智能系统学报
CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS

求解连续型分布式约束优化问题的自适应多点交叉遗传算法

廖鑫, 石美凤, 陈媛

引用本文:
廖鑫,石美凤,陈媛. 求解连续型分布式约束优化问题的自适应多点交叉遗传算法[J]. 智能系统学报, 2023, 18(4): 793–802.
LIAO Xin,SHI Meifeng,CHEN Yuan. Adaptive multi–point crossover genetic algorithm for solving continuous distributed constraint optimization problems[J]. *CAAI Transactions on Intelligent Systems*, 2023, 18(4): 793–802.

在线阅读 View online: https://dx.doi.org/10.11992/tis.202202017

DOI: 10.11992/tis.202202017
网络出版地址: https://kns.cnki.net/kcms/detail/23.1538.TP.20230321.1644.014.html

求解连续型分布式约束优化问题的自适应多点交叉遗传算法

廖鑫, 石美凤, 陈媛
(重庆理工大学 计算机科学与工程学院, 重庆 400000)

摘　要: 针对连续型分布式约束优化问题 (continuous distributed constraint optimization problems, C-DCOPs) 求解算法的 anytime 属性的缺失、约束函数形式的限制和无法保证收敛等局限, 本文提出一种求解 C-DCOP 的自适应多点交叉遗传算法 (adaptive multi-point crossover genetic algorithm based C-DCOP, AMCGA)。在 AMCGA 中, 智能体 (agent) 构建分布式种群和广度优先搜索 (breadth first search, BFS) 伪树以分布式地计算个体适应度; 通过贪婪策略选择精英个体进行自适应多点交叉实现全局搜索, 智能体之间协同通信保证分布式种群中解的一致性; 利用变异算子完成局部搜索。AMCGA 适用于任意形式的约束函数, 并被证明具有任意时间属性和全局收敛性。在 4 类基准问题上的广泛实验结果表明, AMCGA 的求解质量优于最先进的 C-DCOP 求解算法, 能有效地打破目前 C-DCOP 求解算法的局限, 并在求解质量方面存在 20%~30% 的提升。

关键词: 连续型分布式约束优化问题; 任意时间属性; 自适应多点交叉; 遗传算法; 分布式种群; 广度优先搜索伪树; 智能体; 求解质量
中图分类号: TP183　　**文献标志码:** A　　**文章编号:** 1673−4785(2023)04−0793−10

中文引用格式: 廖鑫, 石美凤, 陈媛. 求解连续型分布式约束优化问题的自适应多点交叉遗传算法 [J]. 智能系统学报, 2023, 18(4): 793–802.
英文引用格式: LIAO Xin, SHI Meifeng, CHEN Yuan. Adaptive multi-point crossover genetic algorithm for solving continuous distributed constraint optimization problems[J]. CAAI transactions on intelligent systems, 2023, 18(4): 793–802.

Adaptive multi-point crossover genetic algorithm for solving continuous distributed constraint optimization problems

LIAO Xin, SHI Meifeng, CHEN Yuan
(College of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400000, China)

Abstract: Aiming at the limitations of the solving algorithm for continuous distributed constraint optimization problems (DCOPs), such as the lack of anytime property, the limitation of constraints function form and the inability to guarantee convergence, an adaptive multi-point crossover genetic algorithm for solving C-DCOP (AMCGA) is proposed. In AMCGA, the agent firstly builds distributed population and breadth first search (BFS) pseudo-tree to calculate the individual fitness in a distributed way. Secondly, the agent selects elite individuals through greedy strategy for adaptive multi-point crossover, achieving global search; cooperative communication between agents ensures the consistency of solutions in the distributed population. Finally, the agent uses mutation operator to complete local search. AMCGA is suitable for any form of constraint function and is proved to have anytime property and global convergence. Extensive experimental results on four types of benchmark problems demonstrate that the solution quality of AMCGA is superior to the state-of-the-art C-DCOP solving algorithms, it can effectively break through the limitations of current C-DCOP solving algorithms, and improve the solution quality by 20% to 30%.

Keywords: continuous distributed constraint optimization problems; anytime property; adaptive multi-point crossover; genetic algorithm; distributed population; breadth first search pseudo-tree; agent; solution quality

分布式约束优化问题 (distributed constraint optimization problems, DCOPs)[1] 是解决分布式多智能体系统建模的有效框架。DCOP 由一组智能体构成, 每个智能体控制一组离散变量, 通过与局部的智能体协同通信优化全局目标函数。全局目标函数被定义为约束代价的集合, 其中约束代价由变量所定义。相比于传统的集中式优化, DCOP 强调通过局部交互使得全局最优, 因此 DCOP 具有更强的容错性和更高的并行度[2]。目前 DCOP

收稿日期: 2022−02−22.　**网络出版日期:** 2023−03−22.
基金项目: 重庆市教育委员会科学技术研究计划青年项目 (KJQN202001139); 重庆市基础研究与前沿探索项目 (cstc2018jcyjAX0287); 重庆理工大学研究生创新项目 (clgycx20203110); 重庆理工大学科研启动基金项目 (2019ZD03).
通信作者: 石美凤. E-mail: shimf@cqut.edu.cn.

已经广泛应用于实际生活，如会议调度[3]、微网控制[4]、任务调度[5]、智能家居[6]、传感器网络[7]和资源分配[8]等。DCOP 求解算法主要分为完备算法和非完备算法，完备算法如异步分布式优化(asynchronous distributed optimization, ADOPT)[9]、分布式伪树优化过程(distributed pseudo-tree optimization procedure, DPOP)[10]和基于伪树的前向定界(forward bounding on pseudo-trees, PT-FB)[11]等提供一个全局最优解，但计算开销和内存需求随着问题规模增加而呈现指数级增长；非完备算法如分布式随机算法(distributed stochastic algorithm, DSA)[12]、最大和算法(max-sum algorithm, MS)[13]、最大增益消息(maximal gain message, MGM)[14]和求解 DCOP 的蚁群优化(ant colony optimization for solving DCOP, ACO-DCOP)[15]等在较低的计算开销和内存需求下提供一个近似解。

然而在一些实际应用中，智能体必须取得一个或者多个连续值参数，比如方向、速度或者传感器激活时间等。DCOP 建模连续型变量问题的效果并不理想，因此 Stranders 等[16]提出了连续型 DCOP (continuous DCOP, C-DCOP)来更有效地建模连续型变量应用。与 DCOP 相比，C-DCOP 的变量和值域是连续的，并且约束代价由一组函数所定义。

