989) | 0.0059 |
| 3 | 2 | (0.4986, 0.5014) | (0.4997, 0.5003) | 0.0035 |
| 4 | 1 | (0.5003, 0.4997) | (0.5026, 0.4974) | 0.0058 |
| 5 | 1 | (0.4995, 0.5005) | (0.5033, 0.4967) | 0.0074 |

**Table 4.** Computing result of game $\Gamma(A_4, B_4)$
**表 4.** 博弈 $\Gamma(A_4, B_4)$ 的计算结果

| 计算次数 | 迭代次数 | 局中人 1 混合策略 | 局中人 2 混合策略 | 最优适应值 |
|---|---|---|---|---|
| 1 | 8 | (0.2000, 0.8000) | (0.4005, 0.5995) | 0.0066 |
| 2 | 5 | (0.2000, 0.8000) | (0.4009, 0.5991) | 0.0090 |
| 3 | 8 | (0.1998, 0.8002) | (0.4001, 0.5999) | 0.0047 |
| 4 | 6 | (0.2002, 0.7998) | (0.4002, 0.5998) | 0.0095 |
| 5 | 6 | (0.2001, 0.7999) | (0.4007, 0.5993) | 0.0086 |

**Table 5.** Computing result of game $\Gamma(A_5, B_5)$
**表 5.** 博弈 $\Gamma(A_5, B_5)$ 的计算结果

| 计算次数 | 迭代次数 | 局中人 1 混合策略 | 局中人 2 混合策略 | 最优适应值 |
|---|---|---|---|---|
| 1 | 8 | (1, 0, 0) | (1, 0) | 0 |
| 2 | 6 | (1, 0, 0) | (1, 0) | 0 |
| 3 | 9 | (1, 0, 0) | (1, 0) | 0 |
| 4 | 12 | (0.9958, 0.0022, 0.0020) | (1.0000, 0) | 0.0065 |
| 5 | 10 | (0.9975, 0.0024, 0.0001) | (1.0000, 0) | 0.0048 |

**Table 6.** Computing result of game $\Gamma(A_6, B_6)$
**表 6.** 博弈 $\Gamma(A_6, B_6)$ 的计算结果

| 计算次数 | 迭代次数 | 局中人 1 混合策略 | 局中人 2 混合策略 | 最优适应值 |
|---|---|---|---|---|
| 1 | 43 | (0.3333, 0.3333, 0.3333) | (0.3333, 0.3333, 0.3333) | 9.4477e−007 |
| 2 | 48 | (0.3333, 0.3333, 0.3333) | (0.3333, 0.3333, 0.3333) | 8.4302e−007 |
| 3 | 50 | (0.3333, 0.3333, 0.3333) | (0.3333, 0.3333, 0.3333) | 9.8715e−007 |
| 4 | 46 | (0.3333, 0.3333, 0.3333) | (0.3333, 0.3333, 0.3333) | 8.0621e−007 |
| 5 | 42 | (0.3333, 0.3333, 0.3333) | (0.3333, 0.3333, 0.3333) | 9.7314e−007 |

考虑算法的性能，即获得近似解的平均迭代次数。通过计算实验可知，用本文算法在适应函数精度 $\epsilon = 10^{-2}$ 和群体规模 $N = 10$ 的情况下，例 1~5 的 5 次计算结果分别需要平均迭代 2 代、3 代、2 代、7 代和 9 代。在平均迭代次数方面，优于文献[10]用免疫粒子群算法给出的结果(文献[10]中例 1~5 在适应函数精度 $\epsilon = 10^{-1}$ 的情形下，5 次计算结果需要平均迭代 7 代、3 代、4 代、14 代和 15 代)，特别是例 6(本算法在适应度函数精度 $\epsilon = 10^{-6}$)情况下，5 次的计算结果需要平均迭代为 46，文献[10]在适应度函数精度为 $\epsilon = 10^{-4}$，5 次的计算结果需要平均迭 288 代)，本算法有较大的改进，在计算过程中运行速度非常快。当然，本算法也优于[9]的算法。

## 5. 结束语

本文算法的目标在于找到博弈的一个 Nash 平衡或其近似解。但一个博弈中，可能存在多个 Nash 平衡，究竟哪个平衡才是最终的博弈结果，本文的算法并没有作这方面的考虑。进一步工作中，我们希望能够引入一些机制，使得算法找到的平衡点更接近与现实的博弈结果。此外，实验还表明，当局中人纯策略数增大时，算法的迭代次数呈指数增长。

## 致　谢

感谢匿名审稿人对本文初稿提出的宝贵意见。

## 基金项目

贵州省教育厅青年科技人才成长项目(黔教合 KY 字[2017]225)。

## 参考文献

[1] 卿东升, 邓巧玲, 李建军, 刘帅, 刘鑫, 曾素平. 基于粒子群算法的满载需求可拆分车辆路径规划[J]. 控制与决策, 2020, 35(2): 1-9.

[2] 梁静, 葛士磊, 瞿博阳, 于坤杰. 求解电力系统经济调度问题的改进粒子群优化算法[J]. 控制与决策, 2019, 34(3): 1-10.

[3] 唐红亮, 吴柏林, 胡旺, 康承旭. 基于粒子群优化的地震应急物资多目标调度算法[J]. 电子与信息学报, 2020, 42(3): 737-745.

[4] 李倩. 粒子群优化算法在港口船舶物流中的应用[J]. 舰船科学技术, 2020, 42(2): 205-207.

[5] 余谦, 王先甲. 基于粒子群优化求解纳什均衡的演化算法[J]. 武汉大学学报, 2006, 52(1): 25-29.

[6] Nash, J. (1951) Non-Cooperative Games. *The Annals of Mathematics*, **54**, 286-295. https://doi.org/10.2307/1969529

[7] 陈士俊, 孙永广, 吴宗鑫. 一种求解 Nash 均衡解的遗传算法[J]. 系统工程, 2001(19): 67-70.

[8] 隗立涛, 修乃华. 基于启发搜索算法的纳什均衡计算[J]. 北京交通大学学报, 2001, 40(33): 87-92.

[9] 邱中华, 高洁, 朱跃星. 应用免疫算法求解博弈问题[J]. 系统工程学报, 2006, 21(4): 398-404.

[10] 刘露萍, 贾文生, 蔡江华. 协同免疫量子粒子群算法求非合作博弈 Nash 平衡[J]. 计算机应用于软件, 2019(8): 203-209.

[11] 贾文生, 向淑文, 杨剑锋, 等. 基于免疫粒子群算法的非合作博弈Nash均衡问题求解[J]. 计算机应用研究, 2012, 29(1): 28-31.

[12] 汪贤裕, 肖玉明. 博弈论及其应用[M]. 北京: 科学出版社, 2008: 15-48.

[13] 汪定伟, 王俊伟, 王洪峰, 等. 智能优化方法[M]. 北京: 高等教育出版社, 2007: 217-252.