为了应对 C-DOCP 中变量、值域和约束代价的变化，研究人员提出了连续最大和算法(continuous max-sum, CMS)。在 CMS 中，约束代价函数被近似为分段线性函数，通过对比离散最大和和连续最大和算法，由 CMS 能够得到更高质量的解证明了 C-DCOP 的必要性。然而只有极少数实际应用可以适用于分段线性函数。混合连续最大和(hybrid continuous max-sum, HCMS)[17]首先通过离散最大和获得一组近似解，通过非线性优化方法提高近似解的精度。虽然 HCMS 解决了分段线性函数的局限，但由于存在导数计算，使得 HCMS 不适用于不可微的问题，并且不能保证收敛。精确连续 DPOP (exact continuous DPOP, EC-DPOP)、近似连续 DPOP (approximate continuous DPOP, AC-DPOP)、聚类近似连续 DPOP (clustered AC-DPOP)和连续 DSA (continuous DSA)[18]被同时提出以解决函数形式的限制。EC-DPOP、AC-DPOP 和 CAC-DPOP 能够获得高质量的解，但计算开销和内存需求巨大；C-DSA 逻辑简单，计算开销小，但解的质量不佳。基于 C-DCOP 的粒子群优化算法(particle swarm optimization based C-DCOP, PFD)[19]被提出以减少计算开销和内存需求。PFD 引入种群思想，通过种群寻优的方式找到近似解，然而搜索能力差和容易陷入局部最优的问题限制了解质量的提升。连续协同约束近似(continuous cooperative constraint approximation, C-CoCoA)[20]是一种非迭代算法，不具备 anytime 属性，虽然能够以极快的计算速度和极小的计算开销求解 C-DCOP，但由于 C-CoCoA 中的半贪婪局部搜索策略，使得其在复杂的大规模问题上的求解质量不佳。

基于以上分析，求解 C-DCOP 的自适应多点交叉遗传算法(adaptive multi-point crossover genetic algorithm for C-DCOP, AMCGA)被提出用于解决目前 C-DCOP 求解算法的缺点与局限。为了将遗传算法(genetic algorithm, GA)[21]扩展到 C-DCOP 并进一步改进，本文的贡献概括如下：

1) 采用分布式种群[22]和广度优先搜索(breadth first search, BFS)伪树[23]满足分布式特性，使得算法具有 anytime 属性并适用于任意函数形式。

2) 设计一种自适应的多点交叉方法，增强算法的搜索能力，通过智能体的决策与协同通信保证交叉过程中解的一致性。

3) 设计自适应的种群大小增强算法的鲁棒性并设计自适应的交叉和变异概率平衡算法。

4) 改进的精英筛选与保留策略能够保证算法的收敛。

在 4 类基准问题上的广泛实验表明，AMCGA 的求解质量优于最先进的 C-DCOP 求解算法并有着 20%~30% 的提升。

1 连续型分布式约束优化问题的定义与基础方法

1.1 连续型分布式约束优化问题

在通常情况下，一个 C-DCOP 可以由一个五元组 $<A,X,D,F,\alpha>$ 定义，其中，$A=\{a_1,a_2,\cdots,a_n\}$ 是一组智能体的集合，能够控制一个或者多个变量；$X=\{x_1,x_2,\cdots,x_m\}$ 是一组由智能体控制的连续变量集合；$D=\{D_1,D_2,\cdots,D_m\}$ 是一组连续值域集合，每个变量 x_i 能够取得值域 $D_i=[L_i,U_i]$ 中的任意值，其中 L_i 和 U_i 分别表示值域的下界和上界；$F=\{f_1,f_2,\cdots,f_i\}$ 是一组约束代价函数的集合，其中每一个约束代价函数 $f_i\in F$ 被变量集合 X 的一组子集 $x_i=\{x_{i_1},x_{i_2},\cdots,x_{i_k}\}$ 所定义，意味着 $f_i: D_{i_1}\times D_{i_2}\times\cdots\times D_{i_k}\to R$，$f_i$ 受 k 个变量的约束。本文仅考虑二元约束，即 $k=2$；$\alpha: X\to A$ 是一个映射函数，表示将每个变量 $x_j\in X$ 的控制权分配给智能体 $a_i\in A$。为便于理解，假设一个智能体控制一个变量，因此智能体 a_i

与变量x_i可以认为是同一概念, 在本文中交替使用。

一个 C-DCOP 的解为一个赋值集合X^*, 使得全部约束代价函数之和最小:

$$X^* = \arg\min_{X} \sum_{i=1}^{l} f(x^i) \tag{1}$$

图 1 为一个 C-DCOP 的例子, 是一个具有 4 个变量的约束图, 每个变量由相应的一个智能体控制, 每条边代表一个约束代价函数。变量x_i的值域D_i为 $[-10,10]$。$\forall x_i \in X: D_i = [-10,10], f(x_1, x_2) = 2x_1^2 + 3x_1x_2 - x_2^2, f(x_1, x_3) = \exp\sqrt{x_1^2 + x_3^2}, f(x_1, x_4) = x_1^4 - 2x_1^2x_4 - 2x_4^3, f(x_2, x_3) = 2x_2^2 - \sin(\pi x_3)$。

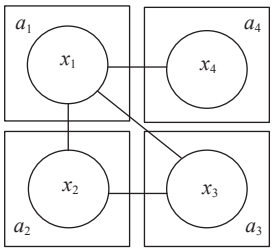

图 1　C-DCOP 的示例

Fig. 1　Example of C-DCOP

1.2　分布式种群

分布式种群是基于种群的 C-DCOP 求解算法中常用的一种方法, 每个智能体持有所有个体一个维度的变量, 所有智能体协同持有一个分布式种群, 一个个体代表一个解。图 2 表示一个具有K个染色体、n个智能体的分布式种群, 使用$C_k.x_n$表示第k个染色体的第n个维度的变量。

	Agent a_1	Agent a_2	Agent a_3	…	Agent a_n
染色体 1	$C_1.x_1$	$C_1.x_2$	$C_1.x_3$		$C_1.x_n$
染色体 2	$C_2.x_1$	$C_2.x_2$	$C_2.x_3$		$C_2.x_n$
⋮	⋮	⋮	⋮		⋮
染色体 k	$C_k.x_1$	$C_k.x_2$	$C_k.x_3$		$C_k.x_n$
⋮	⋮	⋮	⋮		⋮
染色体 K	$C_K.x_1$	$C_K.x_2$	$C_K.x_3$		$C_K.x_n$

图 2　分布式种群的示例

Fig. 2　Example of distributed population

1.3　广度优先搜索伪树

BFS 伪树是 DCOP 和 C-DCOP 中常用的一种通信结构, 其特点为多个分支并行计算, 通信路径和通信时间短。

图 3(a) 表示以图 1 为例的一个 BFS 伪树, 图 3(b) 表示 BFS 伪树的有序排列。BFS 伪树用于智能体之间的通信, 有序排列代表消息传递的顺序, 深度较小的智能体相比于深度较大的智能体有着更高的优先级, 使用H_i和L_i分别表示智能体a_i的高优先级邻居和低优先级邻居。在图 3(b) 中, 智能体a_1表示根智能体, 智能体a_3、a_4为叶智能体; 对于智能体a_2, 其邻居$N_2 = \{a_1, a_3\}$, 高优先级邻居$H_2 = \{a_1\}$, 低优先级邻居$L_2 = \{a_3\}$。

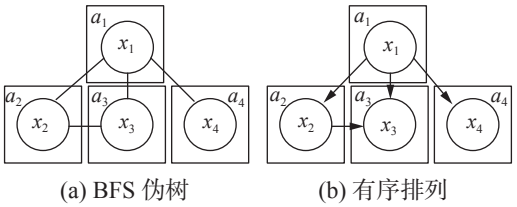

图 3　BFS 伪树构建和有序排列

Fig. 3　BFS pseudo-tree construction and ordered arrangement

2　基于连续型分布式约束优化问题的自适应多点交叉遗传算法

AMCGA 是一种基于种群的 C-DCOP 求解算法, 种群中的个体被称为染色体, 染色体的一个维度称为一个基因。AMCGA 包括 5 个阶段: 初始化、评估、选择、交叉和变异阶段。在初始化阶段, AMCGA 初始化种群和参数; 智能体在评估阶段协同计算染色体的适应度; 选择阶段通过贪婪策略选择精英染色体; 在交叉阶段, 智能体执行多点交叉算子; 在变异阶段, 智能体执行变异算子并保留精英染色体。具体流程见算法 1。

算法 1　ACMGA

1)构造有序的广度优先搜索(BFS)伪树;
2)初始化参数: $K, G, P_{c_1}, P_{c_2}, S_a, P_m$;
3)$C \leftarrow$生成 K个染色体 // 构造种群
4)for $a_i \leftarrow a_1$ to a_I do // I个智能体
5)　　for $C_k \leftarrow C_1$ to C_K do
6)　　$C_k.x_i \leftarrow D_i$
7)　　end for
8)　发送 $C.x_i$给集合L_i中的智能体
9)end for
10)未达到终止条件, for 每一个智能体a_i do
11)　　Messaging()
12)　　if a_i==root do // a_i为根智能体
13)　　Selection(C_{fitness})
14)　　end if
15)　　等待直到消息被接收
16)　　if 从集合 H_i中接收到消息 do
17)　　$C_g.x_i \leftarrow$ CrossList ∪ UncrossList
18)　　$C^{\text{cam}}.x_i \leftarrow$ CrossList
19)　　if agent a_i in Agent$_c$ do
20)　　Crossover($C^{\text{cam}}.x_i$)
21)　　end if
22)　　$C^{\text{cam}}.x_i \leftarrow$ CrossList ∪ UncrossList
23)　　计算 P_{mutation}

24）　　for $C_g^{\text{cam}}.x_i \leftarrow C_1^{\text{cam}}.x_i$ to $C_G^{\text{cam}}.x_i$ do
25）　　Mutation($C_g^{\text{cam}}.x_i$)
26）　　end for
27）　　$C.x_i \leftarrow C_g.x_i \cup C^{\text{cam}}.x_i$
28）　　if $L_i \neq 0$ do
29）　　发送 $C.x_i$给集合 L_i
30）　　发送信息给 L_i中的智能体
31）　　end if
32）　end if
33）end while

初始化阶段：AMCGA 首先构造有序的 BFS 伪树；然后初始化参数：K为染色体的数量；G为精英个体的数量；P_{c_1}, P_{c_2}为共同定义交叉概率；S_a为交叉点的数量；P_m为定义变异概率。

最后，智能体协同构造一个分布式种群并对所持有基因赋予一个值域中的随机值，然后将赋值发送给低优先级邻居 (算法 1：步骤 1）~9)。

以图 1 和图 3 为例，假设染色体的数量为 4，分布式种群可表示为$C=\{C_1,C_2,C_3,C_4\}$，每条染色体的完整赋值可表示为

$$C_1.X=\{x_1=1,x_2=2,x_3=2.5,x_4=3.1\}$$
$$C_2.X=\{x_1=2,x_2=4,x_3=0,x_4=5.2\}$$
$$C_3.X=\{x_1=3,x_2=-3,x_3=6,x_4=-2.5\}$$
$$C_4.X=\{x_1=-2,x_2=1.4,x_3=7,x_4=0\}$$

智能体a_1发送 $C.x_1=\{1, 2, 3, -2\}$给低优先级邻居a_2、a_3和a_4；智能体a_2向智能体a_3发送$C_2.X=\{x_1=2,x_2=4,x_3=-3,x_4=1.4\}$。

函数 1）Messaging()
① for $C.x_j$ received from $H_{i_j} \in H_i$ do // H_{i_j}表示高优先级邻居H_i中的一个邻居智能体
②　　for $C_k \leftarrow C_1$ to C_K do
③　　$C_{k.\text{fitness}} \leftarrow \text{Cost}_{i,j}(C_k.x_i, C_k.x_j)$
④　　end for
⑤　　send C_{fitness} fitness to agent in H_{i_j}
⑥ end for
⑦ Wait C_{fitness} received from all agents in L_i
⑧ if $|L_i| \neq 0$ and C_{fitness} received from L_i
⑨　　for $C_k \leftarrow C_1$ to C_K do
⑩　　$C_{k.\text{fitness}} \leftarrow \sum_{j \in L_i} C_{\text{fitness}}$
⑪　　end for
⑫　　if $a_i \neq$ root do // 智能体a_i不为根智能体
⑬　　send $C_{\text{fitness}}$to an agent $H_{i_j} \in H_i$
⑭　　end if
⑮ end if

函数 2）Selection(C_{fitness})
①$C^{\text{sort}} \leftarrow$ sorted(C_{fitness}) // 适应度排序
②$C_G^{\text{sort}} \leftarrow$Select the top G chromosomes // 选择适应度靠前的G条染色体
③ information：CrossList, UncrossList, $\text{Agent}_c \leftarrow$ {} // 创建 3 个集合
④ 计算 P_{cross}
⑤ for $C_g^{\text{sort}} \leftarrow C_1^{\text{sort}}$ to C_g^{cam} do
⑥　　if $r_p < P_{\text{cross}}$ do
⑦　　CrossList $\leftarrow$ CrossList $\cup C_g^{\text{sort}}$
⑧　　else
⑨　　UnCrossList $\leftarrow$ UnCrossList $\cup C_g^{\text{sort}}$
⑩　　end if
⑪ end for
⑫ if |CrossList|%2 ≠ 0 do // 数量为奇数
⑬　　CrossList $\leftarrow$ CrossList\end
//移除CrossList中最后一个元素
⑭　　UnCrossList $\leftarrow$ UnCrossList $\cup$ end
⑮ end if
⑯ $\text{Agent}_c \leftarrow$Select $S_a$different agents
⑰ 发送信息给集合 L_i中的智能体

函数 3）Crossover($C^{\text{cam}}.x_i$)
① len $\leftarrow |C^{\text{cam}}.x_i|$ // 变量的数量
② start $\leftarrow$ 1
③ while start ⩽ len/2 do
④　　$C_{\text{start}}^{\text{cam}}.x_i \leftrightarrow C_{\text{len-start+1}}^{\text{cam}}.x_i$
// 以中心为对称轴互换前后基因
⑤　　start + +
⑥ end while

函数 4）Mutation($C_g^{\text{cam}}.x_i$)
① if $r_m < P_{\text{mutation}}$do
②　　$C_g^{\text{cam}}.x_i \leftarrow$a random value from D_i
③ end if

评估阶段：当接收到高优先级邻居的赋值后，智能体依次计算与每个高优先级邻居的每条染色体的部分适应度，并发送给相应的高优先级邻居 (函数 1)：①~⑥)。除了叶智能体，每个智能体将从低优先级邻居接收的部分适应度求和，若该智能体不为根智能体，则将求和之后的部分适应度沿着 BFS 伪树发送给一个高优先级邻居 (函数 1)：⑧~⑮)。

$$C_{k.\text{fitness}} = \sum_{f_i \in F} f_i(C_k.x^i) \tag{2}$$

每条染色体的适应度通过式（2）计算，其中$C_k.x_i$表示 2 个具有约束关系的变量赋值。单个智能体不能计算染色体的完整适应度，而是在所有智能体的配合下协同计算染色体的完整适应度。

在上述例子中，智能体a_3分别计算与智能体a_1、a_2的部分适应度$C_{\text{fitness}}(x_1,x_3)$和$C_{\text{fitness}}(x_2,x_3)$并发

送给相应的智能体a_1、a_2。智能体a_4计算与a_1的部分适应度$C_{\text{fitness}}(x_1,x_4)$并发送给$a_1$。此外, 由于智能体$a_2$不是根智能体, 因此不仅计算与智能体$a_1$的部分适应度$C_{\text{fitness}}(x_1,x_2)$并发送给$a_1$, 还会将接收的部分适应度$C_{\text{fitness}}(x_2,x_3)$发送给$a_1$。计算的部分适应度为

$$C_{\text{fitness}}(x_1,x_2)=\{4,16,-18,-2.36\}$$
$$C_{\text{fitness}}(x_1,x_3)=\{14.77,7.39,819.10,1451.15\}$$
$$C_{\text{fitness}}(x_1,x_4)=\{54.38,255.62,94.75,16\}$$
$$C_{\text{fitness}}(x_2,x_3)=\{7,32,18,3.92\}$$

选择阶段: 由于每个智能体都会将部分适应度发送给高优先级邻居, 因此染色体的所有部分适应度都将传递到根智能体。根智能体首先对染色体的适应度排序并选择适应度靠前的G条染色体 (函数 2): ①~②); 然后对每一条染色体进行判断, 若 [0, 1] 之间的随机数r_p小于通过$P_{\text{cross}}=P_{c_1}+\dfrac{P_{c_2}\times(I_{\max}-I_{\text{cur}})}{I_{\max}}$计算的交叉概率$P_{\text{cross}}$, 则将该染色体标记为交叉染色体CrossList, 否则标记为非交叉染色体UnCrossList, 若交叉染色体为奇数, 则将交叉染色体中最后一个染色体移除并添加到非交叉染色体中 (函数 2): ④~⑮); 最后, 根智能体随机选择S_a个不同的智能体作为交叉点并将信息发送给低优先级邻居中的智能体 (函数 2): ⑯~⑰)。$I_{\max}$和I_{cur}分别为最大迭代次数和当前迭代次数。

在例子中, 假设$G=2$并且智能体a_2、a_3被选择为交叉点。智能体a_1获得染色体的完整适应度$C_{\text{fitness}}=\{80.15,311.03,913.85,1468.71\}$并在排序和选择之后得到$C_2^{\text{sort}}=\{C_1,C_2\}$。最后, a_1将信息 = {CrossList, UncrossList, Agent$_c$}发送给低优先级邻居a_2、a_3、a_4。

交叉阶段: 当收到高优先级邻居传递的 information 后, 智能体首先复制交叉染色体和非交叉染色体作为父代; 若智能体被标记为交叉点, 则使用交叉染色体执行交叉算子 (算法 1: 17)~21))。具体来说, 智能体将自身持有染色体一个维度的基因前后互换位置 (函数 3): ①~⑥)。

在之前的例子中, C_1和C_2执行交叉算子, 智能体a_2交换$C_1.x_2$和$C_2.x_2$, 智能体a_3交换$C_1.x_3$和$C_2.x_3$, 如图 4 所示。C_{1*}和C_{2*}为子代染色体:

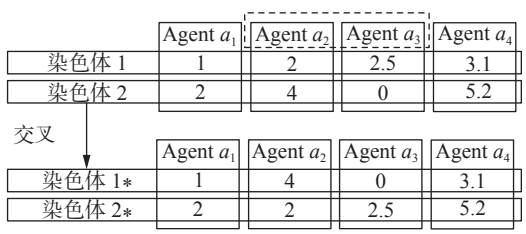

图 4　智能体 a_2 和 a_3 的交叉示例

Fig. 4　Agent a_2 and a_3 as an example of crossover

$$C_{1*}.X=\{x_1=1,x_2=4,x_3=0,x_4=3.1\}$$
$$C_{2*}.X=\{x_1=2,x_2=2,x_3=2.5,x_4=5.2\}$$

变异阶段: 在交叉完成之后, 子代染色体与非交叉染色体合并, 通过$P_{\text{mutation}}=P_m\times\dfrac{I_{\max}-I_{\text{cur}}}{I_{\max}}$计算变异概率并对每一条染色体执行变异算子 (算法 1: 22)~26))。具体来说, 在 [0, 1] 之间的随机数r_m若小于变异概率, 则智能体a_i将此染色体的此维度基因重新赋值为值域D_i里的一个随机值 (函数 4): ①~③)。最后, 智能体将子代染色体和父代染色体合并为一个新的种群 (算法 1: 37))。

假设只有智能体a_4所持有的基因发生变异, C_{1*}和C_{2*}变异后的完整赋值为

$$C_{1*}.X=\{x_1=1,x_2=4,x_3=0,x_4=5\}$$
$$C_{2*}.X=\{x_1=2,x_2=2,x_3=2.5,x_4=-9\}$$

因此, 新种群的完整赋值为

$$C_1.X=\{x_1=1,x_2=4,x_3=0,x_4=5\}$$
$$C_2.X=\{x_1=2,x_2=2,x_3=2.5,x_4=-9\}$$
$$C_3.X=\{x_1=1,x_2=2,x_3=2.5,x_4=3.1\}$$
$$C_4.X=\{x_1=2,x_2=4,x_3=0,x_4=5.2\}$$

3　算法收敛性分析

本节使用通信步骤表示智能体接收邻居智能体发送的消息次数。此外, 在本节使用 H 代表 BFS 伪树的深度。

引理 1　根智能体在通信步骤为 $T+H$ 时获得通信步骤为 T 时的最优解。

证明: 智能体通过 BFS 伪树传递染色体的部分适应度, 根智能体计算染色体的完整适应度需要叶智能体沿着 BFS 伪树向上传递染色体的部分适应度。由于 BFS 伪树的高度为 H, 根智能体需要等待最多 H 次通信步骤来计算染色体的完整适应度, 从而找到最优解。因此, 根智能体在通信步骤为 $T+H$ 时获得通信步骤为 T 时的最优解。

引理 2　所有智能体在通信步骤为 $T+2H$ 时获得通信步骤为 T 时的最优解。

证明　通过引理 1, 根智能体在 $T+H$ 时获得 T 时的最优解, 最优解通过 BFS 伪树到达每一个智能体需要等待最多 H 次通信步骤。因此, 所有智能体在通信步骤为 $T+2H$ 时获得通信步骤为 T 时的最优解。

命题 1　AMCGA 是一种任意时间属性算法。

证明　从引理 2 可得, 所有智能体在通信步骤为 $T+2H+\Delta$ ($\Delta>0$) 时获得通信步骤为 $T+\Delta$ 时的最优解。只有智能体找到了更好的解, 最优解

才会更新，通信步骤为 $T+2H+\Delta$ 的最优解不会差于通信步骤为 $T+2H$ 时最优解。因此，AMCGA 中解的质量不会随着时间增加而降低并且在任意时刻都能提供一个最优解，即 AMCGA 是一种任意时间属性算法。

命题 2　AMCGA 具有全局收敛性。

证明　设 $b(m)$ 为第 m 代种群的最优解，若

$$b_{\text{fitness}}(m) < b_{\text{fitness}}(m+1)$$

则

$$b(m+1) = b(m)$$

$$b_{\text{fitness}}(m+1) = b_{\text{fitness}}(m)$$

从命题 1 可知，每一代种群产生的最优解组成的序列是一个非递增序列。在 AMCGA 中，由于精英个体保留策略，最优解以概率 1 保留到下一代种群，若在第 M 代种群找到了全局最优解 B，则 B 将一直保留而不会丢失。

若 $m > M$, $b_{\text{fitness}}(m) = B_{\text{fitness}}$

则

$$\lim_{m\to\infty} P(b_{\text{fitness}}(m) = B_{\text{fitness}}) = 1$$

因此，AMCGA 具有全局收敛性。

4　复杂度分析

本节定义染色体数量为 K，智能体的数量为 n，高优先级邻居的数量为 H，低优先级邻居的数量为 L，因此邻居数量为 $N = H + L$。此外，图的类型定义为全连接图，即 $N \approx n$。一次迭代定义为经过完整的评估、选择、交叉和变异阶段。

在 AMCGA 中，一个智能体在初始化阶段和选择阶段向低优先级邻居发送消息；在评估阶段，智能体向高优先级邻居发送消息，此外，非叶智能体还会传递一次消息（部分适应度）。因此，在最坏情况，即所有邻居为低优先级邻居，一个智能体在一次迭代中发送的消息数量为 $O(2L+H+1) = O(N+L+1) = O(n)$。

智能体会发送 5 类消息：$C.x_i$、C_{fitness}、ϕ、φ、γ。其中第 1 类和第 2 类消息分别包含 K 条染色体的信息；第 3 类和第 4 类共包含 G 条染色体的信息；最后一类消息远小于 K。因此，一个智能体在一次迭代中发送的消息大小为 $O(2Kn+Gn) = O(Kn)$。

在一次迭代中，智能体计算每条染色体的适应度。此外还会分别计算一次交叉、变异概率。因此一个智能体在一次迭代中的计算复杂度为 $O(Kn+1+1) = O(Kn)$。

5　实验与分析

为了验证 ACMGA 的有效性，本文在 4 类基准问题上使用 AMCGA 与最先进的 C-DCOP 求解算法 C-CoCoA、PFD 和 HCMS 进行实验对比。虽然 AMCGA 适用于任何函数形式，为了更好地展示对比效果，本文遵循文献 [18] 中的约束代价函数形式：$ax^2+bx+cxy+dy+ey^2+f$。其中 a、b、c、d、e、f 均为区间 [−5, 5] 中的随机数。智能体 a_i 控制的变量值域 D_i 设置为 [−50, 50]，迭代次数设置为 500。对于每一种实验配置，每一种算法独立运行 30 次并取平均值作为最终的结果。

5.1　基准问题

随机图　稀疏随机图（密度为 0.1）和稠密随机图（密度为 0.6）。密度表示在随机图的构造过程中任意 2 个点的连接概率。智能体的数量为 10~100，数量间隔为 10。

无尺度网络　无尺度网络由 Barabási-Albert 模型[24] 生成。首先生成一个由 10 个智能体组成的连通图，在每一次迭代中加入一个新的智能体与当前网络中的 7 个智能体相连，直到所有智能体加入到该网络，一个智能体被连接的概率与该智能体的度数成正比。设置智能体的数量为 60~100，数量间隔为 5。

随机树　根据文献 [18] 将随机树作为一个基准问题并设置智能体的数量为 60~100，数量间隔为 5。

小世界网络　小世界网络由 Watts-Strogatz 拓扑模型[25] 生成，在多个节点相连的环中，每个节点与最邻近的 J 个节点相连，顺时针选择一个节点和与其相连的一条边随机与环上的一个节点以概率 P 重连，规定其中不能有重边和自环。设置 J 为 6，P 为 0.5，智能体数量为 60~100，数量间隔为 5。

5.2　参数讨论

AMCGA 的性能取决参数的选择，在本节对交叉点的数量、交叉概率和变异概率进行讨论并选择合适的参数。选择 100 个智能体的稠密随机图来进行实验。根据经验，AMCGA 的初始系数设置为 $K = 10n$，$G = 5n$，$P_{c_1} = 0.9$，$P_{c_2} = 0.05$，$S_a = 0.1n$ 和 $P_m = 0.01$，n 表示智能体的数量。

5.2.1　交叉点数量

设置 8 种不同的交叉点数量以对比参数 S_a 对 AMCGA 性能的影响：$S_a = 0.01n$, $0.02n$, $0.05n$, $0.1n$, $0.2n$, $0.3n$, $0.4n$ 和 $0.5n$。从图 5 可见，AMCGA 的求解质量随着交叉点的增加变得更好。由于 $S_a = 0.3n$ 在较快的收敛速度下能够获得更高质量的解，因此选择该参数设置进行接下来的实验。

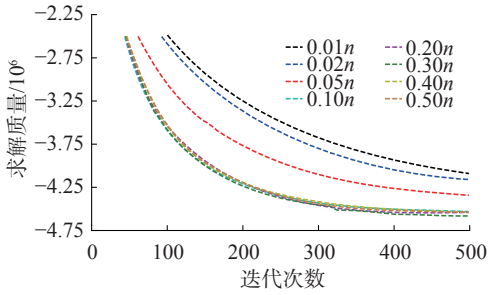

图 5　AMCGA 在不同交叉点数量下的收敛曲线

Fig. 5　Convergence curves of AMCGA with different number of crossover points

5.2.2　交叉概率

为了对比不同交叉概率对 AMCGA 性能的影响, 设置 6 种不同的交叉概率进行试验: $P_{c_1}=0.9$, 0.8, 0.7, 0.6, 0.5, 0.4。在图 6 中, 交叉概率越大, 算法的收敛速度越快, 表示个体之间信息交互越多。由于交叉概率为 0.9 的算法有着更快的收敛速度并且能够得到一个更高质量的解, 因此在接下来的实验中设置交叉概率为 0.9。

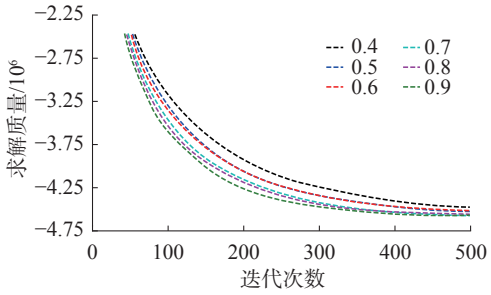

图 6　AMCGA 在不同交叉概率下的收敛曲线

Fig. 6　Convergence curves of AMCGA with different crossover probabilities

5.2.3　变异概率

图 7 表示 AMCGA 在不同变异概率下的求解质量。变异概率影响算法的局部搜索, 变异概率越大, 局部搜索的概率越大, 但算法退化为随机搜索。由于 AMCGA 在变异概率为 0.02 时能够得到更高质量的解, 因此在接下来的实验中设置变异概率为 0.02。

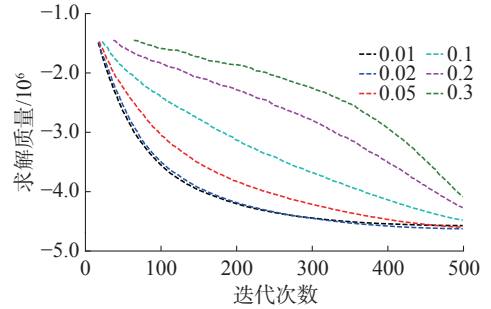

图 7　AMCGA 在不同变异概率下的收敛曲线

Fig. 7　Convergence curves of AMCGA with different mutation probabilities

5.3　算法对比实验

通过 5.2 节的参数讨论, 设置 AMCGA 的参数为: $K=10n$, $G=5n$, $P_{c_1}=0.9$, $P_{c_2}=0.05$, $S_a=0.3n$, $P_m=0.02$。对于 AMCGA 的竞争算法 HCMS、PFD 和 C-CoCoA 的参数设置, 通过文献 [18] 设置 HCMS 中离散点的个数为 3, 梯度下降的步长为 0.001; 遵循文献 [19], 设置 PFD 中的 $K=2000$, $w=0.9$, $c_1=0.9$, $c_1=0.1$, $m_{sc}=15$, $m_{fc}=5$; 最后, 使用文献 [20] 中 C-CoCoA 的参数设置: 离散点的个数为 3, 梯度下降的次数为 100, 步长为 0.01。

图 8、9 分别给出了 AMCGA、HCMS、PFD 和 C-CoCoA 在稀疏和稠密随机图上的求解质量。可以看出 AMCGA 在稀疏和稠密配置下均优于其他竞争算法, 随着智能体数量的增加, AMCGA 的优势程度更加明显。

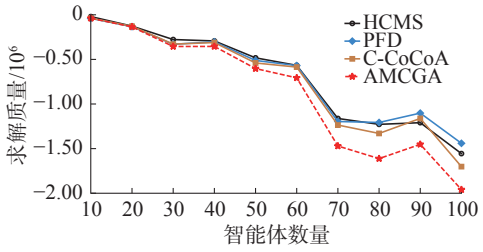

图 8　AMCGA 与竞争算法在稀疏随机图上的求解质量

Fig. 8　Solution quality of AMCGA and competing algorithms on sparse random graphs

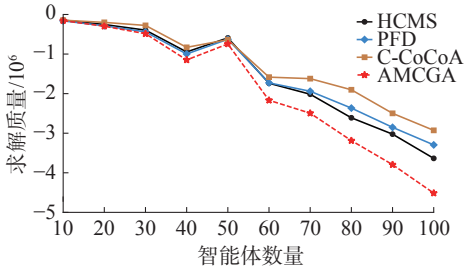

图 9　AMCGA 与竞争算法在稠密随机图上的求解质量

Fig. 9　Solution quality of AMCGA and competing algorithms on dense random graphs

图 10 表示 4 种算法在无尺度网络上的求解质量, 可以看出 AMCGA 在每一种智能体数量设置下都明显优于竞争算法。

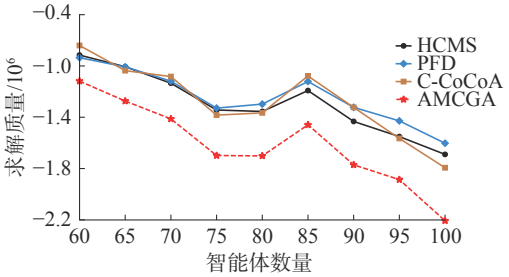

图 10　AMCGA 与竞争算法在无尺度网络上的求解质量

Fig. 10　Solution quality of AMCGA and competing algorithms on scale-free networks

在图 11 中，AMCGA 的求解质量优于竞争算法。但易看出 C-CoCoA 的求解质量在一些智能体数量设置下接近于 AMCGA，这是由于在随机树的拓扑结构中，节点之间的连接较少并且约束关系简单，C-CoCoA 中半贪婪的局部搜索策略更容易在简单的约束图中找到较高质量的解。

从图 12 中可以看出 4 种算法在小世界网络上不同智能体数量设置下的求解质量。与无尺度网络上的对比实验相似，AMCGA 的求解质量优于竞争算法。

图 13 进一步给出 AMCGA、HCMS、PFD 和 C-CoCoA 在 100 个智能体的稠密随机图上的收敛曲线。值得一提的是，本文同时运行 C-CoCoA 和 AMCGA，在 C-CoCoA 运行完成时，记录 AMCGA 的迭代次数作为 C-CoCoA 的迭代次数。此外，由于 HCMS 无法保证收敛，因此使用 HCMS 的历史最优取值作为结果并显示在图中。

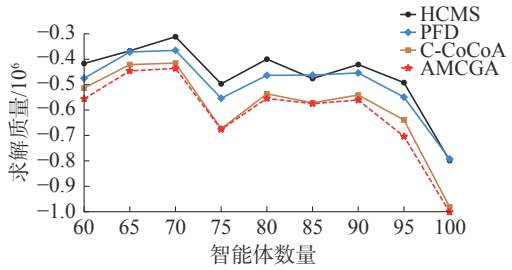

图 11　AMCGA 与竞争算法在随机树上的求解质量

Fig. 11　Solution quality of AMCGA and competing algorithms on random trees

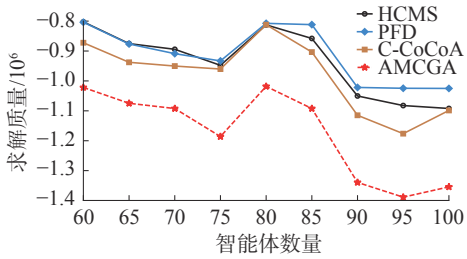

图 12　AMCGA 与竞争算法在小世界网络上的求解质量

Fig. 12　Solution quality of AMCGA and competing algorithms on small-world networks

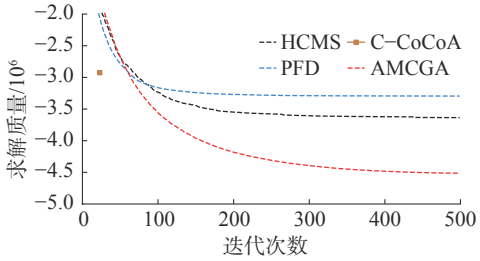

图 13　AMCGA 与竞争算法在稠密随机图上的收敛曲线

Fig. 13　Convergence curves of AMCGA and competing algorithms on dense random graphs

从图 13 中可以看出，C-CoCoA 是第一个完成求解的算法，其他的算法则继续优化。在迭代大约 80 次之后，AMCGA 每一次迭代结果都优于 HCMS 和 PFD。随着迭代次数增加，AMCGA 的优势程度变大并且其最终的求解质量优于竞争算法。

表 1 给出了 AMCGA 相比于 3 种竞争算法在不同类型问题上的提升率。其中提升率为不同基准问题下所有智能体数量设置下提升率的平均值，结果保留 2 位小数。

表 1　AMCGA 相比于竞争算法在基准问题上的提升率

Table 1　Improvement rates of AMCGA compared to competing algorithms on benchmark problems　%

算法	HCMS	PFD	C-CoCoA
稀疏随机图	30.43	18.99	14.30
稠密随机图	21.74	20.38	45.21
无尺度网络	24.85	28.81	27.40
随机树	34.42	22.26	4.36
小世界网络	25.54	28.50	19.78

AMCGA 优于其他竞争算法，对不同大小和密度的基准问题都有着较好的提升率。对比 HCMS 和 PFD，AMCGA 在任意问题配置下都有着较高的提升率。尽管 C-CoCoA 在随机树上能够表现优异的求解性能，但 AMCGA 仍然存在 4.36% 的提升率。

6　结束语

针对现有 C-DCOP 算法的缺点和局限性，提出一种求解 C-DCOP 的自适应多点交叉遗传算法。AMCGA 通过自适应的交叉、变异概率平衡算法的探索与开发，自适应的多点交叉算子增强种群的搜索能力，智能体之间的协同调度保证了分布式环境中解的一致性。本文证明了 AMCGA 是一种 anytime 算法并具有全局收敛性。在 4 类基准问题上的广泛实验表明 AMCGA 的求解质量优于 HCMS、PFD 和 C-CoCoA。尽管 AMCGA 相对于 HCMS 和 PFD 的求解质量有着较高的提升，但收敛速度略微落后于 HCMS 和 PFD，因此未来的工作可以概括为 2 点：1）结合一些策略来提高 AMCGA 的收敛速度。2）将 AMCGA 扩展到多目标 C-DCOP。

参考文献:

[1] 段沛博, 张长胜, 张斌. 分布式约束优化方法研究进展 [J]. 软件学报, 2016, 27(2): 264–279.
ALLAN R L, FABRICIO E, JEAN P A. Distributed constraint optimization problems: review and perspectives[J]. Expert systems with applications, 2016, 27(2): 264–279.

[2] 邓衍晨. 求解分布式约束优化问题的推理算法研究 [D]. 重庆: 重庆大学, 2018.
DENG Yanchen. The study on inference-based algorithms for distributed constraint optimization problems[D]. Chongqing: Chongqing University, 2018.

[3] SULTANIK E A, MODI P J, REGLI W C. On modeling multiagent task scheduling as a distributed constraint optimization problem[C]//Proceedings of the 20th International Joint Conference on Artifical Intelligence. New York: ACM, 2007: 1531−1536.

[4] 马瑞, 金艳, 刘鸣春. 基于机会约束规划的主动配电网分布式风光双层优化配置 [J]. 电工技术学报, 2016, 31(3): 145–154.
MA Rui, JIN Yan, LIU Mingchun. Bi-level optimal configuration of distributed wind and photovoltaic generations in active distribution network based on chance constrained programming[J]. Transactions of China electrotechnical society, 2016, 31(3): 145–154.

[5] 吴玲, 张朱峰, 吴威. 基于分布式约束优化的多 UCAV 协同任务分配 [J]. 海军工程大学学报, 2018, 30(6): 64–68.
BARTHS A, ENEMBRECK F. Distributed constraint optimization with MULBS: a case study on collaborative meeting scheduling[J]. Journal of network and computer applications, 2018, 30(6): 64–68.

[6] FERDINANDO F, ENRICO P, WILLIAM Y. A multiagent system approach to scheduling devices in smart homes[C]// Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems. Paulo: AAMAS Press, 2017: 981–989.

[7] MULDOON C, O'HARE G M P, O'GRADY M J, et al. Distributed constraint optimization for resource limited sensor networks[J]. Science of computer programming, 2013, 78(5): 583–593.

[8] 雷兴明, 邢昌风, 吴玲. 基于分布式约束优化的武器目标分配问题研究 [J]. 计算机工程, 2012, 38(7): 128–130.
CHENG Shanjun, RAJA A, XIE Jiang. Dynamic multiagent load balancing using distributed constraint optimization techniques[J]. Web intelligence and agent systems, 2012, 38(7): 128–130.

[9] JMODI P J, SHEN W M, TAMBE M. Adopt: asynchronous distributed constraint optimization with quality guarantees[J]. Artificial intelligence, 2005, 161(1/2): 149–180.

[10] PETCU A, FALTINGS B. A scalable method for multiagent constraint optimization[C]//Proceedings of the 19th International Joint Conference on Artificial Intelligence. New York: ACM, 2005: 266−271.

[11] LITOV O, MEISELS A. Forward bounding on pseudotrees for DCOPs and ADCOPs[J]. Artificial intelligence, 2017, 252: 83–99.

[12] ZHANG Weixiong, WANG Guandong, ZHAO Xing, et al. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks[J]. Artificial intelligence, 2005, 161(1/2): 55–87.

[13] FARINELLI A, ROGERS A, PETCU A, et al. Decentralised coordination of low-power embedded devices using the max-sum algorithm[C]// Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems. Estoril: AAMAS Press, 2008: 639–646.

[14] MAHESWARAN R T, PEARCE J P, TAMBE M. A family of graphical-game-based algorithms for distributed constraint optimization problems[J]. Coordination of large-scale multiagent systems, 2006: 127–146.

[15] CHEN Ziyu, WU Tengfei, DENG Yanchen, et al. An ant-based algorithm to solve distributed constraint optimization problems[C]//Proceedings of the 32th AAAI Conference on Artificial Intelligence. New Orleans: AAAI Press, 2018:4653–4661.

[16] STRANDERS R, FARINELLI A, ROGERS A, et al. Decentralised coordination of continuously valued control parameters using the max-sum algorithm[C]// Proceedings of the 8th International Conference on Autonomous Agents and Multi-Agent Systems. Budapest: Springer Press, 2009: 601−608.

[17] VOICE T, STRANDERS R, ROGERS A, et al. A hybrid continuous max-sum algorithm for decentralised coordination[C]// Proceedings of the 19th European Conference on Artificial Intelligence. Lison: IOS Press, 2010: 61−66.

[18] KHOI D H, WILLIAM Y, MAKOTO Y, et al. New algorithms for continuous distributed constraint optimization problems[C]// Proceedings of the 19th International

Conference on Autonomous Agents and Multi-Agent Systems. Auckland: Springer Press, 2020: 502−510.

[19] CHOUDHURY M, MAHMUD S, KHAN M M. A particle swarm based algorithm for functional distributed constraint optimization problems[C]// Proceedings of the 34th AAAI Conference on Artificial Intelligence. New York: AAAI Press, 2020:7111−7118.

[20] AMIT S, MOUMITA C, MD M K. A local search based approach to solve continuous DCOPs[C]// Proceedings of the 20th International Conference on Autonomous Agents and Multi-Agent Systems. Richland: Springer Press, 2021: 1127−1135.

[21] EIBEN A E, RAUE P E, RUTTKAY Z. Solving constraint satisfaction problems using genetic algorit-hms[C]// Proceedings of the First IEEE Conference on Evolutionary Computation. Orlando: IEEE Press, 1994:542−547.

[22] CHEN Ziyu, LIU Lizhen, HE Jingyuan, et al. A genetic algorithm based framework for local search algorithms for distributed constraint optimization problems[J]. Autonomous agents and multi-agent systems, 2020, 34(2): 41.

[23] CHEN Ziyu, HE Zhen, HE Chen. An improved DPOP algorithm based on breadth first search pseudo-tree for distributed constraint optimization[J]. Applied intelligence, 2017, 47(3): 607–623.

[24] ALBERT R, BARABASI A L. Statistical mechanics of complex networks[EB/OL]. (2001-06-09)[2022-02-22]. https://arxiv.org/abs/cond-mat/0106096.

[25] WATTS D J, STROGATZ S H. Collective dynamics of ‘small-world’ networks[J]. Nature, 1998, 393(6684): 440–442.

作者简介：

廖鑫，硕士研究生，主要研究方向为计算智能与分布式约束优化。

石美凤，讲师，主要研究方向为计算智能、多智能体系统、分布式约束优化、多目标优化和图像处理。主持重庆市科委基础研究与前沿探索项目、重庆市教委科学技术研究计划青年项目、教育部产学合作协同育人项目等项目 3 项，授权发明专利 2 项。发表学术论文 10 余篇。

陈媛，教授，主要研究方向为智能信息处理、数据挖掘 。主持和参与重庆市科委攻关项目、重庆市科委自然科学基金一般项目等 4 项。发表学术论文 20 余篇，出版著作 5 部。